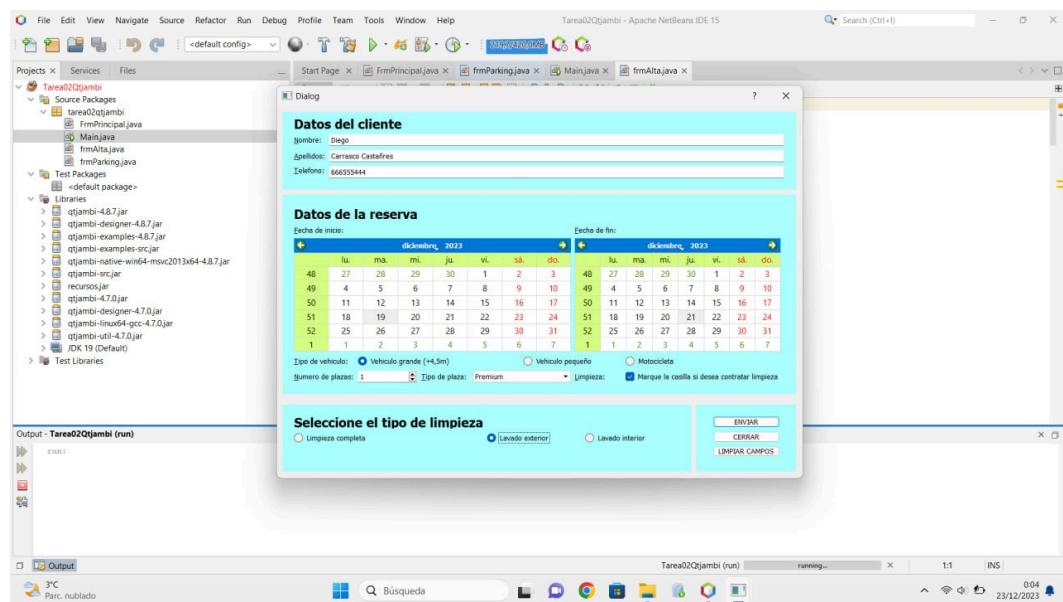
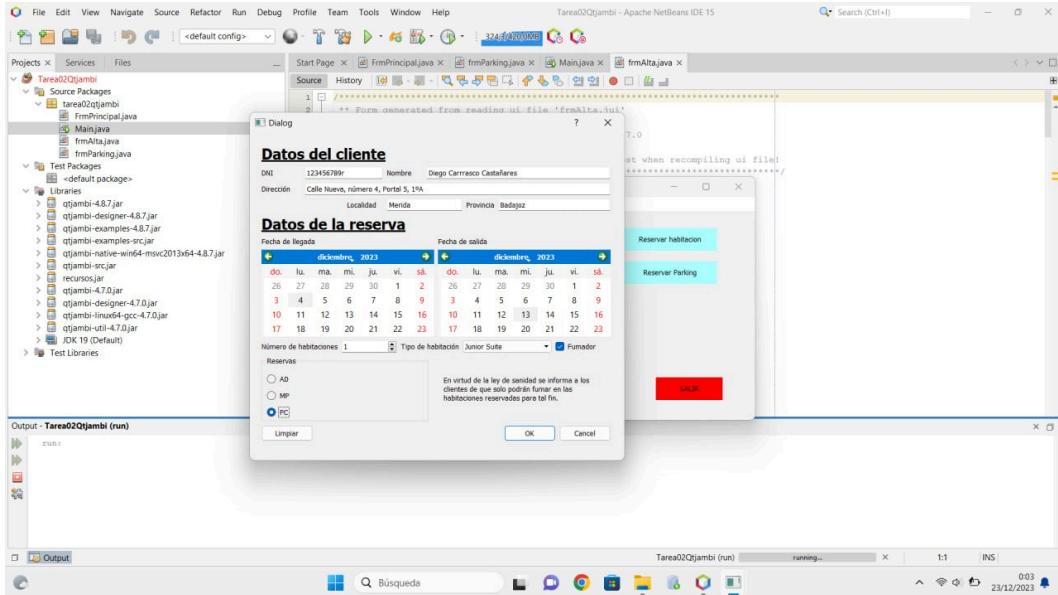


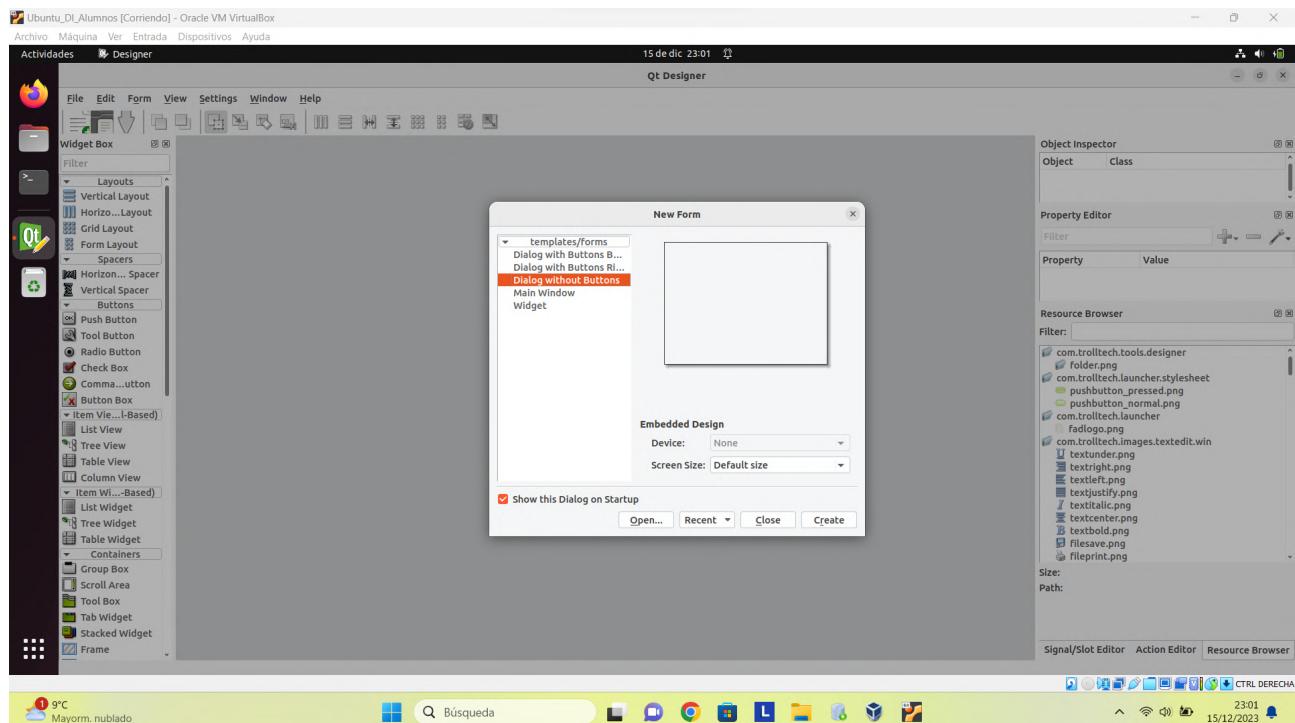
Tarea 2 para Desarrollo de Interfaces



Diego Manuel Carrasco Castañares

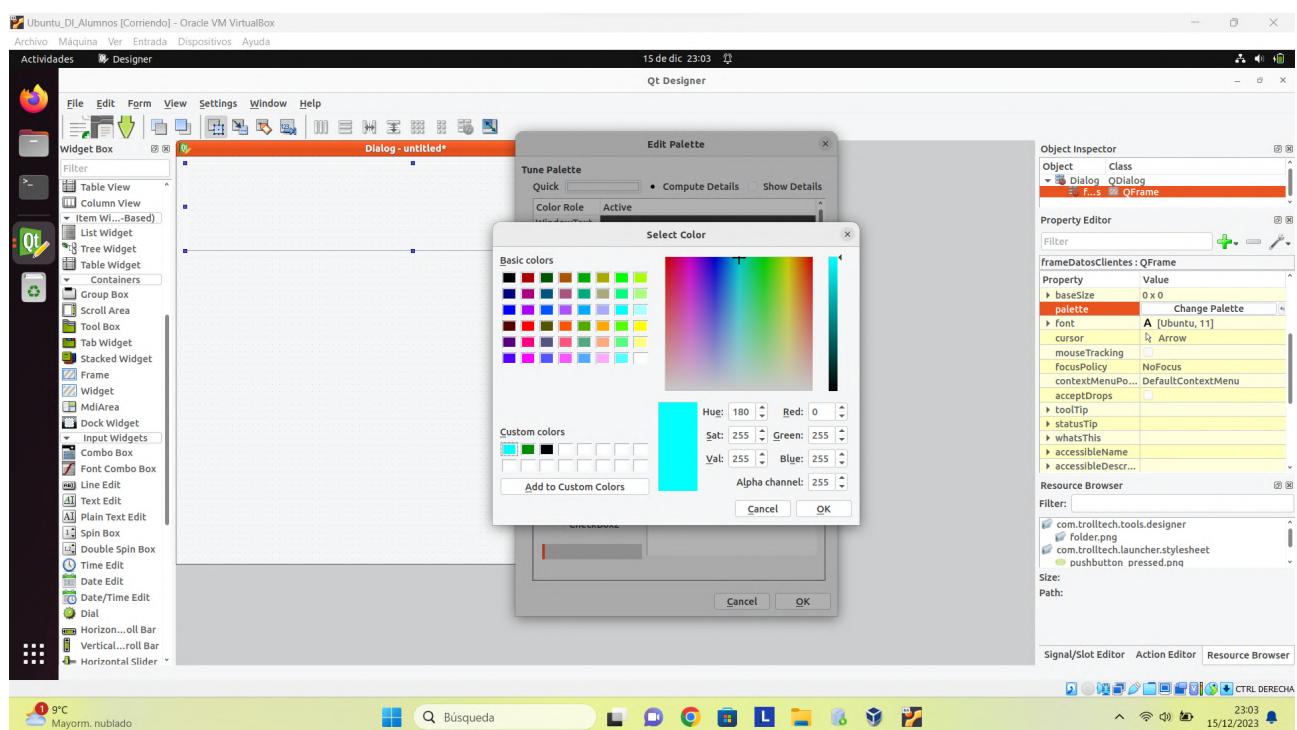
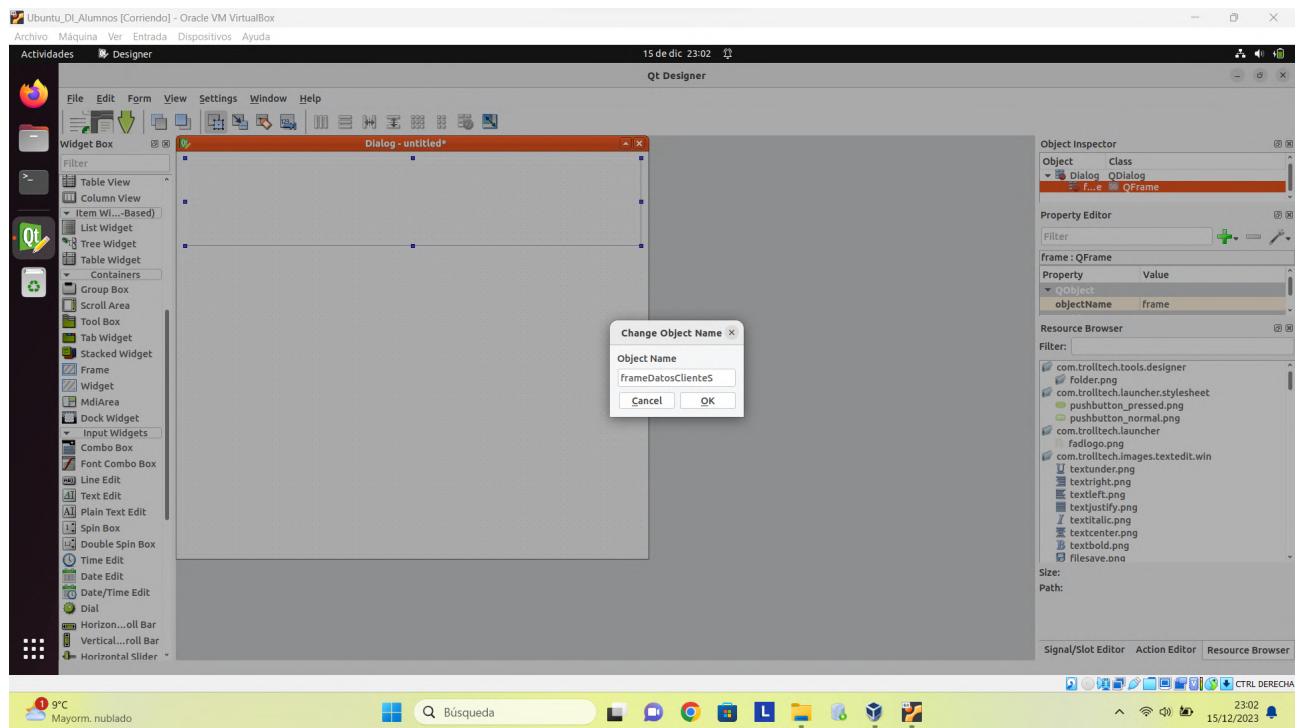
En mi caso, y tras varios intentos frustrados de intentar instalar qtjambi en Mac y no conseguir que funcione en windows 11, voy a realizar la tarea desde la maquina virtual subida a la plataforma. Decir que también me ha fallado en numerosas ocasiones, cerrándose de forma inesperada y no dejándome abrir de nuevo el fichero .jui para poder terminarlo, entre otros problemas.

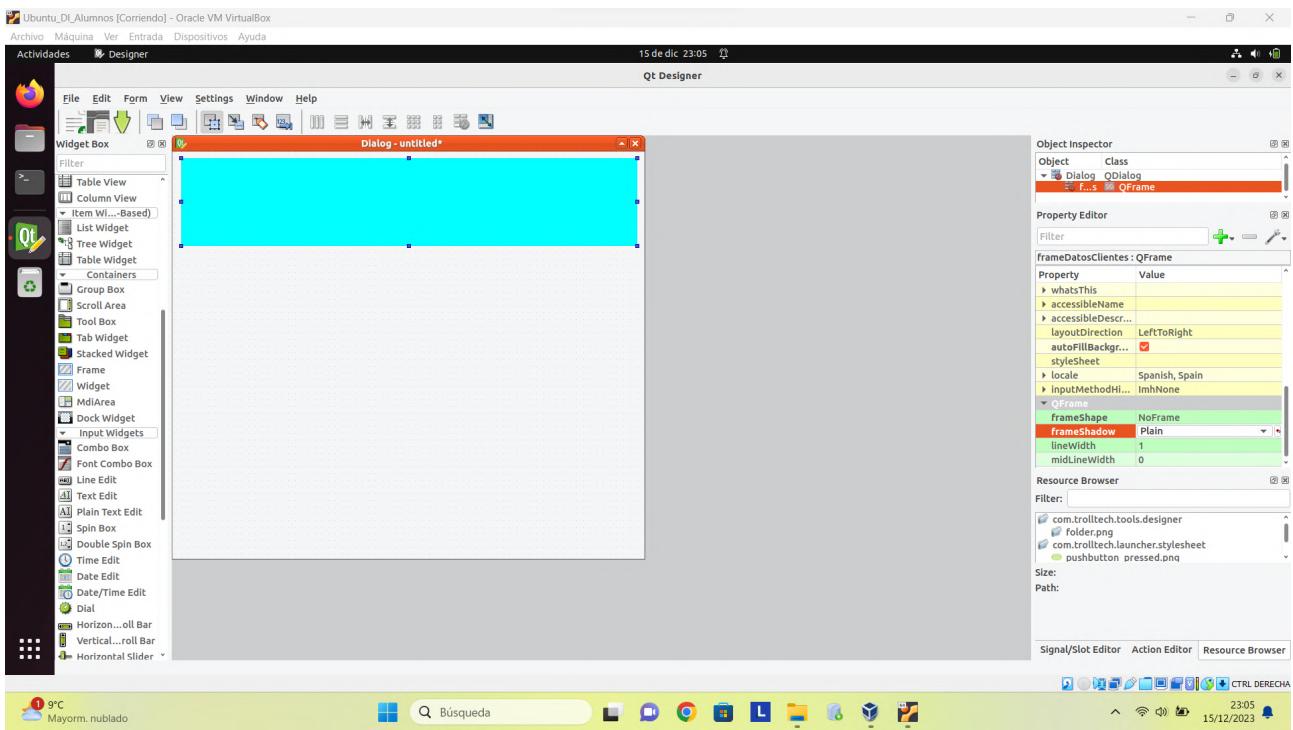
En primer lugar abrimos qtjambi, seleccionamos un dialogo sin botones y le damos a continuar.



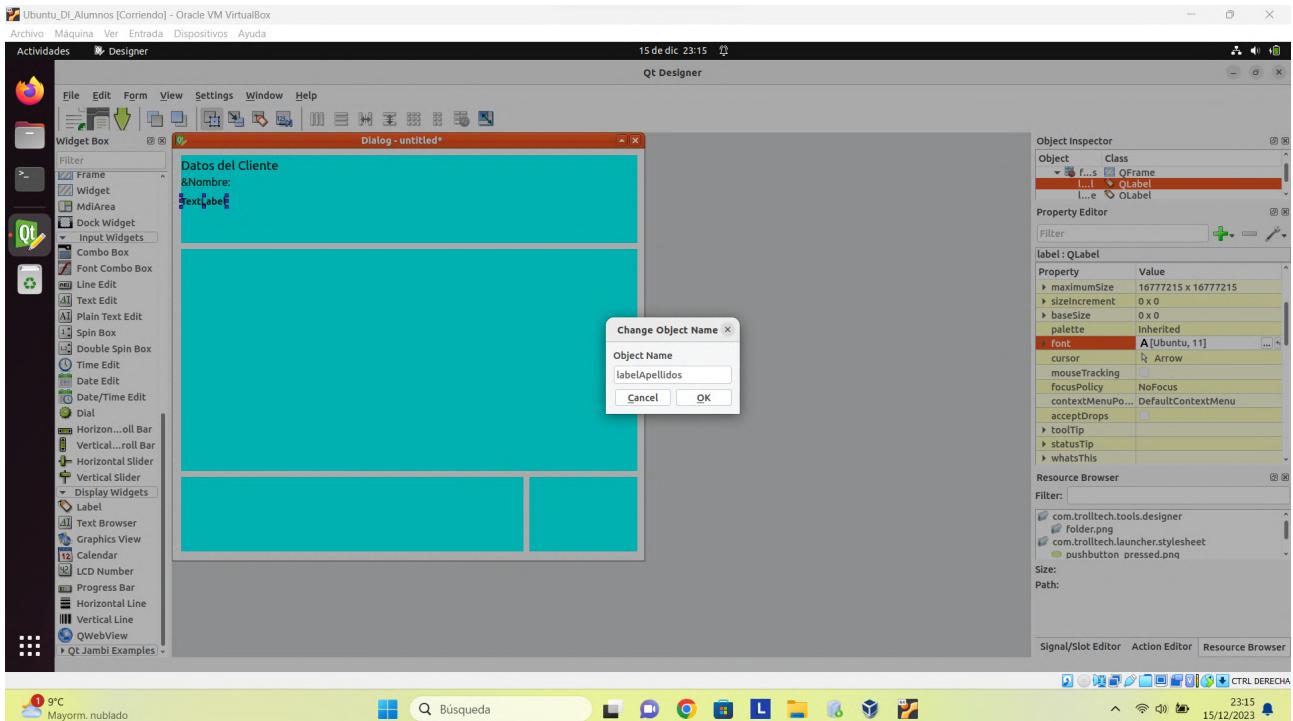
En la nueva ventana que nos abre vamos a crear cuatro Frame's diferentes, uno para los datos del cliente, otro para los datos de la reserva, otro que se activará si el cliente selecciona limpieza para elegir el tipo de limpieza y el último para colocar en el los botones de enviar y cerrar y renombrados los frames para acceder a ellos de forma mas sencilla.

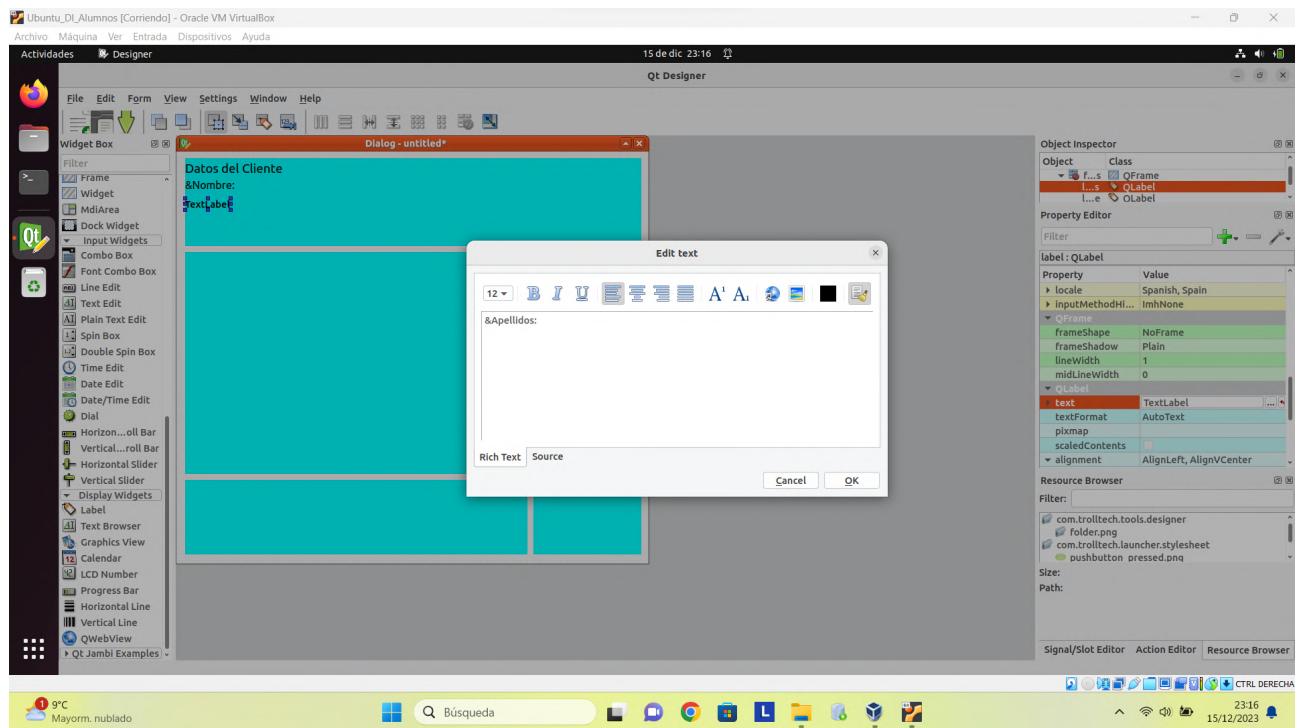
Posteriormente le daremos color a los frames, desde pallete para que sean mas vistosos de cara al usuario y seleccionamos autoFillBackground para aplicar el color al fondo.



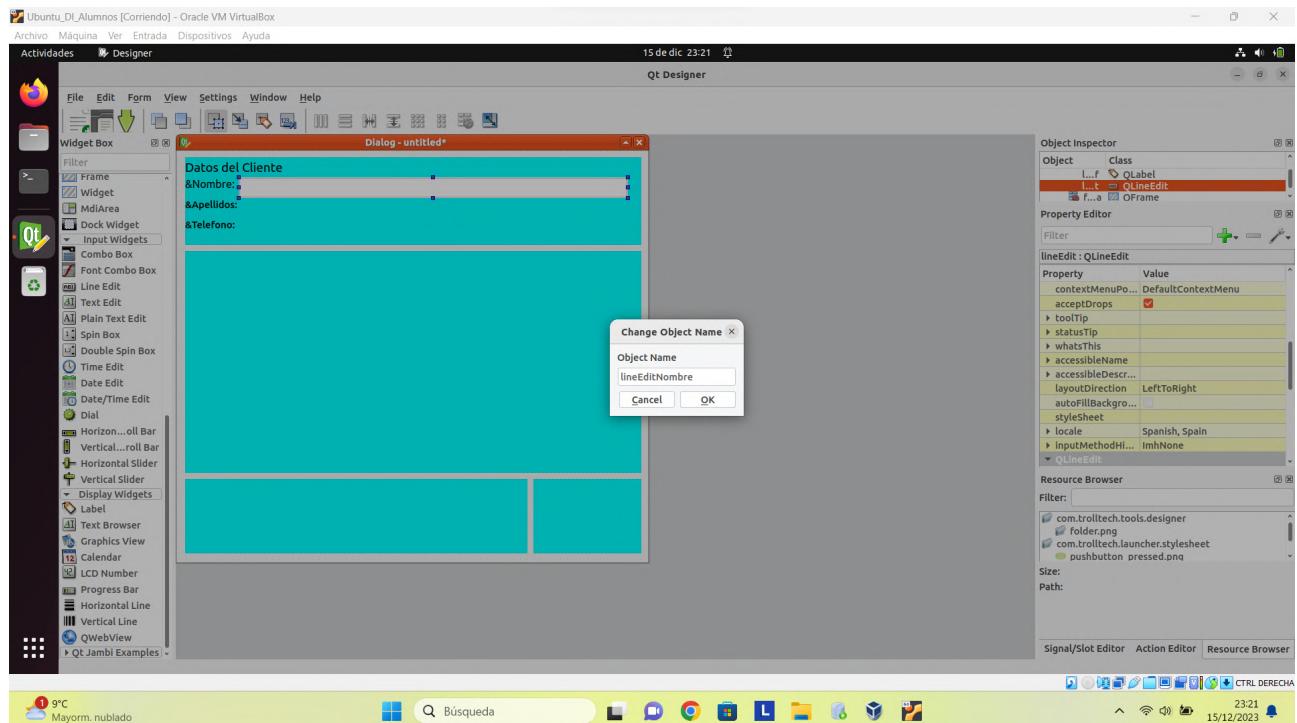


Posteriormente añadiremos las etiquetas datos del cliente, que será con una letra mayor que las demás y en negrita para darle nombre al conjunto de datos, nombre, apellidos y texto, para nombrar a cada campo de los que tiene que llenar el cliente, las nombraremos y le añadiremos texto anteponiendo el símbolo andpersand para acceder a ellas.

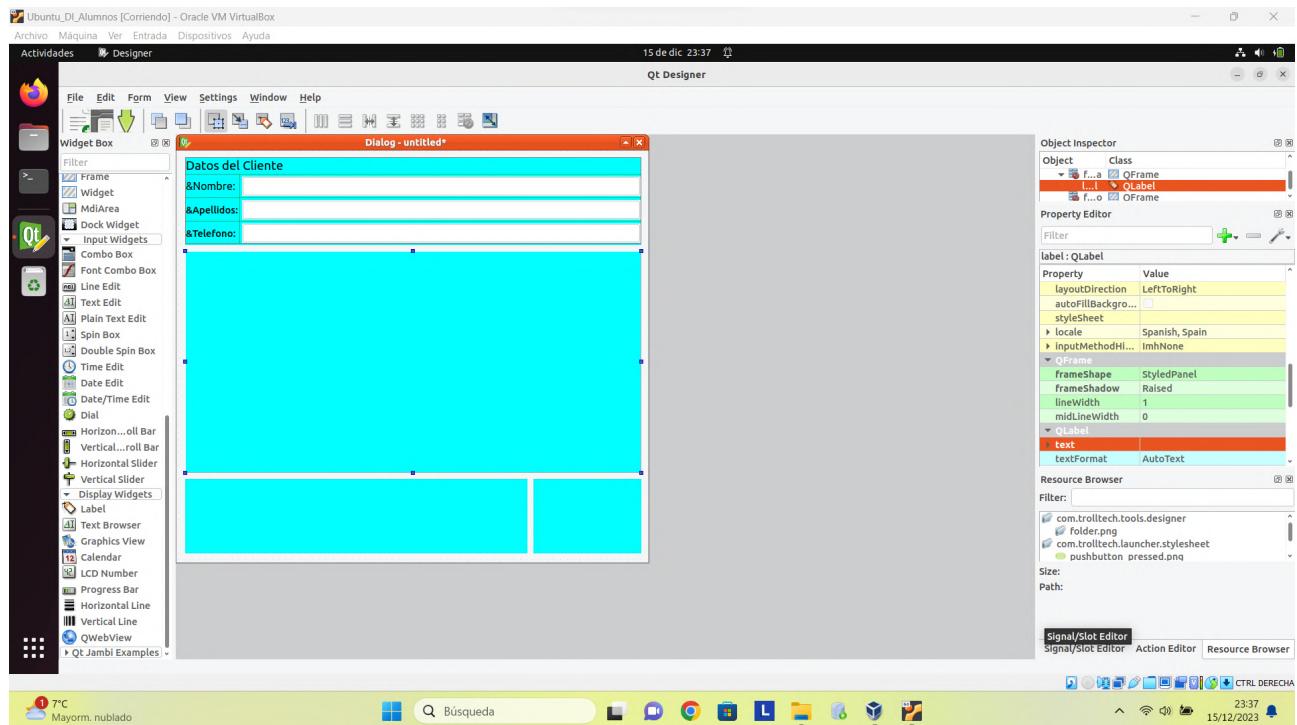




Tras ellos añadiremos un line edit al lado de las etiquetas nombre, apellidos y teléfono para recoger la información correspondiente a cada una de las etiquetas anteriores.



A continuación crearemos el frame "datos de la reserva".



En este frame añadiremos una etiqueta con el texto "datos de la reserva" en un texto de mayor tamaño que las demás etiquetas, añadiremos las etiquetas fecha de inicio, fecha de fin, tipo de vehículo, número de plazas, tipo de plaza y limpieza , le daremos un nombre, pondremos el texto que queremos que muestre cada etiqueta y le antepondremos el símbolo de andpersand.

Lo siguiente será añadir un campo de tipo calendar a la fecha de inicio y a la fecha de fin, a los cuales le pondremos que la fecha de inicio sea 01/01/2023.

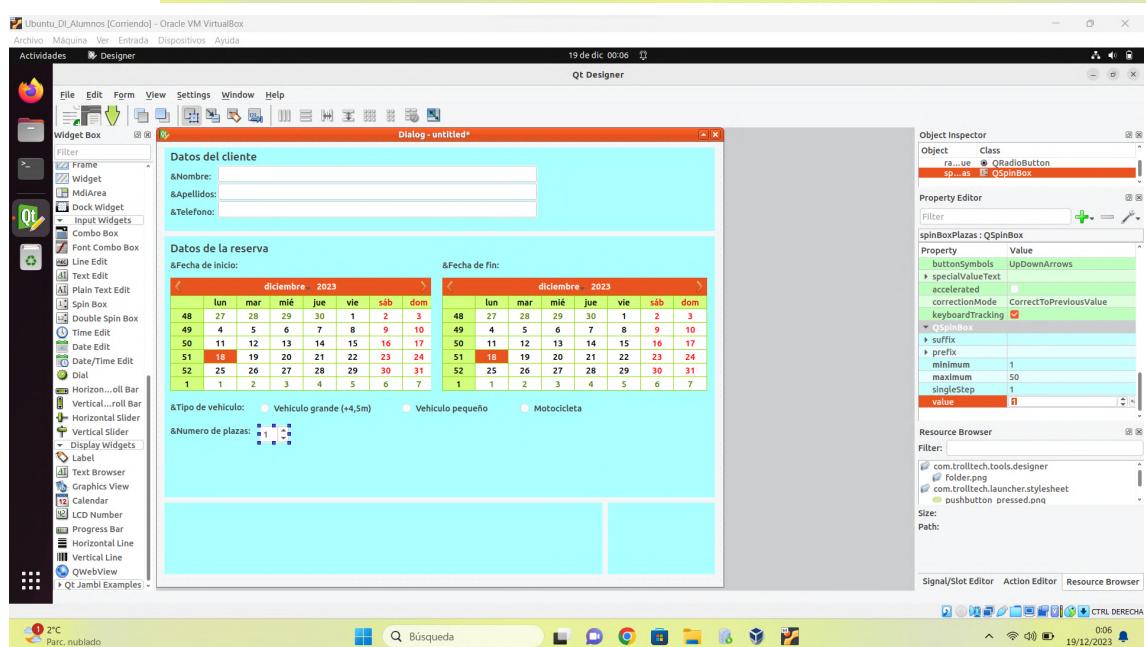
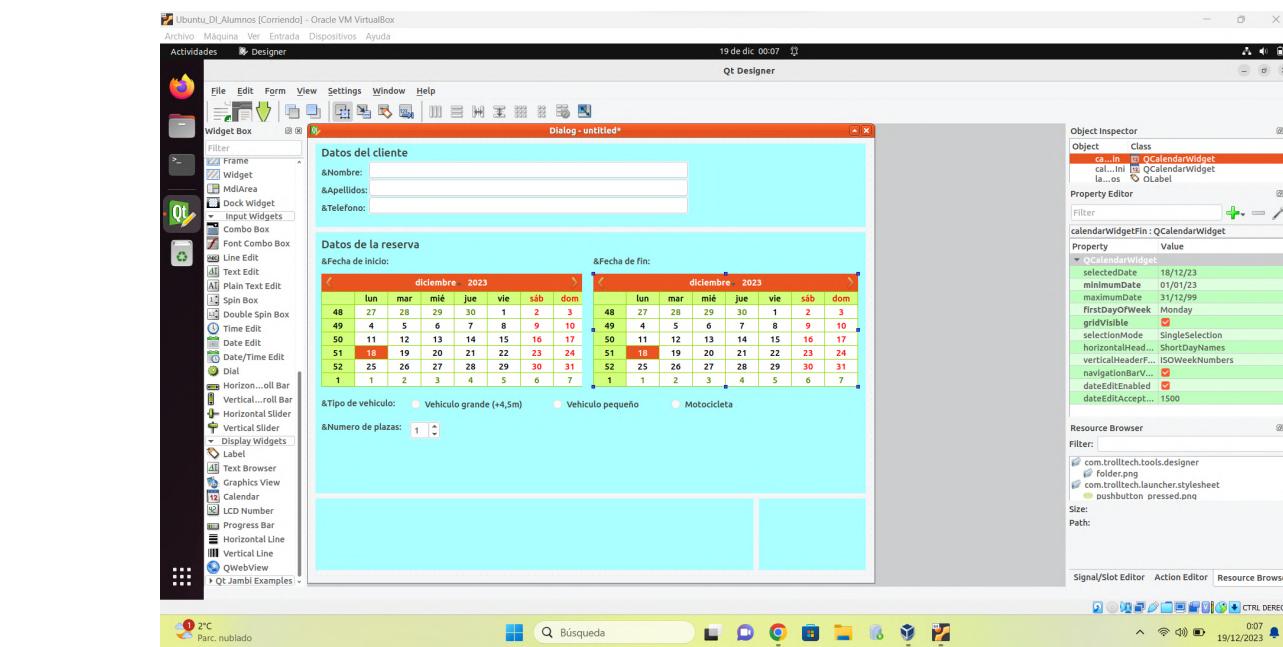
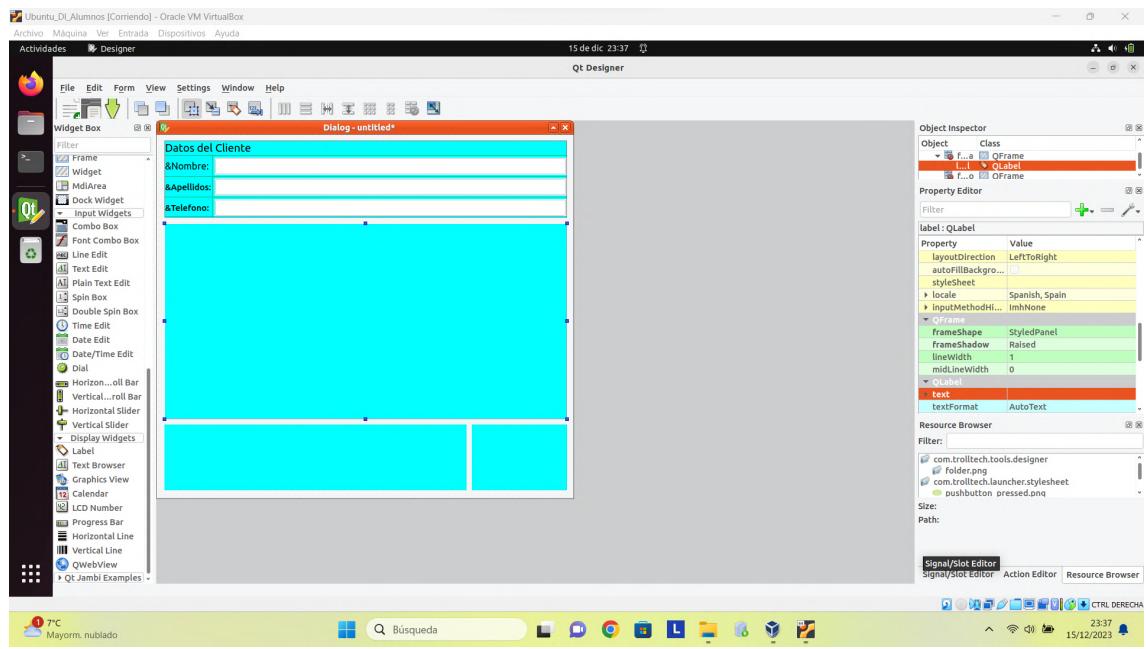
A la etiqueta de tipo de vehículo le añadiremos tres radio button, uno con el texto vehículo grande (+4,5m), otro con el texto vehículo pequeño y otro con el texto motocicleta.

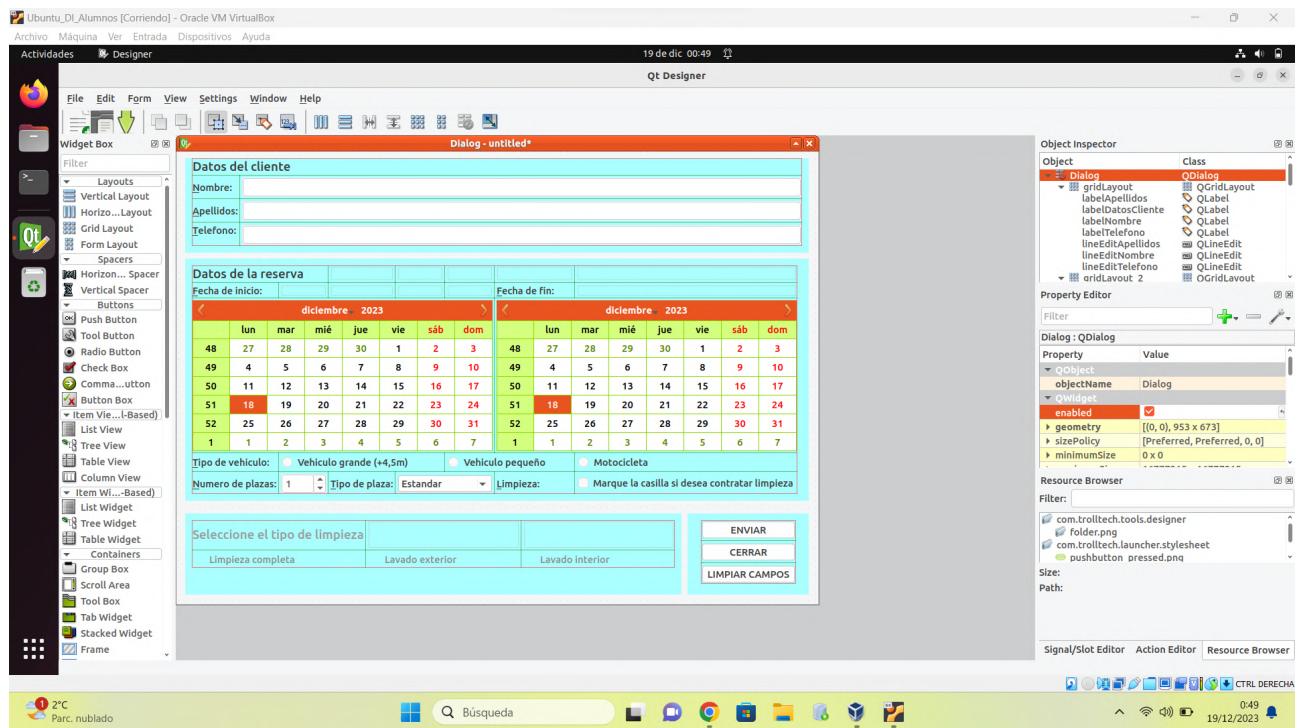
A la etiqueta número de plazas le añadiremos un spin box, el cual por defecto, empezara en 1 y mostrara como máximo 50.

Para tipo de plaza vamos a elegir un combo box, el cual tendrá tres campos, estándar, premium y deluxe y por defecto mostrara la opción estándar.

En el caso de la etiqueta limpieza le asociaremos un check box con el texto "marque esta casilla si desea adquirir el servicio de limpieza para su vehículo".

En las siguientes imágenes podemos ver lo descrito anteriormente.

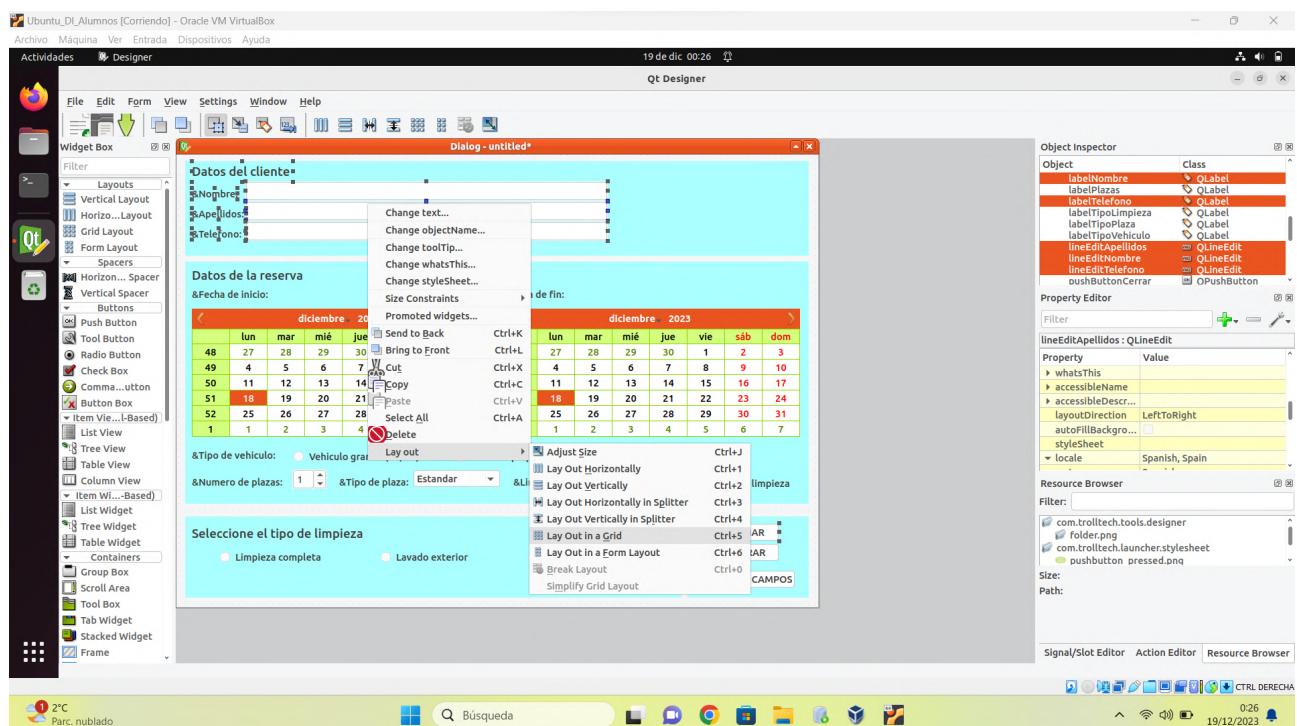


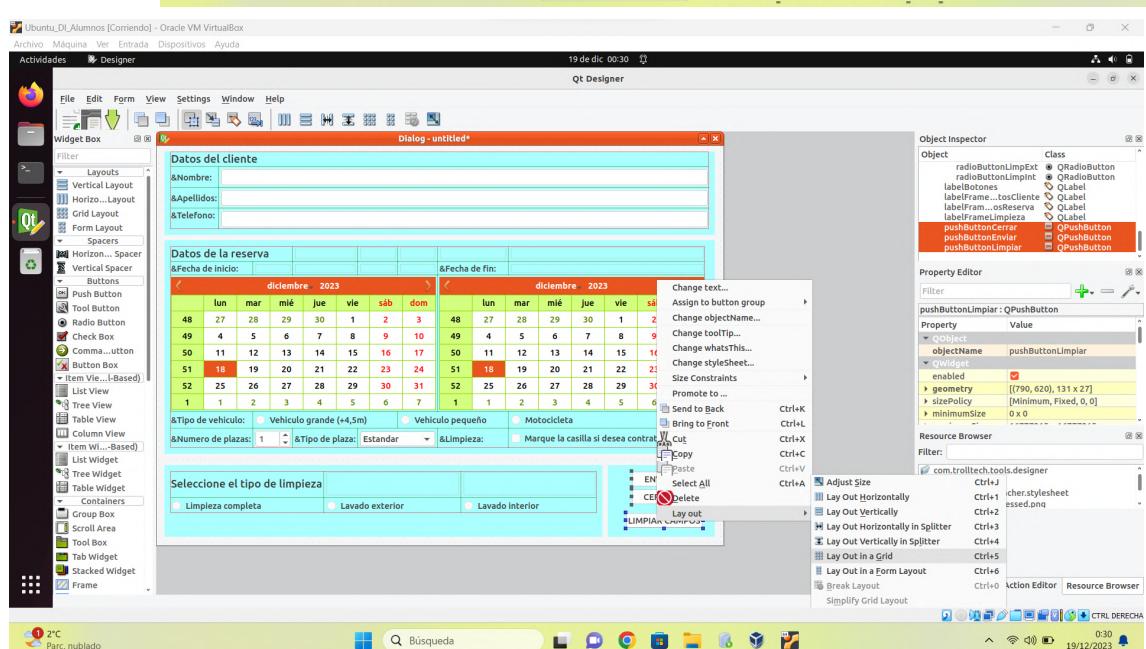
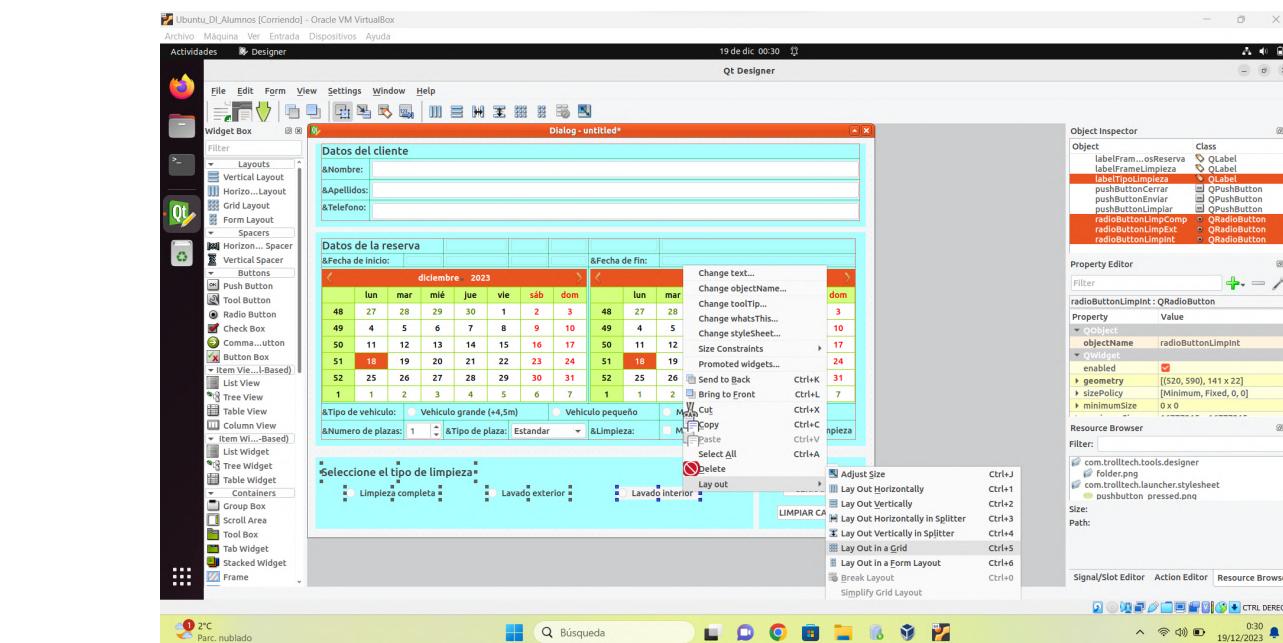
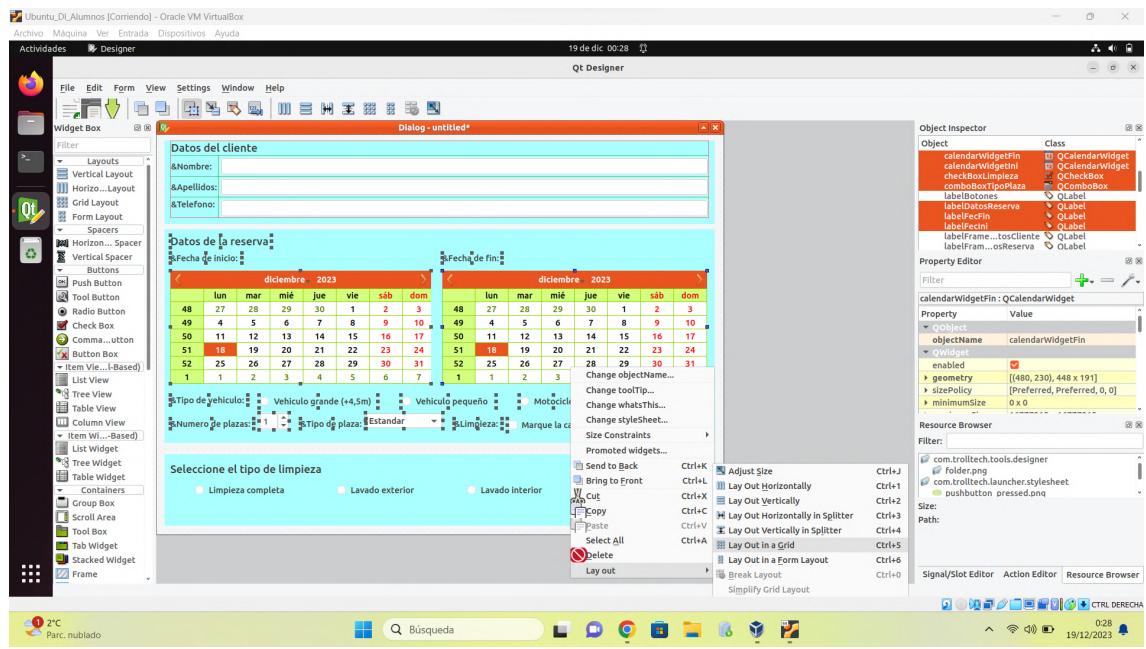


Como podemos observar en el frame anterior hemos añadido las etiquetas tipo de limpieza en un tipo de letra mayor y tres radio button los cuales contendrán los textos limpieza completa, limpieza interior y limpieza exterior.

En el último frame añadiremos los botones de tipo push button que contendrán los textos de enviar y de cerrar respectivamente.

Posteriormente aplicaremos los lay out a todos los frames. El tipo de lay out que hemos elegido es "in a grid" para todos ellos y lo ajustaremos al tamaño del frame.

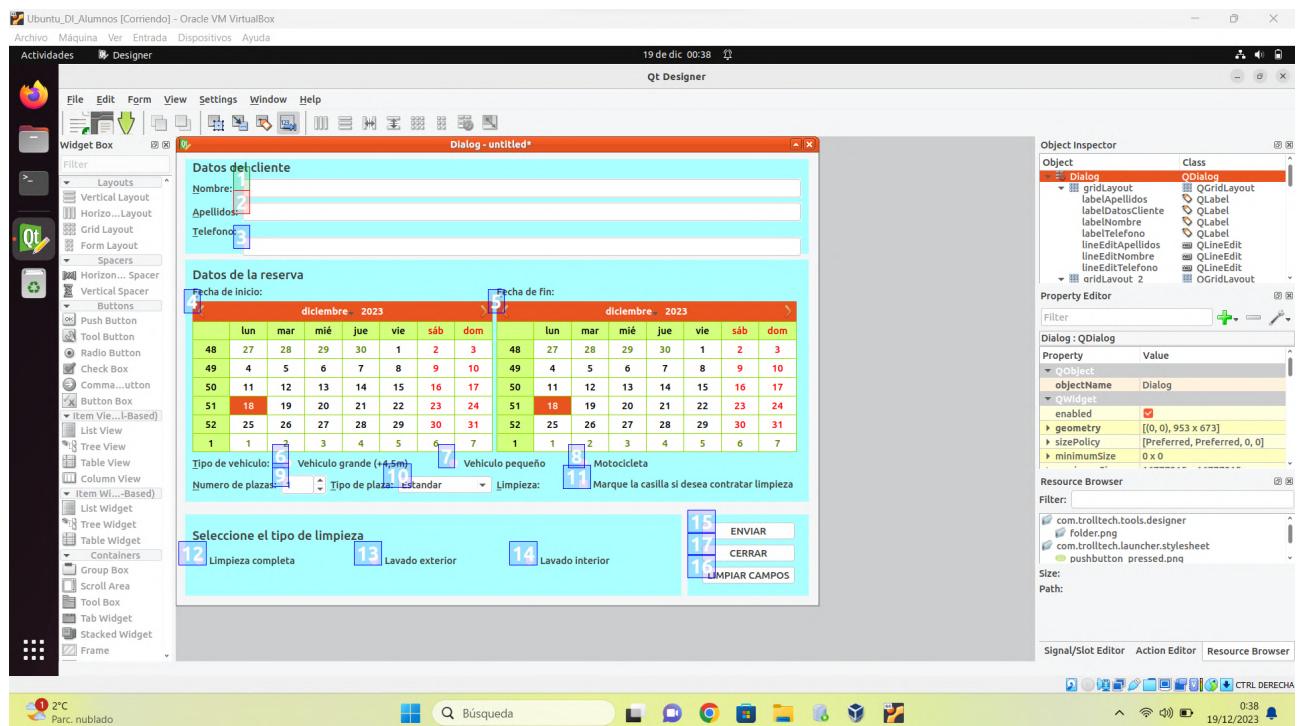
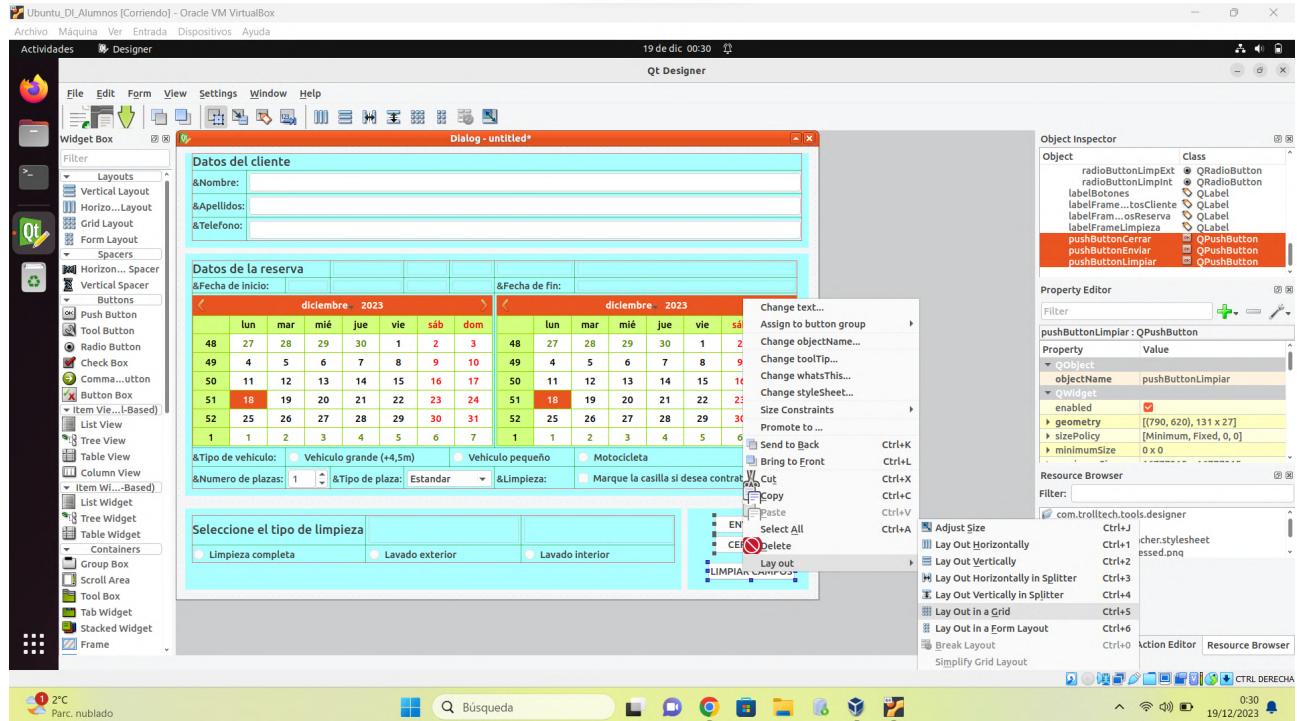




El siguiente paso será aplicar los bucles a todos los frames asociando cada etiqueta a su campo.

Posteriormente seleccionaremos el orden de tabulación para que vaya saltando de campo en campo en función del orden que le hayamos dado cuando el usuario pulse la tecla de tab.

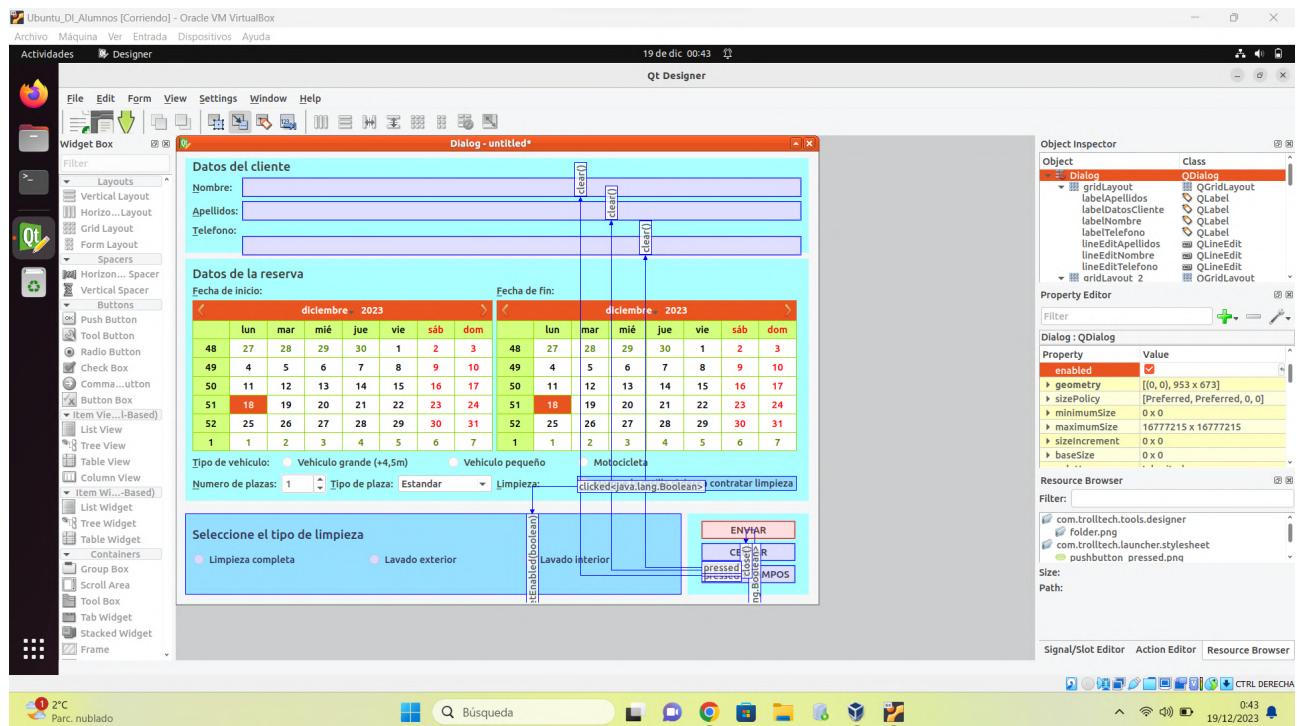
Podemos verlo en la siguiente imagen



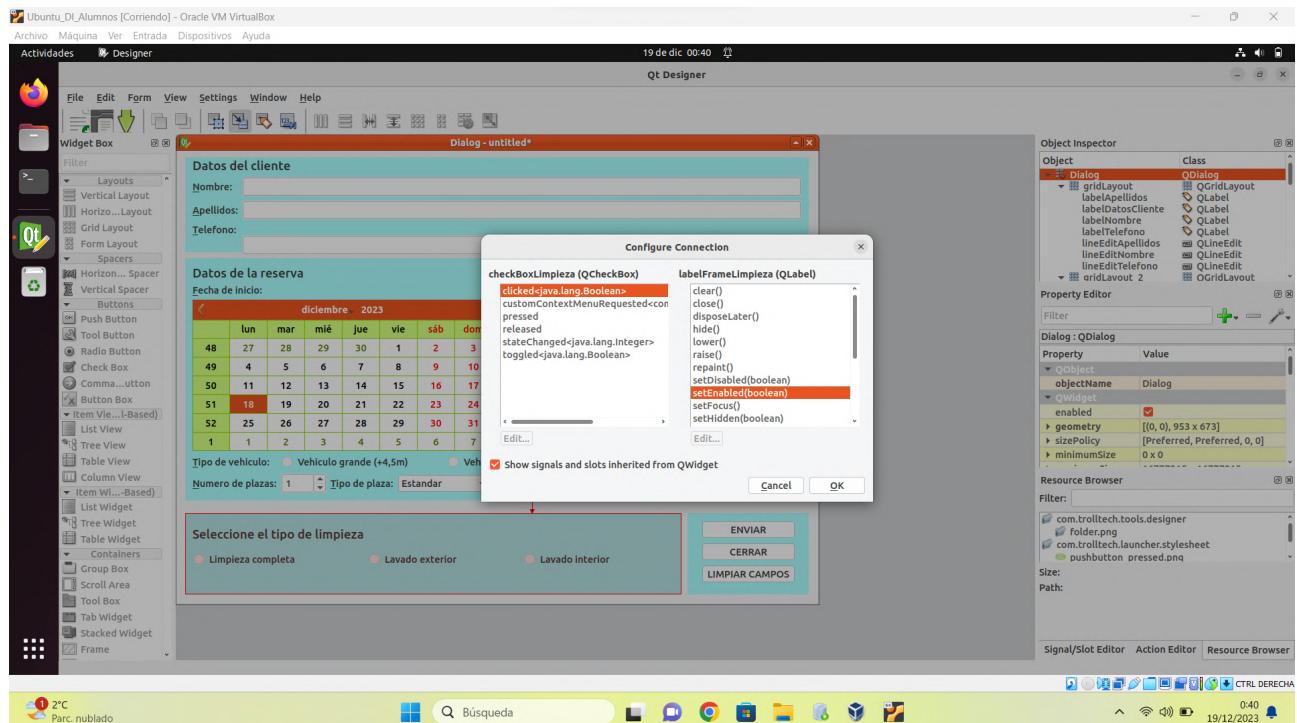
A continuación conectaremos el botón de limpiar mediante los signal slot con los campos editable del frame datos del cliente.

Diego Manuel Carrasco Castañares

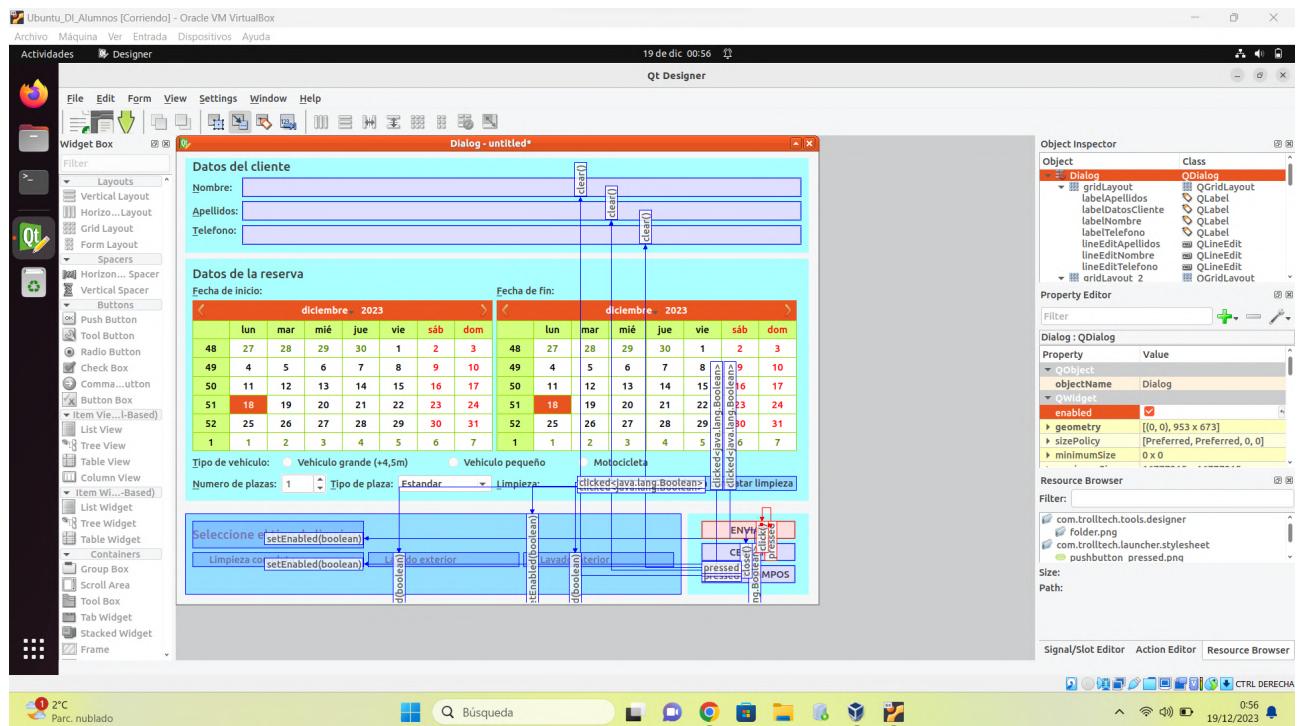
Página 10 de 28



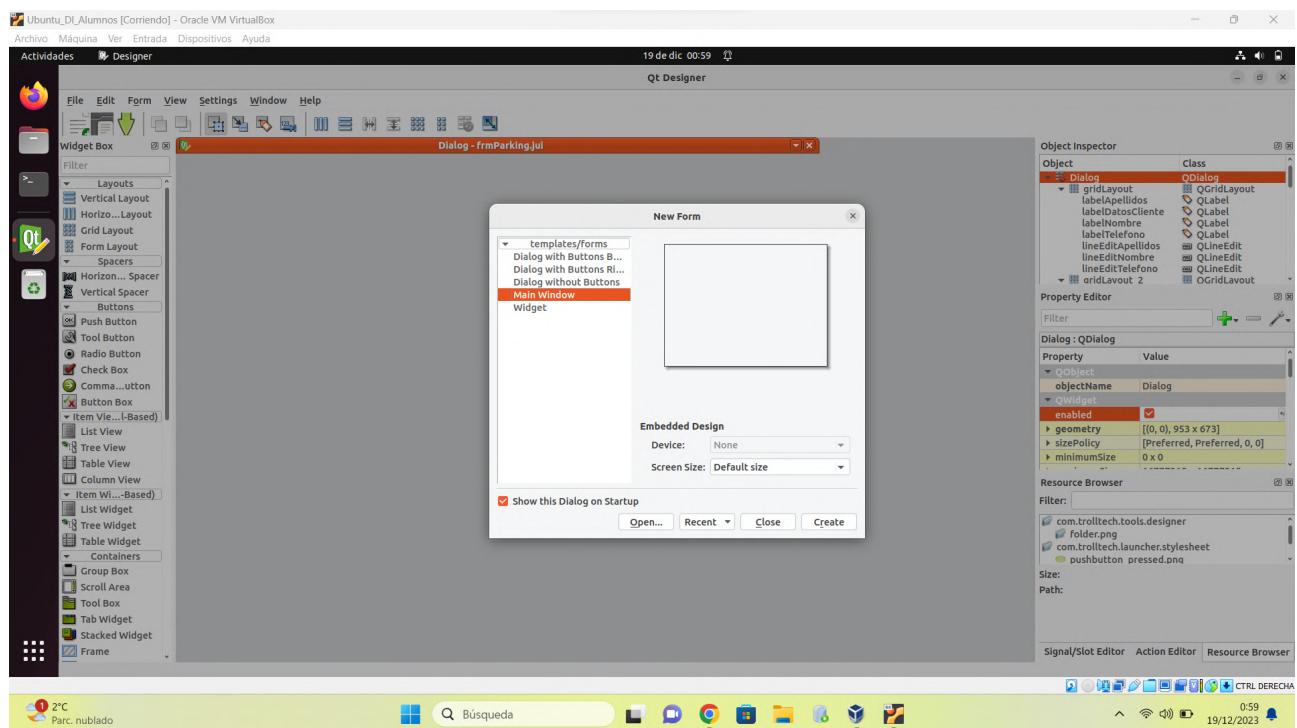
A continuación conectamos, mediante los Signal slot, el check box de limpieza con los campos del frame de limpieza, para que este habilitado o no en función de si esta seleccionado o no el check box mediante el método setEnabled de la clase clicked de java.



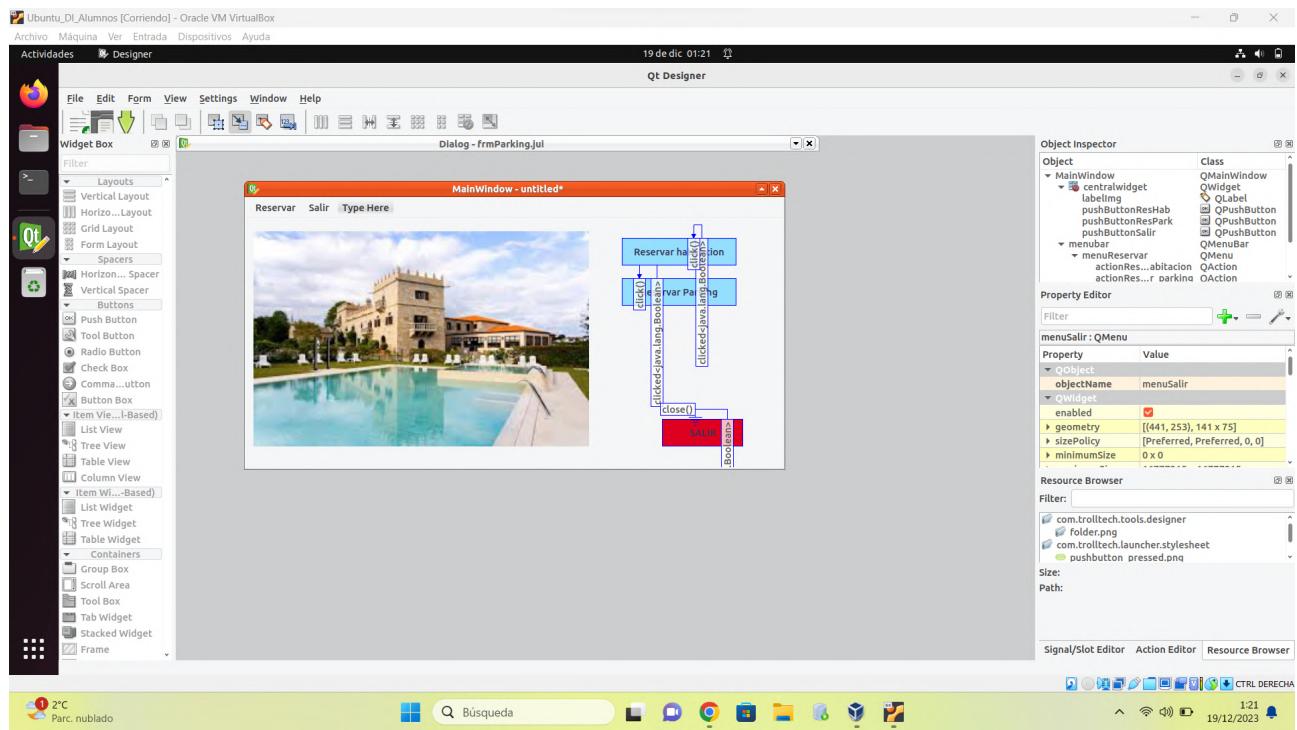
En la siguiente imagen podemos ver todas las conexiones realizadas con los Signal slot.



Como en mi caso no conseguía abrir el archivo .jui de la interfaz principal he creado una interfaz principal nueva.

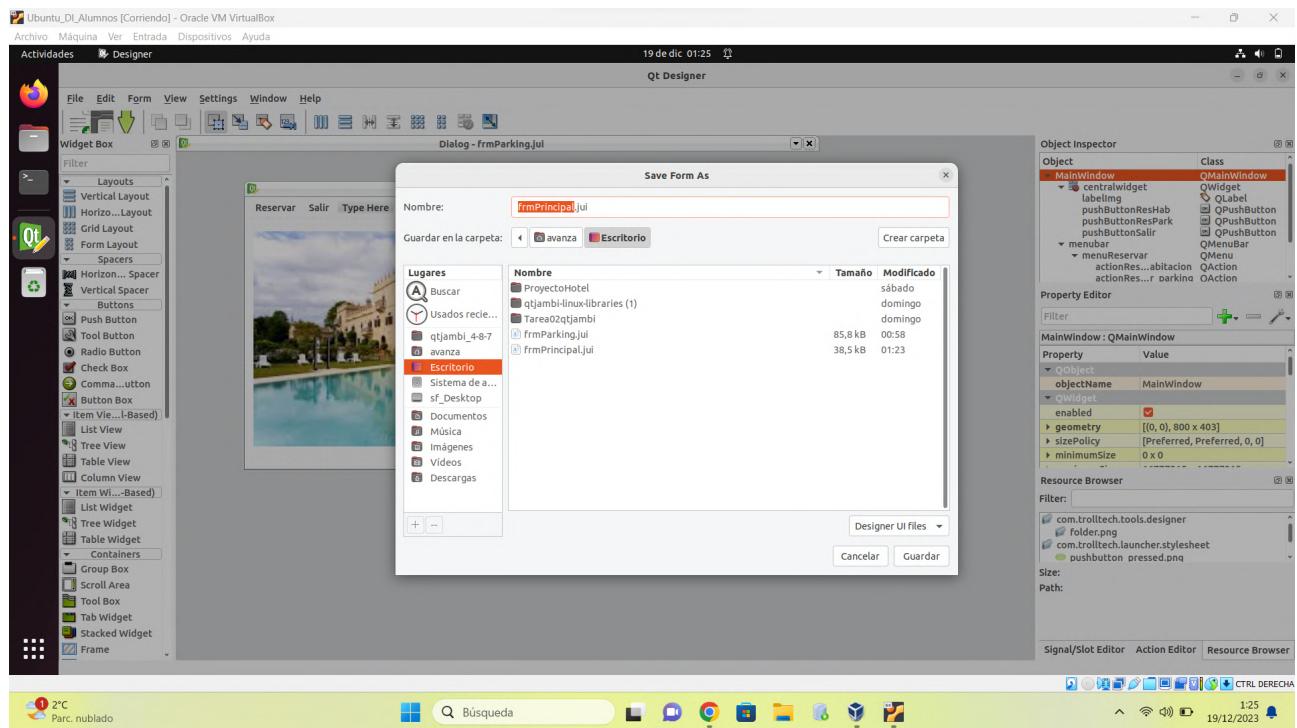


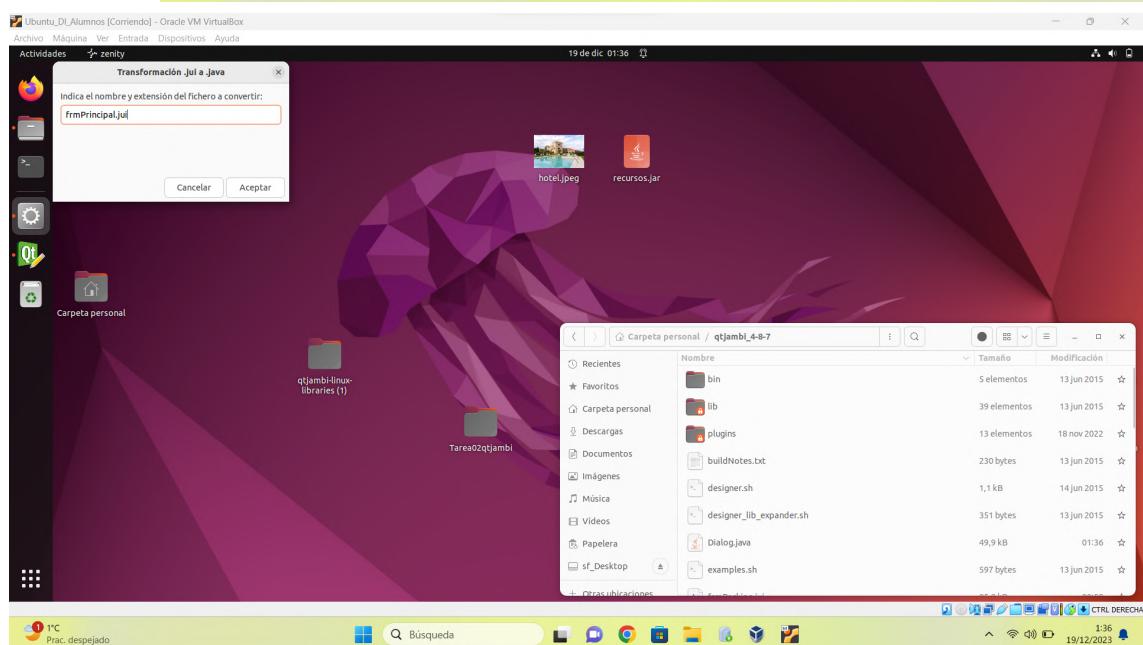
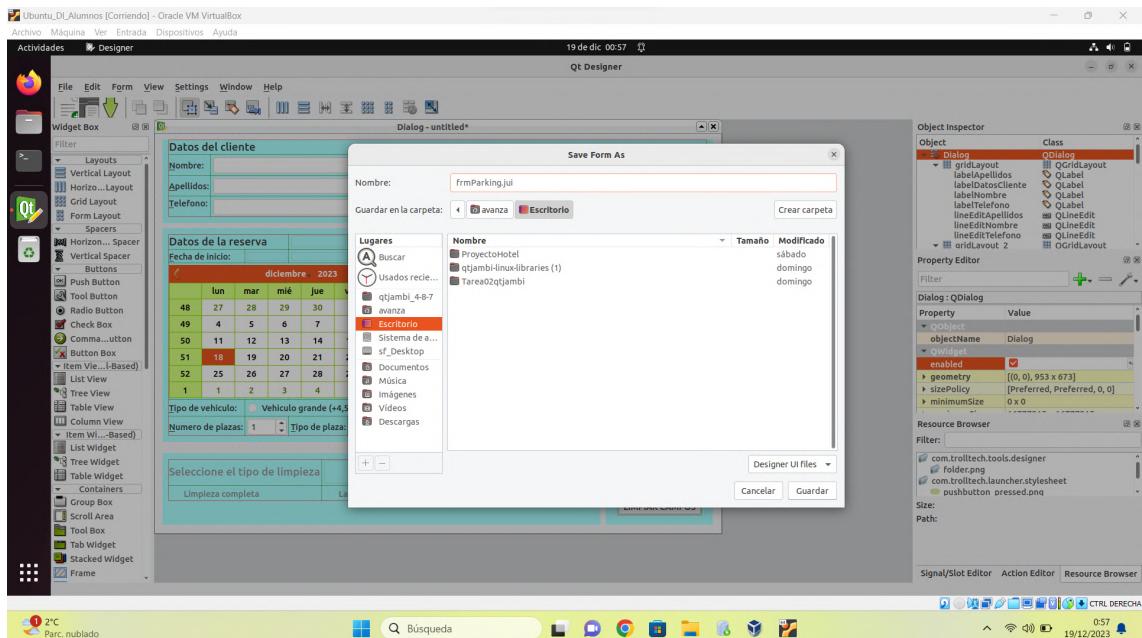
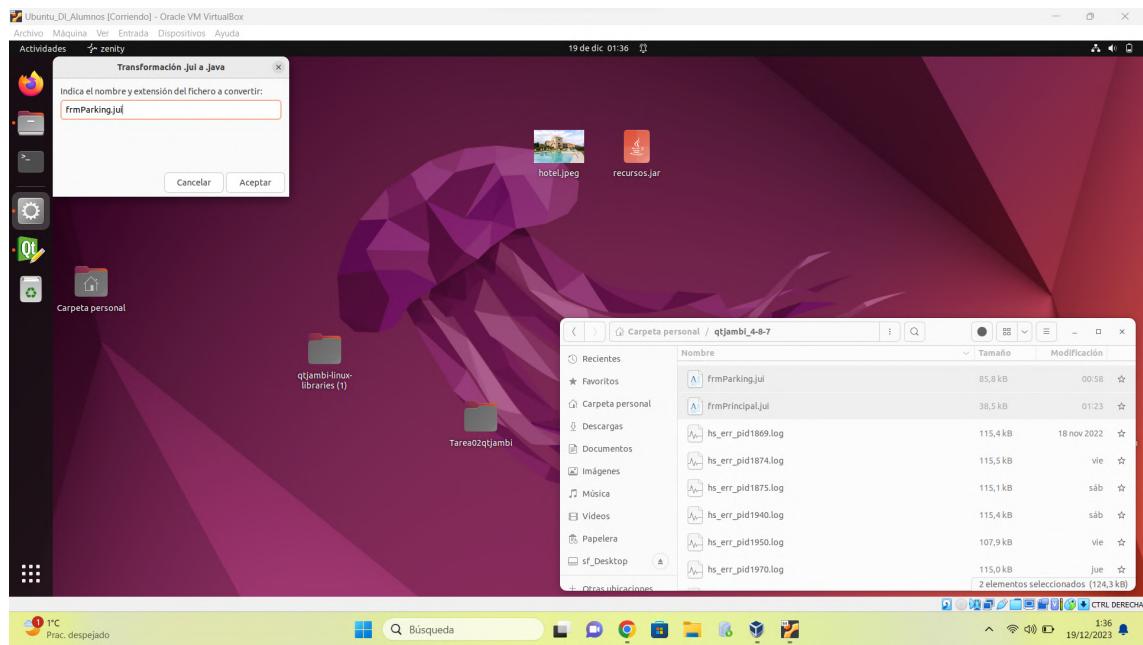
Una vez creada, añadida la imagen, creados los botones y el menú realizamos las conexiones Signal slot para los botones de la interfaz.

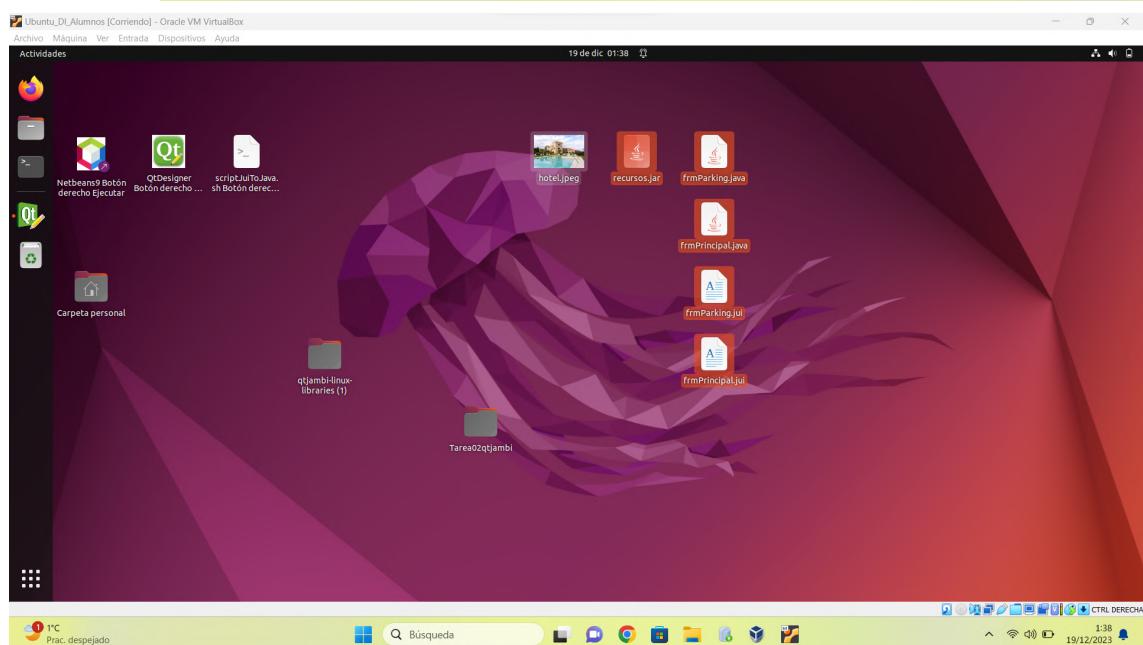
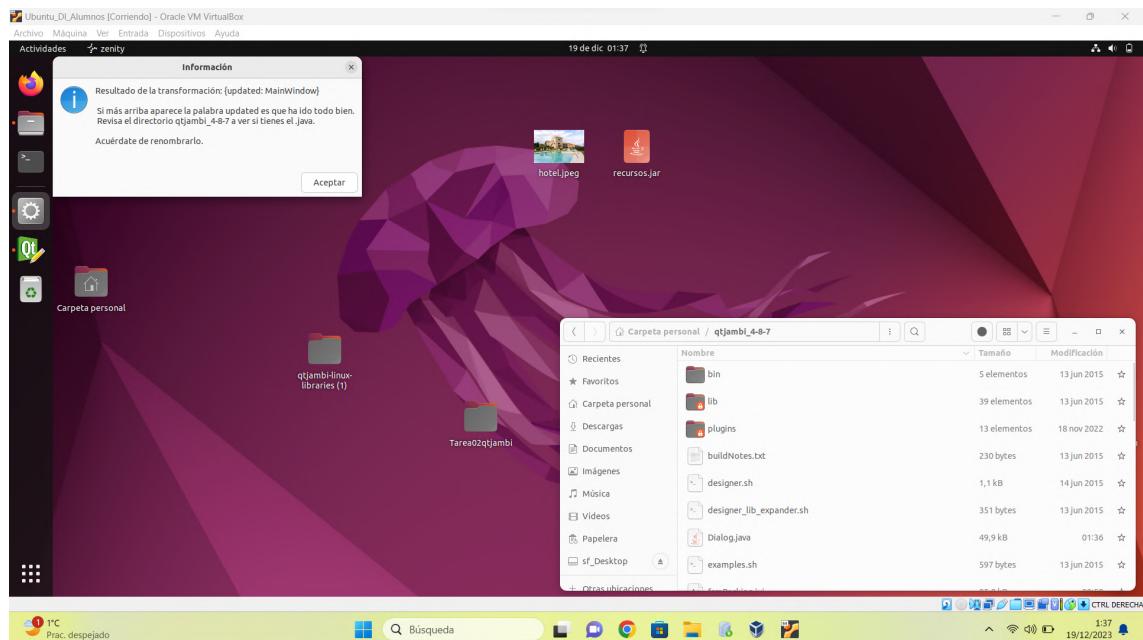
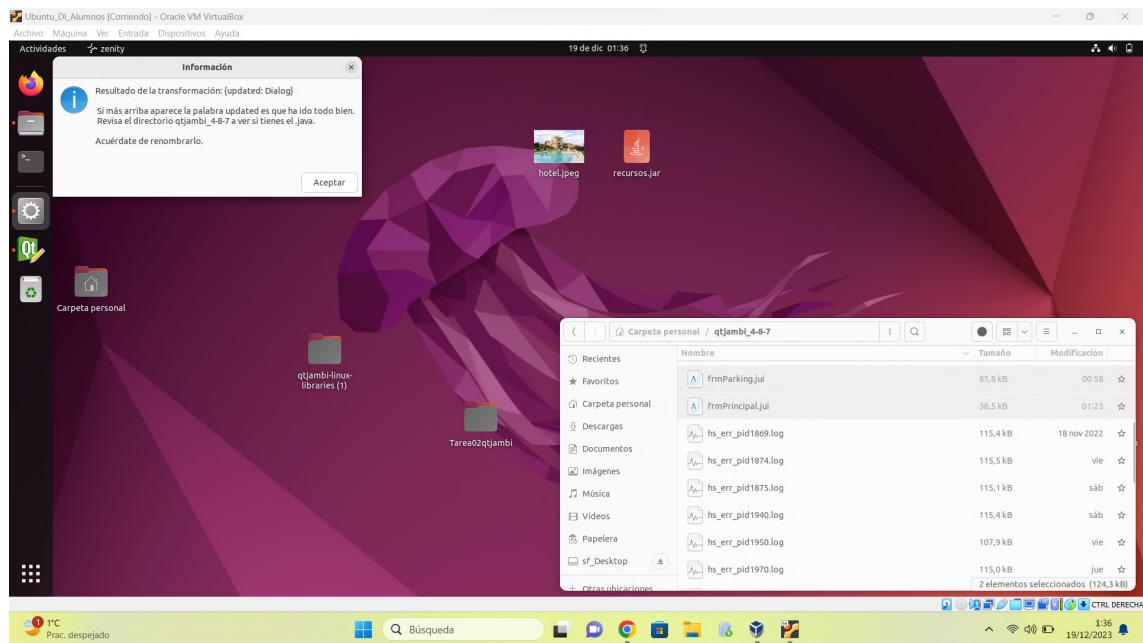


Posteriormente guardaremos los archivos .jui y los convertiremos a java usando el script ya creado en la maquina virtual. Para ello, una vez guardados los archivos .jui, los movemos al directorio qtjambi, después ejecutamos el script, ponemos el nombre completo del archivo que queremos convertir (frmPrincipal.jui y frmParking.jui en nuestro caso) y nos generara los archivos java.

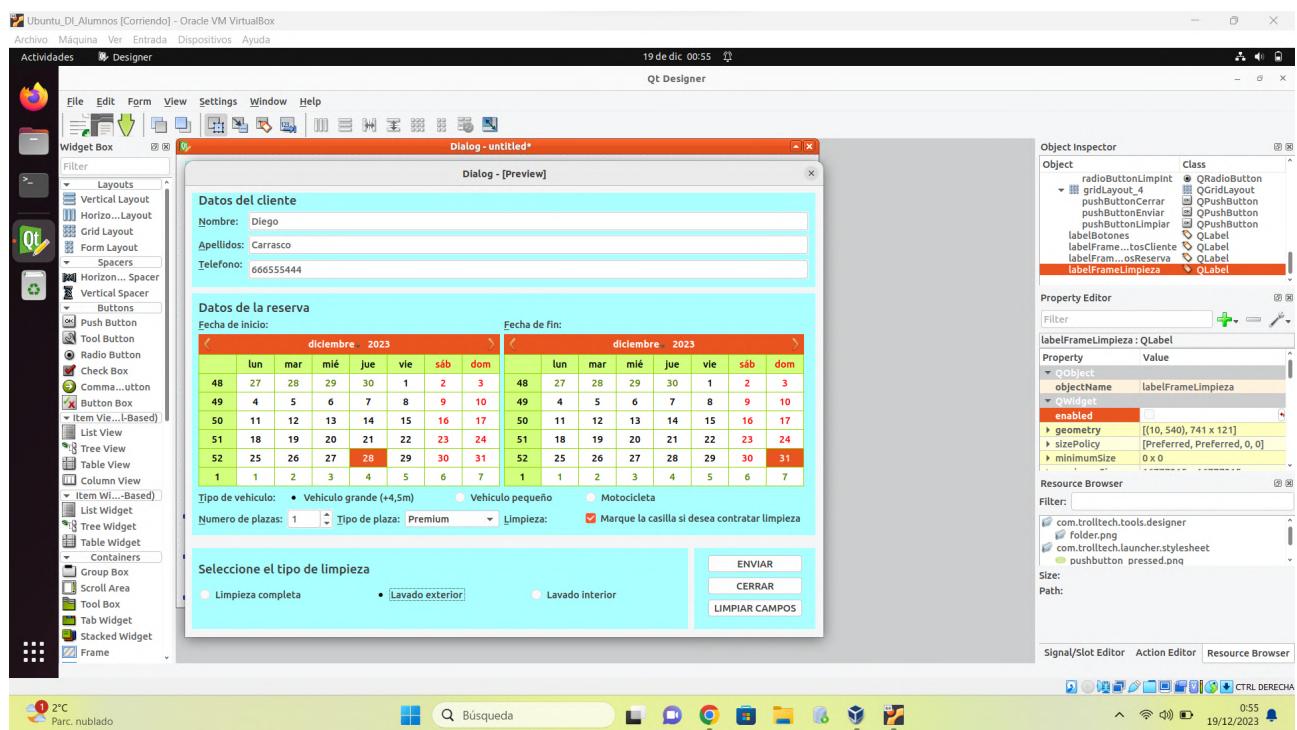
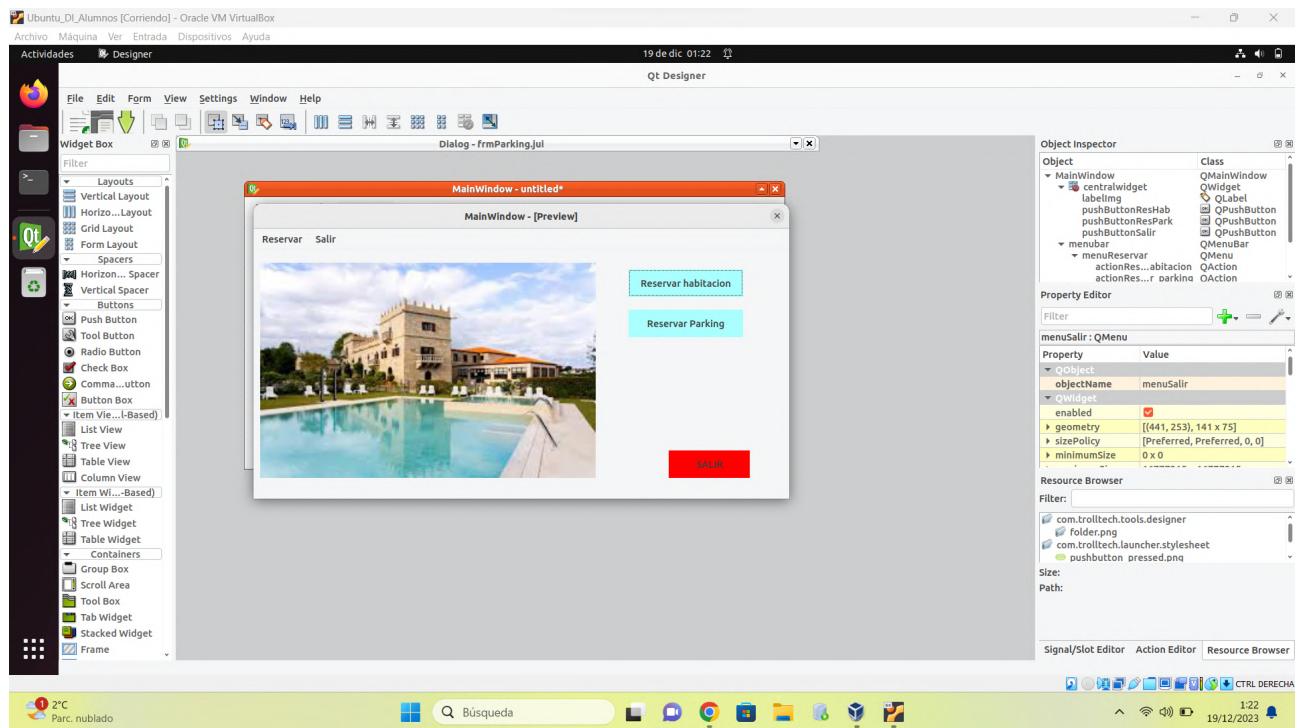
En las siguientes imágenes puede verse el proceso.



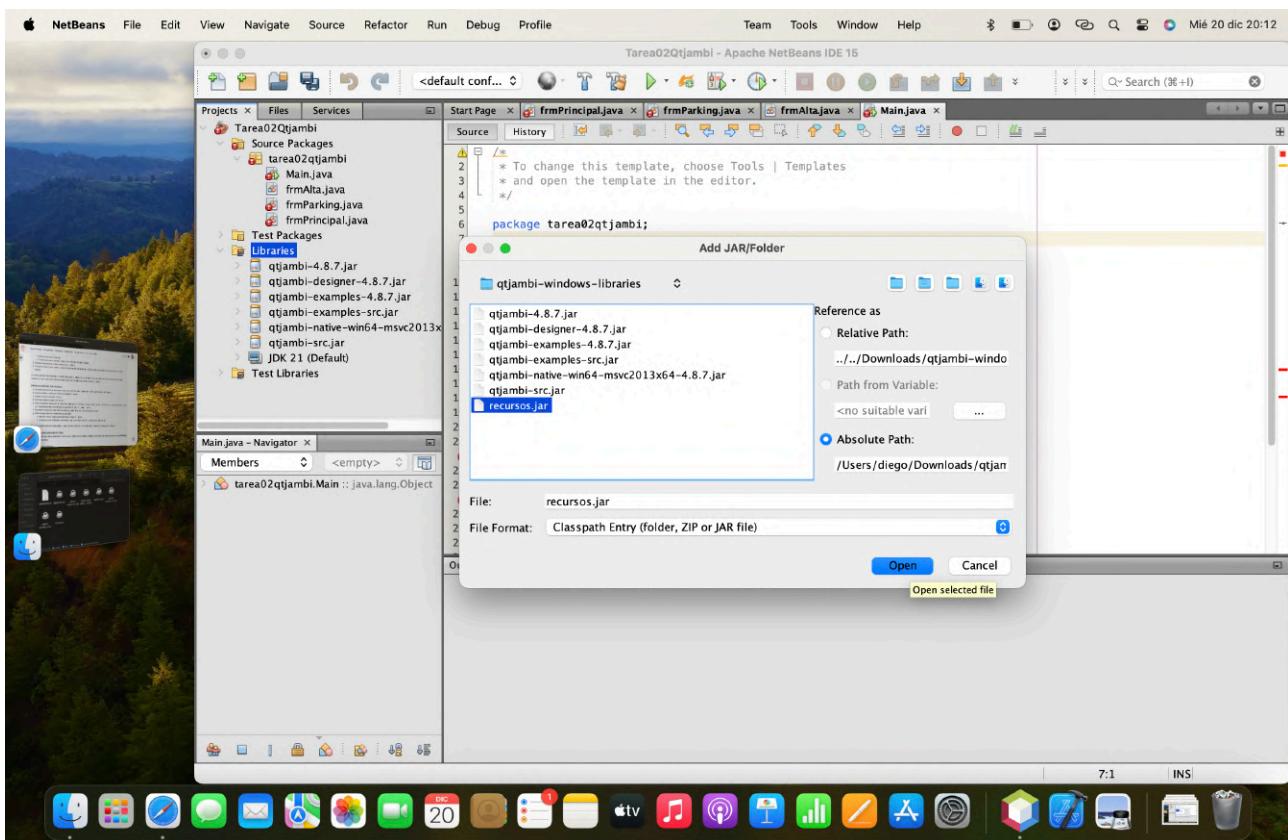
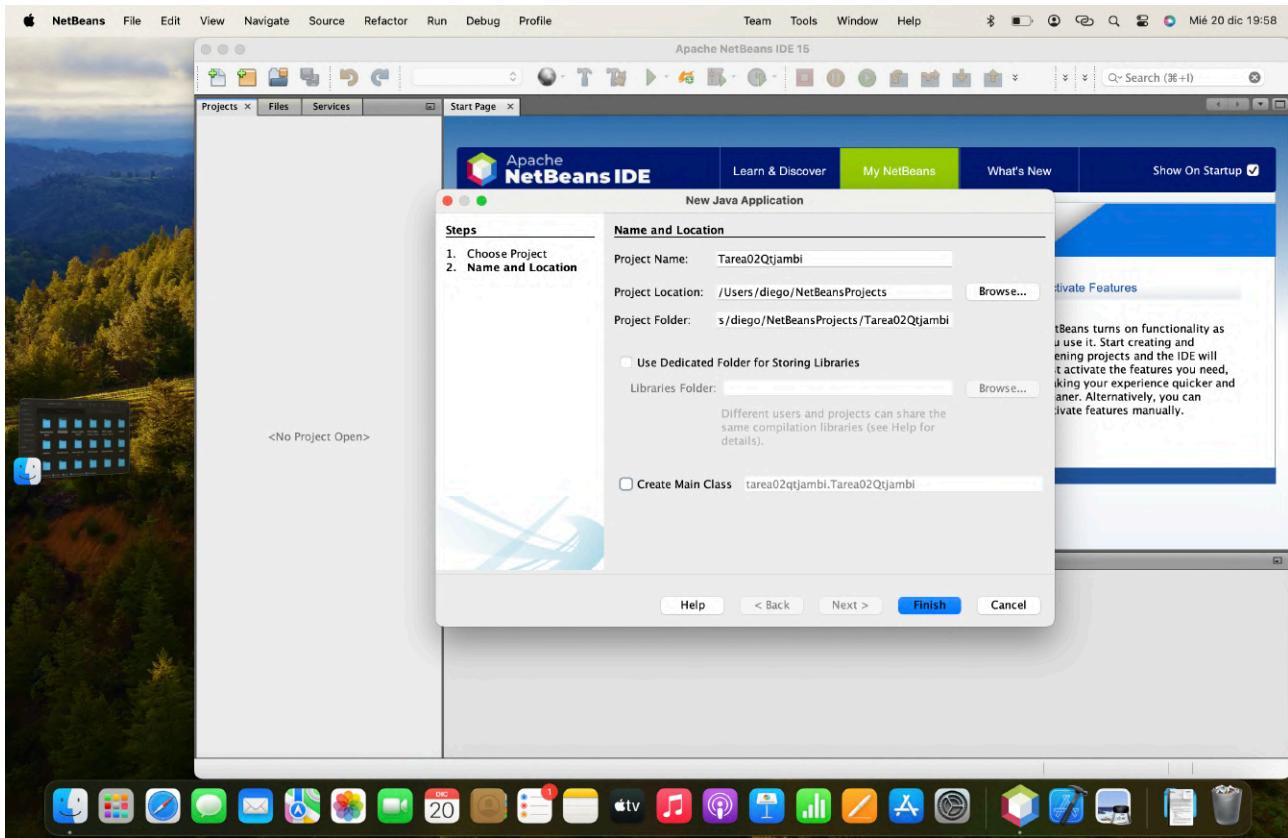


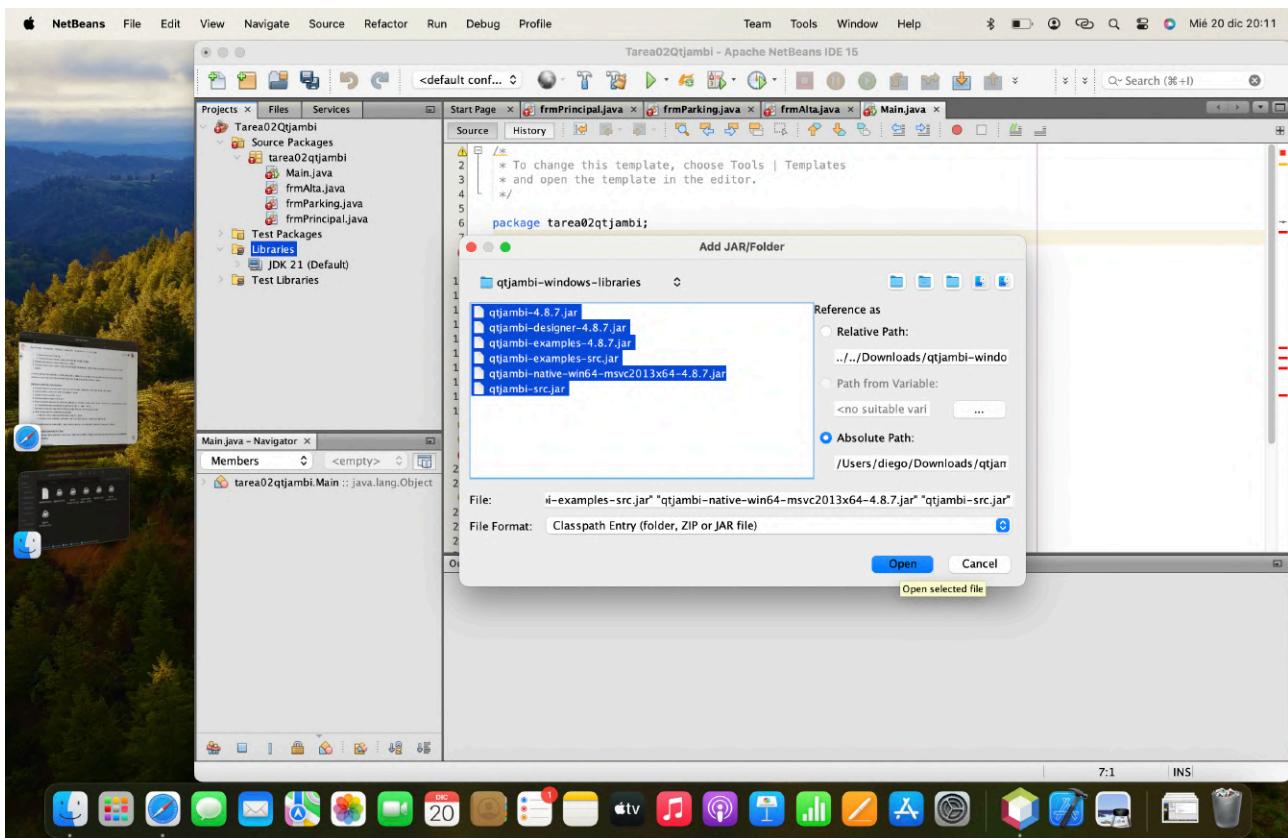
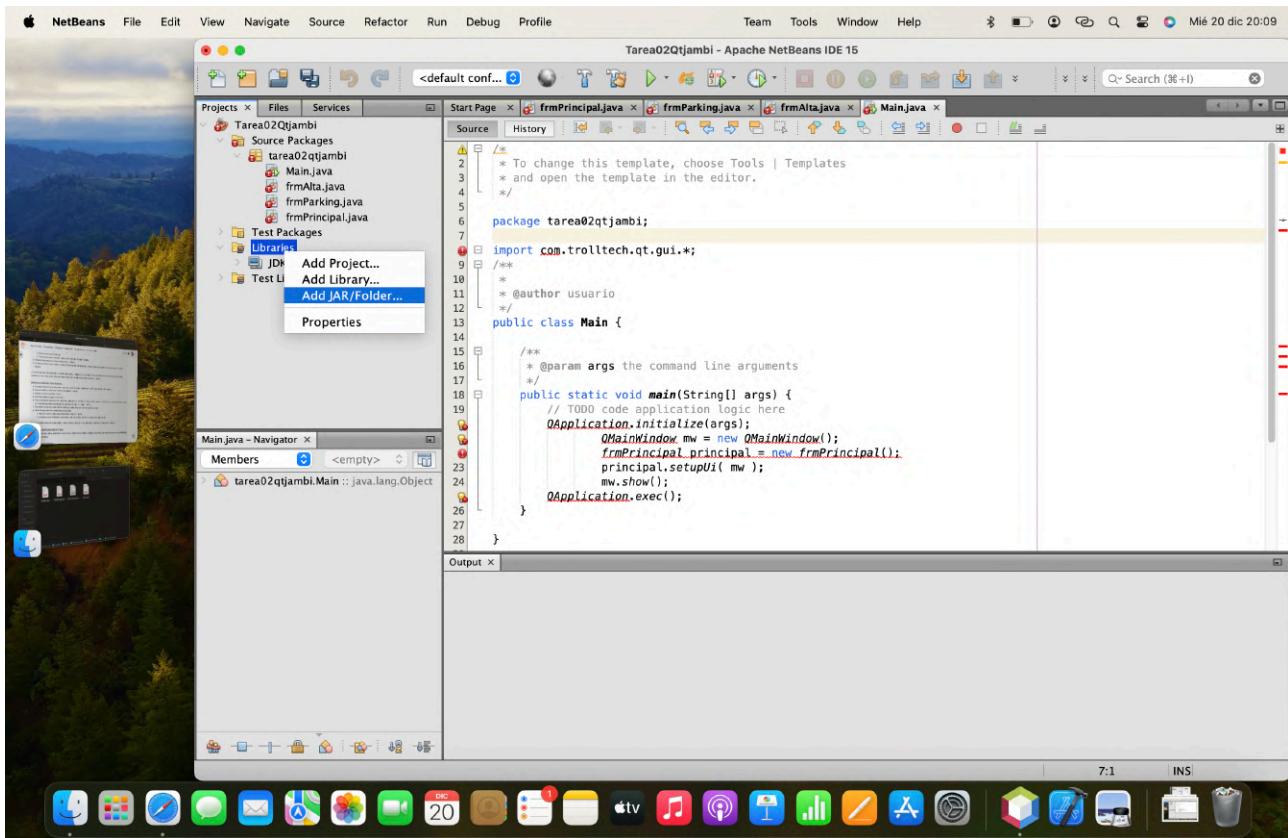


En las dos imágenes siguientes se puede ver como funcionarían las dos interfaces creadas mediante el preview de qtjambi.

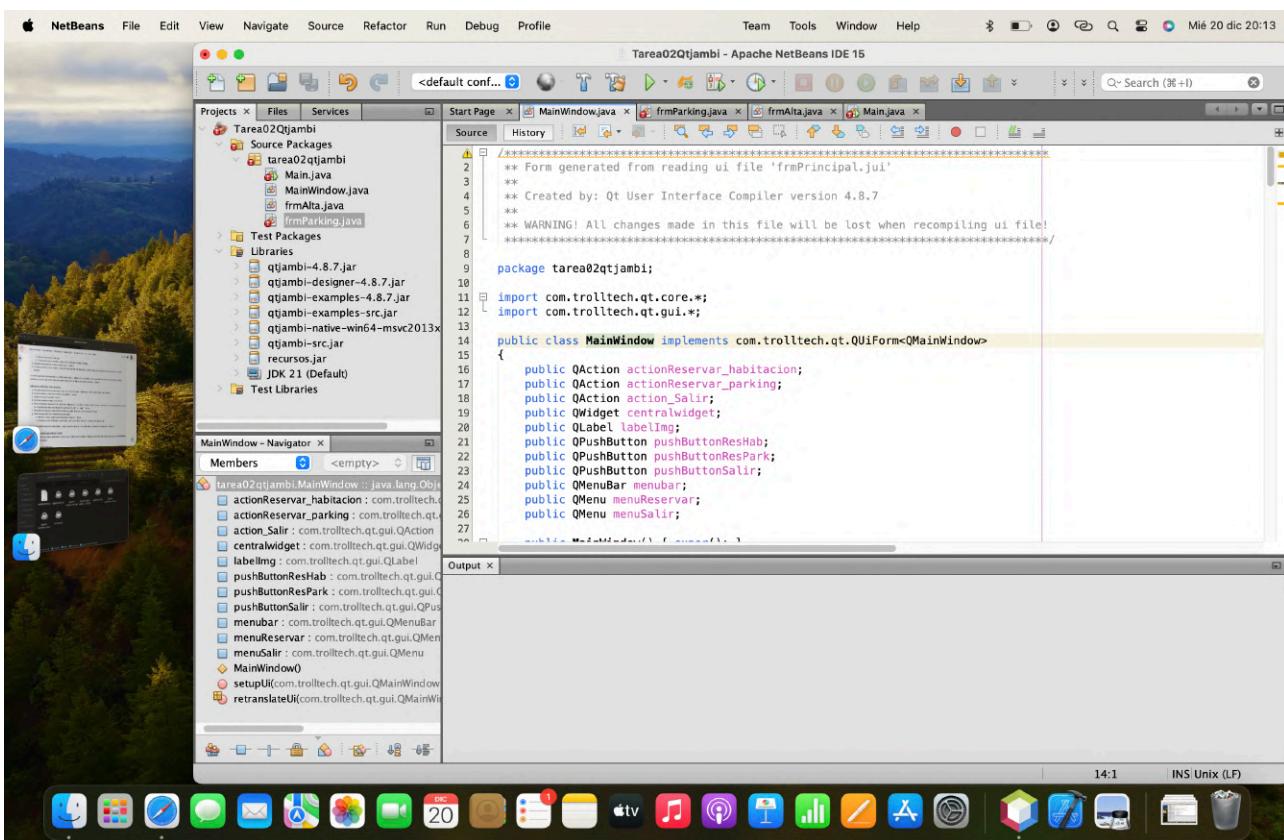
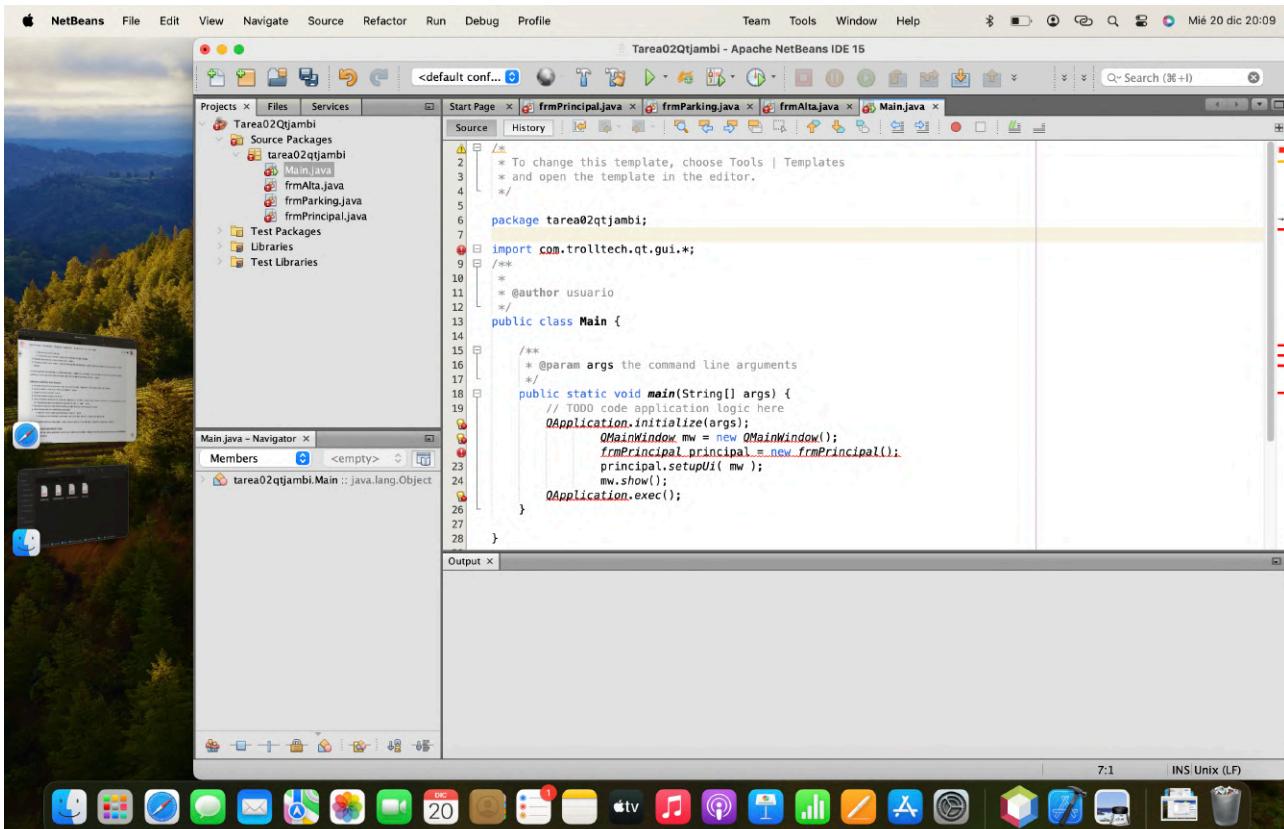


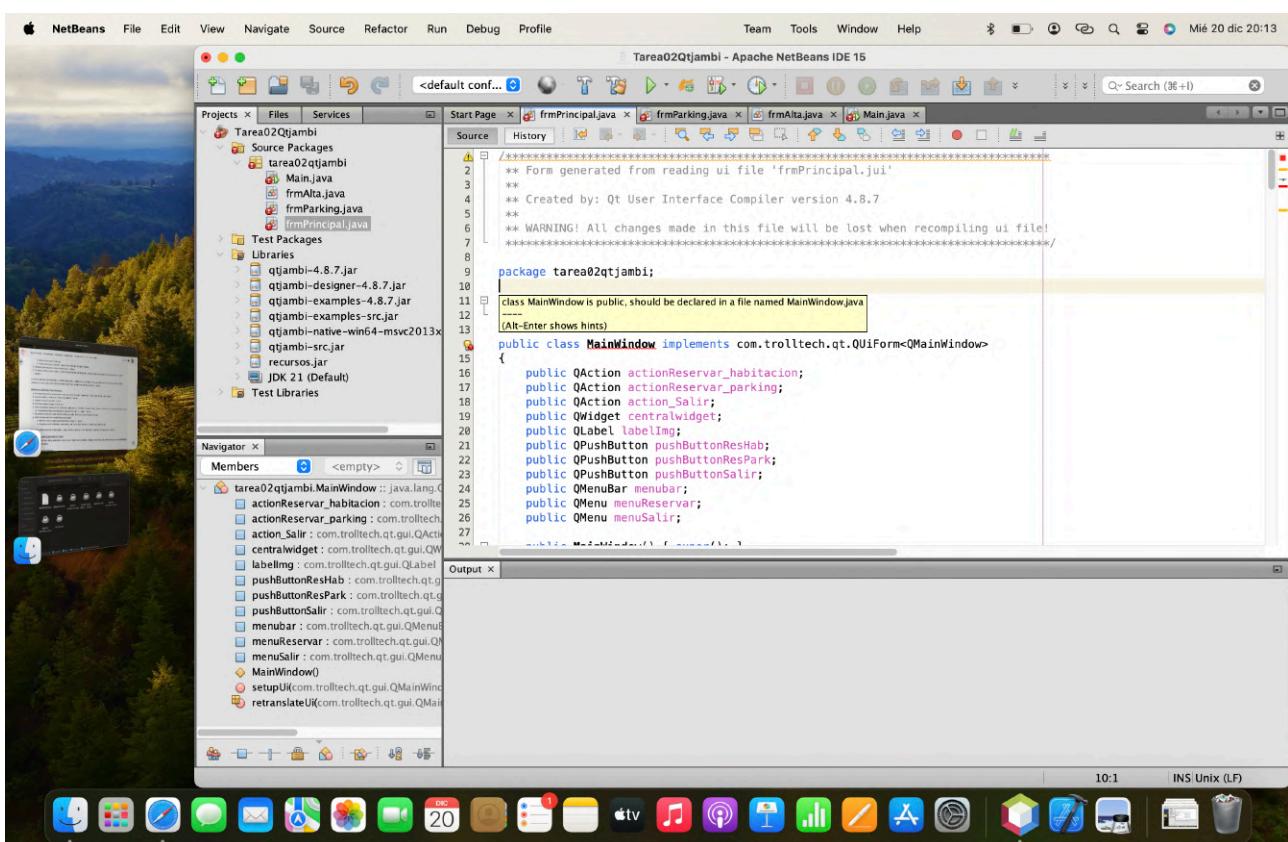
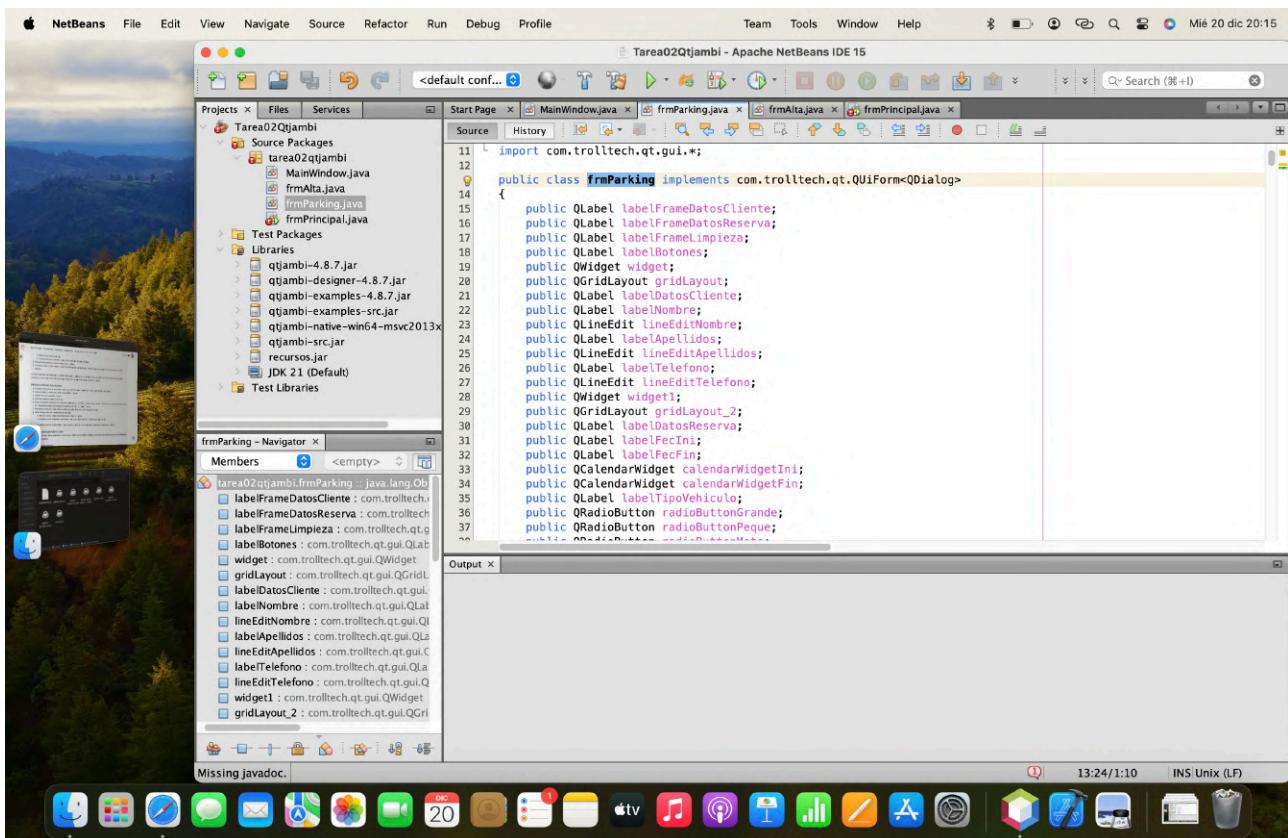
Ahora vamos a pasar a la parte de java. Vamos a crear un proyecto en netbeans llamado Tarea02Qtjambi sin método main, crearemos un paquete llamado igual que el proyecto y meteremos en el los archivos java creados a partir de qtjambi y los incluidos en la tarea. Una vez incluido renombrados los archivos creados en qtjambi, añadimos en cada clase el nombre del paquete donde se encuentran y añadiremos las librerías necesarias para su correcto funcionamiento según podemos ver en las siguientes imágenes.

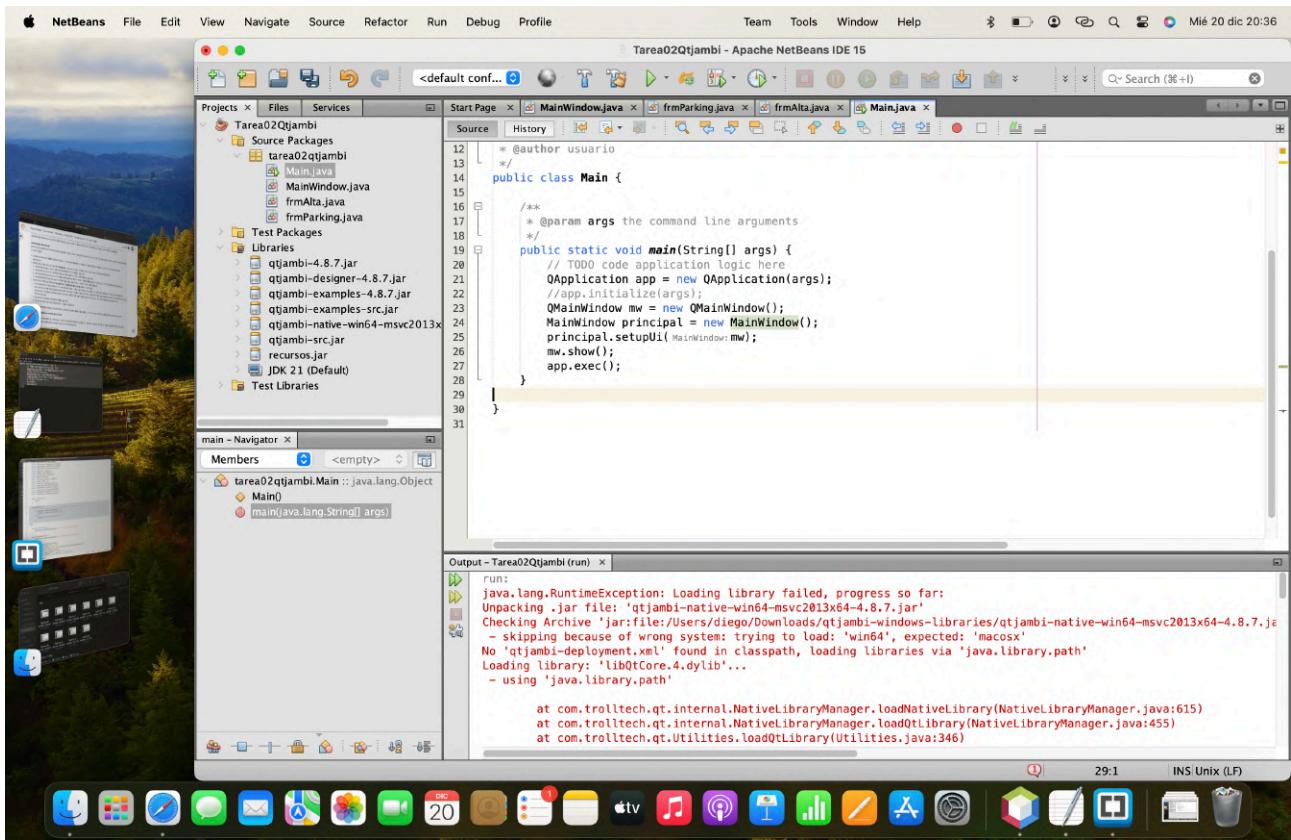




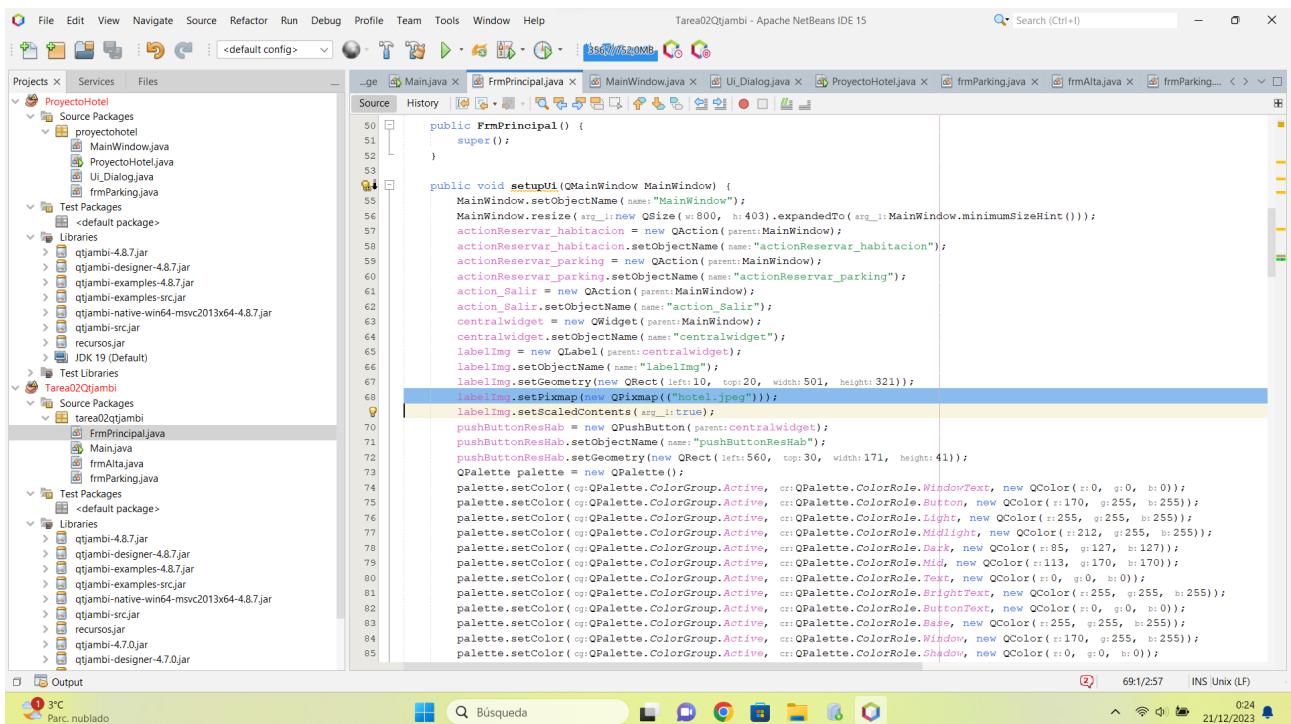
En las siguientes imágenes podemos ver los errores que me he ido encontrando tras añadir las librerías y como los he ido solucionando.







En la siguiente imagen muestro, en el código de la interfaz principal, donde se encuentra la linea que añade la imagen a dicha interfaz.



Posteriormente, en la clase frmPrincipal, crearemos los métodos necesarios para que nos abra los formularios de reservar habitación, reservar parking y salir de la aplicación respectivamente.

Dentro del método abrir crearemos e instanciaremos un objeto frmAlta, llamado Alta, un QDialog llamado dialog, le pasaremos Alta al método setupUi de dialogo y abriremos ese dialogo mediante el método show mostrándonos así el formulario de reservar habitaciones.

El método abriP es exactamente igual que el anterior pero con frmParking llamado p y nos abrirá el formulario para reservar el parking.

El método salir mostrara un mensaje por pantalla "la aplicación ha sido cerrada" y cerrara el formulario mediante el System.exit.

The screenshot shows the Apache NetBeans IDE 15 interface with the following details:

- File Menu:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Toolbar:** Standard Java development tools like New, Open, Save, Cut, Copy, Paste, Find, etc.
- Project Explorer:** Shows the project structure with packages, source files, and libraries.
 - ProyectoHotel:** Contains Source Packages (projectohotel), Test Packages (<default package>), Libraries (qtjambi-4.8.7.jar, qtjambi-designer-4.8.7.jar, qtjambi-examples-4.8.7.jar, qtjambi-examples-srcjar, qtjambi-native-win64-msvc2013x64-4.8.7.jar, qtjambi-srcjar, recursos.jar), and Test Libraries (JDK 19 (Default)).
 - Tarea02Qtjambi:** Contains Source Packages (tarea02qtjambi) which includes FrmPrincipal.java, Main.java, frmAlta.java, and frmParking.java.
 - <default package>:** Contains a file named <default package>.
- Code Editor:** The main window displays the `FrmPrincipal.java` source code. The code implements `com.trolltech.qt.QUIForm<QMainWindow>` and contains methods for opening windows (frmAlta, frmParking), and closing the application (salir). It also includes action definitions for QAction objects.
- Output:** Shows the status "3°C Parc. nublado".
- System Tray:** Shows icons for battery, signal strength, and date/time (21/12/2023).
- Bottom Bar:** Includes a search bar, system tray icons, and the message "INS Unix (LF)".

En la clase frmParking crearemos el método cambiarL para que, mediante un if, según este clicado o no el check box de limpieza, nos muestre o no todos los campos del frame de limpieza mediante el método setVisible y su valor boleado true o false.

También crearemos un método (llamado cerrarP) para mostrar un mensaje al cerrar la ventana del frmParking.

El siguiente método que vamos a crear se llamará enviarP, que mostrara el mensaje "Reserva realizada con éxito", limpiara los campos de los datos del cliente y cerrara la ventana frmParking.

Se puede ver en la siguiente imagen.

```

59     }
60
61     // Hacer visible o no los radiobutton de limpieza y su etiqueta
62     void cambiar() {
63         //si se ha activado la casilla de verificacion mostraremos el mensaje
64         if (checkboxLimpieza.isChecked()) {
65             labelTipoLimpieza.setVisible(true);
66             radioButtonN limpComp.setVisible(true);
67             radioButtonN limpExt.setVisible(true);
68             radioButtonLimpInt.setVisible(true);
69         } else {
70             labelTipoLimpieza.setVisible(false);
71             radioButtonN limpComp.setVisible(false);
72             radioButtonN limpExt.setVisible(false);
73             radioButtonLimpInt.setVisible(false);
74         }
75     }
76
77     // Metodo para que al cerrar la ventana mostramos el mensaje
78     void cerrarP() {
79         JOptionPane.showMessageDialog(parentComponent, null, message: "La ventana se va a cerrar");
80     }
81
82     // Metodo para mostrar mensajes, limpiar los cambios y cerrar la ventana al pulsar enviar
83     void enviarP() {
84         JOptionPane.showMessageDialog(parentComponent, null, message: "Reserva realizada con exito");
85         lineEditNombre.clear();
86         lineEditApellidos.clear();
87         lineEditTelefono.clear();
88         cerrarP();
89     }
90
91     @Override
92     public void setupUi(Dialog Dialog) {
93         ...
94     }

```

En las siguientes imágenes podemos ver las acciones que hacen cada botón, tanto de frmParkin como de frmPrincipal, a las cuales debemos de pasarle el contexto y el método al que deben de llamar al pulsar los botones correspondientes.

```

596     gridLayout_4.addWidget(arg_1:pushButtonLimpiar, row:2, column:0, rowspan:1, columnSpan:1);
597
598     labelNombre.setBuddy(arg_1:lineEditNombre);
599     labelApellido.setBuddy(arg_1:lineEditApellidos);
600     labelTelefono.setBuddy(arg_1:lineEditTelefono);
601     labelFecIni.setBuddy(arg_1:calendarWidgetIni);
602     labelFecFin.setBuddy(arg_1:calendarWidgetFin);
603     labelTipoVehiculo.setBuddy(arg_1:radioButtonGrande);
604     labelPlaza.setBuddy(arg_1:spinBoxPlazas);
605     labelTipoPlaza.setBuddy(arg_1:comboBoxTipoPlaza);
606     labelLimpieza.setBuddy(arg_1:checkboxLimpieza);
607     retranslateUi(Dialog);
608     // Mostrar o no el frame de limpieza si esta clicado
609     checkBoxLimpieza.clicked.connect(receiver:labelFrameLimpieza, method: "setEnabled(boolean)");
610     // Al pulsar cerrar llamamos al metodo cerrar
611     pushButtonCerrar.clicked.connect(receiver:this, method: "cerrarP()");
612     // Al pulsar el boton Limpiar limpiamos los campos
613     pushButtonLimpiar.clicked.connect(receiver:lineEditNombre, method: "clear()");
614     pushButtonLimpiar.clicked.connect(receiver:lineEditApellidos, method: "clear()");
615     pushButtonLimpiar.clicked.connect(receiver:lineEditTelefono, method: "clear()");
616     // Al estar seleccionado el checkbox limpieza desabilitamos los radiobutton y el label de limpieza
617     checkBoxLimpieza.clicked.connect(receiver:radioButtonLimpExt, method: "setEnabled(boolean)");
618     checkBoxLimpieza.clicked.connect(receiver:radioButtonLimpInt, method: "setEnabled(boolean)");
619     checkBoxLimpieza.clicked.connect(receiver:labelTipoLimpieza, method: "setEnabled(boolean)");
620     checkBoxLimpieza.clicked.connect(receiver:radioButtonLimpComp, method: "setEnabled(boolean)");
621     // Al pulsar el boton enviar llamamos al metodo enviarP
622     pushButtonEnviar.clicked.connect(receiver:this, method: "enviarP()");
623
624     Dialog.connectSlotsByName();
625
626     void retranslateUi(Dialog Dialog) {
627         Dialog.setWindowTitle(arg_1:com.trolltech.qt.core.QCoreApplication.translate(context: "Dialog", sourceText: "Dialog", comment: null));
628         labelFrameDatosCliente.setToolTip(arg_1:com.trolltech.qt.core.QCoreApplication.translate(context: "Dialog", sourceText: "<html><br>"));
629         labelFrameDatosCliente.setText(arg_1: "");
630     }
631
632     // setupUi

```

```

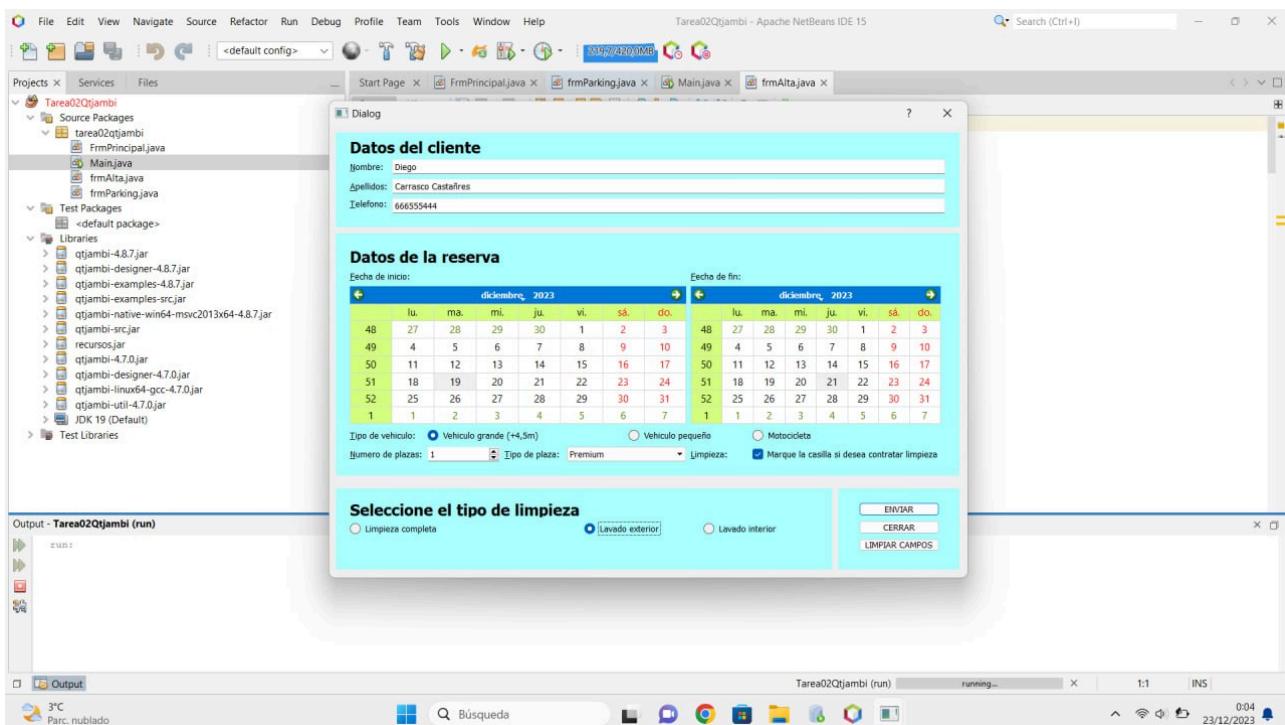
229     menubar.setGeometry(new QRect( left:0, top:0, width:800, height:31));
230     menubar.setContextMenuPolicy(policy:com.trolltech.qt.core.Qt.ContextMenuPolicy.DefaultContextMenu);
231     menuReservar = new QMenu(parent:"menubar");
232     menuReservar.setObjectName(name:"menuReservar");
233     menuSalir = new QMenu(parent:"menubar");
234     menuSalir.setObjectName(name:"menuSalir");
235     MainWindow.setMenuBar(menuBar("menubar"));
236
237     menubar.addAction(action:menuReservar.menuAction());
238     menubar.addAction(action:menuSalir.menuAction());
239     menuReservar.addAction(action:actionReservar_habitacion);
240     menuReservar.addAction(action:actionReservar_parking);
241     menuSalir.addAction(action:action_Salir);
242     retranslateUi(MainWindow);
243     // Abrir frmAlta desde el boton
244     pushButtonResab.clicked.connect(receiver:this, method:"abrir()");
245     // Abrir frmParking desde el boton
246     pushButtonResPark.clicked.connect(receiver:this, method:"abrirP()");
247     // Cerrar frmPrincipal desde el boton
248     pushButtonSalir.clicked.connect(receiver:MainWindow, method:"close()");
249     // Cerrar frmPrincipal desde el boton salir del menubar
250     action_Salir.triggered.connect(receiver:MainWindow, method:"close()");
251     // Abrir frmAlta desde el boton "Reservar habitacion" del menubar
252     actionReservar_habitacion.triggered.connect(receiver:this, method:"abrir()");
253     // Abrir frmParking desde el boton "Reservar parking" del menubar
254     actionReservar_parking.triggered.connect(receiver:this, method:"abrirP()");
255
256     MainWindow.connectSlotsByName();
257 } // setupUi
258
259 void retranslateUi(QMainWindow MainWindow) {
260     MainWindow.setWindowTitle(text:com.trolltech.qt.core.QCoreApplication.translate(context:"MainWindow", sourceText:"MainWindow"));
261     actionReservar_habitacion.setText(text:com.trolltech.qt.core.QCoreApplication.translate(context:"MainWindow", sourceText:"Reservar habitacion"));
262     actionReservar_habitacion.setToolTip(toolTip:com.trolltech.qt.core.QCoreApplication.translate(context:"MainWindow", sourceText:"Se reservara una habitacion"));
263     actionReservar_parking.setText(text:com.trolltech.qt.core.QCoreApplication.translate(context:"MainWindow", sourceText:"Reservar parking"));
264 }

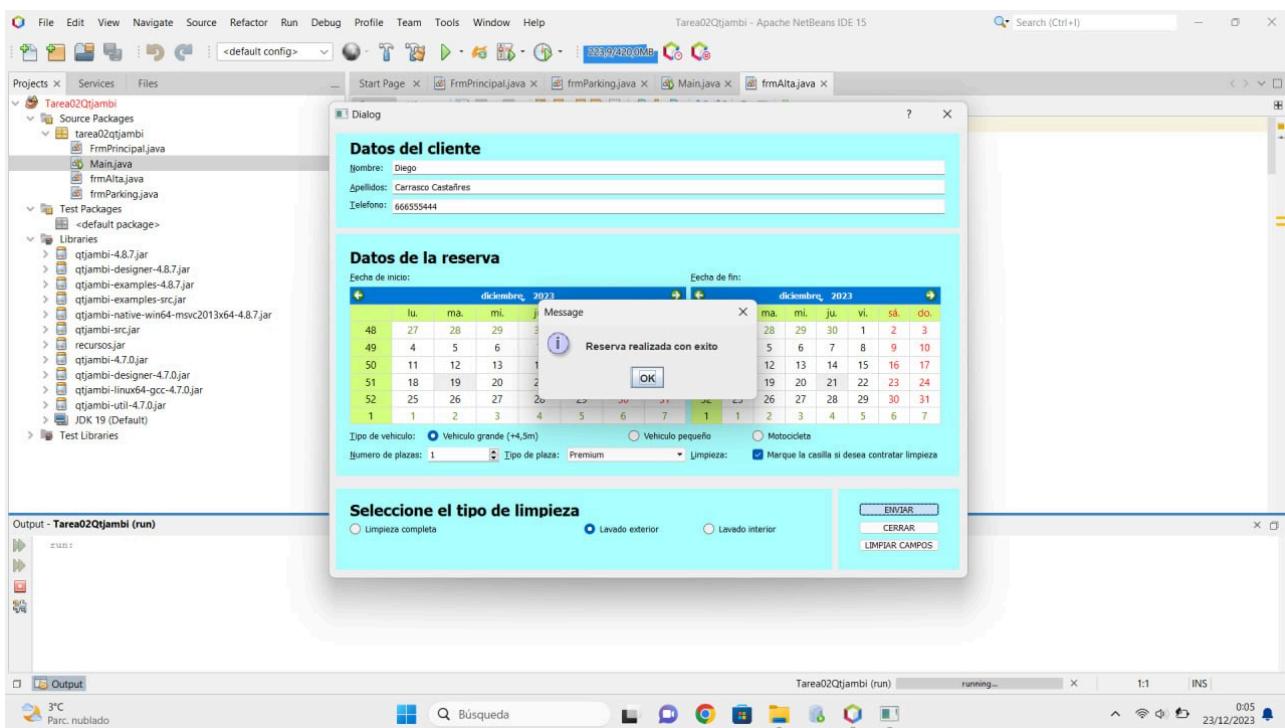
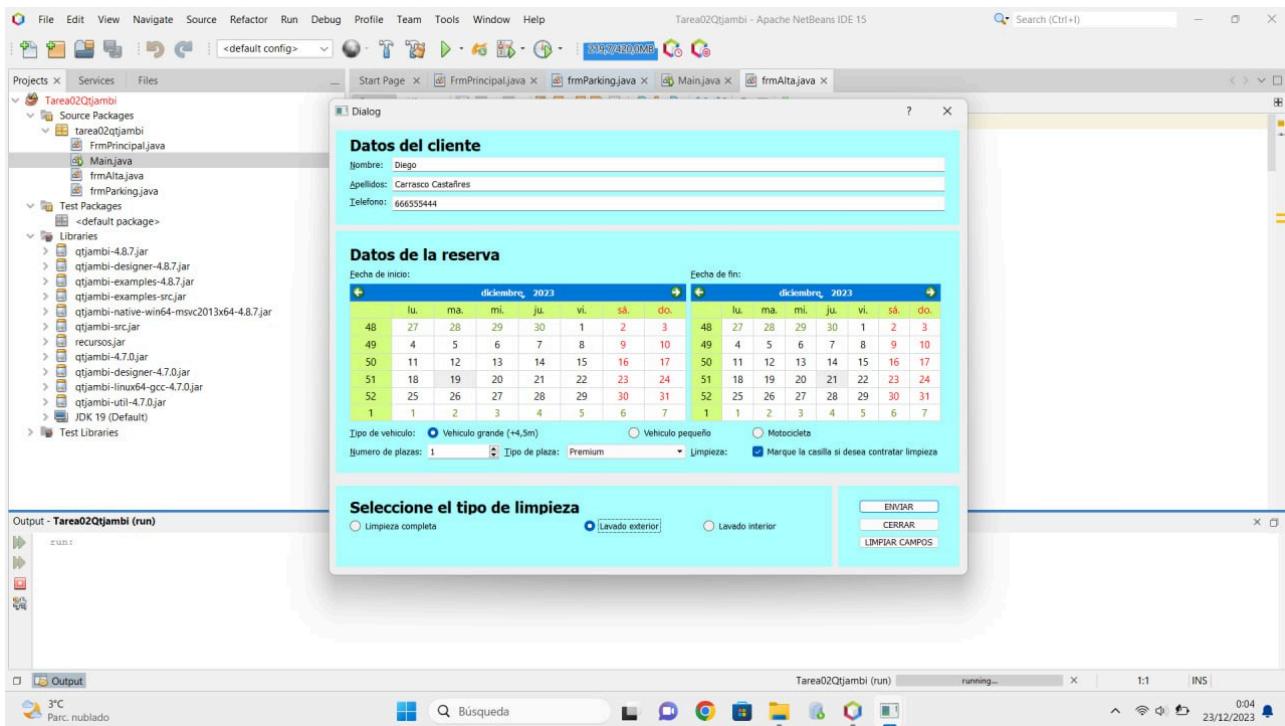
```

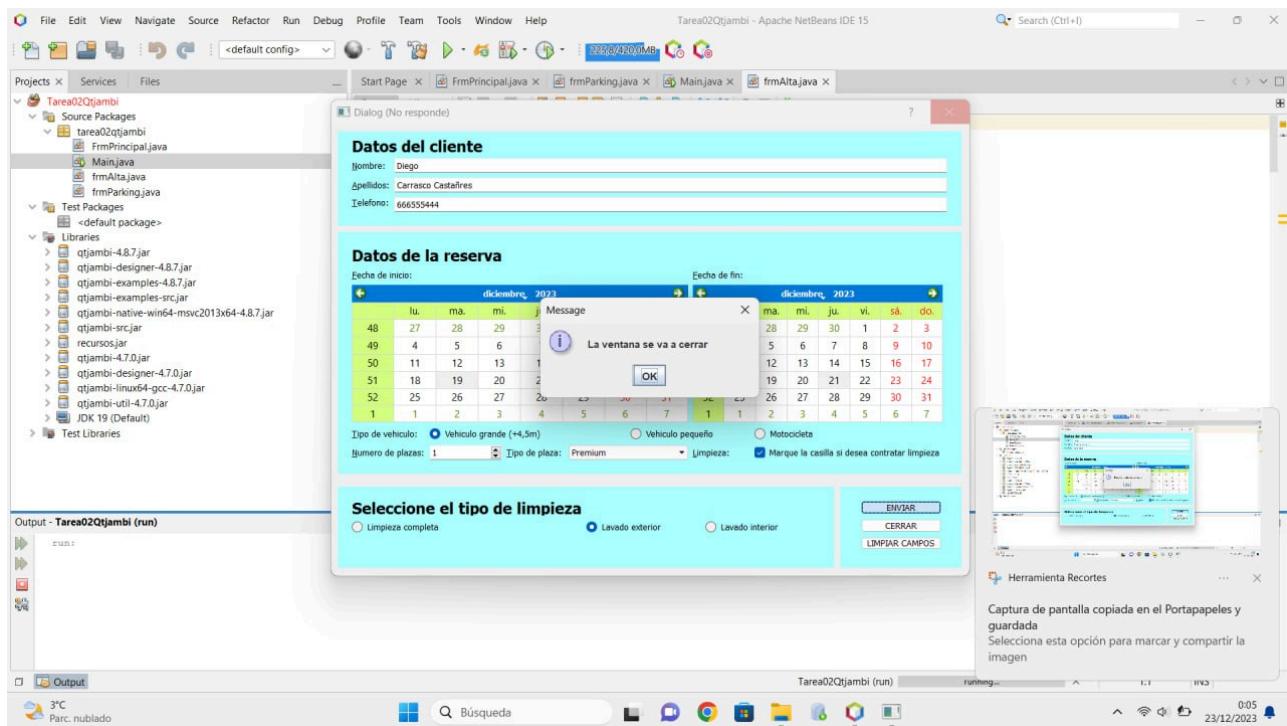
Output: 25741/1:3 INS Unix (LF)

3°C Parc. nublado Búsqueda 028 21/12/2023

En las siguientes imágenes podemos ver como esta la aplicación funcionando.







Y con esto damos por concluida esta tarea.

Detalles de la tarea de esta unidad.

Enunciado.

La empresa BK continua ampliando las interfaces para la aplicación de gestión hotelera.

Además de la reserva de habitaciones (cuyo desarrollo se ha explicado en los contenidos de esta unidad), los usuarios también quieren hacer reservas del parking del que dispone el Hotel. La práctica que debes realizar consiste en crear una interfaz, empleando la herramienta **QTDesigner**, que permita gestionar la reserva de las plazas de parking. Los requisitos que debe cumplir esta interfaz son:

1. Se debe escribir el nombre, apellidos y teléfono de contacto de la persona que hace la reserva.
2. Hay que cumplimentar:
 - Fecha de inicio de la reserva.
 - Fecha de fin de la reserva.
 - Tipo. Se puede escoger entre vehículo grande (mayor de 4,5 m de largo), vehículo pequeño o motocicleta.
 - Número de plazas de parking.
 - Tipo de plaza que se prefiere, a elegir entre Estándar, Premium o Deluxe.
3. Preguntar si se requiere servicio de limpieza del vehículo.
4. Si se activa el servicio de limpieza, el usuario deberá poder cumplimentar si contrata Limpieza completa, Limpieza interior o Lavado exterior.

La interfaz generada irá conectada a la interfaz principal de la aplicación, que también se ha desarrollado durante el estudio de la unidad, mediante una nueva opción del menú principal y también por medio de un botón que abrirá la interfaz.

Criterios de calificación. Total 10 puntos.

- Creación de la interfaz de reservas del parking, con QTDesigner, incluyendo el archivo con formato .jui. 2 puntos.
- Asociar buddies y establecer el orden de tabulación: 1 punto.
- Asociar un layout apropiado: 1 punto.
- Crear las conexiones signal/slot: 1 punto.
- Crear un proyecto Java a partir de los ficheros adjuntos y los creados en los apartados anteriores. (tendría 3 .java correspondientes a los 3 .jui -recuerda que debes renombrarlo tras emplear QTJambi- y el main). 1 punto.
- Documento con la explicación del desarrollo y ejecución de la tarea en formato pdf. 2 puntos.
- Añadir las siguientes funcionalidades programadas:
 - Habilitar o no los widgets para el servicio de limpieza. 1 punto.
 - Código para abrir el dialogo nuevo desde el menú y desde el botón de la interfaz principal. 1 punto.

Se valorará positivamente que seas original y hagas aportaciones propias, por ejemplo, incluyendo una imagen en la interfaz.

Recursos necesarios para realizar la Tarea.

Para realizar esta tarea debes partir de los archivos con el ejemplo desarrollado a lo largo de la unidad, que puedes encontrar como **ARCHIVOS ADJUNTOS** más abajo.

Consejos y recomendaciones.

Se pretende poner en práctica los conceptos aprendidos, de la forma más clara posible, por lo que te recomiendo que hagas un diseño previo de la interfaz y estudies dónde se van a posicionar los elementos de formulario para evitar tener que repetir varias veces el proceso, ya que cada vez que tenemos que generar la interfaz de nuevo con juic, el archivo anterior se borra y los cambios realizados se pierden.

Indicaciones de entrega.

Una vez realizada la tarea tendrás que comprimir los archivos del proyecto. La estructura de archivos a entregar dentro del archivo comprimido es como sigue:

- Un directorio llamado **leeme**, que contenga un archivo .pdf detallando el desarrollo que has seguido para la realización de la tarea (incluye pantallazos).
- Dentro de un directorio de nombre **interfaces** (se ponen en cursiva los ficheros nuevos que se deben desarrollar):

- La interfaz principal, con el nombre de archivo `frmPrincipal.jui` (adjunto).
- La interfaz para reservar habitaciones, con el nombre de archivo `frmAlta.jui` (adjunto).
- La nueva interfaz para realizar la reserva del parking (tal y como se ha especificado más arriba) en un archivo llamado `frmParking.jui`.
- Un directorio llamado recursos con la imagen que aparece en la interfaz principal (adjunto).
- Dentro de un directorio llamado **programa** un **proyecto Java completo** que contenga:
 - Los paquetes con los archivos `.java` correspondientes a cada uno de los tres archivos `.jui` (`frmPrincipal.java` y `frmAlta.java`, adjuntos). Recuerda que `frmPrincipal.java` debe modificarse para incluir el nuevo botón y opción de menú que permita acceder a la reserva de plazas de parking. Así mismo, debes renombrar el `.java` que se obtiene mediante QTJambi a partir del `frmParking.jui`; debe llamarse `frmParking.java`.
 - El archivo `Main.java` con la clase `Main` (adjunto).
 - Un archivo llamado `recursos.jar` con la imagen (adjunto).

El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_DIx_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la segunda unidad del DI, debería nombrar esta tarea como...

sanchez_manas_begona_DI02_Tarea