| **P.P. Integración de Sistemas**  Proyecto “Gestión de Pedidos” | DESCRIPCIÓN DEL PROYECTO  El siguiente proyecto, se basa en la integración de los conceptos aprendidos a lo largo de la carrera de sistemas, dictada por el instituto ESBA. Aquí se abordará el desarrollo de una solución en una problemática de un entorno real.  Jefe de Proyecto:  Leonardo Soto  Alumnos:   | Guillaumet; Ignacio | DNI: 35196199 | | --- | --- | | Leto; Bruno | DNI: 41664006 | | Roser; Ulises | DNI: 27259356 | | Vazquez; Yoel | DNI: 35322669 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

***INDICE***

* **PRESENTACION 6**

**Propuesta de Servicios 7**

¿Quiénes somos? 7

¿Qué hacemos? 7

¿Cómo trabajamos? 7

**ETAPA DE ANALISIS 8**

**PROPOSICION DE ALTERNATIVAS 9**

Propuesta N. º 1 “Selección de Escuelas” – Ingresos de niños al jardín de infantes 10

Descripción: 10

Beneficios/Soluciones posibles: 10

Propuesta N. º 2 “Cienfuegos”- Sistema para control de ventas de revendedores. 10

Descripción: 10

Beneficios/Soluciones posibles: 10

Propuesta N. º3: “Che Miranda” – Sistema de Delivery de comidas (\*) 11

Descripción: 11

El aplicativo tendrá: 11

Beneficios/Soluciones posibles: 11

**PROYECTO ELEGIDO 12**

Informe del Proyecto Seleccionado 13

Propuesta: “Che Miranda” – Gestion de entregas 13

Descripción: 13

El aplicativo tendrá: 13

Beneficios/Soluciones posibles: 14

**Informe escrito 15**

Giro de actividades: principal y secundario 15

Descripción de Tareas 15

Antecedentes informáticos para considerar sobre la empresa elegida 16

Técnicas y Herramientas de relevamiento general a emplear 16

**ESTUDIO PRELIMINAR 17**

Informe preliminar 18

**INFORME DE LA EMPRESA ELEGIDA 19**

Objetivos 20

Ventajas 20

Desventajas 20

**Prediagnóstico. Aspectos de control general. Soluciones generales. 20**

Aspectos generales de la aplicación: 20

La aplicación cumplirá los siguientes requisitos: 21

**PLANEAMIENTO DEL PROYECTO 22**

**DISTRIBUCION DE TAREAS 23**

Distribución de tareas entre los componentes del Equipo de Trabajo 24

**GRAFICO DE ACTIVIDADES 26**

Gantt Planificado 27

**RELEVAMIENTO DETALLADO 28**

Informe detallado sobre estructura de la empresa 29

Técnicas y Herramientas para acceder a la información. 29

Observación personal y directa 29

**ENTREVISTAS** 30

Entrevista Dueño 31

Entrevista Delivery 31

Entrevista Encargado 32

**DATOS DE LA EMPRESA** 34

Objetivos y Procesos generales 35

**PLANO DE UBICACION** 36

Plano de ubicación 37

**DIAGRAMA DE INSTALACIONES** 38

Diagrama de instalaciones 39

**ORGANIGRAMA** 40

Organigrama real de la Empresa: 41

**FORMULARIOS** 42

Principales formularios utilizados 43

Formas de aprovechamiento de los formularios principales 43

**CURSOGRAMAS** 44

Gráficos de los principales circuitos administrativos actuales. 45

Cursograma de Venta a Clientes: 45

**PROCESOS EXISTENTES 47**

Descripción Detallada de Procedimientos existentes en el sistema actual: 48

Metodología Clásica: Informe escrito. 48

Solicitud a Proveedores 48

Venta a Clientes 48

Metodología Estructurada: Diagramas de contexto y Flujo de datos lógicos 48

Evaluación y control. Enumeración de las principales modificaciones a realizar. 48

**EVALUACION Y DIAGNOSTICO 50**

Estudio integral de la información obtenida a través del Análisis elaborado 51

Objetivo del proyecto: 51

Objetivos del informe: 51

Proponemos realizar estas modificaciones para optimizar los procesos: 51

Plan de trabajo propuesto para el diseño del nuevo sistema informático 52

**ETAPA DE DISEÑO 53**

**DISEÑO GLOBAL 54**

Plan de trabajo propuesto: pasos a seguir. Objetivo del sistema a diseñar 55

Sistema de gestión de pedidos. 55

**SISTEMA Y SUBSISTEMAS** 56

Representación gráfica del sistema y sus subsistemas. 57

Sistema GENERAL 57

Subsistema de Pedidos 57

Subsistema de Logueo 58

Subsistema de Clientes 58

Subsistema de Gestión 59

Descripción de entradas, salidas y archivos 59

Entradas 59

Salidas 61

Archivos 62

**PROCESOS GLOBALES GENERADOS** 64

Procesos globales que se generan 65

**Nombre**: Ingreso de usuarios 65

**Nombre**: Creación de pedido 65

**Nombre**: Estado de los pedidos 65

**Nombre**: Informes 65

**MEDIOS DE PROCESAMIENTO** 66

Medios de procesamiento: Análisis del grado de necesidad de comunicación. 67

**PRESUPUESTO** 68

Presupuesto: Costos y beneficios aproximados del Proyecto. Recursos involucrados. Período de recuperación de la inversión. 69

**ESTUDIO DE FACTIBILIDAD** 72

Estudio de Factibilidad 73

Factibilidad técnica: 73

Factibilidad económica: 73

Factibilidad operativa: 74

**DISEÑO DETALLADO 75**

**ORGANIGRAMA PROPUESTO** 76

Organigrama propuesto 77

**FORMULARIOS PROPUESTOS 78**

Diseño de formularios propuestos 79

Inicio de sesión de usuario 79

Menú Principal 80

Pedidos 81

Crear Pedido 81

Gestión de Pedido 82

Clientes 83

Gestión 84

**CURSOGRAMAS PROPUESTOS 85**

Cursogramas propuestos 86

**ARBOL DE PANTALLAS 88**

Árbol de Pantallas 89

**DETALLE BASE DE DATOS 90**

Descripción detallada de archivos o bases de datos, en hojas de especificaciones 91

**DETALLE SALIDAS IMPRESAS 93**

Descripción detallada de salidas impresas en hojas de especificaciones 94

**DETALLE NUEVOS PROCESOS 96**

Descripción detallada de nuevos procesos intervinientes: 97

Metodología Clásica: Informe escrito 97

Solicitud a Proveedores 97

Venta a Clientes (Delivery) 97

Metodología Estructurada: Diagramas de Contexto y de Flujo de Datos Lógicos. 97

**CARACTERISTICAS DEL LENGUAJE 98**

Características del Lenguaje de Programación a utilizar y sus ventajas. Informe escrito 99

Paradigma Orientado a objetos 99

Framework: Visual Studio 2019 99

C# 100

Microsoft SQL Server 100

**CODIFICACIÓN 102**

Codificación: avance en el desarrollo de Programas de Computación 103

CAPA DE DATOS 103

CAPA DE NEGOCIOS 135

CAPA DE USUARIO 180

SENTENCIAS SQL 242

**DICCIONARIO DE DATOS 256**

Diccionario de datos: 257

**MANUAL DE PROCEDIMIENTO 260**

Manual de Procedimiento: 261

Inicio de Sesión 261

Menú Principal 262

Pedidos 263

Crear Pedido 263

Gestión de Pedido 264

Clientes 265

Gestión 266

Informes 266

Productos 267

Personal 267

**ETAPA DE IMPLEMENTACIÓN 268**

**PLANEAMIENTO DE PUESTA EN MARCHA 269**

Planeamiento: Informe correspondiente acerca del método a emplear para la puesta en marcha del sistema. 270

Especificaciones para el funcionamiento del sistema: 270

**COMERCIALIZACION DEL SISTEMA 271**

Posibilidades de comercialización del Sistema diseñado. 272

Servicios adicionales al sistema propuesto: 272

**GRAFICO DE ACTIVIDADES 273**

Gantt FINAL 274

***GRACIAS* 275**

# **PRESENTACIÓN**





## Propuesta de Servicios

*Presentación escrita del equipo de trabajo y servicios que brinda a los usuarios*

### ¿Quiénes somos?

“GUILEROVA Solutions”, somos una empresa joven, que brinda asesoría y soluciones informáticas a PYMES que requieran informatizar y optimizar sus actividades.

Nuestro grupo de trabajo consta de especialistas, que llevan a cabo el análisis, diseño e implementación de sistemas.

### ¿Qué hacemos?

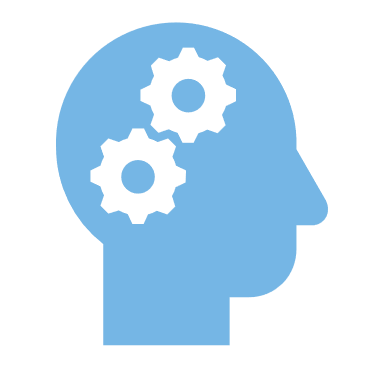
Utilizamos el conocimiento aplicado de metodologías actuales, con la finalidad de simplificar las tareas cotidianas a nuestros clientes y hacer una propuesta innovadora.

### ¿Cómo trabajamos?

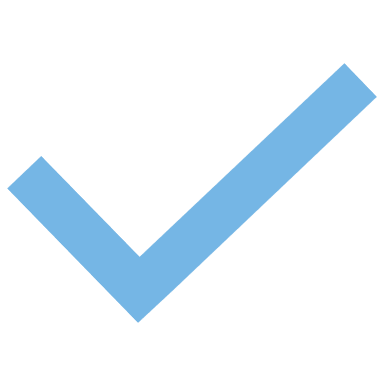
1. Relevamos las necesidades del cliente con un feedback continuo.
2. Analizamos la problemática actual.
3. Desarrollamos un software que cumpla los requerimientos del cliente.
4. Acompañamos al cliente durante el desarrollo e implementación del nuevo sistema.
5. Entregamos manual de uso y capacitaciones.
6. Realizamos soporte Postventa en caso de ser necesario.
7. Realizamos nuevas mejoras, upgrade del sistema.

El establecimiento luego de implantada la solución propuesta, se beneficiará de un mejor uso de la información y gestión de los departamentos que lo componen, particularmente del área de delivery, la cual se verá reflejada en una mejor atención al cliente, y un impacto positivo en el ejercicio de sus tareas. Creemos que nuestro aporte es una mejora sustancial a su empresa y agradecemos que nos haya elegido.

# **ETAPA DE ANALISIS**



## **PROPOSICION DE ALTERNATIVAS**



Organización General

Proposición de Alternativas

*Objetivos generales*

**Propuestas evaluadas: 3**

### Propuesta N. º 1 “Selección de Escuelas” – Ingresos de niños al jardín de infantes

#### Descripción:

Creación de un software capaz de encontrar las instituciones educativas más adecuadas, basándose en diferentes filtros que el usuario pueda seleccionar. Se ingresaría como origen una dirección, el programa mostraría en pantalla las instituciones más cercanas (con un radio determinado por el usuario, en KM por ejemplo).

Se podría filtrar por:

* Secciones disponibles (Salas / grados)
* Turnos disponibles
* Tipo de institución (publica / privada)

El resultado mostrado en pantalla seria, una ficha o listado de las instituciones filtradas, indicando datos como nombre de la escuela, contacto de directivos, dirección, mails, teléfonos, medios de transportes cercanos.

#### Beneficios/Soluciones posibles:

* Facilitar la búsqueda de los padres al momento de elegir el colegio donde ingresaran sus hijos.

### Propuesta N. º 2 “Cienfuegos”- Sistema para control de ventas de revendedores.

#### Descripción:

Creación de un software que tenga como fin, llevar a cabo un control minucioso de las comisiones de los vendedores de la multinacional Cienfuegos, que están distribuidos en las distintas provincias del país, como así también en el resto de Latinoamérica en un periodo a determinar por la empresa.

#### Beneficios/Soluciones posibles:

* Mejorar el sistema de trabajo actual, dando rápido acceso a información referida al sector.
* Facilitar el control de la cantidad de pedidos depositados en cuenta bancaria por cada vendedor.
* Visualización de indicadores diarios y mensuales, que muestran la suma de las ventas.
* Brindar mayor control que permita ver la cantidad de ventas realizada por cada vendedor.

### Propuesta N. º3: “Che Miranda” – Sistema de Delivery de comidas (\*)

#### Descripción:

Se propone crear un software que tenga como fin, la gestión de los pedidos de un establecimiento gastronómico, haciendo foco en la contabilidad de dichos pedidos en un periodo determinado, tanto diario, como mensual. También se deberá discriminar por cadete la cantidad de pedidos que realiza cada uno.

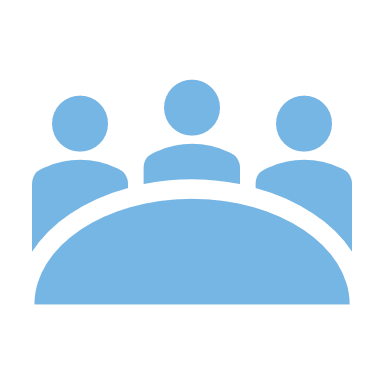
#### El aplicativo tendrá:

1. Un formulario para acceder a una agenda, y ver la disponibilidad de los cadetes en la semana, con distribución en 2 turnos (según relevo).
2. Una consulta (reporte /informe) que permita ver la cantidad de pedidos que llevó cada cadete.
3. Se podrá ver la cantidad de ventas realizadas, en forma diaria y mensuales.
4. Un módulo de quejas que guarde el historial de los clientes, con comentarios respecto a la atención recibida por parte de los cadetes.

#### Beneficios/Soluciones posibles:

* Automatización del trabajo manual, sistematizando varios de sus procedimientos realizadas por el área de delivery.
* Facilitar el control de la cantidad de pedidos realizados por cada cadente.
* Visualización de indicadores diarios y mensuales, que muestren la suma de pedidos netos.
* Mayor control sobre información de los clientes
* Obtención de un mejor control de los pedidos.

## **PROYECTO ELEGIDO**



### Informe del Proyecto Seleccionado

*Justificación*

Se han analizado todas las alternativas, y hemos acordado que este proyecto, se trata de la mejor opción, puesto que el establecimiento gastronómico “Che Miranda”, es de fácil acceso a la hora de relevar su forma de trabajo y por los siguientes puntos:

1. Tenemos la posibilidad de implementar el sistema en una empresa real y beneficiar de forma directa a un colega del equipo, que desempeña tareas en el lugar.
2. Comparando con otras propuestas, nuestro equipo de trabajo se encuentra mejor preparado para abordar el desarrollo del proyecto de una forma ágil.
3. Debido a que uno de los integrantes del equipo forma parte del proyecto relevado, tenemos fácil acceso a la información necesaria.
4. Contamos con una buena predisposición del cliente el cual se encuentra interesado en tener una mejor trazabilidad y gestión, optimizando el área de delivery.
5. En caso de que el sistema satisfaga las necesidades de nuestro cliente y el mismo quede operativo, contamos con la posibilidad de venderlo y a su vez realizar servicios de mantenimiento.
6. Consideramos que este trabajo nos generara un nuevo desafío como profesionales en la rama de programación.

### Propuesta: “Che Miranda” – Gestion de entregas

#### Descripción:

Se propone crear un software que tenga como fin, la gestión de los pedidos de un establecimiento gastronómico, haciendo foco en la contabilidad de dichos pedidos en un periodo determinado, tanto diario, como mensual. También se generará un seguimiento del pedido desde su inicialización hasta su fin ( Etapas del pedido : Inicializado / Delivery / Finalizando o Cancelado )

#### El aplicativo tendrá:

1. Una forma ágil de crear y gestionar los pedidos, con el objetivo de optimizar los tiempos del encargado y en consiguiente del negocio.
2. Una lista de clientes con su historial de pedidos correspondiente.
3. Una consulta (reporte /informe) que permita ver la cantidad de pedidos vendidos.
4. Se podrá ver los productos mas vendidos.

#### Beneficios/Soluciones posibles:

1. Automatización del trabajo manual, sistematizando varios de sus procedimientos realizadas por el área de delivery.
2. Facilitar el control de los pedidos entregados por cada cadete.
3. Visualización de indicadores diarios y mensuales, que muestren la suma de pedidos netos.
4. Mayor control sobre información de los clientes
5. Obtención de un mejor control de los pedidos.
6. Generación automática de la comanda.

## Informe escrito

*Contenido*

### Giro de actividades: principal y secundario

“Che Miranda” fue inaugurado en 2015 con el objetivo de brindar un servicio gastronómico de calidad, ofreciendo dentro de su menú principal la pizza elaborada en horno a leña, variedad de empanadas, calzones, picadas, desayunos, meriendas y la mejor cerveza artesanal de la zona.

### Descripción de Tareas

Para el mejor funcionamiento del restaurante, deberá de requerir una óptima y completa planificación de las tareas efectuadas. Todas de manera coordinada, para hacer frente a los diversos requisitos.

Tareas principales:

• **Cocina**: Es el área encargada de la elaboración de los platos, su preparación, el control de los alimentos y stock.

* Cocinero: realiza la confección de recetas que incluyen, pastas, carnes, minutas, postres.
* Maestro pizzero: realiza la elaboración de pizzas, empanadas.

• **Mozos**: Recepción, ubicación y toma de pedidos del cliente. Preparación de la mesas y cobro del servicio brindado.

• **Delivery**: Se encarga de la entrega de pedidos que fueron solicitados de una forma eficaz, teniendo en cuenta la mejor ruta, según las distancias de los domicilios.

• **Encargado**: Tomar los pedidos y valorizarlos. Control de calidad sobre los productos que proceden del área de cocina. Observar que los sectores cumplan con sus tareas de forma óptima. Arqueo de caja diario y pago de salario diario a cadetes.

• **Compras**: lleva el control de la reposición eficiente de alimentos y cualquier tipo de materiales que se necesiten para correcto funcionamiento del establecimiento.

• **Limpieza:** se encargada de cumplir con el aseo del establecimiento, realizando tareas tales como: el retiro de residuos, baños, cocina, entre otros.

• **Dueño del local:** gestión de personal, contabilidad, facturación, finanzas, etc.

• **Armado y Embalaje:** se encarga del empaquetado de los pedidos a entregar.

### Antecedentes informáticos para considerar sobre la empresa elegida

El establecimiento relevado, carece de un sistema informático que auxilie en las tareas diarias, se detallan a continuación algunos hitos:

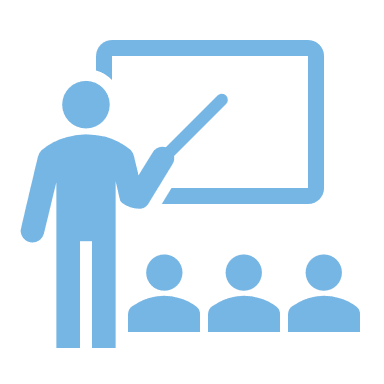
1. El listado de precios se encuentra en papel, la actualización de esto es manual.
2. La toma de pedidos es con lápiz y papel.
3. Los cadetes figuran en una planilla que se actualiza de forma manual, lápiz y papel como formato base.
4. El inventario de la mercadería no lleva control de registros.

El pago de los sueldos es manual y se anota en un bibliorato.

### Técnicas y Herramientas de relevamiento general a emplear

1. Análisis de los procesos actuales.
2. Observación de forma directa de los servicios empleados en el establecimiento, por un colega que trabaja en el sitio.
3. Se realizaron entrevistas virtuales a los encargados del lugar y dueño, con el propósito de recolectar y registrar la información necesaria para el proyecto.
4. Investigación de requisitos faltantes, analizando otros softwares disponibles en el mercado, que cumplan con el requisito planteado.

## **ESTUDIO PRELIMINAR**



ESTUDIO PRELIMINAR

### Informe preliminar

Departamento de análisis de sistemas.

Señores Directivos de “Che Miranda”,

De nuestra consideración:

Según la entrevista realizada el día 05/09/20 en el local comercial, hemos detectado diversas problemáticas a verificar, en lo que respecta al sistema que vigente:

Por el análisis que hicimos, el sistema actual si bien cumple con los requisitos para el funcionamiento de la empresa, pero consideramos que la informatización de este resultaría beneficiosa, optimizando el tiempo de cada proceso de delivery.

Los puntos clave que se van a mejorar en la empresa con la implementación de este sistema, que aumentarán la eficiencia de las tareas y objetivos propuestos por la compañía, serán:

* Automatización del trabajo manual, sistematizando varios de sus procedimientos realizadas por el área de delivery.
* Facilitar el control de la cantidad de pedidos realizados por cada cadente.
* Visualización de indicadores diarios y mensuales, que muestren la suma de pedidos netos.
* SMayor control sobre información de los clientes
* Obtención de un mejor control de los pedidos.

La expuestas anteriormente, son las propuestas principales que marcaran la base de nuestro trabajo y desarrollo del sistema. La cuales se encuentran sujetas a plausibles modificaciones, en base a las recomendaciones que el cliente quiera complementar, manteniendo la comunicación abierta con el mismo.

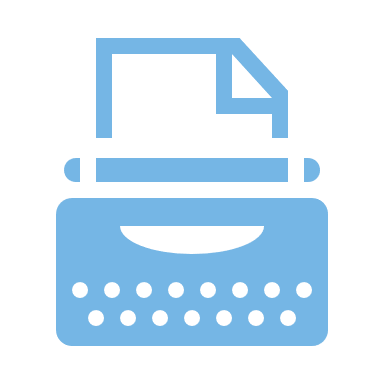
Nuestro equipo de relevamiento en el lugar recomienda la posibilidad de informatizar los procesos existentes, para de esta forma obtener una mayor calidad de información y control sobre el circuito de entregas.

Este software una vez implementado, ayudara a resolver de manera eficiente el área de entregas puerta a puerta, control de tareas, reportes, e indicadores que permitan consultarse para la toma de decisiones.

A su entera disposición.

Atentamente. Guilerova Solutions

## **INFORME DE LA EMPRESA ELEGIDA**



Informe general acerca de la Empresa seleccionada:

*Objetivos, ventajas y desventajas globales acerca del funcionamiento del sistema.*

### Objetivos

* Mejorar la visibilidad, ya que las tareas de delivery son expuesta en un único lugar virtual.
* Visibilizar reportes que el cliente ( Restaurante ) necesite.
* Impulsar la gestión administrativa del área tratada.
* Presentar una imagen profesional.
* Mejorar el servicio al cliente, evitando esperas entre los pedidos.
* Automatizar el control de pedidos.
* Mejorar el servicio actual al cliente agilizando los tiempos de resolución de cada pedido.
* Modernizar su sistema de información con las mejores tecnologías actuales.

### Ventajas

* Mayor organización y comunicación entre los empleados.
* Predisposición a nuevos desafíos.
* Buena predisposición de los propietarios a crecer en el mercado.
* Contar con una PC que permitirá ejecutar el sistema a implementar.

### Desventajas

* Desconocimiento del alcance de las tecnologías actuales.
* Procesos manuales que ocasionan pérdidas de tiempo.   
  Ej.
  + Falta de visión de la gestión de pedidos.
  + Lápiz y papel para la toma de pedidos.

## Prediagnóstico. Aspectos de control general. Soluciones generales.

Desde Guilerova Solutions, planteamos una solución para que su empresa gastronómica pueda seguir creciendo y tener una experiencia de entrega de pedidos optima.

Dicha solución buscará agilizar los procedimientos internos del negocio, reducir los tiempos y carga de trabajo del encargado.

### Aspectos generales de la aplicación:

* El sistema estará basado en el lenguaje de programación C# del framework .NET.
* El frontend estará basado en interfaces graficas que ofrece el lenguaje de programación C#, y cada una contará con un diseño de acuerdo con la orientación o temática del negocio.
* El backend, el programa contara con conexión a un sistema gestor de base de datos relacional.
* La arquitectura del software será de 3 capas (Presentación, Lógica y Base de datos)
* Microsoft SQL Server será el gestor de base de datos elegido.
* Entorno amigable e intuitivo para mejorar la experiencia de usuario.

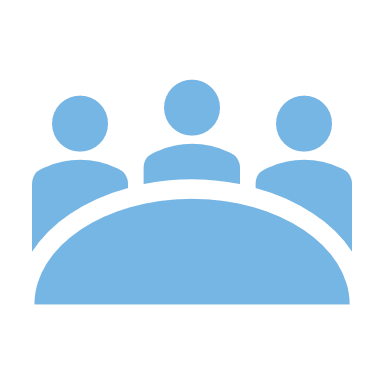
#### La aplicación cumplirá los siguientes requisitos:

* 1. Permitirá asignarle un cadete a cada pedido.
  2. Existirán distintos perfiles de acuerdo con los roles actuales del restaurante:
     1. Encargado y Dueño.
  3. Visualizará listado de pedidos a entregar con su cadete, así como también el valor del estado que puede tomar cada uno de los pedidos.
  4. Tendrá la opción de buscar un cliente existente o añadir un nuevo cliente.
  5. Tendrá la posibilidad de buscar por información básica como:
     1. Teléfono del cliente ó por Nombre.
  6. Podremos acceder a toda la información relacionada de un determinado cliente.
  7. Podremos crear un nuevo pedido y agregar información del cliente.
  8. Al crear un nuevo pedido podrán añadir los productos que están clasificados por categorías.
  9. Desde la sesión de Dueño, se podrá acceder a la sección “Gestion”, en la cual se podrán ver INFORMES, PRODUCTOS y PERSONAL.

Por ejemplo, podremos:

* + - Ver el monto vendido de los “Pedidos de HOY” y “Pedidos por MES” .
    - Podremos ver los productos más vendidos, por: “Productos de HOY” y “Productos por MES” .
    - Se podrá realizar el ABM (alta , baja y modificación) de Productos.
    - Se podrá realizar el ABM (alta , baja y modificación) de Usuarios.
    - Se podrá realizar el ABM (alta , baja y modificación) de Cadetes.

## **PLANEAMIENTO DEL PROYECTO**



## **DISTRIBUCIÓN DE TAREAS**

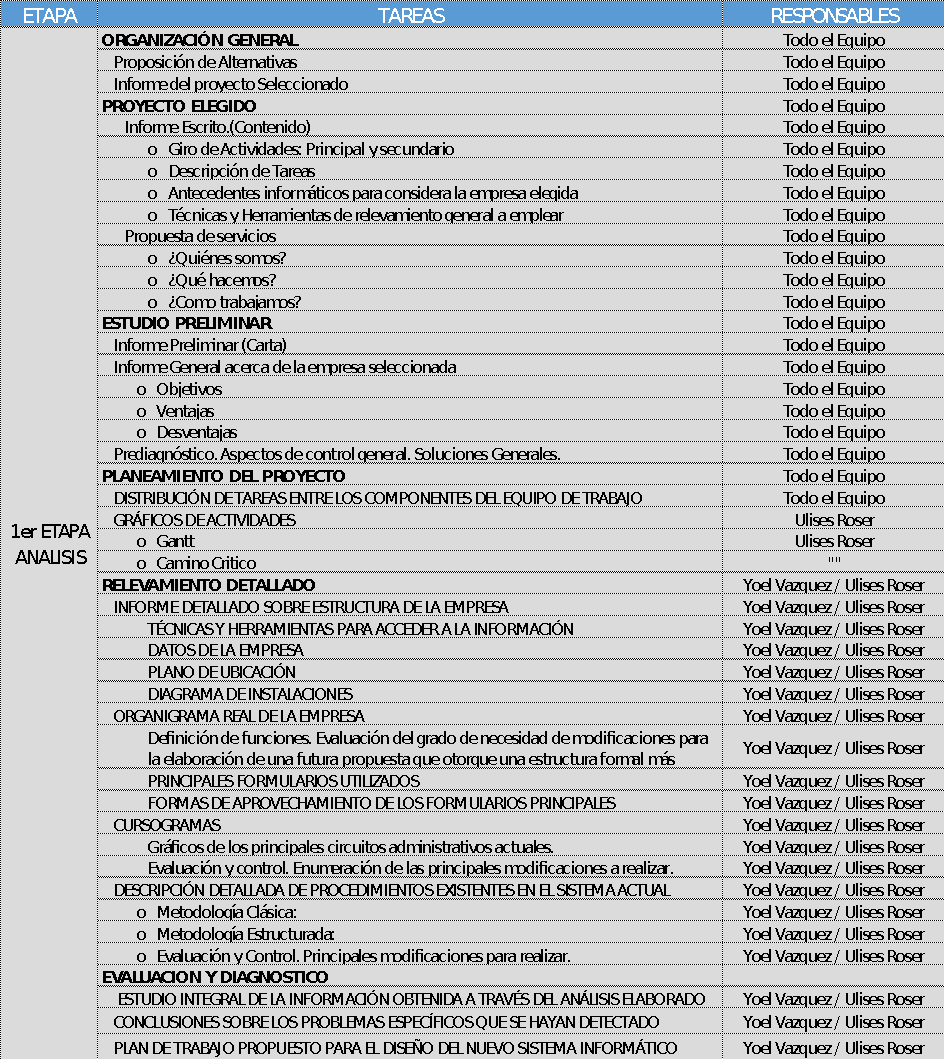


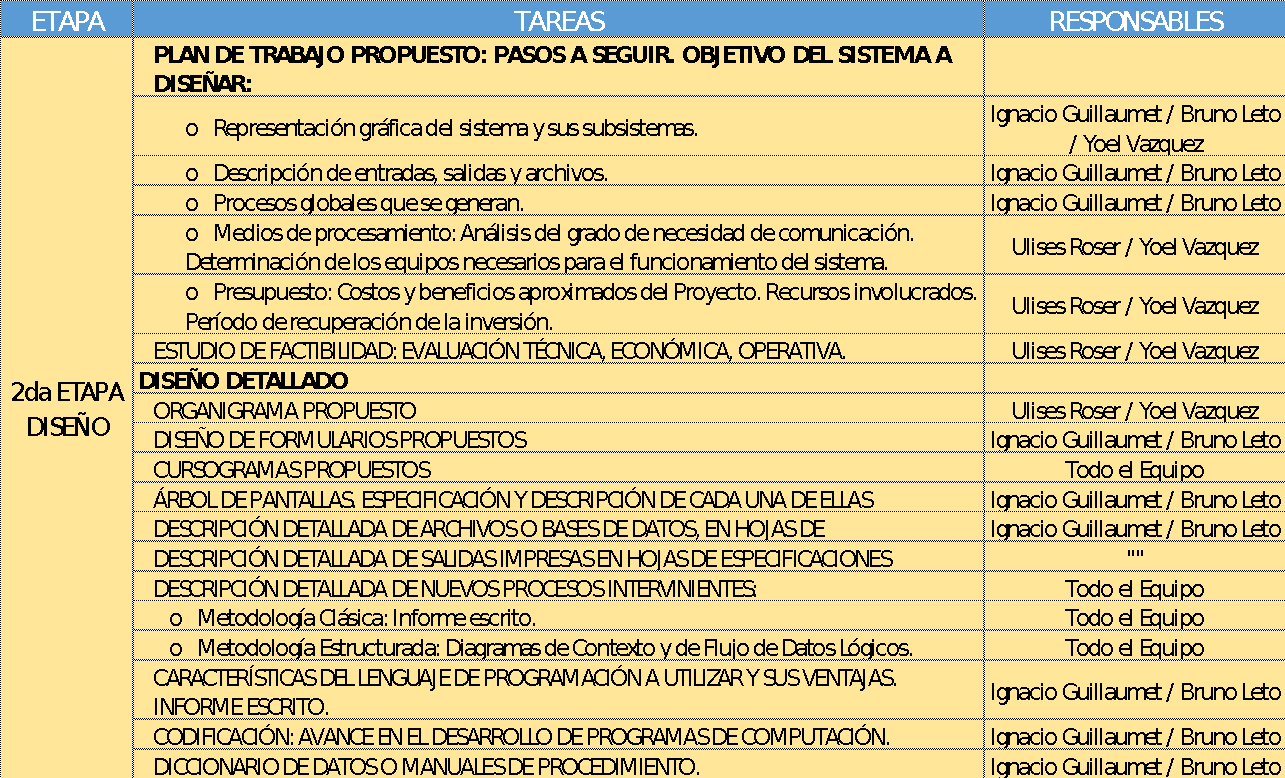
PLANEAMIENTO DEL PROYECTO

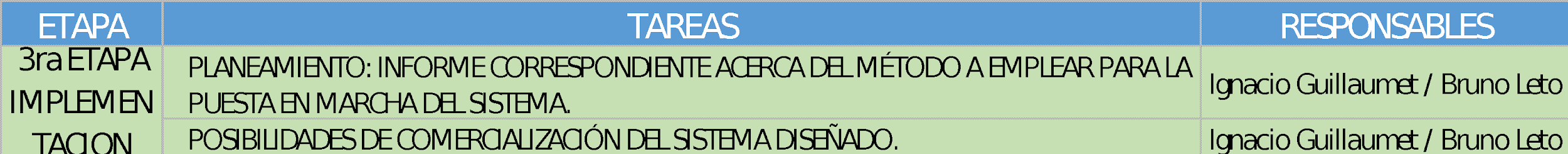
### Distribución de tareas entre los componentes del Equipo de Trabajo

El proyecto consta de tres etapas: ANÁLISIS, DISEÑO e IMPLEMENTACION.

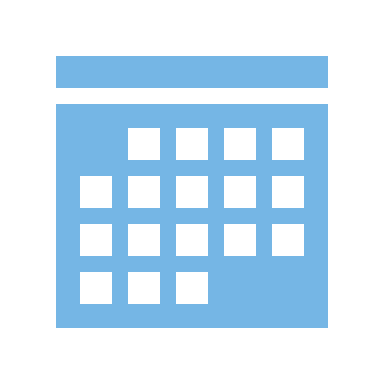
A continuación, se enumeran las Etapas, junto a sus tareas en detalle y los responsables de llevar a cabo las mismas.

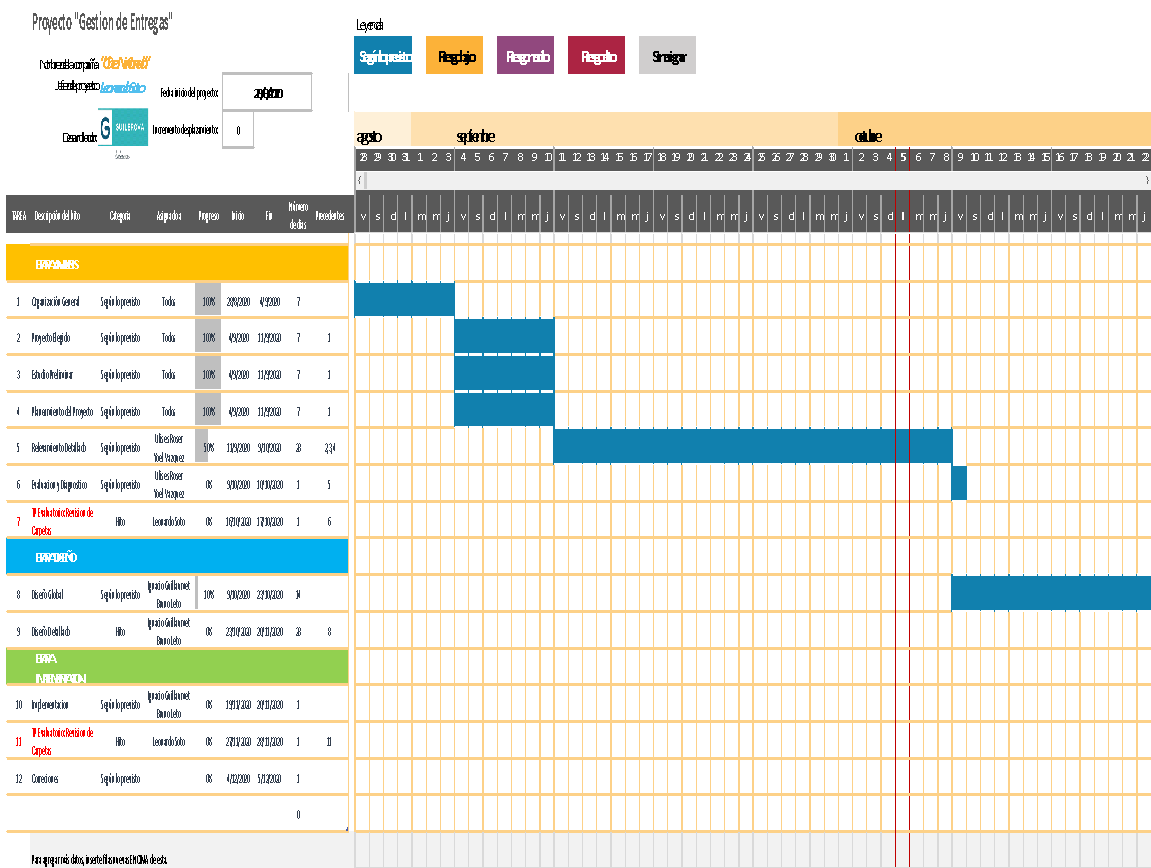






## **GRAFICO DE ACTIVIDADES**

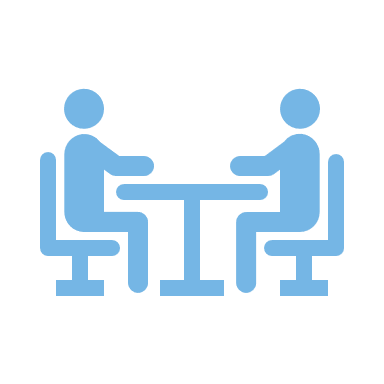


Gráficos de actividades

*Gantt y/o Camino Crítico*

### Gantt Planificado

## **RELEVAMIENTO DETALLADO**



RELEVAMIENTO DETALLADO

*DESARROLLO: 11/09/2020 al 09/10/2020*

*ENTREGA: 09/10/2020*

### Informe detallado sobre estructura de la empresa

### Técnicas y Herramientas para acceder a la información.

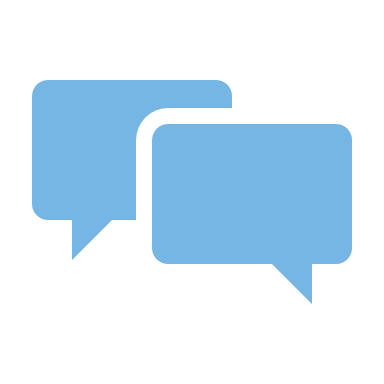
Las herramientas utilizadas para obtener la información fueron las siguientes:

1. Entrevistas presenciales
   1. Reuniones con los dueños
   2. Reuniones con los empleados
2. Observación personal y directa en el local gastronómico

#### Observación personal y directa

Como empleado del Restaurante y representante del equipo Guilerova tengo la oportunidad de conocer personalmente el establecimiento ubicado en la avenida Avellaneda 4199, Floresta, Ciudad Autónoma de Buenos Aires.  
Pude entonces observar como primer medida que el sistema utilizado para la toma, control y seguimiento de los pedidos es obsoleto y totalmente informatizado, ya que estas funciones se hacen tomando nota en una hoja y carecen de un sistema informático que permita a los empleados disponer de tiempo para realizar otras tareas, tales como recibir llamados telefónicos, atender a los clientes que ingresan al local y tomar los pedidos. Podemos concluir que el sistema ofrecido podría mejorar la productividad de los empleados.

### **ENTREVISTAS**



#### Entrevista Dueño

* Fecha entrevista: 27/09/2020
* Rol: **Dueño**
* Entrevistado: Roberto Alvarez
* Entrevistador: Yoel Vazquez
* Tareas: Dueño del establecimiento
* Preguntas claves

1. ¿El sistema utilizado hoy en día le es útil?

El sistema actual demanda mucha pérdida de tiempo en general es por eso por lo que nos

Interesa su propuesta.

1. A partir de la idea planteada para mejorar la problemática del sistema actual con respecto a crear una plataforma para poder facilitar el control de los pedidos ¿Estás dispuesto a implementarlo en tu restaurante?

Francamente estamos muy interesados por que tenemos la necesidad imperiosa de informatizar toda la información referida a la toma de los pedidos. Estamos ansiosos de poder probarlo y mi equipo de trabajo hará lo posible por adaptarse rápidamente a él.

1. ¿Cuentan con alguna computadora? Si es así, para que la utilizan.

Si tenemos una computadora la cual es utilizada para la toma de pedidos para las diversas aplicaciones de mensajería y reparto.

1. ¿Qué que te gustaría que incluyéramos en la plataforma?

Nos gustaría que se puedan agregar promociones, editarlas cuando haya que hacer alguna actualización de precios o cambios, que se puedan visualizar los menús en pantalla, y que haya alguna forma donde pueda consultar un informe mensual de la cantidad de pedidos realizados.

#### Entrevista Delivery

* Fecha entrevista: 18/09/2020
* Rol: **Delivery**
* Entrevistado: Juan Carlos Lamas
* Entrevistador: Yoel Vazquez
* Tareas: Las tareas realizadas son de cadeteria en moto
* Preguntas claves:

1. ¿Qué herramientas utiliza para realizar su trabajo?

Utilizó en primer lugar una moto y una caja en donde voy guardando los pedidos

1. ¿Llevas algún tipo de control para saber los pedidos que entregas?

Si, por supuesto a medida que voy entregando los pedidos los voy anotando en mi celu y utilizo la calculadora para irlos sumando.

1. ¿Te dan algún tipo de ruta desde el restaurante?

La verdad no, si lo hacían al principio con un mapa de la zona que tenían pegado en la pared en el cual podías ir memorizando las calles, pero francamente ahora con un tiempo repartiendo ya las sé de memoria.

1. ¿Cómo se comunica con las distintas áreas de la empresa?

Cuando tengo algún tipo de duda o consulta se la hago al encargado.

1. ¿El sistema utilizado hoy en día le es útil?

Para mí no porque hay veces que me confunde la letra o los números de los encargados o hay veces que se pierden las comandas con la dirección y lo que pidió el cliente. Una vez me paso y me tuve que volver al restaurante y hasta que no llamó el cliente no le pudimos mandar el pedido porque no teníamos la dirección.

#### Entrevista Encargado

* Fecha entrevista: 22/09/2020
* Rol: **Encargado**
* Entrevistado: Kevin Drago
* Entrevistador: Yoel Vazquez
* Tareas: Se encarga del control de los pedidos y el manejo de la caja
* Preguntas Claves:

1. ¿Qué herramientas utiliza para realizar su trabajo?

Actualmente las tareas que desempeño que son básicamente la toma de pedidos y el arqueo de caja las hago de forma manual.

1. ¿Qué cosas pensás que podrían mejorar tu trabajo?

Bueno una de las cosas que me gustaría seria que la toma de pedidos dejará de ser de forma manual para que me dé más tiempo de atender el teléfono ya que todos te realizan los pedidos a la misma hora y se complica bastante.

1. ¿Cómo llevas el control de los pedidos abonados por los delivery?

Los anoto en una planilla y cuando me lo abonan los marco y de esa forma ya se que me lo abono.

1. ¿Qué hacen con los pedidos que tuvieron algún inconveniente?

Depende del problema, la mayoría de las veces se le vuelve a enviar el pedido al cliente si es que salió mal de acá, ósea si lo mandamos equivocado. Hay veces que llaman y los clientes no salen a recibirlos y si no los reclaman se comen acá.

### **DATOS DE LA EMPRESA**Tienda

Datos de la empresa:

*Objetivos y Procesos generales. Personal relevado.*

#### Objetivos y Procesos generales

El restaurante “Che Miranda” fue fundado en julio de 2015 en la Av. Avellaneda 4199, Flores, Ciudad Autónoma de Buenos Aires. Su dueño actual es Roberto Alvarez.

El dueño quiso establecer un negocio que se dedicara a la comercialización de alimentos para el consumo de sus clientes. Tuvo en cuenta la zona , luego de realizar un pequeño estudio de mercado, vio la posibilidad de abrir un local gastronómico.

Las principales tareas que se destacan en “Che Miranda” es la elaboración de pizzas y platos variados, atención al cliente y envíos por delivery.

La empresa notó que los repartos de comida podrían empezar a crecer de forma exponencial y había que encontrar una manera para poder manejar el volumen de pedidos por medio del sector de Delivery.

Por esta razón, surgió la posibilidad de modernizar su empresa a través de un sistema de pedidos.

Actualmente “Che Miranda” cuenta con encargados, cocineros, mozos, mantenimiento, entre otros. Y el dueño se encarga de la administración del restaurante.

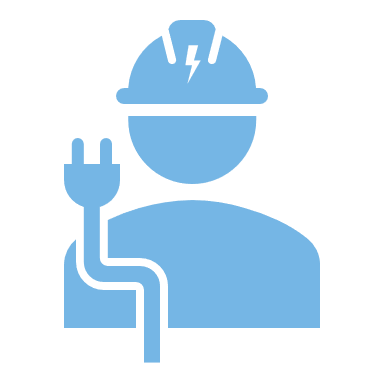
### **PLANO DE UBICACION**



#### Plano de ubicación



### **DIAGRAMA DE INSTALACIONES**

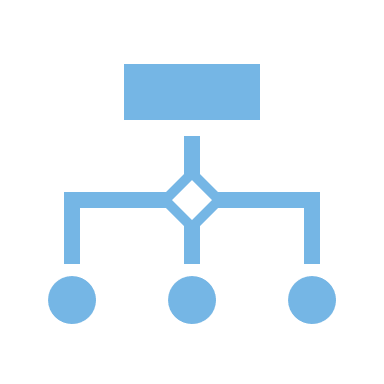


#### Diagrama de instalaciones

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

### **ORGANIGRAMA**



#### Organigrama real de la Empresa:

*Definición de funciones. Evaluación del grado de necesidad de modificaciones para la elaboración de una futura propuesta que otorgue una estructura formal más eficiente.*



### **FORMULARIOS**



### Principales formularios utilizados

*Necesidad de mejoras y/o creación de otros.*

Los formularios utilizados por Che Miranda que por una observación directa entendemos que usan:

**Factura tipo A:** Es solicitado una vez al mes al Proveedor.

**Factura tipo B:** Es solicitado la mayoría de las veces al Proveedor.

**Comprobante de compra:** Una vez realizada una venta de algún producto, ya sea en forma telefónica o por mostrador, se entrega al cliente un ticket de compra original y se guarda una copia.

**Formulario de Productos:** La carta se encuentra a disposición del cliente con todo lo ofrecido en el restaurante y se envía folletería a los clientes que realizan su pedido en forma telefónica

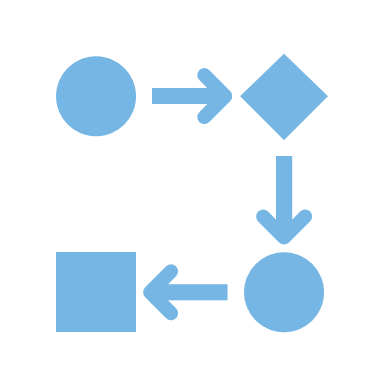
**Formulario de Pedido:** Se realiza una comanda en la cual se encuentra detallado los productos a comprar

#### Formas de aprovechamiento de los formularios principales

Che Miranda se desvincula de los procesos de administración, ya sea pagos de sueldos, pagos de servicios o productos, etc. Contratando un externo para administrar los procesos administrativos.

Los Formularios Tipo A, B y Comprobante de compra no se modificarán por temas administrativos, dejando solo modificar el Formulario de Productos el cual es una guía tanto interno como externo de los productos que of

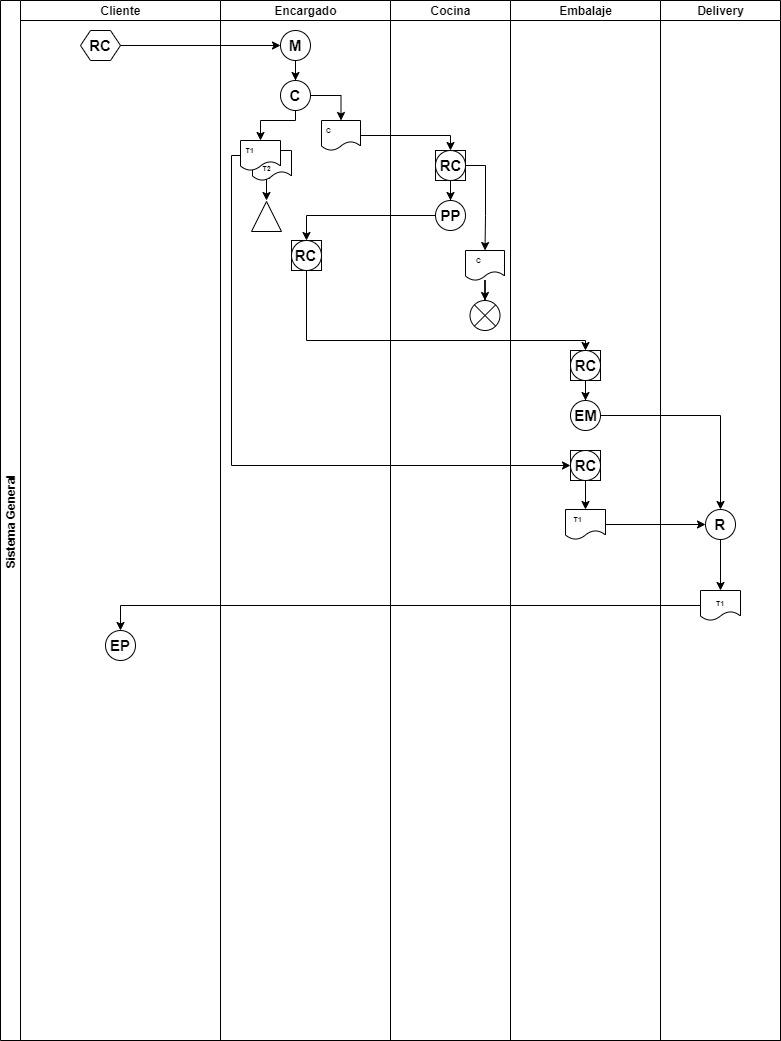
**CURSOGRAMAS**



Cursogramas

#### Gráficos de los principales circuitos administrativos actuales.

##### Cursograma de Venta a Clientes:



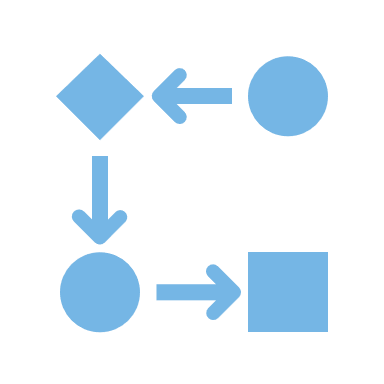


organigrama

## 

## 

## **PROCESOS EXISTENTES**



#### Descripción Detallada de Procedimientos existentes en el sistema actual:

#### Metodología Clásica: Informe escrito.

#### Solicitud a Proveedores

La ***cocina*** de Che Miranda verifica si hay suficiente stock de mercadería para continuar su labor normalmente. En caso de que no haya suficiente mercadería, realiza un pedido de mercadería al ***dueño***.

El ***Dueño*** recibe y controla el pedido.

Los ***Proveedores*** reciben la orden de del ***dueño*** y preparan la mercadería para luego enviarla al sector de la ***cocina***. Al mismo tiempo elaboran una factura por duplicado que será recibida por el ***dueño***.

El ***dueño*** recibe y controla las facturas.

La mercadería es recibida por la ***cocina***. Allí se verifica si la mercadería corresponde al pedido realizado. Si es incorrecto, el pedido, se realiza el reclamo correspondiente.

El ***dueño*** recibe y controla el Reclamo. Acto seguido confecciona un reclamo que es enviado al el/los Proveedor/es.

 Los ***Proveedores*** reciben el reclamo y preparan la mercadería faltante y es enviada al sector de la cocina.

La ***cocina*** recibe la mercadería.

#### Venta a Clientes

El ***cliente*** inicia el proceso solicitando un pedido de menú. El pedido lo recibe el encargado de Che Miranda. El ***encargado***, pasa al sector ***cocina*** el pedido para que preparen el menú para enviarlo y del mismo modo, cuando está listo es embalado por el ***embalador*** y se lo entregan al sector de ***delivery***.

El ***delivery*** recibe el pedido para llevarlas a la dirección del cliente.

El ***cliente*** recibe su pedido. Finalmente sea entregado al ***encargado*** el dinero correspondiente al pedido.

#### Metodología Estructurada: Diagramas de contexto y Flujo de datos lógicos

Debido a que se optó por la metodología clásica, no se desarrollara este método.

#### Evaluación y control. Enumeración de las principales modificaciones a realizar.

Che Miranda no posee ningún tipo de sistema para llevar el control del sector de delivery. Por eso, el principal objetivo que tiene Guilarova es crearlo: Crear un sistema que permita al encargado llevar un control más eficiente.

Permitirá al encargado utilizar el Sistema, a través de la creación de un usuario con sus datos para poder ingresar posteriormente. De esta forma, se podrá identificar a los clientes y mostrar los pedidos que realizó.

Como devolución por ingresar como usuario, el sistema le mostrará la variedad de platos que ofrece Che Miranda, como así también la opción de poder elegir entre retirar su pedido personalmente o ser enviado a su ubicación.

Para ello, el sistema le permitirá completar su pedido, especificando qué productos estaría solicitando, como así también la cantidad y el precio que va a abonar de la compra en su totalidad.

El otro objetivo primordial es incorporar a este sistema una visualización para el Dueño del establecimiento, de todos los pedidos solicitados por los clientes.

En resumen, el objetivo es crear un sistema nuevo que permita el uso de los distintos módulos ensamblados que permitan la posibilidad de realizar la toma de pedidos con una mayor facilidad y comodidad, y de la misma forma una fácil visualización y control  de pedidos de parte del personal de Che miranda.

## **EVALUACION Y DIAGNOSTICO**Lupa

### Estudio integral de la información obtenida a través del Análisis elaborado

*Conclusiones sobre los problemas específicos que se hayan detectado*

#### Objetivo del proyecto:

El objetivo del proyecto se basa en realizar una plataforma que permita gestionar los pedidos de manera eficiente. Esto permitirá a nuestro cliente “Che Miranda”, tener un mayor control y efectividad sobre los mismos.

#### Objetivos del informe:

Luego de realizar varias reuniones de entendimiento, analizar, contemplar las limitaciones del entorno que se maneja en el restaurante, se puedo obtener un estudio preliminar.

Se realizó el siguiente diagnóstico sobre los procesos que actualmente deberían mejorar.

• Los empleados tienen interés en cambiar la forma de trabajo, pero están acostumbrados su modo de operación actual, algunos manifestaron desinterés por el proyecto. Esto podría ser una limitante al momento de adaptarse al nuevo sistema.

• Los empleados realizan en muchas ocasiones tareas no automatizados, y esto ocasiona una pérdida de tiempo sustancial en la entrega de pedidos.

• Los productos son ofrecidos por medio telefónico, y el encargado suele demorar en contestar a los clientes mientras se entera del plato del día.

• Debido a que no hay una base de datos de los pedidos que se dan en un día, el dueño debe confiar en el encargado y su balance mensual es aproximado.

#### Proponemos realizar estas modificaciones para optimizar los procesos:

• Debido a que el sistema propuesto puede ejecutarse en una PC , el encargado y el dueño , podrán consultar en tiempo real los pedidos que se están realizando y los que fueron hechos.

• El backend podrá correr en una PC local de recursos medianos, la cual podrá cumplir con el Rol de servidor con el objeto de mantener el motor de base de datos de pedidos.

• La base de datos que se utilizará es SQL Server, es conocida a nivel a mundial y posee un amplio soporte.

• El software contará con una interfaz amistosa , de fácil acceso y entendimiento por parte de los usuarios finales, con lo cual se maximizará el tiempo de respuesta del empleado a la hora de tomar un pedido.

### Plan de trabajo propuesto para el diseño del nuevo sistema informático

*Determinación de requerimientos en forma detallada.*

A través de la información obtenida se propuso implementar un nuevo sistema modular , que en este caso se orienta a únicamente a los pedidos y reportes de los mismos, pero con opción a extenderse en próximos módulos, en caso de que el cliente lo requiera.

Los módulos permitirán organizar, registrar y administrar las distintas modalidades de trabajo del restaurante, como la toma de pedidos, la gestión de proveedor, la gestión de recursos humanos, las finanzas , entre otros.

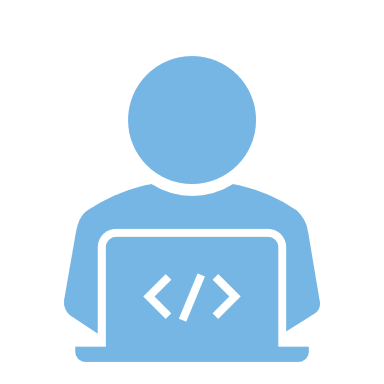
En este caso, nos centraremos el módulo de gestión de pedidos. La cual es una interfaz visual, intuitiva y amigable, que permite navegar entre las opciones, para que los pedidos se concreten lo antes posible.

La gestión de pedidos posee una base de datos, con tablas que permiten registrar los movimientos que involucran este proceso. Dicha base se podrá utilizar en caso futuro de adicionar otros módulos.

El sistema permitirá realizar reportes que llevarán a un mejor control de la información.

Base de datos SQL: Se va a utilizar SQL Server que al ser multiplataforma se puede instalar en distintos S.O, además de ser probadas por distintos usuarios y empresas con alto éxito.

# **ETAPA DE DISEÑO**



## **DISEÑO GLOBAL**Mundo

### Plan de trabajo propuesto: pasos a seguir. Objetivo del sistema a diseñar

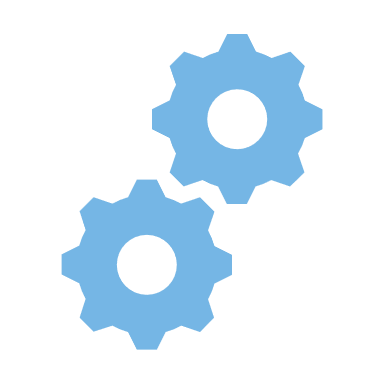
#### Sistema de gestión de pedidos.

El objetivo se compone de lo explicado a continuación:

El sistema permitirá diferenciar al usuario por el rol que cumpla en el restaurante Che Miranda. Dependiendo de cual se trate el usuario podrá llevar a cabo gracias al sistema, diferentes tareas, como las de realizar la toma y la creación de pedidos de una manera informatizada. También se podrán gestionar los productos y los pedidos de una manera eficiente facilitando la comunicación con el área de cocina y de delivery.

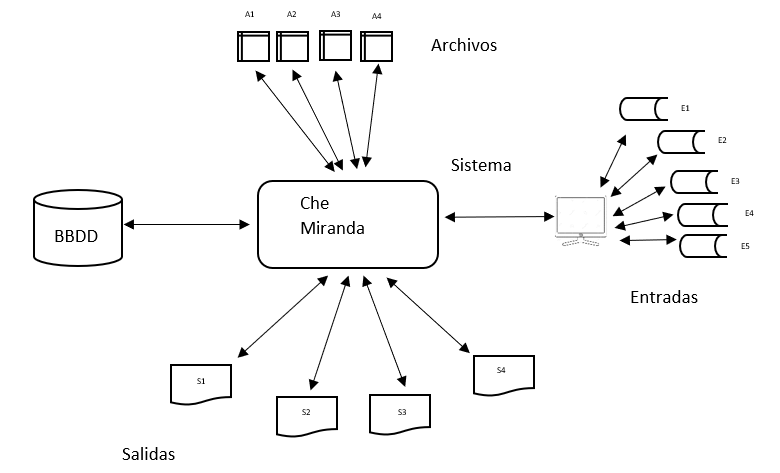
A su vez facilitará el control de las ventas, el personal involucrado en la labor diaria/mensual por medio de informes, modificaciones, y alta de registros de clientes, pedidos y personal.

### **SISTEMA Y SUBSISTEMAS**

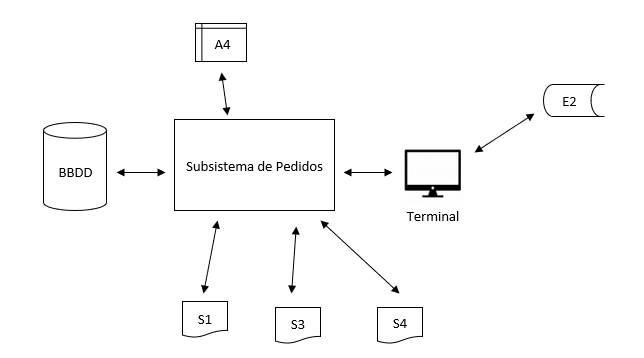


#### Representación gráfica del sistema y sus subsistemas.

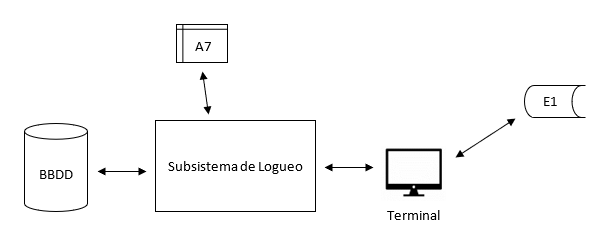
##### Sistema GENERAL



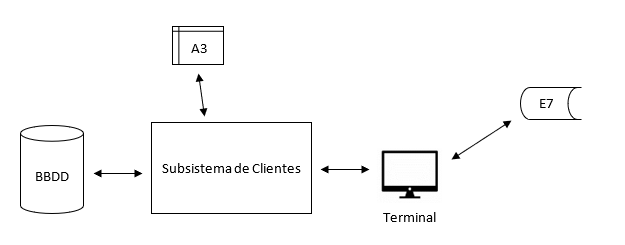
##### Subsistema de Pedidos



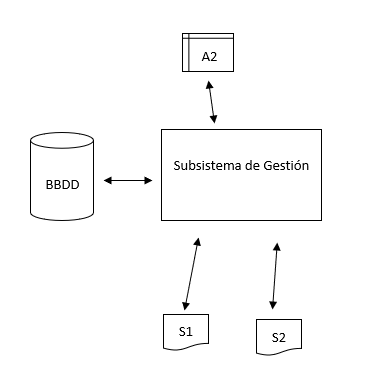
##### Subsistema de Logueo



##### Subsistema de Clientes



##### Subsistema de Gestión



#### Descripción de entradas, salidas y archivos

##### Entradas

###### E1 - Nombre: Acceso al sistema (usuarios)

Objetivo: Resguardo de datos del sistema y restricción de acceso a áreas del mismo.

Descripción:

* DNI
* Contraseña

###### E2 - Nombre: Creación de Pedido

Objetivo: Registrar los pedidos de los clientes para su elaboración y entrega.

Descripción:

* Id pedido
* Forma de Pago
* Cliente
* Usuario
* Fecha y hora
* Cadete
* Estado
* Detalle
* Observación

###### E3 - Nombre: Alta y modificación de productos

Objetivo: Crear nuevos productos y modificar los existentes.

Descripción:

* Id producto
* Descripción
* Categoría
* Precio

###### E4 - Nombre: Alta y modificación de los usuarios (dueño y encargados)

Objetivo: Crear nuevos usuarios y modificar los existentes.

Descripción:

* Id usuario
* Nombre
* Apellido
* CUIL
* Numero de contacto
* Fecha de contratación
* Categoría de usuario

###### E5 - Nombre: Alta y modificación de cadetes

Objetivo: Crear nuevos cadetes y modificar los existentes.

Descripción:

* Id cadete
* Nombre
* Apellido
* Teléfono
* Rodado
* Licencia

###### E6 - Nombre: Alta y modificación de clientes

Objetivo: Crear nuevos clientes y modificar los existentes.

Descripción:

* Id cliente
* Nombre
* Apellido
* Teléfono
* Dirección

##### Salidas

###### S1 - Nombre: Reporte mensual de pedidos

Objetivo: Permite obtener datos acumulados de los pedidos por fecha, de la operatoria diaria.

Descripción:

* Cantidad de pedidos vendidos
* Ingreso total de pedidos vendidos
* Fecha

###### S2 - Nombre: Reporte diario y mensual (TOP) de productos vendidos

Objetivo: Permite obtener los productos vendidos en el día con su cantidad correspondiente.

Descripción:

* Producto
* Categoría
* Cantidad
* Fecha

###### S3 - Nombre: Ticket

Objetivo: Tener un comprobante de la venta realizada.

Descripción:

* Id ticket
* Nombre de la empresa
* CUIT de la empresa
* Dirección de la empresa
* Responsable inscripto
* Pedido (hay mas datos como fecha y hora, detalle pedido, medio de pago, cadete, usuario y cliente)

###### S4 - Nombre: Comanda

Objetivo: Facilitar la comunicación entre el área de pedidos y el área de cocina.

Descripción:

* Id comanda
* Detalle pedido

##### Archivos

###### A1 - Nombre: Cadete

Objetivo: Visualizar los cadetes que manipulan los pedidos.

Descripción:

* Id cadete
* Nombre
* Apellido
* Teléfono
* Rodado

###### A2 - Nombre: Cliente

Objetivo: Registrar y visualizar a los clientes del restaurante.

Descripción:

* Id cliente
* Nombre
* Apellido
* Teléfono
* Dirección

###### A3 - Nombre: Pedido

Objetivo: Resguardar los datos de los pedidos creados.

Descripción:

* Id pedido
* Forma de pago
* Cliente
* Usuario
* Fecha y hora
* Observaciones
* Cadete
* Estado del pedido
* Detalle pedido

###### A4 - Nombre: Producto

Objetivo: Administración de los productos del restaurante.

Descripción:

* Id producto
* Descripción
* Categoría
* Precio

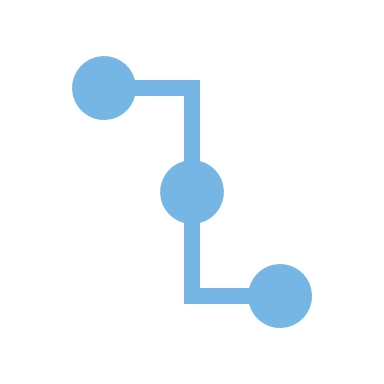
###### A5 - Nombre: Usuario

Objetivo: Restringir el acceso al sistema y sus funciones.

Descripción:

* Id usuario
* Nombre
* Apellido
* CUIL
* Numero de contacto
* Fecha de contratación
* Categoría de usuario

### **PROCESOS GLOBALES GENERADOS**



#### Procesos globales que se generan

Los procesos involucrados en el sistema de gestión de pedidos incluyen:

##### **Nombre**: Ingreso de usuarios

**Objetivo**: Permitir el ingreso al sistema a cada usuario según su rol. El sistema contará con por lo menos dos roles; dueño y encargado. El primero de ellos con acceso libre a todo el sistema.

* **Sub-Proceso**: Alta y modificación

**Objetivo**: Alta y modificación de usuarios según el rol que corresponda.

##### **Nombre**: Creación de pedido

**Objetivo**: Permite tomar el pedido de un cliente para su entrega.

* **Sub-Proceso**: Alta cliente

**Objetivo**: Dar de alta un nuevo cliente para asignarlo al pedido.

* **Sub-Proceso**: Creación ticket

**Objetivo**: Dar de alta un nuevo ticket para tener un registro de la venta

* **Sub-Proceso**: Creación comanda

**Objetivo**: Dar de alta una nueva comanda para que el pedido pueda ser preparado.

* **Sub-Proceso**: Alta y modificación de cadete.

**Objetivo**: Gestionar a los cadetes encargados de la entrega de los pedidos.

##### **Nombre**: Estado de los pedidos

**Objetivo**: Permite controlar al pedido por medio de su estado y visualizar otros datos relativos al mismo.

* **Sub-Proceso**: Consulta de pedido

**Objetivo**: Visualizar los datos del pedido

* **Sub-Proceso**: Modificación del estado del pedido

**Objetivo**: Realizar las actualizaciones de los estados correspondientes de acuerdo a la situación de los pedidos.

##### **Nombre**: Informes

**Objetivo**: Generar informes de los pedidos, y productos, según los criterios que el usuario requiera.

* **Sub-Proceso**: Informe de pedidos

**Objetivo**: Visualizar la cantidad de pedidos realizados diarios y mensuales junto con el total del ingreso.

* **Sub-Proceso**: Informe de productos

**Objetivo**: Visualizar los productos junto con su cantidad vendidos diarios y mensuales.

### **MEDIOS DE PROCESAMIENTO**Equipo

#### Medios de procesamiento: Análisis del grado de necesidad de comunicación.

*Determinación de los equipos necesarios para el funcionamiento del sistema.*

Se requieren para el correcto funcionamiento del sistema planteado, los siguientes equipos con características mínimas:

**PUESTO DE TRABAJO / SERVIDOR :**

* PC con :
  + Procesador: 8 núcleos a 2.3 Ghz.
  + Disco rígido: 1 TB
  + Memoria: 16 GB.
  + Dispositivos periféricos: Mouse, teclado, Parlantes
  + Pantalla: LED de 19 pulgadas.
  + Debe contar con acceso a red RJ45 y WIFI
* Sistema operativo: Windows 10 Pro
* Con conexión a red: Se requiere una red interna de 100 Mbps.

Se utilizará un único equipo ya que el sistema fue diseñado para funcionar Standalone.

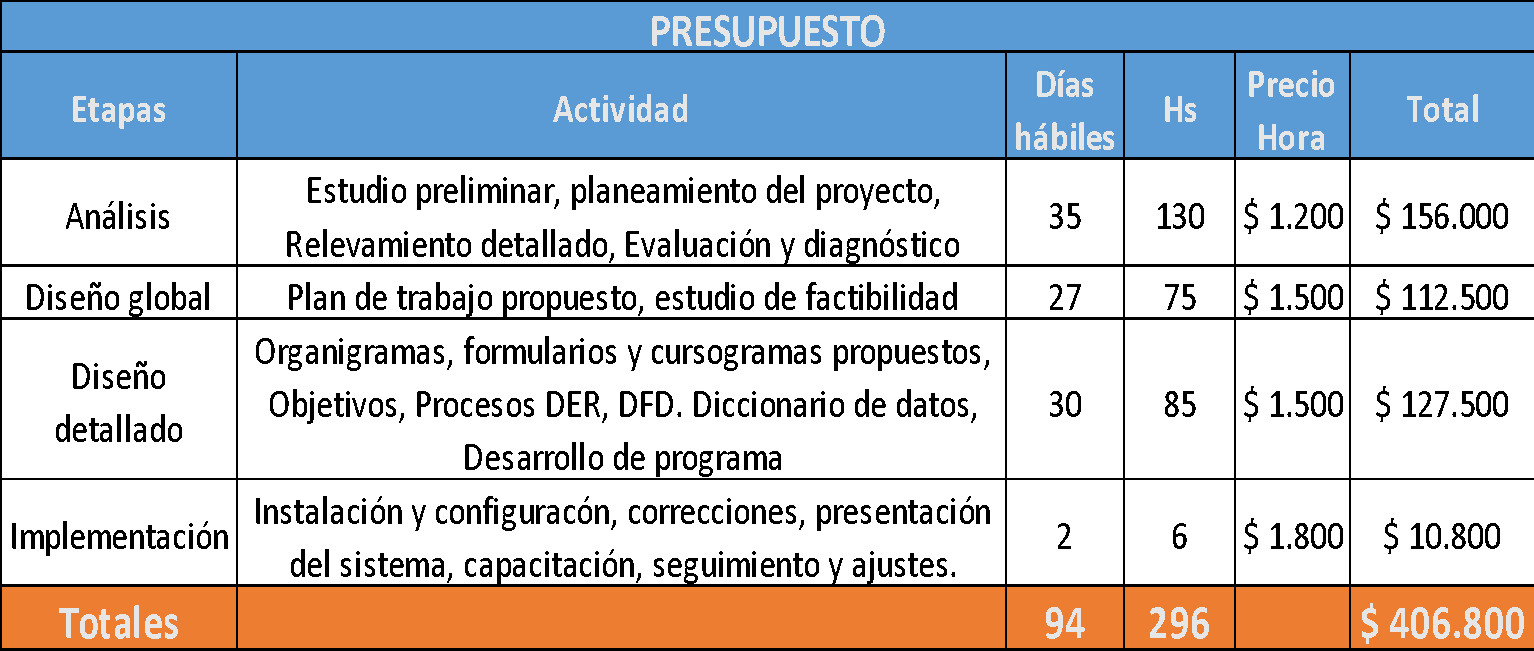
**IMPRESORA:**

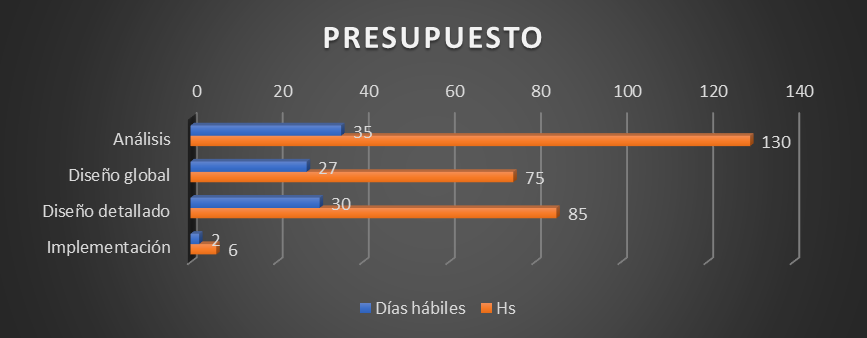
Para la impresión de informes y reportes es necesario contar con una impresora Láser.

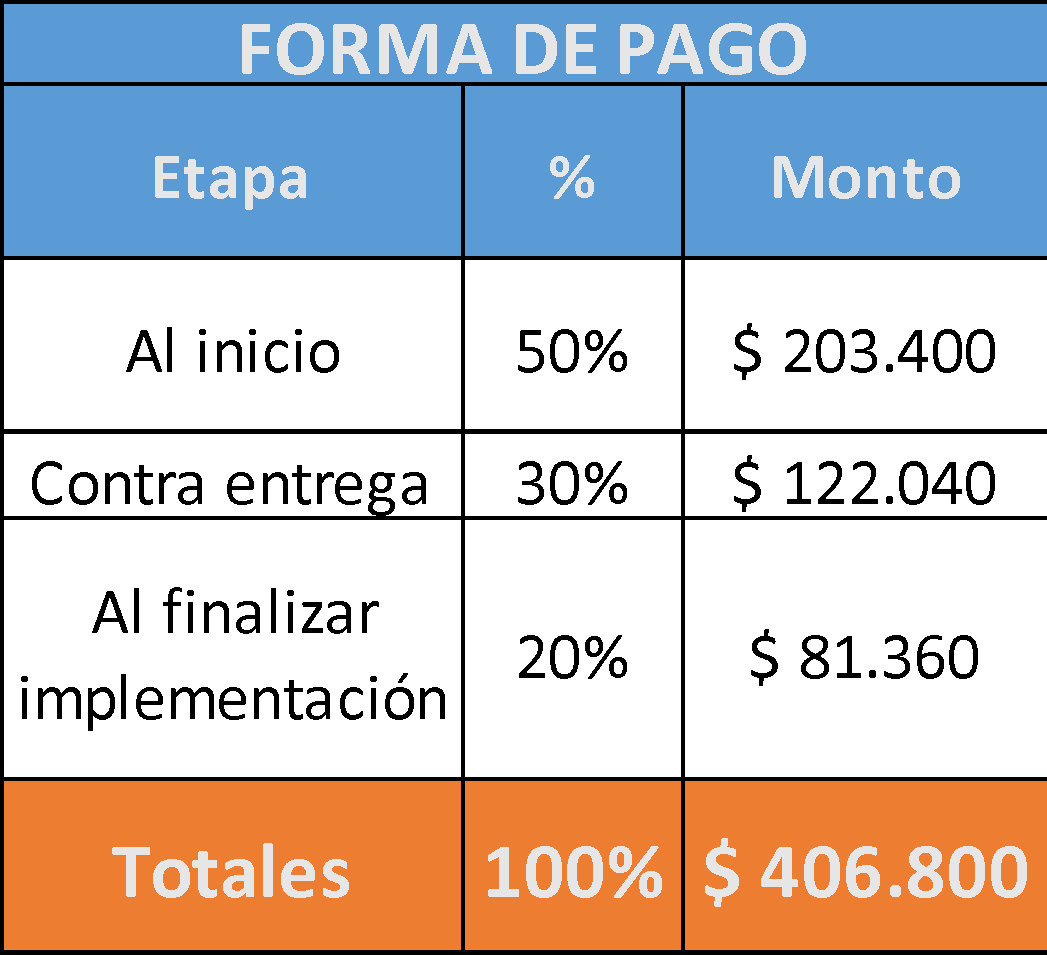
### **PRESUPUESTO**

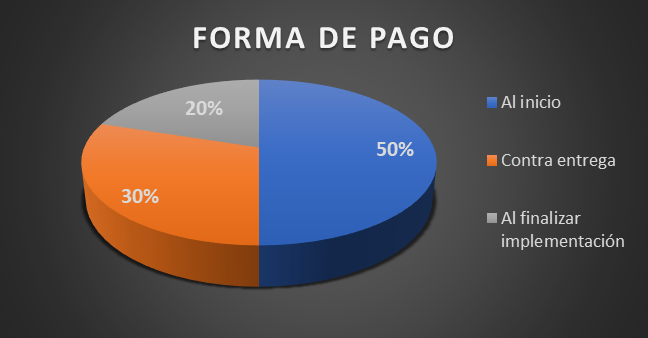


#### Presupuesto: Costos y beneficios aproximados del Proyecto. Recursos involucrados. Período de recuperación de la inversión.









**Beneficios del proyecto**

Tras la implementación de nuestro sistema, se verá un incremento en la velocidad en que se efectúan los procesos, tales como la toma de pedidos y la obtención de informes de ventas. Permitirá al usuario realizar su trabajo de manera intuitiva y eficaz, por medio de una interfaz gráfica amigable.

Un usuario con conocimientos básicos de informática podrá operar eficientemente el sistema. Se podrán generar informes personalizados de acuerdo con la necesidad del usuario.

**Recursos involucrados en el proyecto**

GUILEROVA cuenta con un equipo de cuatro integrantes, que están altamente capacitados para resolver toso los problemas que se puedan presentar en las etapas de análisis, diseño e implementación, del nuevo sistema.

Las tareas están divididas internamente:

**Analista funcional:**

El analista funcional es la persona que se encarga de cumplir con los objetivos planteados por el cliente, extrayendo la información por medio del feedback continuo con el mismo y su organización para resolver dudas de ambas partes.

**Responsable de proyecto:**

Nuestro responsable de proyecto adecuara las tareas de su grupo de trabajo, los objetivos que deberán ser alcanzados , asignando tareas y registrando los progresos de cada etapa . También detectada posibles errores , para su futura corrección.

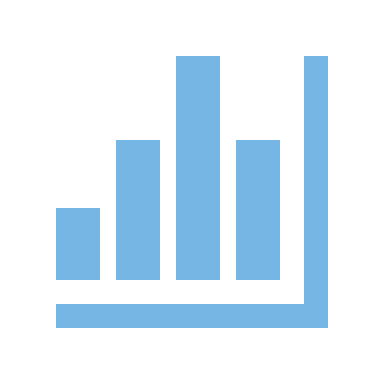
**Arquitecto de Software:**

Nuestro arquitecto de software cuenta con amplios conocimientos técnicos, una gran experiencia en programación, y ejerce las siguientes funciones: Gestión de los requisitos no funcionales y definición de la Arquitectura de Software

**Analista programador:**

Nuestro especialista programador, se codificar de manera efectiva los distintos requisitos relevados por demás integrantes del equipo. También verifica el correcto funcionamiento del programa, también es quien realizara los manuales de uso.

### **ESTUDIO DE FACTIBILIDAD**



#### Estudio de Factibilidad

*Evaluación técnica, económica, operativa.*

##### Factibilidad técnica:

Se ha realizado una evaluación de las tecnologías existentes en la organización relevada, de acuerdo con el nuestro informe, los componentes técnicos que posee la organización serían suficientes para hacer uso del software propuesto por GUILEROVA , aunque recomendamos una actualización de hardware , si se desea que nuestra propuesta de servicio propuesta sin ningún tipo de latencia.

En el apartado de los medios de procesamiento, hemos detallado la incorporación de un equipamiento para ejecutar el nuevo software de manera óptima.

La capacitación del personal en la nueva tecnología está contemplada durante la etapa de implementación. El software se ejecutará en un sistema operativo Windows, lo cual es un entorno conocido por la mayoría de los usuarios , a su vez la interfaz entregada es amistosa, ya que se trata de una aplicación visual, lo cual lo hace sencillo al momento de su utilización, ya que es similar a cualquier aplicativo que podría encontrarse dentro del sistema operativo, con ventanas, barras de desplazamiento y botones descriptivos.

##### Factibilidad económica:

Hemos evaluado el costo – beneficio que implica la implementación de esta solución para la organización. El proyecto tendrá una duración aproximada de tres meses, y durante ese tiempo la organización seguirá operando como lo hacía hasta ahora, con lo cual no genera un impacto negativo en el negocio.

Posterior a la implementación de la solución, el sistema agilizará tiempos, reducirá recursos, lo cual dará como resultado una disminución de costos actuales.

También el equipo acompañara al cliente en todo momento, haciendo reuniones para plantear los avances , mejoras y/o modificaciones.

Los costos netamente se traducen al software propiamente desarrollado, ya que, en este caso, el cliente cuenta con equipamiento necesario para su ejecución y no hay que hacer una inversión en infraestructura de forma inmediata. El costo por el desarrollo y los plazos de pago, están debidamente informados en la sección de presupuesto.

Se aclara que estos montos no contemplan modificaciones, actualizaciones o inconvenientes en el proyecto.

En caso de que se presente la necesidad de modificar el proyecto, ya sea agregando un cambio menor o mayor , a lo estipulado en la etapa de Análisis y relevamiento, se deberá recalcular el tiempo necesario de desarrollo y presupuesto, ya que generarán costos adicionales a los presentados en este documento.

Por todo lo expuesto, consideramos que el costo del proyecto es beneficioso para ambas partes, un costo provechoso para la organización que podrá encargarse de forma eficiente y precisa de su información y reportes, reduciendo la posibilidad de errores y sobre todo generando un ahorro.

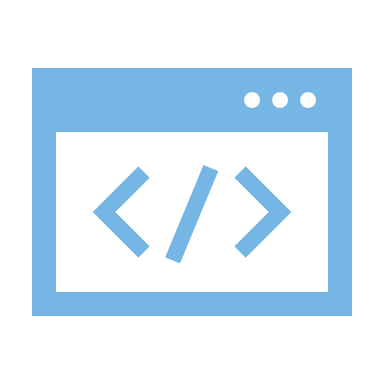
##### Factibilidad operativa:

Este sistema fue pensado en función de simplificar la operación de la organización relevada, es por esto por lo que el principal móvil del equipo de desarrollo es lograr una sinergia entre el usuario y el sistema. Teniendo como objetivo requerir operadores con el menor requerimiento técnico posible.

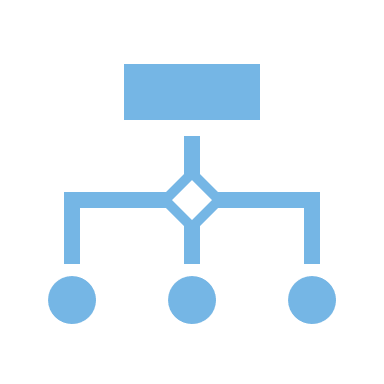
Como se comentó antes, el diseño es simples, intuitivo y lo más concreto posible. Haciendo que la experiencia de usuario sea performante a la hora de realizar una tarea o bien emitir un reporte de estado.

En todo momento el equipo de desarrollo acompañara a los usuarios , con el objeto de brindar una capacitación sobre el uso del nuevo sistema, mostrando las ventajas que tiene por sobre el antiguo modo de operatoria diaria. Además, se pondrá a disposición un manual de uso para los diferentes roles que deban cumplirse, ya sea operadores y administradores.

## **DISEÑO DETALLADO**



### **ORGANIGRAMA PROPUESTO**



DISEÑO DETALLADO

#### Organigrama propuesto

De acuerdo con el análisis realizado por nuestro equipo, y pese a la solución propuesta, no se realizarán cambios en el organigrama del cliente. Por esta razón, el organigrama es el siguiente:



## **FORMULARIOS PROPUESTOS**



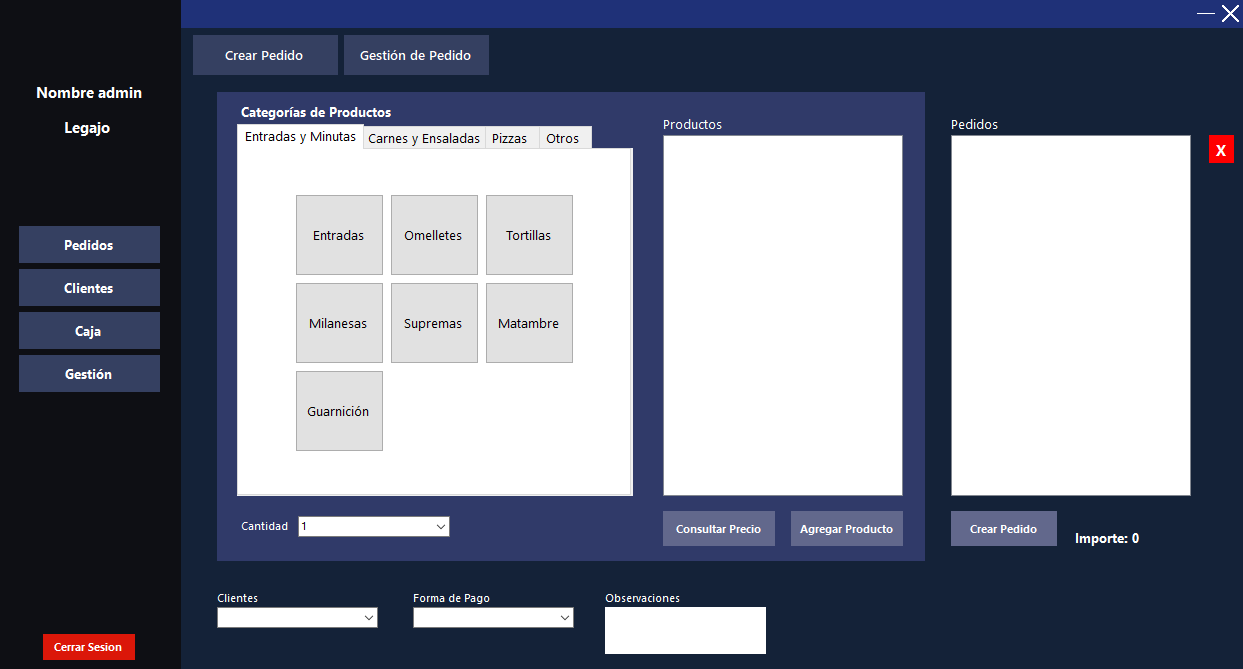
### Diseño de formularios propuestos

#### Inicio de sesión de usuario



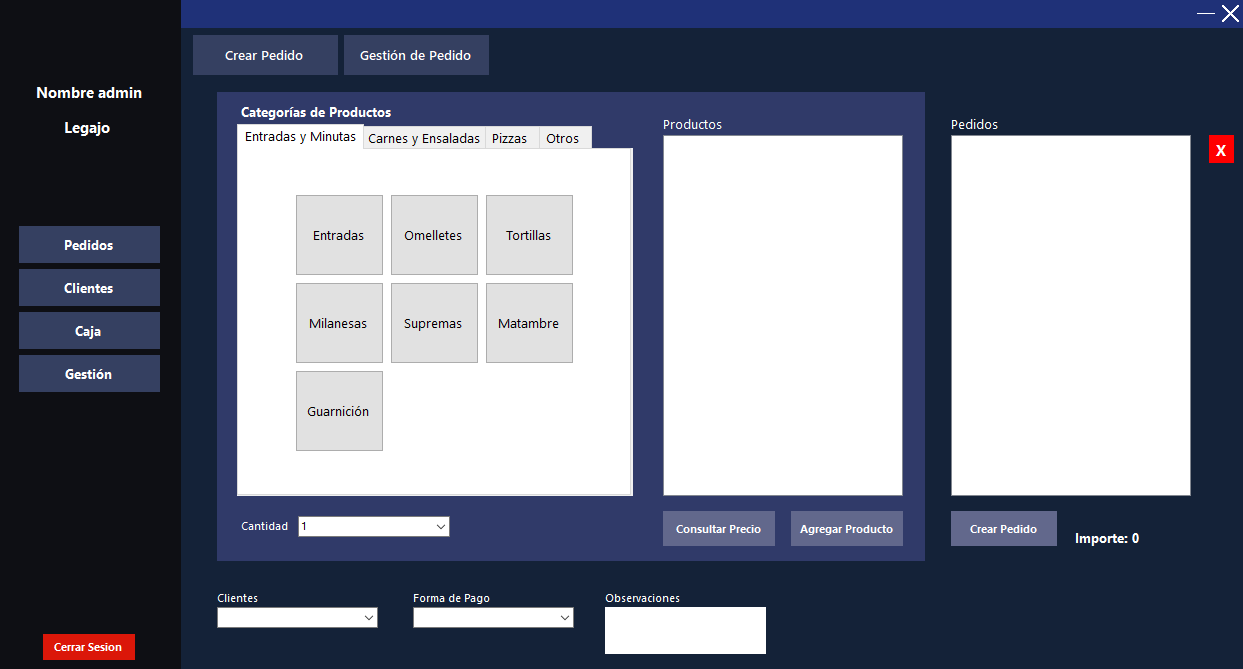
Permite el acceso al sistema por medio del ingreso de un nombre de  usuario y contraseña acorde al mismo. En el caso que el usuario ingresado tenga el rol de dueño, tendrá acceso a funcionalidades correspondientes a su rol generar, modificar y eliminar usuarios nuevos con rol de encargado a través de la sección de gestión y muchas más funcionalidades. En caso de que el rol del usuario ingresado sea encargado este se encontrará limitado en cuanto a poder acceder a ciertas funcionalidades del sistema.

#### Menú Principal

  
Pone en disposición al usuario que ingresó al sistema las diferentes secciones que tiene el mismo las cuales además luego de ingresar a cada una tienen diferentes funcionalidades que resuelven de manera informatizada la problemática planteada por el restaurante, el cual es el cliente. La sección mostrada por defecto luego del inicio de sesión del usuario es la de Pedidos, la cual tiene como subsección la creación de los pedidos.

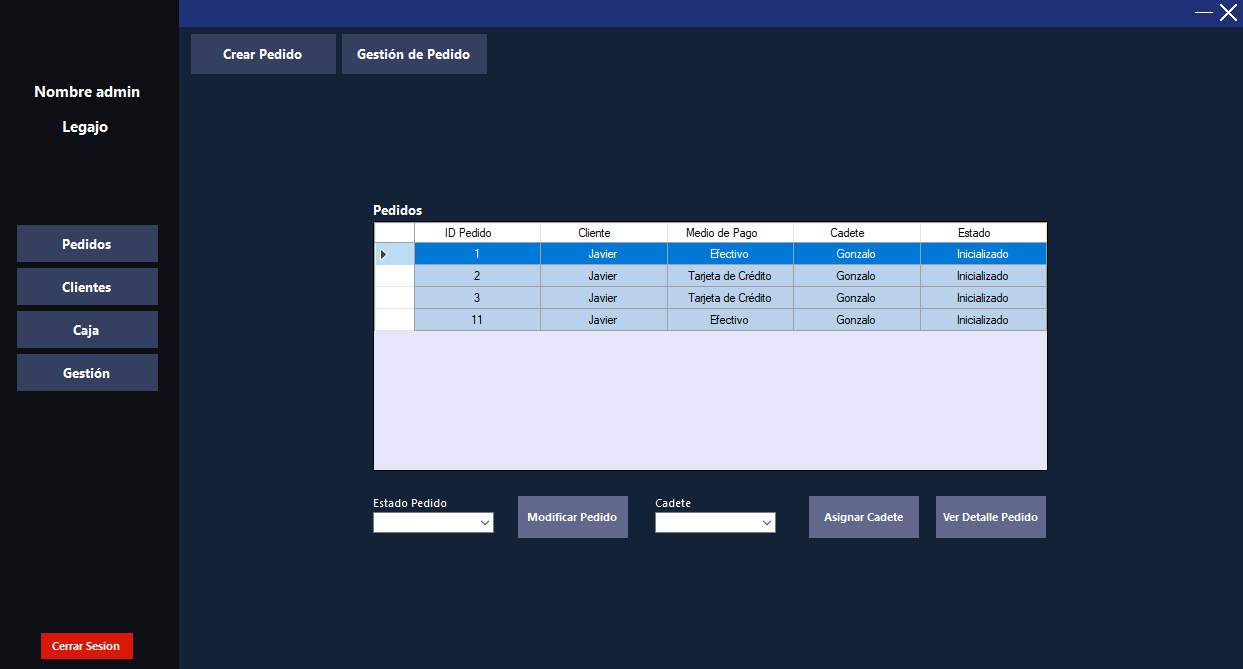
#### Pedidos

##### Crear Pedido



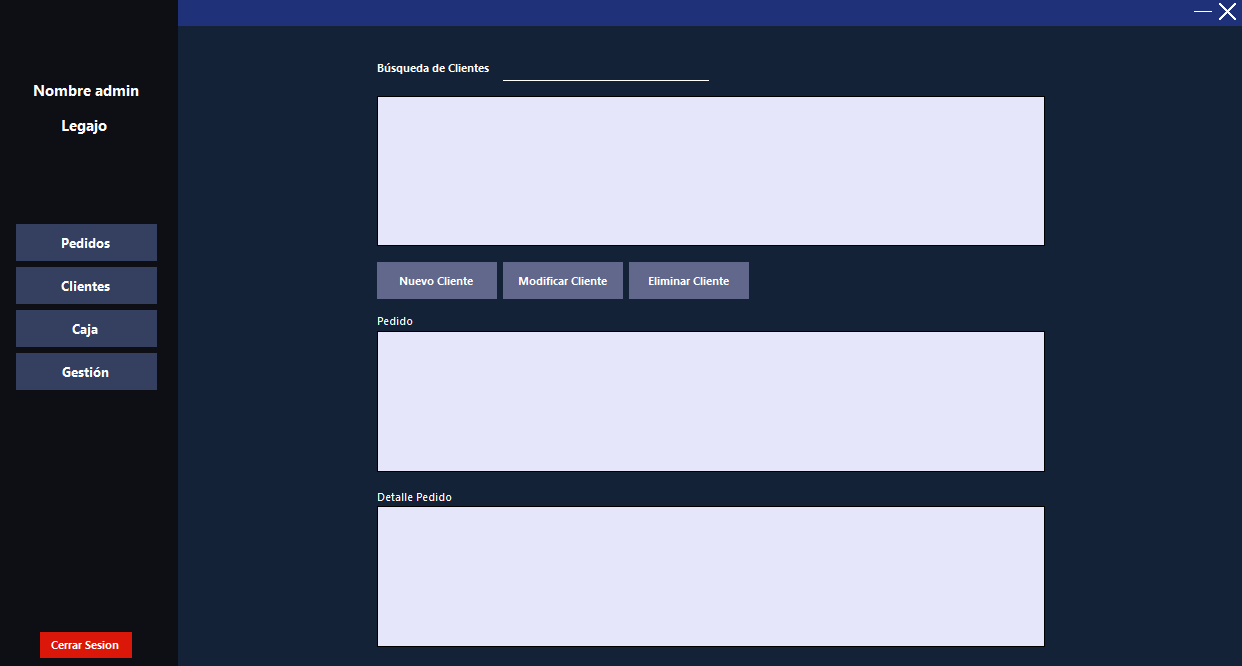
Seleccionando los productos con los que cuenta el restaurante, especificando la cantidad, el cliente que lo recibirá, el medio de pago por el cual éste abonará al restaurante y el cadete que lo entregará, esta subsección de la sección pedidos permite que el usuario tanto encargado como dueño la creación de un pedido y su almacenamiento en la base de datos habiendo previamente registrado al cliente que por teléfono se comunicó con el restaurante.

##### Gestión de Pedido



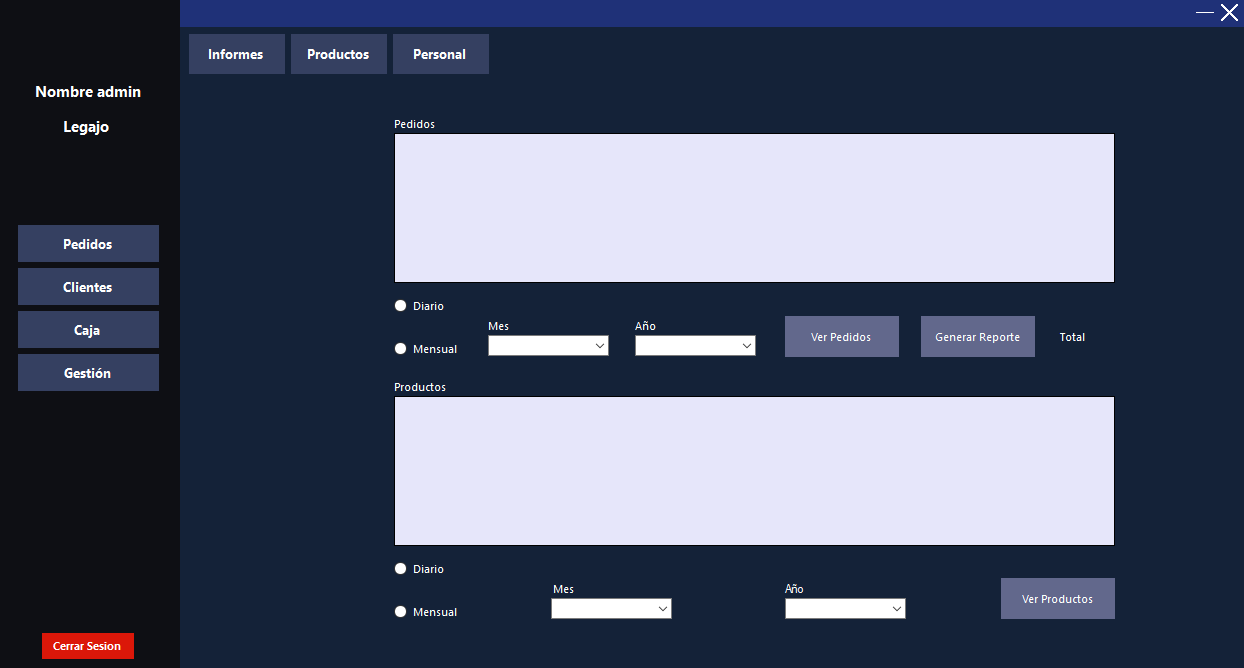
Luego de la creación del pedido, esta subsección permite visualizar correctamente la información relativa a los pedidos creados, desde el estado del pedido que en un primer momento se encuentra iniciado hasta la modificación del mismo y acceder al detalle del pedido para visualizar qué productos conforman al mismo. También tiene la funcionalidad esta subsección de poder asignarle al pedido un cadete para su posterior entrega al cliente.

#### Clientes

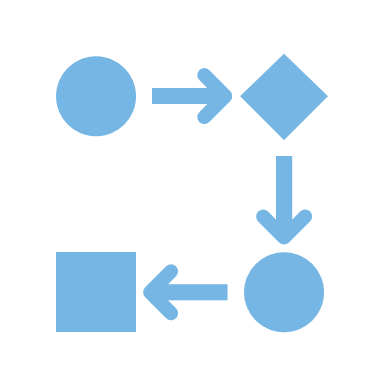
****

Esta sección del sistema permite el registro de los clientes que el restaurante vaya a tener, su visualización y posterior modificación o eliminación en su defecto. Además, hacer posible la rápida búsqueda de un cliente en específico  por medio del ingreso de un dato relacionado al propio cliente, se puede observar del cliente el pedido que solicitó al restaurante además del detalle asociado al mismo, el cual contiene los productos que lo conforman.

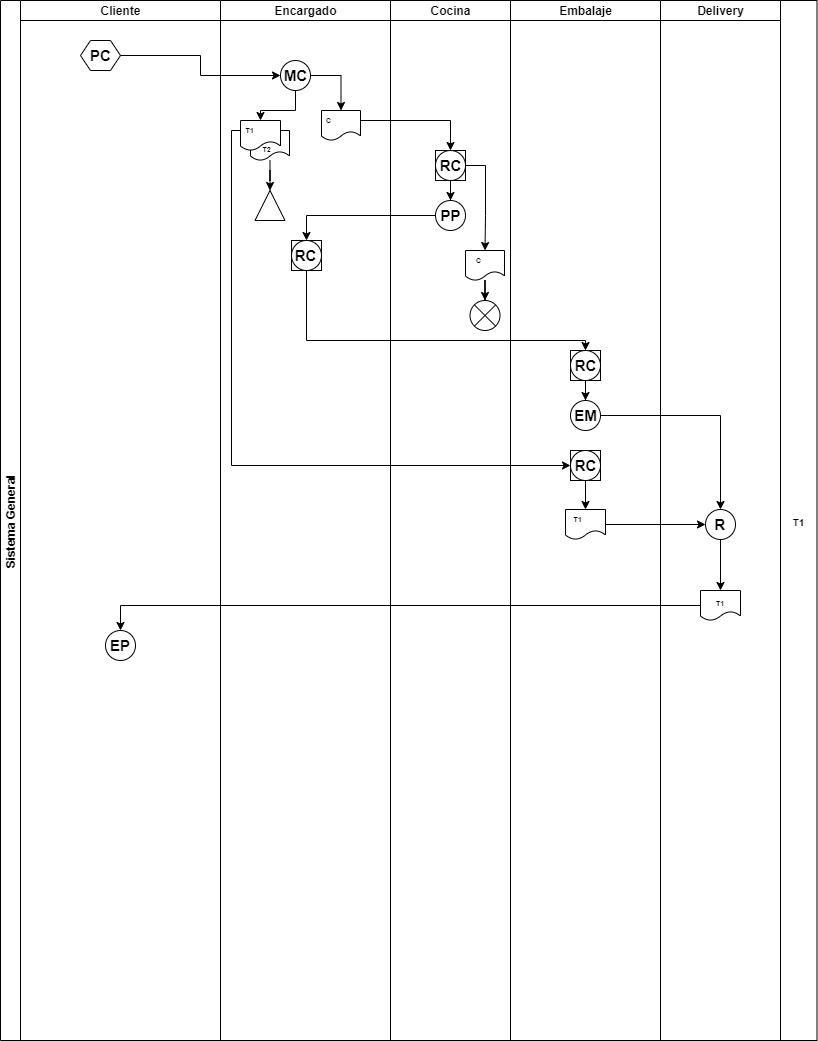
#### Gestión

****

Sección que alberga las funcionalidades únicas y exc**CURSOGRAMAS PROPUESTOS**

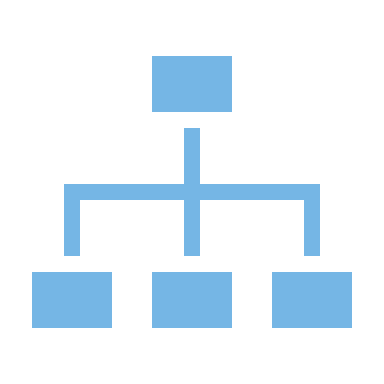


### Cursogramas propuestos



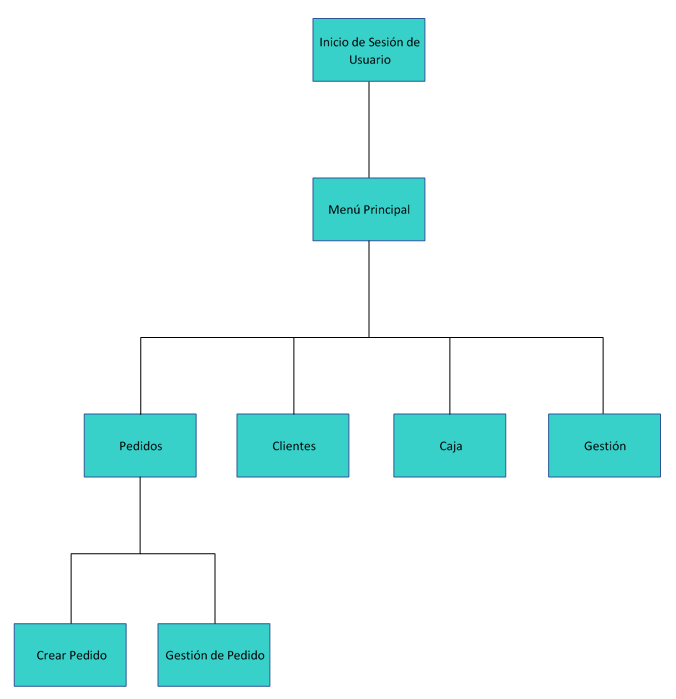


## **ARBOL DE PANTALLAS**

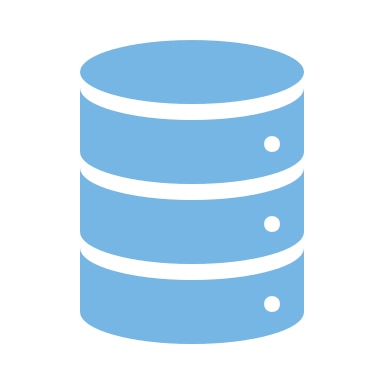


### Árbol de Pantallas

*Especificación*



## **DETALLE BASE DE DATOS**



### Descripción detallada de archivos o bases de datos, en hojas de especificaciones

| Tabla Cadete | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *Idcadete -PK* | int | No |
| nombre | varchar(20) | No |
| apellido | varchar(20) | No |
| telefono | int | No |
| idrodado | int | No |
| licencia | int | Si |

| Tabla Cliente | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idcliente -PK* | int | No |
| nombre | varchar(20) | No |
| apellido | varchar(20) | No |
| telefono | int | No |
| direccion | varchar(50) | No |

| Tabla DetallePedido | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *Iddetallepedido - PK* | int | No |
| idproducto | int | No |
| *Idpedido -PK* | int | No |
| cantidad | int | No |
| precioventa | int | No |

| Tabla EstadoCategoria | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idestadocategoria -PK* | int | No |
| descripcion | varchar(50) | No |

| Tabla FormadePago | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idformadepago -PK* | int | No |
| descripcion | varchar(20) | No |

| Tabla Pedidos | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idpedido -PK* | int | No |
| idformadepago | int | No |
| idcliente | int | No |
| idusuario | int | No |
| fechayhora | datetime | No |
| observaciones | varchar(100) | No |
| idcadete | int | No |
| idestadocategoria | int | No |

| Tabla Productos | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idproducto -PK* | int | No |
| descripcion | varchar(50) | No |
| idproductocategoria | int | No |
| precio | int | No |

| Tabla ProductoCategoria | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idproductocategoria -PK* | int | No |
| descripcion | varchar(50) | No |

| Tabla Usuario | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idusuario -PK* | int | No |
| nombre | varchar(20) | No |
| apellido | varchar(20) | No |
| CUIL | int | No |
| numerodecontacto | int | No |
| fechadecontratacion | date | No |
| idcategoriausuario | int | No |

| Tabla Categoria Usuario | Tipo de datos | Permite nulo |
| --- | --- | --- |
| *idcategoriausuario -PK* | int | No |
| descripcion | varchar(20) | No |

## **DETALLE SALIDAS IMPRESAS**Fax

### Descripción detallada de salidas impresas en hojas de especificaciones

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

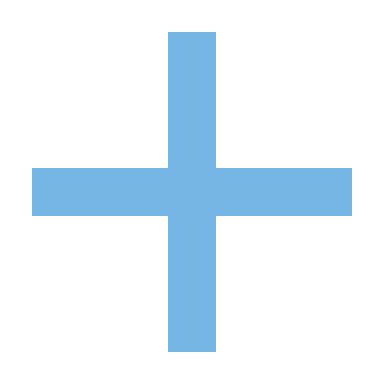
Una captura de pantalla de un celular

Descripción generada automáticamente

Una captura de pantalla de un celular con letras

Descripción generada automáticamente

## **DETALLE NUEVOS PROCESOS**



### Descripción detallada de nuevos procesos intervinientes:

### Metodología Clásica: Informe escrito

#### Solicitud a Proveedores

El cliente determinó que, de ser necesario, se desarrollara esta funcionalidad en una etapa posterior.

#### Venta a Clientes (Delivery)

El ***cliente*** inicia el proceso solicitando un pedido de menú. El pedido lo recibe el ***encargado*** de Che Miranda. El ***encargado***, seleccionara y cotizara el pedido, luego se creará y se registrara en la base de datos, por último, se emite la comanda y los tickets relacionados al pedido ( ticket por duplicado ) , a su vez la comanda pasara al sector ***cocina*** para que preparen el menú para enviarlo y del mismo modo, cuando está listo es embalado por el ***embalador*** y se lo entregan al sector de ***delivery***.

El ***delivery*** recibe el pedido para llevarlas a la dirección del ***cliente***.

El ***cliente*** recibe su pedido. Finalmente sea entregado al ***encargado*** el dinero correspondiente al pedido.

### Metodología Estructurada: Diagramas de Contexto y de Flujo de Datos Lógicos.

* Debido a que se optó por la metodología clásica, no se desarrollara este método.

## **CARACTERISTICAS DEL LENGUAJE**



### Características del Lenguaje de Programación a utilizar y sus ventajas. Informe escrito

#### Paradigma Orientado a objetos

Nuestro software será desarrollado por medio de la programación orientada a objetos. Es una

manera de programar específica, donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para conseguir los objetivos de las aplicaciones. Además, es una forma especial de programar, más cercana a cómo expresaríamos las cosas en la vida real que otros tipos de programación.

#### Framework: Visual Studio 2019

Microsoft Visual Studio 2019, es el entorno de programación de Microsoft que permite desarrollar aplicaciones para Windows, Mac, Android, iOS, páginas web y la nube Azure.

Algunas de las ventajas por la cual nos decidimos por este framework son:

Código administrado: El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

Interoperabilidad multilenguaje: El código puede ser escrito en cualquier lenguaje compatible con

.Net ya que siempre se compila en código intermedio (MSIL).

Compilación just-in-time: El compilador JIT incluido en el Framework compila el código intermedio

(MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.

Garbage collector: El CLR proporciona un sistema automático de administración de memoria

denominado recolector de basura (garbage collector). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por qué liberar la memoria de forma explícita, aunque también sea posible

hacerlo manualmente (mediante el método disponse() liberamos el objeto para que el recolector de basura lo elimine de memoria).

Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos, pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.

Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

#### C#

Las principales ventajas que presenta el uso C# en comparación con otros lenguajes es su potencia como lenguaje, pero también su flexibilidad. Soporta la mayoría de los paradigmas, destacando el paradigma funcional que combinado con el paradigma orientado a objetos hacen del lenguaje uno de los más potentes. C# brinda mejoras frente a lenguajes como C y C++ en los que destaca la inexistencia del uso de los punteros ya que en C# no son necesarios, las clases y las funciones pueden ser declaradas sin antes ser definidas, en dado caso que así sea, no importa el orden en el que hayan sido definidas, a diferencia de C y C++ todo pertenece a una clase, no existen variables globales ni funciones, otro de los beneficios que ofrece C# ante lenguajes como C y C++ es que la recolección de basura es automática.

C# al ser un lenguaje más novedoso y con características mejoradas de Java, es capaz de ofrecer al programador la capacidad de declarar una gran cantidad de variables de tipo primitivas, y aunque sea limitado permite el uso de punteros.

No obstante, en C# se puede realizar una compilación adicional, además de que cuenta con

indicadores que permiten al programador poder acceder a un objeto como se haría con un simple array; y aplicaciones multihilos simplificadas y mayor tolerancia a la sobrecarga de operadores, así como los beneficios mencionados anteriormente, hacen de C# un lenguaje con diversos beneficios sobre lenguajes de alto dominio por la mayoría de los programadores, pero a su vez abre las puertas a muchos de estos ya que ofrece características que mejoran y facilitan el trabajo a los amantes de la programación.

#### Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de base de datos relacional (RDBMS) producido por Microsoft. Como servidor de base de datos, es un producto de software con la función principal de almacenar y recuperar datos según lo soliciten otras aplicaciones de software, que pueden ejecutarse en la misma computadora o en otra computadora a través de una red.

Algunas de las ventajas que nos ofrece y por el cual lo hemos seleccionado son:

● Soporte de transacciones.

● Escalabilidad, estabilidad y seguridad.

● Soporta procedimientos almacenados.

● Incluye también un potente entorno gráfico de administración, que permite el uso de

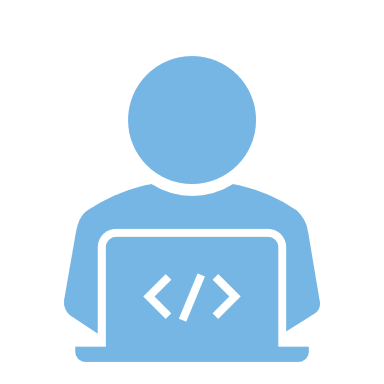
comandos DDL y DML gráficamente.

● Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el

servidor y las terminales o clientes de la red sólo acceden a la información.

Además, los servidores SQL Server suelen presentar como principal característica una alta disponibilidad al permitir un gran tiempo de actividad y una conmutación más rápida. Todo esto sin sacrificar los recursos de memoria del sistema. Gracias a las funciones de memoria integradas directamente en los motores de base de datos SQL Server y de análisis, mejora la flexibilidad y se facilita el uso. Pero quizá su característica más destacada es que ofrece una solución robusta que se integra a la perfección con la familia de servidores Microsoft Server.

## **CODIFICACIÓN**



### Codificación: avance en el desarrollo de Programas de Computación

#### CAPA DE DATOS

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class CadeteDAL

{

public DataTable BuscarCadetes()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCadeteS");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarCadeteYRodado()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCadeteS");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public void AsignarCadete(int pidpedido, int pidcadete)

{

SqlParameter[] parametros = new SqlParameter[2];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idpedido", pidpedido.ToString());

parametros[1] = objetoconexion.CrearParametro("@idcadete", pidcadete.ToString());

objetoconexion.EscribirPorStoreProcedure("SPAsignarCadete", parametros);

objetoconexion.Desconectar();

}

public void CrearCadete(string pnombre, string papellido, int ptelefono, int plicencia)

{

SqlParameter[] parametros = new SqlParameter[4];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@nombre", pnombre.ToString());

parametros[1] = objetoconexion.CrearParametro("@apellido", papellido.ToString());

parametros[2] = objetoconexion.CrearParametro("@tel", ptelefono.ToString());

parametros[3] = objetoconexion.CrearParametro("@licencia", plicencia.ToString());

objetoconexion.EscribirPorStoreProcedure("SPCrearCadete", parametros);

objetoconexion.Desconectar();

}

public DataTable BuscarCadetesReales()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCadetesReales");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public void ModificarCadete(int idcadete, string pnombre, string apellido, int tel, int licencia)

{

SqlParameter[] parametros = new SqlParameter[5];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idcadete", idcadete.ToString());

parametros[1] = objetoconexion.CrearParametro("@nombre", pnombre);

parametros[2] = objetoconexion.CrearParametro("@apellido ", apellido);

parametros[3] = objetoconexion.CrearParametro("@telefono", tel.ToString());

parametros[4] = objetoconexion.CrearParametro("@licencia", licencia.ToString());

objetoconexion.EscribirPorStoreProcedure("SPModificarCadete", parametros);

objetoconexion.Desconectar();

}

public void EliminarCadete(int pid)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idcadete", pid.ToString());

objetoconexion.EscribirPorStoreProcedure("SPEliminarCadete", parametros);

objetoconexion.Desconectar();

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class CategoriaProductoDAL

{

public DataTable BuscarCategoriaProducto()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCategoriaProducto");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class ClienteDAL

{

public DataTable BuscarClientes()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarClientes");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public void CrearCliente(string pnombre , string papellido, int ptelefono, string pdireccion)

{

SqlParameter[] parametros = new SqlParameter[4];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@nombre", pnombre.ToString());

parametros[1] = objetoconexion.CrearParametro("@apellido", papellido.ToString());

parametros[2] = objetoconexion.CrearParametro("@telefono", ptelefono.ToString());

parametros[3] = objetoconexion.CrearParametro("@direccion", pdireccion.ToString());

objetoconexion.EscribirPorStoreProcedure("SPCrearCliente", parametros);

objetoconexion.Desconectar();

}

public void EliminarCliente (int pidcliente)

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

objetoconexion.EscrituraPorComando("DELETE FROM Cliente WHERE idcliente = "+pidcliente);

objetoconexion.Desconectar();

}

public DataTable BuscarClientePorNombre (string pnombre)

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable tablacliente = objetoconexion.LecturaPorComando("SELECT \* FROM Cliente WHERE nombre LIKE '"+pnombre+"%'");

objetoconexion.Desconectar();

return tablacliente;

}

public void ModificarCliente(int pidcliente, string pnombre, string papellido, int ptelefono, string pdireccion)

{

SqlParameter[] parametros = new SqlParameter[5];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idcliente", pidcliente.ToString());

parametros[1] = objetoconexion.CrearParametro("@nombre", pnombre.ToString());

parametros[2] = objetoconexion.CrearParametro("@apellido", papellido.ToString());

parametros[3] = objetoconexion.CrearParametro("@telefono", ptelefono.ToString());

parametros[4] = objetoconexion.CrearParametro("@direccion", pdireccion.ToString());

objetoconexion.EscribirPorStoreProcedure("SPModificarCliente", parametros);

objetoconexion.Desconectar();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Data;

using System.Windows.Forms;

namespace CapaDeDatos

{

public class Conexion

{

private SqlConnection objetoconexion;

private string connectionstring = "";

public Conexion()

{

}

public SqlConnection Objetoconexion { get => objetoconexion; set => objetoconexion = value; }

public string Connectionstring { get => connectionstring; set => connectionstring = value; }

public void Conectar()

{

this.connectionstring = "Integrated Security = SSPI; Persist Security Info = False; Initial Catalog = Che Miranda; Data Source =.\\SQLEXPRESS";

this.objetoconexion = new SqlConnection();

this.objetoconexion.ConnectionString = Connectionstring;

this.objetoconexion.Open();

}

public void Desconectar()

{

this.objetoconexion.Close();

this.objetoconexion.Dispose();

}

public void EscrituraPorComando(string pcomando)

{

SqlCommand objComando = new SqlCommand();

try

{

objComando.CommandText = pcomando;

objComando.CommandType = CommandType.Text;

objComando.Connection = this.objetoconexion;

objComando.ExecuteNonQuery();

}

catch (Exception)

{

throw;

}

}

public DataTable LecturaPorComando(string pcomando)

{

DataTable objetotabla = new DataTable();

SqlCommand objetocomando = new SqlCommand();

try

{

objetocomando.CommandType = CommandType.Text;

objetocomando.Connection = this.objetoconexion;

objetocomando.CommandText = pcomando;

SqlDataAdapter objAdaptador = new SqlDataAdapter(objetocomando);

objAdaptador.Fill(objetotabla);

}

catch

{

objetotabla = null;

throw;

}

return objetotabla;

}

public DataTable LeerPorStoreProcedure(string pNombreStoreProcedure)

{

//Instancio un objeto del tipo DataTable

DataTable unaTabla = new DataTable();

//Instancio un objeto del tipo SqlCommand

SqlCommand objComando = new SqlCommand();

//Me conecto...

this.Conectar();

try

{

objComando.CommandText = pNombreStoreProcedure;

objComando.CommandType = CommandType.StoredProcedure;

objComando.Connection = this.objetoconexion;

//Instancio un adaptador con el parametro SqlCommand

SqlDataAdapter objAdaptador = new SqlDataAdapter(objComando);

//Lleno la tabla, el objeto unaTabla con el adaptador

objAdaptador.Fill(unaTabla);

}

catch (Exception)

{

//Como hay error... por el motivo que sea asigno el resultado a null

unaTabla = null;

throw;

}

finally

{

//Pase lo que pase me desconecto

this.Desconectar();

}

return unaTabla;

}

public SqlParameter CrearParametro(string pNombre, string pValor)

{

SqlParameter objParametro = new SqlParameter();

objParametro.ParameterName = pNombre;

objParametro.Value = pValor;

objParametro.DbType = DbType.String;

return objParametro;

}

public int EscribirPorStoreProcedure(string pTexto, SqlParameter[] pParametrosSql)

{

//Instanció una variable filasAfectadas que va a terminar devolviendo la cantidad de filas afectadas.

int filasAfectadas = 0;

//Instancio un objeto del tipo SqlCommand

var objComando = new SqlCommand();

//Me conecto...

this.Conectar();

try

{

objComando.CommandText = pTexto;

objComando.CommandType = CommandType.StoredProcedure;

objComando.Connection = this.objetoconexion;

if (pParametrosSql.Length > 0)

{

objComando.Parameters.AddRange(pParametrosSql);

//El método ExecuteNonQuery() me devuelve la cantidad de filas afectadas.

filasAfectadas = objComando.ExecuteNonQuery();

}

else

{

//retorno -1 porque la lista de parametros Sql tiene 0 ítems...

filasAfectadas = -1;

}

}

catch (Exception)

{

filasAfectadas = -1;

throw;

}

finally

{

//Me desconecto

this.Desconectar();

}

return filasAfectadas;

}

public DataTable LeerPorStoreProcedureConParametros(string pNombreStoreProcedure, SqlParameter[] pParametrosSql)

{

int filasAfectadas = 0;

DataTable unaTabla = new DataTable();

//Instancio un objeto del tipo SqlCommand

SqlCommand objComando = new SqlCommand();

//Me conecto...

this.Conectar();

try

{

objComando.CommandText = pNombreStoreProcedure;

objComando.CommandType = CommandType.StoredProcedure;

objComando.Connection = this.objetoconexion;

if (pParametrosSql.Length > 0)

{

objComando.Parameters.AddRange(pParametrosSql);

//El método ExecuteNonQuery() me devuelve la cantidad de filas afectadas.

//filasAfectadas = objComando.ExecuteNonQuery();

}

else

{

//retorno -1 porque la lista de parametros Sql tiene 0 ítems...

filasAfectadas = -1;

}

//Instancio un adaptador con el parametro SqlCommand

SqlDataAdapter objAdaptador = new SqlDataAdapter(objComando);

//Lleno la tabla, el objeto unaTabla con el adaptador

objAdaptador.Fill(unaTabla);

filasAfectadas = objComando.ExecuteNonQuery();

}

catch (Exception)

{

//Como hay error... por el motivo que sea asigno el resultado a null

unaTabla = null;

throw;

}

finally

{

//Pase lo que pase me desconecto

this.Desconectar();

}

return unaTabla;

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class DetallePedidoDAL

{

public void CrearDetallePedido(int pidproducto, int pidpedido, int pcantidad, int pprecioventa)

{

SqlParameter[] parametros = new SqlParameter[4];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idproducto", pidproducto.ToString());

parametros[1] = objetoconexion.CrearParametro("@idpedido", pidpedido.ToString());

parametros[2] = objetoconexion.CrearParametro("@cantidad", pcantidad.ToString());

parametros[3] = objetoconexion.CrearParametro("@precioventa", pprecioventa.ToString());

objetoconexion.EscribirPorStoreProcedure("SPCrearDetallePedido", parametros);

objetoconexion.Desconectar();

}

public DataTable getReferenciaIdPedido()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LecturaPorComando("select top 1 idpedido as MayorID from pedido order by idpedido desc");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarDetallePedidoSegunPedido(int pidpedido)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idpedido", pidpedido.ToString());

DataTable objetotabla = objetoconexion.LeerPorStoreProcedureConParametros("SPBuscarDetallePedidoSegunPedido", parametros);

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class EstadoCategoriaDAL

{

public DataTable BuscarCategorias()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEstadosCategorias");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class FormaDePagoDAL

{

public DataTable BuscarFormasDePago()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarFormasDePago");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class PedidoDAL

{

public int CrearPedido(int pidformadepago, int pidcliente, int pidusuario,

string pobservaciones, int pidcadete, int pidestadocategoria, int ptotal)

{

int Filasafectadas = 1;

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

objetoconexion.EscrituraPorComando("INSERT INTO Pedido (idformadepago, idcliente, idusuario, fechayhora, observaciones, idcadete, idestadocategoria, total) VALUES (" +

pidformadepago + "," + pidcliente + "," + pidusuario + ", CURRENT\_TIMESTAMP ,'" + pobservaciones + "'," + pidcadete + ", "+ pidestadocategoria + "," + ptotal+ ")");

objetoconexion.Desconectar();

return Filasafectadas;

}

public DataTable BuscarPedidosActivos()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPedidosActivos");

objetoconexion.Desconectar();

return objetotabla;

}

public void ModificarEstadoPedido(int pidpedido, int pidestadocategoria)

{

SqlParameter[] parametros = new SqlParameter[2];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idpedido", pidpedido.ToString());

parametros[1] = objetoconexion.CrearParametro("@idestadocategoria", pidestadocategoria.ToString());

objetoconexion.EscribirPorStoreProcedure("SPModificarEstadoPedido", parametros);

objetoconexion.Desconectar();

}

public DataTable BuscarPedidosSegunCliente(int pidcliente)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idcliente", pidcliente.ToString());

DataTable objetotabla = objetoconexion.LeerPorStoreProcedureConParametros("SPBuscarPedidosSegunCliente", parametros);

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPedidosdeldia()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPedidopordia");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPedidosSegunMesAño(int mes, int año)

{

SqlParameter[] parametros = new SqlParameter[2];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@mes", mes.ToString());

parametros[1] = objetoconexion.CrearParametro("@año", año.ToString());

DataTable objetotabla = objetoconexion.LeerPorStoreProcedureConParametros("BuscarPedidoporMesAño", parametros);

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class ProductoDAL

{

public DataTable BuscarEntradas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEntradas");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarOmelletes ()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarOmelletes");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarTortillas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarTortillas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarMilanesas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarMilanesas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarSupremas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarSupremas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarMatambre()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarMatambre");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarGuarnicion()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarGuarnicion");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPollos()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPollos");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarParillas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarParillas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPescados()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPescados");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarEnsaladas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEnsaladas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarEnsaladasEsp()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEnsaladasEsp");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarTradicionales()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarTradicionales");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarEspecialesI()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEspecialesI");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarEspecialesII()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEspecialesII");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarFugazzas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarFugazzas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarFugazzetas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarFugazzetas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarCalzones()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCalzones");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarOtrasEspecialidades()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarOtrasEspecialidades");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarEmpanadas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarEmpanadas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarSandwiches()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarSandwiches");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPastas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPastas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarBebidas()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarBebidas");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPostres()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPostres");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarMenus()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarMenus");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarPromos()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarPromos");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarTodo()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscartodoslosproductos");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable FiltroProductos(string pfiltro)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@filtro", pfiltro);

DataTable objetotabla = objetoconexion.LeerPorStoreProcedureConParametros("SPFiltroProductoPorNombreYCategoria", parametros);

objetoconexion.Desconectar();

return objetotabla;

}

public void CrearProducto(string pdescripcion, int idproductocategoria, int pprecio)

{

SqlParameter[] parametros = new SqlParameter[3];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@descripcion", pdescripcion.ToString());

parametros[1] = objetoconexion.CrearParametro("@idproductocategoria", idproductocategoria.ToString());

parametros[2] = objetoconexion.CrearParametro("@precio", pprecio.ToString());

objetoconexion.EscribirPorStoreProcedure("SPCrearProducto", parametros);

objetoconexion.Desconectar();

}

public void ModificarProducto(int pidproducto, string pnombre, int categoria, int precio)

{

SqlParameter[] parametros = new SqlParameter[4];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idproducto", pidproducto.ToString());

parametros[1] = objetoconexion.CrearParametro("@nombre", pnombre);

parametros[2] = objetoconexion.CrearParametro("@idproductocategoria ", categoria.ToString());

parametros[3] = objetoconexion.CrearParametro("@precio", precio.ToString());

objetoconexion.EscribirPorStoreProcedure("SPModificarProducto", parametros);

objetoconexion.Desconectar();

}

public void EliminarProducto(int pidproducto)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idproducto", pidproducto.ToString());

objetoconexion.EscribirPorStoreProcedure("SPEliminarProducto", parametros);

objetoconexion.Desconectar();

}

public DataTable BuscarProductoMasVendidodia()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarProductoMasVendidodia");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarProductoMasVendidoMesAño(int mes, int año)

{

SqlParameter[] parametros = new SqlParameter[2];

Conexion objetoconexion = new Conexion();

parametros[0] = objetoconexion.CrearParametro("@mes", mes.ToString());

parametros[1] = objetoconexion.CrearParametro("@año", año.ToString());

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedureConParametros("SPBuscarProductoMasVendidoMesAño", parametros);

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class UsuarioCategoriaDAL

{

public DataTable BuscarUsuarioCategorias()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarCategoriasUsuario");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeDatos

{

public class UsuarioDAL

{

public DataTable BuscarUsuario()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarUsuario");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public DataTable BuscarUsuariosincontraseña()

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

DataTable objetotabla = objetoconexion.LeerPorStoreProcedure("SPBuscarUsuariosSinContraseña");

//DataTable objetotabla = objetoconexion.LecturaPorComando("SELECT \* FROM Producto WHERE idproductocategoria = 1");

objetoconexion.Desconectar();

return objetotabla;

}

public void CrearUsuario(string pnombre,string papellido,int cuil , int contraseña, int idcategoriausuario)

{

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

objetoconexion.EscrituraPorComando("insert into Usuario(nombre, apellido, CUIL, contraseña, fechadecontratacion, idcategoriausuario)" +

"values('"+ pnombre+"', '" + papellido + "', "+ cuil + ", "+ contraseña + ", CURRENT\_TIMESTAMP,"+ idcategoriausuario + ")");

objetoconexion.Desconectar();

}

public void ModificarUsuario(int pidusuario, string pnombre, string apellido, int CUIL, int contraseña, int idcategoria)

{

SqlParameter[] parametros = new SqlParameter[6];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idusuario", pidusuario.ToString());

parametros[1] = objetoconexion.CrearParametro("@nombre", pnombre);

parametros[2] = objetoconexion.CrearParametro("@apellido ", apellido);

parametros[3] = objetoconexion.CrearParametro("@cuil", CUIL.ToString());

parametros[4] = objetoconexion.CrearParametro("@contraseña", contraseña.ToString());

parametros[5] = objetoconexion.CrearParametro("@idcategoriausuario", idcategoria.ToString());

objetoconexion.EscribirPorStoreProcedure("SPModificarUsuario", parametros);

objetoconexion.Desconectar();

}

public void EliminarUsuario(int pid)

{

SqlParameter[] parametros = new SqlParameter[1];

Conexion objetoconexion = new Conexion();

objetoconexion.Conectar();

parametros[0] = objetoconexion.CrearParametro("@idusuario", pid.ToString());

objetoconexion.EscribirPorStoreProcedure("SPEliminarUsuario", parametros);

objetoconexion.Desconectar();

}

}

}

#### CAPA DE NEGOCIOS

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class CadeteBLL

{

private int idcadete;

private string nombre;

private string apellido;

private int telefono;

private RodadoBLL rodado;

public CadeteBLL(int pidcadete, string pnombre, string papellido, int ptelefono, RodadoBLL prodado)

{

this.Idcadete = pidcadete;

this.Nombre = pnombre;

this.Apellido = papellido;

this.Telefono = ptelefono;

this.Rodado = prodado;

}

public CadeteBLL(int pidcadete, string pnombre, string papellido, int ptelefono)

{

this.Idcadete = pidcadete;

this.Nombre = pnombre;

this.Apellido = papellido;

this.Telefono = ptelefono;

}

public CadeteBLL ()

{

}

public int Idcadete { get => idcadete; set => idcadete = value; }

public string Nombre { get => nombre; set => nombre = value; }

public string Apellido { get => apellido; set => apellido = value; }

public int Telefono { get => telefono; set => telefono = value; }

public RodadoBLL Rodado { get => rodado; set => rodado = value; }

public List<CadeteBLL> CargarCadeteYRodado()

{

CapaDeDatos.CadeteDAL cadetedal = new CapaDeDatos.CadeteDAL();

DataTable tablacadeteyrodado = cadetedal.BuscarCadeteYRodado();

List<CadeteBLL> cadeteyrodadoacargar = new List<CadeteBLL>();

foreach (DataRow fila in tablacadeteyrodado.Rows)

{

int idcadete;

string nombre;

string apellido;

int telefono;

idcadete = int.Parse(fila["idcadete"].ToString());

nombre = fila["nombre"].ToString();

apellido = fila["apellido"].ToString();

telefono = int.Parse(fila["telefono"].ToString());

int idrodado;

string descripcion;

string patente;

idrodado = int.Parse(fila["idrodado"].ToString());

descripcion = fila["descripcion"].ToString();

patente = fila["patente"].ToString();

RodadoBLL rodado = new RodadoBLL(idrodado, descripcion, patente);

CadeteBLL cadete = new CadeteBLL(idcadete, nombre, apellido, telefono, rodado);

cadeteyrodadoacargar.Add(cadete);

}

return cadeteyrodadoacargar;

}

public List<CadeteBLL> CargarCadetes()

{

CapaDeDatos.CadeteDAL cadetedal = new CapaDeDatos.CadeteDAL();

DataTable tabladecadetes = cadetedal.BuscarCadetes();

List<CadeteBLL> cadetesaagregar = new List<CadeteBLL>();

foreach (DataRow fila in tabladecadetes.Rows)

{

int idcadete;

string nombre;

string apellido;

int telefono;

idcadete = int.Parse(fila["idcadete"].ToString());

nombre = fila["nombre"].ToString();

apellido = fila["apellido"].ToString();

telefono = int.Parse(fila["telefono"].ToString());

CadeteBLL cadete = new CadeteBLL(idcadete, nombre, apellido, telefono);

cadetesaagregar.Add(cadete);

}

return cadetesaagregar;

}

public void AsignarCadete (int pidpedido, int pidcadete)

{

CadeteDAL cadete = new CadeteDAL();

cadete.AsignarCadete(pidpedido, pidcadete);

}

public DataTable CargartablaCadetes()

{

CapaDeDatos.CadeteDAL cadete = new CadeteDAL();

DataTable tablacadete = cadete.BuscarCadetesReales();

return tablacadete;

}

public void CrearCadete(string pnombre, string papellido, int ptelefono, int licencia)

{

CapaDeDatos.CadeteDAL cadete = new CapaDeDatos.CadeteDAL();

cadete.CrearCadete(pnombre, papellido, ptelefono, licencia);

}

public void ModificarCadete(int pidusuario, string pnombre, string apellido, int tel, int licencia)

{

CadeteDAL cadete = new CadeteDAL();

cadete.ModificarCadete(pidusuario, pnombre, apellido, tel, licencia);

}

public void EliminarCadete(int pid)

{

CadeteDAL cadete = new CadeteDAL();

cadete.EliminarCadete(pid);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class CategoriaProductoBLL

{

private int idcategoriaproducto;

private string nombre;

public CategoriaProductoBLL(int pidcategoriaproducto, string pnombre)

{

this.Idcategoriaproducto = pidcategoriaproducto;

this.Nombre = pnombre;

}

public CategoriaProductoBLL()

{

}

public int Idcategoriaproducto { get => idcategoriaproducto; set => idcategoriaproducto = value; }

public string Nombre { get => nombre; set => nombre = value; }

public List<CategoriaProductoBLL> CargarProductoCategoria()

{

CapaDeDatos.CategoriaProductoDAL productocategoriadal = new CapaDeDatos.CategoriaProductoDAL();

DataTable tablacategoriaproductos = productocategoriadal.BuscarCategoriaProducto();

List<CategoriaProductoBLL> categoriaproductosacargar = new List<CategoriaProductoBLL>();

foreach (DataRow fila in tablacategoriaproductos.Rows)

{

int idcategoriaproducto;

string descripcion;

idcategoriaproducto = int.Parse(fila["ID del Producto"].ToString());

descripcion = fila["Nombre del producto"].ToString();

CategoriaProductoBLL categoriaproduc = new CategoriaProductoBLL(idcategoriaproducto, descripcion);

categoriaproductosacargar.Add(categoriaproduc);

}

return categoriaproductosacargar;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class ClienteBLL

{

private int idcliente;

private string nombre;

private string apellido;

private int telefono;

private string direccion;

public ClienteBLL (int pidcliente, string pnombre, string papellido, int ptelefono, string pdireccion)

{

this.Idcliente = pidcliente;

this.Nombre = pnombre;

this.Apellido = papellido;

this.Telefono = ptelefono;

this.Direccion = pdireccion;

}

public ClienteBLL ()

{

}

public int Idcliente { get => idcliente; set => idcliente = value; }

public string Nombre { get => nombre; set => nombre = value; }

public string Apellido { get => apellido; set => apellido = value; }

public int Telefono { get => telefono; set => telefono = value; }

public string Direccion { get => direccion; set => direccion = value; }

public List<ClienteBLL> CargarClientes()

{

CapaDeDatos.ClienteDAL clientedal = new CapaDeDatos.ClienteDAL();

DataTable tablaclientes= clientedal.BuscarClientes();

List<ClienteBLL> clientesacargar = new List<ClienteBLL>();

foreach (DataRow fila in tablaclientes.Rows)

{

int idcliente;

string nombre;

string apellido;

int telefono;

string direccion;

idcliente = int.Parse(fila["idcliente"].ToString());

nombre = fila["nombre"].ToString();

apellido = fila["apellido"].ToString();

telefono = int.Parse(fila["telefono"].ToString());

direccion = fila["direccion"].ToString();

ClienteBLL cliente = new ClienteBLL(idcliente, nombre, apellido, telefono, direccion);

clientesacargar.Add(cliente);

}

return clientesacargar;

}

public DataTable CargarClientesDataTable()

{

CapaDeDatos.ClienteDAL clientedal = new CapaDeDatos.ClienteDAL();

DataTable tablaclientes = clientedal.BuscarClientes();

return tablaclientes;

}

public void CrearCliente(string pnombre, string papellido, int ptelefono, string pdireccion)

{

CapaDeDatos.ClienteDAL cliente = new CapaDeDatos.ClienteDAL();

cliente.CrearCliente(pnombre,papellido,ptelefono,pdireccion);

}

public void EliminarCliente (int pidcliente)

{

ClienteDAL cliente = new ClienteDAL();

cliente.EliminarCliente(pidcliente);

}

public DataTable BuscarClientePorNombre (string pnombre)

{

ClienteDAL cliente = new ClienteDAL();

DataTable tablacliente = cliente.BuscarClientePorNombre(pnombre);

return tablacliente;

}

public void ModificarCliente (int pidcliente, string pnombre, string papellido, int ptelefono, string pdireccion)

{

ClienteDAL cliente = new ClienteDAL();

cliente.ModificarCliente(pidcliente, pnombre, papellido, ptelefono, pdireccion);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class DetallePedidoBLL

{

private int iddetallepedido;

private int idproducto;

private int idpedido;

private string producto;

private int cantidad;

private int precioventa;

private string informacion;

private DataTable referenciaIdPedido;

public DetallePedidoBLL()

{

}

public DetallePedidoBLL (int pidproducto, int pcantidad, int pprecioventa, string pinformacion)

{

this.idproducto = pidproducto;

this.cantidad = pcantidad;

this.precioventa = pprecioventa;

this.informacion = pinformacion;

}

public DetallePedidoBLL (int piddetallepedido, int pidpedido, string pproducto, int pcantidad, int pprecioventa)

{

this.iddetallepedido = piddetallepedido;

this.idpedido = pidpedido;

this.producto = pproducto;

this.cantidad = pcantidad;

this.precioventa = pprecioventa;

}

public int Iddetallepedido { get => iddetallepedido; set => iddetallepedido = value; }

public int Idproducto { get => idproducto; set => idproducto = value; }

public int Idpedido { get => idpedido; set => idpedido = value; }

public string Producto { get => producto; set => producto = value; }

public int Cantidad { get => cantidad; set => cantidad = value; }

public int Precioventa { get => precioventa; set => precioventa = value; }

public string Informacion { get => informacion; set => informacion = value; }

public DataTable ReferenciaIdPedido { get => referenciaIdPedido; set => referenciaIdPedido = value; }

public int CrearDetallePedido()

{

int cantidaddefilarafectadas = 0;

CapaDeDatos.DetallePedidoDAL detallepedido = new CapaDeDatos.DetallePedidoDAL();

detallepedido.CrearDetallePedido(this.idproducto, this.idpedido, this.cantidad, this.precioventa);

return cantidaddefilarafectadas;

}

public DataTable getReferenciaIdPedido ()

{

CapaDeDatos.DetallePedidoDAL detallepedido = new CapaDeDatos.DetallePedidoDAL();

this.referenciaIdPedido = detallepedido.getReferenciaIdPedido();

return this.referenciaIdPedido;

}

public List<DetallePedidoBLL> CargarDetallePedidoSegunPedido (int pidpedido)

{

DetallePedidoDAL detallepedidodal = new DetallePedidoDAL();

DataTable tabladetallepedidos = detallepedidodal.BuscarDetallePedidoSegunPedido(pidpedido);

List<DetallePedidoBLL> detallespedidosaagregar = new List<DetallePedidoBLL>();

foreach (DataRow fila in tabladetallepedidos.Rows)

{

int iddetallepedido = int.Parse(fila["ID Detalle Pedido"].ToString());

int idpedido = int.Parse(fila["ID Pedido"].ToString());

string producto = fila["Producto"].ToString();

int cantidad = int.Parse(fila["Cantidad"].ToString());

int precioventa = int.Parse(fila["Precio Venta"].ToString());

DetallePedidoBLL detallepedido = new DetallePedidoBLL(iddetallepedido, idpedido, producto, cantidad, precioventa);

detallespedidosaagregar.Add(detallepedido);

}

return detallespedidosaagregar;

}

public DataTable CargarTablaDetallePedidoSegunPedido (int pidpedido)

{

DetallePedidoDAL detallepedidodal = new DetallePedidoDAL();

DataTable tabladetallepedidos = detallepedidodal.BuscarDetallePedidoSegunPedido(pidpedido);

return tabladetallepedidos;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeNegocios

{

public class EstadoCategoriaBLL

{

private int idestadocategoria;

private string descripcion;

public EstadoCategoriaBLL ()

{

}

public EstadoCategoriaBLL (int pidestadocategoria, string pdescripcion)

{

this.Idestadocategoria = pidestadocategoria;

this.Descripcion = pdescripcion;

}

public int Idestadocategoria { get => idestadocategoria; set => idestadocategoria = value; }

public string Descripcion { get => descripcion; set => descripcion = value; }

public List<EstadoCategoriaBLL> CargarEstadosCategoria()

{

CapaDeDatos.EstadoCategoriaDAL objetoestadocategoria = new CapaDeDatos.EstadoCategoriaDAL();

DataTable tablaestadoscategoria = objetoestadocategoria.BuscarCategorias();

List<EstadoCategoriaBLL> estadoscategoriasaagregar = new List<EstadoCategoriaBLL>();

foreach (DataRow fila in tablaestadoscategoria.Rows)

{

int idestadocategoria;

string descripcion;

idestadocategoria = int.Parse(fila["idestadocategoria"].ToString());

descripcion = fila["descripcion"].ToString();

EstadoCategoriaBLL estadocategoria = new EstadoCategoriaBLL(idestadocategoria, descripcion);

estadoscategoriasaagregar.Add(estadocategoria);

}

return estadoscategoriasaagregar;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeNegocios

{

public class FormaDePagoBLL

{

private int idformadepago;

private string descripcion;

public FormaDePagoBLL (int pidformadepago, string pdescripcion)

{

this.idformadepago = pidformadepago;

this.descripcion = pdescripcion;

}

public FormaDePagoBLL ()

{

}

public int Idformadepago { get => idformadepago; set => idformadepago = value; }

public string Descripcion { get => descripcion; set => descripcion = value; }

public List<FormaDePagoBLL> CargarFormasDePago()

{

CapaDeDatos.FormaDePagoDAL formadepagodal = new CapaDeDatos.FormaDePagoDAL();

DataTable tablaformasdepago = formadepagodal.BuscarFormasDePago();

List<FormaDePagoBLL> formasdepagoaagregar = new List<FormaDePagoBLL>();

foreach (DataRow fila in tablaformasdepago.Rows)

{

int idformadepago;

string descripcion;

idformadepago = int.Parse(fila["idformadepago"].ToString());

descripcion = fila["descripcion"].ToString();

FormaDePagoBLL formadepago = new FormaDePagoBLL(idformadepago, descripcion);

formasdepagoaagregar.Add(formadepago);

}

return formasdepagoaagregar;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class PedidoBLL

{

private int idformadepago;

private int idcliente;

private int idusuario;

private string observaciones;

private int idcadete;

private int idestadocategoria;

private int total;

public PedidoBLL()

{

}

public int Idformadepago { get => idformadepago; set => idformadepago = value; }

public int Idcliente { get => idcliente; set => idcliente = value; }

public int Idusuario { get => idusuario; set => idusuario = value; }

public string Observaciones { get => observaciones; set => observaciones = value; }

public int Idcadete { get => idcadete; set => idcadete = value; }

public int Idestadocategoria { get => idestadocategoria; set => idestadocategoria = value; }

public int Total { get => total; set => total = value; }

public int CrearPedido()

{

int cantidaddefilarafectadas = 0;

CapaDeDatos.PedidoDAL pedido = new CapaDeDatos.PedidoDAL();

cantidaddefilarafectadas = pedido.CrearPedido(this.Idformadepago, this.Idcliente, this.Idusuario, this.Observaciones, this.Idcadete, this.Idestadocategoria, this.Total);

return cantidaddefilarafectadas;

}

public DataTable CargarPedidosActivos()

{

CapaDeDatos.PedidoDAL pedidodal = new CapaDeDatos.PedidoDAL();

DataTable tablapedidosactivos = pedidodal.BuscarPedidosActivos();

return tablapedidosactivos;

}

public void ModificarEstadoPedido (int idpedido, int idestadocategoria)

{

PedidoDAL pedido = new PedidoDAL();

pedido.ModificarEstadoPedido(idpedido, idestadocategoria);

}

public DataTable CargarPedidosSegunCliente (int pidcliente)

{

CapaDeDatos.PedidoDAL pedidodal = new CapaDeDatos.PedidoDAL();

DataTable tablapedidos = pedidodal.BuscarPedidosSegunCliente(pidcliente);

return tablapedidos;

}

public DataTable CargarPedidosdeldia()

{

CapaDeDatos.PedidoDAL pedidodal = new CapaDeDatos.PedidoDAL();

DataTable tablapedidosactivos = pedidodal.BuscarPedidosdeldia();

return tablapedidosactivos;

}

public DataTable CargarPedidosporMesAño(int mes, int año)

{

CapaDeDatos.PedidoDAL pedidodal = new CapaDeDatos.PedidoDAL();

DataTable tablapedidosactivos = pedidodal.BuscarPedidosSegunMesAño(mes, año);

return tablapedidosactivos;

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class ProductoBLL

{

private int idproducto;

private string descripcion;

private int precio;

private int idcategoriaproducto; // Agregar despues el tipo de dato categoriaproducto

public ProductoBLL(int pidproducto, string pdescripcion,

int pprecio, int pidcategoriaproducto)

{

this.Idproducto = pidproducto;

this.Descripcion = pdescripcion;

this.Precio = pprecio;

this.idcategoriaproducto = pidcategoriaproducto;

}

public ProductoBLL ()

{

}

public int Idproducto { get => idproducto; set => idproducto = value; }

public string Descripcion { get => descripcion; set => descripcion = value; }

public int Precio { get => precio; set => precio = value; }

public int Idcategoriaproducto { get => idcategoriaproducto; set => idcategoriaproducto = value; }

public List<ProductoBLL> CargarEntradas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEntradas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarOmelletes ()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarOmelletes();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarTortillas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarTortillas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarMilanesas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarMilanesas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarSupremas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarSupremas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarMatambre()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarMatambre();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarGuarnicion()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarGuarnicion();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarPollos()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarPollos();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarParillas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarParillas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarPescados()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarPescados();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarEnsaladas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEnsaladas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarEnsaladasEsp()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEnsaladasEsp();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarTradicionales()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarTradicionales();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarEspecialesI()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEspecialesI();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarEspecialesII()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEspecialesII();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarFugazzas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarFugazzas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarFugazzetas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarFugazzetas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarCalzones()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarCalzones();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarOtrasEspecialidades()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarOtrasEspecialidades();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarEmpanadas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarEmpanadas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarSandwiches()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarSandwiches();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarPastas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarPastas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarBebidas()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarBebidas();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarPostres()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarPostres();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarMenus()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarMenus();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public List<ProductoBLL> CargarPromos()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarPromos();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

foreach (DataRow fila in tablaproductos.Rows)

{

int idproducto;

string descripcion;

int precio;

int idcategoriaproducto;

idproducto = int.Parse(fila["idproducto"].ToString());

descripcion = fila["descripcion"].ToString();

precio = int.Parse(fila["precio"].ToString());

idcategoriaproducto = int.Parse(fila["idproductocategoria"].ToString());

ProductoBLL producto = new ProductoBLL(idproducto, descripcion, precio, idcategoriaproducto);

productosacargar.Add(producto);

}

return productosacargar;

}

public DataTable CargarTodolosproductos()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarTodo();

return tablaproductos;

}

public DataTable FIltroProductos(string pfiltro)

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.FiltroProductos(pfiltro);

return tablaproductos;

}

public void CrearProducto(string pdescripcion, int idproductocategoria, int pprecio)

{

CapaDeDatos.ProductoDAL producto = new CapaDeDatos.ProductoDAL();

producto.CrearProducto(pdescripcion, idproductocategoria, pprecio );

}

public void ModificarProducto (int pidproducto, string pnombre, int idcategoria, int precio)

{

ProductoDAL producto = new ProductoDAL();

producto.ModificarProducto(pidproducto, pnombre, idcategoria, precio);

}

public void EliminarProducto(int pidproducto)

{

ProductoDAL producto = new ProductoDAL();

producto.EliminarProducto(pidproducto);

}

public DataTable CargarProductoMasVendidoDia()

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarProductoMasVendidodia();

return tablaproductos;

}

public DataTable CargarProductoMasVendidoMesAño(int mes, int año)

{

CapaDeDatos.ProductoDAL productodal = new CapaDeDatos.ProductoDAL();

DataTable tablaproductos = productodal.BuscarProductoMasVendidoMesAño(mes, año);

return tablaproductos;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CapaDeNegocios

{

public class RodadoBLL

{

private int idrodado;

private string descripcion;

private string patente;

public RodadoBLL(int pidrodado, string pdescripcion, string ppatente)

{

this.Idrodado = pidrodado;

this.Descripcion = pdescripcion;

this.Patente = ppatente;

}

public RodadoBLL ()

{

}

public int Idrodado { get => idrodado; set => idrodado = value; }

public string Descripcion { get => descripcion; set => descripcion = value; }

public string Patente { get => patente; set => patente = value; }

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class UsuarioBLL

{

private int legajo;

private string nombre;

private string apellido;

private int cuil;

private int contraseña;

private List<DateTime> diastrabajados;

private DateTime fechadecontratacion;

private int idusuariocategoria;

string usuariocategoria;

public UsuarioBLL()

{

}

public UsuarioBLL(int pid, string pNombre, string pApellido, int pCuil, int pContraseña, DateTime pFechadeContratacion, int pidcategoriausuario)

{

this.legajo = pid;

this.nombre = pNombre;

this.apellido = pApellido;

this.cuil = pCuil;

this.contraseña = pContraseña;

this.fechadecontratacion = pFechadeContratacion;

this.idusuariocategoria = pidcategoriausuario;

}

public UsuarioBLL(int pid, string pNombre, string pApellido, int pCuil, DateTime pFechadeContratacion, string pcategoriausuario)

{

this.legajo = pid;

this.nombre = pNombre;

this.apellido = pApellido;

this.cuil = pCuil;

this.fechadecontratacion = pFechadeContratacion;

this.usuariocategoria = pcategoriausuario;

}

public int Legajo { get => legajo; set => legajo = value; }

public string Nombre { get => nombre; set => nombre = value; }

public string Apellido { get => apellido; set => apellido = value; }

public int Cuil { get => cuil; set => cuil = value; }

public int Contraseña { get => contraseña; set => contraseña = value; }

public List<DateTime> Diastrabajados { get => diastrabajados; set => diastrabajados = value; }

public DateTime Fechadecontratacion { get => fechadecontratacion; set => fechadecontratacion = value; }

public int Idusuariocategoria { get => idusuariocategoria; set => idusuariocategoria = value; }

public string Usuariocategoria { get => usuariocategoria; set => usuariocategoria = value; }

public List<UsuarioBLL> CargarUsuarios()

{

CapaDeDatos.UsuarioDAL usuario = new CapaDeDatos.UsuarioDAL();

DataTable tablausuario = usuario.BuscarUsuario();

List<UsuarioBLL> listausuarios = new List<UsuarioBLL>();

foreach (DataRow fila in tablausuario.Rows)

{

int idusuario;

string Nombre;

string Apellido;

int Cuil;

int Contraseña;

DateTime Fechacontratacion;

int Idcategoriausuario;

idusuario = int.Parse(fila["idusuario"].ToString());

Nombre = fila["nombre"].ToString();

Apellido = fila["apellido"].ToString();

Cuil = int.Parse(fila["CUIL"].ToString());

Contraseña = int.Parse(fila["contraseña"].ToString());

Fechacontratacion = DateTime.Parse(fila["fechadecontratacion"].ToString());

Idcategoriausuario = int.Parse(fila["idcategoriausuario"].ToString());

UsuarioBLL nuevousuario = new UsuarioBLL(idusuario, Nombre, Apellido, Cuil, Contraseña, Fechacontratacion, Idcategoriausuario);

listausuarios.Add(nuevousuario);

}

return listausuarios;

}

public DataTable CargarUsuariossincontraseña()

{

CapaDeDatos.UsuarioDAL usuario = new CapaDeDatos.UsuarioDAL();

DataTable tablausuario = usuario.BuscarUsuariosincontraseña();

return tablausuario;

}

public void CrearUsuario(string nombre,string apellido, int CUIL, int contraseña, int idcategoria)

{

CapaDeDatos.UsuarioDAL usuario = new CapaDeDatos.UsuarioDAL();

usuario.CrearUsuario(nombre, apellido, CUIL, contraseña, idcategoria);

}

public void ModificarUsuario(int pidusuario, string pnombre, string apellido, int CUIl, int contraseña , int idcategoria)

{

UsuarioDAL usuario = new UsuarioDAL();

usuario.ModificarUsuario(pidusuario, pnombre, apellido, CUIl, contraseña, idcategoria);

}

public void EliminarUsuario (int pid)

{

UsuarioDAL usuario = new UsuarioDAL();

usuario.EliminarUsuario(pid);

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CapaDeDatos;

namespace CapaDeNegocios

{

public class UsuarioCategoriaBLL

{

private int idcategoria;

private string descripcion;

public UsuarioCategoriaBLL()

{

}

public UsuarioCategoriaBLL(int pidcategoria, string pdescripcion)

{

this.Idcategoria = pidcategoria;

this.Descripcion = pdescripcion;

}

public int Idcategoria { get => idcategoria; set => idcategoria = value; }

public string Descripcion { get => descripcion; set => descripcion = value; }

public List<UsuarioCategoriaBLL> CargarUsuarioCategoria()

{

UsuarioCategoriaDAL categoria = new UsuarioCategoriaDAL();

DataTable tablaUsuariocategoria = categoria.BuscarUsuarioCategorias();

List<UsuarioCategoriaBLL> categoriasaagregar = new List<UsuarioCategoriaBLL>();

foreach (DataRow fila in tablaUsuariocategoria.Rows)

{

int idcategoria;

string descripcion;

idcategoria = int.Parse(fila["idcategoriausuario"].ToString());

descripcion = fila["descripcion"].ToString();

UsuarioCategoriaBLL Usuariocategoria = new UsuarioCategoriaBLL(idcategoria, descripcion);

categoriasaagregar.Add(Usuariocategoria);

}

return categoriasaagregar;

}

}

}

#### CAPA DE USUARIO

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class FormularioVerDetallePedido : Form

{

private List<DetallePedidoBLL> listadetallespedidos;

public FormularioVerDetallePedido(List<DetallePedidoBLL> plistadetallespedidos)

{

InitializeComponent();

this.listadetallespedidos = plistadetallespedidos;

}

public List<DetallePedidoBLL> Listadetallespedidos { get => listadetallespedidos; set => listadetallespedidos = value; }

private void FormularioVerDetallePedido\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

this.BackColor = Color.FromArgb(20, 34, 56);

this.label1.ForeColor = Color.White;

this.BotonAceptar.ForeColor = Color.White;

this.BotonAceptar.BackColor = Color.FromArgb(98, 104, 140);

foreach (DetallePedidoBLL detallepedido in this.listadetallespedidos)

{

this.Height = this.Height + 65;

}

foreach (DetallePedidoBLL detallepedido in this.listadetallespedidos)

{

this.label1.Text = this.label1.Text + "ID Detalle Pedido: " + detallepedido.Iddetallepedido + "\nProducto: " +

detallepedido.Producto + "\nCantidad: " + detallepedido.Cantidad + "\nPrecio Venta: " + detallepedido.Precioventa+"\n"+Environment.NewLine;

}

}

private void BotonAceptar\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class ModificarCadete : Form

{

public ModificarCadete(int id, string nombre, string apellido, int telefono, int licencia)

{

InitializeComponent();

labelid.Text = id.ToString();

textNombre.Text = nombre;

textApellido.Text = apellido;

texttelefono.Text = (telefono).ToString();

textlicencia.Text = (licencia).ToString();

}

private void ModificarCadete\_Load(object sender, EventArgs e)

{

}

private void botonCancelar\_Click(object sender, EventArgs e)

{

Close();

}

private void BotonAceptar\_Click(object sender, EventArgs e)

{

if (textNombre.Text == "" || textApellido.Text == "" || texttelefono.Text == "" || textlicencia.Text == "" )

{

MessageBox.Show("Complete todos los datos");

}

else

{

int id = int.Parse(labelid.Text);

string nombre = textNombre.Text;

string apellido = textApellido.Text;

int telefono = int.Parse(texttelefono.Text);

int licencia = int.Parse(textlicencia.Text);

CadeteBLL cadete = new CadeteBLL();

cadete.ModificarCadete(id, nombre, apellido, telefono, licencia);

MessageBox.Show("Cadete modificado. Actualize la tabla para visualizar los cambios");

Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class ModificarUsuario : Form

{

public ModificarUsuario(int pid, string pnombre, string papellido, int pCUIl )

{

InitializeComponent();

string nombre = pnombre;

string apellido = papellido;

int CUIL = pCUIl;

labelid.Text = pid.ToString();

textNombre.Text = nombre;

textApellido.Text = apellido;

textCUIL.Text = CUIL.ToString();

comboCategoria.DisplayMember = "descripcion";

}

private void ModificarUsuario\_Load(object sender, EventArgs e)

{

UsuarioCategoriaBLL categoria = new UsuarioCategoriaBLL();

foreach (UsuarioCategoriaBLL cate in categoria.CargarUsuarioCategoria())

{

comboCategoria.Items.Add(cate);

}

}

private void botonCancelar\_Click(object sender, EventArgs e)

{

Close();

}

private void BotonAceptar\_Click(object sender, EventArgs e)

{

if (textNombre.Text == "" || textApellido.Text == "" || textcontraseña.Text == "" || textcontraseña2.Text == "" || textCUIL.Text == "" || comboCategoria.SelectedItem == null)

{

MessageBox.Show("Complete todos los datos");

}

else if (textcontraseña.Text != textcontraseña2.Text)

{

MessageBox.Show("Las contraseñas no coinciden");

}

else

{

int id = int.Parse(labelid.Text);

string nombre = textNombre.Text;

string apellido = textApellido.Text;

int cuil = int.Parse(textCUIL.Text);

int contraseña = int.Parse(textcontraseña.Text);

int idcategoria = ((UsuarioCategoriaBLL)comboCategoria.SelectedItem).Idcategoria;

UsuarioBLL usuario = new UsuarioBLL();

usuario.ModificarUsuario(id, nombre, apellido, cuil, contraseña, idcategoria);

MessageBox.Show("Usuario modificado. Actualize la tabla para visualizar los cambios");

Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Printing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionCrearPedidos : Form

{

private int idusuario;

private List<DetallePedidoBLL> listadetallespedidos;

private int contadorticket = 0;

public SubSeccionCrearPedidos(int pidusuario)

{

InitializeComponent();

this.idusuario = pidusuario;

}

public int Idusuario { get => idusuario; set => idusuario = value; }

public List<DetallePedidoBLL> Listadetallespedidos { get => listadetallespedidos; set => listadetallespedidos = value; }

public int Contadorticket { get => contadorticket; set => contadorticket = value; }

private void SubSeccionPedidos\_Load(object sender, EventArgs e)

{

this.FormBorderStyle = FormBorderStyle.None;

this.BackColor = Color.FromArgb(20, 34, 56);

this.panel1.BackColor = Color.FromArgb(48, 58, 105);

this.label1.ForeColor = Color.White;

this.label2.ForeColor = Color.White;

this.BotonConsultarPrecio.ForeColor = Color.White;

this.BotonConsultarPrecio.BackColor = Color.FromArgb(98, 104, 140);

this.BotonAgregarProducto.ForeColor = Color.White;

this.BotonAgregarProducto.BackColor = Color.FromArgb(98, 104, 140);

this.label3.ForeColor = Color.White;

this.label4.ForeColor = Color.White;

this.label5.ForeColor = Color.White;

this.label6.ForeColor = Color.White;

this.label7.ForeColor = Color.White;

this.BotonCrearPedido.ForeColor = Color.White;

this.BotonCrearPedido.BackColor = Color.FromArgb(98, 104, 140);

this.Total.ForeColor = Color.White;

this.BotonBorrarProducto.BackColor = Color.Red;

this.ListaDeProductos.DisplayMember = "descripcion";

this.comboBoxClientes.DisplayMember = "apellido";

this.comboBoxFormasDePago.DisplayMember = "descripcion";

this.ListaDeDetallesPedidos.DisplayMember = "informacion";

this.comboBoxCantidad.Text = "";

this.labelImporte.ForeColor = Color.White;

this.Total.ForeColor = Color.White;

for (int i = 1; i <= 10; i++)

{

this.comboBoxCantidad.Items.Add(i);

}

// Clientes

CapaDeNegocios.ClienteBLL cliente = new CapaDeNegocios.ClienteBLL();

List<ClienteBLL> clientesacargar = new List<ClienteBLL>();

clientesacargar = cliente.CargarClientes();

foreach (ClienteBLL lineacliente in clientesacargar)

{

this.comboBoxClientes.Items.Add(lineacliente);

}

// Formas de Pago

CapaDeNegocios.FormaDePagoBLL formadepago = new CapaDeNegocios.FormaDePagoBLL();

List<FormaDePagoBLL> formasdepagoaaagregar = new List<FormaDePagoBLL>();

formasdepagoaaagregar = formadepago.CargarFormasDePago();

foreach (FormaDePagoBLL lineaformadepago in formasdepagoaaagregar)

{

this.comboBoxFormasDePago.Items.Add(lineaformadepago);

}

}

private void BotonEntradas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEntradas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonConsultarPrecio\_Click(object sender, EventArgs e)

{

if (this.ListaDeProductos.SelectedItem != null)

{

ProductoBLL entrada = (ProductoBLL)this.ListaDeProductos.SelectedItem;

int precio = entrada.Precio;

MessageBox.Show("El precio es " + precio, "Ver Precio");

}

else

{

MessageBox.Show("No seleccionó ningun producto", "Ver Precio");

}

}

private void BotonOmelletes\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarOmelletes();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonTortillas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarTortillas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonMilanesas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarMilanesas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonSupremas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarSupremas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonMatambre\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarMatambre();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonGuarnicion\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarGuarnicion();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonPollos\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarPollos();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonParillas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarParillas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonPescados\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarPescados();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEnsaladas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEnsaladas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEnsaladasEsp\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEnsaladasEsp();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonTradicionales\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarTradicionales();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEspecialesI\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEspecialesI();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEspecialesII\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEspecialesII();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEspecialesEnFugazza\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarFugazzas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEspecialesEnFugazzeta\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarFugazzetas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonCalzones\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarCalzones();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonOtrasEspeciales\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarOtrasEspecialidades();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonEmpanadas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarEmpanadas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonSandwiches\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarSandwiches();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonPastas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarPastas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonBebidas\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarBebidas();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonPostres\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarPostres();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonMenus\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarMenus();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonPromos\_Click(object sender, EventArgs e)

{

this.ListaDeProductos.Items.Clear();

CapaDeNegocios.ProductoBLL producto = new ProductoBLL();

List<ProductoBLL> productosacargar = new List<ProductoBLL>();

productosacargar = producto.CargarPromos();

foreach (ProductoBLL lineaproducto in productosacargar)

{

this.ListaDeProductos.Items.Add(lineaproducto);

}

}

private void BotonAgregarProducto\_Click(object sender, EventArgs e)

{

if (this.comboBoxCantidad.SelectedItem != null && this.ListaDeProductos.SelectedItem != null)

{

int idproducto = ((ProductoBLL)this.ListaDeProductos.SelectedItem).Idproducto;

string descripcion = ((ProductoBLL)this.ListaDeProductos.SelectedItem).Descripcion;

int cantidad = int.Parse(this.comboBoxCantidad.Text);

int precioventa = ((ProductoBLL)this.ListaDeProductos.SelectedItem).Precio \* cantidad;

string informacion = descripcion + " " + "(" + cantidad + ")" + " " + precioventa;

DetallePedidoBLL detallepedido = new DetallePedidoBLL(idproducto, cantidad, precioventa, informacion);

this.ListaDeDetallesPedidos.Items.Add(detallepedido);

this.comboBoxCantidad.SelectedItem = this.comboBoxCantidad.Items[0];

this.ListaDeProductos.ClearSelected();

int acumuladorprecio = 0;

foreach (DetallePedidoBLL detallepedidolocal in this.ListaDeDetallesPedidos.Items)

{

int preciodetallepedido = detallepedidolocal.Precioventa;

acumuladorprecio += preciodetallepedido;

}

this.Total.Text = acumuladorprecio.ToString();

this.comboBoxCantidad.SelectedItem = null;

this.comboBoxCantidad.Text = "";

}

else

{

if (this.comboBoxCantidad.SelectedItem == null)

{

MessageBox.Show("No se ha especificado la cantidad del producto", "Agregar Producto");

}

else

{

MessageBox.Show("No se ha seleccionado un producto para agregar al pedido", "Agregar Producto");

}

}

}

private void BotonBorrarProducto\_Click(object sender, EventArgs e)

{

int acumuladorprecio = 0;

if (this.ListaDeDetallesPedidos.SelectedItem != null && this.ListaDeDetallesPedidos.Items.Count > 0)

{

this.ListaDeDetallesPedidos.Items.Remove(this.ListaDeDetallesPedidos.SelectedItem);

foreach (DetallePedidoBLL detallepedidolocal in this.ListaDeDetallesPedidos.Items)

{

int preciodetallepedido = detallepedidolocal.Precioventa;

acumuladorprecio += preciodetallepedido;

}

this.Total.Text = acumuladorprecio.ToString();

}

else

{

MessageBox.Show("No ha seleccionado un detalle pedido o bien no hay items en la lista", "Error");

}

}

private void BotonCrearPedido\_Click(object sender, EventArgs e)

{

if (this.ListaDeDetallesPedidos.Items.Count > 0)

{

if (this.comboBoxClientes.SelectedItem == null)

{

MessageBox.Show("No se especificó el cliente del pedido", "Error");

}

else if (this.comboBoxFormasDePago.SelectedItem == null)

{

MessageBox.Show("No se especificó la forma de pago del pedido", "Error");

}

else

{

// Pedido

CapaDeNegocios.PedidoBLL pedido = new CapaDeNegocios.PedidoBLL();

pedido.Idcliente = ((ClienteBLL)this.comboBoxClientes.SelectedItem).Idcliente;

pedido.Idformadepago = ((FormaDePagoBLL)this.comboBoxFormasDePago.SelectedItem).Idformadepago;

pedido.Idusuario = this.idusuario;

pedido.Observaciones = this.Observaciones.Text;

pedido.Idcadete = 1;

pedido.Idestadocategoria = 1;

pedido.Total = int.Parse(this.Total.Text);

int valorcrearpedido = pedido.CrearPedido();

// Detalle Pedido

this.listadetallespedidos = new List<DetallePedidoBLL>();

foreach (DetallePedidoBLL detallepedidolocal in this.ListaDeDetallesPedidos.Items)

{

CapaDeNegocios.DetallePedidoBLL detallepedido = new CapaDeNegocios.DetallePedidoBLL();

detallepedido.Idproducto = detallepedidolocal.Idproducto;

detallepedido.Idpedido = int.Parse(detallepedido.getReferenciaIdPedido().Rows[0][0].ToString());

detallepedido.Cantidad = detallepedidolocal.Cantidad;

detallepedido.Precioventa = detallepedidolocal.Precioventa;

int valordetallepedido = detallepedido.CrearDetallePedido();

}

this.comboBoxCantidad.SelectedItem = null;

this.comboBoxCantidad.Text = "";

this.ListaDeProductos.SelectedItem = null;

this.ListaDeProductos.Items.Clear();

this.comboBoxClientes.SelectedItem = null;

this.comboBoxClientes.Text = "";

this.comboBoxClientes.Text = "";

this.Observaciones.Text = "";

this.Total.Text = "0";

foreach (DetallePedidoBLL detallepedido in this.ListaDeDetallesPedidos.Items)

{

this.listadetallespedidos.Add(detallepedido);

}

MessageBox.Show("Se ha creado el pedido", "Crear Pedido");

this.ConfigurarComanda();

this.ConfigurarTicketOriginal();

this.ConfigurarTicketDuplicado();

this.ListaDeDetallesPedidos.Items.Clear();

this.comboBoxFormasDePago.SelectedItem = null;

this.Total.Text = "0";

}

}

else

{

MessageBox.Show("No se agregaron productos para crear el pedido", "Error");

}

}

private void ConfigurarComanda()

{

Comanda = new System.Drawing.Printing.PrintDocument();

PrinterSettings printersettings = new PrinterSettings();

Comanda.PrinterSettings = printersettings;

Comanda.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm", 100, 100);

Comanda.PrintPage += ImprimirComanda;

Comanda.Print();

}

public void ConfigurarTicketOriginal ()

{

TicketOriginal = new System.Drawing.Printing.PrintDocument();

PrinterSettings printersettings = new PrinterSettings();

TicketOriginal.PrinterSettings = printersettings;

TicketOriginal.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm", 100, 100);

TicketOriginal.PrintPage += ImprimirTicketOriginal;

TicketOriginal.Print();

}

public void ConfigurarTicketDuplicado ()

{

TicketDuplicado = new System.Drawing.Printing.PrintDocument();

PrinterSettings printersettings = new PrinterSettings();

TicketDuplicado.PrinterSettings = printersettings;

TicketDuplicado.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm", 100, 100);

TicketDuplicado.PrintPage += ImprimirTicketDuplicado;

TicketDuplicado.Print();

}

private void ImprimirComanda(object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

DetallePedidoBLL objetodetallepedido = new DetallePedidoBLL();

// Comanda

e.Graphics.DrawString("Comandas - Guilerova Solutions", new Font("Bell MT", 16, FontStyle.Bold),

Brushes.CadetBlue, new PointF(0, 0));

e.Graphics.DrawString("Che Miranda", new Font("Arial", 30, FontStyle.Regular),

Brushes.Black, new PointF(0, 50));

e.Graphics.DrawString("Comanda N° " + int.Parse(objetodetallepedido.getReferenciaIdPedido().Rows[0][0].ToString()), new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 120));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 150));

e.Graphics.DrawString("Fecha: " + DateTime.Now.Date.ToString(), new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 180));

e.Graphics.DrawString("Hora: " + DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString() + ":" + DateTime.Now.Second.ToString(), new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 220));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 250));

e.Graphics.DrawString("Detalle Pedido", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 280));

int y = 320;

foreach (DetallePedidoBLL detallepedido in this.listadetallespedidos)

{

e.Graphics.DrawString(detallepedido.Informacion,

new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, y));

y = y + 30;

}

}

private void ImprimirTicketOriginal(object sender, PrintPageEventArgs e)

{

// Ticket Original

e.Graphics.DrawString("Comandas - Guilerova Solutions", new Font("Bell MT", 16, FontStyle.Bold),

Brushes.CadetBlue, new PointF(0, 0));

e.Graphics.DrawString("Che Miranda", new Font("Arial", 30, FontStyle.Regular),

Brushes.Black, new PointF(0, 50));

e.Graphics.DrawString("Dirección: Av. Avellaneda 4199 - Teléfono: 011-4671-8718", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 120));

e.Graphics.DrawString("Ticket N° "+this.contadorticket+" CUIT: XX-XXXXXXXX-X Original", new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 150));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 190));

e.Graphics.DrawString("Fecha: " + DateTime.Now.Date.ToString(),new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 220));

e.Graphics.DrawString("Hora: " + DateTime.Now.Hour.ToString()+":"+ DateTime.Now.Minute.ToString() + ":"+DateTime.Now.Second.ToString(), new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 260));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 300));

e.Graphics.DrawString("Pedido", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 330));

e.Graphics.DrawString("Num. Pedido Detalle Pedido Cantidad Importe", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 370));

int y = 410;

foreach (DetallePedidoBLL detallepedido in this.listadetallespedidos)

{

e.Graphics.DrawString(detallepedido.Idpedido +" "+detallepedido.Informacion, new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, y));

y = y + 30;

}

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, y));

e.Graphics.DrawString("MED. PAGO: " + this.comboBoxFormasDePago.Text + " IMPORTE TOTAL: " + this.Total, new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, y + 30));

e.Graphics.DrawString("Gracias por su pedido", new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, y + 30));

}

private void ImprimirTicketDuplicado(object sender, PrintPageEventArgs e)

{

// Ticket Duplicado

e.Graphics.DrawString("Comandas - Guilerova Solutions", new Font("Bell MT", 16, FontStyle.Bold),

Brushes.CadetBlue, new PointF(0, 0));

e.Graphics.DrawString("Che Miranda", new Font("Arial", 30, FontStyle.Regular),

Brushes.Black, new PointF(0, 50));

e.Graphics.DrawString("Dirección: Av. Avellaneda 4199 - Teléfono: 011-4671-8718", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 120));

e.Graphics.DrawString("Ticket N° " + this.contadorticket + " CUIT: XX-XXXXXXXX-X Duplicado", new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 150));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 190));

e.Graphics.DrawString("Fecha: " + DateTime.Now.Date.ToString(), new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 220));

e.Graphics.DrawString("Hora: " + DateTime.Now.Hour.ToString() + ":" + DateTime.Now.Minute.ToString() + ":" + DateTime.Now.Second.ToString(), new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, 260));

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 300));

e.Graphics.DrawString("Pedido", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 330));

e.Graphics.DrawString("Num. Pedido Detalle Pedido Cantidad Importe", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 370));

int y = 410;

foreach (DetallePedidoBLL detallepedido in this.listadetallespedidos)

{

e.Graphics.DrawString(detallepedido.Idpedido + " " + detallepedido.Informacion, new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, 370));

y = y + 30;

}

e.Graphics.DrawString("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", new Font("Arial", 18, FontStyle.Regular),

Brushes.Black, new PointF(0, y));

e.Graphics.DrawString("MED. PAGO: " + this.comboBoxFormasDePago.Text + " IMPORTE TOTAL: " + this.Total, new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, y + 30));

e.Graphics.DrawString("Gracias por su pedido", new Font("Arial", 16, FontStyle.Regular),

Brushes.Black, new PointF(0, y + 30));

this.contadorticket = this.contadorticket + 1;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionGestionDePedidos : Form

{

public SubSeccionGestionDePedidos(int pidusuario)

{

InitializeComponent();

}

private void SubSeccionGestionDePedidos\_Load(object sender, EventArgs e)

{

this.BackColor = Color.FromArgb(20, 34, 56);

this.CentrarElemento(this, this.panel1);

this.comboBoxEstadoPedido.DisplayMember = "descripcion";

this.comboBoxCadete.DisplayMember = "nombre";

this.label1.ForeColor = Color.White;

this.label2.ForeColor = Color.White;

this.BotonModificarEstado.ForeColor = Color.White;

this.BotonModificarEstado.BackColor = Color.FromArgb(98, 104, 140);

this.label3.ForeColor = Color.White;

this.BotonAsignarCadete.ForeColor = Color.White;

this.BotonAsignarCadete.BackColor = Color.FromArgb(98, 104, 140);

this.BotonVerDetallePedido.ForeColor = Color.White;

this.BotonVerDetallePedido.BackColor = Color.FromArgb(98, 104, 140);

// Pedidos

CapaDeNegocios.PedidoBLL pedido = new CapaDeNegocios.PedidoBLL();

this.dataGridView1.DataSource = pedido.CargarPedidosActivos();

// Estados de Categorias

CapaDeNegocios.EstadoCategoriaBLL estadocategoria = new CapaDeNegocios.EstadoCategoriaBLL();

List<EstadoCategoriaBLL> estadoscategoriasaagregar = new List<EstadoCategoriaBLL>();

estadoscategoriasaagregar = estadocategoria.CargarEstadosCategoria();

foreach (EstadoCategoriaBLL lineaestadocategoria in estadoscategoriasaagregar)

{

this.comboBoxEstadoPedido.Items.Add(lineaestadocategoria);

}

// Cadetes

CadeteBLL cadete = new CadeteBLL();

List<CadeteBLL> cadetesaagregar = new List<CadeteBLL>();

cadetesaagregar = cadete.CargarCadetes();

foreach (CadeteBLL lineacadete in cadetesaagregar)

{

this.comboBoxCadete.Items.Add(lineacadete);

}

this.dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView1.AllowUserToAddRows = false;

this.dataGridView1.AllowUserToDeleteRows = false;

this.dataGridView1.AllowUserToResizeRows = false;

this.dataGridView1.AllowUserToOrderColumns = false;

this.dataGridView1.AllowUserToResizeColumns = false;

this.dataGridView1.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView1.RowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView1.ReadOnly = true;

this.dataGridView1.BackgroundColor = Color.Lavender;

}

public void CentrarElemento(Control ppadre, Control phijo)

{

int posx;

posx = (ppadre.Width / 2) - (phijo.Width / 2);

phijo.Location = new System.Drawing.Point(posx, phijo.Location.Y);

int posy;

posy = (ppadre.Height / 2) - (phijo.Height / 2);

phijo.Location = new System.Drawing.Point(posx, posy);

}

private void BotonModificarEstado\_Click(object sender, EventArgs e)

{

int idpedido = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

int idestado = ((EstadoCategoriaBLL)comboBoxEstadoPedido.SelectedItem).Idestadocategoria;

CapaDeNegocios.PedidoBLL pedido = new CapaDeNegocios.PedidoBLL();

pedido.ModificarEstadoPedido(idpedido, idestado);

this.dataGridView1.DataSource = pedido.CargarPedidosActivos();

}

private void BotonAsignarCadete\_Click(object sender, EventArgs e)

{

CadeteBLL cadete = new CadeteBLL();

int idpedido = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

int idcadete = ((CadeteBLL)comboBoxCadete.SelectedItem).Idcadete;

cadete.AsignarCadete(idpedido, idcadete);

CapaDeNegocios.PedidoBLL pedido = new CapaDeNegocios.PedidoBLL();

this.dataGridView1.DataSource = pedido.CargarPedidosActivos();

}

private void BotonVerDetallePedido\_Click(object sender, EventArgs e)

{

if (this.dataGridView1.SelectedRows.Count > 0)

{

int idpedido = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

DetallePedidoBLL detallepedido = new DetallePedidoBLL();

List<DetallePedidoBLL> listadetallespedidos = detallepedido.CargarDetallePedidoSegunPedido(idpedido);

FormularioVerDetallePedido formularioverdetallepedido = new FormularioVerDetallePedido(listadetallespedidos);

formularioverdetallepedido.ShowDialog();

}

else

{

MessageBox.Show("Debe seleccionar un pedido", "Ver Detalle Pedido");

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionInformes : Form

{

public SubSeccionInformes()

{

InitializeComponent();

}

private void SubSeccionInformes\_Load(object sender, EventArgs e)

{

this.gridPedidos.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.gridPedidos.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.gridProductos.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.gridProductos.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

gridProductos.AllowUserToAddRows = false;

gridPedidos.AllowUserToAddRows = false;

List<string> mes = new List<string>();

mes.Add("Enero");

mes.Add("Febrero");

mes.Add ("Marzo");

mes.Add ("Abril");

mes.Add ("Mayo");

mes.Add ("Junio");

mes.Add ("Julio");

mes.Add ("Agosto");

mes.Add ("Septiembre");

mes.Add ("Octubre");

mes.Add ("Noviembre");

mes.Add ("Diciembre");

comboBoxAño.Items.Add(2020);

comboBoxAño.Items.Add(2021);

comboBoxAño2.Items.Add(2020);

comboBoxAño2.Items.Add(2021);

foreach (string m in mes)

{

comboBoxMes.Items.Add(m);

comboBoxMes2.Items.Add(m);

}

this.CentrarElemento(this, this.panel2);

}

public void CentrarElemento(Control ppadre, Control phijo)

{

int posx;

posx = (ppadre.Width / 2) - (phijo.Width / 2);

phijo.Location = new System.Drawing.Point(posx, phijo.Location.Y);

int posy;

posy = (ppadre.Height / 2) - (phijo.Height / 2);

phijo.Location = new System.Drawing.Point(posx, posy);

}

private void radioButtonDiario\_CheckedChanged(object sender, EventArgs e)

{

comboBoxMes.Enabled = false;

comboBoxAño.Enabled = false;

}

private void radioButtonMensual\_CheckedChanged(object sender, EventArgs e)

{

comboBoxMes.Enabled = true;

comboBoxAño.Enabled = true;

}

private void radioDiaro2\_CheckedChanged(object sender, EventArgs e)

{

comboBoxMes2.Enabled = false;

comboBoxAño2.Enabled = false;

}

private void BotonVerPedidos\_Click(object sender, EventArgs e)

{

//Diario

if (radioButtonDiario.Checked)

{

PedidoBLL pedido = new PedidoBLL();

gridPedidos.DataSource = pedido.CargarPedidosdeldia();

int total = 0;

foreach (DataGridViewRow fila in gridPedidos.Rows)

{

total = total + int.Parse(fila.Cells[8].Value.ToString());

}

labeltotal.Text = total.ToString();

}

//Mensual

if (radioButtonMensual.Checked )

{

if (comboBoxMes.SelectedItem==null || comboBoxAño.SelectedItem==null)

{

MessageBox.Show("Seleccione una fecha correcta");

}

else

{

int mes = comboBoxMes.SelectedIndex + 1;

int año = int.Parse(comboBoxAño.SelectedItem.ToString());

PedidoBLL pedido = new PedidoBLL();

gridPedidos.DataSource = pedido.CargarPedidosporMesAño(mes, año);

int total = 0;

foreach (DataGridViewRow fila in gridPedidos.Rows)

{

total = total + int.Parse(fila.Cells[8].Value.ToString());

}

labeltotal.Text = total.ToString();

}

}

}

private void BotonVerProductos\_Click(object sender, EventArgs e)

{

//Diario

if (radioDiaro2.Checked)

{

ProductoBLL producto = new ProductoBLL();

gridProductos.DataSource = producto.CargarProductoMasVendidoDia();

}

//Mensual

if (radioMensual2.Checked)

{

if (comboBoxMes2.SelectedItem == null || comboBoxAño2.SelectedItem == null)

{

MessageBox.Show("Seleccione una fecha correcta");

}

else

{

int mes = comboBoxMes2.SelectedIndex + 1;

int año = int.Parse(comboBoxAño2.SelectedItem.ToString());

ProductoBLL producto = new ProductoBLL();

gridProductos.DataSource = producto.CargarProductoMasVendidoMesAño(mes, año);

int total = 0;

foreach (DataGridViewRow fila in gridPedidos.Rows)

{

total = total + int.Parse(fila.Cells[8].Value.ToString());

}

labeltotal.Text = total.ToString();

}

}

}

private void radioMensual2\_CheckedChanged(object sender, EventArgs e)

{

comboBoxMes2.Enabled = true;

comboBoxAño2.Enabled = true;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionModificarProducto : Form {

public SubSeccionModificarProducto(int pidproducto,string pNombre, int pprecio)

{

InitializeComponent();

labelid.Text = pidproducto.ToString();

textnombre.Text = pNombre;

textprecio.Text = pprecio.ToString();

}

private void SubSeccionModificarProducto\_Load(object sender, EventArgs e)

{

combocategoria.DisplayMember = "Nombre";

CategoriaProductoBLL categoria = new CategoriaProductoBLL();

foreach (CategoriaProductoBLL cate in categoria.CargarProductoCategoria())

{

combocategoria.Items.Add(cate);

}

}

private void botonCancelar\_Click(object sender, EventArgs e)

{

Close();

}

private void BotonAceptar\_Click(object sender, EventArgs e)

{

if (textnombre.Text == "")

{

MessageBox.Show("Falta completar el nombre");

}

else if (textprecio.Text == "")

{

MessageBox.Show("Falta completar el precio");

}

else if (combocategoria.SelectedItem == null)

{

MessageBox.Show("Falta completar la categoria");

}

else{

int idcategoria = ((CategoriaProductoBLL)combocategoria.SelectedItem).Idcategoriaproducto;

int idproducto = int.Parse(labelid.Text);

string nombre = textnombre.Text;

int precio = int.Parse(textprecio.Text);

ProductoBLL producto = new ProductoBLL();

producto.ModificarProducto(idproducto, textnombre.Text, idcategoria, precio);

MessageBox.Show("Producto modificado correctamente, actualice la lista para visualizar los cambios");

Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionPersonal : Form

{

public SubSeccionPersonal()

{

InitializeComponent();

}

private void SubSeccionPersonal\_Load(object sender, EventArgs e)

{

this.GridPersonal.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.GridPersonal.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

comboCategoria.DisplayMember = "descripcion";

UsuarioBLL usuario = new UsuarioBLL();

GridPersonal.DataSource = usuario.CargarUsuariossincontraseña();

UsuarioCategoriaBLL categoria = new UsuarioCategoriaBLL();

foreach (UsuarioCategoriaBLL cate in categoria.CargarUsuarioCategoria())

{

comboCategoria.Items.Add(cate);

}

////Cadete

this.gridcadete.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.gridcadete.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

CadeteBLL cadete = new CadeteBLL();

gridcadete.DataSource = cadete.CargartablaCadetes ();

}

private void botonactulizar\_Click(object sender, EventArgs e)

{

UsuarioBLL usuario = new UsuarioBLL();

GridPersonal.DataSource = usuario.CargarUsuariossincontraseña();

}

private void BotonCrearUsuario\_Click(object sender, EventArgs e)

{

if (textNombre.Text == "" || textApellido.Text=="" || textcontraseña.Text=="" || textcontraseña2.Text=="" || textCUIL.Text =="" || comboCategoria.SelectedItem== null )

{

MessageBox.Show("Falta completar datos");

}

else if (textcontraseña.Text != textcontraseña2.Text)

{

MessageBox.Show("Las contraseñas no coinciden");

}

else

{

string nombre = textNombre.Text;

string apellido = textApellido.Text;

int cuil = int.Parse(textCUIL.Text);

int contraseña = int.Parse(textcontraseña.Text);

int idcategoria = int.Parse(((UsuarioCategoriaBLL)comboCategoria.SelectedItem).Idcategoria.ToString());

UsuarioBLL usuario = new UsuarioBLL();

usuario.CrearUsuario(nombre, apellido, cuil, contraseña, idcategoria);

GridPersonal.DataSource = usuario.CargarUsuariossincontraseña();

MessageBox.Show("El usuario se creo de forma correcta");

}

}

private void botonModificarUsuario\_Click(object sender, EventArgs e)

{

int idusuario = int.Parse(GridPersonal.SelectedRows[0].Cells[0].Value.ToString());

string nombre = (GridPersonal.SelectedRows[0].Cells[1].Value.ToString());

string apellido = (GridPersonal.SelectedRows[0].Cells[2].Value.ToString());

int CUIL = int.Parse(GridPersonal.SelectedRows[0].Cells[3].Value.ToString());

ModificarUsuario modi = new ModificarUsuario(idusuario, nombre, apellido, CUIL);

modi.ShowDialog();

}

private void botonEliminarUsuario\_Click(object sender, EventArgs e)

{

int id;

id = int.Parse(GridPersonal.SelectedRows[0].Cells[0].Value.ToString());

string nombre = (GridPersonal.SelectedRows[0].Cells[1].Value.ToString());

UsuarioBLL usuario = new UsuarioBLL();

if (GridPersonal.SelectedRows.Count == 1)

{

DialogResult dialogResult = MessageBox.Show("¿Realmente quiere eliminar el usuario " + nombre + "?", "Eliminar usuario", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

usuario.EliminarUsuario(id);

MessageBox.Show("El usuario ha sido eliminado");

GridPersonal.DataSource = usuario.CargarUsuariossincontraseña();

}

}

else

{

MessageBox.Show("No se selecciono ningun usuario");

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (textnom.Text == "" || textape.Text == "" || texttel.Text == "" || textlicencia.Text == "" )

{

MessageBox.Show("Falta completar datos");

}

else

{

string nombre = textnom.Text;

string apellido = textape.Text;

int tel = int.Parse(texttel.Text);

int licencia = int.Parse(textlicencia.Text);

CadeteBLL cadete = new CadeteBLL();

cadete.CrearCadete(nombre, apellido, tel, licencia);

gridcadete.DataSource = cadete.CargarCadetes();

MessageBox.Show("El Cadete se creo de forma correcta");

}

}

private void actulizarcadete\_Click(object sender, EventArgs e)

{

CadeteBLL cadete = new CadeteBLL();

gridcadete.DataSource = cadete.CargartablaCadetes();

}

private void motonmodicadete\_Click(object sender, EventArgs e)

{

int id = int.Parse(gridcadete.SelectedRows[0].Cells[0].Value.ToString());

string nombre = (gridcadete.SelectedRows[0].Cells[1].Value.ToString());

string apellido = (gridcadete.SelectedRows[0].Cells[2].Value.ToString());

int telefono = int.Parse(gridcadete.SelectedRows[0].Cells[3].Value.ToString());

int licencia = int.Parse(gridcadete.SelectedRows[0].Cells[4].Value.ToString());

ModificarCadete cadete = new ModificarCadete(id, nombre, apellido, telefono, licencia);

cadete.ShowDialog();

}

private void botoneliminarcadete\_Click(object sender, EventArgs e)

{

int id;

id = int.Parse(gridcadete.SelectedRows[0].Cells[0].Value.ToString());

string nombre = (gridcadete.SelectedRows[0].Cells[1].Value.ToString());

CadeteBLL cadete = new CadeteBLL();

if (gridcadete.SelectedRows.Count == 1)

{

DialogResult dialogResult = MessageBox.Show("¿Realmente quiere eliminar el cadete " + nombre + "?", "Eliminar cadete", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

cadete.EliminarCadete(id);

MessageBox.Show("El cadete ha sido eliminado");

gridcadete.DataSource = cadete.CargartablaCadetes();

}

}

else

{

MessageBox.Show("No se selecciono ningun cadete");

}

}

private void groupBox2\_Enter(object sender, EventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace SeccionesFormularios

{

public partial class SubSeccionProductos : Form

{

public SubSeccionProductos()

{

InitializeComponent();

}

private void SubSeccionProductos\_Load(object sender, EventArgs e)

{

comboCategoriaP.DisplayMember = "Nombre";

CategoriaProductoBLL categoria = new CategoriaProductoBLL();

foreach (CategoriaProductoBLL cate in categoria.CargarProductoCategoria())

{

comboCategoriaP.Items.Add(cate);

}

ProductoBLL productobll = new ProductoBLL();

GridProductos.DataSource = productobll.CargarTodolosproductos();

this.GridProductos.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.GridProductos.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

}

private void textfiltroproducto\_TextChanged(object sender, EventArgs e)

{

ProductoBLL productobll = new ProductoBLL();

string filtro = "%"+textfiltroproducto.Text+"%";

GridProductos.DataSource= productobll.FIltroProductos(filtro);

}

private void BotonCrearProducto\_Click(object sender, EventArgs e)

{

int precio = 0;

int idcate = 0;

ProductoBLL producto = new ProductoBLL();

string descri = textNombreP.Text;

if (descri == "" )

{

MessageBox.Show("No se coloco nombre al producto");

}

else if(comboCategoriaP.SelectedItem == null)

{

MessageBox.Show("Categoria no seleccionada");

}

else if (textPrecioP.Text=="")

{

MessageBox.Show("Precio no definido");

}

else

{

idcate = ((CategoriaProductoBLL)comboCategoriaP.SelectedItem).Idcategoriaproducto;

precio = int.Parse(textPrecioP.Text);

producto.CrearProducto(descri, idcate, precio);

MessageBox.Show("Producto creado con exito");

}

}

private void botonModificarProducto\_Click(object sender, EventArgs e)

{

int idproducto;

string nombre;

CategoriaProductoBLL categoria;

int precio;

if (GridProductos.SelectedRows.Count == 1) {

idproducto = int.Parse(GridProductos.SelectedRows[0].Cells[0].Value.ToString());

nombre = GridProductos.SelectedRows[0].Cells[1].Value.ToString();

precio = int.Parse(GridProductos.SelectedRows[0].Cells[3].Value.ToString());

SubSeccionModificarProducto modificar = new SubSeccionModificarProducto(idproducto, nombre, precio);

modificar.ShowDialog();

}

else

{

MessageBox.Show("No se selecciono nigun producto");

}

}

private void botonactulizar\_Click(object sender, EventArgs e)

{

ProductoBLL productobll = new ProductoBLL();

GridProductos.DataSource = productobll.CargarTodolosproductos();

}

private void botonEliminarProducto\_Click(object sender, EventArgs e)

{

int idproducto;

idproducto = int.Parse(GridProductos.SelectedRows[0].Cells[0].Value.ToString());

string nombre = (GridProductos.SelectedRows[0].Cells[1].Value.ToString());

ProductoBLL producto = new ProductoBLL();

if (GridProductos.SelectedRows.Count == 1)

{

DialogResult dialogResult = MessageBox.Show("¿Realmente quiere eliminar el producto "+nombre+"?", "Eliminar Producto", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

producto.EliminarProducto(idproducto);

MessageBox.Show("El producto ha sido eliminado");

GridProductos.DataSource = producto.CargarTodolosproductos();

}

}

else

{

MessageBox.Show("No se selecciono nigun producto");

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace WindowsFormsApp1.SeccionClientes

{

public partial class FormularioClientes : Form

{

public FormularioClientes()

{

InitializeComponent();

}

private void FormularioClientes\_Load(object sender, EventArgs e)

{

this.BackColor = Color.FromArgb(20, 34, 56);

this.CentrarElemento(this, this.panel1);

this.labelBúsquedaClientes.ForeColor = Color.White;

this.textBox1.BackColor = Color.FromArgb(20, 34, 56);

this.BotonNuevoCliente.ForeColor = Color.White;

this.BotonNuevoCliente.BackColor = Color.FromArgb(98, 104, 140);

this.BotonModificarCliente.ForeColor = Color.White;

this.BotonModificarCliente.BackColor = Color.FromArgb(98, 104, 140);

this.labelPedido.ForeColor = Color.White;

this.labelDetallePedido.ForeColor = Color.White;

// Clientes

this.dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView1.AllowUserToAddRows = false;

this.dataGridView1.AllowUserToDeleteRows = false;

this.dataGridView1.AllowUserToResizeRows = false;

this.dataGridView1.AllowUserToOrderColumns = false;

this.dataGridView1.AllowUserToResizeColumns = false;

this.dataGridView1.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView1.RowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView1.ReadOnly = true;

this.dataGridView1.BackgroundColor = Color.Lavender;

ClienteBLL cliente = new ClienteBLL();

this.dataGridView1.DataSource = cliente.CargarClientesDataTable();

this.dataGridView1.MultiSelect = false;

// Pedidos

this.dataGridView2.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView2.AllowUserToAddRows = false;

this.dataGridView2.AllowUserToDeleteRows = false;

this.dataGridView2.AllowUserToResizeRows = false;

this.dataGridView2.AllowUserToOrderColumns = false;

this.dataGridView2.AllowUserToResizeColumns = false;

this.dataGridView2.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView2.RowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView2.ReadOnly = true;

this.dataGridView2.BackgroundColor = Color.Lavender;

// Productos

this.dataGridView3.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

this.dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView3.AllowUserToAddRows = false;

this.dataGridView3.AllowUserToDeleteRows = false;

this.dataGridView3.AllowUserToResizeRows = false;

this.dataGridView3.AllowUserToOrderColumns = false;

this.dataGridView3.AllowUserToResizeColumns = false;

this.dataGridView3.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView3.RowsDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

this.dataGridView3.ReadOnly = true;

this.dataGridView3.BackgroundColor = Color.Lavender;

}

public void CentrarElemento(Control ppadre, Control phijo)

{

int posx;

posx = (ppadre.Width / 2) - (phijo.Width / 2);

phijo.Location = new System.Drawing.Point(posx, phijo.Location.Y);

int posy;

posy = (ppadre.Height / 2) - (phijo.Height / 2);

phijo.Location = new System.Drawing.Point(posx, posy);

}

private void BotonNuevoCliente\_Click(object sender, EventArgs e)

{

FormularioCrearCliente formulariocrearcliente = new FormularioCrearCliente();

formulariocrearcliente.ShowDialog();

ClienteBLL cliente = new ClienteBLL();

this.dataGridView1.DataSource = cliente.CargarClientesDataTable();

}

private void BotonEliminarCliente\_Click(object sender, EventArgs e)

{

ClienteBLL cliente = new ClienteBLL();

if (this.dataGridView1.SelectedRows.Count > 0)

{

int idcliente = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

cliente.EliminarCliente(idcliente);

this.dataGridView1.DataSource = cliente.CargarClientesDataTable();

}

else

{

MessageBox.Show("Debe seleccionar un cliente para eliminarlo", "Eliminar Cliente");

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

ClienteBLL cliente = new ClienteBLL();

if (this.textBox1.Text.Length != 0 && !(this.textBox1.Text.Contains(" ")))

{

this.dataGridView1.DataSource = cliente.BuscarClientePorNombre("%"+this.textBox1.Text+"%");

// Pedidos

int idcliente;

if (dataGridView1.Rows.Count==0)

{

}

else

{

idcliente = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

PedidoBLL pedido = new PedidoBLL();

this.dataGridView2.DataSource = pedido.CargarPedidosSegunCliente(idcliente);

// Detalle Pedido

if (this.dataGridView2.Rows.Count > 0)

{

int idpedido = int.Parse(this.dataGridView2.SelectedRows[0].Cells[0].Value.ToString());

DetallePedidoBLL detallepedido = new DetallePedidoBLL();

this.dataGridView3.DataSource = detallepedido.CargarTablaDetallePedidoSegunPedido(idpedido);

}

else if(this.dataGridView2.Rows.Count == 0)

{

this.dataGridView2.DataSource = null;

this.dataGridView3.DataSource = null;

}

else

{

this.dataGridView1.DataSource = cliente.CargarClientesDataTable();

this.dataGridView2.DataSource = null;

this.dataGridView3.DataSource = null;

}

}

}

if (textBox1.Text == "")

{

dataGridView1.DataSource = cliente.CargarClientes();

}

}

private void BotonModificarCliente\_Click(object sender, EventArgs e)

{

int idcliente, telefono;

string nombre, apellido, direccion;

idcliente = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

telefono = int.Parse(this.dataGridView1.SelectedRows[0].Cells[3].Value.ToString());

nombre = this.dataGridView1.SelectedRows[0].Cells[1].Value.ToString();

apellido = this.dataGridView1.SelectedRows[0].Cells[2].Value.ToString();

direccion = this.dataGridView1.SelectedRows[0].Cells[4].Value.ToString();

ClienteBLL cliente = new ClienteBLL(idcliente, nombre, apellido, telefono, direccion);

FormularioModificarCliente formulariomodificarcliente = new FormularioModificarCliente(cliente);

formulariomodificarcliente.ShowDialog();

this.dataGridView1.DataSource = cliente.CargarClientesDataTable();

int idclientedatagridview;

idclientedatagridview = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

PedidoBLL pedido = new PedidoBLL();

this.dataGridView2.DataSource = pedido.CargarPedidosSegunCliente(idclientedatagridview);

this.dataGridView3.DataSource = null;

}

private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

// Pedidos

int idcliente;

idcliente = int.Parse(this.dataGridView1.SelectedRows[0].Cells[0].Value.ToString());

PedidoBLL pedido = new PedidoBLL();

DataTable tablapedidosseguncliente = pedido.CargarPedidosSegunCliente(idcliente);

if (tablapedidosseguncliente.Rows.Count > 0)

{

this.dataGridView2.DataSource = tablapedidosseguncliente;

}

else

{

this.dataGridView2.DataSource = null;

this.dataGridView3.DataSource = null;

}

// Detalle Pedido

if (this.dataGridView2.Rows.Count > 0)

{

int idpedido = int.Parse(this.dataGridView2.SelectedRows[0].Cells[0].Value.ToString());

DetallePedidoBLL detallepedido = new DetallePedidoBLL();

this.dataGridView3.DataSource = detallepedido.CargarTablaDetallePedidoSegunPedido(idpedido);

}

else

{

this.dataGridView3.DataSource = null;

}

}

private void dataGridView2\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (this.dataGridView2.Rows.Count > 0)

{

int idpedido = int.Parse(this.dataGridView2.SelectedRows[0].Cells[0].Value.ToString());

DetallePedidoBLL detallepedido = new DetallePedidoBLL();

this.dataGridView3.DataSource = detallepedido.CargarTablaDetallePedidoSegunPedido(idpedido);

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace WindowsFormsApp1.SeccionClientes

{

public partial class FormularioCrearCliente : Form

{

public FormularioCrearCliente()

{

InitializeComponent();

}

private void FormularioCrearCliente\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

this.BackColor = Color.FromArgb(20, 34, 56);

this.label1.ForeColor = Color.White;

this.textBox1.BackColor = Color.FromArgb(20, 34, 56);

this.label2.ForeColor = Color.White;

this.textBox2.BackColor = Color.FromArgb(20, 34, 56);

this.label3.ForeColor = Color.White;

this.textBox3.BackColor = Color.FromArgb(20, 34, 56);

this.label4.ForeColor = Color.White;

this.textBox4.BackColor = Color.FromArgb(20, 34, 56);

this.BotonGuardarCambios.ForeColor = Color.White;

this.BotonGuardarCambios.BackColor = Color.FromArgb(98, 104, 140);

this.BotonCancelar.ForeColor = Color.White;

this.BotonCancelar.BackColor = Color.FromArgb(98, 104, 140);

}

private void BotonGuardarCambios\_Click(object sender, EventArgs e)

{

string nombre, apellido, telefono, direccion;

nombre = this.textBox1.Text;

apellido = this.textBox2.Text;

telefono = this.textBox3.Text;

direccion = this.textBox4.Text;

int variable = 1;

if (nombre != "" && apellido != "" && telefono != "" && direccion != "")

{

if (int.TryParse(telefono, out variable))

{

int telefononumero = int.Parse(telefono);

ClienteBLL cliente = new ClienteBLL();

cliente.CrearCliente(nombre, apellido, telefononumero, direccion);

this.Close();

}

else

{

MessageBox.Show("El teléfono debe ser un número", "Registrar Cliente");

}

}

else

{

MessageBox.Show("Debe completar todos los campos", "Registrar Cliente");

}

}

private void BotonCancelar\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace WindowsFormsApp1

{

public partial class InterfazGeneral : Form

{

private InterfazInicioDeSesion interfaziniciodesesion;

private FormularioPedidos formulariopedidos;

private SeccionClientes.FormularioClientes formularioclientes;

private SeccionGestión.FormularioGestion formulariogestion;

private UsuarioBLL usuario;

public InterfazGeneral(UsuarioBLL pusuario, InterfazInicioDeSesion pinterfaziniciodesesion)

{

InitializeComponent();

this.usuario = pusuario;

this.interfaziniciodesesion = pinterfaziniciodesesion;

}

private void Form1\_Load(object sender, EventArgs e)

{

this.Text = "Che Miranda - Menú Principal";

this.FormBorderStyle = FormBorderStyle.None;

this.CenterToScreen();

int idusuario = this.usuario.Legajo;

this.formulariopedidos = new FormularioPedidos(idusuario);

this.formulariopedidos.TopLevel = false;

this.panel1.BackColor = Color.FromArgb(31, 49, 120);

this.MenuOpciones.BackColor = Color.FromArgb(14, 15, 20);

this.label1.ForeColor = Color.White;

this.label1.Text = "Nombre " + this.usuario.Nombre;

this.CentrarElemento(this.MenuOpciones, this.label1);

this.label2.ForeColor = Color.White;

this.label2.Text = "Legajo "+ this.usuario.Legajo;

this.CentrarElemento(this.MenuOpciones, this.label2);

this.label3.ForeColor = Color.White;

if (this.usuario.Idusuariocategoria == 1)

{

this.label3.Text = "[Dueño]";

}

else

{

this.label3.Text = "[Encargado]";

}

if (this.label3.Text == "[Encargado]")

{

this.BotonGestion.Enabled = false;

}

this.CentrarElemento(this.MenuOpciones, this.label3);

this.BotonPedidos.ForeColor = Color.White;

this.BotonPedidos.BackColor = Color.FromArgb(53, 64, 97);

this.BotonPedidos.FlatAppearance.MouseOverBackColor = Color.FromArgb(95, 109, 149);

this.CentrarElemento(this.MenuOpciones, this.BotonPedidos);

this.BotonClientes.ForeColor = Color.White;

this.BotonClientes.BackColor = Color.FromArgb(53, 64, 97);

this.BotonClientes.FlatAppearance.MouseOverBackColor = Color.FromArgb(95, 109, 149);

this.CentrarElemento(this.MenuOpciones, this.BotonClientes);

this.BotonGestion.ForeColor = Color.White;

this.BotonGestion.BackColor = Color.FromArgb(53, 64, 97);

this.BotonGestion.FlatAppearance.MouseOverBackColor = Color.FromArgb(95, 109, 149);

this.CentrarElemento(this.MenuOpciones, this.BotonGestion);

this.BotonCerrarSesion.ForeColor = Color.White;

this.BotonCerrarSesion.BackColor = Color.FromArgb(219, 23, 9);

this.BotonCerrarSesion.FlatAppearance.MouseOverBackColor = Color.FromArgb(235, 71, 59);

this.CentrarElemento(this.MenuOpciones, this.BotonCerrarSesion);

this.PanelInterfaz.Controls.Add(formulariopedidos);

this.PanelInterfaz.Tag = formulariopedidos;

this.formulariopedidos.Show();

}

private void BotonPedidos\_Click(object sender, EventArgs e)

{

if (!(this.PanelInterfaz.Contains(this.formulariopedidos)))

{

this.PanelInterfaz.Controls.Remove(this.PanelInterfaz.Controls[0]);

int idusuario = this.usuario.Legajo;

this.formulariopedidos = new FormularioPedidos(idusuario);

this.formulariopedidos.TopLevel = false;

this.PanelInterfaz.Controls.Add(formulariopedidos);

this.PanelInterfaz.Tag = formulariopedidos;

this.formulariopedidos.Show();

}

}

private void BotonClientes\_Click(object sender, EventArgs e)

{

if (!(this.PanelInterfaz.Contains(this.formularioclientes)))

{

this.PanelInterfaz.Controls.Remove(this.PanelInterfaz.Controls[0]);

this.formularioclientes = new SeccionClientes.FormularioClientes();

this.formularioclientes.TopLevel = false;

this.PanelInterfaz.Controls.Add(this.formularioclientes);

this.PanelInterfaz.Tag = this.formularioclientes;

this.formularioclientes.Dock = DockStyle.Fill;

this.formularioclientes.Show();

}

}

private void BotonGestion\_Click(object sender, EventArgs e)

{

if (!(this.PanelInterfaz.Contains(this.formulariogestion)))

{

this.PanelInterfaz.Controls.Remove(this.PanelInterfaz.Controls[0]);

this.formulariogestion = new SeccionGestión.FormularioGestion();

this.formulariogestion.TopLevel = false;

this.PanelInterfaz.Controls.Add(this.formulariogestion);

this.PanelInterfaz.Tag = this.formulariogestion;

this.formulariogestion.Show();

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

this.interfaziniciodesesion.Close();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

public void CentrarElemento(Control ppadre, Control phijo)

{

int posx;

posx = (ppadre.Width / 2) - (phijo.Width / 2);

phijo.Location = new System.Drawing.Point(posx, phijo.Location.Y);

}

private void BotonCerrarSesion\_Click(object sender, EventArgs e)

{

this.Close();

this.interfaziniciodesesion.Visible = true;

}

private void MenuOpciones\_Paint(object sender, PaintEventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CapaDeNegocios;

namespace WindowsFormsApp1

{

public partial class InterfazInicioDeSesion : Form

{

private List<UsuarioBLL> listadeusuarios;

public List<UsuarioBLL> Listadeusuarios { get => listadeusuarios; set => listadeusuarios = value; }

public InterfazInicioDeSesion()

{

InitializeComponent();

}

private void InterfazInicioDeSesion\_Load(object sender, EventArgs e)

{

this.Text = "Che Miranda - Iniciar Sesión";

this.FormBorderStyle = FormBorderStyle.None;

this.MaximizeBox = false;

this.CenterToScreen();

this.BackColor = Color.FromArgb(20, 34, 56);

this.panel1.BackColor = Color.FromArgb(73, 85, 117);

this.panel2.BackColor = Color.FromArgb(73, 85, 117);

this.CentrarElemento(this, this.panel2);

this.label1.Text = "Che Miranda";

this.CentrarElemento(this, this.label1);

this.label1.ForeColor = Color.White;

this.label2.Text = "Nombre de Usuario";

this.label2.ForeColor = Color.White;

this.textBox1.BackColor = Color.FromArgb(73, 85, 117);

this.label3.Text = "Contraseña";

this.label3.ForeColor = Color.White;

this.textBox2.PasswordChar = '\*';

this.textBox2.BackColor = Color.FromArgb(73, 85, 117);

this.BotonIniciarSesion.Text = "Iniciar Sesión";

this.BotonIniciarSesion.ForeColor = Color.White;

this.BotonIniciarSesion.BackColor = Color.FromArgb(53, 64, 97);

this.BotonIniciarSesion.FlatAppearance.MouseOverBackColor = Color.FromArgb(95, 109, 149);

this.BotonIniciarSesion.TabStop = false;

this.label4.Text = "Guilerova Solutions - Versión 1.0";

this.CentrarElemento(this, this.label4);

this.label4.ForeColor = Color.White;

}

private void BotonIniciarSesion\_Click(object sender, EventArgs e)

{

// Cargar Usuarios

UsuarioBLL usuariobll = new UsuarioBLL();

this.listadeusuarios = usuariobll.CargarUsuarios();

string nombredeusuario;

string contraseña;

nombredeusuario = this.textBox1.Text;

contraseña = this.textBox2.Text;

if (nombredeusuario == "" || contraseña == "")

{

MessageBox.Show("Debe ingresar el nombre de usuario o la contraseña", "Iniciar Sesión");

}

else

{

UsuarioBLL usuario = null;

int contadorusuarios = 0;

foreach (UsuarioBLL usuarioelemento in this.listadeusuarios)

{

contadorusuarios++;

if ((nombredeusuario == usuarioelemento.Nombre) && (contraseña == usuarioelemento.Contraseña.ToString()))

{

usuario = usuarioelemento;

break;

}

}

if (usuario == null)

{

MessageBox.Show("Usuario no registrado", "Iniciar Sesión");

this.textBox1.Text = "";

this.textBox2.Text = "";

}

else

{

InterfazGeneral interfazgeneral = new InterfazGeneral(usuario, this);

interfazgeneral.Show();

this.Visible = false;

this.textBox1.Text = "";

this.textBox2.Text = "";

}

}

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

public void CentrarElemento(Control ppadre, Control phijo)

{

int posx;

posx = (ppadre.Width / 2) - (phijo.Width / 2);

phijo.Location = new System.Drawing.Point(posx, phijo.Location.Y);

}

}

}

#### SENTENCIAS SQL

USE [Che Miranda]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Cadete] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cadete](

[idcadete] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](20) NOT NULL,

[apellido] [varchar](20) NOT NULL,

[telefono] [int] NOT NULL,

[idrodado] [int] NOT NULL,

[licencia] [int] NULL,

CONSTRAINT [PK\_Cadete] PRIMARY KEY CLUSTERED

(

[idcadete] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Cliente] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Cliente](

[idcliente] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](20) NOT NULL,

[apellido] [varchar](20) NOT NULL,

[telefono] [int] NOT NULL,

[direccion] [varchar](50) NOT NULL,

CONSTRAINT [PK\_Cliente] PRIMARY KEY CLUSTERED

(

[idcliente] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[DetallePedido] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[DetallePedido](

[iddetallepedido] [int] IDENTITY(1,1) NOT NULL,

[idproducto] [int] NOT NULL,

[idpedido] [int] NOT NULL,

[cantidad] [int] NOT NULL,

[precioventa] [int] NOT NULL,

CONSTRAINT [PK\_\_DetalleP\_\_47D240BD3F466844] PRIMARY KEY CLUSTERED

(

[idpedido] ASC,

[iddetallepedido] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[EstadoCategoria] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[EstadoCategoria](

[idestadocategoria] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](50) NOT NULL,

CONSTRAINT [PK\_EstadoCategoria] PRIMARY KEY CLUSTERED

(

[idestadocategoria] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[FormaPago] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[FormaPago](

[idformadepago] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](20) NOT NULL,

CONSTRAINT [PK\_FormaPago] PRIMARY KEY CLUSTERED

(

[idformadepago] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Pedido] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Pedido](

[idpedido] [int] IDENTITY(1,1) NOT NULL,

[idformadepago] [int] NOT NULL,

[idcliente] [int] NOT NULL,

[idusuario] [int] NOT NULL,

[fechayhora] [datetime] NOT NULL,

[observaciones] [varchar](100) NOT NULL,

[idcadete] [int] NOT NULL,

[idestadocategoria] [int] NOT NULL,

CONSTRAINT [PK\_Pedido] PRIMARY KEY CLUSTERED

(

[idpedido] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Producto] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Producto](

[idproducto] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](50) NOT NULL,

[idproductocategoria] [int] NOT NULL,

[precio] [int] NOT NULL,

CONSTRAINT [PK\_Producto] PRIMARY KEY CLUSTERED

(

[idproducto] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[ProductoCategoria] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ProductoCategoria](

[idproductocategoria] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](50) NOT NULL,

CONSTRAINT [PK\_ProductoCategoria] PRIMARY KEY CLUSTERED

(

[idproductocategoria] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Rodado] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Rodado](

[idrodado] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](20) NULL,

[patente] [varchar](20) NULL,

CONSTRAINT [PK\_Rodado] PRIMARY KEY CLUSTERED

(

[idrodado] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Usuario] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Usuario](

[idusuario] [int] IDENTITY(1,1) NOT NULL,

[nombre] [varchar](20) NOT NULL,

[apellido] [varchar](20) NOT NULL,

[CUIL] [int] NOT NULL,

[numerodecontacto] [int] NOT NULL,

[fechadecontratacion] [date] NOT NULL,

[idcategoriausuario] [int] NOT NULL,

CONSTRAINT [PK\_Usuario] PRIMARY KEY CLUSTERED

(

[idusuario] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[UsuarioCategoria] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[UsuarioCategoria](

[idcategoriausuario] [int] IDENTITY(1,1) NOT NULL,

[descripcion] [varchar](20) NOT NULL,

CONSTRAINT [PK\_UsuarioCategoria] PRIMARY KEY CLUSTERED

(

[idcategoriausuario] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Cadete] WITH CHECK ADD CONSTRAINT [FK\_Cadete\_Rodado] FOREIGN KEY([idrodado])

REFERENCES [dbo].[Rodado] ([idrodado])

GO

ALTER TABLE [dbo].[Cadete] CHECK CONSTRAINT [FK\_Cadete\_Rodado]

GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK\_Pedido\_Cadete] FOREIGN KEY([idcadete])

REFERENCES [dbo].[Cadete] ([idcadete])

GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK\_Pedido\_Cadete]

GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK\_Pedido\_Cliente] FOREIGN KEY([idcliente])

REFERENCES [dbo].[Cliente] ([idcliente])

GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK\_Pedido\_Cliente]

GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK\_Pedido\_EstadoCategoria] FOREIGN KEY([idestadocategoria])

REFERENCES [dbo].[EstadoCategoria] ([idestadocategoria])

GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK\_Pedido\_EstadoCategoria]

GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK\_Pedido\_FormaPago] FOREIGN KEY([idformadepago])

REFERENCES [dbo].[FormaPago] ([idformadepago])

GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK\_Pedido\_FormaPago]

GO

ALTER TABLE [dbo].[Pedido] WITH CHECK ADD CONSTRAINT [FK\_Pedido\_Usuario] FOREIGN KEY([idusuario])

REFERENCES [dbo].[Usuario] ([idusuario])

GO

ALTER TABLE [dbo].[Pedido] CHECK CONSTRAINT [FK\_Pedido\_Usuario]

GO

ALTER TABLE [dbo].[Producto] WITH CHECK ADD CONSTRAINT [FK\_Producto\_ProductoCategoria] FOREIGN KEY([idproductocategoria])

REFERENCES [dbo].[ProductoCategoria] ([idproductocategoria])

GO

ALTER TABLE [dbo].[Producto] CHECK CONSTRAINT [FK\_Producto\_ProductoCategoria]

GO

ALTER TABLE [dbo].[Usuario] WITH CHECK ADD CONSTRAINT [FK\_Usuario\_UsuarioCategoria] FOREIGN KEY([idcategoriausuario])

REFERENCES [dbo].[UsuarioCategoria] ([idcategoriausuario])

GO

ALTER TABLE [dbo].[Usuario] CHECK CONSTRAINT [FK\_Usuario\_UsuarioCategoria]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarBebidas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarBebidas]

as

select \* from Producto

where idproductocategoria = 24

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarCadetes] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarCadetes]

as

begin

select \* from Cadete

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarCadeteYRodado] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE procedure [dbo].[SPBuscarCadeteYRodado]

as

begin

select Cadete.idcadete, Cadete.nombre, Cadete.apellido, Cadete.telefono, Rodado.idrodado, Rodado.descripcion, Rodado.patente

from Cadete join Rodado on Cadete.idrodado = rodado.idrodado

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarCalzones] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarCalzones]

as

select \* from Producto

where idproductocategoria = 19

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarClientes] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE procedure [dbo].[SPBuscarClientes]

as

begin

select \* from Cliente

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEmpanadas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEmpanadas]

as

select \* from Producto

where idproductocategoria = 21

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEnsaladas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEnsaladas]

as

select \* from Producto

where idproductocategoria = 12

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEnsaladasEsp] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEnsaladasEsp]

as

select \* from Producto

where idproductocategoria = 13

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEntradas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE procedure [dbo].[SPBuscarEntradas]

as

select \* from Producto

where idproductocategoria = 1

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEspecialesI] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEspecialesI]

as

select \* from Producto

where idproductocategoria = 15

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEspecialesII] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEspecialesII]

as

select \* from Producto

where idproductocategoria = 16

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarEstadosCategorias] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarEstadosCategorias]

as

begin

select \* from EstadoCategoria

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarFormasDePago] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarFormasDePago]

as

begin

select \* from FormaPago

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarFugazzas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarFugazzas]

as

select \* from Producto

where idproductocategoria = 17

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarFugazzetas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarFugazzetas]

as

select \* from Producto

where idproductocategoria = 18

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarGuarnicion] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarGuarnicion]

as

select \* from Producto

where idproductocategoria = 8

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarMatambre] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarMatambre]

as

select \* from Producto

where idproductocategoria = 6

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarMenus] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarMenus]

as

select \* from Producto

where idproductocategoria = 26

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarMilanesas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarMilanesas]

as

select \* from Producto

where idproductocategoria = 4

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarOmelletes] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarOmelletes]

as

select \* from Producto

where idproductocategoria = 2

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarOtrasEspecialidades] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarOtrasEspecialidades]

as

select \* from Producto

where idproductocategoria = 20

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarParillas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarParillas]

as

select \* from Producto

where idproductocategoria = 10

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarPastas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarPastas]

as

select \* from Producto

where idproductocategoria = 23

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarPescados] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarPescados]

as

select \* from Producto

where idproductocategoria = 11

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarPollos] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarPollos]

as

select \* from Producto

where idproductocategoria = 9

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarPostres] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarPostres]

as

select \* from Producto

where idproductocategoria = 25

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarPromos] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarPromos]

as

select \* from Producto

where idproductocategoria = 27

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarRodados] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarRodados]

as

begin

select \* from Rodado

end

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarSandwiches] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarSandwiches]

as

select \* from Producto

where idproductocategoria = 22

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarSupremas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarSupremas]

as

select \* from Producto

where idproductocategoria = 5

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarTortillas] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarTortillas]

as

select \* from Producto

where idproductocategoria = 3

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPBuscarTradicionales] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create procedure [dbo].[SPBuscarTradicionales]

as

select \* from Producto

where idproductocategoria = 14

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPCrearDetallePedido] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[SPCrearDetallePedido](

@idproducto INT,

@idpedido INT,

@cantidad INT,

@precioventa INT

)

AS

BEGIN

INSERT INTO DetallePedido(

[idproducto],

[idpedido],

[cantidad],

[precioventa]

)

VALUES (

@idproducto,

@idpedido,

@cantidad,

@precioventa

)

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[SPCrearPedido] Script Date: 20/11/2020 14:21:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[SPCrearPedido](

@idformadepago INT,

@idcliente INT,

@idusuario INT,

@fechayhora DATETIME,

@observaciones VARCHAR(100),

@idcadete INT,

@idestadocategoria INT

)

AS

BEGIN

INSERT INTO Pedido(

[idformadepago],

[idcliente],

[idusuario],

[fechayhora],

[observaciones],

[idcadete],

[idestadocategoria]

)

VALUES (

@idformadepago,

@idcliente,

@idusuario,

@fechayhora,

@observaciones,

@idcadete,

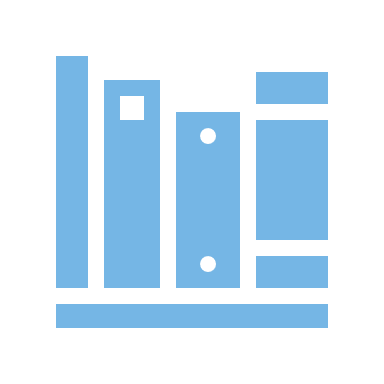
@idestadocategoria

)

END

GO

## **DICCIONARIO DE DATOS**



### Diccionario de datos:

| Tabla: Cadete  Descripción: Tabla que contendrá los datos de los Cadete Relación: Tabla Pedidos campo IDcadete | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *Idcadete -PK* | int | No | Id del cadete |
| nombre | varchar(20) | No | Nombre del Cadete |
| apellido | varchar(20) | No | Apellido del Cadete |
| telefono | int | No | Telefono del Cadete |
| idrodado | int | No |  |
| licencia | int | Si | Licencia del del Cadete |

| Tabla: Cliente  Descripción: Tabla que contendrá los datos de los Clientes Relación: Tabla Pedido campo IDCliente | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *idcliente -PK* | int | No | Id del Cliente |
| nombre | varchar(20) | No | Nombre del Cliente |
| apellido | varchar(20) | No | Apellido del Cliente |
| telefono | int | No | Telefono del Cliente |
| direccion | varchar(50) | No | Direccion del Cliente |

| Tabla: Detalle Pedido  Descripción: Tabla que contendrá los datos de cada producto perteneciente a un pedido  Relación: Tabla Productos campo IDProducto // Tabla Pedido campo IDPedido | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *Iddetallepedido - PK* | int | No | Id del detalle pedido |
| Idproducto-FK | int | No | ID del Producto |
| *Idpedido -PK* | int | No | Id del pedido |
| cantidad | int | No | Cantidad del producto que contiene |
| precioventa | int | No | Precio unitario del producto |

| Tabla: EstadoCategoria  Descripción: Tabla que los distintos estados de un pedido Relación: Tabla Pedido campo IDEstadoCategoria | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *idestadocategoria -PK* | int | No | Id del estado categoría del pedido |
| descripcion | varchar(50) | No | Nombre del estado |

| Tabla: FormadePago  Descripción: Tabla que contendrá las distintas formas de pago de un pedido Relación: Tabla Pedido campo IDForma de pago | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *idformadepago -PK* | int | No | ID de la forma de pago |
| descripcion | varchar(20) | No | Forma de pago que podrá tener un pedido |

| Tabla: Pedidos  Descripción: Tabla que contendrá los datos de los Pedidos. | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *idpedido -PK* | int | No | Id del pedido |
| Idformadepago-FK | int | No | Id de la forma de pago del pedido |
| Idcliente-FK | int | No | Id del cliente |
| Idusuario-FK | int | No | Id del usuario |
| Fechayhora | datetime | No | Fecha y hora del pedido |
| observaciones | varchar(100) | No | Aclaraciones del pedido |
| idcadete | int | No | ID del cadete que entregara el pedido |
| idestadocategoria | int | No | Estado del pedido |

| Tabla: Productos  Descripción: Tabla que contendrá los datos de los productos Relación: Tabla DetallePedido campo IDProducto | | | | |
| --- | --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** | **Descripción** |
| *idproducto -PK* | int | No | ID del producto |
| descripcion | varchar(50) | No | Nombre del producto |
| Idproductocategoria -FK | int | No | ID de la categoría del producto |
| precio | int | No | Precio del producto |

| Tabla: ProductoCategoria  Descripción: Tabla que contendrá las categorías de los productos Relación: Tabla Productos campo Idproductocategoria | | | |
| --- | --- | --- | --- |
| Campos | Tipo de datos | Permite nulo |  |
| *idproductocategoria -PK* | int | No | ID de la categoría del producto |
| descripcion | varchar(50) | No | Nombre de la categoría del producto |

| Tabla: Usuario  Descripción: Tabla que contendrá los datos de los usuarios Relación: Tabla Pedido campo IDUsuario | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** |  |
| *idusuario -PK* | int | No | ID del usuario |
| nombre | varchar(20) | No | Nombre del del usuario |
| apellido | varchar(20) | No | Apellido del usuario |
| CUIL | int | No | CUIL del usuario |
| numerodecontacto | int | No | Telefono del usuario |
| fechadecontratacion | date | No | Fecha de contratación del usuario |
| idcategoriausuario | int | No | ID de la categoría del usuario |

| Tabla: CategoriaUsuario  Descripción: Tabla que contendrá las categorías de los usuarios Relación: Tabla Usuario campo IDcategoriausuario | | | |
| --- | --- | --- | --- |
| Campos | **Tipo de datos** | **Permite nulo** |  |
| *idcategoriausuario -PK* | int | No | ID de la categoría del usuario |
| descripcion | varchar(20) | No | Nombre de la categoria |

## **MANUAL DE PROCEDIMIENTO**



### Manual de Procedimiento:

Sobre la aplicación Che Miranda:

La aplicación Che Miranda es un programa concebido para registrar, analizar y presentar datos relacionados con la situación y las necesidades de la entrega de comida por delivery. Es fácil aprender a usar este software, pero se asume que el usuario estará ya familiarizado con las rutinas, datos y procedimiento que se necesitan para cada función. Se debe estudiar detenidamente dicho documento antes de empezar a usar el software. En esta guía del usuario se ofrece una visión general de las características de la aplicación y se indican las instrucciones que deben seguirse paso a paso para realizar diversas tareas.

#### Inicio de Sesión

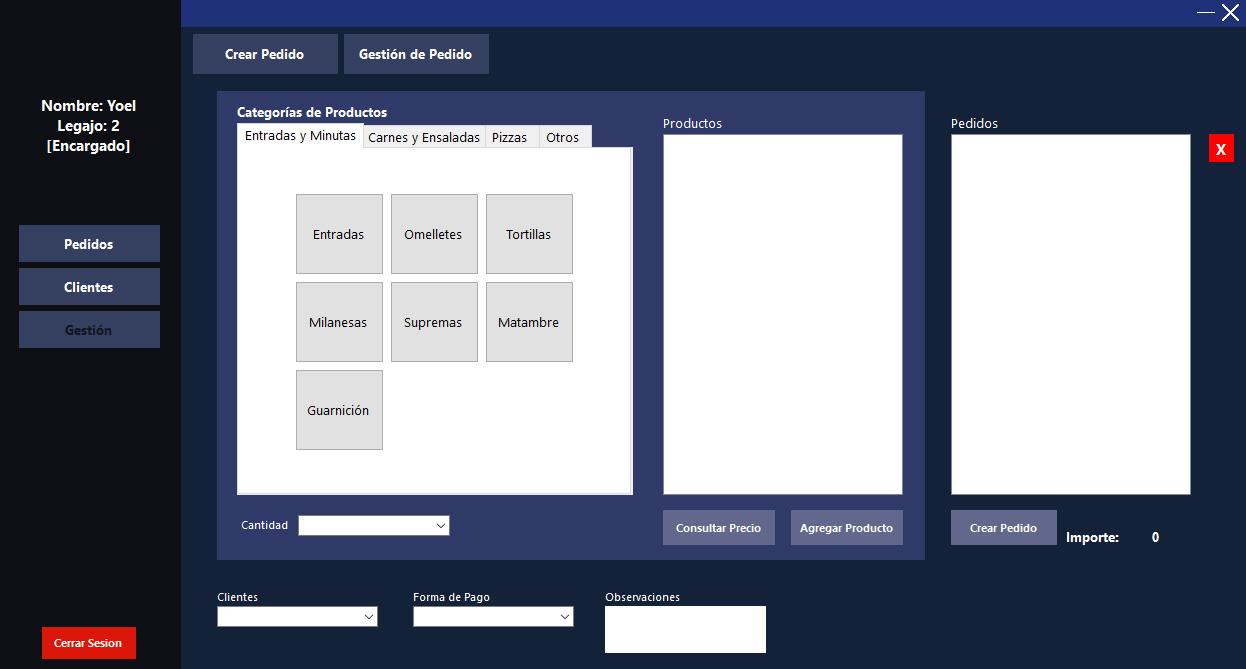
Para acceder al software que necesita un usuario y contraseña que será otorgada por el dueño.

#### Menú Principal

La sección que carga de manera predeterminada el software es la de Pedidos, con su subsección de “Crear Pedido” ya que es la funcionalidad principal. Se puede observar el botón “Cerrar Sesión” en la parte inferior izquierda. En caso de que se haga clic sobre él, se cerrará la sesión del usuario que previamente la inicio, y se volverá a la parte de inicio de sesión del software.

#### Pedidos

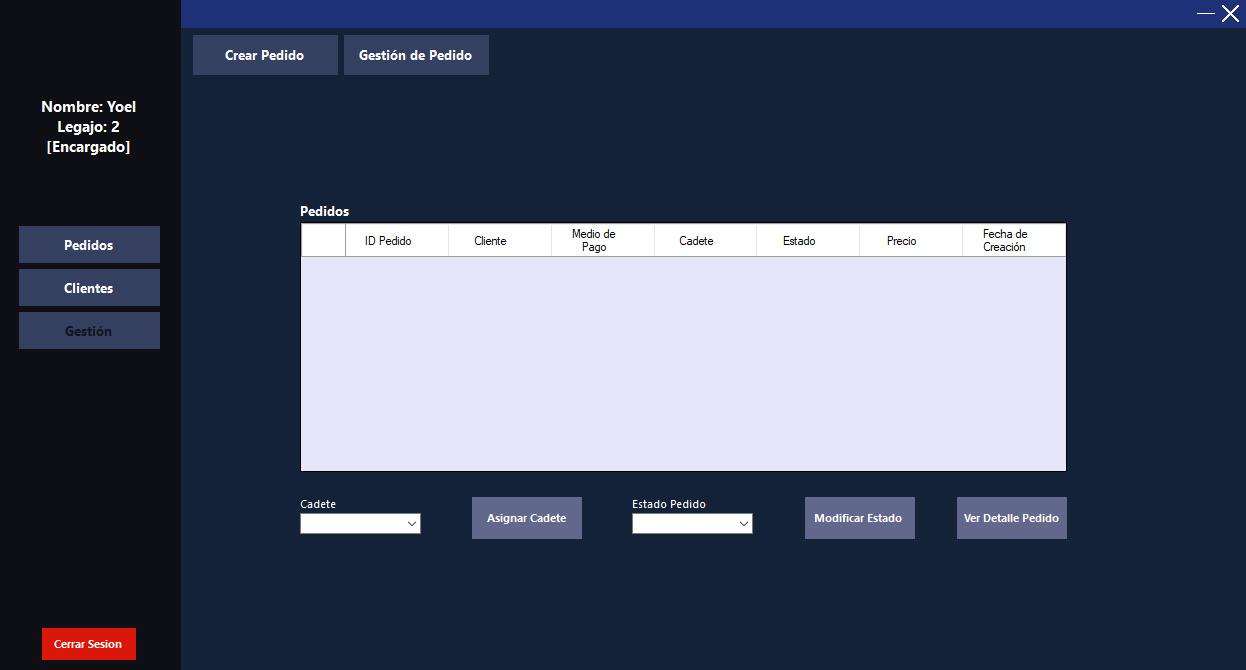
##### Crear Pedido



Se deberá elegir los productos necesarios para el pedido, estos están clasificados por categoría. A cada producto seleccionado se le deberá asignar una cantidad, que por defecto es 1 y luego apretar el botón de Agregar. Si se requiere saber el precio de un producto tiene la posibilidad de consultarlo con el botón consultar. Si algún ítem agregado necesita ser eliminado, puede seleccionar del cuadro de Pedido y apretar el botón con la cruz.

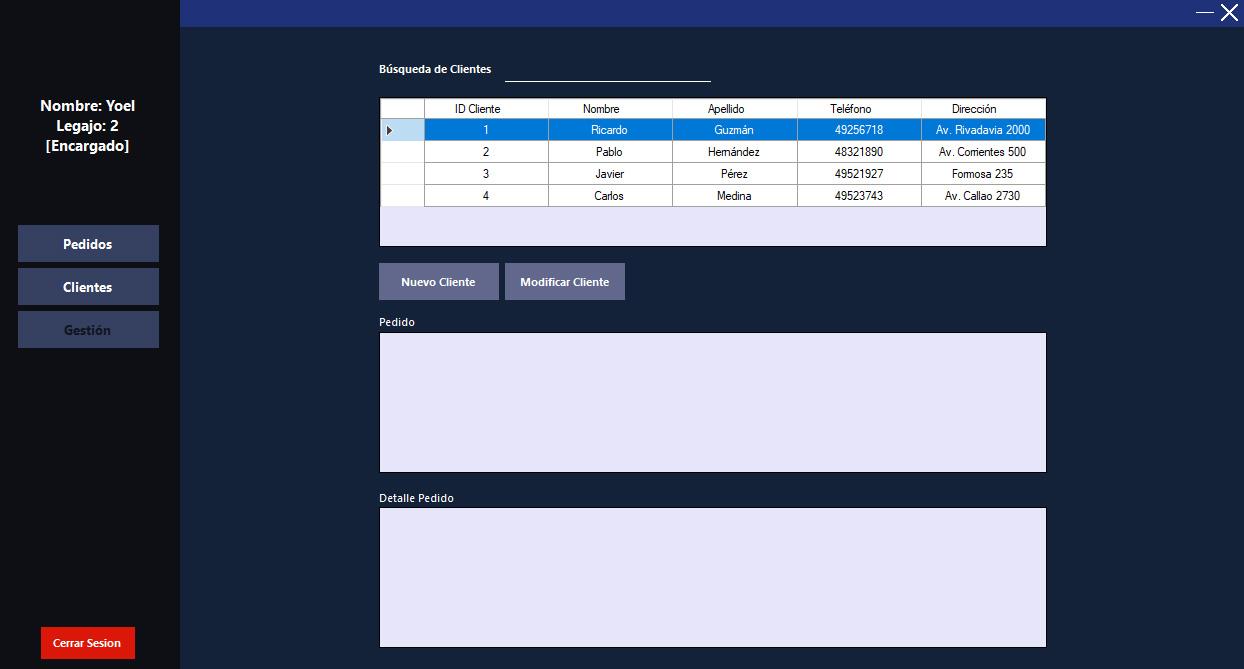
Luego se le deberá asignar un cliente, una forma de pago y si es necesaria una observación. A continuación de esto ya podrá hacer click en “Crear Pedido” para la creación del mismo.

##### Gestión de Pedido



En este apartado de la sección de pedidos, el usuario podrá asignarle un cadete al pedido ya preparado para tener un control de este. A su vez podrá cambiar los estados de los pedidos entre: Inicializado, Delivery, Cancelado y Finalizado según se requiera. Al final del día todos los pedidos deberán quedar finalizados o en su defecto en cancelado. También se podrá visualizar los alimentos que forman parte de cada pedido.

#### Clientes



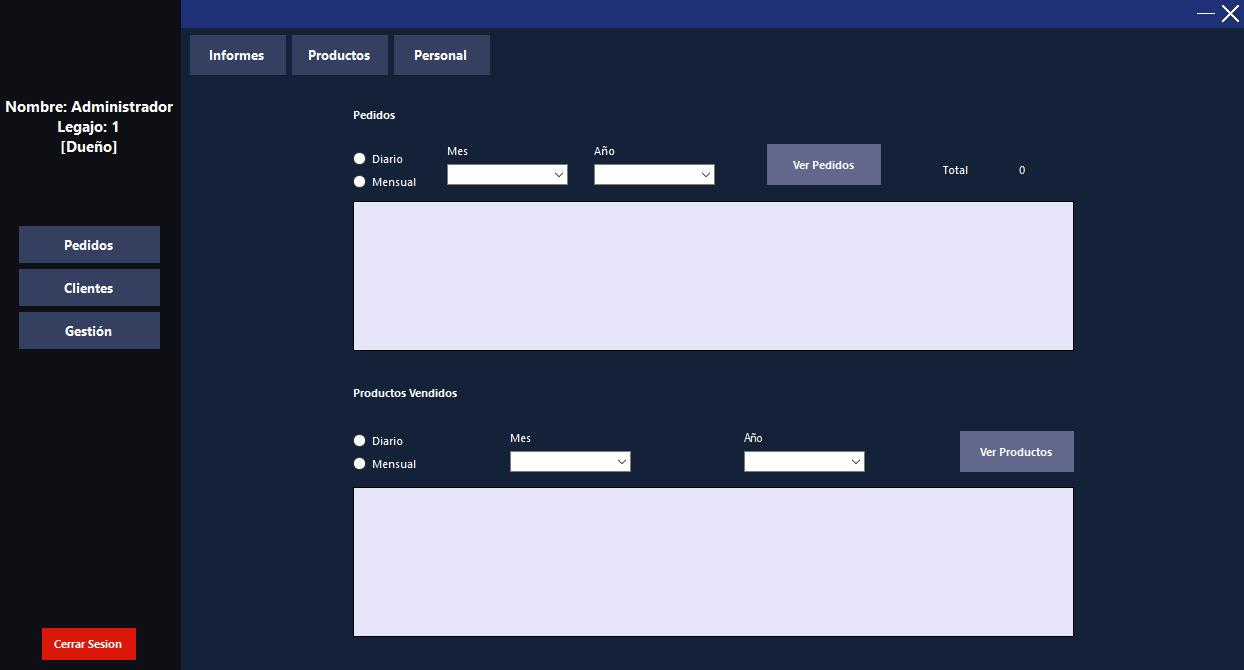
En esta sección se podrá agregar, modificar y crear clientes nuevos para después poder asignarle un pedido. No obstante, para asignarle un pedido a un cliente, primero este debe ser creado por medio de la sección de Pedidos. Los clientes se crean por medio del botón “Nuevo Cliente”, se modifican por medio del botón “Modificar Cliente” y por último se eliminan por medio del botón “Eliminar Cliente”.

En caso de seleccionarse un cliente en el primer rectángulo, el cual posee todos los registros de los clientes del restaurante, en los siguientes dos rectángulos aparecerán el pedido asociado al cliente y en el tercer rectángulo el detalle pedido asociado a ese pedido en particular.

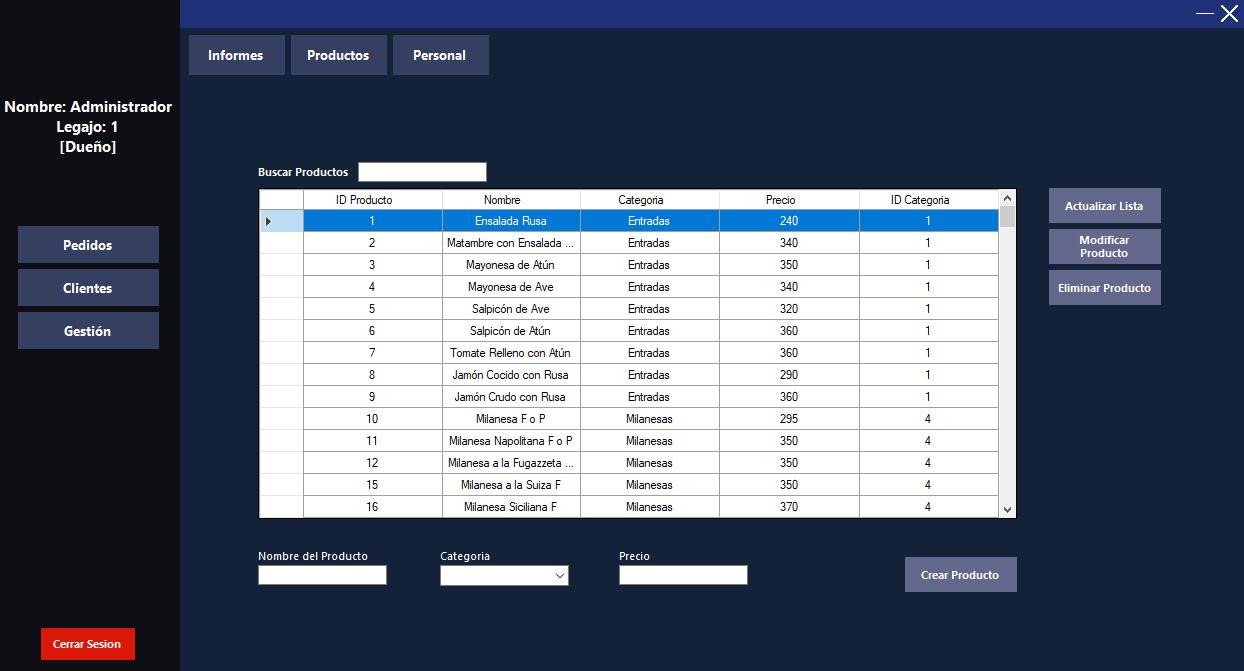
#### Gestión

Esta sección sólo podrá ser ingresada sólo por un usuario que tenga el rol de dueño. La primera subsección cargada es la de los informes, que obviamente proporcionan información útil acerca de los pedidos y los productos que pueden determinar el rumbo del restaurante al tomar decisiones basándose en ella. Ya en esta subsección de informes de la sección de gestión, los elementos que comprenden a la misma son los pedidos y los productos. En cuanto a los primeros se podrá obtener información acerca de aquellos de los cuales se realizaron ventas, filtrándose por un periodo de tiempo ya sea diario o mensual. Para ello elegir la opción “Diario” y luego hacer clic en el botón “Ver Pedidos” en caso de que se desee verlos. Se obtendrá información sobre el día actual. Para generar el reporte en un papel, se deberá hacer clic en el botón “Generar Reporte”. En caso de que desee obtener información mensual acerca de los pedidos se deberá seleccionar la opción “Mensual” y luego especificar el mes y el año, ambos serán una lista de opciones para seleccionar los valores deseados. Si se desea ver la información se deberá hacer clic en el botón en “Ver Pedidos”. Si se desea generar un reporte por medio de un papel se deberá hacer clic en el botón “Generar Reporte”.

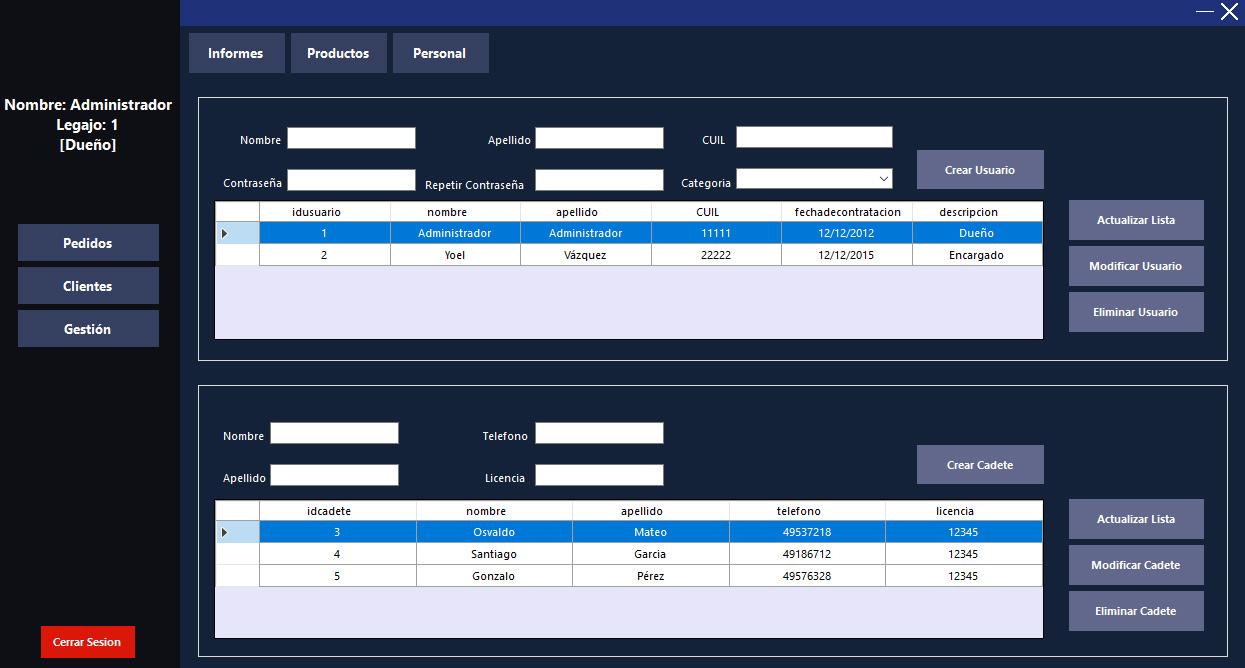
##### Informes



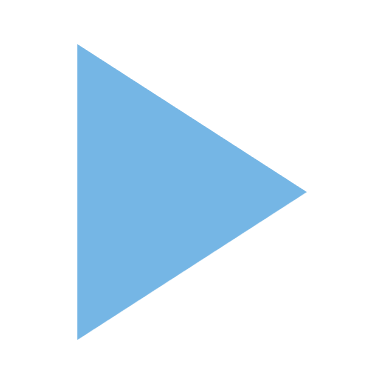
##### Productos



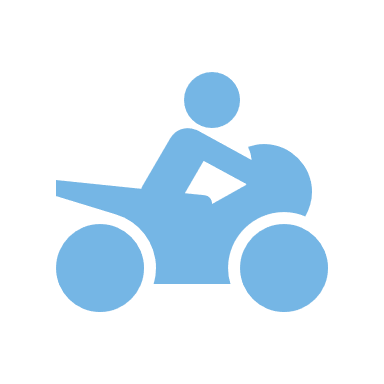
##### Personal



# **ETAPA DE IMPLEMENTACION**



## **PLANEAMIENTO DE PUESTA EN MARCHA**



### Planeamiento: Informe correspondiente acerca del método a emplear para la puesta en marcha del sistema.

El sistema desarrollado para el restaurante “Che Miranda” está enfocado principalmente al sector de ventas por Delivery.

El sistema se implementará una vez terminado y tendrá como tarea principal brindar mayor optimización a las tareas relacionados con  el sector Delivery. Funcionando en conjunto con los procesos actuales de venta en el salón del restaurante. Por otra parte,  será necesaria la capacitación del personal encargado C ejecutar el sistema. La idea de este es optimizar el tiempo reemplazando tareas que hasta el momento se realizan de forma manual.

#### Especificaciones para el funcionamiento del sistema:

1. Configuración y puesta en marcha de la base de datos SQL en la terminal local.
2. Capacitación del personal.
3. Entrega de manuales de uso.
4. Soporte técnico durante los primeros meses, atendiendo las necesidades que vayan surgiendo.
5. Entrega de usuario administrador al usuario que tendrá este rol.
6. Carga inicial de datos de la empresa en la base de datos del cliente.
7. Configuración de impresoras para la impresión de tickets y comandas.

## **COMERCIALIZACION DEL SISTEMA**Apretón de manos

### 

### 

### 

### 

### 

### 

### 

### 

### 

### Posibilidades de comercialización del Sistema diseñado.

El sistema solicitado y desarrollado para el restaurante “Che Miranda” tiene una alta probabilidad comercialización, ya que consideramos que es de gran atractivo para el sector gastronómico en general. Cumpliendo con las pautas demandadas, cubriendo las necesidades planteadas y a un costo accesible para el cliente.

En cada una de las distintas reuniones entre los propietarios de “Che Miranda” y representantes de Guilerova, se coordinaron y alinearon los distintos requerimientos solicitados para solucionar la problemática, logrando un sistema acorde al cliente.

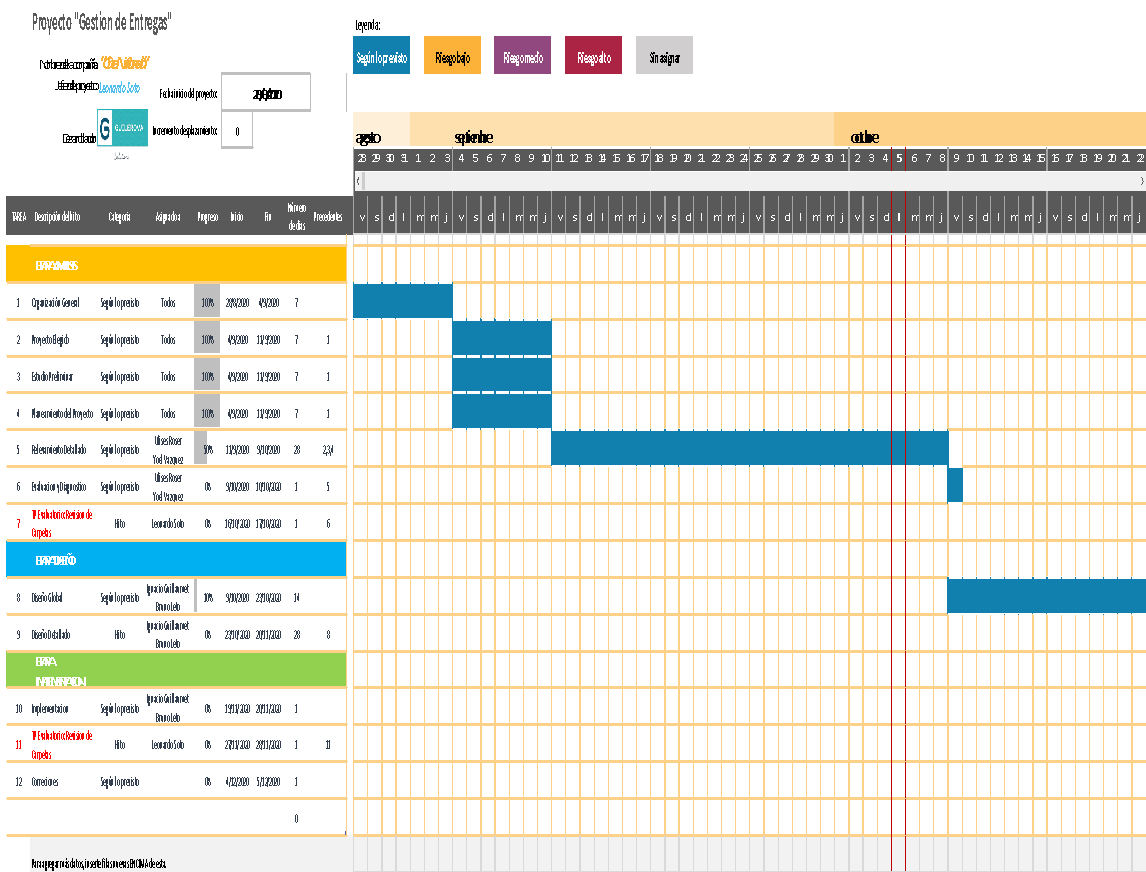
Al ser un sistema exclusivo del sector delivery, caracterizado por una gran demanda y que se encuentra en pleno auge en la actualidad, evaluamos la posibilidad de ofrecerlo a todo el sector gastronómico, ya que al ser un sistema que optimiza al sector, solo con unos pequeños cambios para cada caso en particular podría ser utilizado sin problema.

##### Servicios adicionales al sistema propuesto:

Una vez implementado el sistema, Guilerova Solutions capacitará a los empleados y ofrecerá un servicio de soporte técnico y mantenimiento , por 3 meses con posibilidad de extensión según requerimientos.

## **GRAFICO DE ACTIVIDADES**Diagrama de Gantt contorno

Gráficos de actividades

*Gantt y/o Camino Crítico*

### aGantt FINAL

# **GRACIAS**





**¡POR ELEGIRNOS!**