facultad de informática.valencia

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROYECTO FINAL DE CARRERA:

eRestaurante: Prototipo de un Restaurante Digital

ALUMNA: Silvia Puchol Herrero
DIRECTOR: Joan Fons i Cors
CURSO: 2009/2010

ÍNDICE

1. INTRODUCCIÓN

Para empezar se explicará la finalidad, en qué consiste y la estructura de elaboración de este proyecto.

1.1 OBJETIVOS Y RESUMEN DEL PROYECTO

El objetivo del proyecto, a grandes rasgos, es la implementación de una aplicación web, que gestione un restaurante desde el punto de vista de los usuarios gerente, camarero y cliente.

No obstante, el objetivo más concreto es llevar a cabo la implementación utilizando un concepto innovador. Este concepto consiste en que el cliente tenga acceso a dicha aplicación desde su propia mesa y pueda seleccionar los productos que desee directamente desde ésta, sin tener que necesitar al camarero para realizar su pedido.

A todo esto debemos añadir que la aplicación debe permitir interactuar con ella a los usuarios de una manera ágil y sencilla, dentro de las posibilidades de su funcionalidad.

A su vez, otro de los objetivos, éste desde el punto de vista del desarrollador, es adquirir experiencia en la implementación de este tipo de sitios web, que cada vez están más extendidos en la sociedad.

1.2 PROBLEMA PLANTEADO

Desarrollar un sitio web para un restaurante que permita:

- A los gerentes gestionar el catálogo de productos y los empleados del restaurante.

- A los camareros gestionar y atender los pedidos, obteniendo en tiempo real los pedidos de los clientes en sus terminales.

- A los clientes realizar pedidos desde su mesa desde el primer momento en que llegan al restaurante, visualizar en qué estado se encuentran en tiempo real y realizar reservas cuando se conecten a la aplicación desde fuera del restaurante.

- A cualquier usuario que se conecte a la aplicación conocer el restaurante consultando información diversa acerca del mismo y registrarse en el sistema.

1.3 ESTRUCTURA DEL PROYECTO

En este apartado describiremos el proceso de elaboración completa de un proyecto de una aplicación web. En él se tratarán distintos puntos, como la especificación de requisitos; aquí trataremos de describir cómo será la aplicación final que obtendremos, las funciones que realizará, los tipos de usuarios que la utilizarán, etc. Para ello hemos seguido una guía, recomendada por la mayor parte de expertos en este campo, la especificación IEEE Standard 830 1998.

La siguiente parte será la de análisis, en la cual se creará el diagrama de clases que ayudará a comprender la estructura de la aplicación.

La sección de diseño vendrá a continuación, donde se estudiará el diseño por capas que vamos a utilizar (tres capas: de presentación, de lógica de la aplicación y de persistencia), y describiremos la interfaz gráfica de la aplicación, la base de datos, etc.

En el siguiente capítulo, implementación e integración, explicaremos las tecnologías utilizadas para el desarrollo, así como las herramientas usadas. Además se explicará todo el desarrollo realizado hasta obtener el producto final.

Para finalizar, se presentan una serie de conclusiones que se han obtenido durante la creación del sitio web y se hace referencia a la bibliografía utilizada.

2. ESPECIFICACIÓN DE REQUISITOS

2.1 INTRODUCCIÓN

2.1.1 Propósito

Este sistema permitirá a los clientes realizar reservas y pedidos sobre la carta, y a los camareros conocer el estado de cada mesa y saber los platos a servir en cada momento.

El principal objetivo de este software es permitir a los clientes de un restaurante obtener una mejor atención y a los camareros facilitarles y organizarles el trabajo.

2.1.2 Ámbito

El producto software a desarrollar se denominará “eRestaurante”.

2.1.3 Definiciones, acrónimos y abreviaturas

- **ERS.** Documento de Especificación de Requisitos Software.

- **Internet.** Red de redes a escala mundial con millones de computadores interconectados entre ellos mediante el conjunto de protocolos TCP/IP.
  También se utiliza este nombre para designar cualquier red de redes que utilice las mismas tecnologías que Internet.

- **Web.** El web o WWW (acrónimo en inglés de World Wide Web, gran telaraña mundial) es una red de páginas escritas en hipertexto, con el lenguaje de marcado HTML, y conectadas entre sí. Para acceder la única herramienta indispensable es un navegador web.

- **Software.** Programas, aplicaciones.

- **Aplicación Web.** Aplicación que los usuarios utilizan desde un servidor web a través de Internet o una intranet. La facilidad para actualizar y mantener aplicaciones sin la necesidad de instalar programas en los millones de clientes potenciales es una de las principales causas de su popularidad.

- **Usuario.** Persona que después de haberse identificado hace uso de las funciones de la aplicación.

- **Sistema.** Conjunto de partes interrelacionadas, hardware, software y de recurso humano.

- **Apache.** Es un software libre, servidor HTTP de código abierto que implementa el protocolo HTTP/1.1.

- **Código abierto.** Término usado para referirse a programas que se ofrecen con total libertad de modificación, uso y distribución bajo la regla implícita de no modificar dichas libertades hacia el futuro.

- **Protocolo.** Conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre sistemas.

- **Servidor web.** Es un programa que se ejecuta continuamente en un ordenador manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

- **MySQL.** Sistema de gestión de base de datos.

2.1.4 Referencias

Guía del IEEE std. 830 1998 para la especificación de requisitos del Software.

2.1.5 Visión global

El producto a desarrollar es un sitio web, llamado "*eRestaurante*", orientado a la realización de pedidos en un restaurante mediante un terminal, de forma que los camareros tengan un control individualizado, a través de la aplicación, de lo que se ha pedido y servido en cada mesa.

El objetivo es facilitar a los clientes la realización de sus pedidos y a los camareros la atención de los mismos. Entre las ventajas de la implantación de este sistema destacan entre otras que los clientes no deberán sufrir esperas para realizar su pedido (en cuanto lleguen podrán comenzar a pedir y cuando confirmen el pedido podrán pagar con la tarjeta si lo desean), lo que agilizará el servicio y permitirá conseguir una mayor satisfacción del cliente.

2.2 DESCRIPCIÓN GENERAL

2.2.1 Perspectiva del producto

El producto software no depende de ningún sistema mayor, es independiente.

2.2.2 Funciones del producto

Podemos clasificarlas en dos partes diferenciadas:

Funciones de visualización:

Nuestra aplicación visualizará información relacionada con el restaurante, los clientes y los pedidos.

a) Función de gestión de la información

- Consulta de los productos existentes y sus precios.
- Consulta de los menús de oferta.
- Envío de comentarios y sugerencias.
- Recordatorio de pedidos anteriores realizados por el cliente
- Visualizar los pedidos (camareros). Los camareros podrán ver en todo momento la descripción de los pedidos realizados de sus mesas, así como el estado en el que se encuentran: servido o sin servir.
- Visualización de pedidos (clientes). El cliente podrá acceder a un listado con los pedidos realizados, los productos que componen cada pedido y el estado (servido o sin servir) de cada producto del mismo.

Funciones de mantenimiento/actualización de la base de datos

a) Función de control de usuarios

- Registro e identificación de usuarios
- Asignar clientes a una mesa. Los camareros serán los encargados de comprobar qué mesas hay libres en el restaurante y asignar a los clientes a una mesa libre.

b) Función de gestión de los productos

- Añadir productos. Se podrán añadir productos al restaurante.
- Gestionar productos. Se podrán modificar y eliminar productos existentes.

c) Función de gestión de ofertas

- Crear ofertas. El administrador podrá crear nuevos menús de oferta compuestos por los productos que elija disponibles en el restaurante.
- Gestionar las ofertas existentes. Se podrán modificar el precio de las ofertas así como los productos que las componen. También se podrán eliminar.

d) Función de gestión de pedidos

- Servir pedidos. Los camareros podrán marcar un producto como servido.

e) Función de pedidos

- Realización de pedidos. El cliente podrá realizar pedidos, para ello seleccionará los productos que le interesen y lo confirmará cuando lo desee.
- Modificación de pedidos. El cliente podrá modificar los productos del pedido mientras no lo haya confirmado. En caso de ya haber confirmado sería el camarero el que tendría que modificarle el pedido.

2.2.3 Características del usuario

Tipos de usuario:

- **Clientes:**
  - **Identificados.** Son los usuarios que disponen de una vista personalizada. Podrán realizar y modificar pedidos, consultar su historial de los mismos y realizar reservas de mesas vía web.

  - **Invitados.** Son los usuarios que sólo pueden consultar el catálogo de productos y precios del restaurante, así como información general del mismo.

- **Personal del restaurante:**
  - **Gerente.** Podrá modificar precios, añadir y eliminar tipos de producto y productos, así como gestionar empleados y crear, modificar y eliminar menús de oferta.

  - **Camareros.** Son los encargados de asignar mesas a los clientes y atender los pedidos. Cada producto que sirvan lo deberán marcar como “servido”. También podrán excepcionalmente modificar

pedidos de los clientes en caso de que estos hayan confirmado algún pedido por error o algún incidente similar.

La intención es que cualquier usuario pueda utilizar esta aplicación software, por ello, es fundamental que sea usable y sencilla. Una pérdida de sencillez podría provocar pérdida de clientes por la incomodidad y/o dificultad para realizar sus pedidos.

2.2.4 Restricciones generales

Cada usuario sólo podrá realizar las funciones propias del mismo.

2.3 REQUISITOS ESPECÍFICOS

2.3.1 Requisitos de interfaces externas

2.3.1.1 Interfaces de usuario

Invitado. Tendrá acceso a consultar información general y productos del restaurante así como enviar comentarios.

Cliente. Tendrá el mismo acceso que el usuario invitado y además podrá realizar reservas de mesas vía web, consultar sus pedidos anteriores, realizar y modificar pedidos en el restaurante.

Camarero. Podrá ver las mesas que hay libres, asignarlas a clientes y ver el estado de los pedidos de todas las mesas, modificar pedidos y marcarlos como “servidos”.

Gerente. Podrá realizar tareas de gestión: añadir, modificar y eliminar tipos de producto, productos, menús y empleados.

2.3.1.2 Interfaces hardware

Para poder utilizar la aplicación, el usuario necesitará un dispositivo que tenga instalado un navegador web; no tendrá importancia la plataforma utilizada.

También será necesario una conexión a Internet, ya sea mediante un módem de marcado, tarjeta Ethernet, inalámbrica, etc.

2.3.1.3 Interfaces software

Como se ha señalado en el punto anterior, en el caso del cliente solamente se necesita un navegador web. Por tanto, el sistema operativo utilizado será cualquiera sobre el que corra un navegador.

En el lado del servidor, será necesario tener instalado un servidor web, en nuestro caso *Apache* y un servidor de bases de datos, *MySQL*.

2.3.1.4 Interfaces de comunicaciones

Los usuarios tendrán que acceder a Internet para poder utilizar la aplicación. Para ello tendrán instalado el protocolo TCP/IP, y también el protocolo HTTP, que funciona por encima del TCP y sirve para poder realizar las conexiones.

2.3.2 Requisitos específicos

Esta sección está organizada por tipos de usuarios. Para cada clase de usuario de la organización se especifican los requisitos funcionales que le afectan. Esta elección se debe a que cada usuario tiene acceso a distinta funcionalidad claramente diferenciada dentro del sistema.

2.3.2.1 Modo 1 (Anónimo)

Requisito funcional 1: Registro de usuario

El cliente introducirá sus datos de registro, siendo obligatorios el nombre y apellidos, teléfono, DNI, contraseña y dirección de correo.

Requisito funcional 2: Identificación del usuario

Se solicitará DNI y contraseña para validar los clientes en el sistema y una cadena cualquiera más una contraseña para los empleados del restaurante. Si los datos son correctos mostrará la página personalizada del usuario, y en caso contrario mostrará un error.

Requisito funcional 3: Envío de comentarios

Cualquier usuario podrá enviar comentarios para aclarar cualquier duda. Los datos se enviarán al restaurante.

Requisito funcional 4: Consulta de la carta

Cualquier usuario podrá acceder a la consulta de los platos más representativos del restaurante.

Requisito funcional 5: Consulta de los menús

Cualquier usuario podrá visualizar los menús más destacados del restaurante.

Requisito funcional 6: Visualización de las instalaciones del restaurante

Se visualizarán fotografías que muestren la decoración y organización del restaurante.

Requisito funcional 7: Visualización de información general del restaurante

Información que pueda ser del interés de los clientes, una descripción general del restaurante: ambiente, atención, estilo, etc.

2.3.2.2 Modo 2 (Identificado)

El usuario identificado incluye todos los requisitos funcionales del invitado así como los siguientes:

Requisito funcional 8: Cambio de datos del usuario

Una vez identificado, el usuario podrá cambiar sus datos.

Requisito funcional 9: Realización de un nuevo pedido

El cliente elegirá los productos que desee consumir clasificados por categorías. Cuando desee que su pedido le sea servido lo confirmará.
Además, podrá modificarlo en todo momento, añadiendo, modificando y eliminando productos cuando lo desee.

Requisito funcional 10: Acceder al historial de pedidos

La aplicación guardará el historial de pedidos de cada cliente y éste podrá acceder a información sobre dichos pedidos.

Requisito funcional 11: Acceder al pedido activo

El cliente podrá acceder al pedido que tiene activo, ver los productos que lo componen, los detalles de los mismos y si han sido servidos o no.

Requisito funcional 12: Modificación de pedidos

El cliente podrá modificar los productos que forman parte de su pedido mientras no lo haya confirmado.

Requisito funcional 13: Pagar el pedido activo

El cliente podrá elegir entre pagar con tarjeta o en efectivo. Si elige pagar con tarjeta podrá realizar el pago inmediatamente con su número de cuenta y fecha de caducidad de la tarjeta. Si elige pagar en efectivo se le mandará un aviso al camarero automáticamente como que se ha solicitado la cuenta en la mesa por el método tradicional.
Una vez haya pagado el pedido activo, podrá realizar más pedidos mientras no se cierre la sesión.

Requisito funcional 14: Realizar reserva

Los clientes registrados del restaurante podrán realizar reservas de mesas desde fuera del restaurante. Para ello tendrán que entrar en la página web del mismo, y dentro de ésta, en la zona para usuarios registrados. Una vez allí, visualizarán el estado del restaurante (mesas libres, ocupadas y reservadas), pudiendo seleccionar la mesa libre que deseen.

Requisito funcional 15: Cerrar la sesión

Se podrá cerrar la sesión una vez no haya ningún pedido activo sin pagar.

Requisito funcional 16: Consulta del catálogo de productos

Los clientes podrán consultar el catálogo de productos y precios disponible del restaurante.

2.3.2.3 Modo 3 (Camarero)

Requisito funcional 17: Creación y modificación de mesas

El usuario de perfil camarero podrá crear mesas nuevas, eliminarlas, así como modificar sus datos: número, máximo número de comensales y número de mesas que la componen.

Requisito funcional 18: Gestión de clientes

Podrá eliminar clientes y modificar sus datos.

Requisito funcional 19: Estado de las mesas del restaurante

Se accederá a una imagen gráfica del restaurante con las mesas libres, ocupadas y reservadas. A las mesas libres se les podrá asignar un cliente y un camarero. Las mesas ocupadas se podrán reasignar.

También se mostrará el camarero encargado de cada mesa, para que cada uno pueda ver rápidamente qué mesas tiene asignadas.

Requisito funcional 20: Manipular pedidos activos

El camarero accederá a la vista detallada de los pedidos activos de todas las mesas. Podrá modificar sus pedidos, cancelarlos, marcar productos como servidos y la cuenta como pagada (en caso de pago en efectivo).

Requisito funcional 21: Lectura y respuesta de comentarios

Los camareros podrán acceder a los comentarios enviados por los usuarios y responderlos.

Requisito funcional 22: Historial de pedidos

Podrá consultar el historial de pedidos del restaurante que ya están inactivos y visualizar sus detalles.

2.3.2.4 Modo 4 (Gerente)

Requisito funcional 23: Crear ofertas

El gerente podrá elegir los productos que componen una determinada oferta y especificar el precio de la misma.

Requisito funcional 24: Modificar ofertas

Podrá modificar los productos que componen una determinada oferta así como su precio.

Requisito funcional 25: Eliminar ofertas

Podrá eliminar una oferta existente.

Requisito funcional 26: Añadir producto

El gerente podrá añadir un producto al catálogo de productos del restaurante especificando las características del mismo.

Requisito funcional 27: Modificar producto

Podrá modificar las características de un producto del catálogo.

Requisito funcional 28: Eliminar producto

Podrá eliminar un producto existente en el catálogo de productos del restaurante.

Requisito funcional 29: Registrar camarero

Podrá dar de alta en el sistema a un nuevo camarero introduciendo como dato el nombre y la foto solamente, ya que sólo interesa tener a los camareros registrados para controlar su asignación a las mesas.

Requisito funcional 30: Dar de baja camarero

Se podrán borrar los datos de cualquier camarero que haya en el sistema.

Requisito funcional 31: Crear tipo de producto

El gerente podrá añadir un tipo de producto al restaurante especificando el nombre de la categoría de producto.

Requisito funcional 32: Modificar tipo de producto

Se podrá modificar el nombre de la categoría y los productos que la componen.

Requisito funcional 33: Eliminar tipo de producto

Se podrá eliminar la categoría del producto.

2.3.3 Restricciones de diseño

2.3.3.1 Estándares cumplidos

En el desarrollo de la aplicación se hará uso de XHTML para tener la certeza de una mayor compatibilidad con los navegadores. Se implementará siguiendo la versión XHTML 1.0 transicional junto con hojas de estilo CSS 2.1 para optimizar posibles cambios futuros en la estética de la aplicación.

2.3.3.2 Limitaciones hardware

Para que la aplicación funcione correctamente y que los tiempos de espera sean aceptables, es recomendable una buena conexión a Internet. En cuanto a la instalación del servidor web con soporte de ASP.NET y el de la base de datos, se podrá realizar en un computador de prestaciones medias,

pero para soportar una mayor carga de usuarios es recomendable un ordenador de mayores prestaciones.

3. ANÁLISIS

El propósito principal del análisis es obtener una descripción lógica del sistema a desarrollar, es decir, describir formalmente mediante modelos las características de la aplicación. Estos modelos servirán posteriormente de guía para obtener el producto deseado. Para ello utilizaremos el lenguaje de modelado UML (Lenguaje Unificado de Modelado). Se trata de un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Éste tiene varios diagramas aunque únicamente desarrollaremos el diagrama de clases.

3.1 DIAGRAMA DE CLASES

El diagrama de clases en UML es el diagrama principal para el modelado y el diseño en la programación orientada a objetos y sirve para representar las clases del sistema, que corresponden a tipos de usuarios, opciones, las relaciones que se establecen entre ellas, ya sean de herencia o de tipo estructural. A continuación se muestra el diagrama de clases para el *erestaurante*.

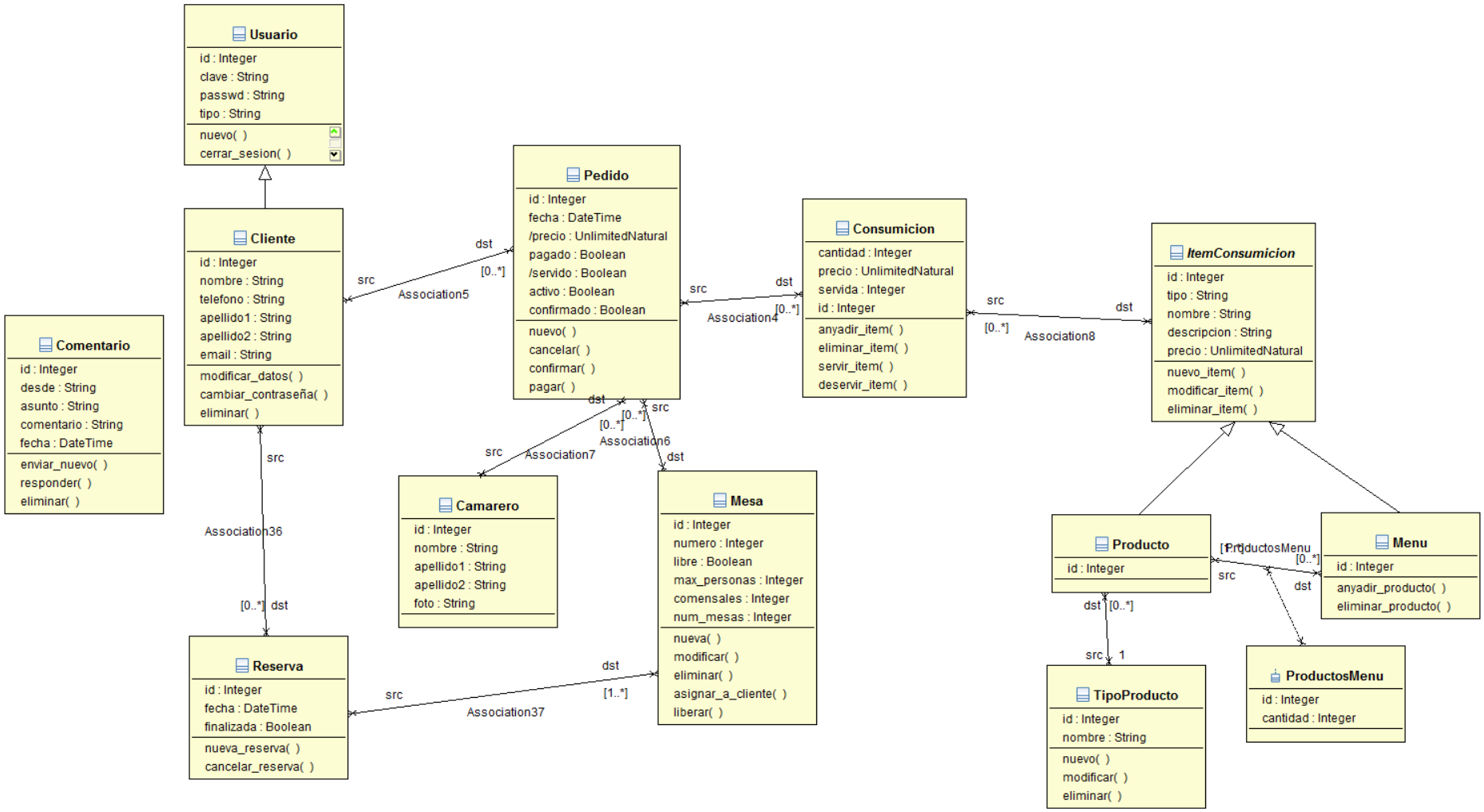


Figura 3.1.1. Diagrama de clases

Utilizamos la entidad <Usuario> para representar los usuarios que pueden autenticarse en el sistema: gerente, camarero y cliente, es por este motivo que <Cliente> hereda de <Usuario>, porque los clientes tienen las propiedades del <Usuario> más otras extra, pero el usuario gerente no necesita almacenar la información del los clientes tales como: nombre, teléfono, apellido1, etc. Algo similar sucede con los camareros, se ha decidido diferenciar entre el usuario camarero (puede autenticarse) y el empleado camarero. Esto se ha realizado en base a que cada camarero no necesita un usuario, esta aplicación está pensada para que varios camareros entren con el mismo usuario para así tener una vista general del restaurante.

Cada cliente puede tener asignados pedidos, aunque por la aplicación se controla que cada cliente solamente tenga un pedido activo, el resto de pedidos pasarán a formar parte del histórico. Cada pedido está, a su vez relacionado con un camarero que lo atenderá, una mesa y las consumiciones seleccionadas por el cliente. Una consumición no es más que un ítem de consumición que almacena las siguientes propiedades: cantidad (del ítem de consumición), precio, servida y confirmada. Asimismo, un ítem de consumición puede ser un producto (de un tipo de producto) o un menú.

Observamos también en el diagrama la entidad <Reserva>, de manera que un cliente podrá realizar una reserva para una fecha y una mesa, cuando llegue al restaurante se le creará un pedido para la mesa reservada en esa fecha o posterior.

Por último, nuestra aplicación está preparada para almacenar comentarios procedentes de clientes o usuarios anónimos.

3.2 DIAGRAMA DE USUARIOS

En nuestra aplicación web tenemos 5 tipos de usuarios y cada tipo de usuario representa un conjunto de usuarios con objetivos y responsabilidades comunes en el sistema. Estos usuarios son: anónimo, cliente, cliente web, camarero y gerente, y sus inter-relaciones así como su modo de acceso al sistema se muestran en el diagrama siguiente:

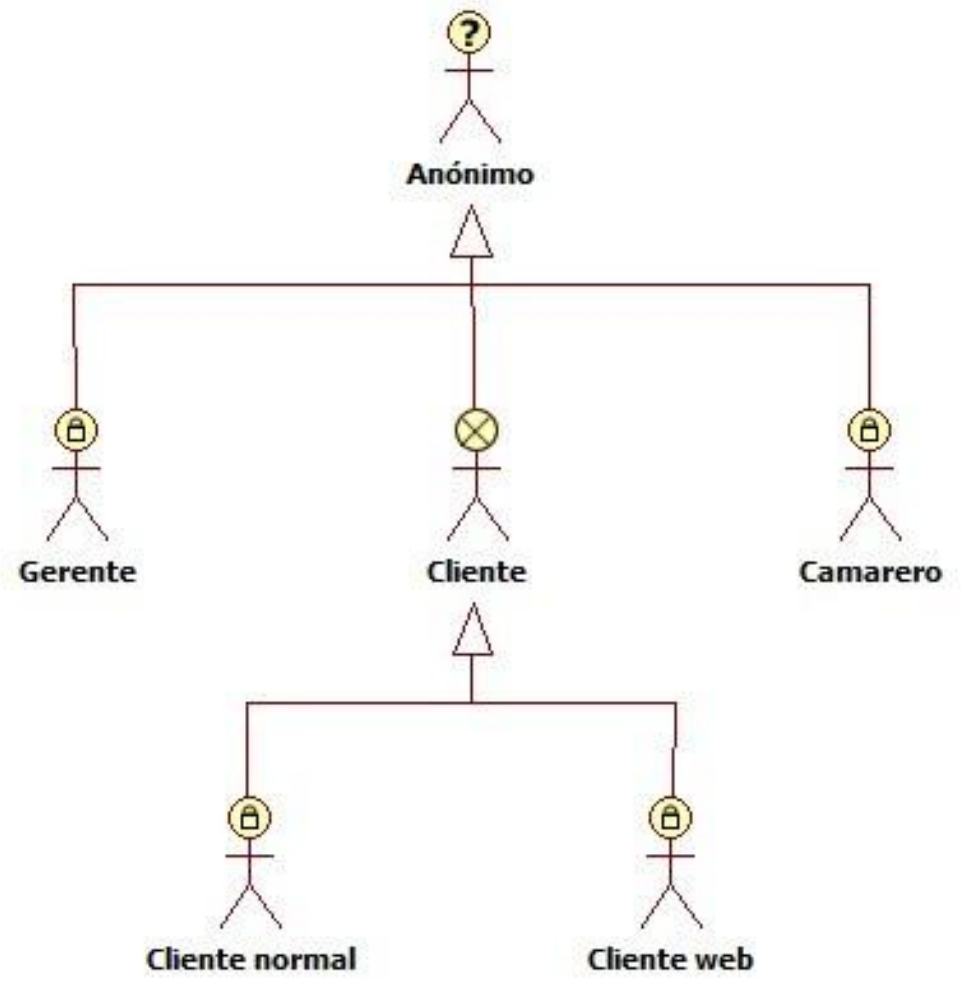


Figura 3.2.1. Diagrama de usuarios

Como se ve en la figura, a nuestra aplicación podemos acceder como usuario anónimo o como autenticado, esto es, como: gerente, camarero o cliente. El usuario cliente se especializa en dos: Cliente normal y Cliente web. El cliente normal se asocia a un cliente que accede a la aplicación físicamente desde el restaurante, mientras el cliente web a uno que accede desde fuera del restaurante y que tiene acceso a funcionalidad diferente. Resulta importante destacar que el cliente como tal no existe en la aplicación, se utiliza en el diagrama para indicar que el cliente normal (estándar) y el web comparten funcionalidad.

3.3 MODELO NAVEGACIONAL

Para representar las interfaces de usuario nos hemos basado en OOWS, un método de producción de aplicaciones web que introduce nuevos conceptos orientados a objetos para dar una noción de semántica navegacional y de presentación. Así que

para cada usuario vamos a comentar su interfaz y su navegabilidad en el sistema mediante mapas navegacionales.

3.3.1 Modelo navegacional anónimo

3.3.1.1 Mapa navegacional

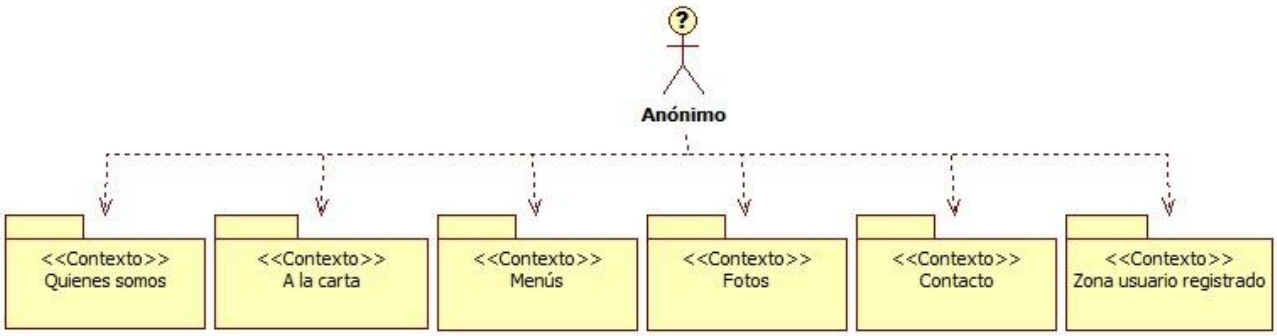


Figura 3.3.1.1.1 Mapa navegacional usuario anónimo

Todos los contextos a los que pueden acceder los usuarios anónimos son de exploración, como se puede ver su mapa navegacional es muy sencillo, con sólo un nivel. Ninguno de los contextos tiene acceso a la base de datos así que omitimos el apartado de “Contextos navegacionales”.

3.3.2 Modelo navegacional gerente

3.3.2.1 Mapa navegacional

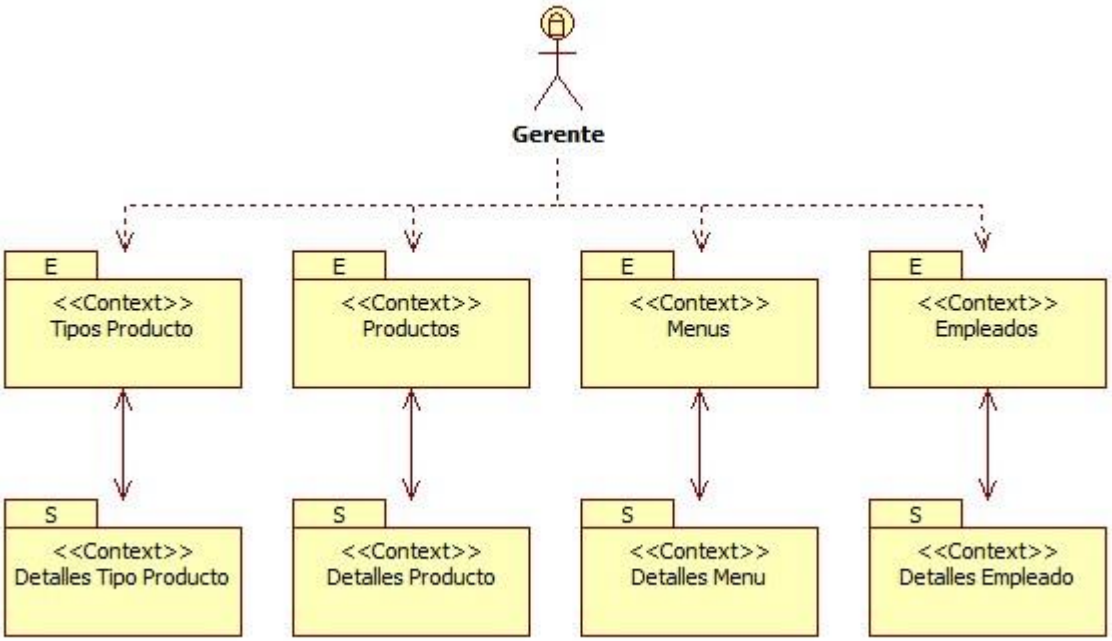
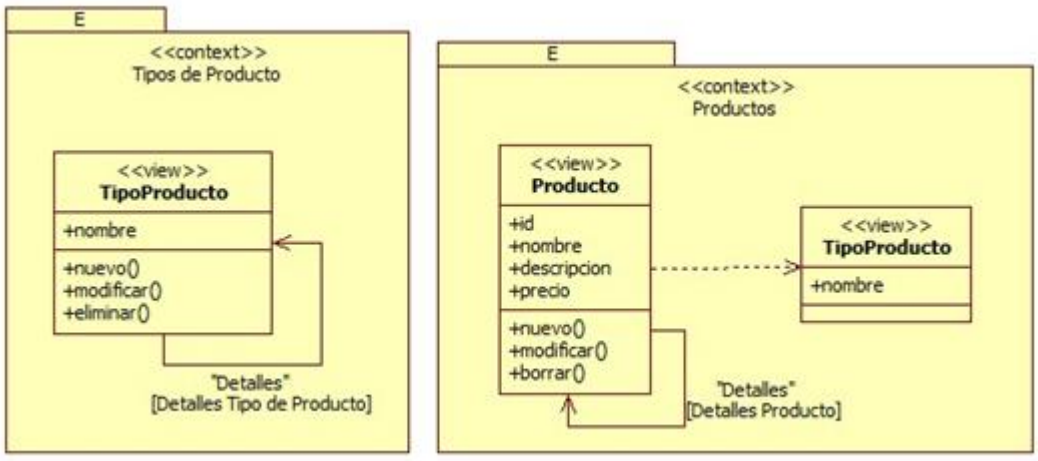
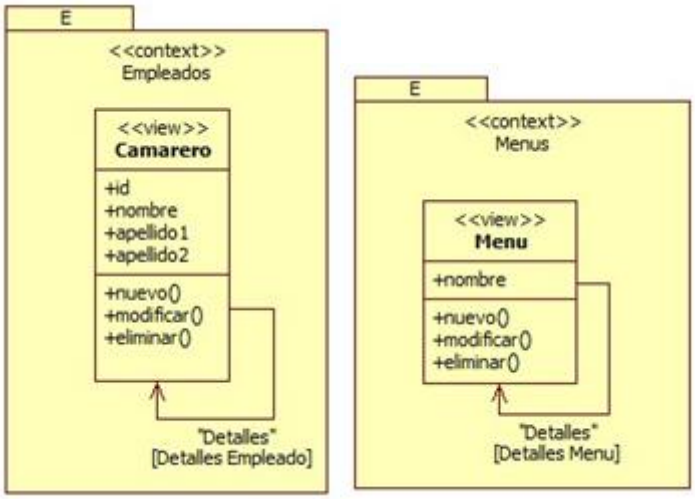


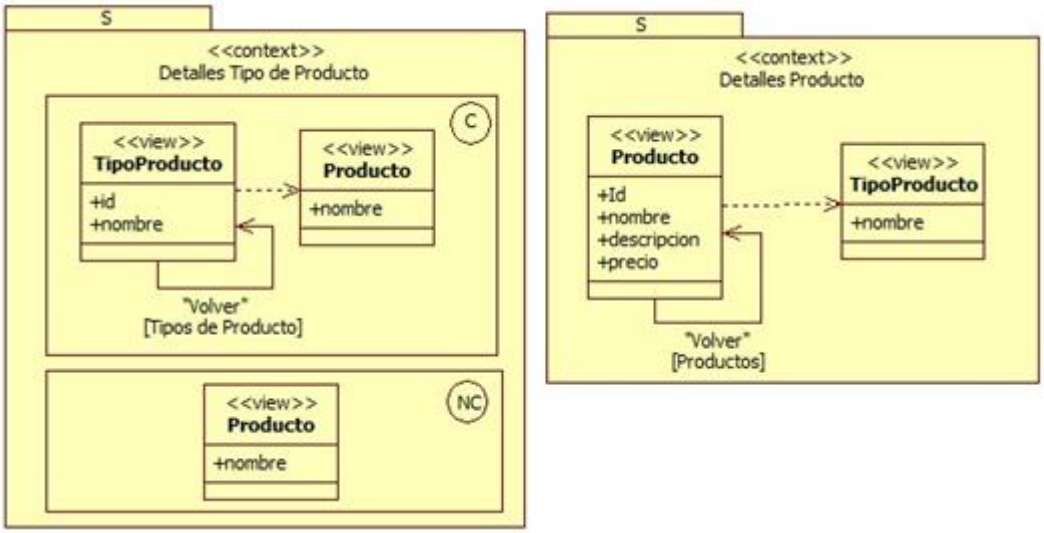
Figura 3.3.2.1.1 Mapa navegacional usuario gerente

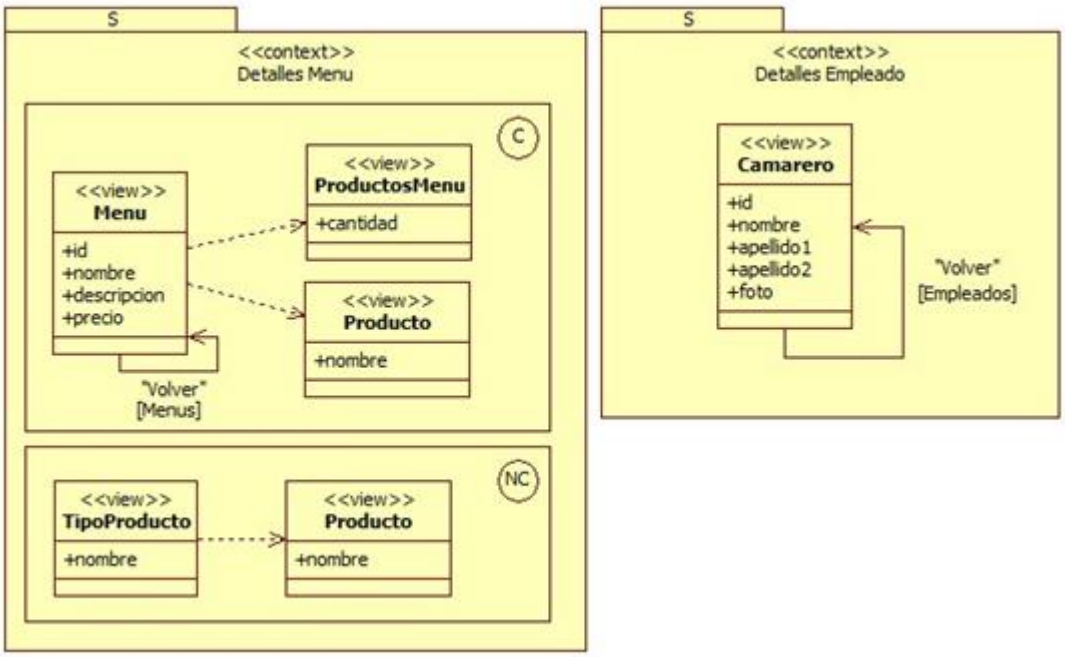
Se trata de un mapa navegacional sencillo, ya que el usuario de tipo gerente es el encargado de realizar tareas de administración: creación y modificación de productos, empleados, etc., acciones relativamente sencillas, es por esto que la longitud máxima de los caminos de navegación es uno. Se ha decidido que desde cada contexto de secuencia se pueda volver directamente al contexto de exploración desde el que llegó, sin necesidad de utilizar los enlaces de exploración para proporcionar una mayor agilidad, imprescindible en tareas de administración.

3.3.2.2 Contextos navegacionales




E
<<context>>
Empleados
<<view>>
Camarero
+id
+nombre
+apellido1
+apellido2
+nuevo()
+modificar()
+eliminar()
"Detalles"
[Detalles Empleado]
E
<<context>>
Menus
<<view>>
Menu
+nombre
+nuevo()
+modificar()
+eliminar()
"Detalles"
[Detalles Menu]


S
<<context>>
Detalles Tipo de Producto
C
<<view>>
TipoProducto
+id
+nombre
<<view>>
Producto
+nombre
"Volver"
[Tipos de Producto]
NC
<<view>>
Producto
+nombre
S
<<context>>
Detalles Producto
<<view>>
Producto
+Id
+nombre
+descripcion
+precio
<<view>>
TipoProducto
+nombre
"Volver"
[Productos]


S
<<context>>
Detalles Menu
C
<<view>>
Menu
+id
+nombre
+descripcion
+precio
<<view>>
ProductosMenu
+cantidad
<<view>>
Producto
+nombre
"Volver"
[Menus]
NC
<<view>>
TipoProducto
+nombre
<<view>>
Producto
+nombre
S
<<context>>
Detalles Empleado
<<view>>
Camarero
+id
+nombre
+apellido1
+apellido2
+foto
"Volver"
[Empleados]

Entre los ocho contextos de navegación del usuario gerente destacan entre el resto los contextos secuenciales *Detalles Tipo de Producto* y *Detalles Menu.* Los dos utilizan el concepto de VAI para la representación de los datos. Se ha recurrido a esta técnica ante la necesidad de resolver los siguientes problemas:

- *Detalles Tipo de Producto.* Se querían mostrar los datos del tipo de producto seleccionado, los productos que eran de ese tipo y, además, el resto de productos que no lo eran (que fueran de otro o que no tuvieran tipo asignado). En esto último reside el problema, ya que, la vista Producto a mostrar tenía que ser independiente del tipo de producto seleccionado.

- *Detalles Menu.* Sucede algo parecido. Es necesario mostrar información del menú seleccionado, y, además, todos los productos que no forman parte del menú, con la posibilidad de filtrarlos por tipo de producto. De nuevo tenemos que mostrar información independiente del menú seleccionado.

  Utilizamos Ⓒ para indicar que se trata del VAI principal y ⓃⒸ para informar de que es un VAI no principal.

3.3.3 Modelo navegacional camarero

3.3.3.1 Mapa navegacional

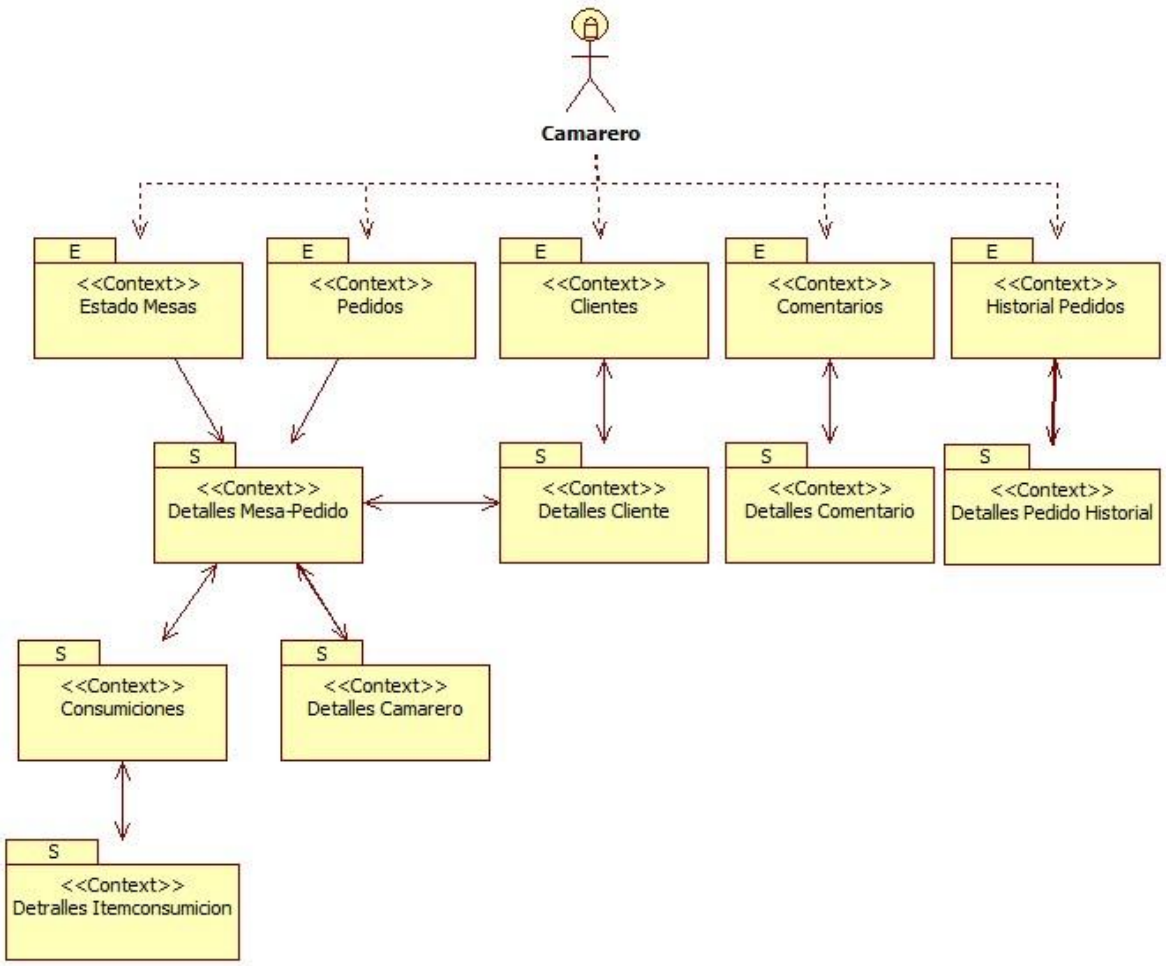
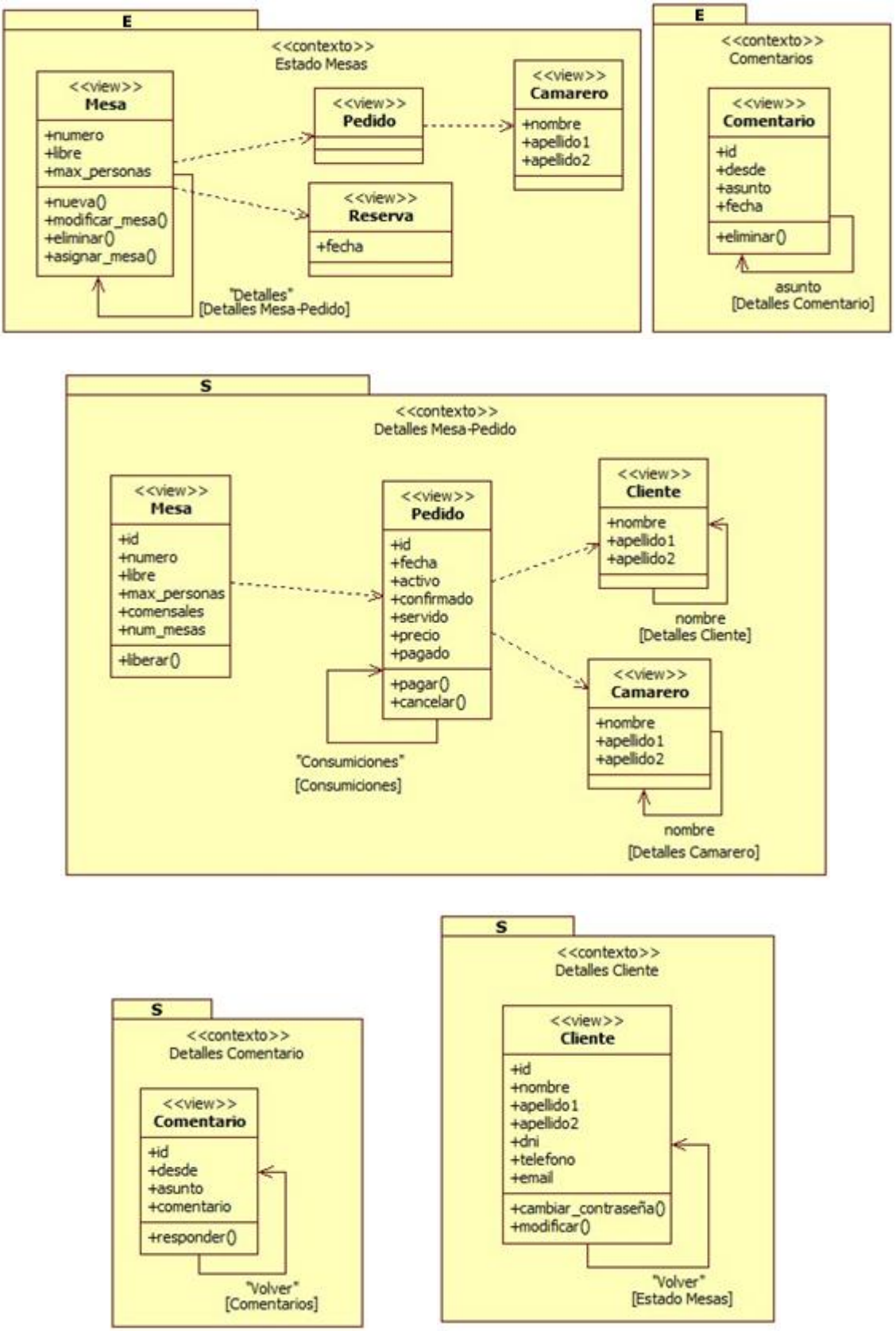


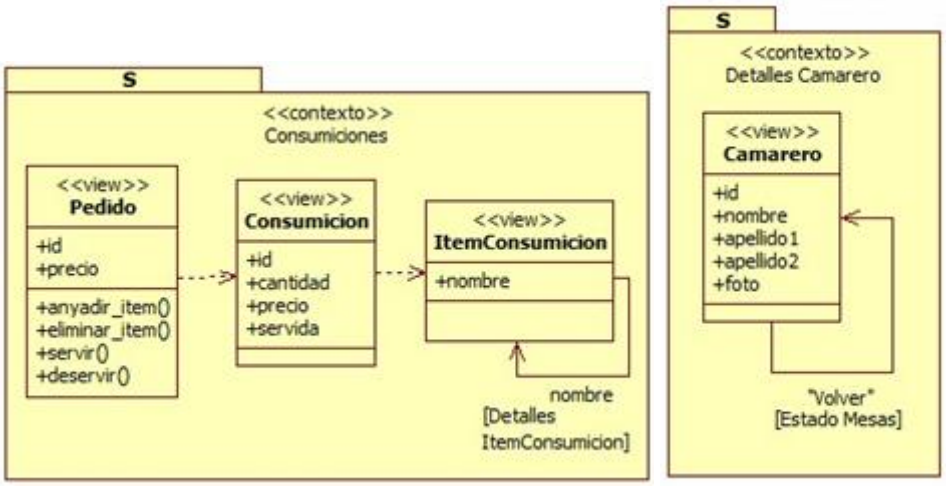
Figura 3.3.3.1.1 Mapa navegacional usuario camarero

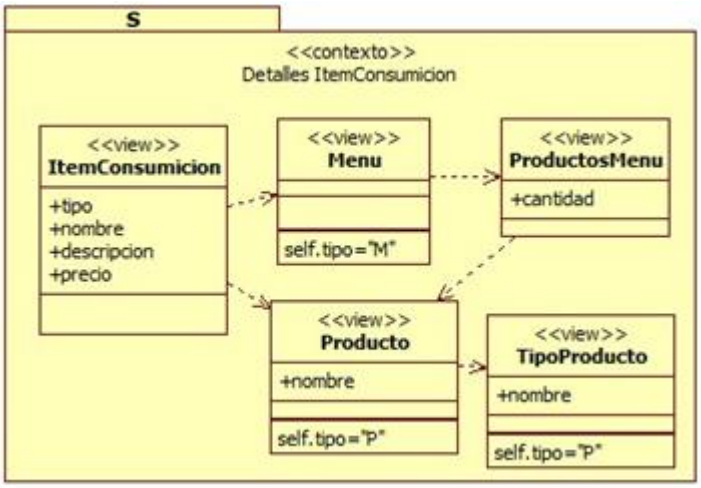
En este mapa navegacional se puede apreciar la importancia para el usuario camarero de los pedidos, ya que en relación a estos se consigue el camino de máxima longitud (3). De hecho, el contexto de mayor importancia es *Detalles Mesa-Pedido,* está muy relacionado porque muestra información muy diversa.

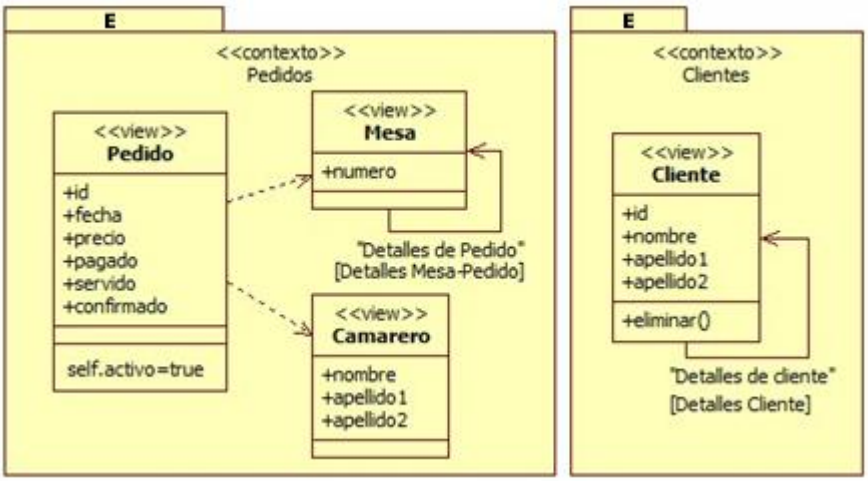
Se puede decir que el mapa está algo desequilibrado, puesto que muestra mucha información pero de importancia muy diferente. Así, los contextos más importantes son los relacionados con *Estado Mesas* y *Pedidos,* y, por lo tanto, los más completos. Mientras, los contextos *Clientes*, *Comentarios* e *Historial de Pedidos,* no tienen tanto peso en el mapa y muestran información de una manera más sencilla.

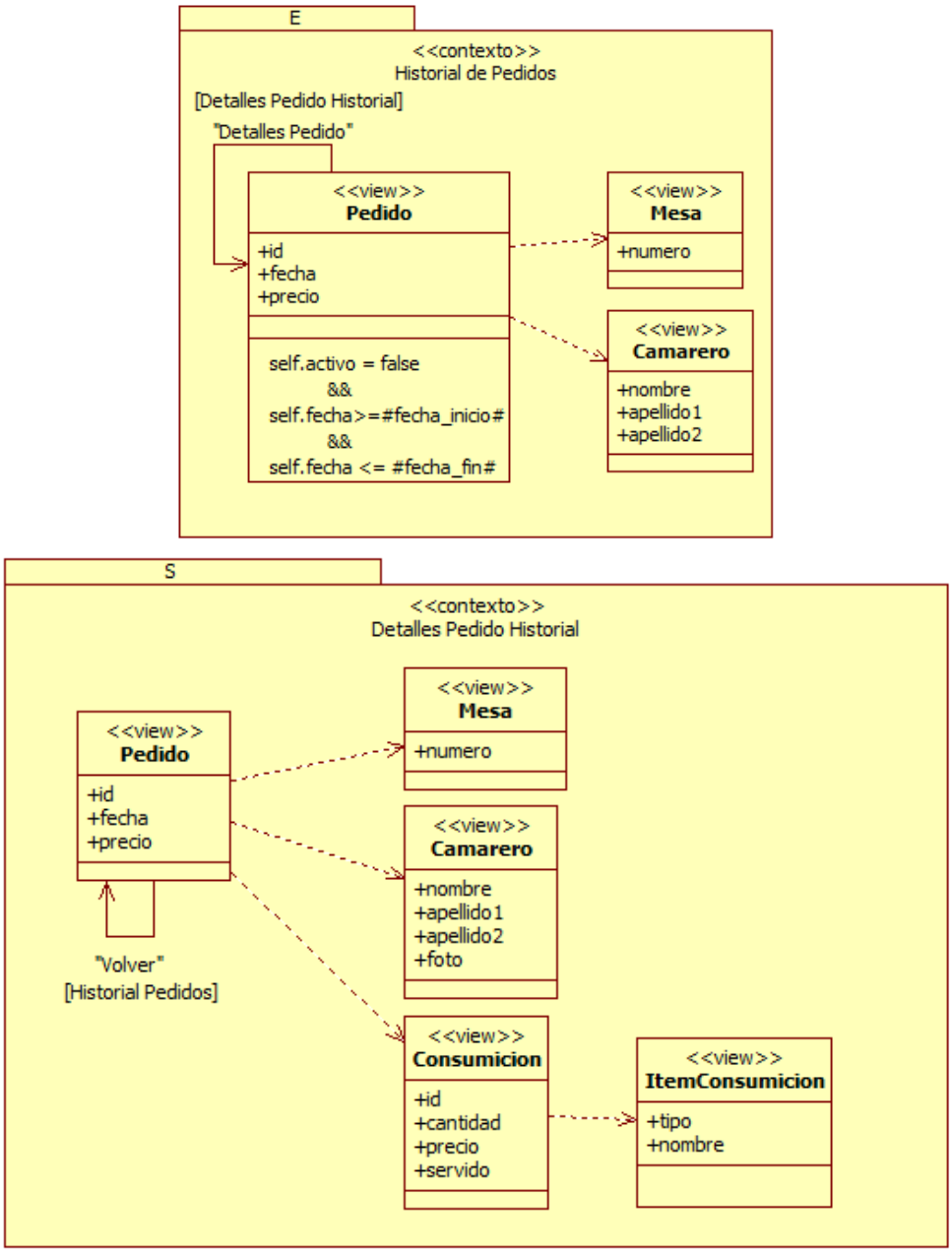
3.3.3.2 Contextos navegacionales




S
<<contexto>>
Consumiciones
<<view>>
Pedido
+id
+precio
+anyadir_item()
+eliminar_item()
+servir()
+deservir()
<<view>>
Consumicion
+id
+cantidad
+precio
+servida
<<view>>
ItemConsumicion
+nombre
nombre
[Detalles
ItemConsumicion]
S
<<contexto>>
Detalles Camarero
<<view>>
Camarero
+id
+nombre
+apellido1
+apellido2
+foto
"Volver"
[Estado Mesas]


S
<<contexto>>
Detalles ItemConsumicion
<<view>>
ItemConsumicion
+tipo
+nombre
+descripcion
+precio
<<view>>
Menu
self.tipo="M"
<<view>>
ProductosMenu
+cantidad
<<view>>
Producto
+nombre
self.tipo="P"
<<view>>
TipoProducto
+nombre
self.tipo="P"


E
<<contexto>>
Pedidos
<<view>>
Pedido
+id
+fecha
+precio
+pagado
+servido
+confirmado
self.activo=true
<<view>>
Mesa
+numero
"Detalles de Pedido"
[Detalles Mesa-Pedido]
<<view>>
Camarero
+nombre
+apellido1
+apellido2
E
<<contexto>>
Clientes
<<view>>
Cliente
+id
+nombre
+apellido1
+apellido2
+eliminar()
"Detalles de cliente"
[Detalles Cliente]



La complejidad de los contextos navegacionales del usuario camarero es bastante superior a la del usuario gerente, así que, en este caso sí vamos a proceder a la explicación de cada uno de los mismos.

- ***Estado Mesas***. El objetivo es mostrar el estado de las mesas del restaurante, si están libres, ocupadas o reservadas. Para mostrar si una mesa está libre o ocupada recuperamos el atributo “libre” de la clase “Mesa”. Recuperamos el atributo “fecha” de la clase “Reserva” para saber si una mesa tiene reservas asignadas. En el caso de tener un pedido asignado se recupera el camarero que está al cargo de dicho pedido en esa mesa.
- ***Detalles Mesa-Pedido***. Este contexto es el que más información muestra. Se recupera información de la mesa, del pedido, del cliente y del camarero.

También tenemos tres relaciones de contexto a *Consumiciones*, *Detalles Camarero* y *Detalles Cliente*.

- ***Consumiciones****.* Se muestran las consumiciones que están asociadas a un pedido. Para ello recuperamos información de la consumición y del ítem de consumición, así como el precio del pedido. Tenemos una relación de contexto con *Detalles ItemConsumicion*.
- ***Detalles Cliente****.* Muestra información ampliada del cliente. Destacan las operaciones de cambio de contraseña y de modificación de los datos del mismo. Tenemos también una relación de contexto a través del botón *Volver*.
- ***Detalles Camarero****.* Muestra información ampliada del camarero. Tenemos una relación de contexto a través del botón *Volver*.
- ***Detalles ItemConsumicion****.* Este contexto visualiza información extra del ítem de consumición. En función del tipo de ítem se presentan por pantalla datos del menú (productos del mismo y cantidad de cada uno) o datos del producto (nombre y tipo de producto).
- ***Pedidos****.* Se visualizan los pedidos que están activos, junto con la mesa y el camarero asignados. Existe una relación de contexto en *Mesa* que indica navegabilidad a *Detalles Mesa-Pedido* a través del botón *Detalles de Pedido*.
- ***Clientes****.* Se muestra la información más relevante del cliente. Destacan la operación de *eliminar* cliente y la relación de contexto a *Detalles de cliente* a través de *Detalles Cliente*.
- ***Comentarios****.* Visualiza todos los comentarios, para cada uno de ellos *id*, *desde*, *asunto* y *fecha*. Destaca la operación de *eliminar* comentario y una relación que define navegabilidad a *Detalles Comentario.*
- ***Detalles Comentario****.* Muestra pare el comentario seleccionado la misma información que en el contexto *Comentarios* a excepción de la fecha, en cuyo lugar se muestra *comentario*. Es relevante la operación de *responder* comentario y la relación de contexto para volver a la página anterior.
- ***Historial de pedidos.*** Este contexto muestra los información de los pedidos que ya no están activos (han sido pagados ya), para un rango de fechas seleccionadas. Define dos relaciones de dependencia contextual con *Mesa* y *Camarero* para recuperar información extra, y una relación de contexto, recíproca, en *Pedido,* que define navegabilidad a *Detalles de Pedido* a través del enlace “Detalles Pedido”.
- ***Detalles de pedido.*** Visualiza datos extra del pedido seleccionado, mediante relaciones de dependencia contextual con *Mesa*, *Camarero*, *Consumicion* e *ItemConsumicion*. También tiene una relación que define navegabilidad recíproca en “Pedido” a “Historial de pedidos” a través del enlace “Volver”.

3.3.4 Modelo navegacional cliente

3.3.4.1 Mapa navegacional

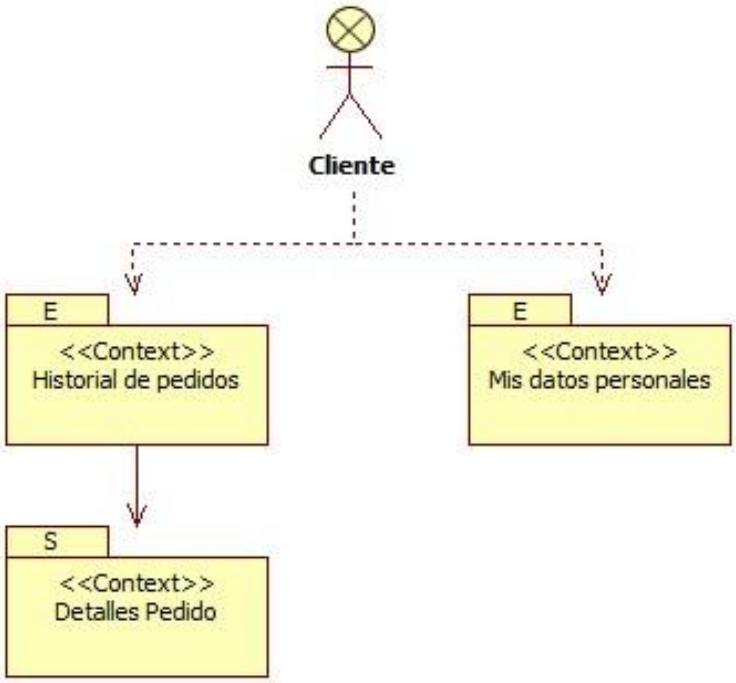
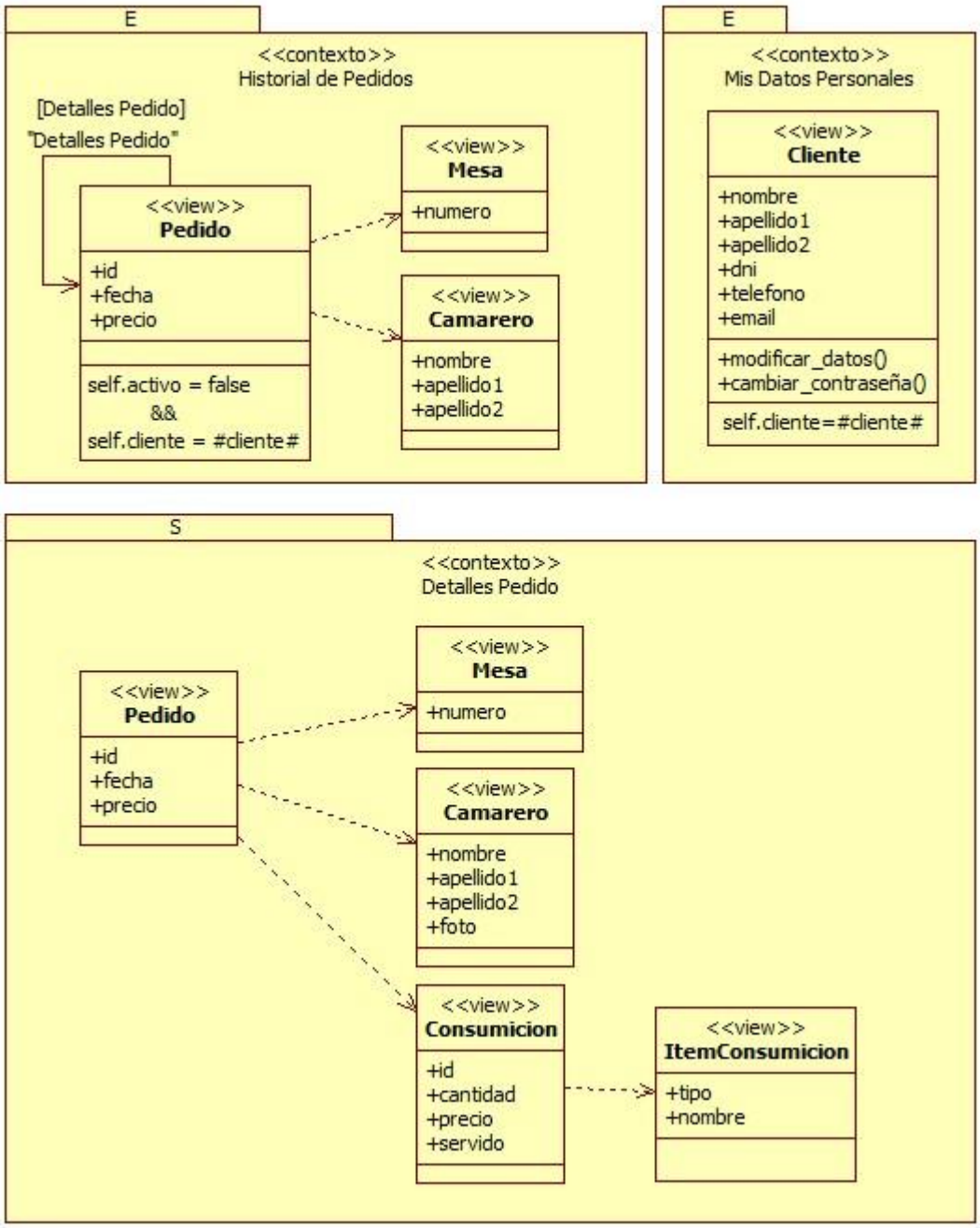


Figura 3.3.4.1.1 Mapa navegacional usuario cliente

El mapa navegacional del usuario cliente, con tal sólo dos contextos de exploración y uno de secuencia, se caracteriza por su sencillez, ya que el usuario cliente está relacionado con los usuarios *cliente normal* y *cliente web*, que son especializados de el primero, y que tienen funcionalidad extra. Precisamente esta sencillez de la que hablamos es uno de los objetivos de tener en nuestro diagrama de usuarios un usuario virtual ("sin permiso").

3.3.4.2 Contextos navegacionales



Descripción de los contextos:

- ***Historial de pedidos.*** Este contexto muestra los información de los pedidos que ya no están activos (han sido pagados ya), para el cliente conectado. Define dos relaciones de dependencia contextual con *Mesa* y *Camarero* para recuperar información extra, y una relación de contexto, recíproca, en *Pedido,* que define navegabilidad a *Detalles de Pedido* a través del enlace "Detalles pedido".

- ***Detalles de pedido.*** Visualiza datos extra del pedido seleccionado, mediante relaciones de dependencia contextual con *Mesa*, *Camarero*, *Consumicion* e *ItemConsumicion*.

- ***Mis datos personales.*** Muestra información personal del cliente autenticado en la aplicación y permite realizar las operaciones: *modificar_datos()* y *cambiar_contraseña().*

3.3.5 Modelo navegacional cliente normal

3.3.5.1 Mapa navegacional

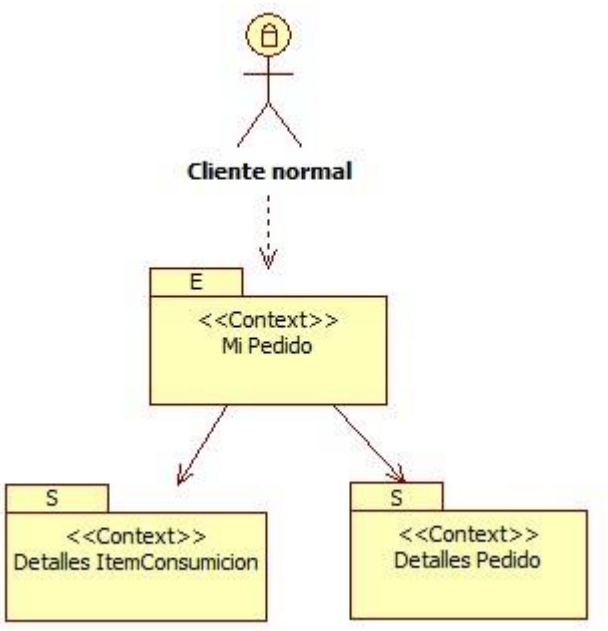
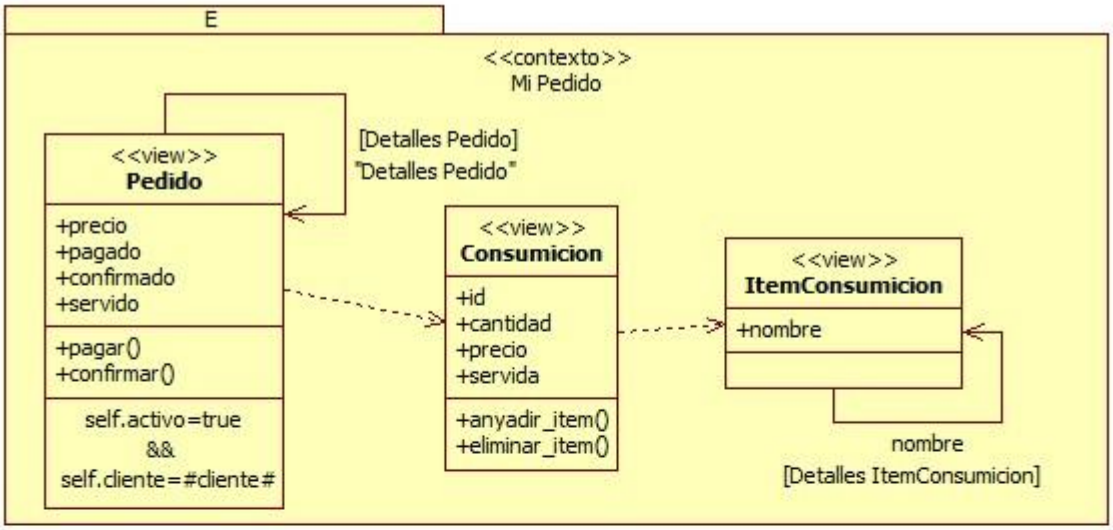


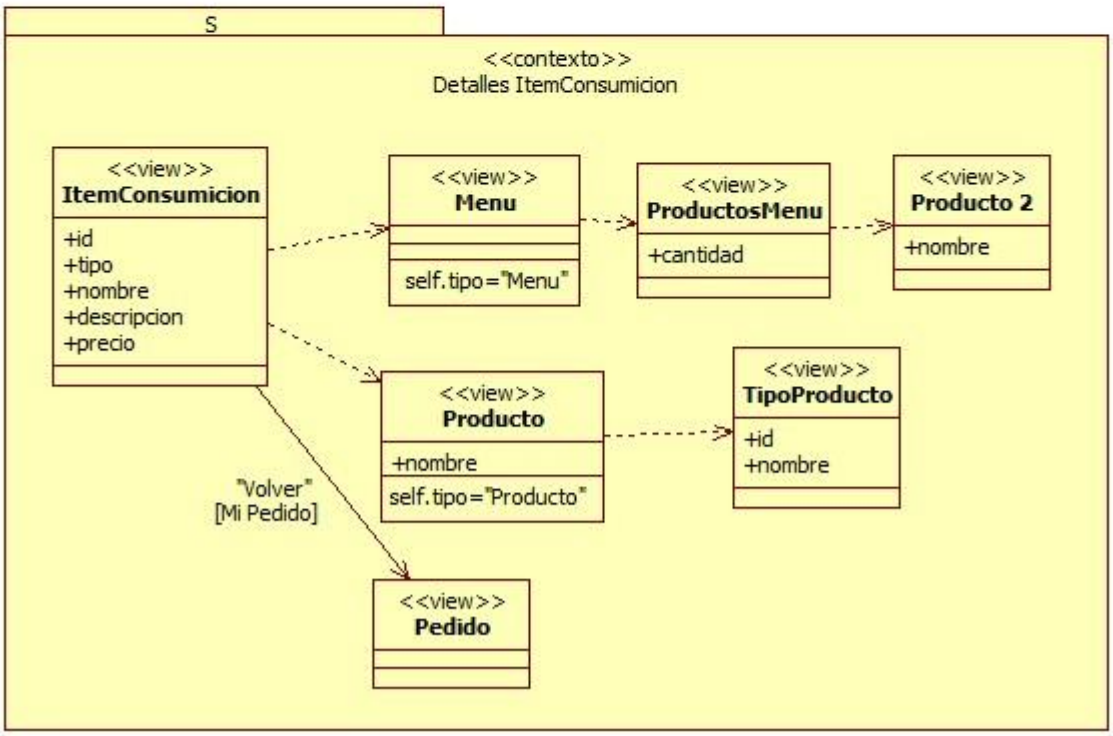
Figura 3.3.5.1.1 Mapa navegacional cliente estándar

De este mapa navegacional podemos destacar lo mismo que del mapa navegacional anterior, el del usuario virtual Cliente. Se trata de un mapa navegacional de un usuario especializado, por lo tanto, aunque el resultado es un mapa sencillo, no implica que la funcionalidad del usuario sea reducida, ya que, hereda la del usuario padre.

3.3.5.2 Contextos navegacionales

A continuación se adjuntan los contextos navegacionales del usuario "Cliente estándar", mostramos solamente los contextos que añaden nuevas propiedades navegacionales.





Descripción de los contextos:

- ***Mi Pedido.*** Muestra información sobre el pedido activo para el usuario conectado. Establece dos relaciones de dependencia contextual con *Consumicion* e *ItemConsumicion* para recuperar información extra del pedido, y dos relaciones recíprocas que definen navegabilidad: en *Pedido* con destino *Detalles Pedido* y en *ItemConsumicion* con destino *Detalles ItemConsumicion*. Además, permite realizar operaciones de pago y confirmación del pedido, y de adición y eliminación de consumiciones a/del mismo.

- ***Detalles ItemConsumicion.*** Visualizar los datos del ítem de consumición seleccionado de una manera detallada. Para ello, utiliza relaciones sin navegabilidad con *Menu*, *Producto*, *ProductosMenu* y *TipoProducto.* También tenemos una relación de contexto entre *ItemConsumicion* y *Pedido* con destino *Mi Pedido*.

  Destaca en el contexto que la vista Producto aparece duplicada y es que el contexto *Detalles ItemConsumicion* intenta mostrar información diferente según el ítem seleccionado sea un producto o un menú. De manera que sí seleccionamos un menú seguiremos la primera rama y se mostraran los productos del mismo (cantidad + nombre), mientras que si se selecciona un producto se seguirá la segunda rama y se visualizará el nombre del producto más el tipo de producto al que pertenece.

3.3.6 Modelo navegacional cliente web

3.3.6.1 Mapa navegacional

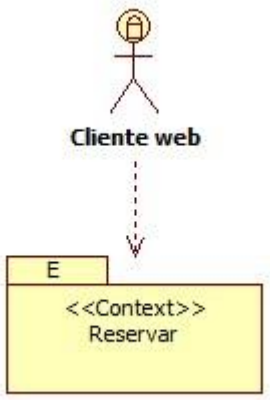
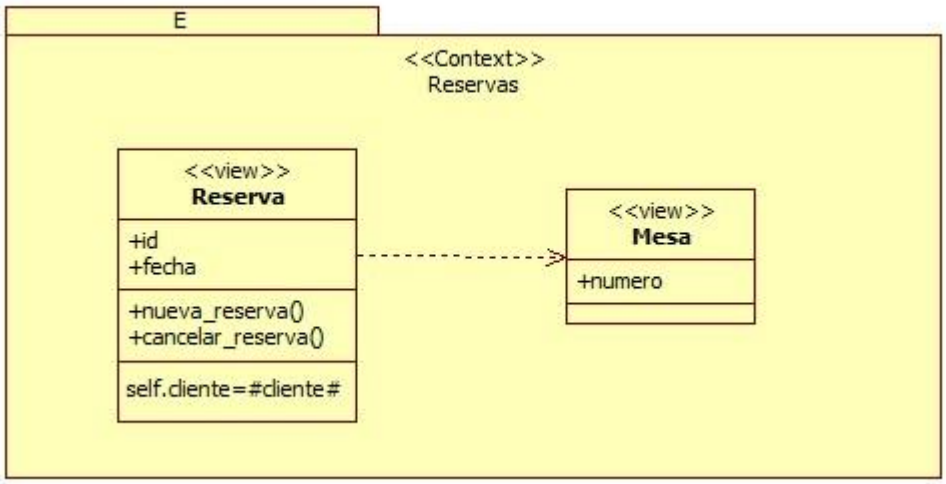


Figura 3.3.6.1.1 Mapa navegacional usuario Cliente web

La explicación es la misma que para el cliente estándar (**ver 3.3.5.1**), si bien este mapa todavía es más sencillo, ya que la funcionalidad del cliente web prácticamente se reduce a realizar reservas.

3.3.6.2 Contextos navegacionales



Descripción del contexto:

- ***Reservas.*** Este contexto muestra un listado de las reservas del cliente autenticado en el sistema, tan sólo tiene una relación de dependencia contextual con Mesa, para recuperar el número de la misma, y permite realizar las operaciones creación y cancelación de una reserva.

4. DISEÑO

El diseño de la aplicación se basa en una de las arquitecturas multicapa que se está utilizando actualmente de forma más extendida es la arquitectura de tres capas (*three-tier*) lógicas. En ella tenemos las siguientes capas:

- **Nivel de Presentación.**
- **Nivel de Dominio o de Aplicación.**
- **Nivel de Persistencia.**

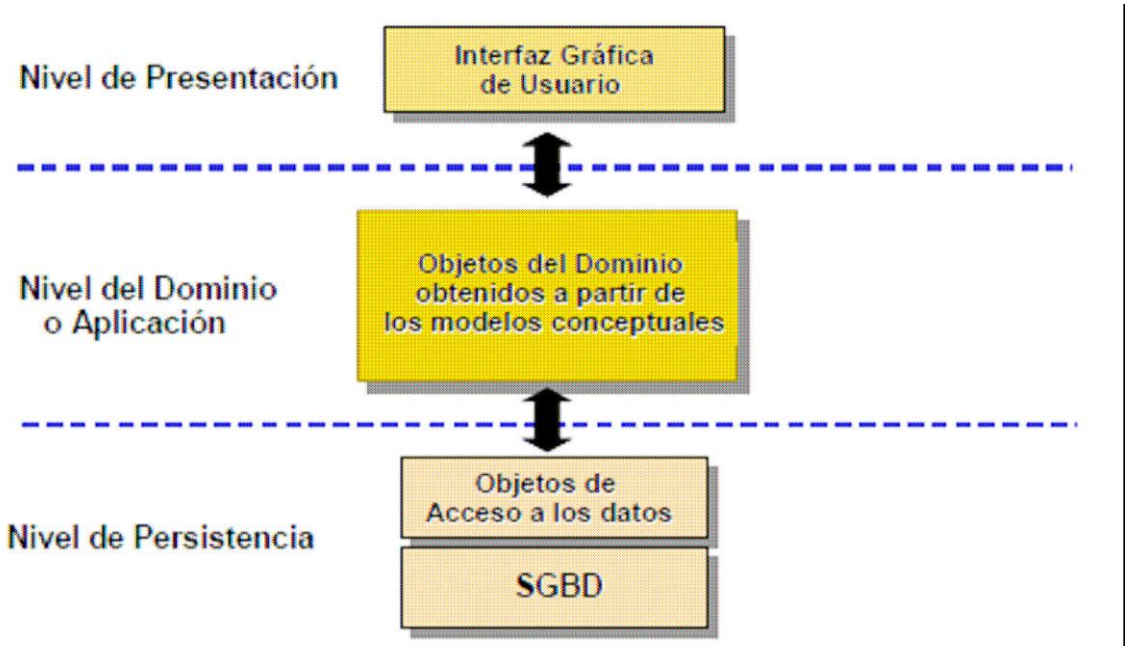


Figura 4.1. Arquitectura genérica de tres capas

4.1 NIVEL DE PRESENTACIÓN

Son los componentes software que implementan la interacción con los usuarios, a través de una representación visual de la aplicación, proporcionando a estos una forma de acceder y controlar los datos y los servicios de los objetos. En nuestro caso serán las páginas web, formularios, enlaces, tablas, etc., y darán acceso a usuarios invitados, camareros, clientes y administrador.

Cada contexto del modelo navegacional tiene como resultado una interfaz en nuestra aplicación: cada atributo representa la información que se mostrará en

cada página, y cada operación las acciones que podremos realizar desde cada una de ellas.

A continuación, mostramos las interfaces de cada uno de los usuarios.

4.1.1 Interfaz del usuario anónimo

Al acceder a la página web principal el navegador nos mostrará la página de bienvenida a eRestaurante.

Figura 4.1.1.1 Página principal

Podemos diferenciar varias zonas:

- Zona institucional **(1)**. La compone el logo de la empresa.
- Zona de enlaces de aplicación **(2)**. Aparecen enlaces a funcionalidades/enlaces comunes a todas las aplicaciones para la web: login, home, a la carta, menús, etc.
- Zona de información **(3)**. Zona encargada de mostrar información de interés. Cuando naveguemos por la aplicación, la información irá cambiando solamente en esta zona, las demás mantendrán sus contenidos, adecuándose al contexto navegacional.
- Zona de entrada de datos **(4)**. Zona encargada de proporcionar un formulario de entrada de datos al usuario para su autenticación en la aplicación. Esta zona cambiará en función de desde dónde acceda el usuario a la aplicación, mostramos a continuación un árbol informativo de qué se muestra dependiendo de la ubicación mencionada:

    - Restaurante:

- PC correspondiente a empleados → Zona empleados
- PC correspondiente a cliente → Zona clientes

○ Fuera del restaurante → Zona clientes (Web)

Seguidamente, mostramos las capturas que lo confirman.

Figura 4.1.1.2 Zona empleados

Figura 4.1.1.3 Zona clientes web

Si navegamos por los distintos enlaces de aplicación podremos acceder a información sobre la carta, el menú y fotografías del restaurante, así como enviar

comentarios que leerán y responderán los camareros del mismo. También podremos acceder a la zona de clientes, que nos permite autenticarnos en la aplicación.

A continuación podemos ver las capturas de pantalla de las páginas mencionadas y, a la vez, comprobar cómo solamente va cambiando la zona de información.

Figura 4.1.1.4 A la carta

Figura 4.1.1.5 Menús

Figura 4.1.1.6 Fotos

Figura 4.1.1.7 Contacto

Figura 4.1.1.8 Zona clientes

Si queremos dar de alta un cliente nuevo si pulsamos en el enlacé “aquí” de “regístrate aquí” accederemos a un formulario donde podremos introducir los datos del nuevo cliente.

Figura 4.1.1.9 Nuevo cliente

4.1.2 Interfaz del usuario gerente

Al autenticarnos como gerente en la aplicación accederemos a la página que mostramos a continuación.

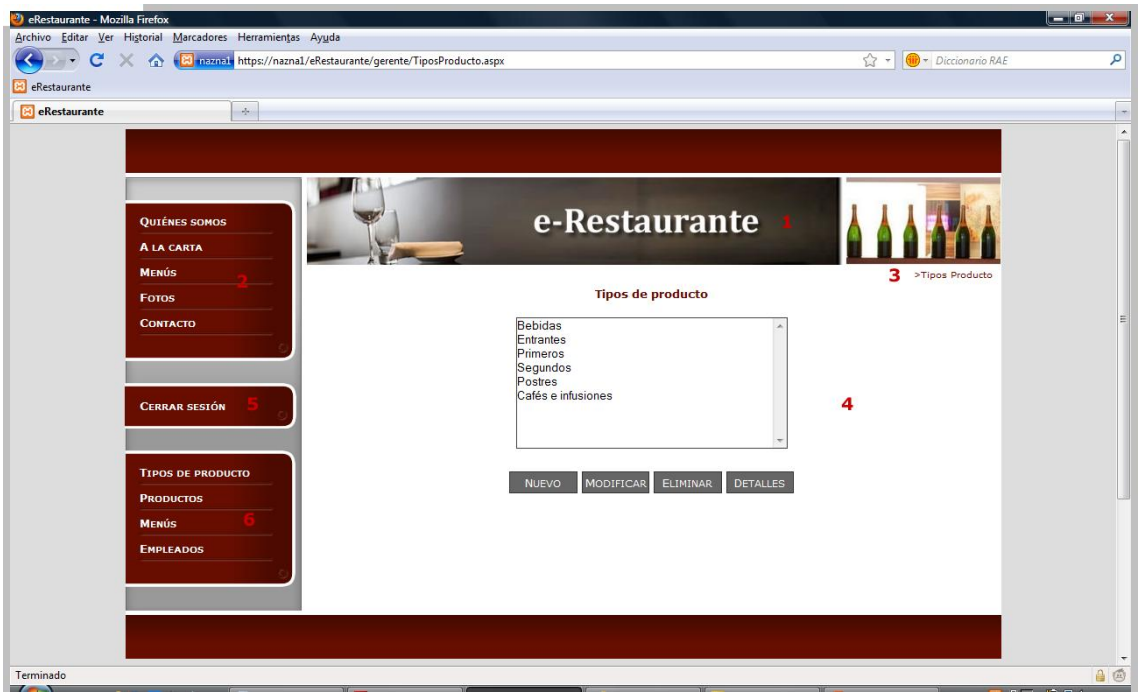


Figura 4.1.2.1 Página principal gerente: Tipos de producto

A continuación podemos ver las zonas en las que se divide la página con una breve explicación. No obstante las zonas comunes a la interfaz del usuario anónimo sólo las citamos:

- Zona institucional **(1)**.
- Zona de enlaces de aplicación **(2)**.
- Zona de ubicación **(3)**. Indica dónde estamos en el sitio web y cómo hemos llegado, es el camino navegacional.
- Zona de información **(4)**.
- Zona de logout **(5)**. Permite al usuario autentificado cerrar la sesión existente.
- Zona de navegación **(6)**. Muestra el conjunto de enlaces de exploración que el usuario puede activar.

Si navegamos por los distintos enlaces de aplicación, accederemos a información sobre los tipos de producto, productos, menús y empleados que hay registrados en el sistema. Podemos verlo a continuación, también podemos comprobar cómo va cambiando la zona de información y de ubicación en cada una de las páginas, mientras el resto se mantiene igual.

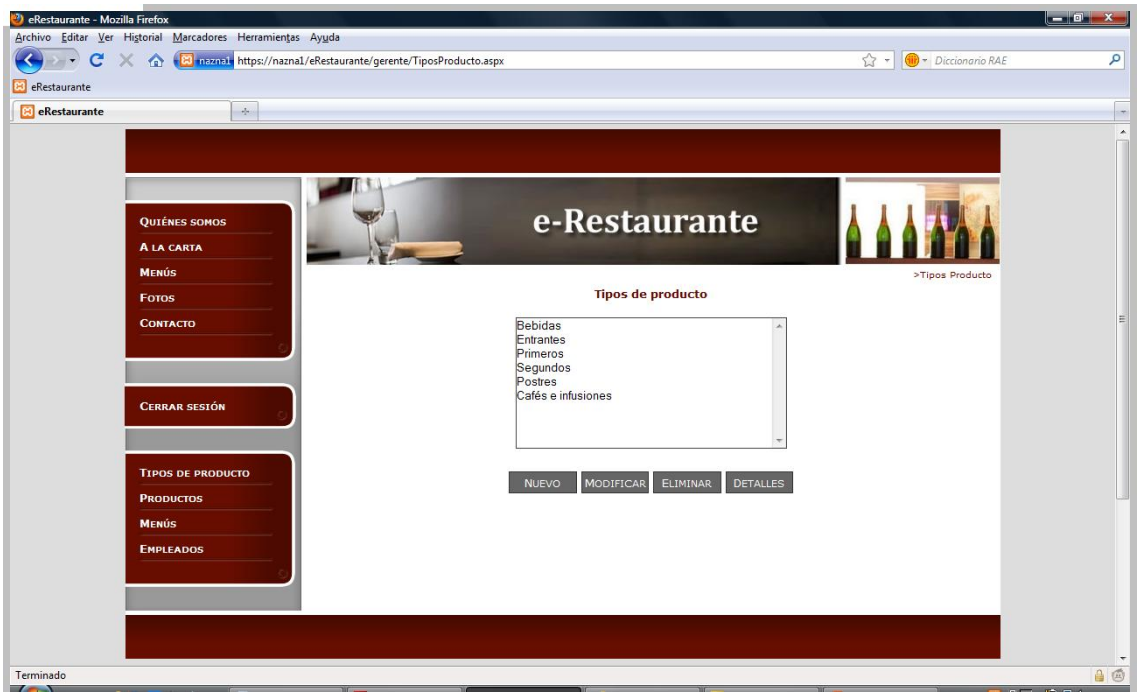


Figura 4.1.2.2 Tipos de producto

Figura 4.1.2.3 Productos

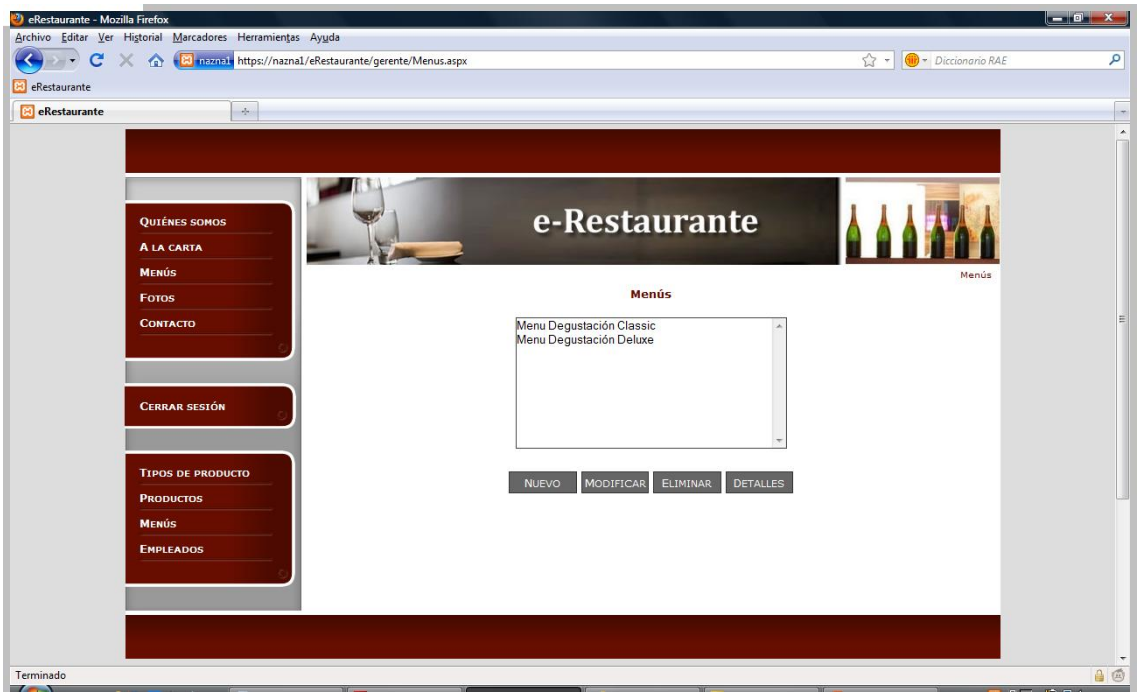


Figura 4.1.2.4 Menús

Figura 4.1.2.5 Empleados

4.1.3 Interfaz del usuario camarero

Al autenticarnos como camarero en la aplicación accederemos a la página que mostramos a continuación.

Figura 4.1.3.1 Página principal camarero: Estado de las mesas

Seguidamente se muestran las zonas en las que se divide la página, las que ya han sido explicadas en el usuario anónimo o gerente solamente las citamos.

- Zona institucional **(1)**.
- Zona de enlaces de aplicación **(2)**.
- Zona de ubicación **(3)**.
- Zona de información **(4)**.
- Zona de logout **(5)**.
- Zona de navegación **(6)**.
- Zona de estructuras de acceso **(7)**. Zona que contiene los mecanismos avanzados de exploración, en este caso los filtros de día y hora, en función de los cuales se muestra el estado de las mesas.

Si navegamos por los distintos enlaces de aplicación, podremos consultar información sobre el estado de las mesas, pedidos, clientes, comentarios e histórico de pedidos. A continuación, mostramos las páginas que se corresponden con dicha información. De nuevo, podemos ver cómo van cambiando las zonas de información y ubicación, y, también la de estructuras de acceso, ya que no es común a todas las páginas del usuario de tipo camarero.

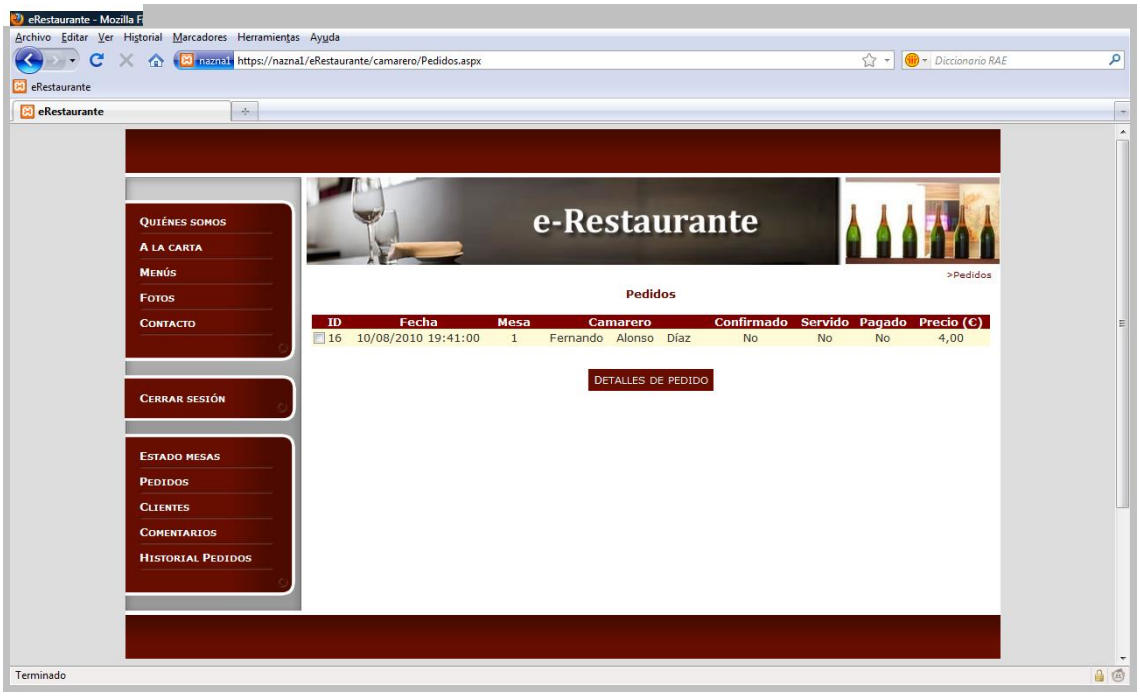


Figura 4.1.3.2 Pedidos

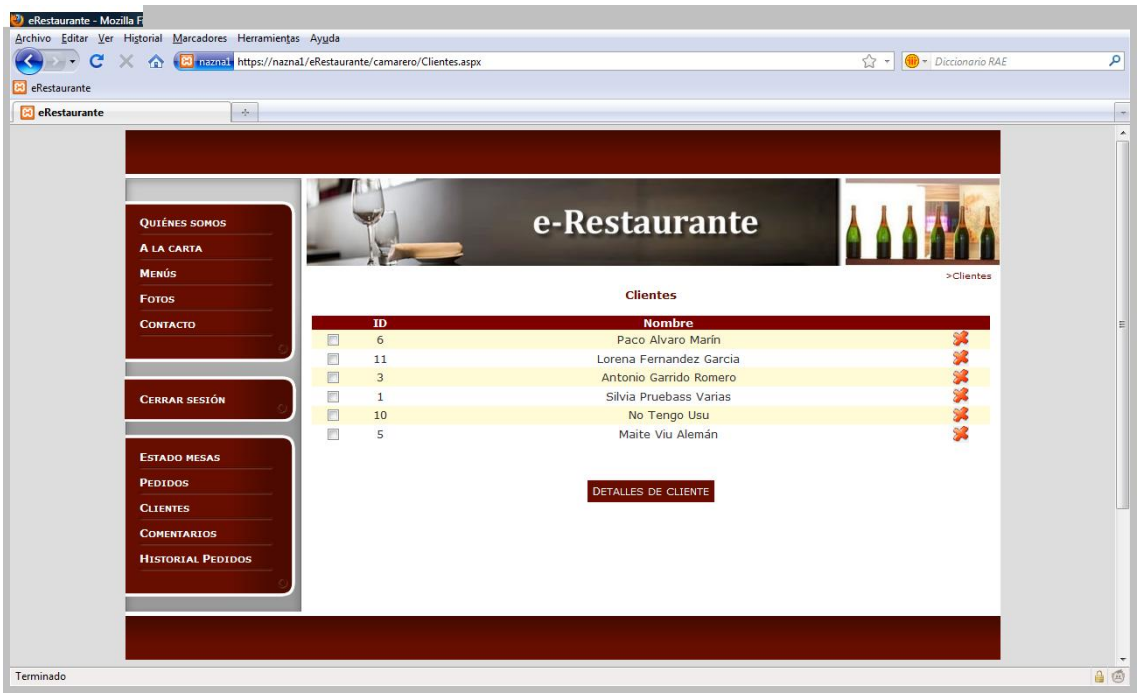


Figura 4.1.3.3 Clientes

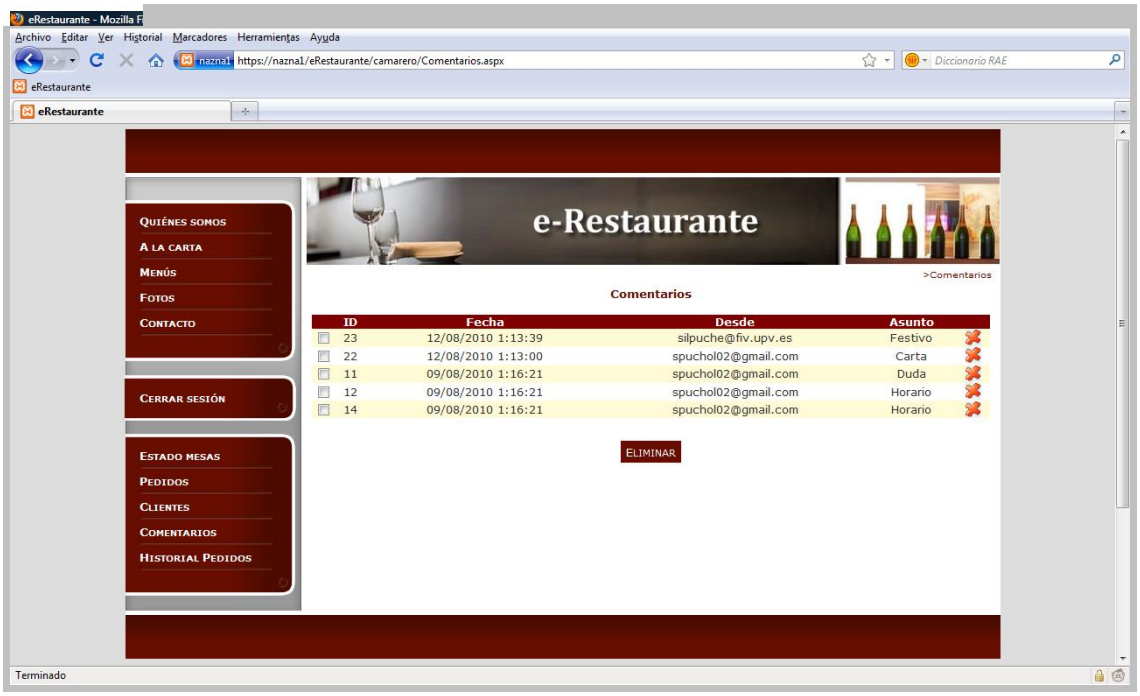


Figura 4.1.3.4 Comentarios

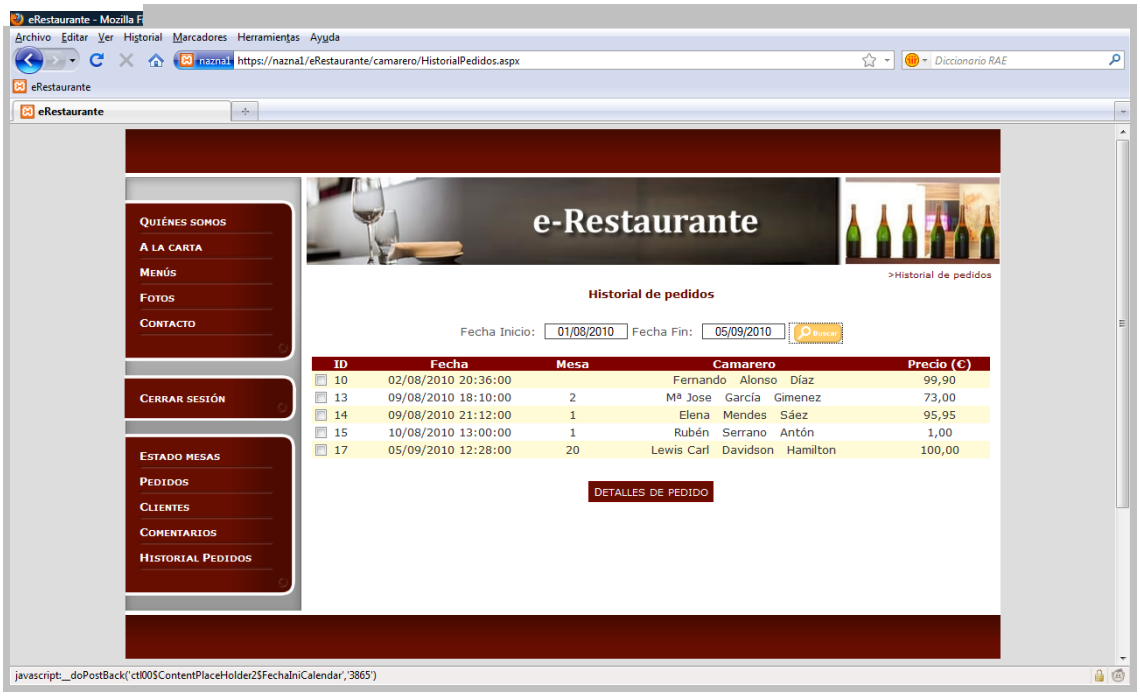


Figura 4.1.3.5 Historial de pedidos

En la página de *Historial de pedidos* también tenemos zona de estructuras de acceso, tenemos un filtro de fechas para los pedidos, ya que el volumen de pedidos inactivos en un restaurante al cabo de un mes puede ser superior a cien, y sin filtro la página sería muy poco legible.

4.1.4 Interfaz del usuario cliente

Como se comenta en la sección 3.2, el usuario cliente como tal no existe en la aplicación, se trata de un usuario abstracto. Como se ve en el diagrama de usuarios de la aplicación (**Figura 4.1.1**), de él extienden los clientes normal (estándar) y web.

Así pues, ambos tipos de cliente tienen en común los enlaces de exploración: *Historial de pedidos* y *Mis datos personales*.

Figura 4.1.4.1 Historial de pedidos (Cliente estándar)

Figura 4.1.4.2 Mis datos personales (Cliente web)

4.1.5 Interfaz del usuario cliente estándar

Al entrar a la aplicación como cliente estándar accederemos a la página de *Mi Pedido.*

Figura 4.1.5.1 Página principal usuario cliente: Mi pedido

Las zonas en las que se divide la página son las siguientes (las ya explicadas sólo las citamos:

- Zona institucional **(1)**.
- Zona de enlaces de aplicación **(2)**.
- Zona de ubicación **(3)**.
- Zona de logout **(4)**. Permite al usuario autentificado cerrar la sesión existente.
- Zona de información **(5)**.
- Zona de navegación **(6)**.

Las pantallas *Historial de pedidos* y Mis *datos personales* ya han sido mostradas para el usuario virtual Cliente y pueden ser consultadas en las figuras **4.1.4.1** y **4.1.4.2**, respectivamente.

4.1.6 Interfaz del usuario cliente web

Al entrar a la aplicación como cliente web accederemos a la página de *Reservas.*

Figura 4.1.6.1 Página principal usuario Cliente web: Reservas

Las zonas en las que se divide la página son las siguientes (las ya explicadas sólo las citamos:

- Zona institucional **(1)**.
- Zona de enlaces de aplicación **(2)**.
- Zona de ubicación **(3)**.

- Zona de logout **(4)**. Permite al usuario autentificado cerrar la sesión existente.
- Zona de información **(5)**.
- Zona de navegación **(6)**.

Las pantallas *Historial de pedidos* y Mis *datos personales* ya han sido mostradas para el usuario virtual Cliente y pueden ser consultadas en las figuras **4.1.4.1** y **4.1.4.2**, respectivamente.

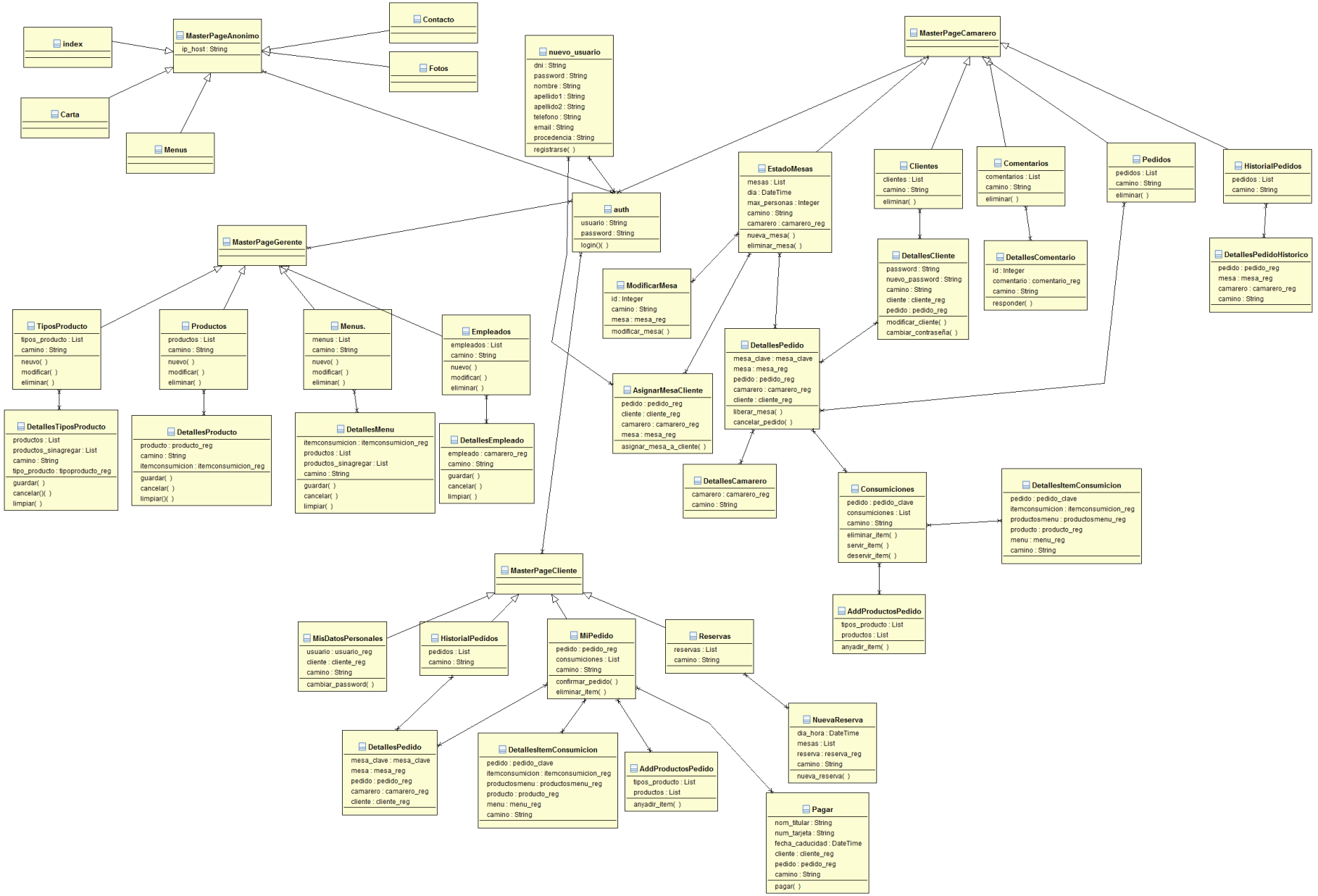
4.2 NIVEL DE APLICACIÓN

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

En esta capa se implementa el comportamiento del sistema, las operaciones descritas en el diagrama de clases, siguiendo una metodología OO (Orientada a Objetos) y son todas las clases, atributos, funciones, etc. Además, al tratarse de una aplicación web, también abarca nuevos servicios, como la gestión de usuarios (log in, altas) o la monitorización de la navegación (sesiones, caminos navegacionales).

En cuanto al acceso a la base de datos, en nuestro sistema, cada tabla de la misma se corresponde con una clase, la cual implementa los métodos de acceso. Sintaxis: “<nombretabla>_bd“ (ver más en sección 5).

A continuación, mostramos el diagrama de clases de diseño, donde se representa lo comentado en los párrafos anteriores. Podemos observar como cada clase depende de una superior identificada por el nombre “MasterPage<TipoUsuario>”. De esta manera conseguimos simplificar el diagrama y a posteriori el código de la aplicación (ver más en sección 5).



4.2.1 Diagrama de clases de diseño

4.3 NIVEL DE PERSISTENCIA

Es donde residen los datos y es el nivel encargado de acceder a los mismos. En nuestro caso está formado por un gestor de base de datos que realiza todo el almacenamiento de datos y recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Seguidamente mostramos el diagrama de base de datos del eRestaurante. Para consultar la información detallada de las tablas y columnas véase el Anexo A: Tablas de la base de datos.

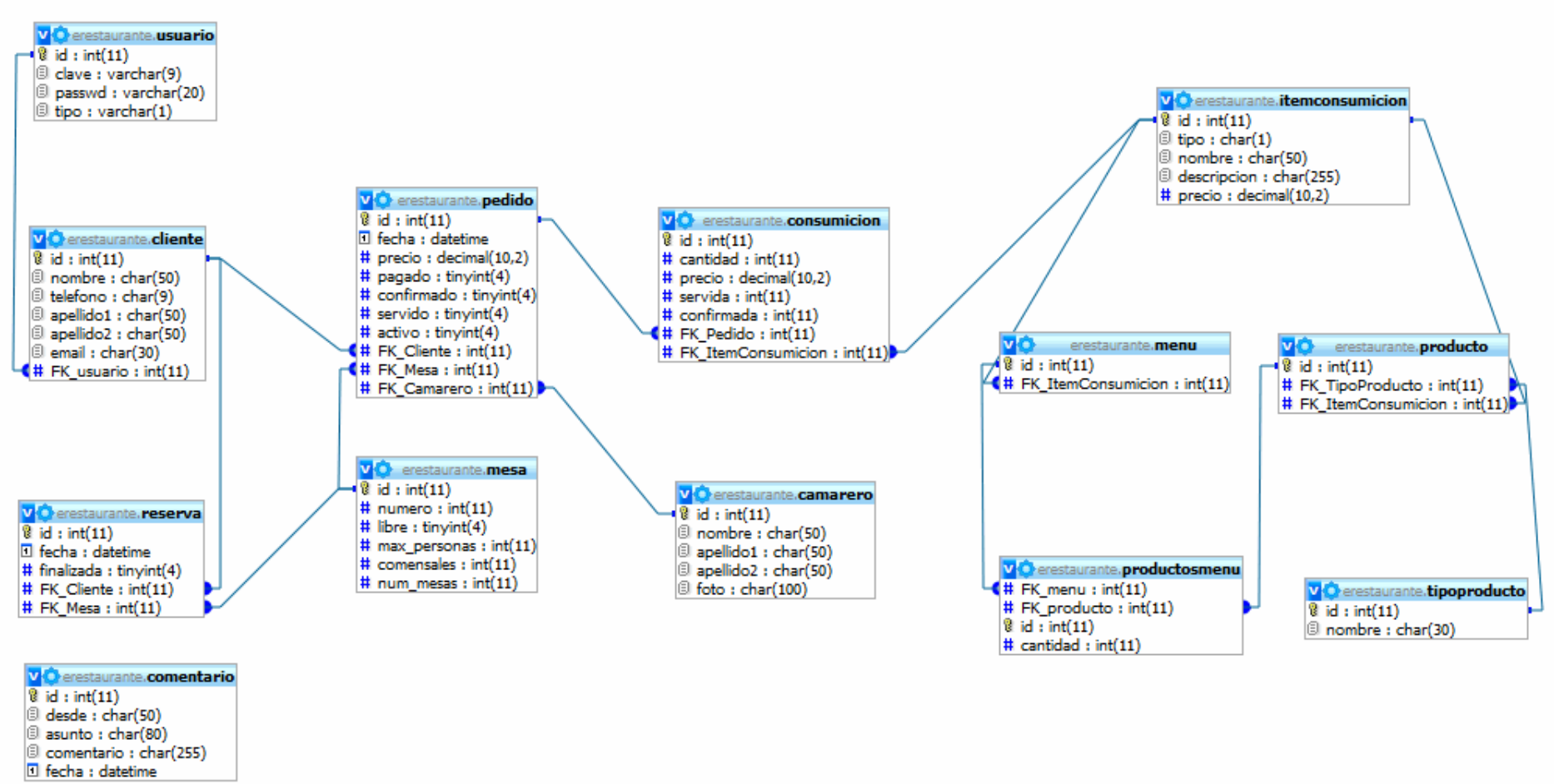


Figura 4.3.1 Diagrama de base de datos

5. IMPLEMENTACIÓN E INTEGRACIÓN

5.1 TECNOLOGÍAS

Nuestra aplicación está construida sobre tres servidores, dos de código abierto: un servidor HTTP Apache y un servidor de base de datos MySQL, y un servidor de correo Mercury que no es libre ni de código abierto, pero que sí dispone de una versión gratuita siempre que su uso sea privado y sin fines comerciales, que es nuestro caso.

Utilizamos el lenguaje ASP.NET (con VISUAL BASIC), AJAX y javascript para crear páginas web dinámicas. Resulta importante destacar también la utilización del protocolo HTTPS, que permite enviar información entre páginas de manera segura mediante cifrado SSL, así que cuando nos autentificamos en la aplicación web ya sea como cliente, camarero o gerente las páginas se muestran utilizando este protocolo seguro. A continuación describiremos dichos componentes, así como también las tecnologías en las que se sustentan.

- **Xampp**. Es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl. No obstante, nosotros lo hemos utilizado con el lenguaje ASP.NET, gracias al módulo *modaspnet*. Incluye módulos como OpenSSL y phpMyAdmin para la gestión de las bases de datos.

- **Servidor HTTP Apache.** Servidor web HTTP de código abierto, multiplataforma, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Entre otras ventajas destacan que es modular, extensible y que es utilizado por una gran comunidad de usuarios, por lo que es fácil de conseguir y existe mucha información disponible al respecto en la web.

  La mayor parte de la configuración se realiza en el fichero *apache2.conf* o *httpd.conf*, según el sistema donde esté corriendo. Cualquier cambio en este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente.

- **MySQL.** Es uno de los SGBD más empleados del mercado para aplicaciones web debido a su sencillez. Es relacional, multihilo y multiusuario. Pertenece a Sun Microsystems.

  Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso.

  Al contrario de proyectos como Apache, donde el software es desarrollado

por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional **MyISAM** (utilizado en este PFC), pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

- **Mercury Mail Transport System.** Es un servidor de correo, gratuito, basado en estándares, proporcionando un completo apoyo y un servidor rápido para los principales protocolos de correo electrónico. Va integrado en el paquete XAMPP.

- **HTML.** Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender.

  HTML es una aplicación de SGML conforme al estándar internacional ISO 8879. XHTML es una reformulación de HTML 4 como aplicación XML 1.0, y que supone la base para la evolución estable de este lenguaje. Además XHTML permite la compatibilidad con los agentes de usuario que ya admitían HTML 4 siguiendo un conjunto de reglas.

  En sus orígenes, HTML era un lenguaje diseñado para compartir información entre científicos de todo el mundo. Era puramente un lenguaje estructural, donde no había forma de describir la apariencia de las páginas (ni tan solo la posibilidad de poner un texto en negrita o cursiva). Más adelante se añadieron numerosas opciones para dar formato y presentar texto y gráficos.

  A mediados de los 90 empezaron las ampliaciones de HTML para conseguir presentaciones mejoradas, pero siempre desde diferentes perspectivas de cada desarrollador, que acabaron en diferentes soluciones no estándares para diferentes navegadores. Esto provocó la aparición de un consorcio que controla la evolución del HTML: elW3C (*World Wide Web Consortium*). Esta evolución tenía un punto clave: la separación del contenido y la apariencia. Con la versión 4 del HTML se recomendaba otro mecanismo para controlar la visualización del contenido HTML, las hojas de estilo (CSS: *Cascade Style Sheet*).

Actualmente se recomienda el uso del XHTML, que mantiene la misma sintaxis y mecanismos que el HTML, pero reformulado con un documento XML, preparándose para aprovechar las ventajas de este lenguaje.

- **ASP.NET.** ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones web comercializado por Microsoft. Es usado para construir sitios web domésticos, aplicaciones web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP).

  Se trata una tecnología de páginas activas que permite el uso de diferentes scripts y componentes en conjunto con el tradicional HTML para mostrar páginas generadas dinámicamente.

  Microsoft desarrolló una nueva tecnología denominada ASP.NET - como parte de su estrategia .NET- para el desarrollo Web, con el objetivo de resolver las limitaciones de ASP.

  ASP.NET es mucho más que la próxima versión de ASP. Su arquitectura ha sido rehecha desde cero para facilitar al máximo la creación de aplicaciones Web dinámicas, y el modo en que estructuramos el código ASP.NET también promueve una mejor reutilización. Mientras que las aplicaciones tradicionales ASP utilizan la extensión .asp, las páginas ASP.NET utilizan la extensión .aspx. Sin embargo, podemos utilizar tanto páginas ASP como ASP.NET en un mismo sitio Web. El modelo de ASP.NET, con muchas características nuevas, permite escribir código más limpio y más fácil de reutilizar y compartir, incrementando el rendimiento y la escalabilidad al poder acceder a lenguajes compilados, no interpretados.

  Otra de sus ventajas es que soporta muchos lenguajes compilados. La versión actual de ASP está basada en lenguajes de scripting como VBScript y JScript. No hay nada malo en ello, pero el modelo presenta el inconveniente de la propia interpretación del lenguaje y que los lenguajes de scripting no están fuertemente tipados. Ambos conllevan a considerar con rigor los aspectos relacionados con el rendimiento.

  ASP.NET, aunque no descarta totalmente la idea de los lenguajes de scripting, introduce soporte para lenguajes completamente compilados, ofreciendo al desarrollador escribir el código en Visual Basic, C++ o C#. De hecho, gracias a la incorporación del conjunto de tipos comunes y el funcionamiento global del CLR, ofrece un verdadero entorno neutral independiente del lenguaje para las aplicaciones Web.

- **AJAX.** Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la

comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

No obstante, no todo en Ajax son ventajas, a continuación enumeramos algunos problemas e inconvenientes: Las páginas con AJAX son más difíciles de desarrollar que las páginas estáticas; las páginas creadas dinámicamente mediante peticiones sucesivas AJAX, no son registradas de forma automática en el historial del navegador, así que haciendo clic en el botón de "volver" del navegador, el usuario no será devuelto a un estado anterior de la página, en cambio puede volver a la última página que visitó; los motores de búsquedas no entienden JavaScript, el sitio con Ajax usa más recursos en el servidor; es posible que páginas con Ajax no puedan funcionar en teléfonos móviles, PDA u otros aparatos, etc.

- **JavaScript.** Es un lenguaje de scripting basado en objetos no tipados y liviano, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. JavaScript ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no programan.

  Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

  JavaScript se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

- **CSS.** Es un mecanismo simple que describe cómo se va a mostrar un

documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los *Estilos* definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

- **HTTP.** El protocolo de transferencia de hipertexto (*HTTP, HyperText Transfer Protocol*) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceder a una página web, y la respuesta de esa web, remitiendo la información que se verá en pantalla. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con mensajes y otros similares.

  HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. Está basado en el modelo cliente-servidor: Un cliente HTTP abre una conexión y realiza su solicitud al servidor, el cual responde generalmente el recurso solicitado y la conexión se cierra. Al finalizar la transacción todos los datos se pierden.

  Por esto se popularizaron las cookies, que son pequeños ficheros guardados en el propio ordenador que puede leer un sitio web al establecer conexión con él, y de esta forma reconocer a un visitante que ya estuvo en ese sitio anteriormente. Gracias a esta identificación, el sitio web puede almacenar gran número de información sobre cada visitante, ofreciéndole así un mejor servicio.

  La versión actual de HTTP es la 1.1, y su especificación está en el documento RFC-2616. HTTP dispone de una variante cifrada mediante SSL llamada HTTPS.

- **HTTPS.** El protocolo HTTPS es la versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor

remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. Cabe mencionar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. Es aquí, cuando nuestro navegador nos advertirá sobre la carga de elementos no seguros (HTTP), estando conectados a un entorno seguro (HTTPS).

Los protocolos https son utilizados por navegadores como: Safari (navegador), Internet Explorer, Mozilla Firefox, Opera,... entre otros.

Es utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.

El puerto estándar para este protocolo es el **443**.

Para conocer si una página web que estamos visitando, utiliza el protocolo https y es, por tanto, segura en cuanto a la transmisión de los datos que estamos transcribiendo, debemos observar si en la barra de direcciones de nuestro navegador, aparece https al comienzo, en lugar de http.

Algunos navegadores utilizan un icono (generalmente un candado) en la parte derecha de la barra de direcciones para indicar la existencia de un protocolo de comunicaciones seguro e incluso cambian el color del fondo de la barra de direcciones por amarillo (Firefox) o verde (Internet Explorer) para identificar páginas web seguras.

5.2 HERRAMIENTAS

Se ha utilizado como herramienta base de este proyecto, por su sencillez y facilidad de instalación y de gestión, el paquete **XAMPP 1.7.1**, que contiene:

- **Servidor web Apache 5.0.51a.**
- **SGBD MySQL 5.1.33.**
- **MyPhpAdmin 3.1.3.1.**
- **Servidor de correo Mercury/32 v4.6.**

Para la creación del certificado de seguridad que nos permite usar el protocolo https en nuestro servidor web , por lo tanto, que la información comprometida se envíe de forma segura entre páginas, hemos utilizado **OpenSSL**, de código abierto y libre descarga.

Para el análisis y el diseño de nuestra aplicación hemos empleado:

- **MOSkitt 0.9.0.** Herramienta Case libre basada en Eclipse. La hemos utilizado para la creación de los diagramas de clases y base de datos, ya

que permite realizar una transformación bastante acertada desde el primero al segundo.

- **StarUML.** Herramienta de modelado UML de código abierto, gratuita, destacada por su rapidez y sencillez. Con ella se ha realizado el resto de diagramas de la etapa de diseño:

    - Diagrama de usuarios
    - Mapas navegacionales
    - Contextos navegacionales

Para la implementación del código hemos utilizado **Microsoft Visual Web Developer 2008 Express,** ya que, es el entorno de desarrollo con ASP.NET más completo del mercado.

Llegado a este punto resulta necesario explicar el porqué de la utilización de un servidor Apache con sistema de gestión de base de datos MySQL en combinación con el lenguaje ASP.NET, que, obviamente, no es lo más habitual ni lo más eficiente. El porqué es muy sencillo, simplemente es una cuestión de investigación y, de alguna manera, un reto: conseguir hacerlas funcionar y, comprobar de primera mano cómo se comporta una aplicación mezclando dos tecnologías que, a priori, no están preparadas o pensadas para trabajar juntas. En el apartado correspondiente se expondrán las conclusiones.

5.3 DETALLES DE LA IMPLEMENTACIÓN

PÁGINAS PRINCIPALES

Para que el desarrollo del sitio web resultara más sencillo, rápido y organizado a medida que avanzáramos en él, hemos utilizado páginas principales (*master pages*), concretamente, una página principal por cada tipo de usuario (anónimo, empleado, gerente y cliente), ya que, cada uno de estos usuarios precisan de una zona de navegación distinta (un menú distinto). De esta forma, una vez tenemos las cuatro páginas principales creadas ya sólo tenemos que preocuparnos del contenido de las mismas, y es lo único que tendremos que implementar en el resto de páginas, que se basaran en éstas.

Las páginas principales permiten definir el aspecto, el diseño y el comportamiento estándar que se desea que tengan todas las páginas (o un grupo de páginas) de la aplicación en una sola página principal. A partir de ella se pueden crear páginas de contenido individuales que incluyan lo que se desea mostrar. Cuando los usuarios solicitan las páginas de contenido, éstas se combinan con la página principal para dar como resultado una página con el diseño de la página principal y el contenido de la página de contenido.

En definitiva, su utilización nos da muchas ventajas, ya que proporcionan una funcionalidad que tradicionalmente los programadores creaban copiando el código, el texto y los elementos de control existentes repetidamente, mediante conjuntos de marcos, archivos de inclusión de elementos comunes, controles de usuario de ASP.NET, etc. Entre las ventajas de las páginas principales se incluyen las siguientes:

- Permiten centralizar las funciones comunes de las páginas para que las actualizaciones puedan llevarse a cabo en un solo lugar.

- Facilitan la creación de un conjunto de controles y código, y aplican los resultados en un conjunto de páginas. Por ejemplo, se pueden utilizar los controles en la página principal para crear un menú que se aplique a todas las páginas.

- Proporcionan un modelo de objetos que permite personalizar la página principal a partir de páginas de contenido individuales.

A continuación, le voy a dar un enfoque más práctico a este apartado para tratar de demostrar lo comentado.

Para que una página sea página principal debe tener un encabezado similar al que ponemos de ejemplo al comienzo de la página de diseño (la que tendrá el código HTML):

```
<%@ Master Language="VB" CodeFile="MasterPageAnonimo.master.vb"
Inherits="MasterPageAnonimo" %>
```

En la propiedad “CodeFile” deberemos poner la página que almacena el código de nuestra página principal.

Deberemos especificar en ella la zona o zonas de contenido (pueden haber varias), es decir, donde se situará la información de las páginas que hereden de la página principal. Esto se consigue mediante uno o varios controles ASP.NET llamados “ContentPlaceHolder”. Además, como la aplicación la hemos desarrollado con AJAX debemos añadir también un ScriptManager en la sección en la que queramos utilizar estos controles:

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
EnablePartialRendering="true" />
<asp:ContentPlaceHolder id="ContentPlaceHolder2" runat="server">
</asp:ContentPlaceHolder>
```

A excepción de estos dos detalles el código HTML será como el de cualquier página (teniendo en cuenta que cualquier control asp.net debe ir dentro de una etiqueta <form> que se ejecute en el servidor <form runat=”Server”>) y para que se vea claro ponemos un ejemplo:

```
<%@ Master Language="VB" CodeFile="MasterPageAnonimo.master.vb"
Inherits="MasterPageAnonimo" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>eRestaurante</title>
    <link rel="Stylesheet" type="text/css" href="css/style.css"/>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <div id="container">
        <div id="page">
            <div id="header"></div>
            <div id="menu">
                <div class="menu_top"></div>
                <div>
                   <div class="menu_middle"><a id="quienes_somos"
            href="/eRestaurante/index.aspx">Qui&eacute;nes
            somos</a></div>
                   <div class="menu_middle"><a id="a_la_carta"
            href="/eRestaurante/secciones/Carta.aspx">A la
            carta</a></div>
                   <div class="menu_middle"><a id="menus"
            href="/eRestaurante/secciones/Menus.aspx">Men&uacute;s</a
            ></div>
                   <div class="menu_middle"><a id="fotos"
            href="/eRestaurante/secciones/Fotos.aspx">Fotos</a></div>
                   <div class="menu_middle"><a id="contacto"
            href="/eRestaurante/secciones/Contacto.aspx">Contacto</a>
            </div>
                </div>
                <div class="menu_bottom"></div>
                <div id="zona_cliente">
                    <a href="/eRestaurante/anonimo/auth.aspx">
                        <asp:HyperLink ID="ZonaHyperLink"
            NavigateUrl="/eRestaurante/anonimo/auth.aspx"
            runat="server" Text=""></asp:HyperLink>
                    </a>
                </div>
            </div>
```

```
<div id="content_ext">
   <div id="navigational_way"> </div>
   <div id="content_int">
         <form id="form1" runat="server">
            <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server" />
            <asp:ContentPlaceHolder id="ContentPlaceHolder1"
            runat="server">

            </asp:ContentPlaceHolder>
            </div>
         </form>
    </div>
</div>
<div id="footer">
</div>
</div>
</div>
</body>
</html>
```

En cuanto a las páginas que hereden de una página principal deben comenzar con esta cabecera:

```
<%@ Page Title="" Language="VB"
MasterPageFile="~/MasterPageAnonimo.master" AutoEventWireup="false"
CodeFile="index.aspx.vb" Inherits="img_Default" %>
```

MaterPageFile: ruta de la página principal de la que hereda.
CodeFile: página que tiene el código fuente de esta página.
Y donde queramos que vaya el contenido utilizar un control ASP.NET llamado "Content":

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
    AQUI VA EL CONTENIDO
</asp:Content>
```

Seguidamente mostramos al igual que hemos hecho con la página principal un código completo de página.

```
<%@ Page Title="" Language="VB"
MasterPageFile="~/MasterPageAnonimo.master" AutoEventWireup="false"
CodeFile="index.aspx.vb" Inherits="img_Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
    <div class="texto">
```

```
<div><span class="destacado">eRestaurante</span> le da la bienvenida
y espera que quede satisfecho con nuestros servicios.</div>
          <br />

            <div>Nuestro objetivo es que disfrute con nuestra
            comida, elaborada con la máxima calidad y el mayor de
            los cuidados, as&iacute; como
            ofrecerle una <span class="negrita">metodolog&iacute;a
            innovadora</span> en la atenci&oacute;n al cliente, que
            pretende darle acceso a un servicio en el que las
            esperas se
            minimizan.</div>
         <img id="eRestauranteTenedores" src="img/tenedores.jpg"
alt="eRestaurante" title="Carlos Porto / FreeDigitalPhotos.net"
class="imagen-tenedores" />

          <br />

          <div>Usted podr&aacute; realizar su pedido desde el primer
instante en el que tome asiento, seleccionando los productos que
desee en nuestra <span class="negrita">carta online</span>.
          Nuestros camareros quedar&aacute;n informados al minuto
autom&aacute;ticamente y estar&aacute;n, por supuesto, en todo
momento a su disposici&oacute;n.</div>
          <br />
          <div>Le ofrecemos adem&aacute;s un ambiente agradable,
moderno y acogedor, acorde a nuestro <span class="negrita">estilo
innovador</span>.</div>

      </div>
</asp:Content>
```

HOJAS DE ESTILO (CSS)

Otro detalle que puede resultar interesante comentar es en relación a la implementación de las CSS. Tenemos una sola hoja de estilo para toda nuestra aplicación, no obstante, esto no es trivial en cualquier aplicación, ya que, Internet Explorer 8 cumple los estándares establecidos por el W3C en mucha menor medida que Mozilla, lo que ocasiona que lo que con una hoja de estilos se ve perfectamente en Mozilla, se vea distinto en IE8 y al revés.

Para solucionar este problema utilizamos un *hack*, los caracteres "\9" detrás de la propiedad css que nos convenga, lo que hace que Visual Studio nos avise de que tenemos fallos pero no evitan de ninguna manera que la aplicación web funcione correctamente. Este *hack* lo que hace es que el navegador IE entienda el estilo pero Mozilla no, de manera que podemos definir estilos diferentes para la misma propiedad.

A continuación se muestra un ejemplo para terminar de comprenderlo.

```
.mini-boton
{
   background-color: #666666;
   color: White;
   border: solid 1px #333333;
   width: 70%;
   width: 100%\9;
}
```

El navegador Mozilla cogerá el ancho 70% y el Internet Explorer 100%.

AJAX

La incorporación de la tecnología AJAX al proyecto ha sido fundamental para dotarla de rapidez y mayor dinamismo.

Para el funcionamiento de esta tecnología en nuestra aplicación sobre Apache hemos tenido que añadir los ficheros *ScriptResource.axd* y WebResource.axd a la raíz del proyecto web, si no los ponemos, el servidor los busca, no los encuentra y no funciona correctamente.

UpdatePanel y Triggers

Para el desarrollo con esta tecnología se ha necesitado tener la utilización de los siguientes controles.

- **UpdatePanel**

  La utilización del control *UpdatePanel* es la base del desarrollo web con AJAX. Definimos dentro de un *UpdatePanel* los controles que queremos que se actualicen sin necesidad de refrescar toda la página, de forma que solamente se actualizarán las zonas que recojan el evento lanzado por un control determinado.

  Seguidamente, mostramos un ejemplo gráfico de la distribución de *UpdatePanel* sobre nuestra aplicación.

Figura 5.3.1 Dos UpdatePanel

Como se ve en la figura anterior, en la página *MisDatosPersonales.aspx* tenemos dos UpdatePanel:

- **UpdatePanel 1**. Contiene los datos personales del cliente.
- **UpdatePanel2**. Contiene los botones de acción.

Al pulsar sobre alguno de los botones del UpdatePanel2 se actualizarán los datos personales (UpdatePanel1) sin necesidad de hacer *postback*. Además, en este caso, también cambiarán los botones. Resultado:

Figura 5.3.2 Botón modificar pulsado

Como se ve en la figura los campos de datos han cambiado y también los botones, sin embargo, la página no ha sufrido una actualización completa.

- **Triggers**

  Se utiliza el control *AsyncPostBackTrigger* para que los controles puedan ser los desencadenadores de un control *UpdatePanel*. Los controles que son los desencadenadores de un panel de actualización provocan una actualización del contenido del panel después de una devolución de datos asincrónica.

  Utilizamos un Trigger de tipo *AsyncPostBackTrigger* por cada evento de un control que queramos que actualice nuestro panel al desencadenarse.

  Siguiendo con nuestro ejemplo, los triggers para el UpdatePanel 1 son el evento click de los botones:

  - Modificar.
  - Cambiar contraseña.
  - Cancelar cambiar contraseña.
  - Cancelar.

Resulta necesario destacar que estos dos controles explicados son la base, pero ni mucho menos los únicos, AJAX es mucho más rico y extenso, pero son los que más predominan en nuestra aplicación.

A continuación mostramos un ejemplo de nuestro código con estos controles.

```
<asp:Content ID="Content3" ContentPlaceHolderID="ContentPlaceHolder2"
Runat="Server">
    <div class="centrar">
       <div class="marron"><span class="negrita">Mis Datos
       Personales</span></div><br />
    </div>
    <div class="contenedor">
        <div class="caja-centrada-3">
            <asp:UpdatePanel ID="DatosUpdatePanel" runat="server"
            UpdateMode="Conditional">
                <ContentTemplate>
                    <!— AQUI VAN LOS TEXTBOX PARA LOS DATOS -->
                </ContentTemplate>
                <Triggers>
                    <asp:AsyncPostBackTrigger ControlID="ModificarButton"
                    EventName="Click" />
                    <asp:AsyncPostBackTrigger ControlID="CancelarButton"
                    EventName="Click" />
                   <asp:AsyncPostBackTrigger ControlID="CambiarPasswdButton"
                   EventName="Click" />
                   <asp:AsyncPostBackTrigger
                   ControlID="CancelarCambiarPasswdButton" EventName="Click" />
                </Triggers>
            </asp:UpdatePanel>
       </div>
    </div>
    <asp:UpdatePanel ID="BotonesPassUpdatePanel" runat="server"
              UpdateMode="Conditional">
        <ContentTemplate>
            <div class="centrar">
                <asp:Button ID="ConfirmarNuevoPasswdButton" runat="server"
                      CssClass="boton2-autoancho" Text="Confirmar contraseña"/>
                <asp:Button ID="CancelarCambiarPasswdButton" runat="server"
                CssClass="boton2-autoancho" Text="Cancelar"/>
            </div>
        </ContentTemplate>
        <Triggers>
              <asp:AsyncPostBackTrigger ControlID="CambiarPasswdButton"
              EventName="Click" />
              <asp:AsyncPostBackTrigger ControlID="CancelarCambiarPasswdButton"
              EventName="Click" />
        </Triggers>
    </asp:UpdatePanel>

    <div class="centrar">
         <asp:UpdatePanel ID="BotonesUpdatePanel" runat="server"
              UpdateMode="Conditional" RenderMode="Inline">
            <ContentTemplate>
                 <asp:Button ID="ModificarButton" runat="server" CssClass="boton2-
                     autoancho" Text="Modificar"/>
                 <asp:Button ID="CambiarPasswdButton" runat="server"
                     CssClass="boton2-autoancho" Text="Cambiar contraseña"/>
                 <asp:Button ID="GuardarButton" runat="server" CssClass="boton2-
                     autoancho" Text="Guardar" />
                 <asp:Button ID="CancelarButton" runat="server" CssClass="boton2-
                     autoancho" Text="Cancelar" />
            </ContentTemplate>
            <Triggers>
                 <asp:AsyncPostBackTrigger ControlID="CambiarPasswdButton"
                    EventName="Click" />
                 <asp:AsyncPostBackTrigger ControlID="CancelarCambiarPasswdButton"
                    EventName="Click" />
            </Triggers>
        </asp:UpdatePanel>
       </div>
</asp:Content>
```

AjaxControlToolKit

AjaxControlToolKit es un proyecto de código abierto que proporciona una poderosa infraestructura para utilizar extensores y controles AJAX.

En este proyecto lo hemos utilizado para mostrar los *pop up* de la aplicación: ventanas modales y calendarios. La razón: simplificaba en gran medida el código y nos proporciona una solución AJAX, siguiendo con nuestra intención de conseguir una aplicación ágil y usable.

- **Ventanas modales**

  Para la implementación de las ventanas modales hemos necesitado tener en cuenta cuatro elementos: botón que activa el extensor, extensor del botón para que muestre la ventana, ventana en código ASP.NET (un panel) y código asociado al pulsado del botón que originó la apertura de la ventana modal.

  A continuación mostramos un ejemplo concreto en el que al pulsar el botón *Eliminar todos* se muestra una ventana modal para solicitar la confirmación de la eliminación.

  - Botón que activa el extensor

```
<asp:Button ID="EliminarTodosButton" runat="server"
Text="Eliminar todos" CssClass="boton-link" />
```

  - Extensor del botón para que muestre la ventana

```
<cc1:ModalPopupExtender
      ID="EliminarTodosButton_ModalPopupExtender"
      runat="server" DynamicServicePath="" Enabled="True"
      TargetControlID="EliminarTodosButton"
      PopupControlID="EliminarConsumicionesPanel"
      OkControlID="OkButton"
      CancelControlID="CancelButton"
      BackgroundCssClass="gris-opacidad">
</cc1:ModalPopupExtender>
```

  TargetControlID. Asignamos el extensor al control *EliminarTodosButton*.
  PopUpControlID. Asignamos el panel que se mostrará al pulsar el botón y activar el extensor.
  OkControlID. Botón de confirmación de la ventana modal.
  CancelControlID. Botón de cancelación de la ventana modal.

- Ventana en código ASP.NET (un panel).

```
<asp:Panel ID="EliminarConsumicionesPanel" runat="server"
CssClass="confirmacion-panel" Style="display: none;">
       <div class="titulo-panel-confirmacion">Eliminar
           consumiciones</div>
        <br />
        <div>¿Desea eliminar las consumiciones?</div>
        <br />
        <div>
            <asp:Button ID="OkButton" runat="server"
               Text="Aceptar" CssClass="boton2-2" />
            <asp:Button ID="CancelButton" runat="server"
               Text="Cancelar" CssClass="boton2-2" />
        </div>
</asp:Panel>
```

- Código asociado al pulsado del botón de acción inicial.

```
Protected Sub EliminarTodosButton_Click(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
EliminarTodosButton.Click
        For i = 0 To Me.PedidoGridView.Rows.Count - 1
            EliminarConsumicion(i)
            ActualizarPrecioPedido(i)
        Next
        ActualizarDatos()
End Sub
```

Si en la ventana modal pulsamos el botón asociado a *OkControlID* se ejecutará el código anterior, por el contrario, si pulsamos el asociado a *CancelControlID* la ventana se cerrará y no se ejecutará ninguna acción.

- **Calendarios**

Para la implementación de los calendarios de *pop up* hemos necesitado tres elementos o conceptos: el calendario ASP.NET, un control extensor *pop up* y el código del evento de selección de día.

Seguidamente adjuntamos un ejemplo de la aplicación de estos conceptos en nuestra aplicación.

- Calendario ASP.NET.

```
<asp:Panel ID="CalendarFechaIniPanel" runat="server"
Style="display: none;">
   <asp:UpdatePanel ID="CalendarUpdateFechaIniPanel"
   runat="server" UpdateMode="Conditional">
      <ContentTemplate>
            <asp:Calendar ID="FechaIniCalendar"
                  runat="server" CssClass=”Calendar”>
            </asp:Calendar>
      </ContentTemplate>
    </asp:UpdatePanel>
</asp:Panel>
```

Como se puede ver en el código, el calendario debe estar dentro de un *UpdatePanel,* para que su funcionamiento no implique el refresco de toda la página, como comentábamos en puntos anteriores. A su vez, el *UpdatePanel* debe de estar contenido en un *Panel* estándar de ASP.NET, para poder mostrarlo oculto por defecto.

- Control extensor Pop up.

```
<cc1:PopupControlExtender
      ID="FechaIniCalendar_PopupControlExtender"
      runat="server"
      DynamicServicePath="" Enabled="True"
      TargetControlID="FechaIniTxt"
      PopupControlID="CalendarFechaIniPanel">
</cc1:PopupControlExtender>
```

TargetControlID. Indicamos el control que recibirá el evento de selección de día en el calendario, en dicho control se mostrará el día.
PopUpControlID. Asignamos el panel que se mostrará al pulsar el botón y activar el extensor.

- Código evento selección de día.

```
Protected Sub Calendar1_SelectionChanged(ByVal sender As
      Object, ByVal e As System.EventArgs) Handles
      FechaIniCalendar.SelectionChanged,
      FechaFinCalendar.SelectionChanged

        Dim pce As PopupControlExtender

        pce =
            AjaxControlToolkit.PopupControlExtender.GetProxyF
            orCurrentPopup(Page)
        pce.Commit(CType(sender,
            Calendar).SelectedDate.ToShortDateString)

End Sub
```

Recoge el día seleccionado y lo devuelve al control *TargetControlID*.

PUESTOS DE USUARIO: CLIENTE Y EMPLEADO

La aplicación está preparada para que clientes y empleados se autentiquen en ella, si el puesto de usuario es de cliente veremos que en la interfaz aparece la *Zona clientes* (Figura 4.1.2.1), mientras que si el puesto es de empleado (gerente o camarero) nos aparece *Zona empleados* (Figura 4.1.2.2). Además, si un usuario accede a la aplicación web desde fuera del restaurante visualizará la *Zona clientes web* (Figura 4.1.2.3).

Este control del acceso lo hemos realizado en la clase *variables_globales.vb*, dentro del paquete *App_Code* de nuestro proyecto web. En este fichero guardamos las direcciones IP de los puestos de usuario que pertenecen al restaurante, concretamente, las que pertenecen a puestos de cliente por un lado y las que lo hacen a puestos de empleado por otro. De esta manera, cuando un usuario acceda desde fuera del restaurante, se comprueba que su IP no está registrada y se le muestra la interfaz para cliente web.

Podemos consultar aquí un extracto del fichero comentado.

```
Public Class variables_globales
…
…

   Public Shared ip_empleados As String() = {"84.123.62.89","84.123.62.90"}
   Public Shared ip_clientes As String() = {"84.123.62.91", "84.123.62.92"}
…
…
End Class
```

ACCESO A LA BASE DE DATOS

Resulta importante destacar que el acceso a la base de datos se trata en un proyecto distinto al proyecto web. Se ha utilizado un proyecto de biblioteca de clases para almacenar los archivos que realizan operaciones sobre la base de datos pensando en las siguientes ventajas:

- Tener un código más limpio y claro en la aplicación web.

- Normalmente el código para realizar las consultas básicas sobre la base de datos es muy similar entre tablas, así que copiando y pegando el código y cambiando lo que sea necesario ganamos mucho tiempo. Lo ideal para evitar copiar y pegar sería implementar un programa que te generara por cada tabla un archivo independiente con las operaciones sobre la base de datos o bien utilizar un framework que ya lo haga.

- La aplicación web se adaptará mucho más fácil a futuros cambios.

Así que tenemos una biblioteca de clases formada por tantos ficheros como tablas hay en la base de datos, cada fichero con las operaciones necesarias para la aplicación web. Lo podemos ver en la siguiente imagen donde aparece el explorador de soluciones del proyecto:

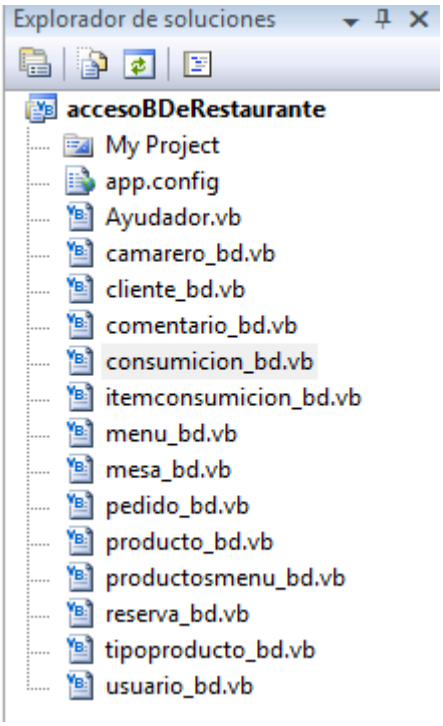


Figura 5.3.3 Explorador de soluciones

Vamos a profundizar un poco más en esto describiendo parte del funcionamiento de una de las clases, por ejemplo *camarero_bd*.

Cada una de las clases se caracterizan por tener dos estructuras declaradas, una que contiene todos los atributos de la tabla y otra sólo con las claves, que sirve por ejemplo para recuperar un registro completo de una tabla, ya que la clave identifica cada fila de una tabla, no necesitamos más.

```
Public Class camarero_bd

    Public Structure camarero_reg
        Dim id As Integer
        Dim nombre As String
        Dim apellido1 As String
        Dim apellido2 As String
        Dim foto As String
    End Structure

    Public Structure camarero_clave
        Dim id As String
    End Structure

    'Métodos
      …
```

Por último, contiene los métodos para consultar o actualizar información en la base de datos. Para que se vea como se utilizan las estructuras vamos a adjuntar el método para recuperar un camarero.

```
Public Shared Function get_camarero(ByVal tmp_camarero_clave
As camarero_clave, ByVal cadena_conexion As String) As
camarero_reg
    Dim conexion As New
            MySqlClient.MySqlConnection(cadena_conexion)
    Dim comando As MySqlClient.MySqlCommand
    Dim lector As MySqlClient.MySqlDataReader = Nothing
    Dim strSQL As String
    Dim tmp_camarero_reg As New camarero_reg

   Try
      strSQL = " SELECT * FROM camarero WHERE id = " &
      tmp_camarero_clave.id & " "

      'Abro la conexion
      conexion.Open()

      comando = New MySqlClient.MySqlCommand(strSQL,
      conexion)
      'Ejecuto la consulta y la guardo
      lector = comando.ExecuteReader()

      'Leo el registro
      Try
         With lector
           If .Read() Then
             If Not (lector Is Nothing) Then
               tmp_camarero_reg.id = lector("id")
               tmp_camarero_reg.nombre = lector("nombre")
               tmp_camarero_reg.apellido1 =
                  lector("apellido1")
               tmp_camarero_reg.apellido2 =
                  lector("apellido2")
               tmp_camarero_reg.foto = lector("foto")
             Else
               tmp_camarero_reg = error_camarero()
             End If
           Else
             tmp_camarero_reg = error_camarero()
           End If
         End With

       Catch ex As Exception
          tmp_camarero_reg = error_camarero()
       End Try

       'Cierro la conexion
       conexion.Close()

  Catch e As Exception
    tmp_camarero_reg = error_camarero()
  End Try

  Return tmp_camarero_reg

End Function
```

Ahora si por ejemplo queremos acceder a la tabla *camarero* en el código de la página de la aplicación web importaremos el archivo correspondiente a esa tabla, en este caso *camarero_bd*, de la siguiente manera:

```
Imports accesoBDeRestaurante.camarero_bd
```

Y llamaremos al método que nos convenga, por ejemplo, si queremos recuperar un camarero escribiremos:

```
' Declaramos el registro de tipo clave y tipo registro
Dim tmp_camarero_clave As camarero_clave
Dim tmp_camarero_reg As camarero_reg
' Asignamos el identificador de camarero al registro de
'clave
tmp_camarero_clave.id = tmp_pedido_reg.FK_camarero
' Obtenemos el camarero
tmp_camarero_reg = get_camarero(tmp_camarero_clave,
cadena_conexion)
```

El nombre de la DLL es “accesoBDeRestaurante.dll” y se encuentra en la carpeta “bin” del proyecto Web.

6. CONCLUSIONES

La principal conclusión que obtengo de la realización de este Proyecto Final de Carrera es la importancia de seguir una metodología de desarrollo correcta, siguiendo fases bien definidas (especificación de requisitos, análisis, diseño, implementación y pruebas), las aprendidas a lo largo de la carrera y en las que se ha incidido tanto en respetar, ya no sólo en el orden, sino en la dedicación a cada una de ellas. A lo largo de mi corta experiencia laboral ya he aprendido que la existencia de diagramas es vital para un correcto desarrollo, sobre todo para el mantenimiento de una aplicación de gran envergadura y, su utilización, en general, en el mundo del software, evitaría muchos contratiempos.

Por otro lado, analizando el proceso de desarrollo y centrándome en la configuración del servidor web, he de decir que he llegado a la conclusión de que combinar ASP.NET con el servidor web Apache no me parece una buena idea. Decidí llevarlo a cabo como un experimento, ya que, ya había desarrollado varias veces con ASP.NET contra el servidor de Microsoft IIS, y consideré que no estaría de más probar sobre Apache, para saber de primera mano cómo funciona. No obstante, la configuración fue complicada y, a medida que fui avanzando en la implementación me encontré con que algunos controles no funcionaban correctamente debido a este hecho y había que de alguna manera "arreglarlos". Conclusión: bajo mi experiencia de este proyecto no es nada recomendable.

Siguiendo con las tecnologías utilizadas, he de decir que nunca había utilizado AJAX, no tenía ningún conocimiento, y, para realizar este PFC he tenido que aprenderlo. La verdad es que la diferencia entre el sitio web sin y con AJAX es muy grande, los tiempos de respuesta son mucho mejores con AJAX y la aplicación se vuelve mucho más ágil y usable.

En cuanto a posibles ampliaciones destacaría ampliar la funcionalidad del cliente web, permitiendo que pueda realizar pedidos con servicio a domicilio.

Para finalizar, he de decir que estoy satisfecha con la realización de un sitio web de estas características, ya que este tipo de sitios están cada vez más extendidos, así que tener experiencia en este campo creo que es altamente positivo.

7. BIBLIOGRAFÍA

- Web oficial de Microsoft: MSDN para ASP.NET.

  http://msdn.microsoft.com/es-es/asp.net/default.aspx

- Blog de ASP.NET, VB.NET y desarrollo web en general

  http://www.blog-de.net/

- Guías sobre CSS de la web del consorcio W3C.

  http://www.w3c.es/Divulgacion/GuiasReferencia/CSS21/

- Web oficial Mercury Mail.

  http://www.pmail.com/

- Artículo sobre instalación de ssl en Apache.

  http://www.symantec.com/connect/es/articles/apache-2-ssltls-step-step-part-2

- Artículo sobre la seguridad en Apache.

  http://www.tufuncion.com/configuracion_apache

- Wikipedia.

  http://www.wikipedia.org/

- Apuntes de asignaturas de las carreras Ingeniería Técnica en Informática e Ingeniería Informática.
  - Creación de Documentos en Hipertexto.
  - Ingeniería del Software de Sistemas.
  - Bases de Datos Relacionales.
  - Diseño de Bases de Datos.
  - Desarrollo de Aplicaciones para Entornos Web.
  - Ingeniería de Requerimientos.
  - Ingeniería de la Programación.
  - Servidores Web.

- Otros proyectos finales de carrera.

  - Sistema para la gestión web del catálogo de productos de una empresa
  - Aplicación de comercio electrónico - Desoras
  - Desarrollo de un sitio web para un camping
  - Gestor de referencias bibliográficas

8. ANEXOS

8.1 ANEXO A: TABLAS DE LA BASE DE DATOS

Importante: las claves principales aparecen en negrita.

Tabla *usuario*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
clave	char(9)	Clave de acceso a la aplicación.
passwd	char(20)	Password de acceso asociado a la clave.
Tipo	char(1)	Tipo de usuario: ▪ "G". Gerente ▪ "C". Cliente ▪ "E". Camarero

Tabla *cliente*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
nombre	char(50)	Nombre del cliente.
telefono	char(9)	Teléfono de contacto del cliente.
apellido1	char(50)	Primer apellido del cliente.
apellido2	char(50)	Segundo apellido del cliente.
email	char(30)	Dirección de correo del cliente.
FK_usuario	int(11)	Clave ajena al atributo id de la tabla *usuario*.

Tabla *pedido*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
fecha	datetime	Fecha de realización del pedido.
precio	decimal(10,2)	Precio total del pedido.
pagado	tinyint(4)	Indica si el pedido ha sido pagado por el cliente. ▪ 0: Pedido sin pagar. ▪ 1: Pedido pagado.
confirmado	tinyint(4)	Indica si el pedido ha sido confirmado por el cliente y ya pueden comenzar a servirle: ▪ 0: Pedido sin confirmar. ▪ 1: Pedido confirmado.

servido	tinyint(4)	Indica si todos las consumiciones del pedido han sido ya servidas: ▪ 0: Quedan consumiciones por servir. ▪ 1: Todo el pedido está servido.
activo	tinyint(4)	Indica si el pedido está todavía abierto, aún no ha pasado al histórico: ▪ 0: Pedido inactivo. ▪ 1: Pedido activo.
FK_Cliente	int(11)	Clave ajena a la tabla *cliente*.
FK_Mesa	int(11)	Clave ajena a la tabla *mesa*.
FK_Camarero	int(11)	Clave ajena a la tabla *camarero.*

Tabla *mesa*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
numero	int (11)	Número de la mesa.
libre	tinyint(4)	Indica si la mesa está libre en el momento actual: ▪ 0: Mesa libre. ▪ 1: Mesa ocupada.
max_personas	int(11)	Máximo número de personas recomendado para la mesa.
comensales	int(11)	Número de comensales que hay en el momento actual en la mesa.
num_mesas	int(11)	Número de mesas que componen la mesa, identificadas por el mismo número.

Tabla *camarero*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
nombre	char(50)	Nombre del camarero.
apellido1	char(50)	Primer apellido del camarero.
apellido2	char(50)	Segundo apellido del camarero.
foto	char(100)	Fotografía en la que aparece el camarero.

Tabla *reserva*

Nombre	**Tipo**	**Descripción**
id	int (11)	Identificador de la tabla.
fecha	datetime	Fecha de realización de la reserva.
finalizada	tinyint(4)	Indica si la reserva ya ha sido efectuada y finalizada:

		▪ 0: Reserva por finalizar. ▪ 1: Reserva finalizada.
FK_Cliente	int(11)	Clave ajena a la tabla *cliente.*
FK_Mesa	int(11)	Clave ajena a la tabla *mesa.*

Tabla *comentario*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
desde	char(50)	Dirección de correo desde la cual se recibe el comentario.
asunto	char(80)	Explicación breve del comentario.
comentario	char(255)	Comentario del usuario.
fecha	datetime	Fecha de envío del comentario.

Tabla *consumicion*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
cantidad	int(11)	Cantidad de ítems de consumición que forman la consumición.
precio	decimal(10,2)	Precio total de la consumición (cantidad * precio del itemconsumicion).
servida	int(11)	Indica la cantidad de consumiciones servidas.
FK_Pedido	int(11)	Clave ajena a la tabla *pedido*.
FK_Itemconsumicion	int(11)	Clave ajena a la tabla *itemconsumicion.*

Tabla *itemconsumicion*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
tipo	char(1)	Tipo del ítem de consumición: ▪ P: Producto ▪ M: Menú
nombre	char(50)	Nombre del ítem de consumición.
descripcion	char(255)	Descripción del ítem de consumición.
precio	decimal(10,2)	Precio del ítem de consumición.

Tabla *menu*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
FK_ItemConsumicion	int (11)	Clave ajena a la tabla *itemconsumicion*.

Tabla *producto*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
FK_TipoProducto	int (11)	Clave ajena a la tabla *tipoproducto*.
FK_ItemConsumicion	int (11)	Clave ajena a la tabla *itemconsumicion*.

Tabla *productosmenu*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
cantidad	int (11)	Cantidad de cada producto en el menú.
FK_producto	int (11)	Clave ajena a la tabla *producto*.
FK_menu	int (11)	Clave ajena a la tabla *menú*.

Tabla *tipoproducto*

Nombre	Tipo	Descripción
id	int (11)	Identificador de la tabla.
nombre	char (30)	Nombre del tipo de producto.

8.2 ANEXO B: CONFIGURACIÓN SERVIDOR WEB APACHE

Con la instalación del paquete XAMPP se obtiene una configuración por defecto del servidor Web Apache, no obstante, esta configuración hay que adecuarla a nuestras necesidades y, además, tenemos que añadir la configuración específica para nuestra aplicación.

Primero de todo, instalamos el plugin para poder utilizar ASP.NET con Apache: ***mod_aspdotnet-2.2.0.2006.*** Se puede descargar de la siguiente dirección:

http://sourceforge.net/projects/mod-aspdotnet/

En segundo lugar, se ha de tener en cuenta que en nuestra versión de Apache tenemos varios ficheros de configuración, no solamente el clásico *httpd.conf,* así que vamos a ir explicando sobre que fichero realizamos cada cambio.

httpd.conf

- Establezco el ServerRoot, la ruta a la instalación de apache.

```
ServerRoot "C:/xampp/apache"
```

- Desactivo los módulos que no vayamos a necesitar, por ejemplo, el módulo para autoindex: *mod_autoindex.so*. Lo comentamos.

- Establezo las propiedades ServerAdmin, ServerName y DocumentRoot.

```
ServerAdmin spuchol02@gmail.com
ServerName nazna1:80
DocumentRoot "C:/xampp/htdocs"
```

- Permito la ejecución de ASP.NET sobre el servidor Apache y doy de alta la aplicación en éste.

```
#asp.net

# Se carga el modulo asp
LoadModule aspdotnet_module "modules/mod_aspdotnet.so"
AddHandler asp.net asax ascx ashx asmx aspx axd config cs
csproj licx rem resources resx soap vb vbproj vsdisco webinfo

<IfModule mod_aspdotnet.cpp>
  AspNet Files Directories Virtual
  # Monta la aplicación ASP.NET en /eRestaurante
  AspNetMount /eRestaurante "c:/xampp/htdocs/eRestaurante"
  #/eRestaurante es el alias de ASP.NET para ejecutarla
  #"c:/xampp/htdocs/eRestaurante es la ubicación del proyecto
  # eRestaurante
```

```
  # Mapea las peticiones de /eRestaurante a los ficheros
  Alias /eRestaurante "c:/xampp/htdocs/eRestaurante"
  #Para acceder a la aplicacion: http://nazna1/eRestaurante
  # Permitimos a todos los scripts asp ejecutarse en
  #/eRestaurante
  <Directory "c:/xampp/htdocs/eRestaurante">
    Options FollowSymlinks
    Order allow,deny
    Allow from all
    # Se establecen las paginas por defecto
    DirectoryIndex index.htm index.aspx
  </Directory>
  AliasMatch
  /aspnet_client/system_web/(\d+)_(\d+)_(\d+)_(\d+)/(.*)
   "C:/Windows/Microsoft.NET/Framework/v$1.$2.$3/ASP.NETClientF
   iles/$4"
  <Directory
   "C:/Windows/Microsoft.NET/Framework/v*/ASP.NETClientFiles">
    SSLRequireSSL
    Options FollowSymlinks
    Order allow,deny
    Allow from all
  </Directory>

</IfModule>
#asp.net
```

- Habilito el módulo para utilizar ssl.

```
LoadModule ssl_module modules/mod_ssl.so
```

httpd-default.conf

- Desactivo que se muestre la versión de apache, de mi sistema operativo, de asp, etc. De esta manera le damos mayor seguridad a nuestro servidor, ya que,

es más fácil realizar una intrusión si se conoce la versión instalada, puesto que cada versión tiene unos agujeros de seguridad, muchos de ellos conocidos.

```
ServerTokens Prod
ServerSignature Off
```

httpd-ssl.conf

- Establezo la escucha para ssl en el puerto 443.

```
Listen 443
```

- Establezco *DocumentRoot*, *ServerName* y *ServerAdmin* para el host virtual.

```
DocumentRoot "C:/xampp/htdocs/eRestaurante/"
ServerName nazna1:443
ServerAdmin spuchol02@gmail.com
```

- Activamos el ssl para este host virtual.

```
SSLEngine on
```

- Establecemos la ubicación del certificado de seguridad del servidor y de la clave privada de este certificado (ver Anexo 8.3).

```
SSLCertificateFile conf/ssl.crt/server_my_cert.crt
SSLCertificateKeyFile conf/ssl.key/server_my_cert.key
```

- Añadimos una condición que haga que las páginas que no queramos que utilicen el protocolo HTTPS funcionen utilizando HTTP: *index.aspx* y las contenidas en la carpeta *secciones*.

```
RewriteEngine On
RewriteCond %{HTTPS} on
RewriteRule (./index.*)http://%{HTTP_HOST}%{REQUEST_URI}
RewriteRule (./secciones/*)http://%{HTTP_HOST}%{REQUEST_URI}
```

8.3 ANEXO C: CONFIGURACIÓN CERTIFICADOS SEGURIDAD

Para realizar comunicaciones seguras, hay que encriptar los mensajes enviados. Se utiliza una técnica denominada clave asimétrica. Esto consiste en que una parte genera dos claves: Una privada que se guarda de un modo muy seguro y una pública que se da a todo el mundo. Estas claves están matemáticamente relacionadas, de tal modo que lo que se codifica con una clave, solo se desencripta con la otra.

En el mundo Web, este tercero de confianza se denomina CA (entidad certificadora) la cual, previa presentación de documentación notarial, proporciona un fichero que asegura al cliente Web que el servidor es quien dice ser.

Este fichero es lo que se denomina un certificado digital (la entidad certificadora emite y firma un certificado digital). Este certificado digital, contiene, además de información que identifica del servidor, una clave (la clave pública) del servidor Web. Si se encripta algo con la clave pública solo nuestro servidor puede decodificarla.

Así que cuando un usuario desde un navegador se conecta a un Web seguro, el servidor le manda el certificado. Como este certificado ha sido emitido por la entidad certificadora, el navegador verifica si es correcto o no, y si es así, utiliza la clave pública de nuestro servidor Web para encriptar mensajes y comunicarse con el servidor.

Generación del certificado

En primer lugar, abrimos una consola de Windows y nos situamos en la carpeta *bin* del programa *OpenSSL*. En nuestro proyecto hemos utilizado un certificado auto-firmado, aunque sólo es recomendable para entornos de prueba o Intranets. Para crear el par de claves privada y pública, y el certificado auto-firmado ejecutamos el siguiente comando:

```
openssl req -new -x509 -days 365 -sha1 -newkey rsa:1024 -nodes -keyout
server_my_cert.key -out server_my_cert.crt
```

Este comando crea un nuevo (-new) certificado (-x509) que es válido por un año (-days 365) y que está firmado utilizando el algoritmo SHA1 (-sha1). La clave privada RSA tiene 1024 bits (-newkey rsa:1024), y no está protegida por ningún *passphase* (-nodes). El certificado y el par de claves pública y privada, se crean en los ficheros *server_my_cert.crt* y *server_my_cert.key* (-out server.crt -keyout server.key).

Los ficheros generados los situamos en las carpetas del servidor apache *conf/ssl.key* y *conf/ssl.crt*.

Instalación del certificado

El siguiente paso es instalar el certificado en cada navegador donde vayamos a utilizarlo. En nuestro caso: Mozilla e Internet Explorer.

- **Instalación en Mozilla**

1. Abrimos el navegador Mozilla, vamos a *Opciones* y seleccionamos *Ver certificados*.

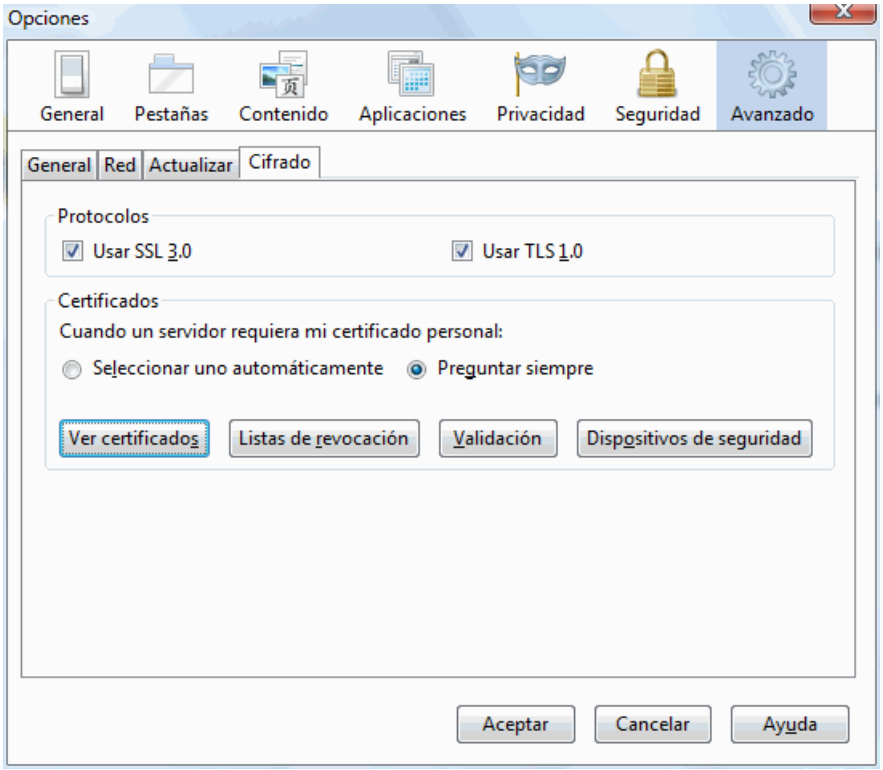


Figura 8.3.1 Certificado Mozilla paso 1

2. Vamos a la pestaña *Autoridades* y seleccionamos *Importar*.

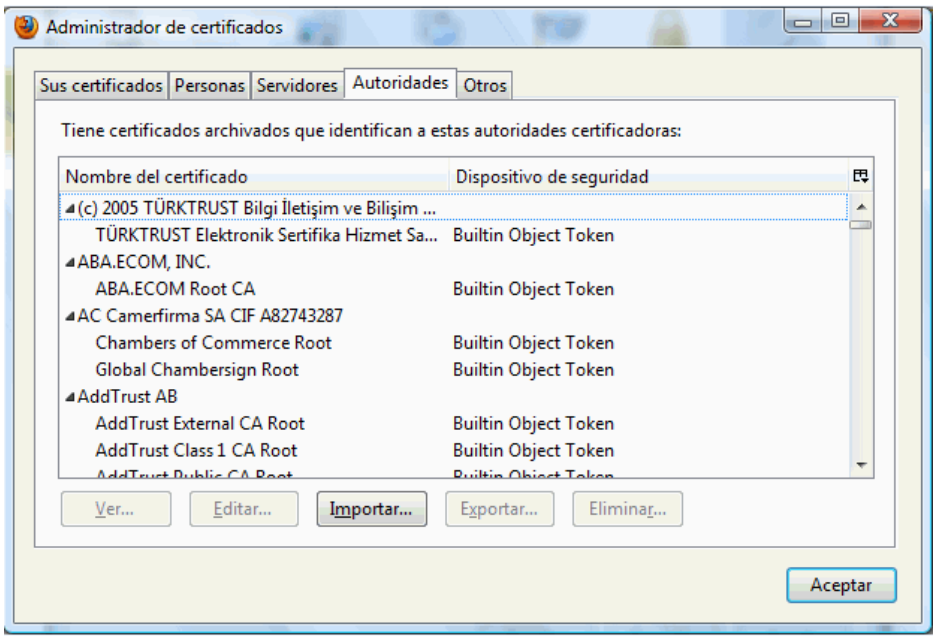


Figura 8.3.2 Certificado Mozilla paso 2

3. Seleccionamos en el sistema de ficheros el certificado creado anteriormente.

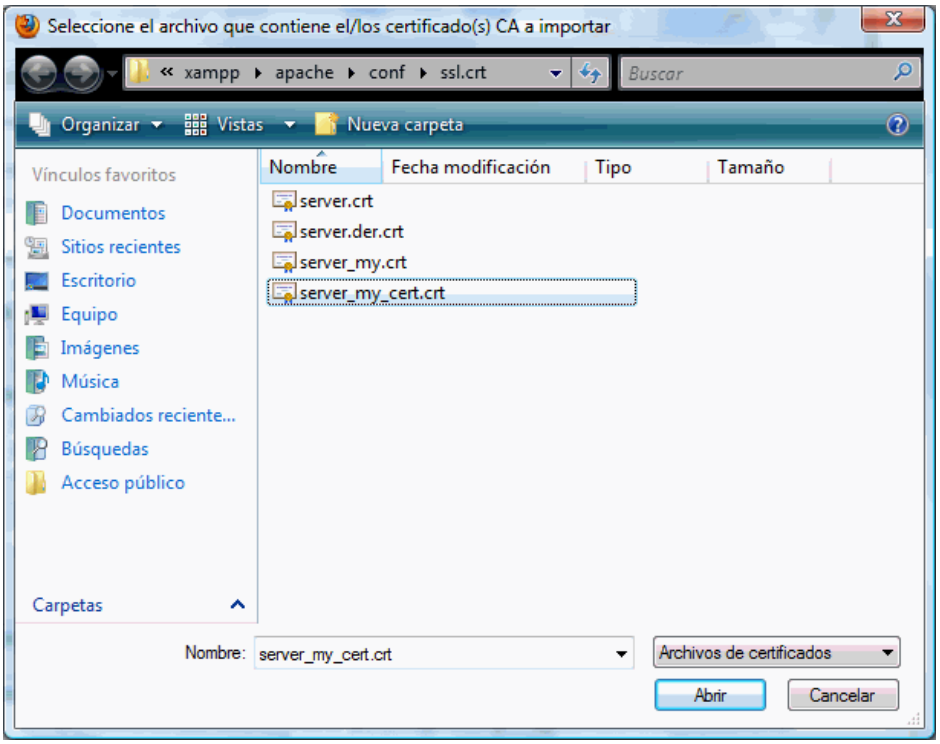


Figura 8.3.3 Certificado Mozilla paso 3

4. Seleccionamos confiar en la Autoridad Certificadora (CA) para identificar sitios web.

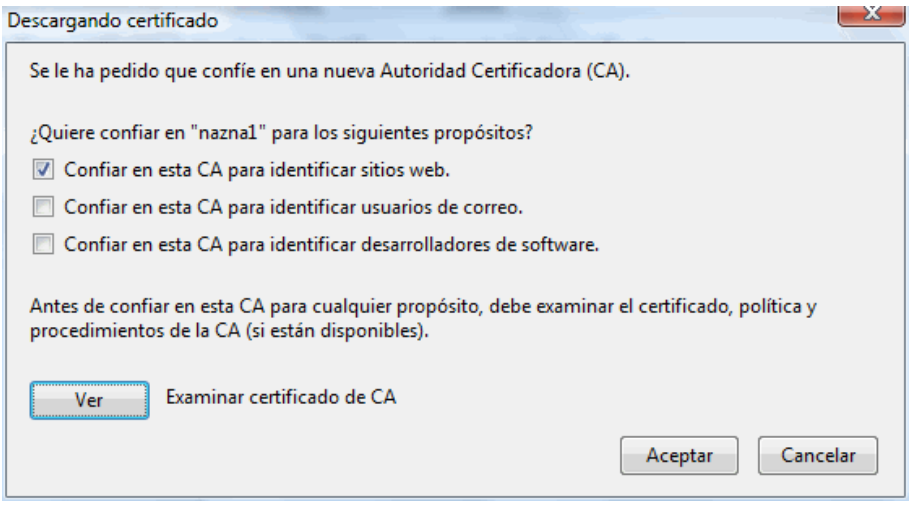


Figura 8.3.4 Certificado Mozilla paso 4

5. Ya tenemos instalado el certificado en el navegador Mozilla Firefox.

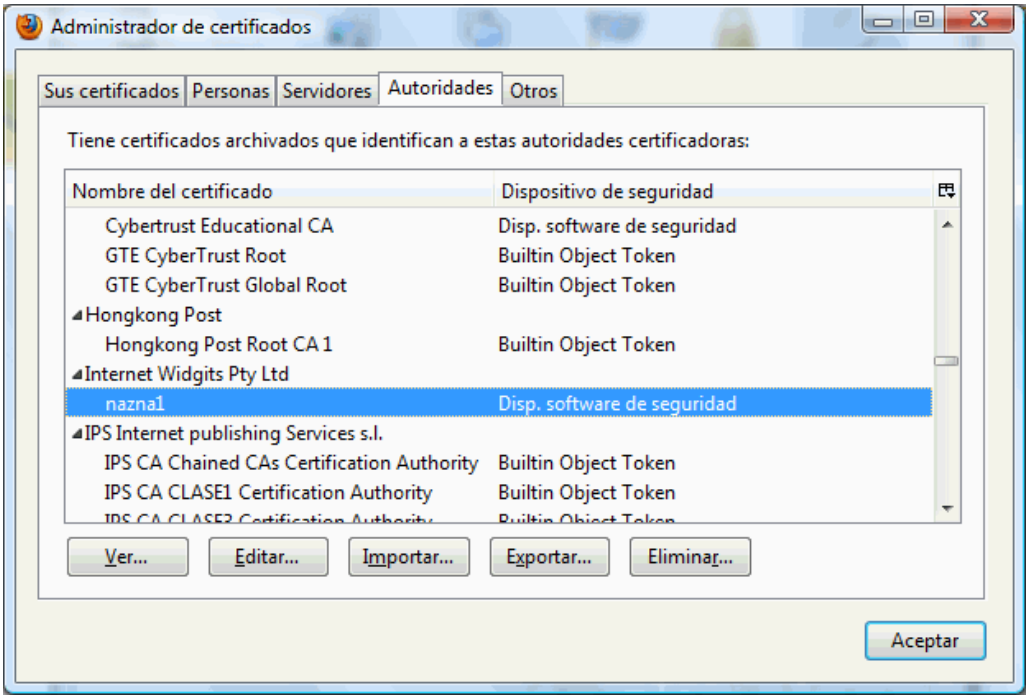


Figura 8.3.5 Certificado Mozilla paso 5

- **Instalación en Internet Explorer**

1. Vamos al certificado, hacemos doble clic sobre él y seleccionamos *Instalar certificado*.

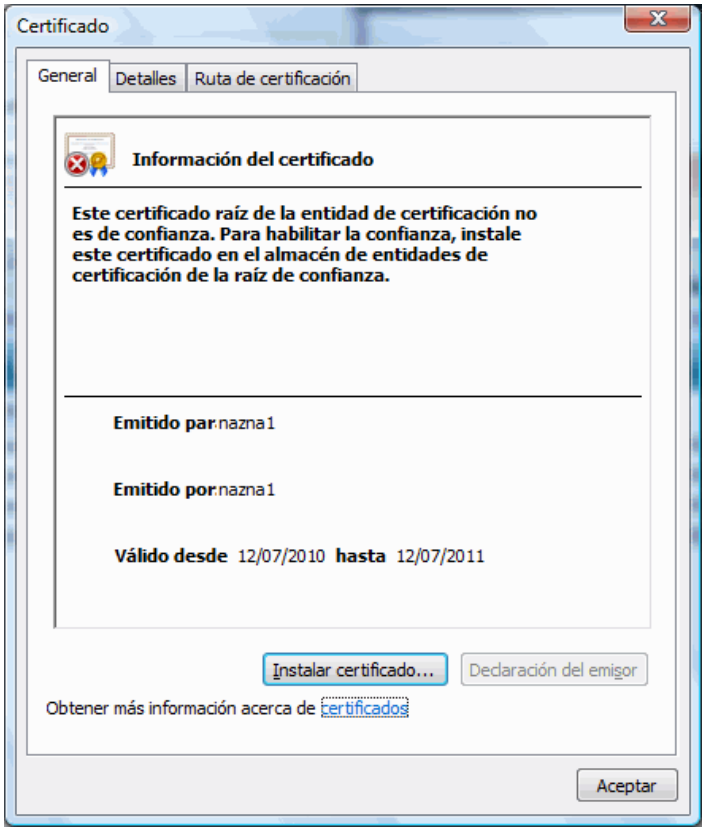


Figura 8.3.6 Certificado IE paso 1

2. Se abre el asistente para importación de certificados. Siguiente.

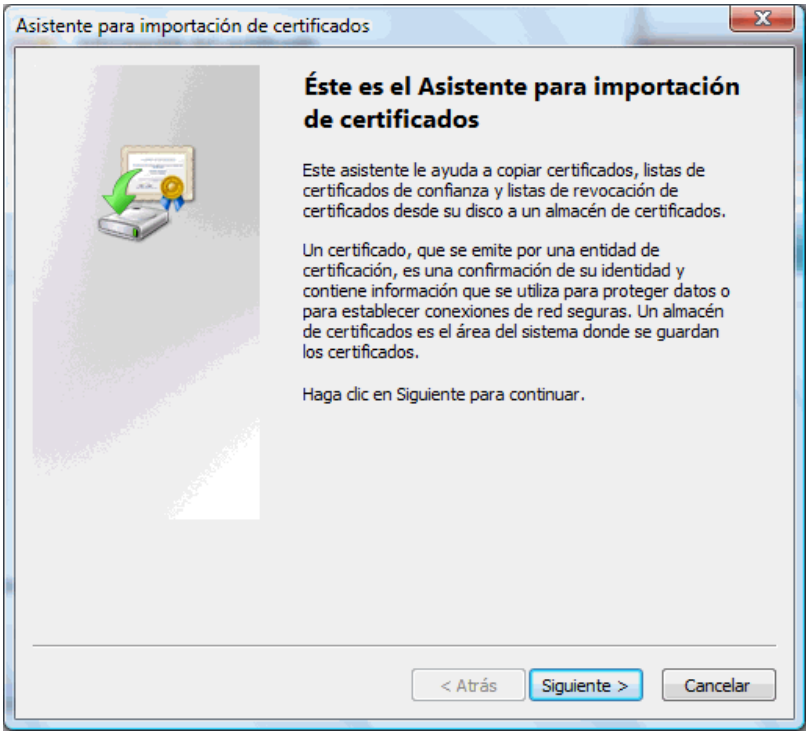


Figura 8.3.7 Certificado IE paso 2

3. Elegimos colocar el certificado en el almacén de certificados *Entidades de certificación raíz de confianza*.

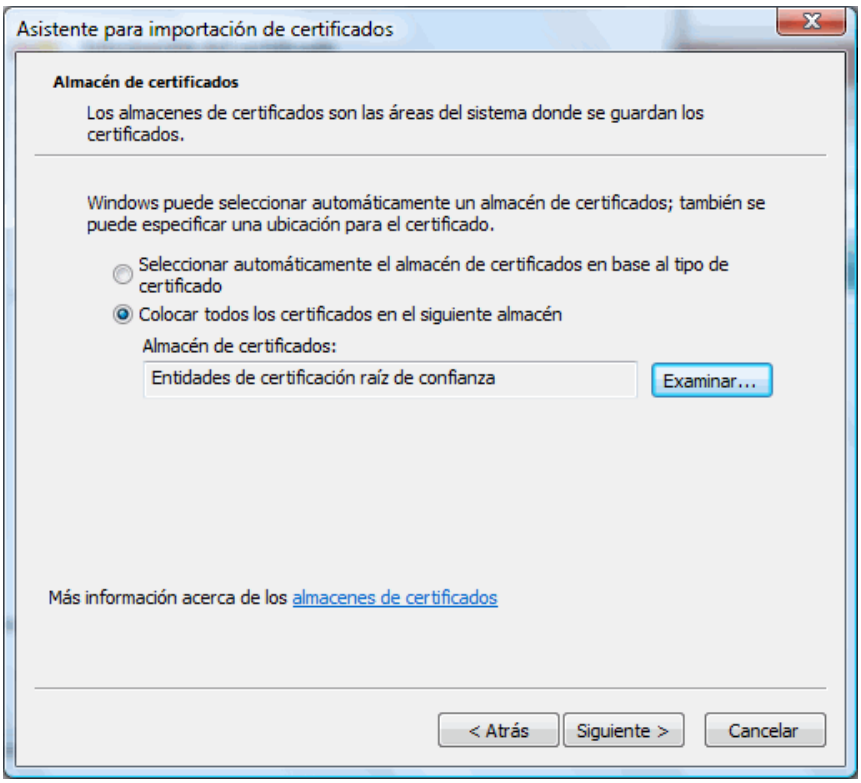
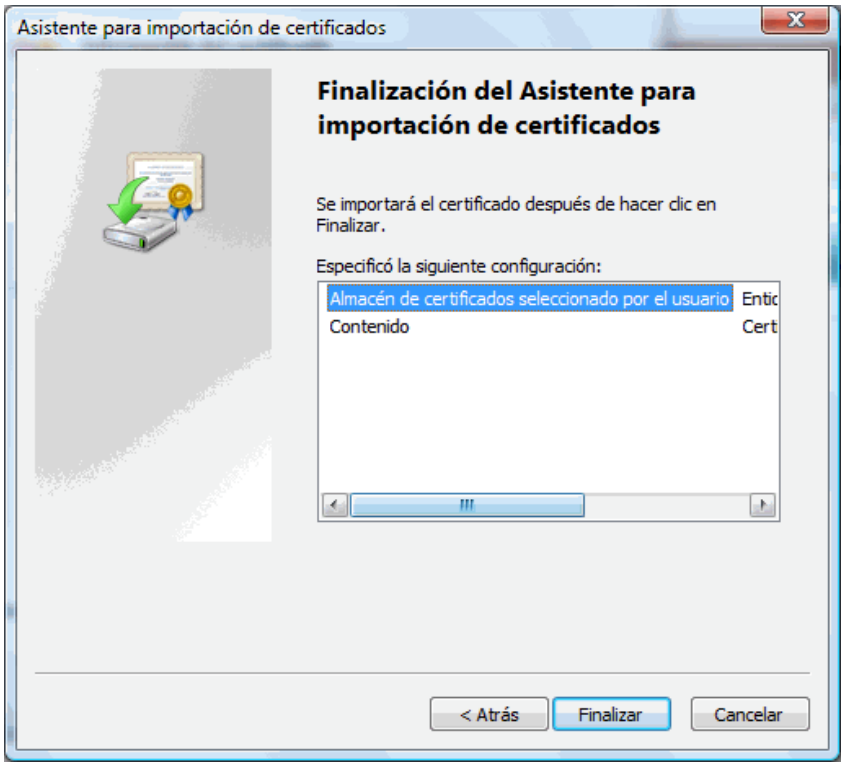


Figura 8.3.8 Certificado IE paso 3

4. Pulsamos en finalizar para instalar el certificado y aceptamos el aviso.



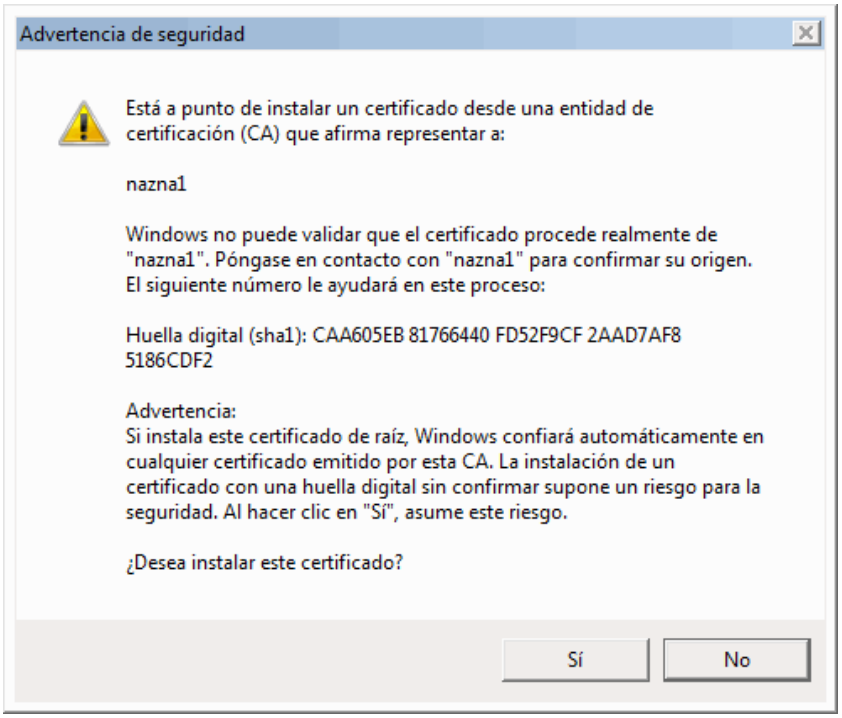


Figura 8.3.9 Certificado IE paso 4

5. El certificado ya está importado e instalado.

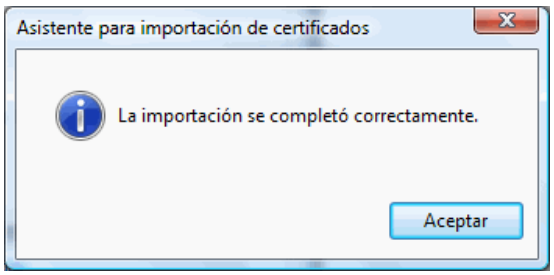


Figura 8.3.10 Importación correcta

Si ahora accedemos a cualquiera de las páginas de la aplicación web que utilizan HTTPS accederemos de manera segura. Por ejemplo, la página de autenticación de usuario:

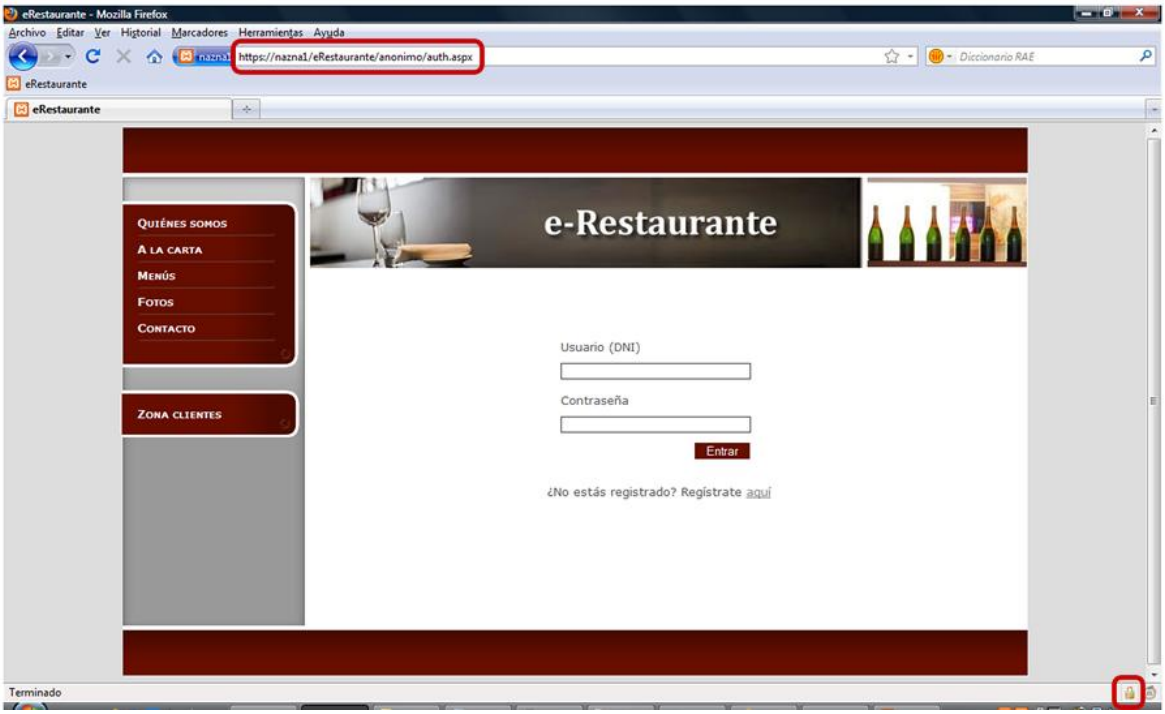
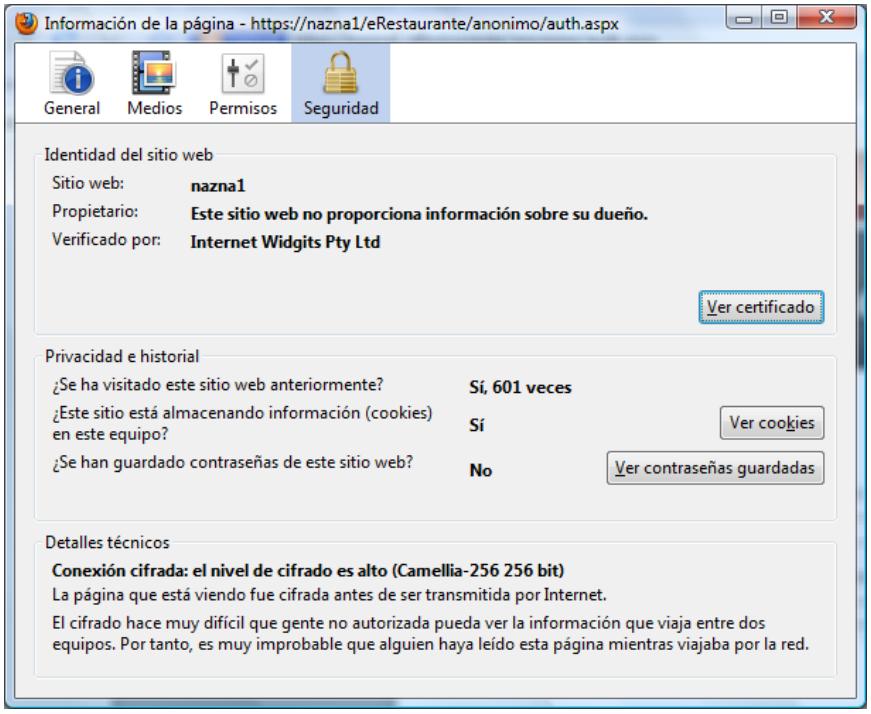


Figura 8.3.11 Página autenticación con HTTPS.

Si hacemos doble clic en el candado accederemos a la información de seguridad y podemos ver que el certificado creado está instalado, y se nos permite verlo.



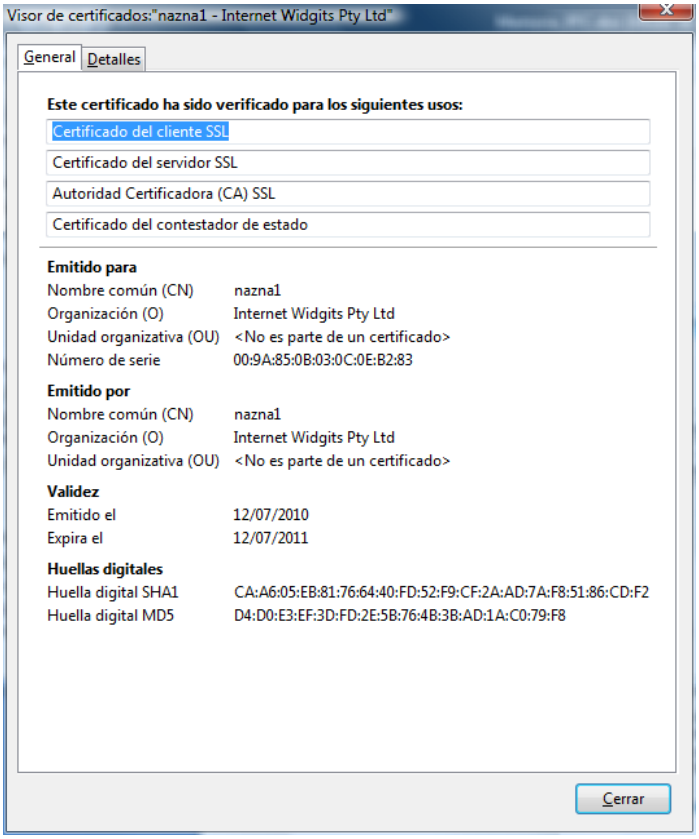


Figura 8.3.12 Certificado en nuestra aplicación

8.4 ANEXO D: MANUAL DEL USUARIO GERENTE

Una vez autenticado en la aplicación, tiene acceso desde el menú situado a la izquierda, en la parte inferior, a la gestión de: tipos de producto, productos, menús y empleados.

- **TIPOS DE PRODUCTO**

  Muestra un listado con todos los tipos de producto disponibles.

Figura 8.4.1 Tipos de producto

Acciones posibles:

- **Nuevo.** Creación de un nuevo tipo de producto. Se debe proporcionar el nombre.

Figura 8.4.2 Nuevo tipo de producto

- **Modificar.** Modificación de un tipo de producto ya existente. Se debe seleccionar uno.

Figura 8.4.3 Modificar/Detalles tipo de producto

Se permite también añadir productos al tipo de producto.

- **Eliminar.** Se elimina uno o varios tipos de productos al seleccionarlos en la lista.

- **Detalles.** Acceso a información detallada del tipo de producto seleccionado en modo sólo lectura (ver Figura 8.4.3).

- **PRODUCTOS**

Muestra un listado con todos los productos disponibles y permite filtrarlos por tipo de producto.

Figura 8.4.4 Productos

Acciones posibles:

- **Cambiar de tipo de producto.** Podemos cambiar de tipo de producto y ver sus productos asociados desplegando y seleccionando el tipo de producto deseado en el combo que aparece en la parte superior de la pantalla.

- **Nuevo.** Creación de un nuevo producto. Se debe introducir: nombre, descripción, precio y tipo de producto.

Figura 8.4.5 Nuevo producto

- **Modificar.** Modificación de los datos de un producto seleccionado (Figura 8.4.5)

- **Eliminar.** Permite borrar uno o varios productos seleccionados en la tabla.

- **Detalles.** Visualiza los datos del producto en modo sólo lectura (Figura 8.4.5).

- **MENÚS**

  Muestra un listado con todos los menús que hay disponibles.

Figura 8.4.6 Menús

Acciones posibles:

- **Nuevo.** Crear un menú nuevo. Se debe introducir nombre, descripción y precio. Además, seleccionamos los productos que forman parte del menú utilizando la flecha hacia la izquierda del listado de productos (1).

Figura 8.4.7 Nuevo menú

- **Modificar.** Modificar un menú existente. Hay que seleccionar uno de la lista de menús. Se permite la modificación de los mismos datos que al crear el menú.

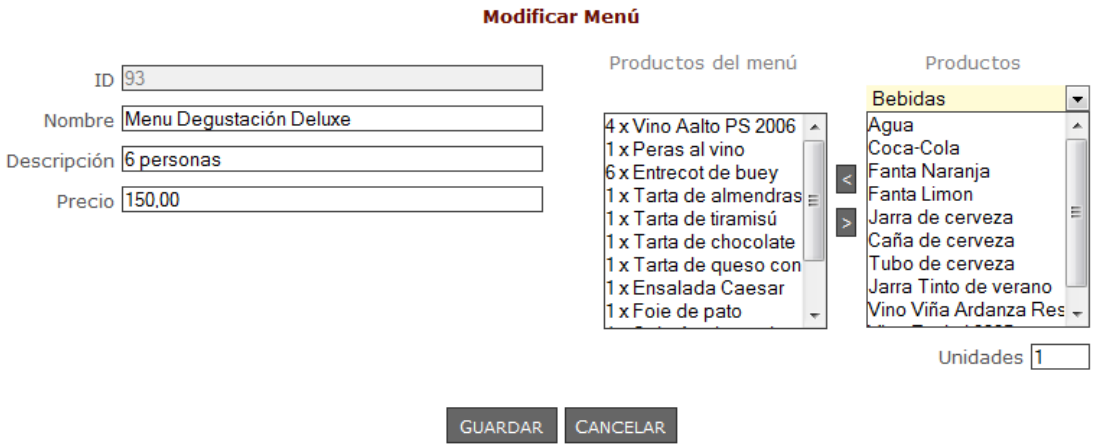


Figura 8.4.8 Modificar/Detalles menú.

- **Eliminar.** Borra uno o varios menús seleccionados.

- **Detalles.** Muestra los datos del menú seleccionado sin opción a modificarlos (Figura 8.4.8).

- **EMPLEADOS**

Muestra un listado con todos los empleados disponibles.

Figura 8.4.9 Empleados

Acciones posibles:

- **Nuevo.** Se registra un empleado nuevo en la aplicación. Se deben introducir los siguientes datos: nombre, apellidos y fotografía.

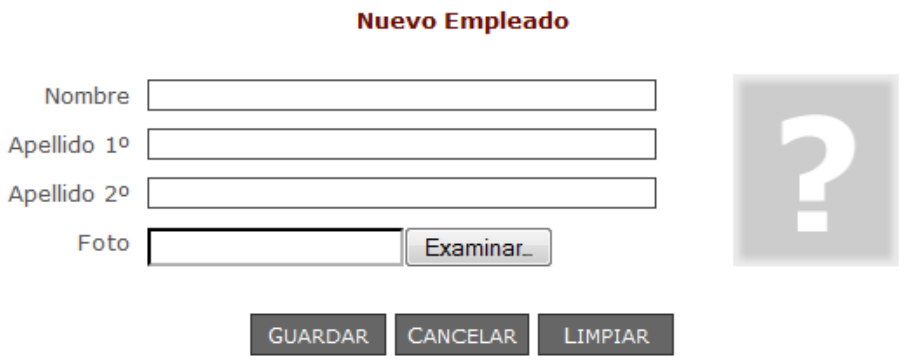


Figura 8.4.10 Nuevo empleado

- **Modificar.** Se muestran los datos guardados del empleado y se permite modificarlos.

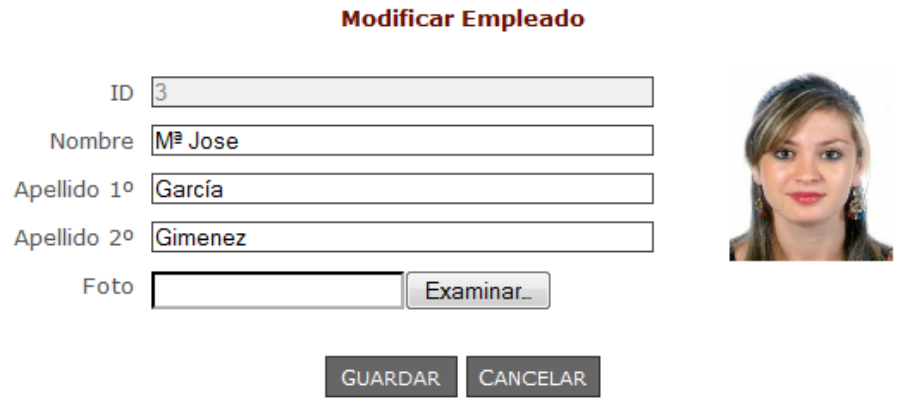


Figura 8.4.11 Modificar/Detalles empleado

- **Eliminar.** Se borra uno o varios empleados seleccionados.

- **Detalles.** Se visualizan los datos del empleado pero sin opción de modificarlos.

8.5 ANEXO E: MANUAL DEL USUARIO CAMARERO

Una vez autenticado en la aplicación, tiene acceso desde el menú situado a la izquierda, en la parte inferior, a: estado de las mesas, pedidos, clientes, comentarios e historial de pedidos.

- **ESTADO MESAS**

  Muestra la ocupación del restaurante en un día y una hora.

Figura 8.5.1 Estado de las mesas

Al cambiar el día y la hora cambiará la ocupación del restaurante al día y hora seleccionados.

Mesa ocupada

Mesa reservada

Mesa libre

Si no seleccionamos ninguna mesa solamente veremos el botón *Nueva*. Por el contrario, si seleccionamos cualquier mesa se mostrarán todos los botones que aparecen en la Figura 8.5.1 y que se describen a continuación.

- **Nueva mesa.** Creación de una nueva mesa en el restaurante. Se debe introducir: número de mesa, número de mesas que forman la mesa y máximo número de comensales.

Figura 8.5.2 Nueva mesa

- **Modificar.** Se abre una ventana con los datos de la mesa y se pueden modificar.

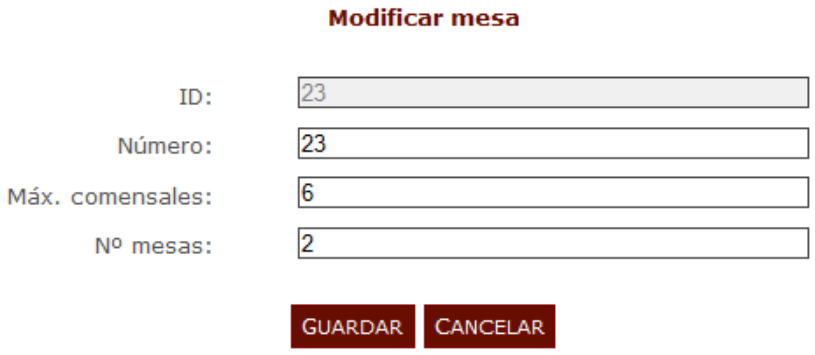


Figura 8.5.3 Modificar mesa

- **Eliminar.** Se elimina una mesa seleccionada siempre que no tenga pedidos asignados.

- **Detalles.** Se muestran los datos de la mesa seleccionada y del pedido si tuviera uno asignado y activo, y permite gestionarlo.

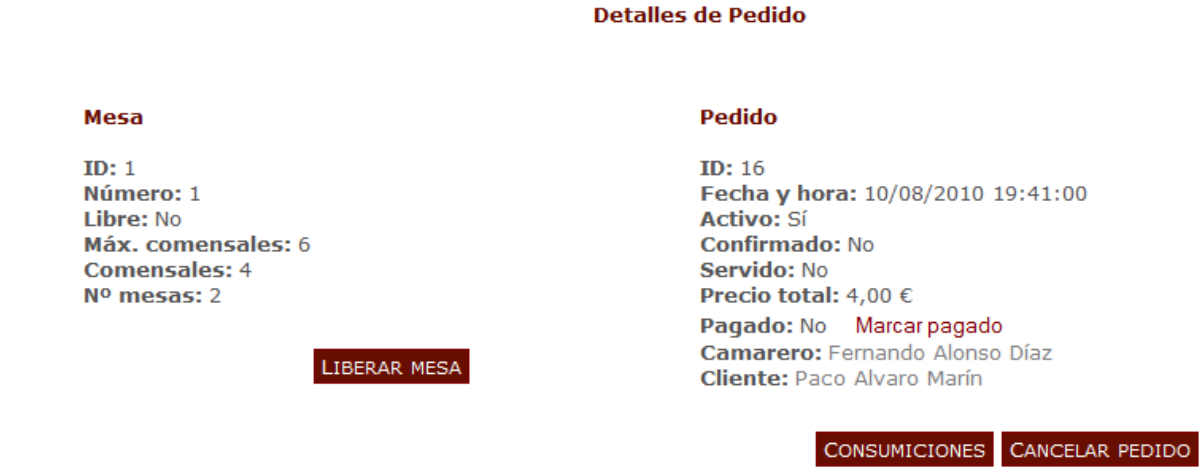


Figura 8.5.4 Detalles mesa/pedido

Desde esta pantalla podemos:

- Liberar mesa. Marcamos la mesa como libre.
- Marcar pagado. El pedido se da por pagado.
- Consumiciones. Accedemos a la pantalla de consumiciones del pedido.

Figura 8.5.5 Gestión de consumiciones

Añadir un ítem de consumición más al pedido
Eliminar un ítem de consumición menos del pedido
Marca como servido un ítem de consumición.
Resta un ítem de consumición a los ítems servidos.
Elimina una consumición.

Añadir productos. Se abre una ventana para añadir más productos a la lista.

- Cancelar pedido. El pedido pasa a estado inactivo.

- **Asignar a cliente.** Se asigna una mesa a un cliente y a un camarero. Esta acción hay que realizarla a la llegada del cliente al restaurante. Al realizar la

asignación se le creará un pedido activo y vacío al cliente, de manera que cuando llegue a su mesa ya podrá empezar a pedir.

Figura 8.5.6 Asignar mesa a cliente

Al escribir el nombre del cliente se desplegará un listado para seleccionar uno de ellos. Lo mismo sucede con el camarero.

- Registro rápido. Si el cliente no está registrado en el sistema, porque es la primera vez que visita el restaurante haremos un registro rápido haciendo clic en este enlace.

- **PEDIDOS**

Muestra un listado con los pedidos activos del restaurante.

Figura 8.5.7 Pedidos

Acciones posibles:

- **Detalles de pedido.** Muestra información del pedido seleccionado y de su mesa asignada (Figura 8.5.4). Ver las acciones posibles en *Estado Mesas*).

- **CLIENTES**

  Muestra un listado con los clientes registrados en el restaurante.

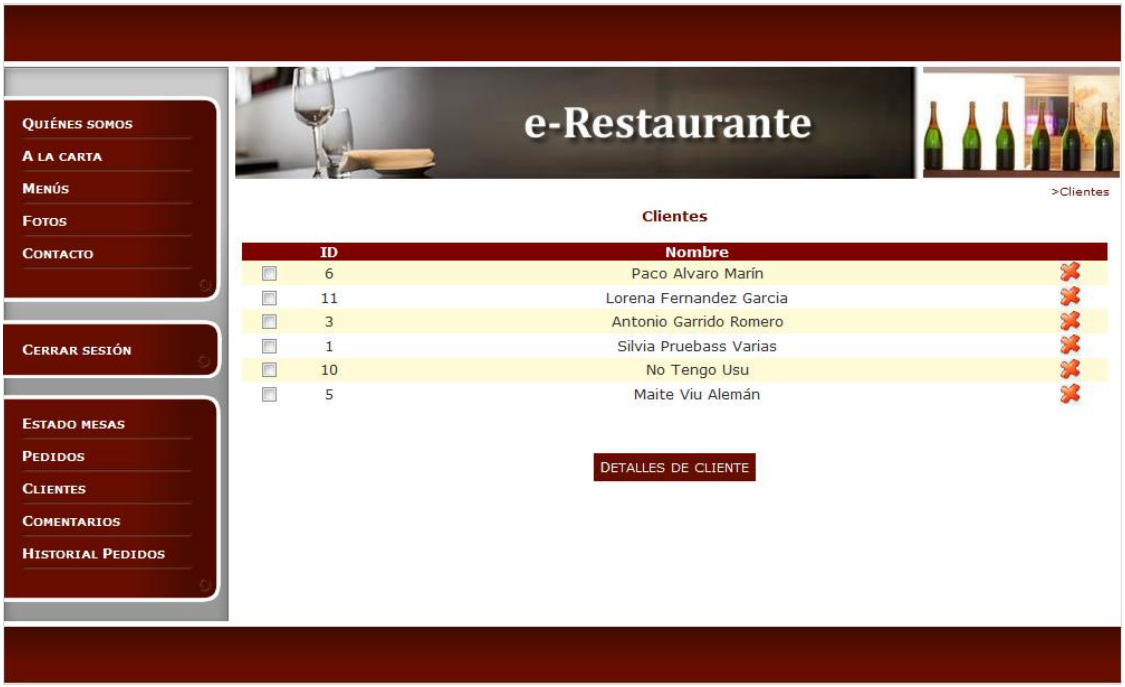


Figura 8.5.8 Clientes

Acciones posibles:

- **Detalles de cliente.** Muestra información del cliente seleccionado y permite modificar sus datos y su contraseña.

Figura 8.5.9 Detalles de cliente

Desde esta pantalla podemos:

- Modificar. Aparece el mismo formulario de la figura pero con los campos habilitados y la opción de guardarlos.

- Cambiar contraseña. Aparece un formulario para realizar el cambio de contraseña. Se solicita introducir la antigua y la nueva.

- **COMENTARIOS**

Muestra un listado con todos los comentarios recibidos en el restaurante.

Figura 8.5.10 Comentarios

Acciones posibles:

- **Ver comentario.** Accedemos al comentario. Hay que pulsar en el asunto del comentario que se quiera consultar.

Figura 8.5.11 Detalles de comentario

Desde esta pantalla podemos:

- Responder. Se carga la pantalla para introducir la respuesta que queremos dar al cliente y poder enviarla.

Figura 8.5.12 Responder comentario

- **Eliminar.** Este botón está pensado para eliminar varios comentarios seleccionándolos previamente en el listado.

- Elimina un comentario.

- **HISTORIAL PEDIDOS**

  Se visualizan los pedidos que ya no están activos (ya han sido pagados) filtrados por fecha, y se permite consultar sus detalles.

Figura 8.5.13 Historial de pedidos

Acciones posibles:

- **Buscar pedidos.** Seleccionamos fecha de inicio y de fin haciendo clic en los campos de texto *Fecha Inicio* y *Fecha Fin*, y pulsamos el botón *Buscar*.

- **Detalles de pedido.** Seleccionamos un pedido de la tabla de pedidos y pulsamos en el botón Detalles de Pedido para acceder a los datos del mismo.

Figura 8.5.14 Detalles de pedido

8.6 ANEXO F: MANUAL DEL USUARIO CLIENTE

Una vez autenticado en la aplicación, tiene acceso desde el menú situado a la izquierda, en la parte inferior, a: su pedido actual, su historial de pedidos y sus datos personales.

- **MI PEDIDO**

  Es la pantalla principal del cliente. Le permite ver su pedido en todo momento y, por supuesto, modificarlo, añadiendo o quitando productos, confirmándolo, etc.

Figura 8.6.1 Mi Pedido

Acciones posibles:

- **Añadir productos.** Se abre una ventana para añadir más productos a la lista.

Figura 8.6.2 Añadir productos

  En esta pantalla podemos filtrar por tipo de producto, utilizando los botones superiores. Se añaden productos al pedido marcándolos en la tabla y pulsando el botón *Añadir al pedido*.

- **Detalles de pedido.** Se accede a información detallada del pedido actual, en modo lectura: fecha, mesa, camarero, consumiciones y precio.

Figura 8.6.3 Detalles de pedido

- **Confirmar pedido.** Confirma el pedido, ya no se podrá modificar. Se le notificará automáticamente a los camareros, apareciendo como confirmado, para que pongan en marcha el pedido.

- **Pagar.** Se podrá pagar una vez el pedido se confirme. Si se pulsa en pagar se solicita si se desea realizar el pago en efectivo o con tarjeta. En caso de seleccionar *en efectivo* se informará al camarero automáticamente de que lleve la cuenta a esa mesa. Si se selecciona *con tarjeta* se solicitarán los siguientes datos: nombre del titular, número de tarjeta y fecha de caducidad.

- **Eliminar todos.** Se eliminan todas las consumiciones.

 Se añade un ítem de consumición más del producto o menú.

 Se resta un ítem de consumición del producto o menú.

 Se eliminan todos los ítems de consumición del producto o menú.

- **HISTORIAL DE PEDIDOS**

Muestra un listado con los pedidos anteriores del cliente.

Figura 8.6.4 Historial de pedidos

Acciones posibles:

- **Detalles de pedido.** Se accede a información detallada del pedido seleccionado, en modo lectura: fecha, mesa, camarero, consumiciones y precio (Figura 8.6.3).

- **MIS DATOS PERSONALES.**

  Muestra los datos personales del cliente: nombre, apellidos, DNI, teléfono y dirección de correo, y permite modificarlos. Permite cambiar la contraseña.

Figura 8.6.5 Mis datos personales

Acciones posibles:

- **Modificar.** Modificación de los datos mostrados en la figura 8.6.5.

- **Cambiar contraseña.** Se accede a una pantalla de cambio de contraseña. Hay que introducir la contraseña antigua y la nueva.

Figura 8.6.6 Cambiar contraseña

8.7 ANEXO G: MANUAL DEL USUARIO CLIENTE WEB

Una vez autenticado en la aplicación, tiene acceso desde el menú situado a la izquierda, en la parte inferior, a: sus reservas, su historial de pedidos y sus datos personales.

- **RESERVAS**

  Se muestra un listado con las reservas del cliente conectado a la aplicación y se permite crear nuevas.

Figura 8.7.1 Reservas

Acciones posibles:

- **Eliminar.** Pulsando en  se elimina la reserva.
- **Nueva reserva.** Se accede a la página para reservar una mesa.

Figura 8.7.2 Nueva reserva

Por defecto, se muestra la ocupación del restaurante para el siguiente horario de reservas del día en que nos encontremos. Cambiando el día y/o la hora se actualizará la ocupación.

 Mesa ocupada

Mesa reservada

Mesa libre

Desde esta pantalla podemos:

- Reservar. Seleccionar un día y pulsar el botón *Reservar*. Se crea una reserva para la mesa y el día y la hora seleccionadas.

- **HISTORIAL DE PEDIDOS.**

  El funcionamiento es exactamente el mismo que para el cliente estándar (ver Anexo E).

- **MIS DATOS PERSONALES**

  El funcionamiento es exactamente el mismo que para el cliente estándar (ver Anexo E).