

# MANUAL DEL PROGRAMADOR

**Proyecto Recetario**

**Creado por:**

Alberto Rodríguez Pérez

Renzo Sandoval Villanueva

# DIAGRAMA DE CLASES

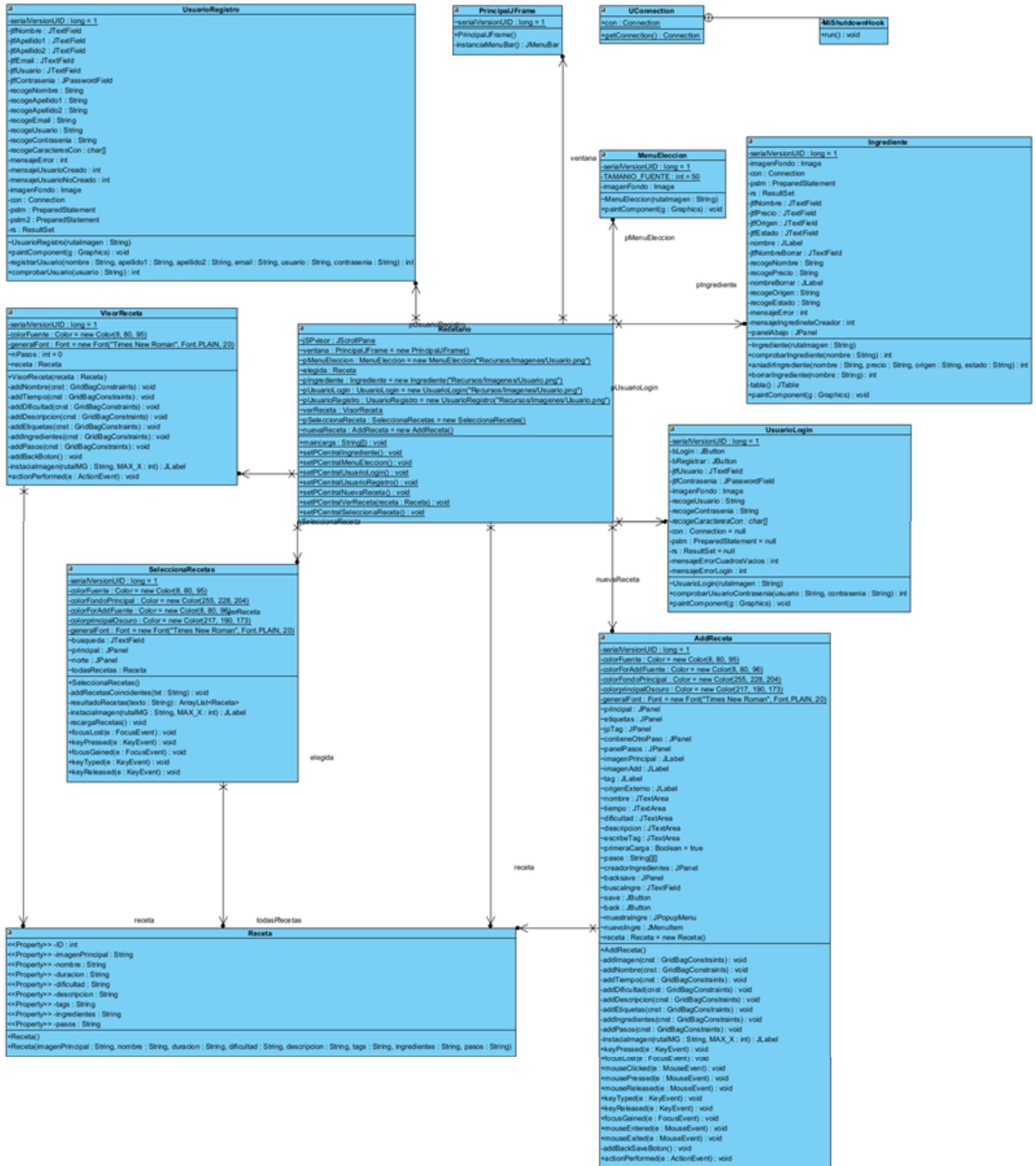
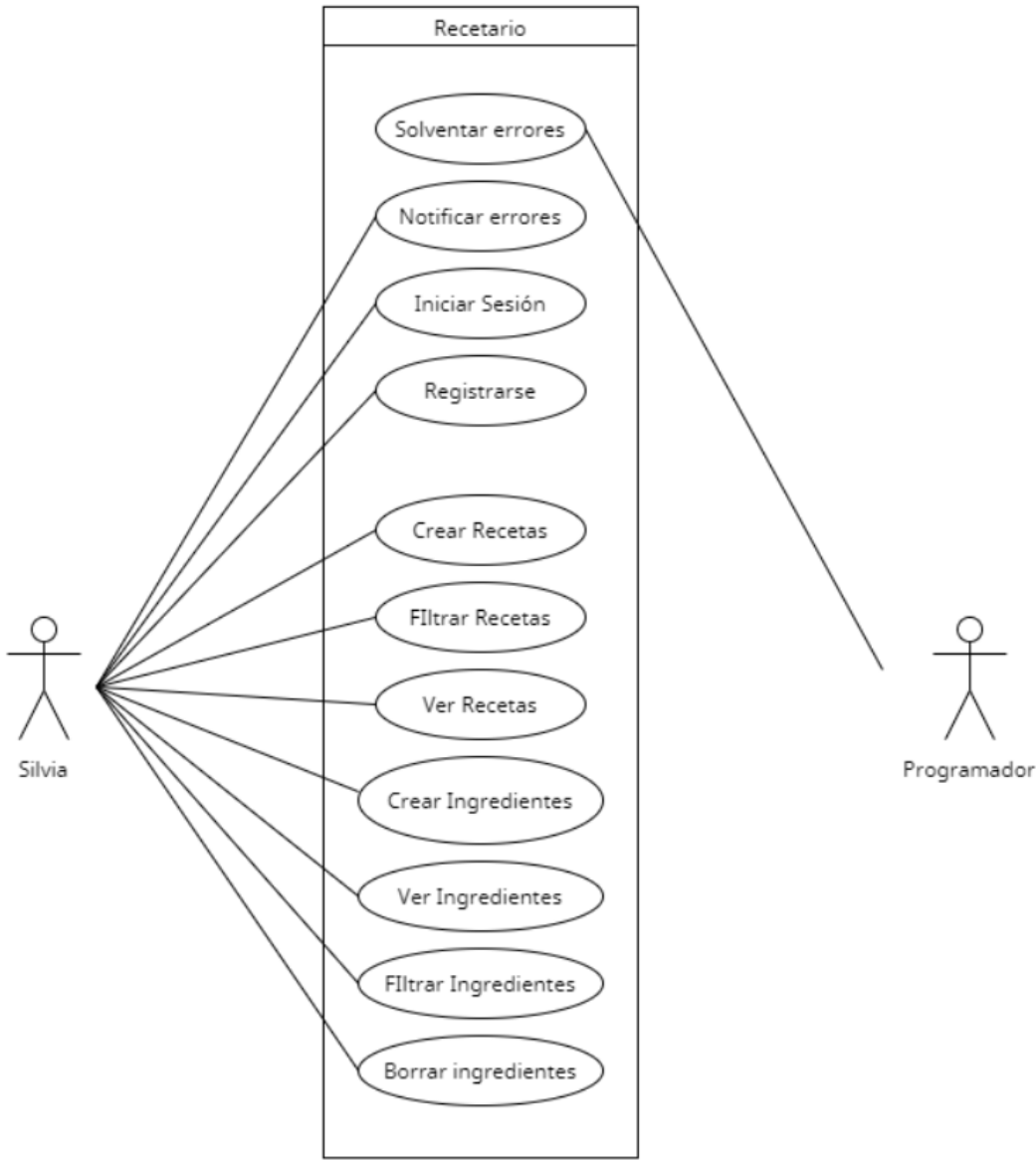


Diagrama de casos de uso



## Documentación de 2 casos de uso

Primer caso - **Inicio de sesión:**

Caso	Inicio de sesión
Autor	Renzo Sandoval
Descripción	Pantalla para hacer conexión con la base de datos
Condición previa	Tener un usuario registrado en la aplicación
Excepciones	Fallo en la contraseña, base de datos no iniciada, fallo de la conexión, usuario no creado.

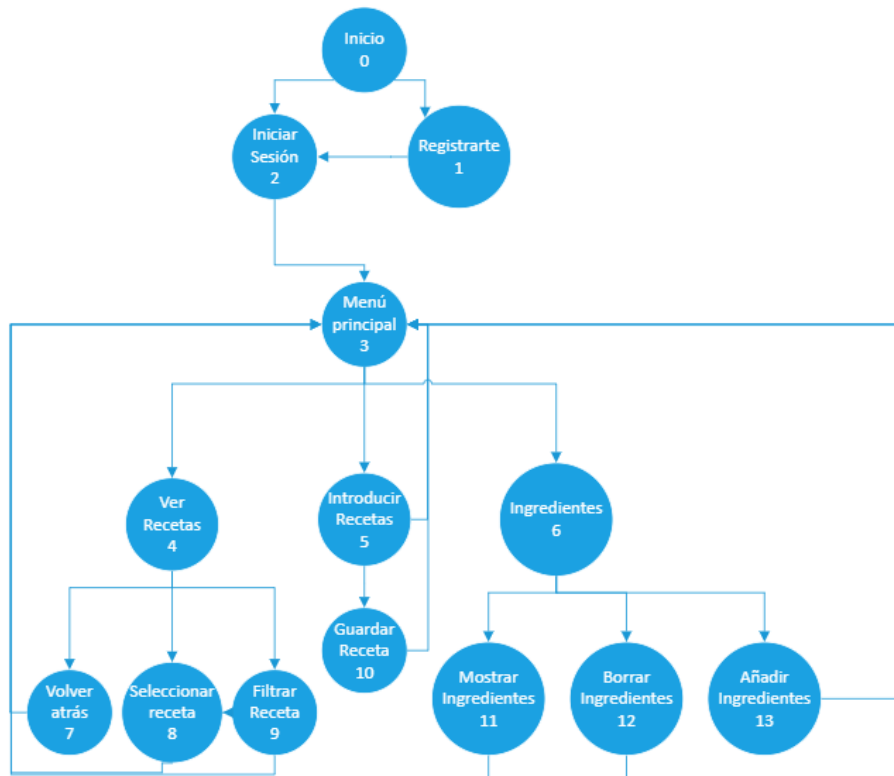
Segundo caso - **Ver una receta:**

Caso	Ver receta
Autor	Alberto Rodríguez
Descripción	Ver una receta insertada en la base de datos
Condición previa	Debe haber una receta introducida
Excepciones	No hay recetas, está mal registrada, problema con la conexión a la base.

Primer caso - **Filtra Receta:**

Caso	Filtra Receta
Autor	Alberto Rodríguez
Descripción	Panel donde filtrar recetas en función del nombre y seleccionarlás para su posterior visualización
Condición previa	Debe haber recetas introducidas, se debe buscar por nombre
Excepciones	No hay recetas, no se ha conectado a la base de datos, el nombre introducido no tiene coincidencias

## Diagrama de flujo, regiones, complejidad y caminos



### Regiones

- Región externa (1)
- Región entre 0-2 (2)
- Región entre 3-9 (3)
- Región entre 3-13 (4)
- Región entre 4-9 (5)
- Región entre 6-13 (6)

### CAMINOS:

1 -> 0 1/2 3 4 7/8 3

2 -> 0 1/2 3 9 [8] 3

3 -> 0 1/2 3 5 3

4 -> 0 1/2 3 5 10 3

5 -> 0 1/2 3 6 11+12 3

5 -> 0 1/2 3 6 11+13 3

### Complejidad:

17 aristas - 13 nodos + 2 = Complejidad 6

5 nodos predicados + 1 = Complejidad 6

## DIAGRAMA ENTIDAD RELACIÓN DE LA BBDD

RECETA
ID integer PRIMARY KEY AUTO_INCREMENT, imagenPrincipal text DEFAULT (NULL), nombre text DEFAULT (NULL), duracion text DEFAULT (NULL), dificultad text DEFAULT (NULL), descripcion text DEFAULT (NULL), tags text DEFAULT (NULL), ingredientes text DEFAULT (NULL), pasos text DEFAULT (NULL)

INGREDIENTE
ID integer PRIMARY KEY AUTO_INCREMENT, nombre text DEFAULT (NULL), precio text DEFAULT 0, origen text DEFAULT (NULL), estado_preferente text DEFAULT 0

USUARIO
id INTEGER PRIMARY KEY AUTO_INCREMENT, nombre TEXT DEFAULT (NULL), apellido1 TEXT DEFAULT (NULL), apellido2 TEXT DEFAULT (NULL), email TEXT DEFAULT (NULL), usuario TEXT DEFAULT (NULL), psw TEXT DEFAULT (NULL)

## CLASES DE LA APLICACIÓN

### Clase AddReceta:

Como podemos observar, tiene serialVersionUID como todos los paneles. Cuenta con diferentes atributos para poder gestionar colores, fuentes o modificaciones tanto dentro del constructor, como en los métodos de los listeners.

Esta clase es la más grande de toda la aplicación, pues todo cuenta con su propio listener.

Permite añadir una imagen principal desde un FileChooser, almacenando esta imagen entre sus archivos para poder acceder a ella siempre.

Cuando añades nombre, duración, dificultad y descripción se almacenan directamente sobre el objeto receta que luego irá a la base de datos.

La barra de búsqueda de los ingredientes, cuenta con un PopupMenu que se actualiza en tiempo real cada vez que se añade un carácter, para mostrar todos los ingredientes que existen en la base de datos conservando coherencia.

Muestra también la posibilidad de agregar pasos y etiquetas a voluntad.

Al final, tiene el botón “guardar” que coge toda la información recabada y la añade a la base de datos en uso.

La clase **addReceta**, es igual en forma de muestreo de la información, aunque no implementa los listeners ni permite editar los campos.

```
▼ C AddReceta
  serialVersionUID : long
  colorFuente : Color
  colorForAddFuente : Color
  colorFondoPrincipal : Color
  colorprincipalOscuro : Color
  generalFont : Font
  pasos : String[][]
  receta : Receta
  principal : JPanel
  etiquetas : JPanel
  jpTag : JPanel
  contieneOtroPaso : JPanel
  panelPasos : JPanel
  creadorIngredientes : JPanel
  backsave : JPanel
  imagenPrincipal : JLabel
  imagenAdd : JLabel
  tag : JLabel
  origenExterno : JLabel
  nombre : JTextArea
  tiempo : JTextArea
  dificultad : JTextArea
  descripcion : JTextArea
  escribeTag : JTextArea
  buscalngre : JTextField
  save : JButton
  back : JButton
  primeraCarga : Boolean
  muestrealngre : JPopupMenu
  nuevolangre : JMenuItem
  AddReceta()
  addImagen(GridBagConstraints) : void
  addNombre(GridBagConstraints) : void
  addTiempo(GridBagConstraints) : void
  addDificultad(GridBagConstraints) : void
  addDescripcion(GridBagConstraints) : void
  addEtiquetas(GridBagConstraints) : void
  addIngredientes(GridBagConstraints) : void
  addPasos(GridBagConstraints) : void
  addBackSaveBoton() : void
  instacialimagen(String, int) : JLabel
  keyPressed(KeyEvent) : void
  keyTyped(KeyEvent) : void
  focusLost(FocusEvent) : void
  focusGained(FocusEvent) : void
  mouseClicked(MouseEvent) : void
  mousePressed(MouseEvent) : void
  mouseReleased(MouseEvent) : void
  actionPerformed(ActionEvent) : void
  keyReleased(KeyEvent) : void
  mouseEntered(MouseEvent) : void
  mouseExited(MouseEvent) : void
```

## Clase Ingrediente:

En esta clase se encuentran 2 funciones como agregar y borrar cada ingrediente, además de que muestra una tabla que cada vez que se agregue o borre un ingrediente se “actualiza” para que el usuario vea los ingredientes que “en directo” en la base de datos.

Para cada ingrediente que se quiera agregar se deben rellenar los campos nombre, precio y origen, con la posibilidad de dejar vacío el campo de estado, pues hay algunos ingredientes que no tienen estado.

De la misma forma, para cada ingrediente que se quiera borrar se debe rellenar el campo del nombre del ingrediente.

En los en ambos botones AÑADIR o BORRAR, se inician métodos como comprobarIngrediente() que nos sirve para ambas acciones, tal es el caso que comprueba si ya existe un ingrediente e impide que se vuelva a repetir, algo parecido pasa con borrar un ingrediente, primero comprueba si existe para poder borrarlo.

AÑADIR INGREDIENTE, se usa aniadirIngrediente()

BORRAR INGREDIENTE, se usa borrarIngrediente()

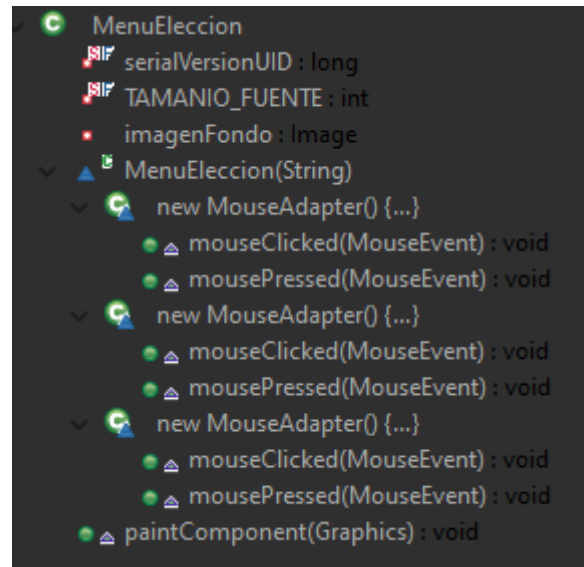
```
Ingrediente
  serialVersionUID : long
  imagenFondo : Image
  con : Connection
  pstmt : PreparedStatement
  rs : ResultSet
  jtfNombre : JTextField
  jtfPrecio : JTextField
  jtfOrigen : JTextField
  jtfEstado : JTextField
  nombre : JLabel
  nombreBorrar : JLabel
  jtfNombreBorrar : JTextField
  recogeNombre : String
  recogePrecio : String
  recogeOrigen : String
  recogeEstado : String
  mensajeError : int
  mensajeIngredienteCreador : int
  panelAbajo : JPanel
  Ingrediente(String)
    > new MouseAdapter() {...}
    > new MouseAdapter() {...}
    > new MouseAdapter() {...}
    comprobarIngrediente(String) : int
    aniadirIngrediente(String, String, String, String) : int
    borrarIngrediente(String) : int
    tabla() : JTable
    paintComponent(Graphics) : void
```



## Clase MenuEleccion:

Integramos el serialVersionUID al ser un JPanel es serializable.

En esta clase, podemos observar las 3 opciones que nos ofrece nuestro programa, cada botón con sus listeners necesarios.



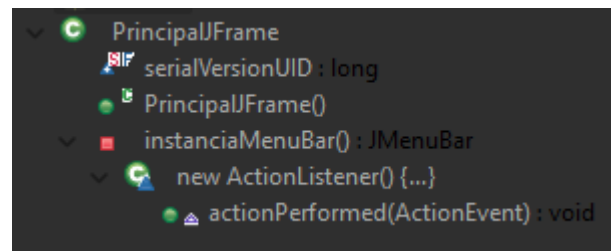
```
MenuEleccion
  serialVersionUID : long
  TAMANIO_FUENTE : int
  imagenFondo : Image
  MenuEleccion(String)
    new MouseAdapter() {...}
      mouseClicked(MouseEvent) : void
      mousePressed(MouseEvent) : void
    new MouseAdapter() {...}
      mouseClicked(MouseEvent) : void
      mousePressed(MouseEvent) : void
    new MouseAdapter() {...}
      mouseClicked(MouseEvent) : void
      mousePressed(MouseEvent) : void
  paintComponent(Graphics) : void
```

## Clase PrincipalJFrame:

Integramos el serialVersionUID al ser un JPanel es serializable.

Esta es una clase muy sencilla. Es el JFrame que contendrá todos los paneles creados

A este se le añadirá un JScrollPane que será el responsable de ir mostrando los diferentes paneles por los que nos vamos moviendo.

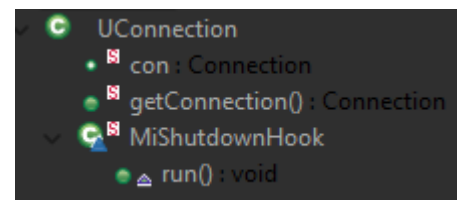


```
PrincipalJFrame
  serialVersionUID : long
  PrincipalJFrame()
  instanciaMenuBar() : JMenuBar
  new ActionListener() {...}
    actionPerformed(ActionEvent) : void
```

## Clase UConnction:

Esta es la clase vista para ejecutar de forma cómoda la conexión con la base de datos.

Solo presenta un cambio, debido a que el archivo jar daba problemas para encontrar el jdbc.properties, hemos optado por no incluirlo, y añadir directamente la información de la conexión



```
UConnection
  con : Connection
  getConnection() : Connection
  MiShutdownHook
    run() : void
```

## Clase Receta:

Integramos el serialVersionUID al ser un JPanel es serializable.

Esta clase es plantilla para los objetos “Receta” que se crean, almacenan y recogen de la base de datos.

Contiene todos los parámetros de una receta con su tipo, sus getters y setters.

Consta de diferentes constructores en función de uso.

El constructor vacío es el que se utiliza para ir rellenando cuando se recoge la información del método addReceta,

el constructor sin ID, se utiliza cuando queremos mover o mostrar las recetas y trabajar con ellas sin depender de la base de datos.

Por último, el constructor con la ID es el que se usa para recogerlas cuando se buscan de la base de datos.

## Clase Recetario:

Integramos el serialVersionUID al ser un JPanel es serializable.

Esta es la clase principal de la aplicación.

En ella, creamos todos los posibles paneles principales para dejarlos cargados y colocarlos cuando sea necesario.

En su main, se instala el JFrame, y en este, un JScrollPane que es el encargado de contener los paneles según nos vayamos moviendo entre ellos.

Para cambiar los paneles, se hace uso de los setters creados. Por ejemplo, cuando se pulsa en “Iniciar Sesión”, si la información es correcta, se utilizará setPCentralMenuEleccion() para pasar al menú principal de la aplicación.

```
Receta
  ID : int
  imagenPrincipal : String
  nombre : String
  duracion : String
  dificultad : String
  descripcion : String
  tags : String
  ingredientes : String
  pasos : String
  Receta()
  Receta(String, String, String, String, String, String, String, String, String)
  Receta(int, String, String, String, String, String, String, String, String, String)
  getID() : int
  getImagenPrincipal() : String
  setImagenPrincipal(String) : void
  getNombre() : String
  setNombre(String) : void
  getDuracion() : String
  setDuracion(String) : void
  getDificultad() : String
  setDificultad(String) : void
  getDescripcion() : String
  setDescripcion(String) : void
  getTags() : String
  setTags(String) : void
  getIngredientes() : String
  setIngredientes(String) : void
  getPasos() : String
  setPasos(String) : void
```

```
Recetario
  ventana : PrincipalUFrame
  pIngrediente : Ingrediente
  pMenuEleccion : MenuEleccion
  pUsuarioLogin : UsuarioLogin
  pUsuarioRegistro : UsuarioRegistro
  pSeleccionaReceta : SeleccionaRecetas
  nuevaReceta : AddReceta
  jSPvisor : JScrollPane
  elegida : Receta
  verReceta : VisorReceta
  main(String[]) : void
  setPCentralIngrediente() : void
  setPCentralMenuEleccion() : void
  setPCentralUsuarioLogin() : void
  setPCentralUsuarioRegistro() : void
  setPCentralSeleccionaReceta() : void
  setPCentralNuevaReceta() : void
  setPCentralVerReceta(Receta) : void
```

## Clase SeleccionaRecetas:

Integramos el serialVersionUID al ser un JPanel es serializable.

Esta clase es la encargada de recoger las recetas coincidentes de la base de datos, mostrarlas, y permitirnos acceder a ellas.

Consta de un panel de búsqueda, el cual busca en la base de datos lo que se haya escrito, y devuelve todas las recetas que coincidan en nombre.

Estas se muestran y permiten ser clicadas para acceder a ellas.

Tiene diferentes métodos para ahorrar líneas de código, pues sino, sería necesario hacer muchas de sus líneas de forma repetida.

```
SeleccionaRecetas
  serialVersionUID : long
  colorFuente : Color
  colorFondoPrincipal : Color
  colorForAddFuente : Color
  colorprincipalOscuro : Color
  generalFont : Font
  busqueda : JTextField
  principal : JPanel
  norte : JPanel
  todasRecetas : ArrayList<Receta>
  SeleccionaRecetas()
  new ActionListener() {...}
    actionPerformed(ActionEvent) : void
  addRecetasCoincidentes(String) : void
  new MouseAdapter() {...}
    mouseClicked(MouseEvent) : void
  resultadoRecetas(String) : ArrayList<Receta>
  instacialmagen(String, int) : JLabel
  recargaRecetas() : void
  focusLost(FocusEvent) : void
  keyPressed(KeyEvent) : void
  focusGained(FocusEvent) : void
  keyTyped(KeyEvent) : void
  keyReleased(KeyEvent) : void
```

## Clase UsuarioLogin:

Integramos el serialVersionUID al ser un JPanel es serializable.

La función principal de esta clase es permitir al cliente acceder a su cuenta de recetario, teniendo que introducir su nombre de usuario y contraseña.

Dicho usuario y contraseña serán evalúan con el método comprobarUsuario()

Integramos el serialVersionUID al ser un JPanel es serializable

Después a cada botón se le establece los Listeners necesarios para la interacción con el resto de la aplicación recetario.

```
UsuarioLogin
  serialVersionUID : long
  bLogin : JButton
  bRegistrar : JButton
  jtfUsuario : JTextField
  jtfContrasenia : JPasswordField
  imagenFondo : Image
  recogeUsuario : String
  recogeContrasenia : String
  recogeCaracteresCon : char[]
  con : Connection
  pstmt : PreparedStatement
  rs : ResultSet
  mensajeErrorCuadrosVacios : int
  mensajeErrorLogin : int
  UsuarioLogin(String)
  new MouseAdapter() {...}
  new MouseAdapter() {...}
  comprobarUsuarioContrasenia(String, String) : int
  paintComponent(Graphics) : void
```

## Clase UsuarioRegistro:

Integramos el serialVersionUID al ser un JPanel es serializable.

La función principal de esta clase es registrar a un nuevo cliente, para ello nos apoyamos en el método registrarUsuario(), pero no sin antes comprobar si el usuario que se va a crear ya se encuentra en nuestra base de datos, en ese caso se deberá introducir otro nombre de usuario.

Con la ayuda de nuestros Listeners necesarios que le asignamos a nuestros respectivos botones, siendo

USUARIO = vuelves a la ventana de iniciar sesión

REGISTRARSE = intenta el registro del nuevo cliente, intenta porque antes realiza comprobaciones sobre si faltan campos por rellenar o como se ha leído antes, si el usuario existe, si todo está en orden, se registra al nuevo cliente y muestra la ventana de iniciar sesión para que introduzca el usuario y contraseña creadas recientemente.

```
UsuarioRegistro
  serialVersionUID : long
  jtfNombre : JTextField
  jtfApellido1 : JTextField
  jtfApellido2 : JTextField
  jtfEmail : JTextField
  jtfUsuario : JTextField
  jtfContraseña : JPasswordField
  recogeNombre : String
  recogeApellido1 : String
  recogeApellido2 : String
  recogeEmail : String
  recogeUsuario : String
  recogeContraseña : String
  recogeCaracteresCon : char[]
  mensajeError : int
  mensajeUsuarioCreado : int
  mensajeUsuarioNoCreado : int
  imagenFondo : Image
  con : Connection
  pstmt : PreparedStatement
  pstmt2 : PreparedStatement
  rs : ResultSet
  UsuarioRegistro(String)
  new MouseAdapter() {...}
  mouseClicked(MouseEvent) : void
  new MouseAdapter() {...}
  mouseClicked(MouseEvent) : void
  paintComponent(Graphics) : void
  registrarUsuario(String, String, String, String, String, String) : int
  comprobarUsuario(String) : int
```

Esperamos que utilicéis con mucho gusto la aplicación, pues nos ha llevado mucho trabajo. Sabemos que no es una aplicación perfecta y probablemente sigamos agregándole cambios, con esto, concluimos el manual.

**Escrito y desarrollado íntegramente por:**

Alberto Rodríguez Pérez

Renzo Sandoval Villanueva