MESA INTERACTIVA

ZUR RESTAURANT

Sistema de gestión de pedidos desde mesa interactiva

Adrián Espí Peña

Desarrollo de aplicaciones Multiplataforma

20/05/2023

ABSTRACT

This project aims to develop a software solution to streamline the ordering process in a restaurant by deploying an interactive table or board that allows customers to place their orders directly from their seats. The proposed system will include a touch screen display featuring the restaurant menu and an integrated kiosk application that enables customers to place and customize their orders without having to queue up at the cash register.

The primary objective of this project is to enhance the efficiency of the restaurant ordering process while improving the overall customer experience. By implementing an interactive ordering system, the restaurant can reduce waiting times and increase customer satisfaction, ultimately resulting in improved sales and revenue.

In conclusion, this project proposes an innovative solution to improve the ordering process in a restaurant, enhancing customer satisfaction and improving business efficiency. The proposed interactive table will allow customers to order and customize their meals directly from their seats and thus, an improvement for both clients and restaurants.

RESUMEN

Este proyecto tiene como objetivo desarrollar una solución de software para agilizar el proceso de pedidos en un restaurante mediante la implementación de una mesa o panel interactivo que permita a los clientes realizar sus pedidos directamente desde sus asientos. El sistema propuesto incluirá una pantalla táctil que mostrará el menú del restaurante y una aplicación de quiosco integrada que permitirá a los clientes realizar y personalizar sus pedidos sin tener que hacer cola en la caja registradora.

El objetivo principal de este proyecto es mejorar la eficiencia del proceso de pedido del restaurante y, al mismo tiempo, mejorar la experiencia general del cliente. Al implementar un sistema de pedidos interactivo, el restaurante puede reducir los tiempos de espera y aumentar la satisfacción del cliente, lo que finalmente resulta en mejores ventas e ingresos.

En conclusión, este proyecto propone una solución innovadora para mejorar el proceso de pedidos en un restaurante, aumentando la satisfacción del cliente y mejorando la eficiencia del negocio. La mesa interactiva propuesta permitirá a los clientes ordenar y personalizar sus comidas directamente desde sus asientos, mientras reduce los tiempos de espera y mejora la experiencia gastronómica en general.

Índice

INTRODUCCIÓN

Objetivo

La empresa Zur Restaurant, S.L, un restaurante, ha expresado la necesidad de mejorar la gestión de sus pedidos y reducir el tiempo que sus clientes esperan para hacer sus pedidos. Por lo tanto, la empresa BioTechSiLogy Ti, S.L ha propuesto una solución informática en forma de una mesa interactiva que permitirá a los clientes realizar sus pedidos desde la comodidad de su mesa sin tener que esperar en la fila de la caja.

Justificación

La mesa interactiva no solo mejorará la eficiencia del proceso de pedido, sino que también mejorará la experiencia del cliente al proporcionar una interfaz más interactiva y fácil de usar. Además, la solución informática permitirá a Zur Restaurant, S.L recopilar datos sobre los pedidos y los patrones de compra de los clientes, lo que les permitirá tomar decisiones informadas sobre cómo mejorar aún más su negocio.

En términos de funcionalidad, la mesa interactiva contará con una pantalla táctil que mostrará el menú completo del restaurante y permitirá a los clientes realizar y personalizar sus pedidos. Los pedidos serán enviados automáticamente al personal de cocina del restaurante para su preparación. La mesa interactiva también permitirá a los clientes realizar pagos directamente desde la mesa, lo que agilizará el proceso de pago y reducirá la cantidad de tiempo que los clientes deben pasar en la fila de la caja.

En resumen, la implementación de una mesa interactiva para la gestión de pedidos en el restaurante de la empresa Zur Restaurant, S.L mejorará la eficiencia del proceso de pedido, mejorará la experiencia del cliente y proporcionará a la empresa Zur Restaurant, S.L información valiosa sobre los patrones de compra de los clientes.

Análisis de los existente

El nivel de innovación de este proyecto podría considerarse alto, ya que la implementación de una mesa interactiva para realizar pedidos en un restaurante es una solución novedosa y no muy común en el mercado actual. Aunque existen aplicaciones móviles para realizar pedidos en restaurantes, la incorporación de una pantalla táctil en la mesa misma es un enfoque diferente y creativo para mejorar la experiencia del cliente y la eficiencia del negocio.

Algunas aplicaciones existentes en el mercado que podrían compararse con este proyecto son las aplicaciones móviles de pedidos de restaurantes y plataformas de entrega de alimentos, como Uber Eats y Just Eats. Sin embargo, la implementación de una mesa interactiva en el restaurante proporciona una experiencia de cliente única y personalizada que difiere de estas aplicaciones.

Es cierto que algunas empresas de comida rápida ya han incorporado mesas o paneles interactivos en sus establecimientos, especialmente en países como Estados Unidos y algunos países de Europa. Sin embargo, este tipo de tecnología aún no se encuentra ampliamente

implementada en todo el mercado de restaurantes, por lo que todavía representa una solución innovadora en muchas regiones del mundo.

Además, el proyecto en sí puede tener un nivel de innovación en función de las funcionalidades específicas que se incorporen en el sistema de mesa interactiva. Por ejemplo, si se incluyen opciones para personalizar los pedidos de los clientes, integraciones con sistemas de pago móvil, o incluso integraciones con servicios de entrega a domicilio, esto podría diferenciar el proyecto de otras soluciones similares en el mercado. En resumen, aunque algunas empresas ya han implementado este tipo de tecnología, todavía hay margen para la innovación y mejora en esta área.

NECESIDADES EMPRESARIALES PARA EL DESARROLLO

Recursos humanos

Trabajadores de la empresa

Como actualmente es nuestro primer proyecto, creo que no es necesario tener mucho personal, por el momento la plantilla estará compuesta por:

- Socio / Administrador / Programador de la empresa.
- Programador Junior.

En caso de necesitar algún apoyo puntual podríamos contratar los servicios de algún programador freelance.

Contratos de trabajo a realizar

El programador junior tendrá un contrato laboral a tiempo completo, a 40 horas semanales.

En el caso de necesitar contratar un programador freelance, se puede establecer un contrato de prestación de servicios, donde se especifican las condiciones de trabajo, el alcance del proyecto, la remuneración, entre otros términos relevantes.

Obligaciones en materia de Seguridad Social

En el caso del programador junior con un contrato laboral a tiempo completo, el empleador está obligado a realizar los siguientes aportes y contribuciones:

- Cotizar a la Seguridad Social: el empleador debe realizar los aportes correspondientes a la Seguridad Social, que incluye la cotización a la seguridad social para la salud, pensiones y prestaciones de desempleo.
- Aportar al fondo de garantía salarial (FOGASA): el empleador debe realizar los aportes correspondientes al Fondo de Garantía Salarial, que tiene como objetivo garantizar el pago de salarios e indemnizaciones a los trabajadores en caso de insolvencia o quiebra del empleador.
- Pagar las prestaciones sociales: el empleador está obligado a pagar las prestaciones sociales correspondientes, que incluyen la paga extra, las vacaciones, el salario por enfermedad y el salario por maternidad, entre otros.

En el caso del programador freelance con un contrato de prestación de servicios, el contratista o cliente está obligado a realizar los siguientes aportes y contribuciones:

- Cotizar a la Seguridad Social: el contratista o cliente debe realizar los aportes correspondientes a la Seguridad Social, que incluye la cotización a la seguridad social para la salud, pensiones y prestaciones de desempleo.
- No existen prestaciones sociales: en este caso, no hay pago de prestaciones sociales como vacaciones, paga extra, salario por enfermedad, entre otros, ya que se trata de un contrato de prestación de servicios.

En resumen, tanto en el caso del programador junior con un contrato laboral a tiempo completo como en el caso del programador freelance con un contrato de prestación de servicios, existen obligaciones en materia de seguridad social que deben cumplirse en función del tipo de contrato y del rol que desempeña cada parte en la relación laboral.

Prevención de riesgos laborales

Para escribir un plan de riesgos laborales, me basaré en el Documento Único de Evaluación de Riesgos Laborales (DUERL), que es un documento en el que se identifican y evalúan los riesgos laborales presentes en la empresa, con el objetivo de establecer medidas preventivas que permitan minimizar o eliminar los riesgos identificados.

El DUERL es una herramienta obligatoria que deben tener todas las empresas en España, y es un documento dinámico que debe actualizarse y revisarse periódicamente para garantizar que los riesgos laborales se mantengan bajo control. Además, el DUERL debe estar disponible para los trabajadores y ser comunicado a las autoridades competentes, en caso de que se les solicite.

Para elaborar un plan de riesgos laborales, se deben identificar los riesgos laborales presentes en la empresa, evaluar su probabilidad de ocurrencia y su impacto potencial en la salud y seguridad de los trabajadores, establecer medidas preventivas y de control que permitan minimizar o eliminar los riesgos, y definir los procedimientos de emergencia en caso de que se produzca un accidente o incidente.

Riesgos laborales

En el caso de un programador junior, los riesgos laborales más comunes pueden ser:

- Fatiga visual: pasar largas horas frente a la pantalla puede causar fatiga ocular, visión borrosa, dolor de cabeza y otros problemas relacionados con los ojos.
- Lesiones por movimientos repetitivos: al estar sentado por largas horas, pueden surgir problemas como el síndrome del túnel carpiano o lesiones en los músculos y articulaciones, debido a la realización de movimientos repetitivos en el teclado y el mouse.
- Estrés laboral: la presión para cumplir plazos, trabajar en proyectos complejos y otras responsabilidades laborales pueden generar niveles de estrés elevados.

Medidas preventivas

A continuación, se presentan algunas medidas preventivas que se pueden tomar para minimizar estos riesgos laborales:

- Fatiga visual: tomar descansos regulares y realizar ejercicios para los ojos, ajustar la iluminación adecuada de la pantalla, utilizar filtros de pantalla o antirreflejos, utilizar la regla 20-20-20 (descansar la vista durante 20 segundos mirando a 20 pies de distancia cada 20 minutos).
- Lesiones por movimientos repetitivos: utilizar teclados ergonómicos, ratones ergonómicos, mantener una postura correcta, tomar descansos regulares y hacer estiramientos.
- Estrés laboral: fomentar una cultura de bienestar en la empresa, establecer objetivos realistas y alcanzables, proporcionar apoyo emocional y mental, fomentar una comunicación abierta y honesta.

Estas son solo algunas medidas preventivas básicas que se pueden tomar para minimizar los riesgos laborales en un entorno de trabajo para un programador junior. Es importante recordar que la prevención de riesgos laborales debe ser una prioridad en cualquier lugar de

trabajo y que los trabajadores deben estar capacitados y sensibilizados para garantizar un entorno de trabajo seguro y saludable.

Iniciativa emprendedora

Forma jurídica

La forma jurídica escogida es una Sociedad Limitada de Formación Sucesiva, aunque he barajado otras opciones como la Sociedad Limitada, o hacerme Freelance.

La principal ventaja de la Sociedad Limitada de Formación Sucesiva es que no es necesario aportar los 3.000 euros de capital inicial que se requiere para dar de alta una Sociedad Limitada.

En cuanto a los inconvenientes, el principal problema es que, acabado el ejercicio fiscal, hay que tener siempre en mente destinar los beneficios a conseguir el capital mínimo, quedando otras prioridades en un segundo plano.

Yo seré el único socio y administrador de la empresa.

Trámites administrativos

Los trámites administrativos a realizar son:

- Certificación negativa de denominación social: antes de constituir la sociedad, es necesario solicitar en el Registro Mercantil Central la certificación negativa de denominación social, para comprobar que el nombre elegido para la sociedad no está ya registrado.
- Solicitud del número de identificación fiscal provisional, el cual debemos solicitar en la Agencia Tributaria.
- Escritura pública de constitución: una vez obtenida la certificación negativa de denominación social, se debe formalizar la escritura pública de constitución ante un notario. En este documento se incluyen los estatutos de la sociedad y se establece el capital social y la distribución de participaciones.
- Liquidación del Impuesto de Transmisiones Patrimoniales y Actos Jurídicos Documentados: la constitución de una sociedad conlleva el pago del Impuesto de Transmisiones Patrimoniales y Actos Jurídicos Documentados. El importe a pagar dependerá del capital social de la sociedad.
- Inscripción en el Registro Mercantil: una vez formalizada la escritura pública de constitución, se debe inscribir la sociedad en el Registro Mercantil correspondiente al territorio donde se vaya a desarrollar la actividad.
- Obtención del Número de Identificación Fiscal (NIF) definitivo: tras la inscripción en el Registro Mercantil, se debe solicitar el NIF en la Agencia Tributaria.
- Alta en la Seguridad Social: la Sociedad Limitada de Formación Sucesiva también deberá darse de alta en la Seguridad Social para poder contratar a trabajadores.
- Obtención de licencias y permisos: según la actividad a desarrollar, puede ser necesario obtener licencias y permisos adicionales, dependiendo del sector y la ubicación geográfica.

REQUISITOS FUNCIONALES DE LA APLICACIÓN

La aplicación que BioTechSiLogy Ti desea desarrollar tiene como objetivo principal resolver el problema de la gestión de pedidos en un restaurante mediante una mesa interactiva. Como propuesta inicial, se planteó el desarrollo de dos aplicaciones: una para la gestión de los productos y otra para realizar los pedidos a través de la mesa interactiva.

Si trasladásemos este proyecto a un entorno real, la aplicación de la mesa debería conectarse con el TPV (Terminal Punto de Venta) del restaurante, el cual recibiría los pedidos. Sin embargo, dado que no disponemos de un TPV real para el desarrollo del proyecto, se ha creado una tercera aplicación muy básica, con la única finalidad de simular un entorno real.

Aplicación de Gestión

La funcionalidad de la aplicación de gestión incluirá:

- Inicio de sesión con su nombre de usuario y contraseña.
- Gestión de usuarios, opción de crear, eliminar o modificar usuarios.
- Gestión de productos, opción de crear, eliminar o modificar productos.
- Visualización de pedidos.
- Opción de filtrado o búsqueda.
- Exportar todos los pedidos, para manipular los datos.

Aplicación Mesa Interactiva

La funcionalidad de la aplicación de la mesa interactiva incluirá:

- Interfaz de usuario amigable: la mesa interactiva contará con una pantalla táctil de fácil manejo para que los clientes puedan realizar sus pedidos de manera intuitiva.
- Selección de productos: la aplicación permitirá a los clientes visualizar los productos ofrecidos por el restaurante y seleccionar aquellos que deseen pedir.
- Visualización de precios: la aplicación mostrará el precio total del pedido, incluyendo impuestos y cargos adicionales.
- Envío de pedidos a cocina: una vez que el cliente haya realizado su pedido, la aplicación lo enviará automáticamente a la cocina del restaurante para su preparación.
- Gestión de pagos: los clientes podrán realizar el pago de su pedido directamente desde la mesa interactiva, utilizando tarjetas de crédito o débito, paypal o pago al camarero. (*Obviamente es una simulación no se ha implementado ningún sistema de pagos real*).

La aplicación está dirigida a clientes de restaurantes que buscan una experiencia moderna y fácil de usar en la gestión de sus pedidos. Además, el personal del restaurante también se beneficiará de la aplicación al permitir una gestión más eficiente de los pedidos y reducir el tiempo de espera del cliente.

Aplicación Móvil

Como he mencionado anteriormente, esta aplicación no se incluyó en la propuesta inicial, pero he decidido añadirla para que el proyecto sea más completo.

En el momento de realizar la aplicación, no contaba con los conocimientos necesarios para desarrollarla, ya que no había seguido la asignatura durante el curso. Por lo tanto, prácticamente he tenido que empezar desde cero y repasar todo el temario.

Aunque no era necesario, dado que se trata de una aplicación muy simple diseñada únicamente para visualizar los pedidos entrantes, he decidido utilizar todas las herramientas proporcionadas como una oportunidad de repaso y aprendizaje de la asignatura.

La funcionalidad de la aplicación móvil:

- Visualización de todos los pedidos.
- Filtrado de los pedidos por Estado (Pagado, Pagar en Mesa, Todos, Solicitado, Entregado, Cancelado)
- Visualización del pedido completo para poder ejecutarlo.
- Opción de cambio de estado del pedido.

Api Rest en servidor LAMP Azure

Implementación de una API REST en un servidor LAMP en Azure: Para garantizar la comunicación y el intercambio de datos entre las diferentes aplicaciones, se desarrollará una API REST en un entorno LAMP (Linux, Apache, MySQL, PHP) en Azure. Esta API permitirá la interacción entre la aplicación de gestión, la aplicación de la mesa interactiva y la aplicación móvil, facilitando la sincronización de los pedidos, la gestión de usuarios y productos, y la actualización de estados de los pedidos en tiempo real.

Al implementar esta API REST en el servidor LAMP en Azure, se establecerán los siguientes requisitos funcionales:

- Definición de endpoints: Se establecerán endpoints claros y consistentes para cada función relevante de la API, como la gestión de usuarios, productos y pedidos.
- Manipulación de datos: La API permitirá la creación, lectura, actualización y eliminación de usuarios, productos y pedidos, manteniendo la integridad de los datos y aplicando validaciones necesarias.
- Consultas y filtros: Se proporcionarán métodos de consulta y filtrado de datos para obtener información específica, como listar pedidos por estado o buscar productos por nombre.
- Actualización en tiempo real: La API será capaz de recibir y procesar eventos en tiempo real, como la actualización del estado de un pedido, y notificar a las aplicaciones conectadas sobre dichos cambios para mantener la información sincronizada en todos los dispositivos.
- Manejo de errores y excepciones: Se implementará un manejo adecuado de errores y excepciones para asegurar la robustez y estabilidad de la API, proporcionando respuestas claras y significativas en caso de fallos o situaciones inesperadas.

Implementando estos requisitos funcionales en la API REST del servidor LAMP en Azure, se logrará una integración efectiva y fluida entre todas las aplicaciones del proyecto, permitiendo una gestión eficiente de los pedidos, una experiencia intuitiva para los clientes y un aumento en la productividad del personal del restaurante.

DISEÑO Y ANÁLISIS

Diagrama de la arquitectura

En la siguiente imagen podemos ver de forma gráfica el diseño de la arquitectura:

BioTechSiLogy Ty desarrolla las aplicaciones:

- Aplicación de Gestión. (Escritorio)
- Aplicación Interactiva (Mesa)
- Aplicación Trabajadores (App Móvil)
- ApiRest (Alojada en Servidor LAMP en Azure)

BioTechSiLogy Ty mantiene y gestiona el servidor creado en Azure, únicamente proporcionando acceso al gerente del restaurante.

El gestor del Restaurante tiene acceso a la aplicación de Escritorio (Aplicación de Gestión), mediante la cual gestiona los productos.

El cliente o comensal del restaurante hará uso de la mesa interactiva, en la cual está instalada la aplicación interactiva.

Trabajadores Restaurante harán uso de la app móvil, para recibir los pedidos realizados por los clientes.

Por último, he añadido una aplicación desarrollada en pySpark, que será encargada a un científico de datos freelance para manipular y visualizar los datos.

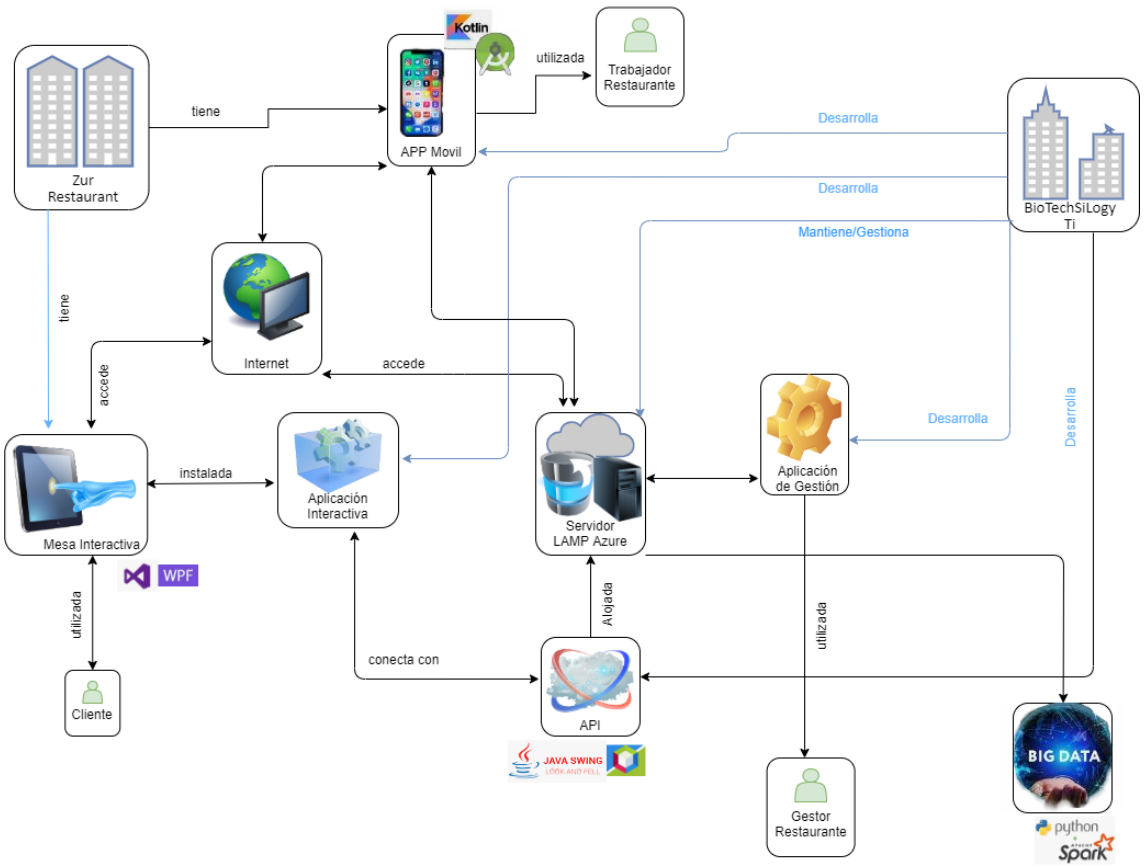


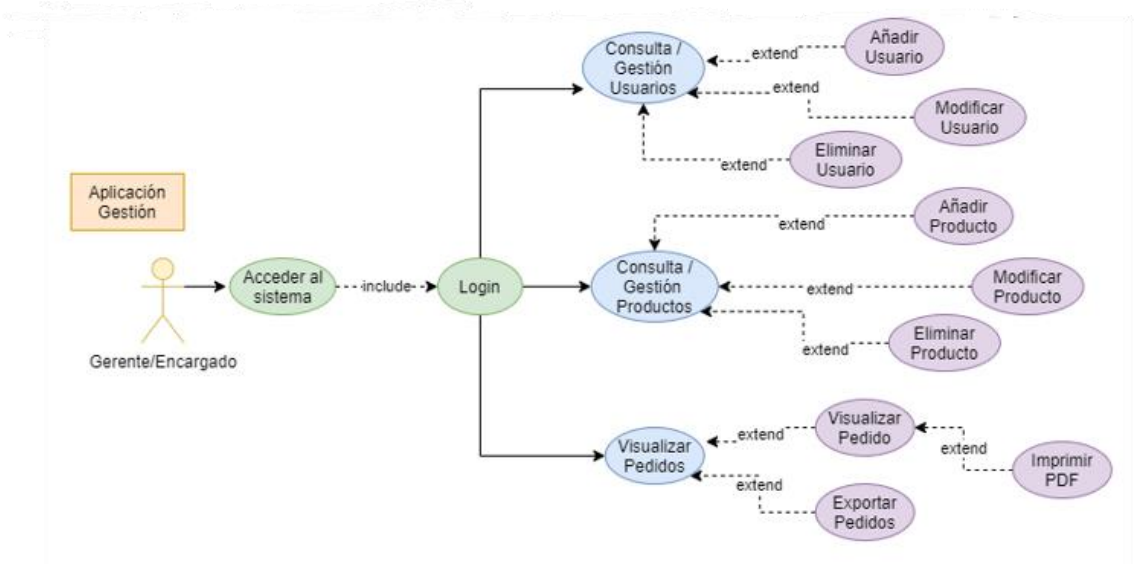
Diagrama de casos de usos

Aplicación de Gestión

La aplicación de gestión es usada por el gerente o encargado del restaurante, para gestionar los usuarios, productos y visualizar los pedidos realizados.

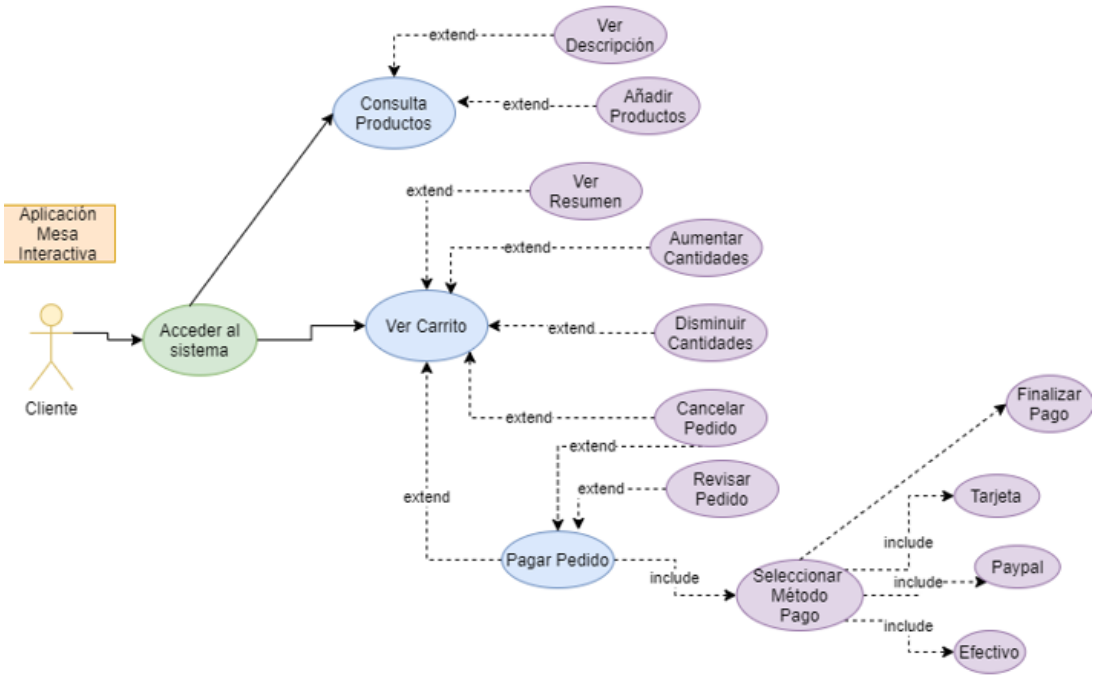
Accedemos al sistema mediante usuario y contraseña, una vez realizado el login, podemos ver los usuarios, los productos, o los pedidos que se han realizado.

Cada una de estas vistas, tiene distintas opciones para su gestión.



Aplicación Mesa Interactiva

La aplicación de la mesa interactiva, está siempre activa, no es necesario realizar autentificación para acceder al menú principal.



Una vez hemos accedido al sistema, tenemos la opción de consultar los productos de cada categoría. Podemos ver una descripción más detallada de cada producto o añadir estos al carrito.

También podemos ver el carrito para ver que productos han sido añadidos, donde:

- Se muestra un resumen de los productos, y el precio total.
- Podemos añadir la cantidad de un producto añadido.
- Podemos disminuir la cantidad de un producto añadido.
- Podemos cancelar un pedido.
- Pagar el pedido.

Si seleccionamos la opción de pagar el pedido, nos llevara a otra pantalla, donde podemos:

- Revisar el pedido.
- Cancelar el pedido
- Seleccionar método de pago (tarjeta/paypal/efectivo)
  (*Estos métodos de pago, son de prueba, no es necesario ningún dato real, ni realiza ninguna comprobación)*

Y finalizar el pago.

Aplicación móvil

La aplicación móvil está siempre activa, no es necesario realizar autentificación para acceder al menú principal.

Una vez hemos accedido al sistema, tenemos la opción de consultar los pedidos. Podemos ver el detalle del pedido y si fuera necesario modificar su estado.

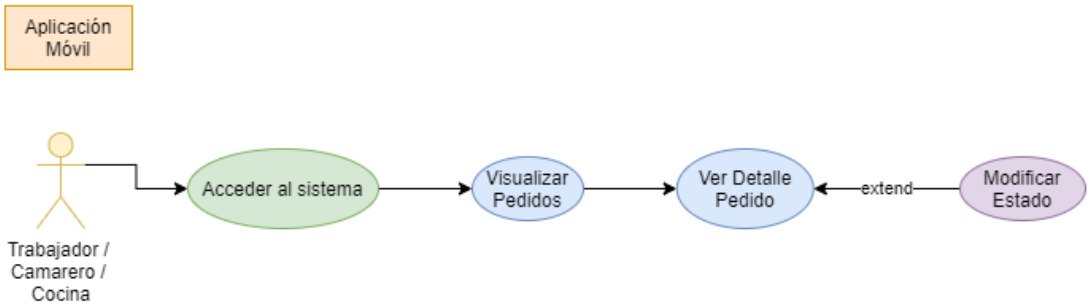
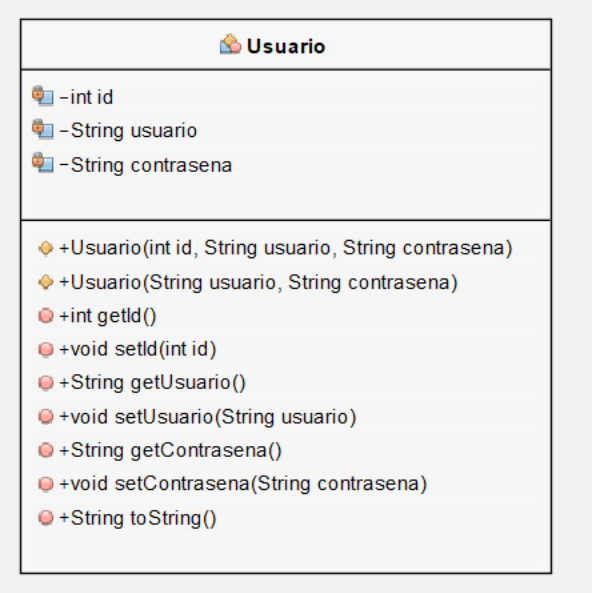
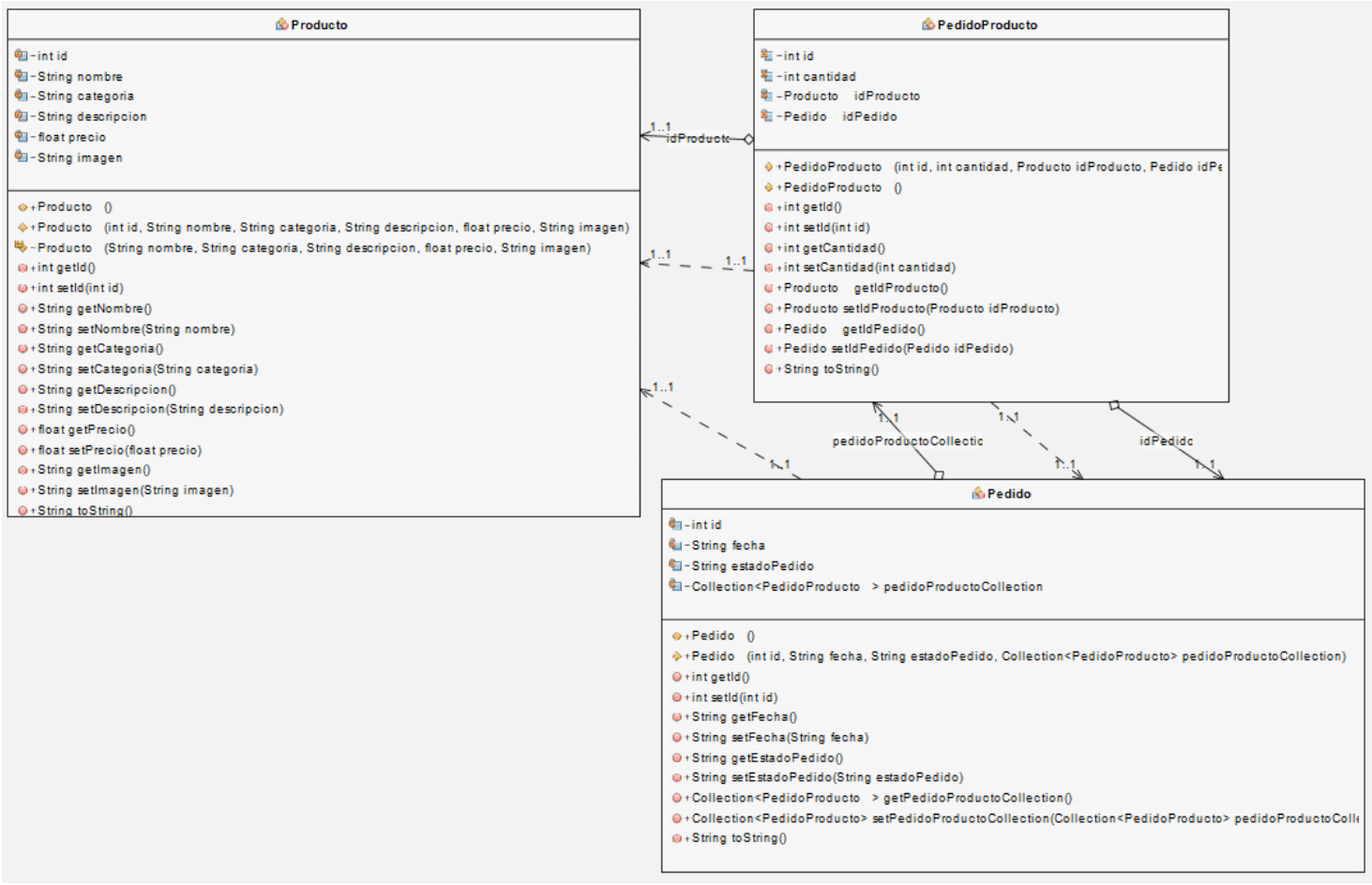


Diagrama de clases

Para extraer el diagrama de clases, he usado el plugin easyUML para NetBeans.

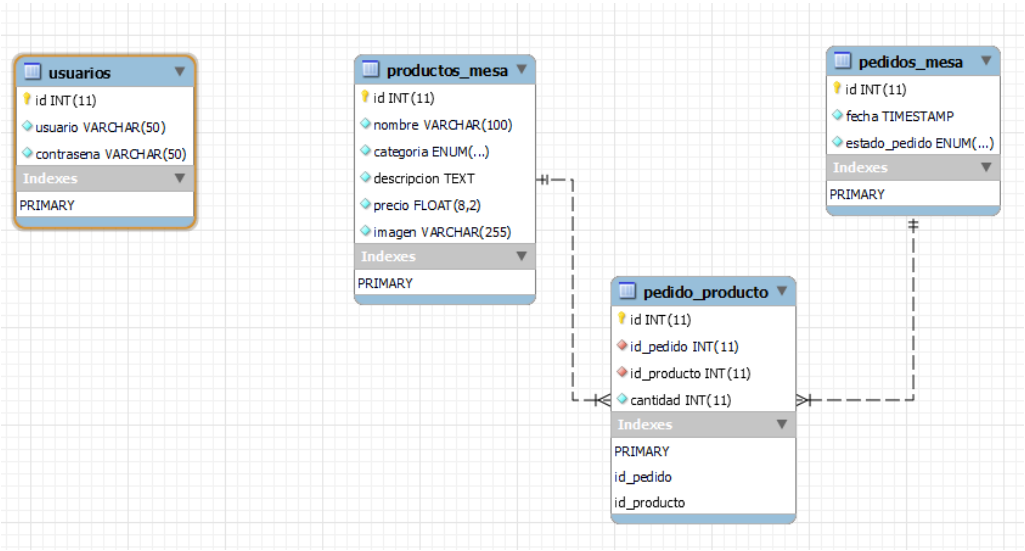




Podemos observar la relación existente entre las clases, sus atributos y métodos.

Diseño de datos (Base de datos relacional)

El diseño de la base de datos es bastante simple, únicamente este compuesto por 4 tablas.



La tabla usuarios, independiente del resto, únicamente es utilizada para la gestión de los usuarios que pueden acceder al sistema, o aplicación de gestión.

La tabla "productos_mesa" contiene información sobre los productos que se pueden pedir en la mesa. Tiene las siguientes columnas:

- id: identificador único del producto (clave primaria)
- nombre: nombre del producto
- categoria: categoría del producto (se especifica mediante una enumeración)
- descripcion: descripción del producto
- precio: precio del producto
- imagen: nombre del archivo de imagen del producto

La tabla "pedidos_mesa" contiene información sobre los pedidos realizados en la mesa. Tiene las siguientes columnas:

- id: identificador único del pedido (clave primaria)
- fecha: fecha y hora en la que se realizó el pedido (se especifica mediante un valor de fecha y hora)
- estado_pedido: estado actual del pedido (se especifica mediante una enumeración)

La tabla "pedido_producto" es una tabla de relación que relaciona los productos de la tabla "productos_mesa" con los pedidos de la tabla "pedidos_mesa". Tiene las siguientes columnas:

- id: identificador único de la relación (clave primaria)
- id_pedido: identificador del pedido al que se refiere la relación (clave ajena que hace referencia a la columna "id" de la tabla "pedidos_mesa")
- id_producto: identificador del producto al que se refiere la relación (clave ajena que hace referencia a la columna "id" de la tabla "productos_mesa")
- cantidad: cantidad de productos pedidos en el pedido

Documentación Api

Métodos del Api Rest.

Accesible desde http://servidormesa.westeurope.cloudapp.azure.com:8080/zurmesaapi/

API Rest Zur Mesa Interactiva - MySQL Server

Esta API Rest permite la gestión de usuarios, productos y pedidos de la aplicación Zur Mesa Interactiva.

Database	bdMesa
Tables	usuarios / productos_mesa / pedidos_mesa
Content-Type	application/json
Accept	application/json

USUARIOS

MÉTODO	PATH
GET	bdmesa/usuarios
GET	/bdMesa/usuarios/{ID} bdmesa/usuarios/3
POST	bdmesa/usuarios
PUT	bdmesa/usuarios/update/{id}
DELETE	bdmesa/usuarios/delete/{id}

PRODUCTOS

MÉTODO	PATH
GET	bdmesa/productos
GET	/bdmesa/productos/{ID} bdmesa/productos/3
POST	bdmesa/productos
PUT	bdmesa/productos/update
DELETE	bdmesa/productos/delete/{id}

PEDIDOS

MÉTODO	PATH
GET	bdmesa/pedidos
GET	/bdmesa/pedidos/{ID} bdmesa/pedidos/1
POST	bdmesa/pedidos
PUT	bdmesa/pedidos/update
DELETE	bdmesa/pedidos/delete/{id}

PRODUCTOS DE PEDIDOS

MÉTODO	PATH
GET	bdmesa/pedido_producto
GET	/bdmesa/pedido_producto/{ID} bdmesa/pedido_producto/1
POST	bdmesa/pedido_producto
PUT	bdmesa/pedido_producto/update
DELETE	bdmesa/pedido_producto/delete/{id}

CODIFICACIÓN

Lenguajes de programación

Para el desarrollo de este proyecto he utilizado varios lenguajes de programación.

· Aplicación de escritorio o gestión (backend): Es una de las aplicaciones que utilizará el gerente o encargado del establecimiento, desarrollada en Java Swing.

· Aplicación de la mesa interactiva: Utilizada por el usuario para solicitar el pedido, desarrollada en Microsoft .NET / WPF / C#.

· Aplicación móvil: Utilizada por el personal del establecimiento para recibir los pedidos, desarrollada en Kotlin.

· Api Rest, desarrollada en Java Web Application.

· Base de datos relacional (SQL) alojada en servidor LAMP de Azure.

Se ha elegido cada lenguaje de programación con el único fin de tratar de agrupar o utilizar todos los empleados durante el curso.

Por poner un ejemplo, podría haber desarrollado la aplicación de escritorio/gestión en .NET, pero a pesar de no verlo en profundidad durante el ciclo, decidí hacerlo en Swing, para no repetir ese lenguaje.

Herramientas utilizadas, entornos de desarrollo, y plataformas

Para el desarrollo del proyecto he empleado diversas herramientas y entornos, que enumero brevemente a continuación:

- NetBeans (Java): Desarrollo de aplicación de escritorio, y API Rest.
- Visual Studio 2019 (.NET/WPF/C#): Desarrollo de aplicación de usuario.
- Android Studio (Kotlin): Desarrollo de aplicación de Empleados, para recibir pedidos.
- MySQLWorkbench: Gestión y creación de la Base de datos:
- DrawIO: Desarrollo de diagramas.
- Servidor LAMP Azure: Despliegue de Base de datos y Api Rest.
- Blob Storage Azure: Almacenamiento de imágenes e informes.

Todo esto permitió crear un sistema eficiente y escalable que cumplió con los requisitos del proyecto.

Otras aplicaciones utilizadas:

- HelpNDoc: Desarrollo de Manuales.
- Launch4j: Generar .exe a partir de .jar (Java).
- Inno Setup Compiler: Crear Instalador de aplicación java.
- Display32.exe: Giro Pantalla.

Código con los aspectos relevantes de la implementación.

En esta sección, me enfocaré en mencionar las partes de código y la arquitectura más relevante que fueron implementadas en las distintas aplicaciones.

Aplicación Mesa Interactiva

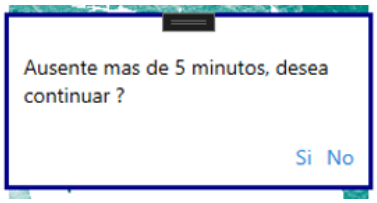
Temporizadores/Inactividad

He creado o utilizado varios temporizadores, para gestionar diferentes funciones de la aplicación:

- Controlar el giro de la pantalla
- Controlar la inactividad
- Aceptar la inactividad
- Cancelar el pago/pedido en caso de inactividad

Vamos a ver uno de los ejemplos anteriores en más profundidad.

Control de inactividad; en este caso si la aplicación no detecta ninguna actividad durante 5 minutos, nos mostrará un mensaje en la pantalla para comprobar si seguimos ahí, debemos confirmarlo, porque, pasados 20 segundos se cerrará solo, volviendo a la pantalla de inicio.




```
private DispatcherTimer inactividadTemp = new DispatcherTimer();
```

Creamos una nueva instancia de la clase DispatcherTimer y la asignamos a la variable inactividadTemp. DispatcherTimer es una clase en C# que proporciona un temporizador que se puede utilizar para programar tareas en un intervalo de tiempo determinado. En este caso, la variable inactividadTemp se utiliza probablemente para controlar la inactividad del usuario en la aplicación y para desencadenar acciones específicas después de cierto tiempo de inactividad.

```
private void TemporizadorInactividad() //Temporizador de inactividad en
pantalla principal
{
	if (!inactividadTemp.IsEnabled)
	{
		Console.WriteLine("Inactividad Temporizador Iniciado");
		inactividadTemp.Interval = TimeSpan.FromMinutes(5); //Ponerlo
		a 5 y 0.1 pruebas
		inactividadTemp.Tick += Inactividad_Tick;
		ResetTemporizadorInactividad();
		this.PreviewMouseMove += (s, e) => {
		ResetTemporizadorInactividad(); };
		this.PreviewTouchDown += (s1, ev1) => {
		ResetTemporizadorInactividad(); };
	}
}
```

La función "TemporizadorInactividad()" es un método privado que se encarga de configurar y controlar el temporizador de inactividad en la pantalla principal de la aplicación.

El primer bloque de código comprueba si el temporizador de inactividad no está activo y, en caso afirmativo, configura el intervalo del temporizador en 5 minutos (o 0.1 minutos para

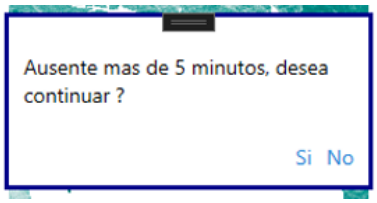
pruebas), y suscribe el método "Inactividad_Tick" al evento Tick del temporizador. Este método será llamado cuando el temporizador de inactividad expire.

El segundo bloque de código establece los eventos "PreviewMouseMove" y "PreviewTouchDown" del objeto actual para que cuando se produzca un evento de movimiento del ratón o toque en la pantalla, se restablezca el temporizador de inactividad. Esto significa que cada vez que el usuario interactúa con la pantalla, el temporizador de inactividad se reinicia y vuelve a contar desde cero.

```
private void ResetTemporizadorInactividad()
{
	inactividadTemp.Stop();
	inactividadTemp.Start();
}

private void Inactividad_Tick(object sender, EventArgs e)
{
	inactividadTempCierre.Interval = TimeSpan.FromSeconds(20); //Cerrar
	el tiempo de inactividad y vuelve a pantalla de inicio
	inactividadTempCierre.Tick += InactividadCierre_Tick;
	inactividadTempCierre.Start();
	messageBoxResult = new MiMessageBox("Cerrar sesion", "Ausente mas de
5 minutos, desea continuar ? ", "Si", "No");
	if (messageBoxResult.ShowDialog() == true)
	{
		ResetTemporizadorInactividad();
	}
	else if (messageBoxResult.DialogResult == false)
	{
		Console.WriteLine("Inactividad Temporizador Cerrado desde el
	boton");
		MainWindow pantallaPrincipal = new MainWindow();
		pantallaPrincipal.Show();
		inactividadTemp.Stop();
		this.Close();
	}
}
```

Este es un método que se ejecuta cada vez que se produce un Tick (un pulso del temporizador) en el temporizador de inactividad. En primer lugar, se establece un nuevo temporizador (inactividadTempCierre) con un intervalo de 20 segundos, que se iniciará cuando se produzca el Tick actual. Luego se crea una nueva instancia de la clase MessageBox personalizada (MiMessageBox) que muestra un mensaje preguntando si el usuario desea continuar la sesión o no después de haber estado inactivo durante más de 5 minutos.



Si el usuario hace clic en el botón "Sí", se reinicia el temporizador de inactividad (ResetTemporizadorInactividad). Si el usuario hace clic en el botón "No", se muestra un mensaje en la consola indicando que se ha cerrado el temporizador de inactividad y se crea una nueva instancia de la ventana MainWindow (pantallaPrincipal) que se muestra y la ventana actual se cierra.

```
public PantallaAñadirProductos()
{
	TemporizadorInactividad();

	…
```

En el constructor de la clase PantallaAñadirProductos llamamos al método TemporizadorInactividad()

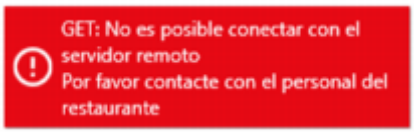
Iniciar Servidor Azure mediante el control de la excepción

Otra parte de código que me parece bastante interesante, como iniciar el servidor Azure, donde se encuentra alojado el ApiRest, en el caso de que este no esté activo y el código genere una excepción.

He probado diferentes formas de implementarlo, creando scripts en PowerShell o cmd, para tratar de autentificarme automáticamente, pero no ha sido posible. Al final opte por la siguiente implementación:

```
catch (Exception ex)
{
	VariablesGlobales.ErrorAzure = true;
	ShowMessage(_vm.ShowError, "GET", ex.Message + "\nPor favor contacte
con el personal del restaurante");
	return null;
}
```

Un bloque catch que maneja la excepción de la petición GET al ApiREst del servidor de Azure. Cuando se produce una excepción en el servidor, se establece una variable global en true para indicar que se produjo un error. A continuación, se muestra un mensaje de error en la pantalla y se indica que se debe contactar al personal del restaurante.




```
void ShowMessage(Action<string, MessageOptions> action, string name,
string message)
{
	clickCount = 0;
	MessageOptions opts = new MessageOptions
	{
		FontSize = 12,
		ShowCloseButton = false,
		FreezeOnMouseEnter = true,
		UnfreezeOnMouseLeave = true,
		NotificationClickAction = n => {
			clickCount++;
			if (clickCount >= 6)
			{
				VariablesGlobales.ErrorAzure = false;
				Console.WriteLine("clickCount "+ clickCount);
				if (!VariablesGlobales.ErrorAzure)
				{
					ProcessStartInfo startInfo1 = new
					ProcessStartInfo();
					startInfo1.FileName = "powershell.exe";
					startInfo1.UseShellExecute = false;
					startInfo1.CreateNoWindow = true;
```

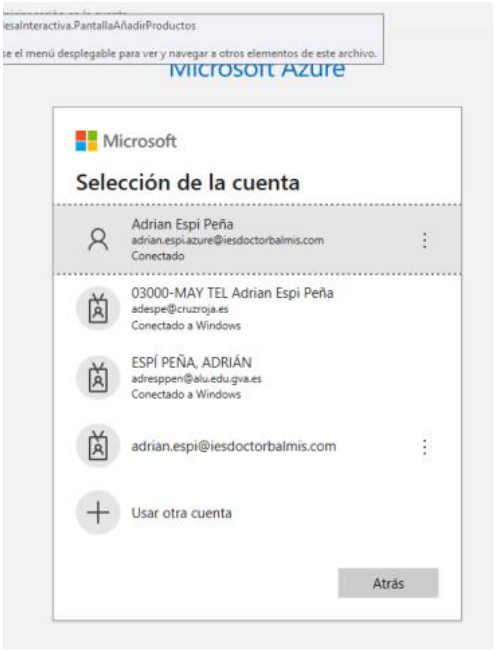
```
                        startInfo1.Arguments = "Connect-AzAccount -
                    TenantId 59744db4-988e-4f00-869b-dc1a7bf3b63e;
                    Start-AzVM -ResourceGroupName
                    servidorbdMesa_group -Name servidorbdMesa";
                        Process process1 =
                        Process.Start(startInfo1);
                        process1.WaitForExit();
                        Console.WriteLine("Iniciando Azure");
                        ShowMessage(_vm.ShowInformation,
                    "Solucinado!", "Iniciando Azure\nIntentelo de
                    nuevo en 1 minuto");
                    }
                }
            }
        };
        action($"{name}: {message}", opts);
}
```

La función ShowMessage muestra un mensaje de notificación en la interfaz de usuario. Toma una acción y un mensaje como entrada y también configura algunas opciones para el mensaje, como el tamaño de fuente, si se debe mostrar un botón de cierre y cómo se debe comportar la notificación al pasar el mouse sobre ella.

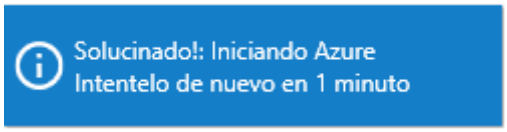
También incluye una acción que se ejecuta si el usuario hace clic en la notificación, en este caso, se cuenta el número de clics y se intenta reiniciar el servidor de Azure si se ha producido un error de conexión. Si se han producido suficientes clics, se establece la variable global ErrorAzure en false, lo que significa que se ha intentado solucionar el problema.

Luego se llama a Process.Start para iniciar una nueva instancia de PowerShell y ejecutar un comando para reiniciar el servidor.

Nos solicitará realizar login



Si se reinicia correctamente, se muestra un mensaje de notificación informativo indicando que el servidor se está iniciando y que el usuario debe esperar un minuto antes de intentar nuevamente.



Este código solo esta implementado en la carga de los productos, ya que si los productos no se han cargado no se puede continuar con el proceso de pedido.

Debemos configurar nuestra cuenta de azure previamente en el dispositivo en que se instalará la aplicación.

Giro de pantalla

Otra función interesante que hemos añadido a nuestra aplicación es el giro de pantalla, ya que, al ser una mesa, el usuario puede sentarse en cualquier de los 4 lados de esta, aunque solo hemos añadido giro para 180º.

```
private void Button_Giro(object sender, RoutedEventArgs e)
{
	_timer.Start();
	giroPantallaBtn.IsEnabled = false;
	//Obtener en que ruta estoy y moficiar ruta
	string ruta = AppDomain.CurrentDomain.BaseDirectory;
	ruta = ruta.Substring(0, ruta.Length - 10);
	ruta = ruta + "scripts";
	ProcessStartInfo startInfo1 = new ProcessStartInfo();
	startInfo1.WorkingDirectory = ruta;
	startInfo1.FileName = "cmd.exe";
	startInfo1.UseShellExecute = false; //No Abrir Cmd al hacer el giro
	startInfo1.CreateNoWindow = true;
	if (contador == 0)
	{
		startInfo1.Arguments = "/C display32 /device 1 /rotate:180";
		contador++;
	}
	else if (contador == 1)
	{
		startInfo1.Arguments = "/C display32 /device 1 /rotate:0";
		contador = 0;
	}
	Process.Start(startInfo1);
}
```

Este código corresponde a un manejador de eventos para el botón "Girar pantalla.

Al presionar el botón, se inicia un temporizador y se deshabilita el botón. Luego, se obtiene la ruta del directorio donde se encuentran los scripts necesarios para girar la pantalla y se configura la información de inicio del proceso. Dependiendo del valor de la variable contador, se especifica el comando de giro a través del argumento startInfo1.Arguments del objeto ProcessStartInfo.

Finalmente, se inicia el proceso para girar la pantalla.

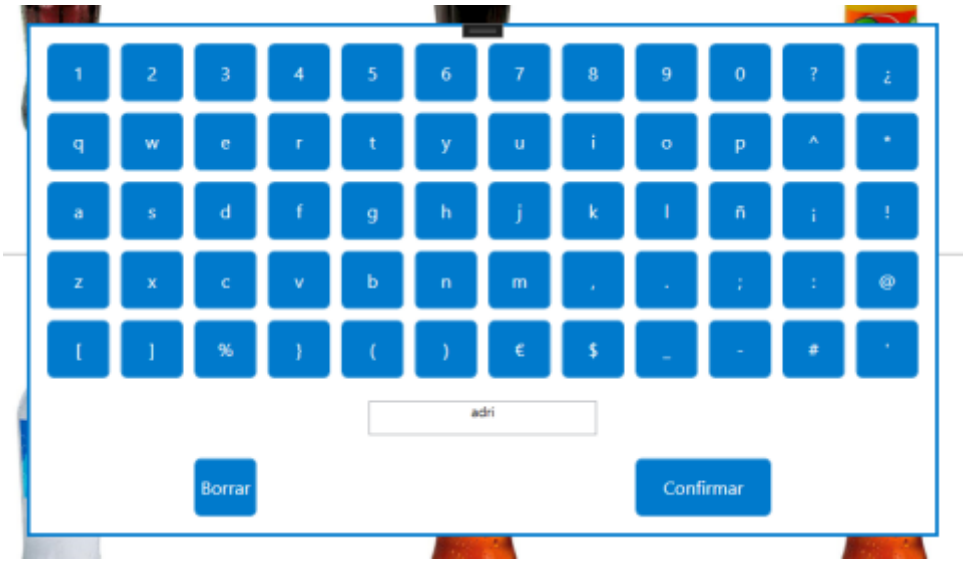
Teclado en pantalla

Aunque esta funcionabilidad pueda programar sin dificultad, considero relevante mencionarla, y el porqué de su creación.

Tras preparar la documentación y realizar las últimas pruebas del proyecto, he tratado de recrear una simulación real, usando una pantalla táctil de 7” de una Raspberry Pi.

Anteriormente había realizado todas las pruebas desde el pc, y no había contemplado la situación de realizar un pedido completo desde la pantalla táctil; y al tratar de completar los datos de la tarjeta de crédito o paypal, no tenía forma de hacerlo.

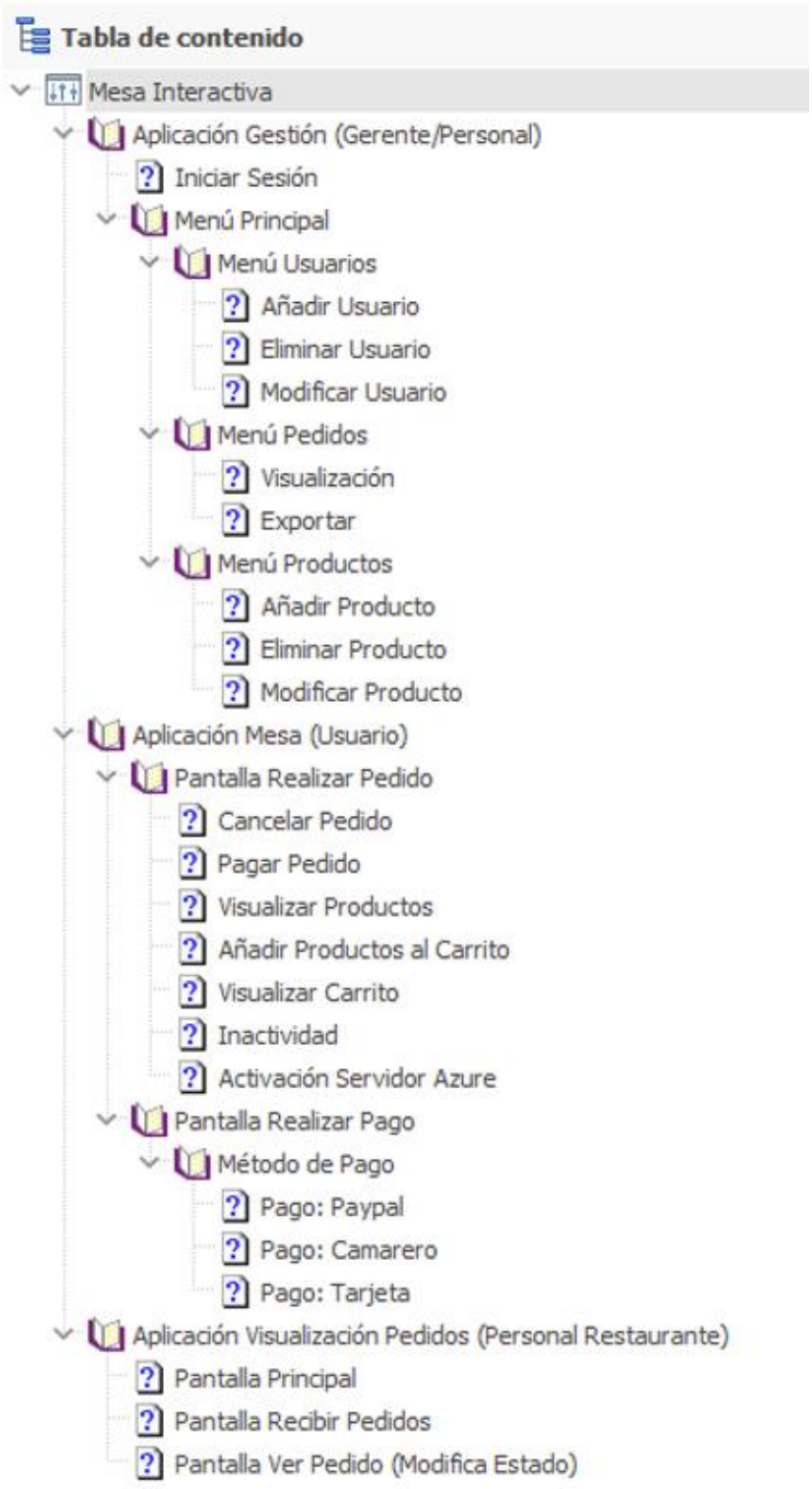
He tratado de utilizar el propio teclado virtual de Windows, sin éxito, por lo que he decido crear una nueva ventana Wpf y crear mi propio teclado.



MANUAL DE USUARIO

Se ha desarrollado un manual en HelpNDoc, de las 3 aplicaciones desarrolladas, el cual se adjuntará con la documentación aportada.

Aquí podemos ver una imagen del índice o tabla de contenido del manual;



Se ha generado en formato *.pdf y *.chm (ayuda Windows).

REQUISITOS E INSTALACIÓN

En este apartado voy a tratar de enumerar y detallar todos los componentes, ya sea hardware o software, necesarios para llevar a cabo la instalación.

Componentes o hardware

Ordenador Sobremesa

No necesariamente necesitamos un equipo muy potente, ya que nuestro único requisito es poder utilizar la aplicación de escritorio. Durante la realización del proyecto he utilizado mi ordenador personal:

- Procesador: 11th Gen Intel(R) Core (TM) i7-11370H @ 3.30GHz 3.30 GHz.
- RAM: 16,0 GB.
- Sistema Operativo: Windows 11.

Pantalla Mesa Interactiva

- Resolución Máxima: 1920*1080*3840*2160
- Pantalla táctil: 10-, punto táctil capacitiva
- Resolución: 1920*1080(3840*2160 opcional)
- Waterproof and anti-explosion glass.

Un posible ejemplo de la pantalla para la mesa interactiva, lo podemos observar en el siguiente enlace: https://spanish.alibaba.com/p-detail/4K-1600534345613.html?spm=a2700.wholesale.0.0.52076ccaAancPK

Mini Ordenador Mesa Interactiva

- Maxesla Mini PC Windows 11 Pro,
- Celeron N5105 (hasta 2,9GHz)
- 8GB DDR4
- 256GB M.2 SSD

Un posible ejemplo del mini ordenador para la mesa interactiva, lo podemos observar en el siguiente enlace:

https://www.amazon.es/Maxesla-Ordenador-Sobremesa-Bluetooth4-2-Business/dp/B0BM8TCRLT/ref=sr_1_5?adgrpid=1298523667816307&hvadid=81157817970394&hvbmt=be&hvdev=c&hvlocphy=164592&hvnetw=s&hvqmt=e&hvtargid=kwd-81157965894633%3Aloc-170&hydadcr=23171_1834351&keywords=mini+pc+barato&qid=1683701445&sr=8-5

Móvil o Tablet

Para la realización de las pruebas he utilizado mi móvil personal, que ya está bastante anticuado, por lo que los requisitos o prestaciones no son muy elevadas.

- Pantalla: 5,45 pulgadas HD+ IPS LCD Aspecto 18:9 1.440 x 720 295 píxeles por pulgada
- Procesador: Helio P22 de ocho núcleos a 2GHz GPU PowerVR GE8320

- RAM: 4GB
- Batería: 3.000 mAh
- Otros: LTE, WiFi, Bluetooth, GPS, lector de huellas

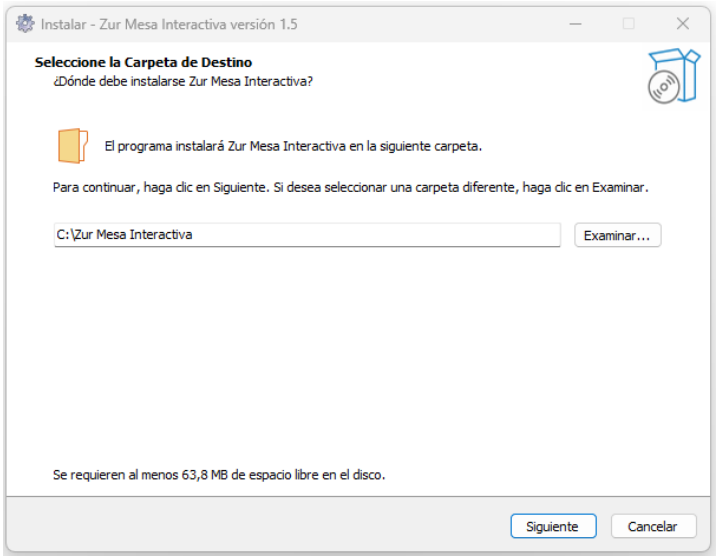
Descripción del entregable

Gráfico con la estructura de los ficheros que se entregan y su utilidad.

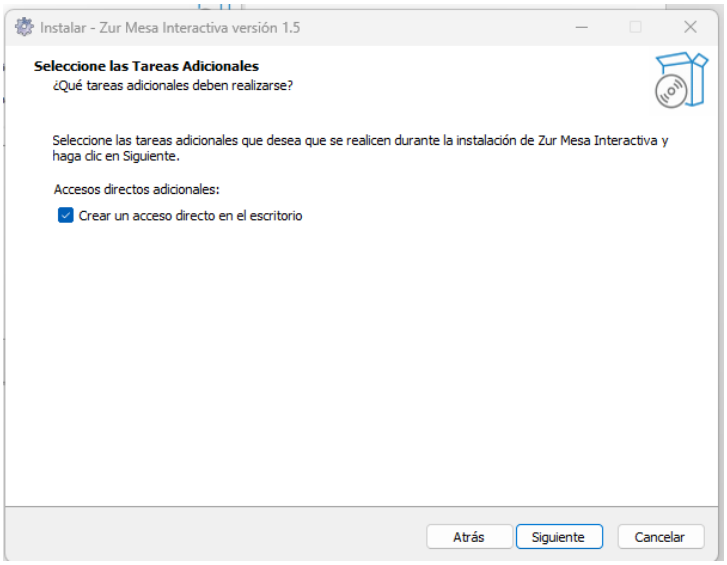
Procedimientos de instalación y prueba

Aplicación de gestión

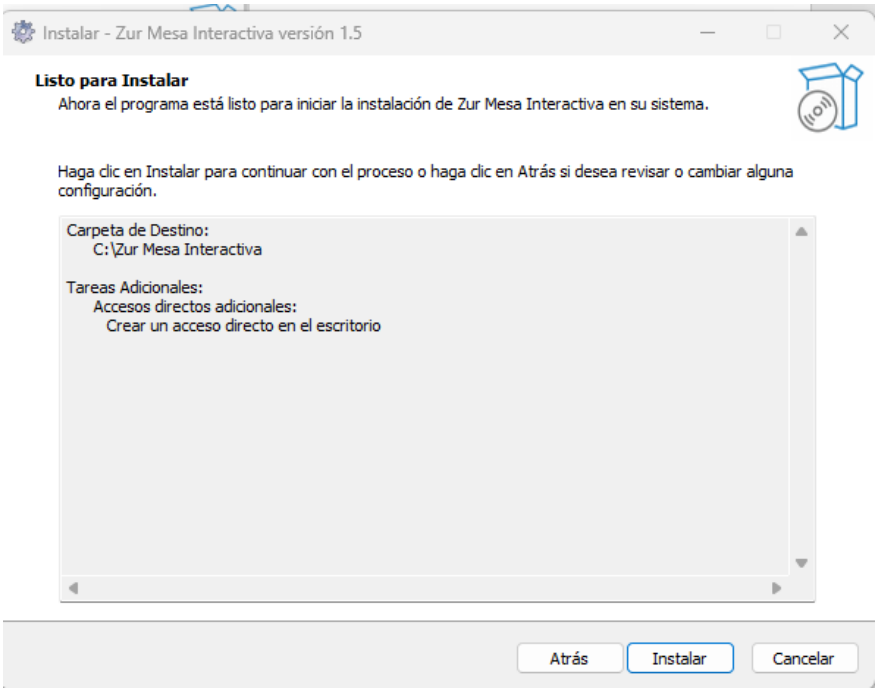
Vamos a ver los pasos necesarios para instalar la aplicación de gestión en un sistema operativo Windows 11. Ejecutamos el instalador, que hemos creado, podemos consultar el procedimiento de creación de un instalador para java, en los Anexos del proyecto.



Siguiente, y Seleccionamos Crear un acceso directo en el escritorio.



Siguiente



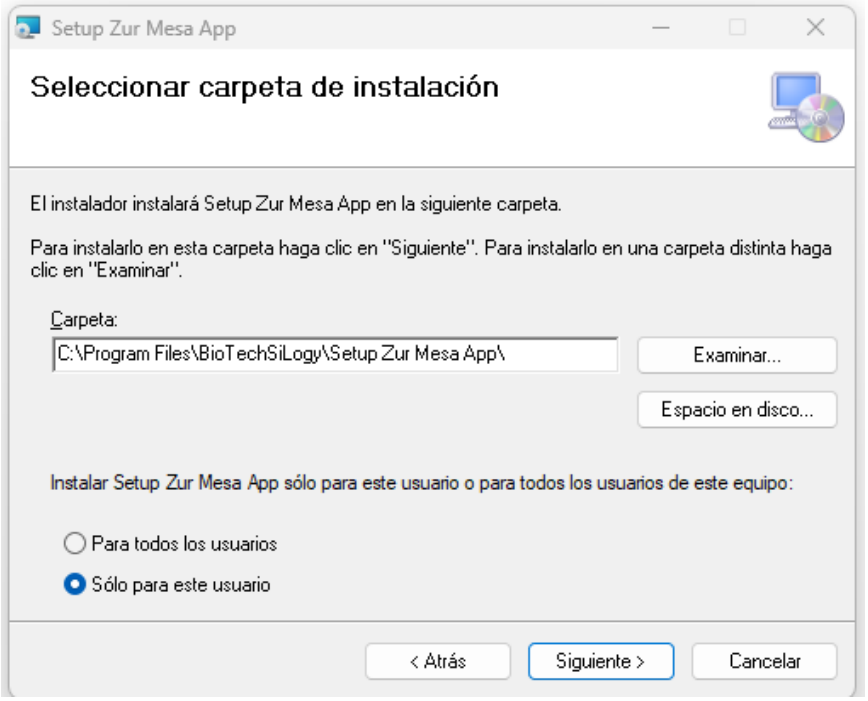
Instalar, una vez completado el proceso. Finalizar

Se genera nuestro acceso directo

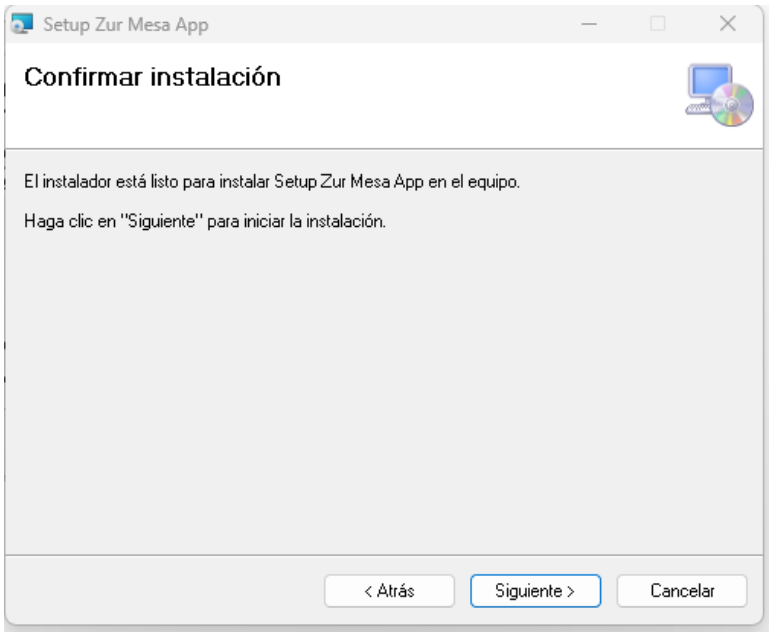
Aplicación de la mesa interactiva

Vamos a ver los pasos necesarios para instalar la aplicación de la mesa interactiva en un sistema operativo Windows 11. Ejecutamos el instalador, que hemos creado, podemos consultar el procedimiento de creación de un instalador para net/wpf, en los Anexos del proyecto.

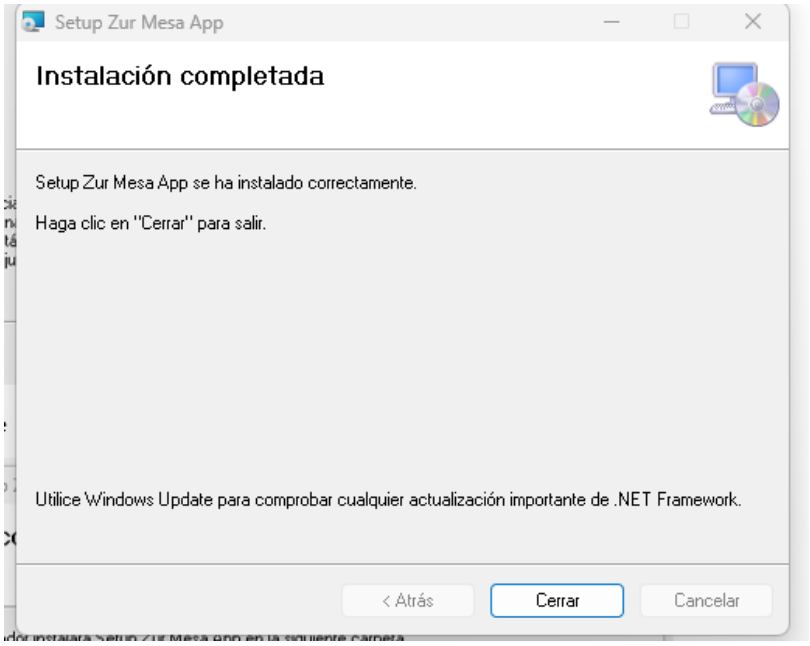
Siguiente



Siguiente



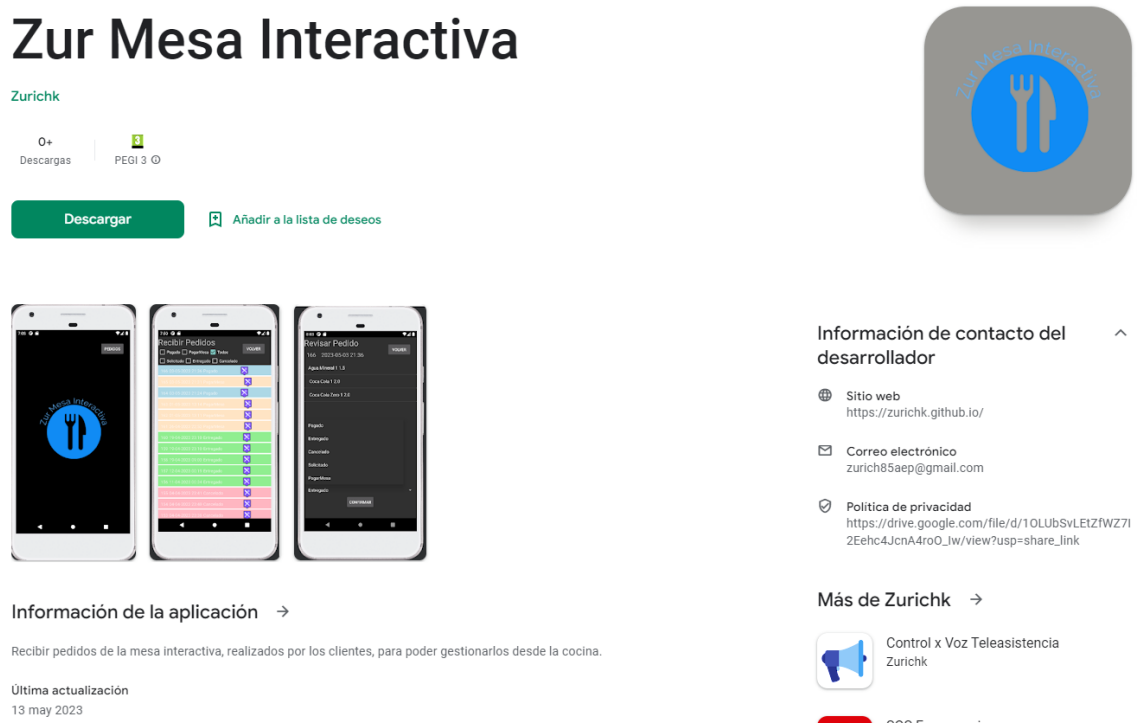
Siguiente, y se iniciar el proceso de instalación



Se genera nuestro acceso directo

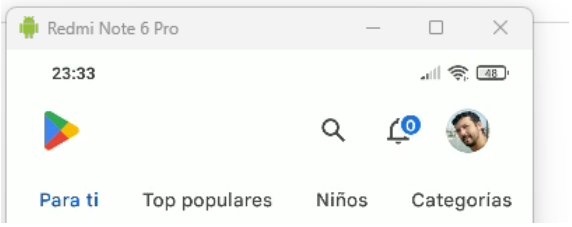
Aplicación móvil

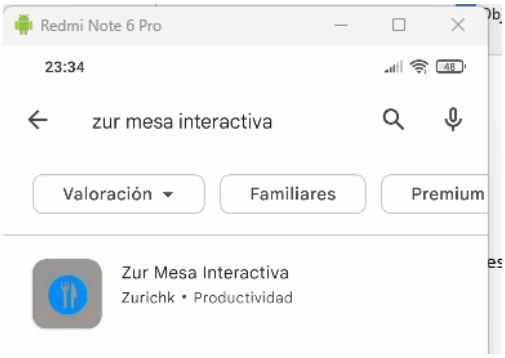
Podemos instalar la aplicación desde el apk proporcionado o podemos instalarla desde Google play store:



Abrimos Play Store 

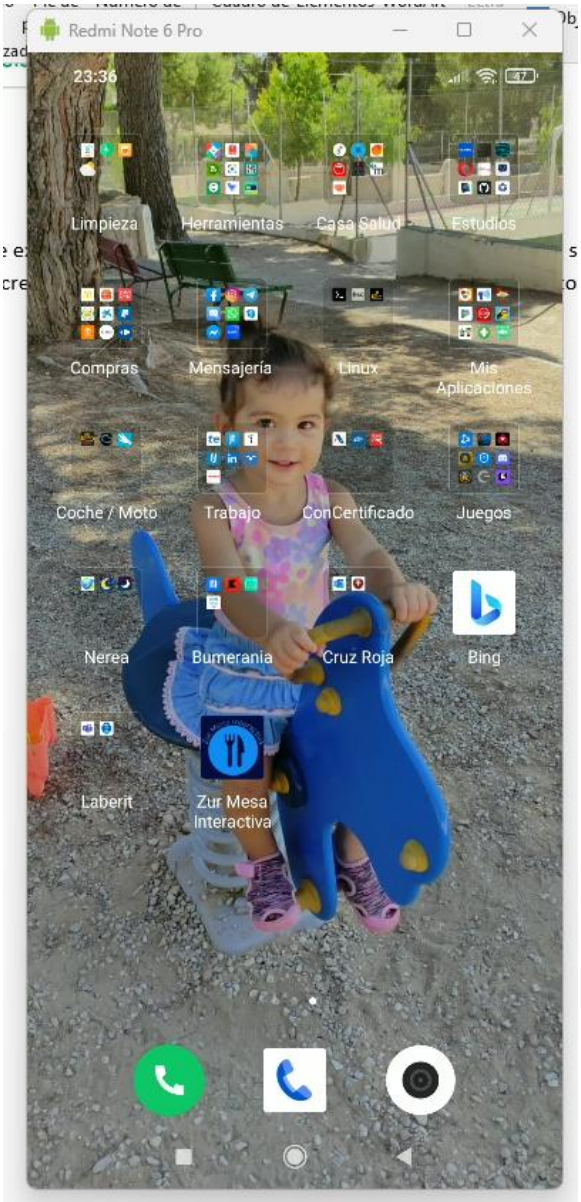
y utilizamos el buscador





Clic en Instalar

Una vez instalada nos aparece un acceso directo en nuestro teléfono móvil:



CONCLUSIONES

Posibles mejoras

Durante el desarrollo del proyecto han ido surgiendo distintas ideas y mejoras, de las cuales muchas han sido incorporadas.
Y otras mejoras que podríamos incluir en el futuro, podrían ser:

Aplicación de gestión

- Opción de crear y gestionar nuevas categorías de producto.
- Mejoras visuales en la interfaz.
- Opción de crear y modificar las publicidades de la aplicación de la mesa.

Aplicación de la mesa interactiva

- Podríamos añadir registro de usuarios/cliente.
- Mejoras visuales en la interfaz.
- Podríamos incluir identificación de que mesa está en uso, aunque este proyecto está basado en una única mesa, es posible que el futuro se instalen nuevas mesas y necesitemos identificar desde cual se ha realizado el pedido. *(En el caso de incluir esta mejora, también debería incorporarse al resto de aplicaciones incluso modificar la base de datos).*
- Realizar una interfaz configurable, es decir la posibilidad de modificar los tiempos de espera, modificar el fondo de la pantalla de pedido, etc.
- Personalización de pedidos: los clientes tendrán la opción de personalizar sus pedidos, como la selección de ingredientes o la adición de complementos.

Aplicación móvil

- Podríamos implementar un sistema de colas de mensajes, para que la aplicación de la mesa interactiva “avise” a la aplicación móvil de que hay un nuevo pedido. Podríamos usar distintos servidores: Mosquitto, RabbitMQ, myqtthub.com Aprovechando que disponemos de cuenta Azure podríamos hacer uso de:
  Azure Service Bus o Events Hub.
- Mejoras visuales en la interfaz.

Base de datos

- Incluir roles en la tabla usuarios, para la diferente gestión.
- Ocultar las contraseñas o que solo puedan ser consultadas por un determinado grupo de usuarios.

Conclusión final

A la hora de elegir el proyecto, surgen muchas dudas y pocas ideas, la fecha de elección es muy temprana y no sabemos realmente que podemos llegar a hacer.

No conocemos todas las herramientas que vamos a necesitar utilizar, a ni desarrollar.

He tratado de que el proyecto fuera lo más visual e interactivo posible, combinando todas las herramientas o habilidades adquiridas durante el curso, y algunas otras que he tenido que aprender una vez finalizado.

Actualmente en el mercado, hay frameworks o plantillas que facilitarían mucho este tipo de proyectos.

Ha sido un proyecto en el que he aprendido mucho, en el que he podido acercarme a un entorno de trabajo real, valorando que tecnologías pueden ser más convenientes, para cada tipo de funcionalidad.

He invertido muchas más horas de las que el módulo proyecto recomendaba, y aun así creo que me he quedado un poco corto, podría mejorar la visualización y la funcionabilidad de las aplicaciones, etc. Son muchas las posibles mejoras, que podríamos implementar, y algunas de ellas las hemos mencionado anteriormente.

BIBLIOGRAFIA

Para desarrollar el proyecto he hecho uso de la documentación aportada por el profesorado durante el curso.
Además de este material, he consultado distintas enlaces y videos web, los cuales voy a detallar en distintas categorías de uso.

Microsoft .NET/WPF/C#

Táctil en .NET

Para dotar a la aplicación de usuario de sistema táctil, he consultado:
https://social.msdn.microsoft.com/Forums/vstudio/en-US/2709c74f-3353-4677-88a2-ccf97bcf245f/click-and-drag-scrolling-using-scrollviewer?forum=wpf

https://learn.microsoft.com/es-es/dotnet/desktop/wpf/advanced/walkthrough-creating-your-first-touch-application?view=netframeworkdesktop-4.8

Personalización Componentes

Cuadro de Mensaje: https://www.youtube.com/watch?v=lPOzRBp7Q1E
ComboBox Personalizado: https://www.youtube.com/watch?v=6PafcmPZiBU

Notificaciones: https://github.com/rafallopatka/ToastNotifications

Giro Pantalla: https://github.com/nwjs/nw.js/issues/7071

Generar Instalador

https://www.youtube.com/watch?v=hlCGa6K56hA

Java Swing + NetBeans:

Personalización Componentes

Interfaz Personalizada: https://www.youtube.com/watch?v=LdBl0th_U_Q

JFileChooser Personalizado: https://www.youtube.com/watch?v=jqS-1a2ndVo

JTable: https://www.youtube.com/watch?v=QNg5uTCew14

https://www.youtube.com/watch?v=wTjQ5HJMRHc

Cambiar Icono: https://www.youtube.com/watch?v=BeFy7qRWa-Y

Generar Diagrama de clases

https://www.youtube.com/watch?v=MiysL0MC2ck

Generar Instalador

https://www.youtube.com/watch?v=BN_SFmEzCpI

Android Studio:

Personalización Componentes

Crear icono: https://www.youtube.com/watch?v=oI6WgrHwsuI

Cambiar Icono: https://www.youtube.com/watch?v=e1o43z6MKNg

Api Retrofit (Código)

Consumiendo Api Retrofit: https://cursokotlin.com/capitulo-20-consumiento-apis-retrofit-2

Generar APK

https://www.youtube.com/watch?v=H0-W3Xb4HMY

Microsoft Azure:

Blob Storage: https://www.youtube.com/watch?v=31rZf-73_jw
API WEB: https://www.youtube.com/watch?v=qqDSb5Q6hQY

Bases de datos:

Diagrama entidad relación con Workbench:
https://www.javierrguez.com/generar-diagrama-entidad-relacion-mysql/

Desarrollar documentación:

Portada: https://www.youtube.com/watch?v=n4ZE5A_J1ss

Índice: https://www.youtube.com/watch?v=BeBzOsIiIoU

Logos, imágenes e iconos

Para la generación de logos, imágenes, iconos, he usado distintas páginas:
https://logomakr.com/

https://express.adobe.com/es-ES/sp/

https://www.istockphoto.com/

https://logo-maker.freelogodesign.org/

http://icongal.com/

Creación de Empresa

Para la elección del tipo de empresa a crear, he consultado:

https://www.autonomosyemprendedor.es/articulo/guias-de-emprendimiento/emprender-desembolsar-solo-euro/20190214174048018895.html

https://www.infoautonomos.com/tipos-de-sociedades/como-crear-una-sociedad-limitada/

https://www.txerpa.com/blog/sociedad-limitada-formacion-sucesiva

http://www.ipyme.org/es-ES/creaciondelaempresa/ProcesoConstitucion/Paginas/SLFS.aspx?cod=SLFS&nombre=Sociedad+Limitada+de+Formaci%C3%B3n+Sucesiva&idioma=es-ES

Riesgos Laborales

Para desarrollar los posibles riesgos laborales, y como prevenirlos, he consultado:

https://www.mutualia.eus/wp-content/uploads/2018/03/Documento_unificado_plan_prevencion_Mutualia.pdf

ANEXOS

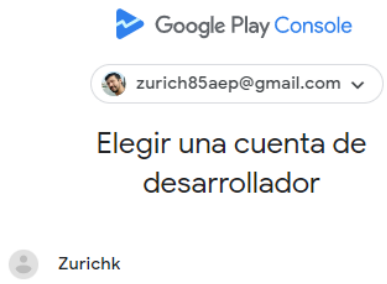
He creado un apartado de Anexos, para incluir ciertas tareas que he realizado durante el proyecto, como la generación de apk firmada, subir apk a Google play store, etc.

Subir Aplicación Google Play

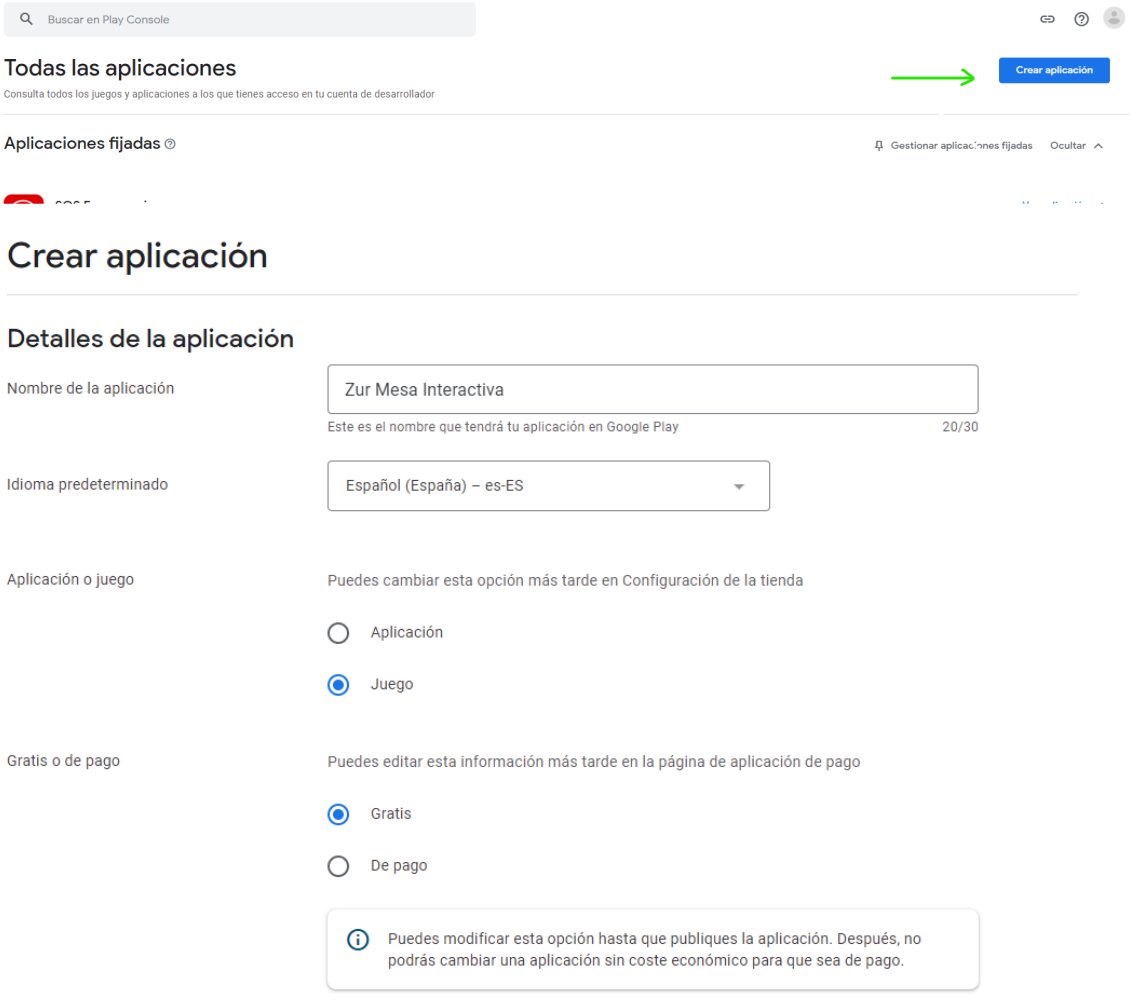
El link para acceder a Google Console es: https://play.google.com/console/developers/?pli=1

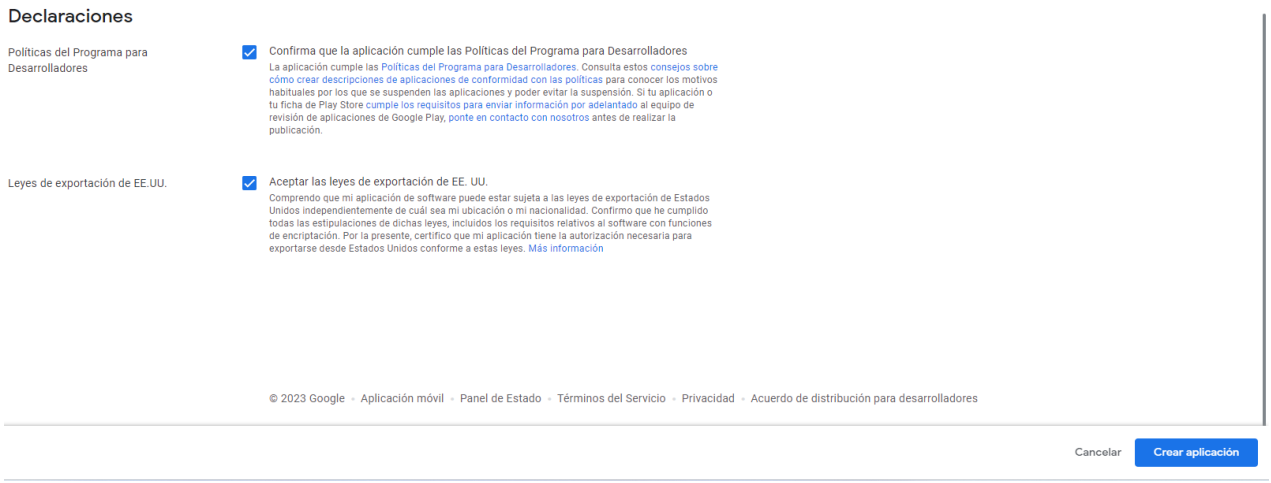
Necesitas una cuenta de desarrollador para subir tu aplicación

Accedo a mi cuenta de usuario.



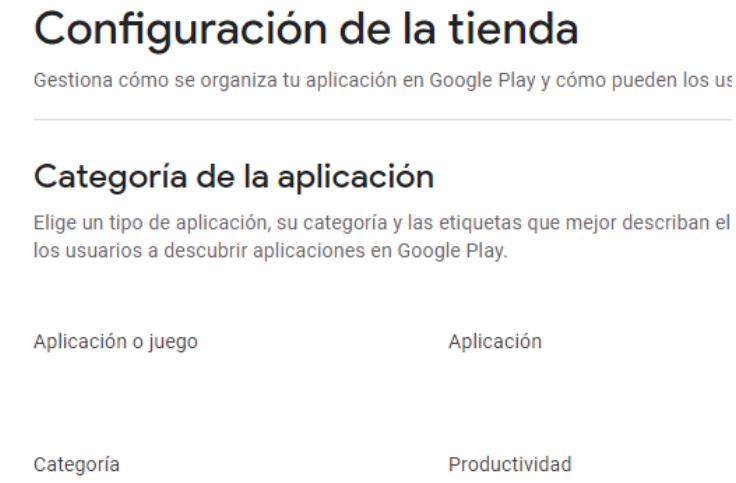
Vamos a crear Aplicación:



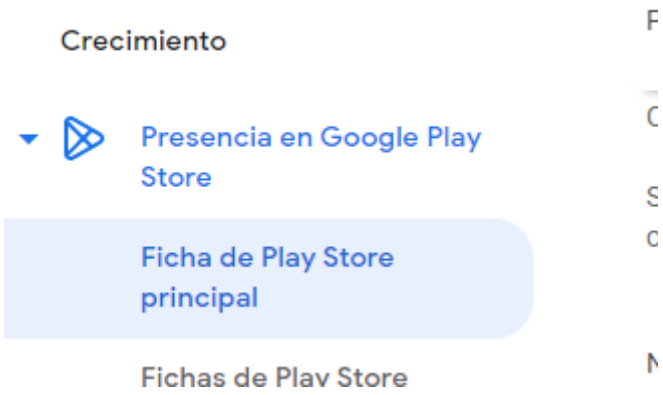


Aceptamos Declaraciones y pulsamos en Crear Aplicación.

NOTA: En la revisión de la aplicación comprobamos que habíamos marcado Juego por Error y lo corregimos:



Una vez creada la aplicación, vamos al apartado de Crecimiento en el menú de la izquierda, y accedemos a la Fiche de Play Store Principal para completar los datos.



Completamos el contenido solicitados, y subimos algunas imágenes(iconos, portada, capturas)

Recursos de la ficha

Consulta la Política de metadatos y las indicaciones del Centro de Ayuda para evitar los problemas habituales con tu ficha de Play Store. Consulta todas las políticas del programa antes de enviar tu aplicación.

Si cumples los requisitos para enviar información por adelantado al equipo de revisión de aplicaciones de Google Play, ponte en contacto con nosotros antes de publicar tu ficha de Play Store.

Nombre de la aplicación *

Zur Mesa Interactiva

Este es el nombre que tendrá tu aplicación en Google Play 20/30

Descripción breve *

Recibir pedidos de la mesa interactiva

Breve descripción de tu aplicación. Los usuarios pueden mostrar la descripción completa. 38/80

Descripción completa *

Recibir pedidos de la mesa interactiva, realizados por los clientes, para poder gestionarlos desde la cocina.

Gráficos

Gestiona el icono, las capturas de pantalla y los vídeos de tu aplicación para promocionarla en Google Play. Revisa las directrices de contenido antes de subir nuevos elementos gráficos. Si añades traducciones de tu ficha de Play Store que no incluyan elementos gráficos localizados, usaremos los gráficos del idioma predeterminado.

Icono de la aplicación *

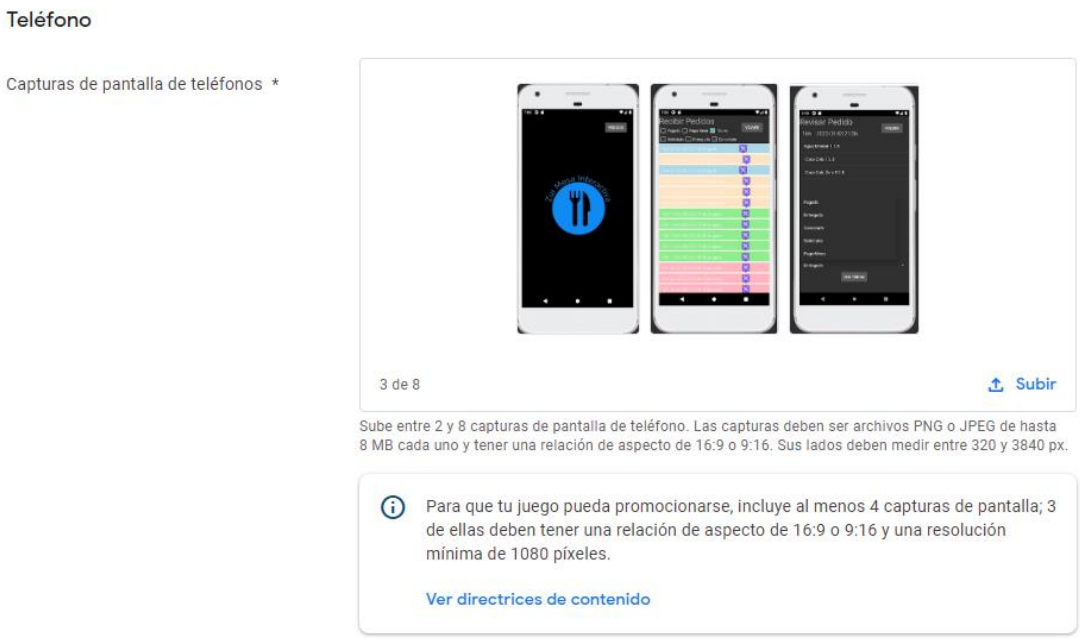
Reemplazar

El icono de tu aplicación debe ser un archivo PNG o JPEG transparente de 1 MB como máximo y 512x512 px, y debe cumplir las especificaciones de diseño y la política de metadatos

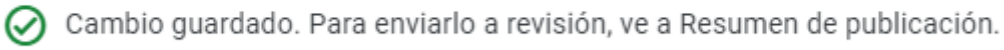
Gráfico de funciones *

Reemplazar

Tu gráfico de funciones debe ser un archivo PNG o JPEG de 15 MB como máximo y medir 1024x500 px

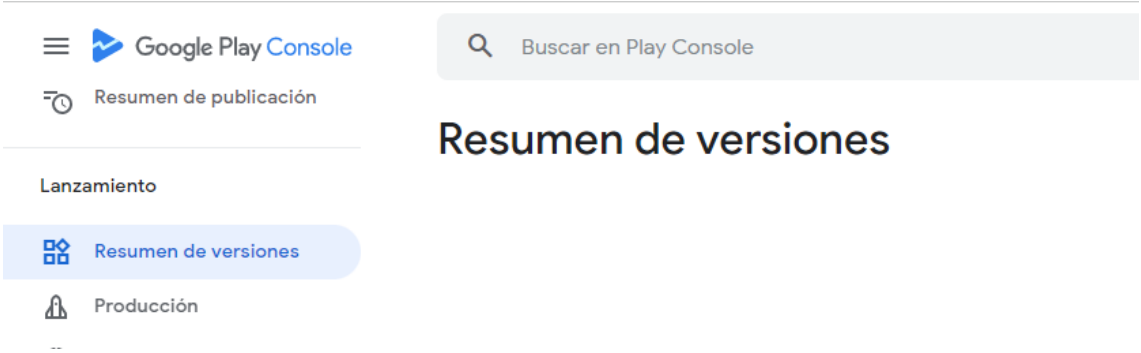


Una vez completado pulsamos en GUARDAR para agregar los cambios.

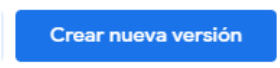


Ahora vamos a subir nuestro bundle firmado.

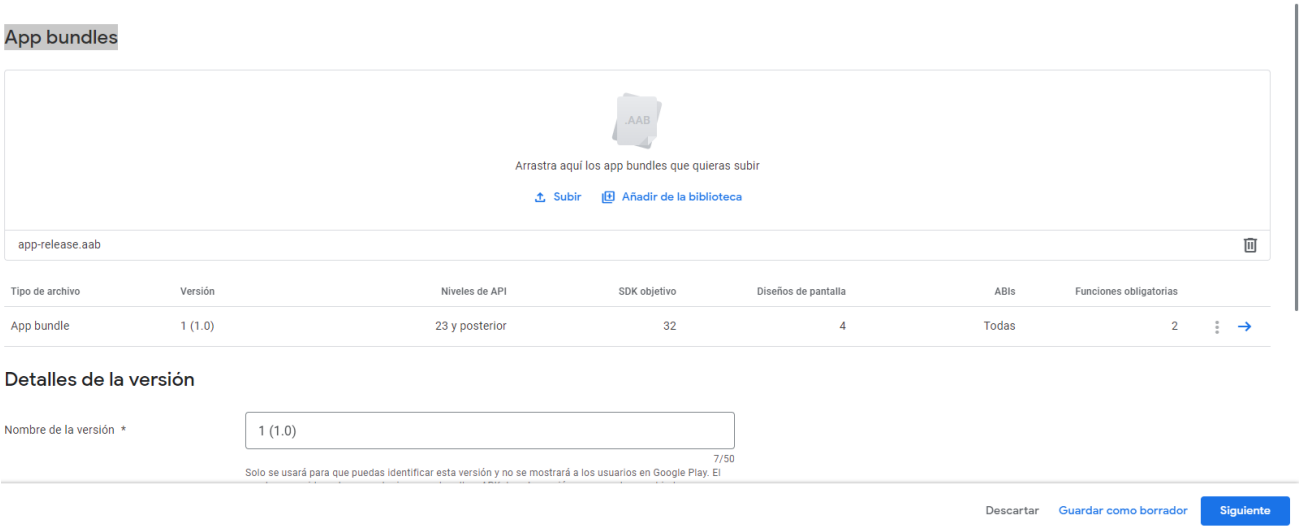
En la parte izquierda vamos al menú Lanzamiento/Resumen de versiones



Vamos a Producción y luego Crear nueva versión



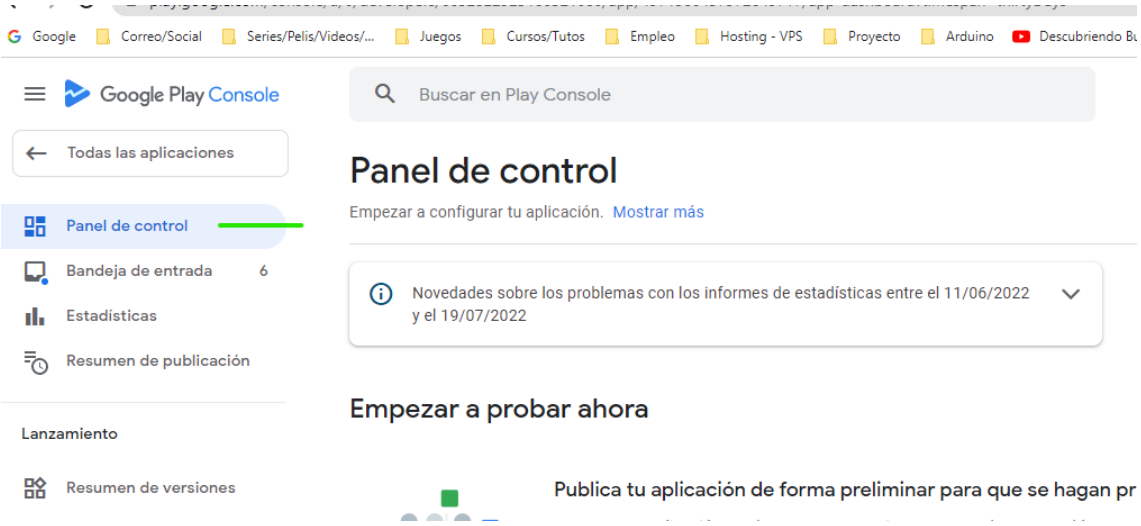
Aquí es donde debemos subir nuestra Bundle firmado, podemos consultar el anexo Generar APK / Bundle Firmado, para generarlo y firmarlo.



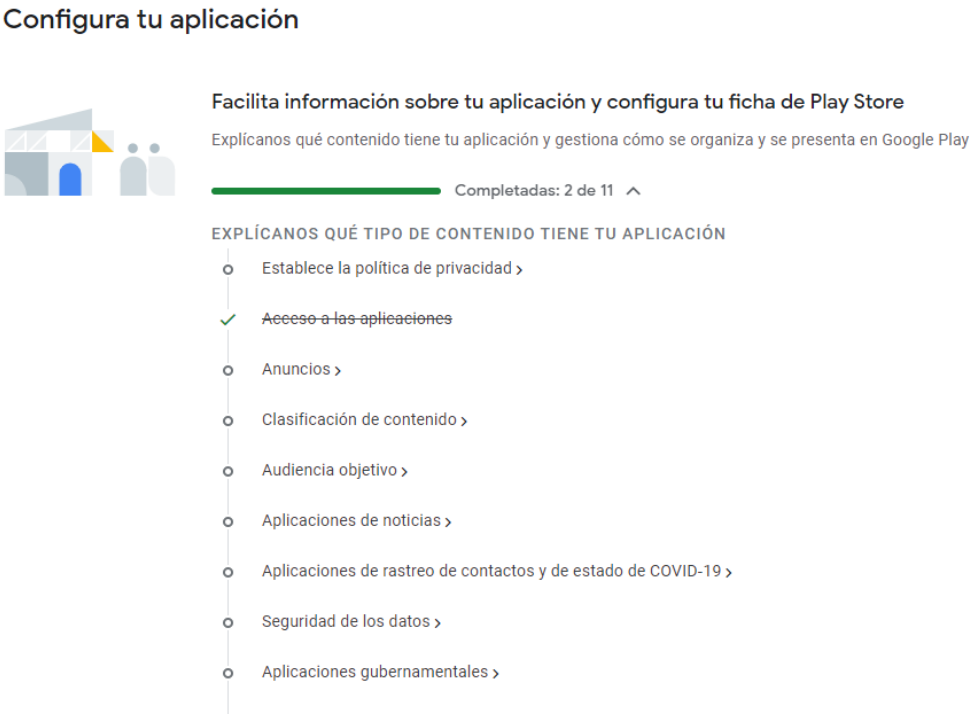
Añadimos la versión, y si queremos podemos añadir también alguna nota aclaratoria sobre esta versión.

Guardamos como borrador y vamos a siguiente, para revisar la versión, donde obtenemos varios errores que debemos subsanar.

De nuevo en el menú de la izquierda, vamos a Panel de Control.

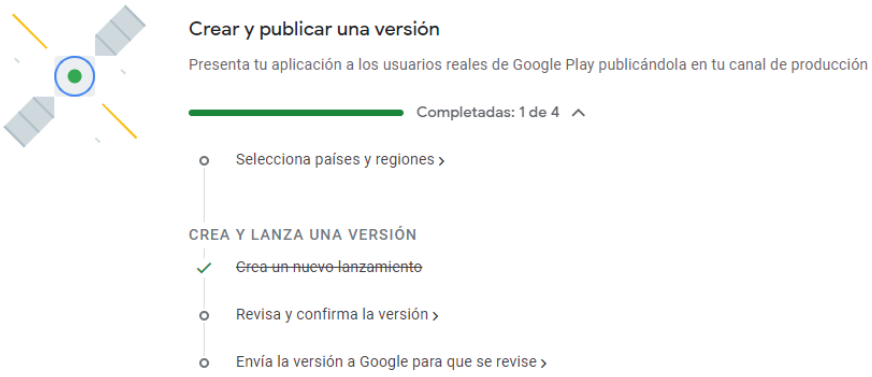


Debemos completar todos estos puntos:



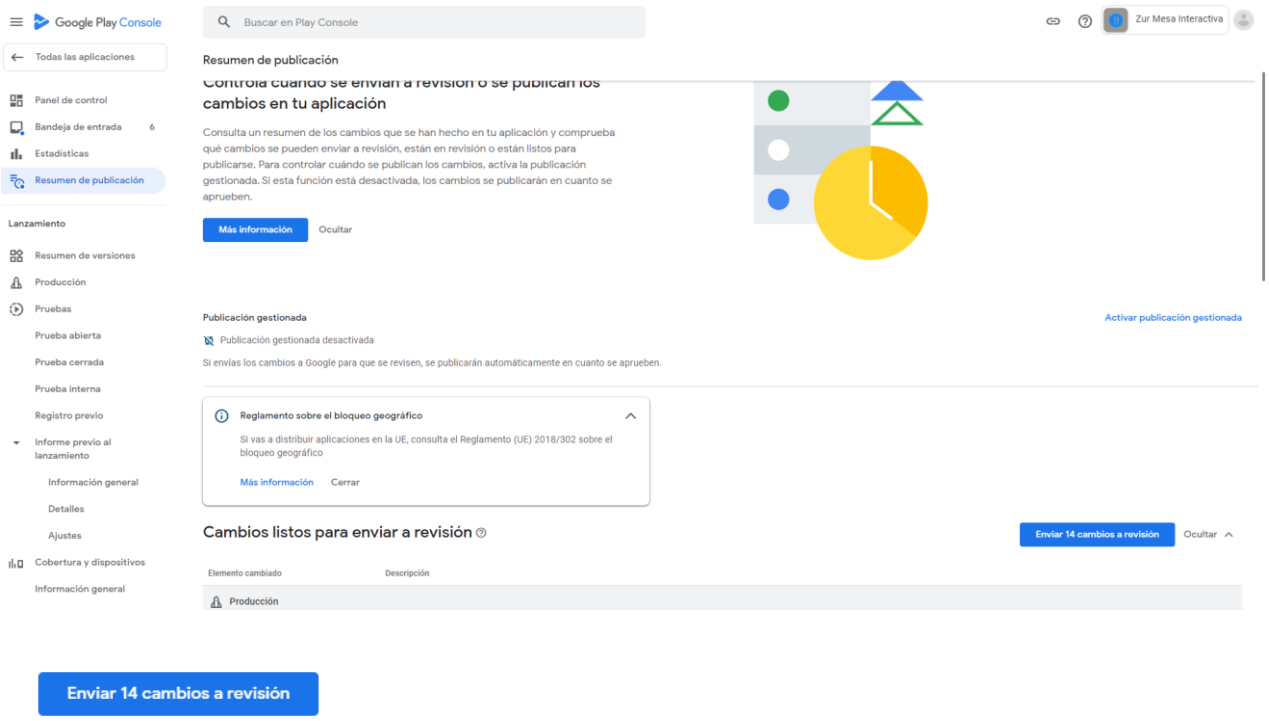
En el acceso a las aplicaciones, debemos introducir la contraseña que incluimos en la generación del key store en Android Studio.

Una vez completados todos los puntos, vamos al apartado:



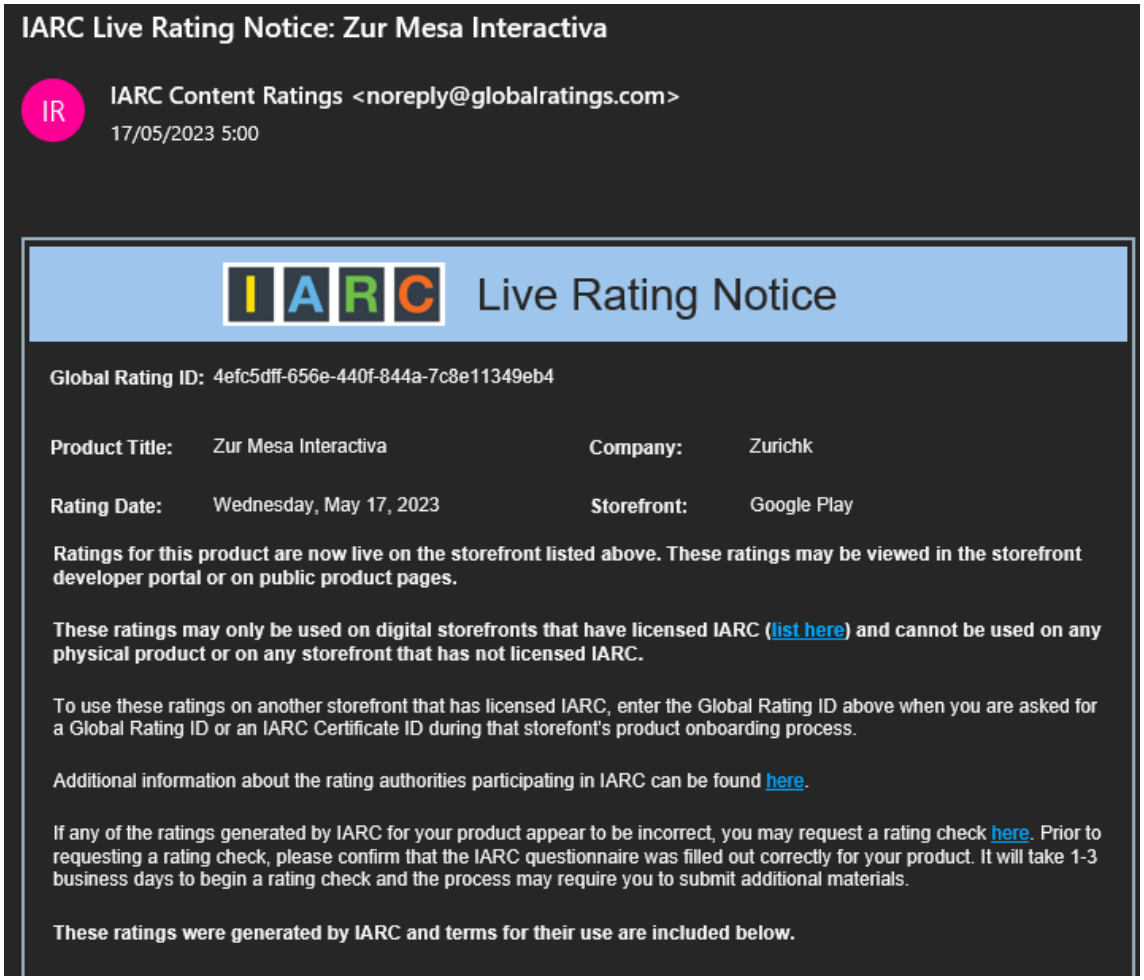
Y Añadimos los países en los que queremos que esté disponible nuestra aplicación.

La revisamos, y la enviamos a Google para que la revise.



Comprobamos el estado actual: En Revisión.

En unos 3 días recibimos un correo informando que la aplicación ha sido publicada.



IARC Live Rating Notice: Zur Mesa Interactiva

IR

IARC Content Ratings <noreply@globalratings.com>

17/05/2023 5:00

IARC Live Rating Notice

Global Rating ID: 4efc5dff-656e-440f-844a-7c8e11349eb4

Product Title:	Zur Mesa Interactiva	**Company:**	Zurichk
Rating Date:	Wednesday, May 17, 2023	**Storefront:**	Google Play

Ratings for this product are now live on the storefront listed above. These ratings may be viewed in the storefront developer portal or on public product pages.

These ratings may only be used on digital storefronts that have licensed IARC (list here) and cannot be used on any physical product or on any storefront that has not licensed IARC.

To use these ratings on another storefront that has licensed IARC, enter the Global Rating ID above when you are asked for a Global Rating ID or an IARC Certificate ID during that storefont's product onboarding process.

Additional information about the rating authorities participating in IARC can be found here.

If any of the ratings generated by IARC for your product appear to be incorrect, you may request a rating check here. Prior to requesting a rating check, please confirm that the IARC questionnaire was filled out correctly for your product. It will take 1-3 business days to begin a rating check and the process may require you to submit additional materials.

These ratings were generated by IARC and terms for their use are included below.

Panel de control

Novedades sobre los problemas con los informes de estadísticas entre el 11/06/2022 y el 19/07/2022

Zur Mesa Interactiva

com.zurichk.mesainteractiva • Ver en Google Play

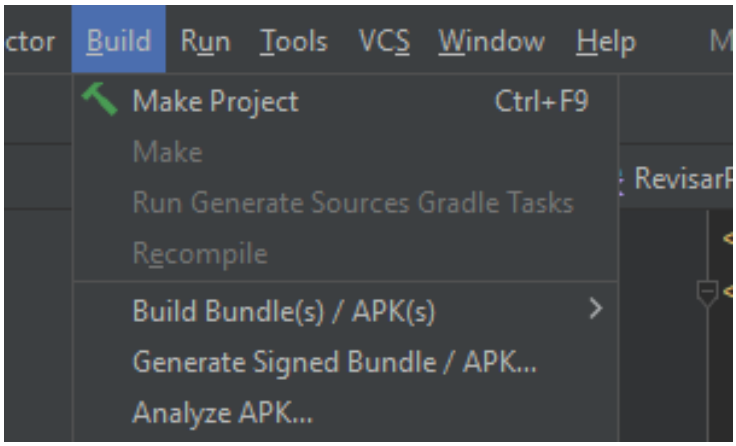
Producción

Activo • 0 dispositivos activos • 176 países o regiones

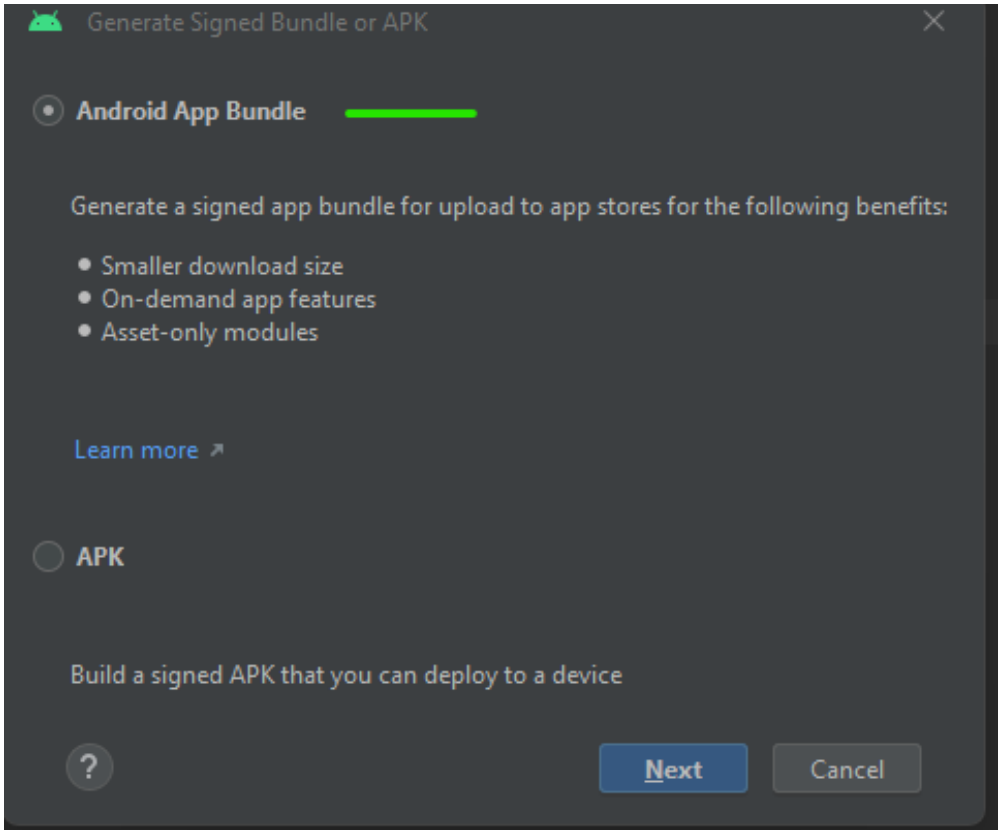
Generar APK / Bundle Firmado

En este caso vamos a explicar el proceso para generar un bundle firmado.

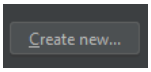
Vamos al menú desplegable Build de Android Studio, y clicamos en la opción Generate Signed Bundle / APK.



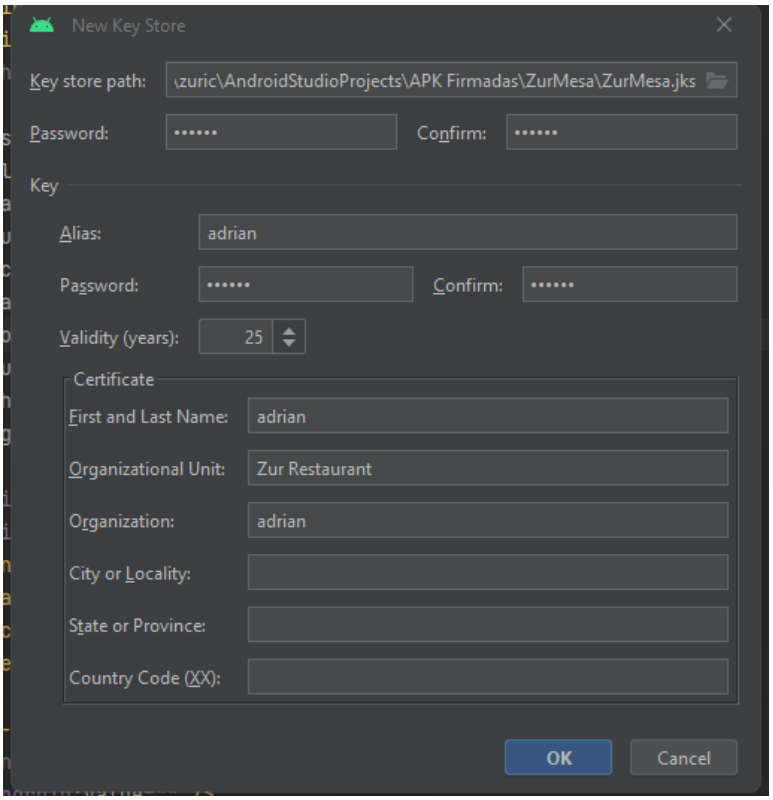
Seleccionamos la opción Android APP Bundle, pulsamos NEXT.



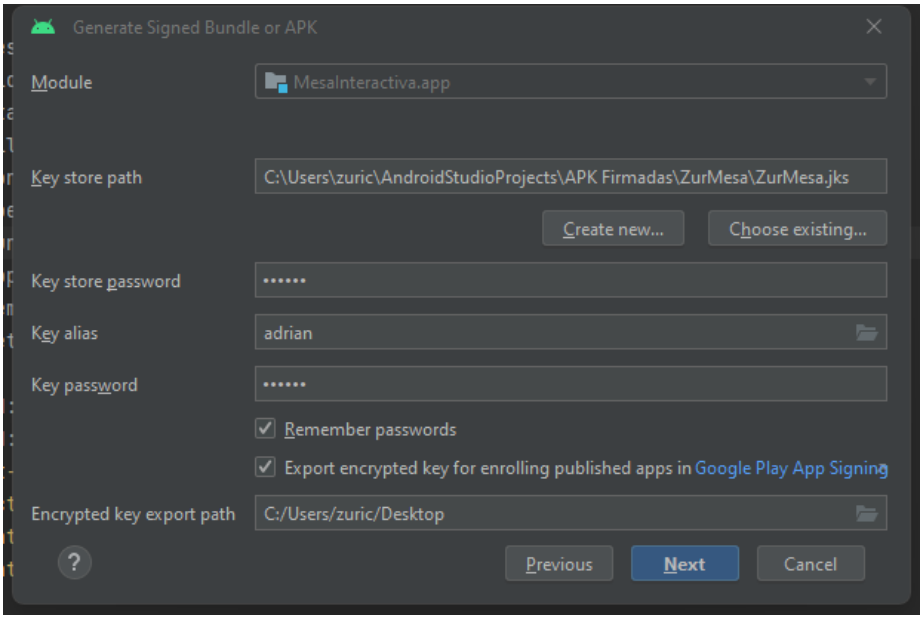
En la siguiente pantalla, pulsamos en Create new, para crear nuestra Key Store.



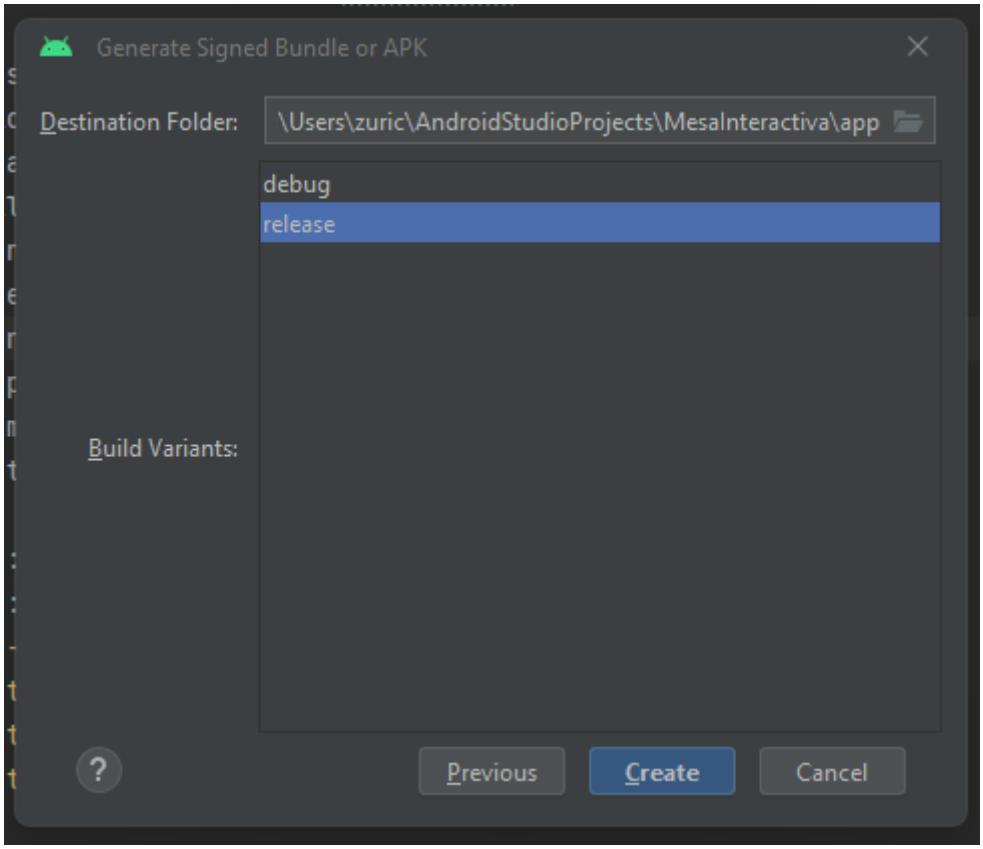
Completamos los datos, para este proyecto, hemos puesto de contraseña adrian; *debemos recordar este password ya que debemos incluirlo en acceso a las aplicaciones, al subir la aplicación a Google play console.*



Una vez generado, debemos completar el resto de datos, y pulsamos en NEXT.

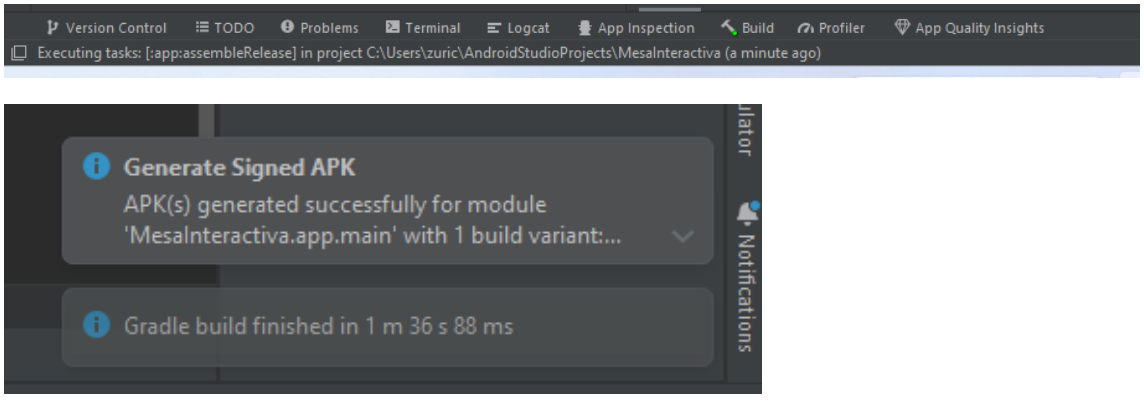


Elegimos la opción reléase / producción.



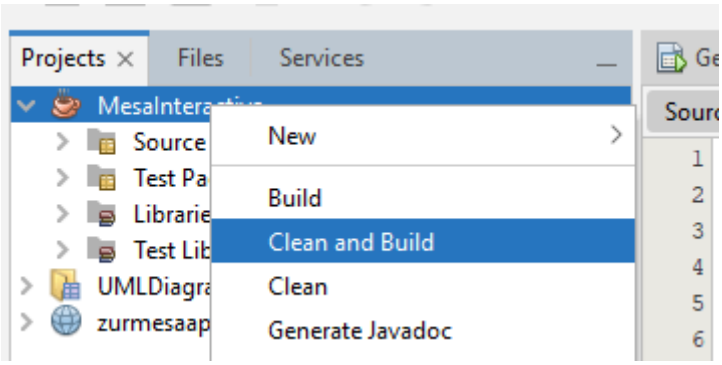
Y le clicamos en CREATE.

He iniciara el proceso de creación de bundle.

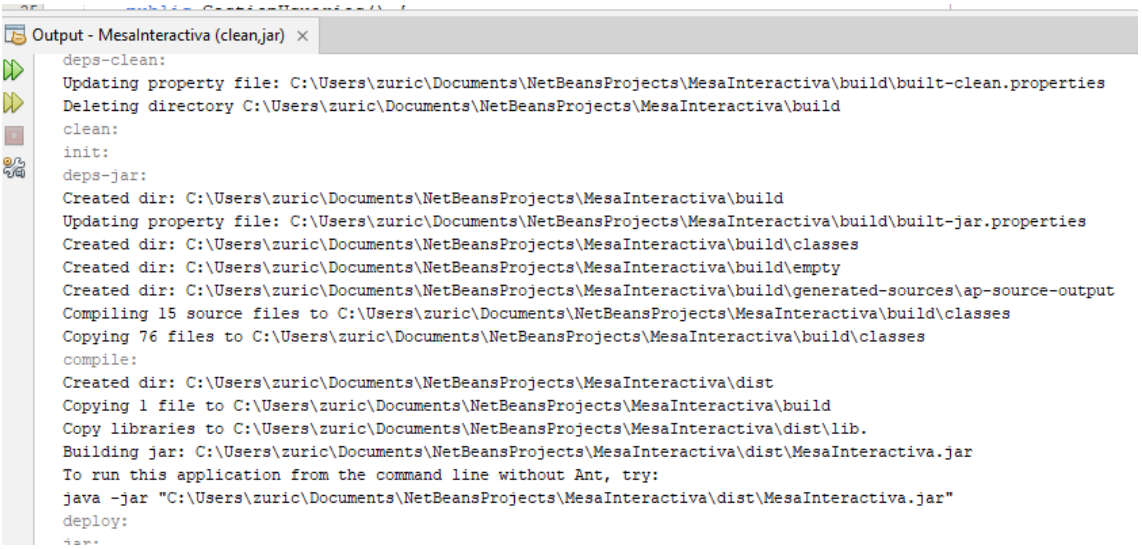


Generar .JAR desde NetBeans

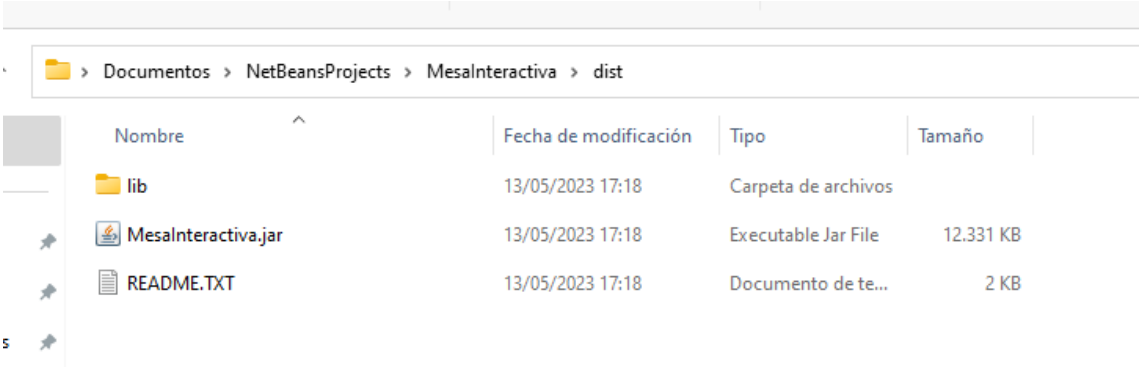
Para generar un .jar desde NetBeans, debemos utilizar la opción Clean & Build, seleccionando nuestro proyecto y realizando clic derecho.



Se iniciará el proceso para generar el .jar:

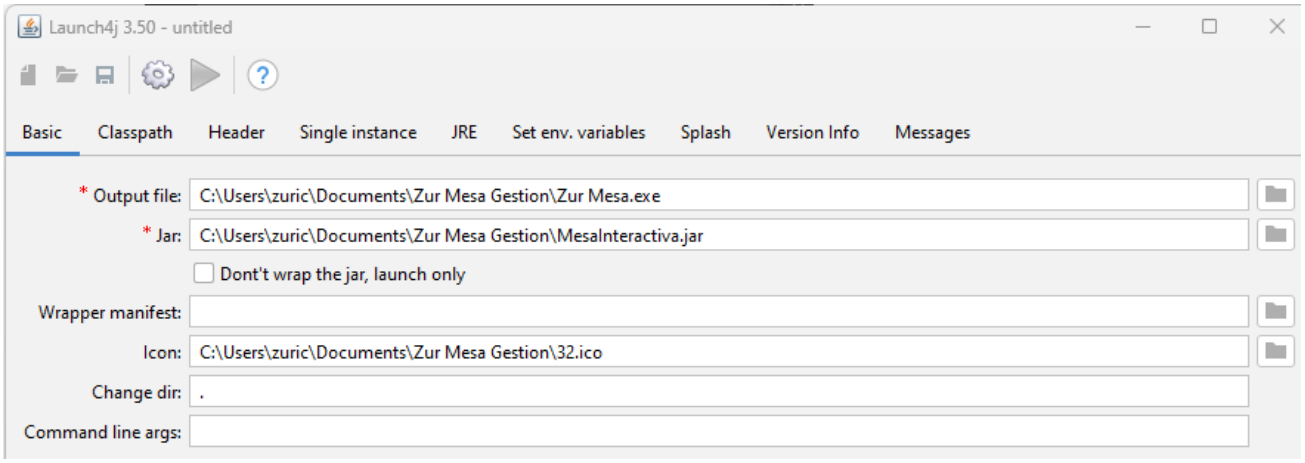


Una vez finalizado, podemos consultar y ejecutar nuestro .jar.



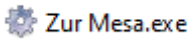
Generar .EXE a partir de .JAR

Para generar un .exe a partir de un .jar vamos a utilizar la herramienta Launch4j.

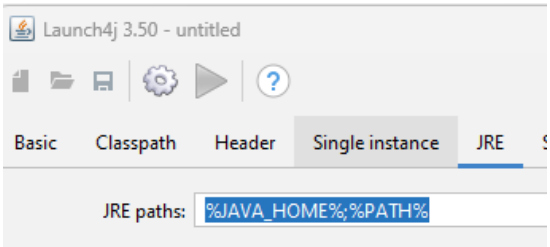


En la pestaña BASIC, completamos los siguientes campos:

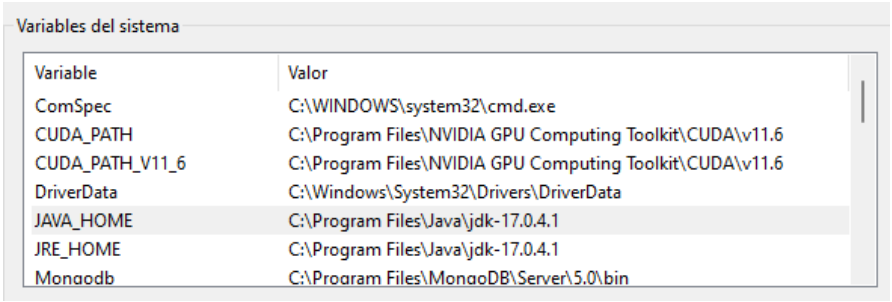
- Output file: Añadimos el nombre o archivo de salida (el .exe que se generará).
- Jar: Seleccionamos nuestro archivo .jar.
- Icon: El icono que queremos que tenga nuestro .exe.



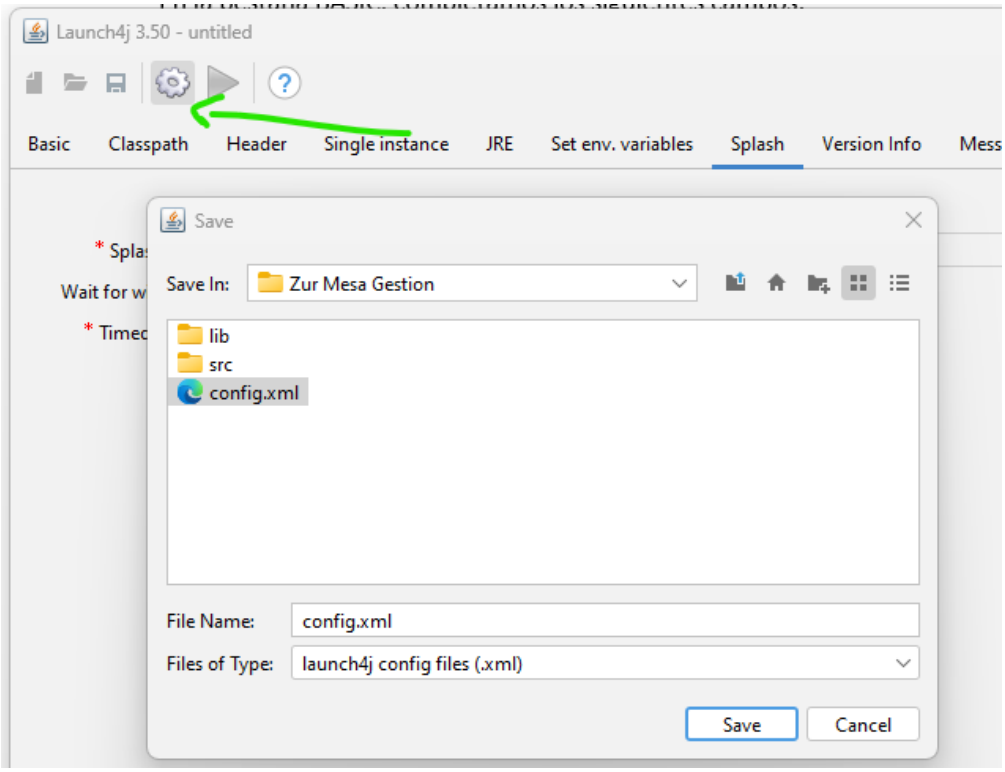
Vamos a la pestaña JRE, he incluimos el path de nuestra versión de java.



Podemos revisarlo en las variables de entorno.



Vamos a la “rueda” /engranaje.



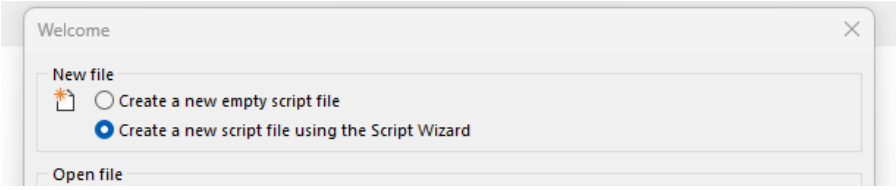
Y damos un nombre a un fichero de configuración que se va a generar.

En el caso de que nuestra aplicación incluya librerías externas, tenemos varias opciones, para solucionarlo, o bien generar el .jar con estas librerías ya incluidas, o incluir la carpeta lib, donde se encuentre el .exe.

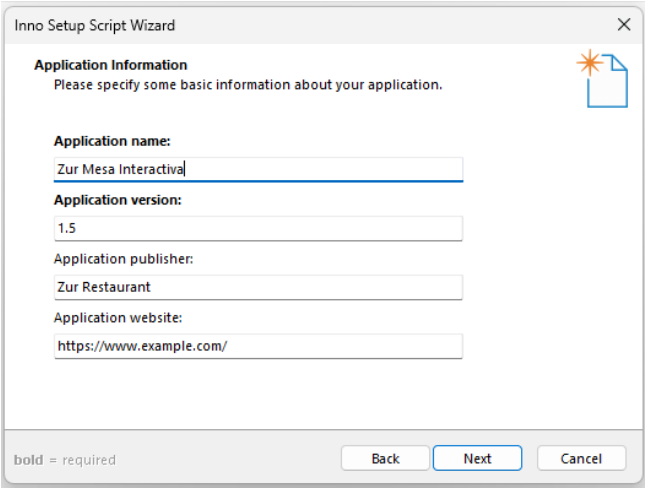
Generar Instalador a partir de .EXE

Para generar un instalador de nuestra aplicación a partir del .exe, vamos a utilizar la aplicación Inno Setup Compiler.

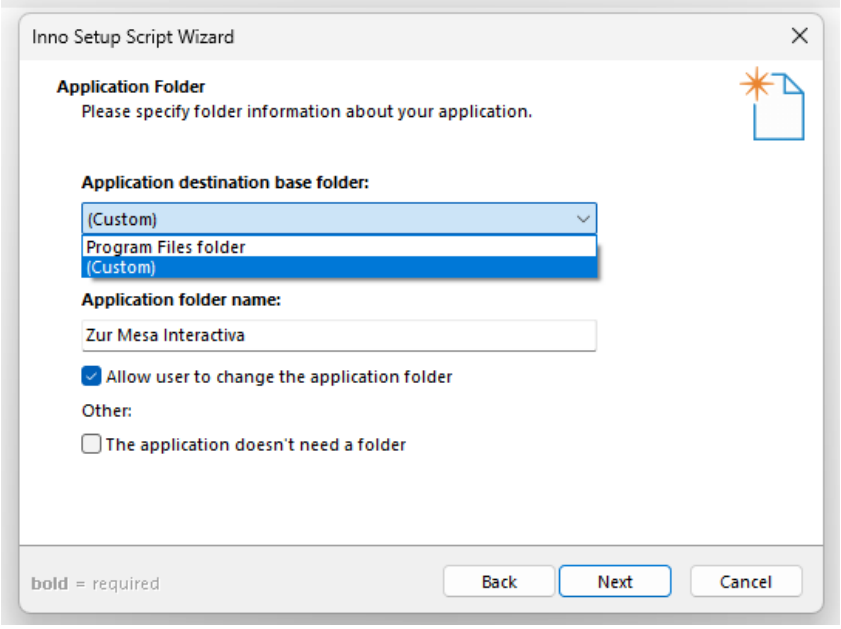
Al iniciar la aplicación, seleccionamos la opción, Create a new script file using the Script Wizard.



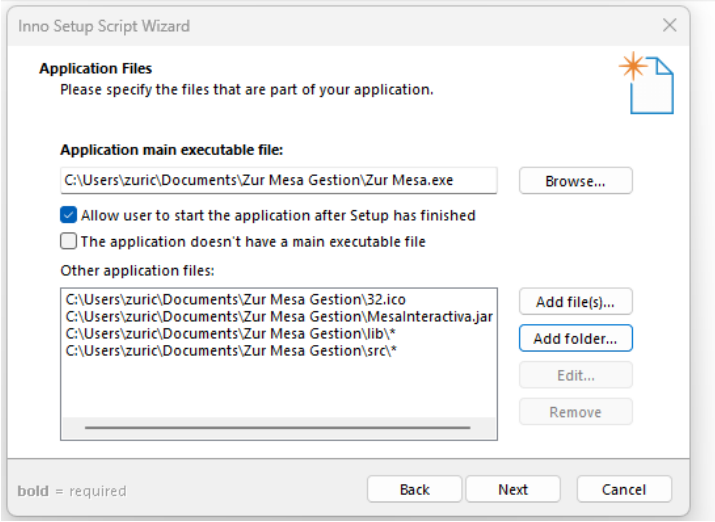
Completamos los siguientes campos y clicamos NEXT.



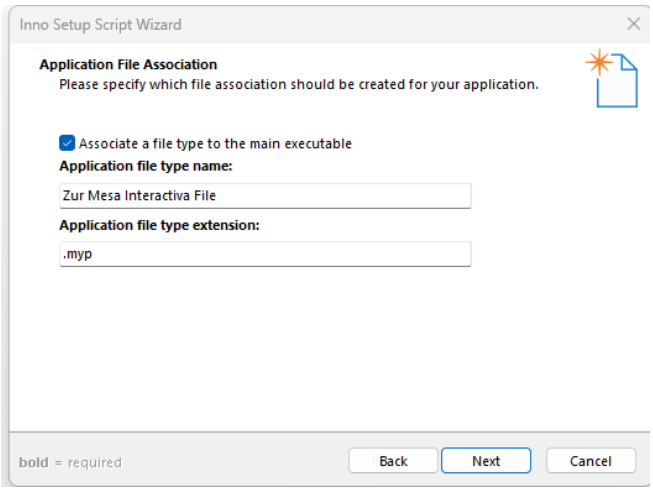
Aquí podemos seleccionar la carpeta donde queremos que se instale por defecto, y el nombre de la carpeta (*para este ejemplo he seleccionado la ruta C:\Zur Mesa Interactiva*):



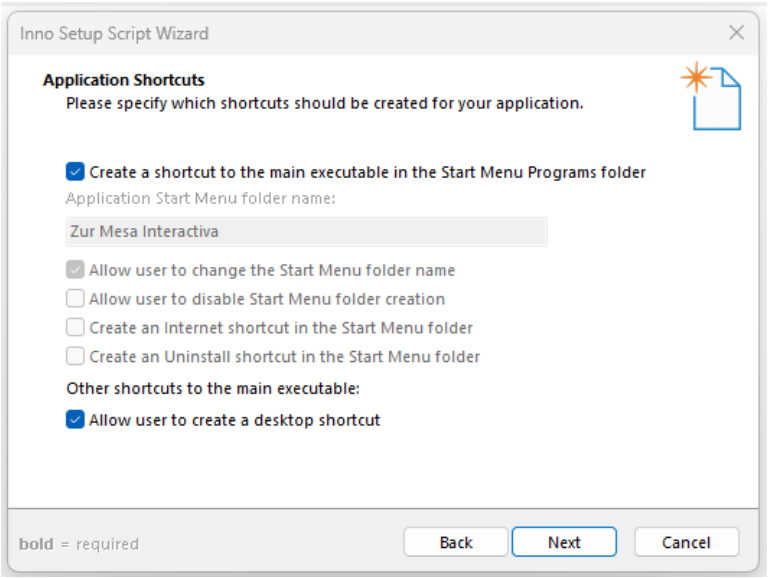
Seleccionamos el Exe que contiene nuestra aplicación y añadimos las carpetas necesarias para el correcto funcionamiento de nuestra aplicación.



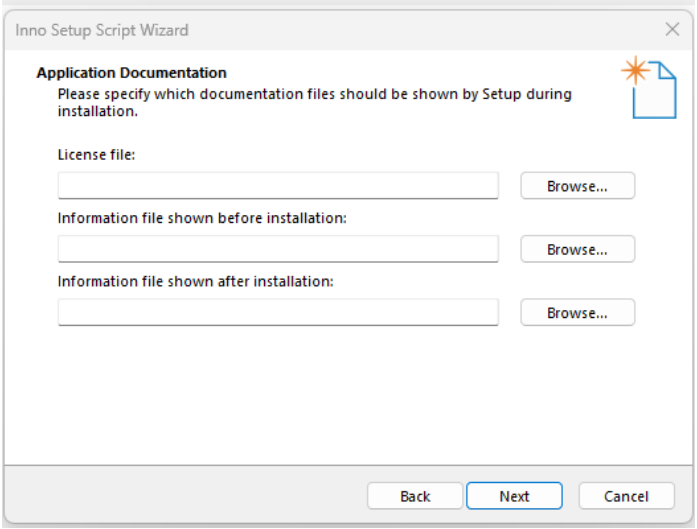
NEXT.



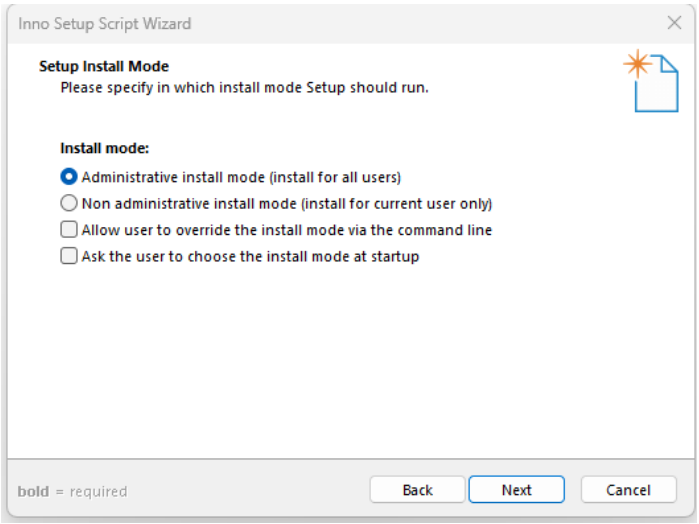
NEXT.



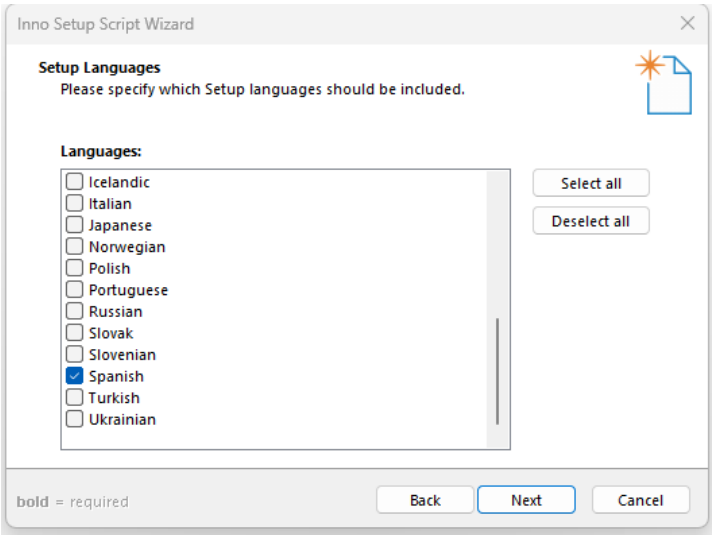
NEXT.



NEXT.

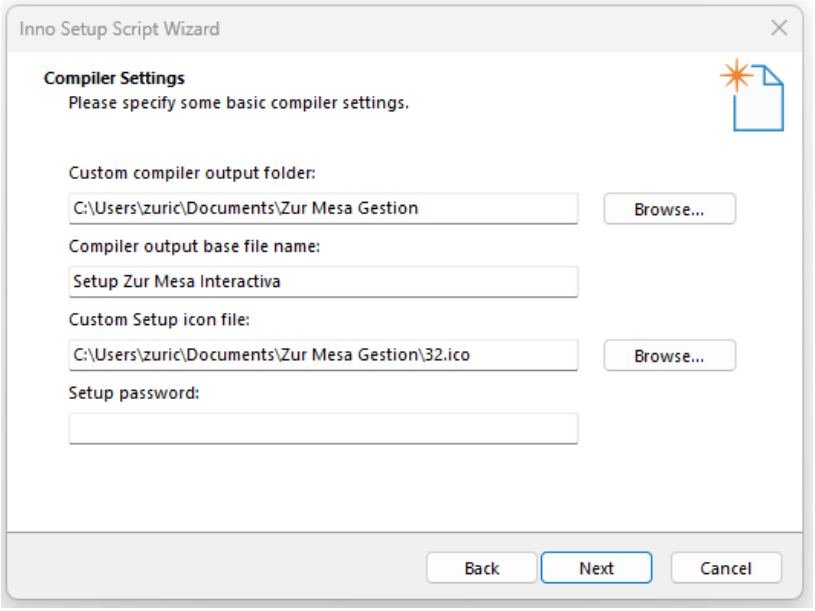


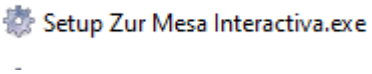
NEXT. Seleccionamos el idioma.



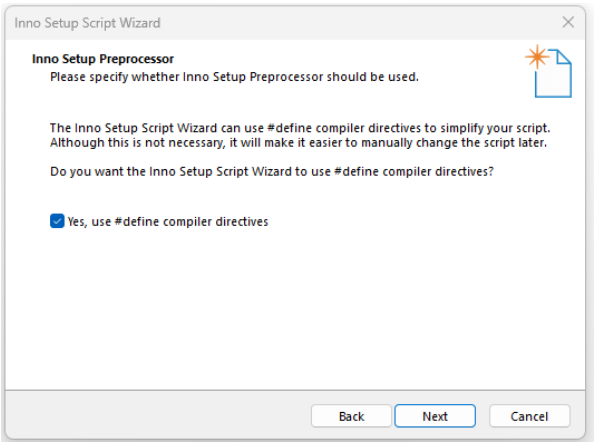
NEXT.

Añadimos la carpeta de salida, el nombre del archivo de instalación y el icono que tendrá.



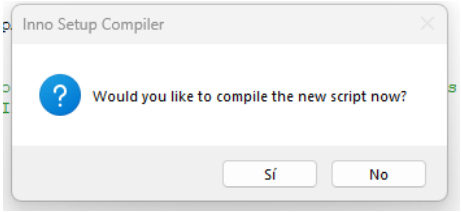


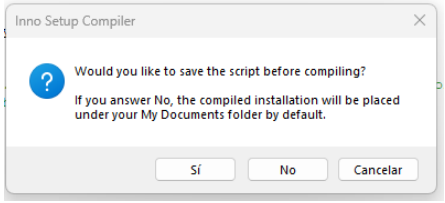
NEXT.



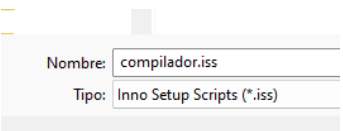
FINISH.

Clic en SI.





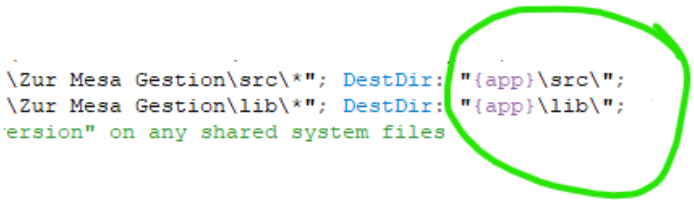
Guardamos el script.



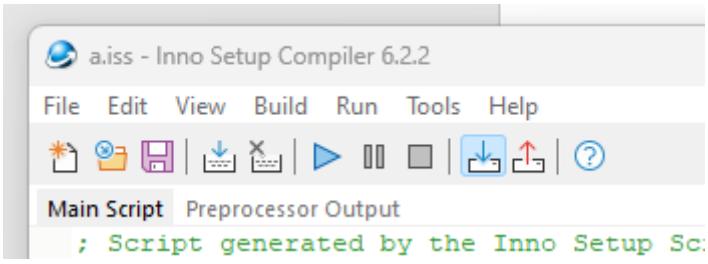
En el script resultante, debemos localizar estas 2 lineas:

```
[Files]
Source: "C:\Users\zuric\Documents\Zur Mesa Gestion\{#MyAppExeName}"; DestDir: "{app}"; Flags: ignoreversion
Source: "C:\Users\zuric\Documents\Zur Mesa Gestion\32.ico"; DestDir: "{app}"; Flags: ignoreversion
Source: "C:\Users\zuric\Documents\Zur Mesa Gestion\MesaInteractiva.jar"; DestDir: "{app}"; Flags: ignoreversion
Source: "C:\Users\zuric\Documents\Zur Mesa Gestion\lib\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs
Source: "C:\Users\zuric\Documents\Zur Mesa Gestion\src\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs
; NOTE: Don't use "Flags: ignoreversion" on any shared system files
```

Y añadimos su dirección completa.



Volvemos a compilar, podemos hacerlo pulsando en la opción RUN o play.



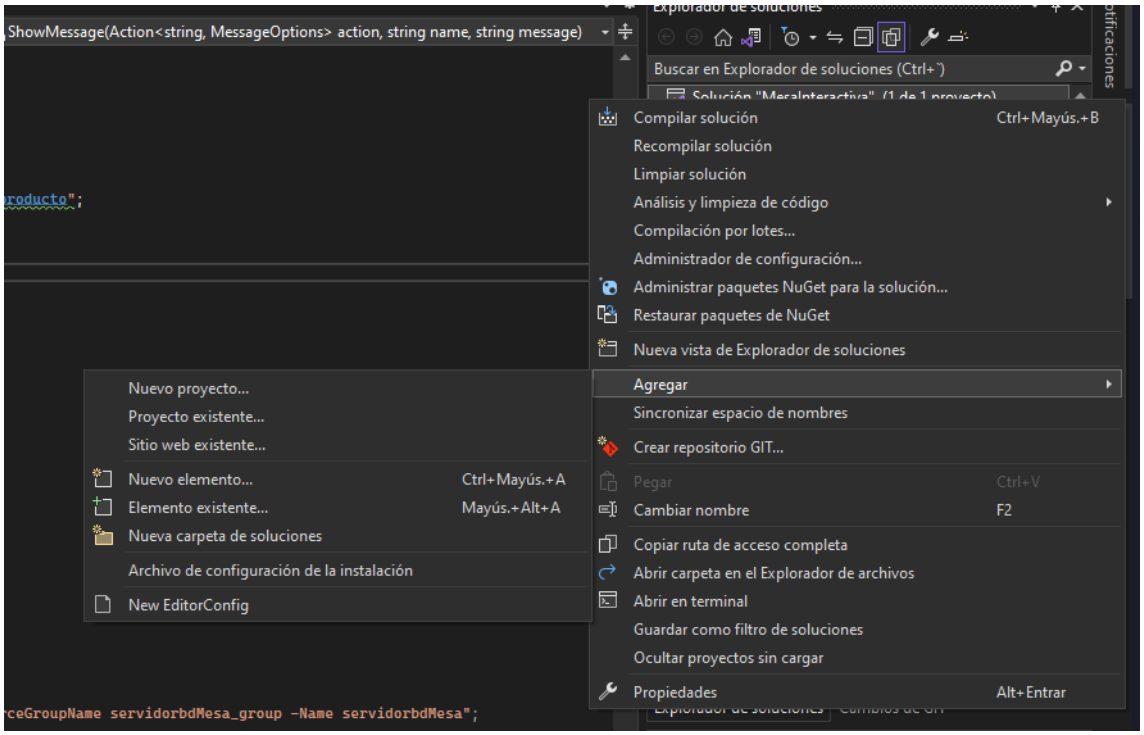
Y se generará nuestro instalador:

Generar Instalador App/WPF con Microsoft Visual Studio Installer Projects

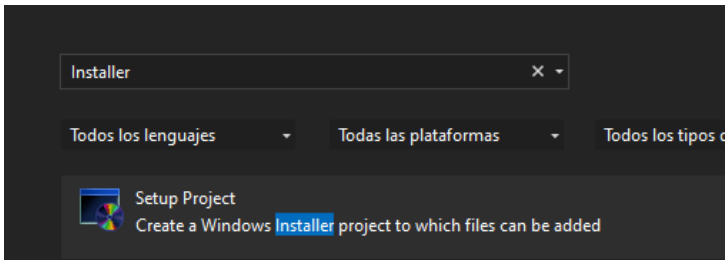
Para generar un instalador para nuestra aplicación, vamos a utilizar la extensión Microsoft Visual Studio Installer Projects.

Descarga:
https://marketplace.visualstudio.com/items?itemName=VisualStudioClient.MicrosoftVisualStudio2017InstallerProjects

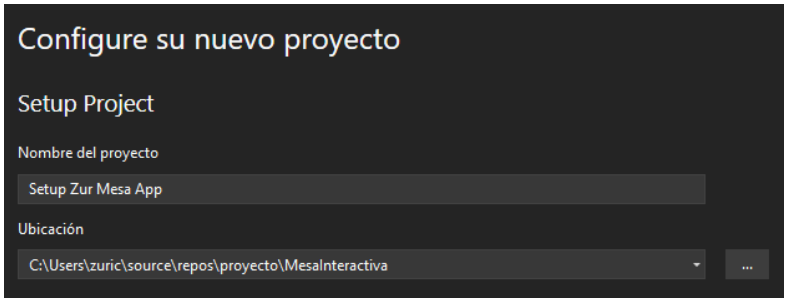
En nuestra solución, hacemos clic derecho/Agregar/Nuevo proyecto

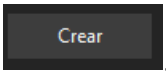


Usamos el buscador.



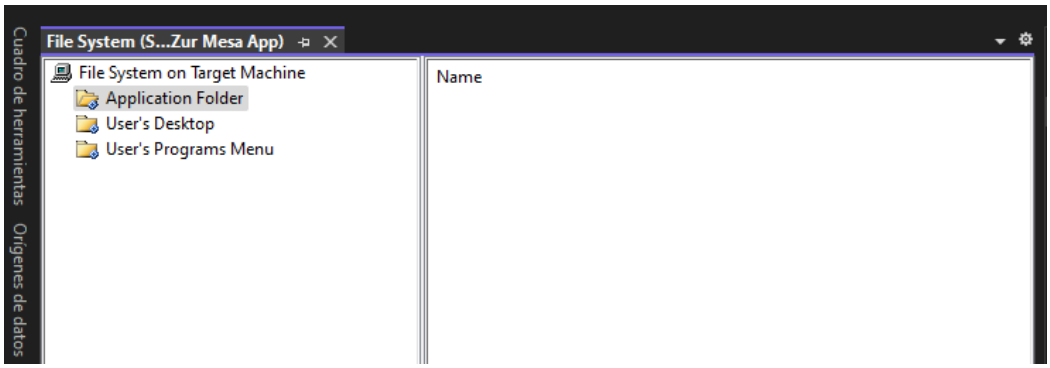
Le damos un nombre a nuestro instalador:



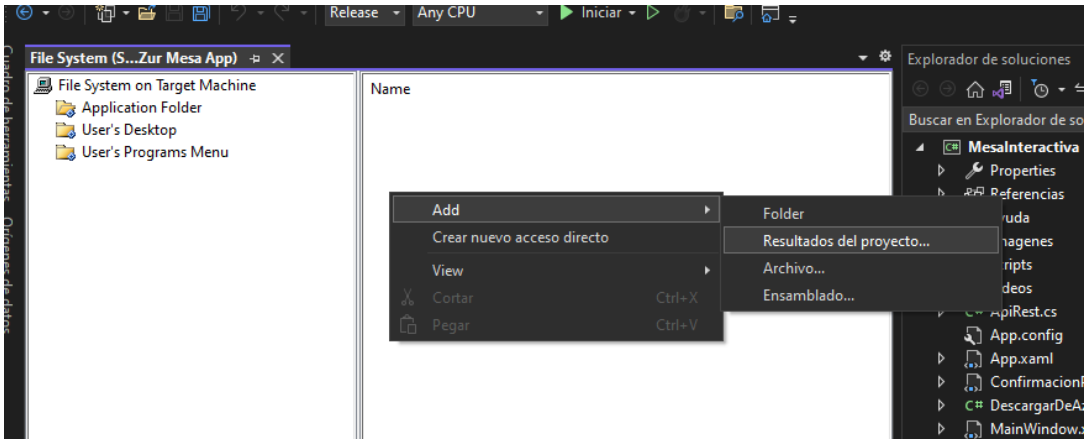
Y hacemos clic en Crear 

.

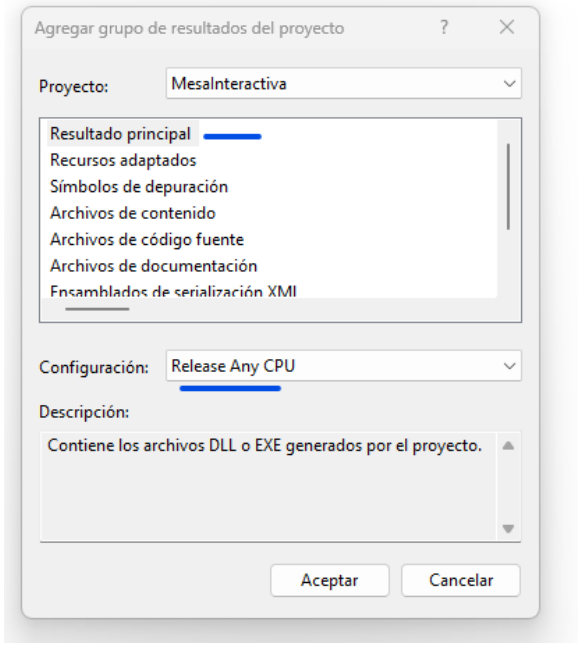
Seleccionamos Application Folder en la parte izquierda.



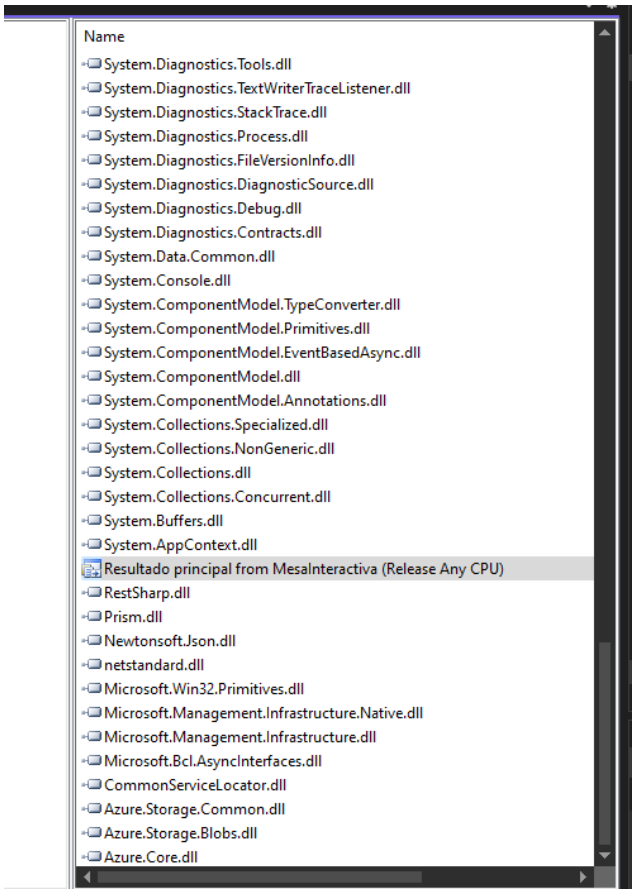
Hacemos clic con el botón derecho en la parte derecha y seleccionamos Add/Resultados del proyecto.



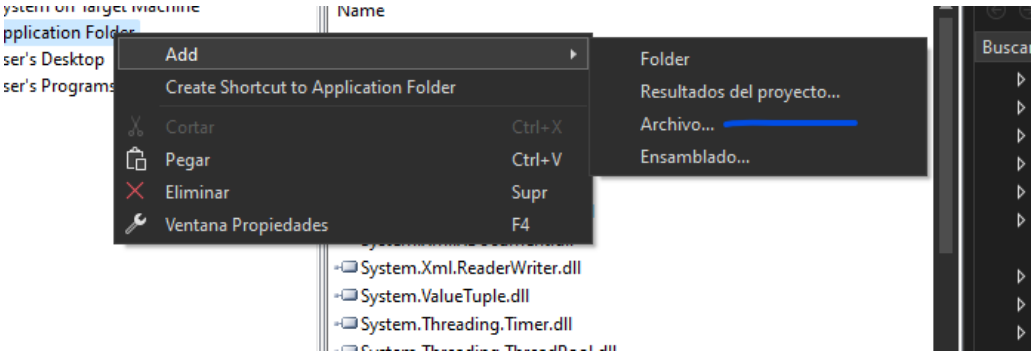
Seleccionamos Resultado Principal y en Configuración Release Any CPU y clic Aceptar.



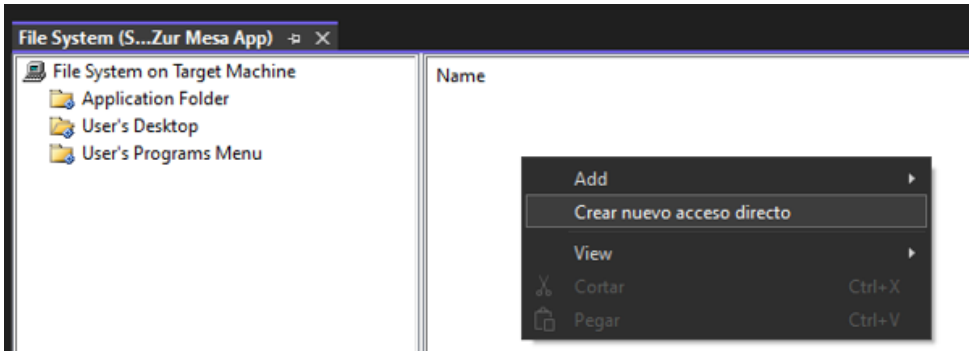
Y se añaden todas las librerías necesarias de nuestro proyecto:



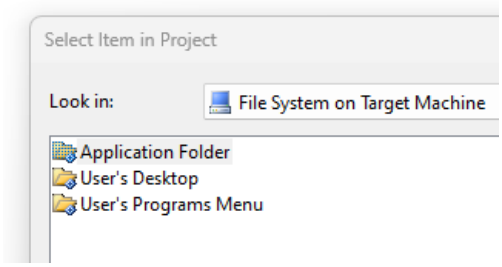
Para añadir un icono.



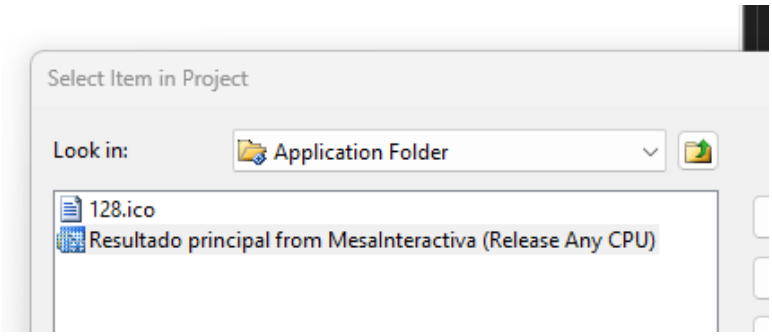
Ahora vamos a crear accesos directos.



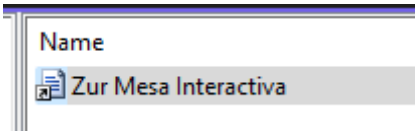
Doble clic en Application Folder.



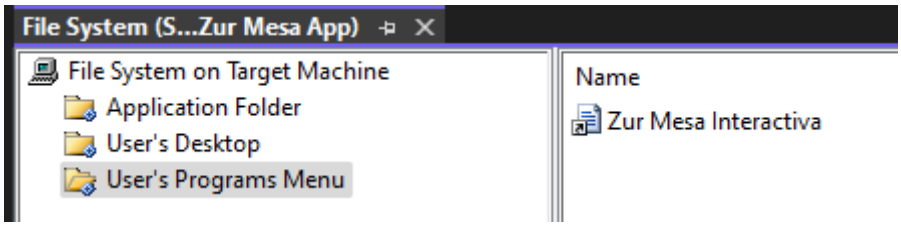
Y seleccionamos Resultado principal.



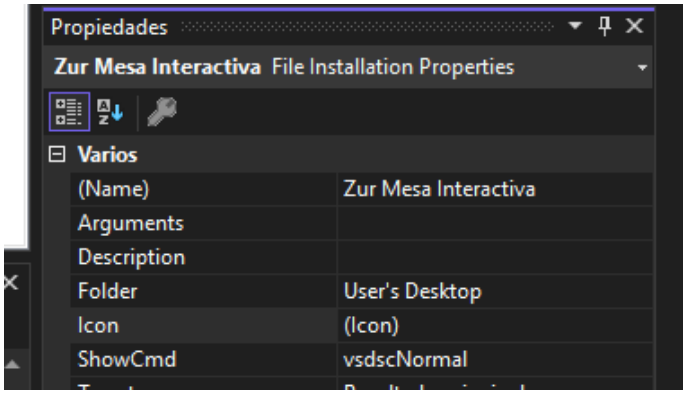
Clic en Ok. Una vez generado, le cambiamos el nombre.



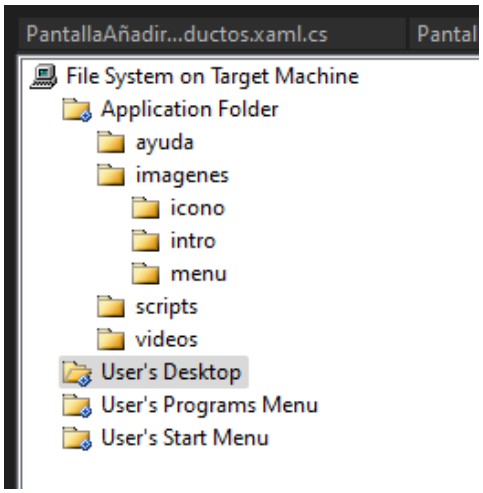
Realizamos el mismo procedimiento para añadirlo a User's Programs Menu.

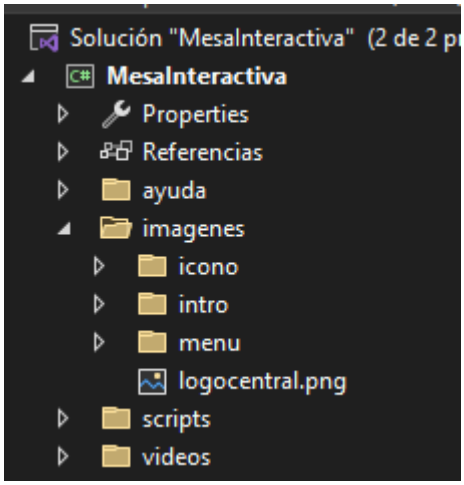


Agregamos un icono a los accesos directos que acabamos de crear, señalando el acceso directo, podemos visualizar sus propiedades en la parte derecha, cambiamos el icono, buscando el que añadido al principio.

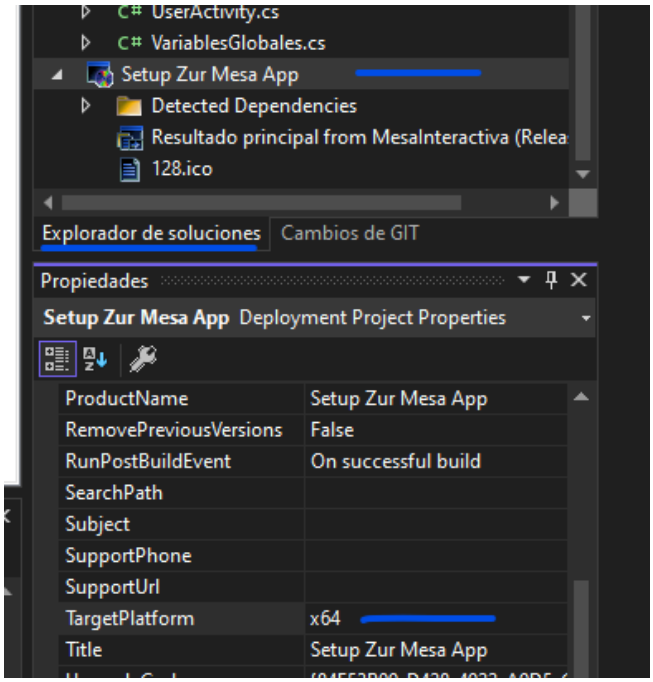


Debemos agregar la misma distribución de carpetas y archivos que tenemos en nuestro programa, ya que hay imágenes y funciones que utilizan estos archivos.

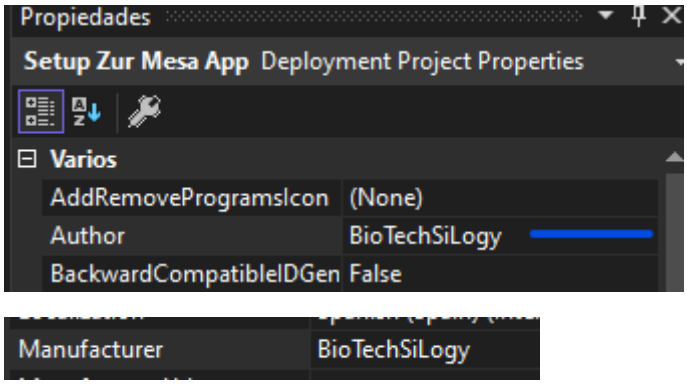




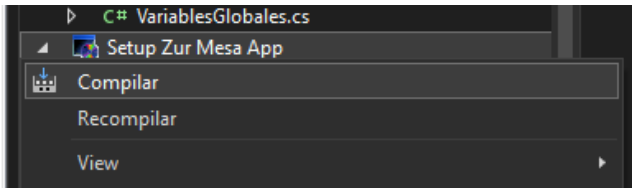
Ahora debemos añadir la arquitectura de nuestra aplicación, seleccionamos nuestro proyecto Setup Zur Mesa App, y en Propiedades, configuramos el TargetPlaform.



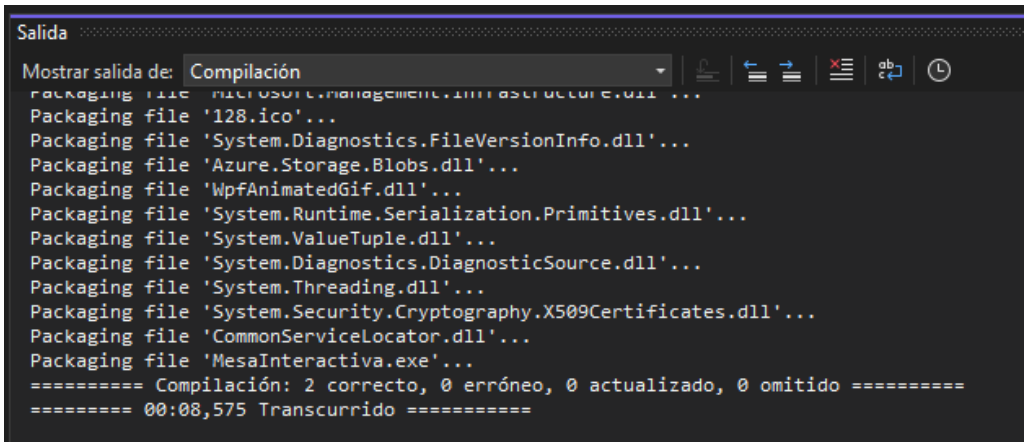
También cambiamos el nombre del Autor y Manufacturer, por el de nuestra empresa.

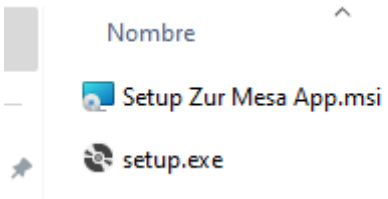


Ahora vamos a compilarlo, pulsamos botón derecho sobre Setup Zur Mesa App y hacemos clic en Compilar.



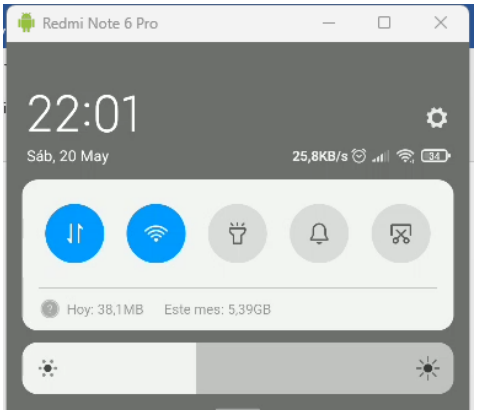
Ya tenemos nuestro instalador:





Conexión Móvil PC con Scrcpy

Antes de realizar una conexión y poder controlar el móvil desde el pc, debemos configurar varias opciones en el móvil. Primero vamos a ajustes  deslizando el dedo en la pantalla del móvil de arriba abajo.

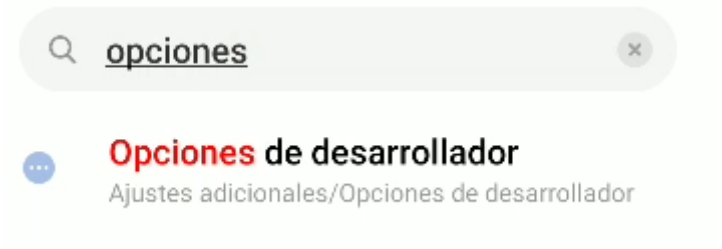


Debemos activar las opciones de desarrollador, accedemos a la opción Sobre el teléfono.

Y hacemos varios clics en Versión de MIUI, hasta que aparezca un toast informado que se han habilitado las opciones de desarrollador.

Volvemos 

Y utilizamos el buscador.



Activamos las opciones de desarrollador.

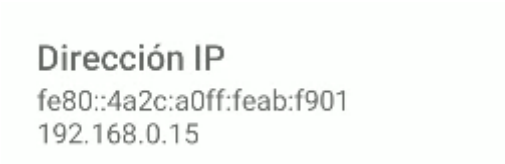
Y activamos la depuración USB.

Volvemos 

Usamos de nuevo el buscador, para obtener la dirección IP.

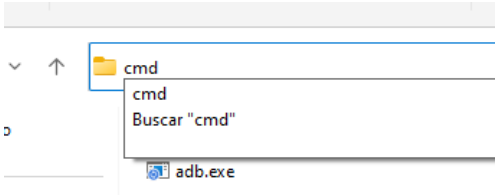
Se abrirá la opción de ajustes adicionales y nos desplazamos hasta abajo del todo.

Anotamos la Dirección IP.



Ahora vamos al pc, y abrimos la carpeta donde se encuentra nuestro scrcpy.

En la barra del explorador escribimos cmd.



Si es la primera vez que nos conectamos la móvil desde adb, debemos conectarlo por usb y reiniciar el puerto 555.

adb tcpip 5555

```
G:\Mi unidad\Cursos - FP - Master\DAM - Desarrollo de Apl
caciones Multiplataforma\Proyecto\Mesa\Programas usados\s
>adb tcpip 5555
restarting in TCP mode port: 5555
```

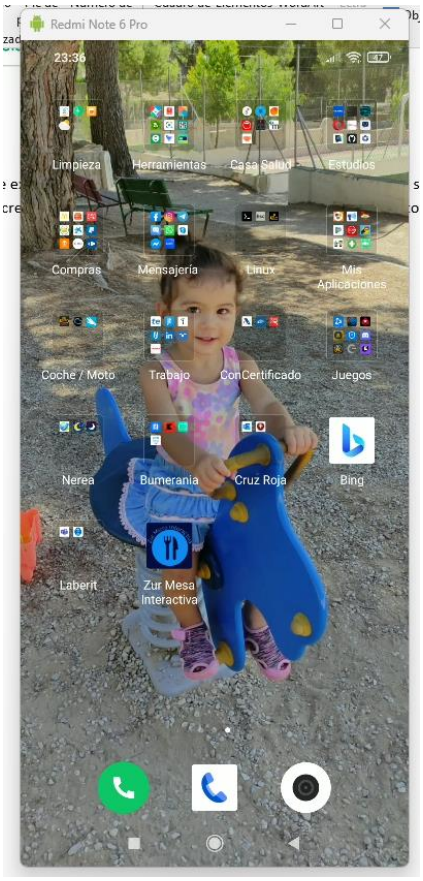
adb connect 192.168.0.15

```
G:\Mi unidad\Cursos - FP - Master\DAM - Desarrollo de Apli
caciones Multiplataforma\Proyecto\Mesa\Programas usados\sc
>adb connect 192.168.0.15
connected to 192.168.0.15:5555
```

Tras realizar la primera conexión, ya podemos desconectar el cable usb y escribir el comando:

scrcpy --bit-rate 2M --max-size 800

```
G:\Mi unidad\Cursos - FP - Master\DAM - Desarrollo de Apli
caciones Multiplataforma\Proyecto\Mesa\Programas usados\sc
>scrcpy --bit-rate 2M --max-size 800
INFO: scrcpy 1.19 <https://github.com/Genymobile/scrcpy>
G:\Mi unidad\Cursos - FP - ...B/s (37330 bytes in 0.018s)
[server] INFO: Device: Xiaomi Redmi Note 6 Pro (Android 9)
INFO: Renderer: direct3d
INFO: Initial texture: 376x800
```



A partir de aquí ya podemos utilizar le móvil desde el pc.

EXTRAS

Análisis de datos / Bigdata

He mencionado en ciertas partes del proyecto la importancia de recabar datos de los clientes a la hora de realizar los pedidos.

Gracias a estos datos podemos obtener información muy valiosa como que:

- producto es el más consumido.
- productos que suelen pedirse juntos.
- días que se realizan más pedidos.

Gracias a esto podemos:

- realizar ofertas combinadas de productos.
- calcular el stock necesario de cada producto.

Mi idea era realizar un script que descargase la última relación de pedidos manipulase los datos, y enviase un informe por correo electrónico, no he tenido tiempo de investigar cómo realizarlo, por lo que podría ser una tarea pendiente para incluir en las mejoras de la aplicación.

He preparado un Jupyter notebook, en pySpark, con el cual descargo los datos del último informe generado (por fecha), que esta almacenado en Azure y realizo una pequeña limpieza y transformación con los datos.

Los datos de los pedidos son inventados por lo que el resultado se aleja de una posible realidad.

Se adjunta con la documentación entregable el cuaderno de Jupyter con el código.

A continuación, se muestran algunos ejemplos de las transformaciones realizadas.

Ordenados de mayor a menor, los productos más consumidos.

```
df_producto = df_producto.orderBy("cantidad", ascending=False)
df_producto.show()
```

✓ 18.6s

```
+--------------------+--------+
|            producto|cantidad|
+--------------------+--------+
|     Cerveza Alambra|      76|
|        Agua Mineral|      61|
|           Coca Cola|      58|
|Cerveza Mahou Mae...|      54|
|                Cafe|      53|
|      Coca Cola Zero|      51|
|       Fanta Naranja|      31|
|                  Te|      28|
|            Aquarius|      25|
|             Platano|      22|
|   Cerveza Mahou 0.0|      22|
|              Nestea|      22|
|         Nestea Zero|      19|
|Cerveza Mahou Ran...|      19|
|       Cerveza Mahou|      18|
|     Pizza Pepperoni|      18|
|             Naranja|      17|
|       Flan de huevo|      17|
|              Sandia|      17|
|      Pizza Hawaiana|      16|
+--------------------+--------+
```

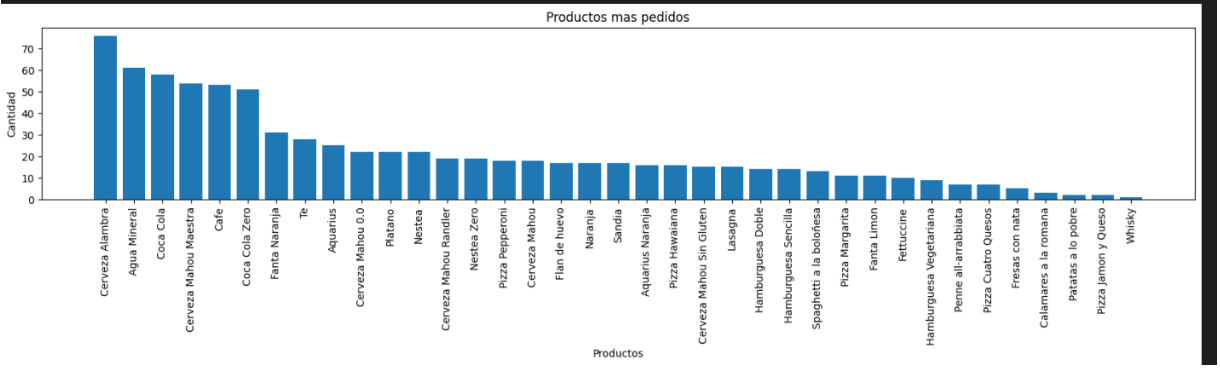
Un gráfico con los productos.

```python
import matplotlib.pyplot as plt
import numpy as np

#Obtenemos los datos de la columna alimento df_alimentos
alimentos = df_producto.select("producto").rdd.flatMap(lambda x: x).collect()

#Grafico de columnas con los alimentos
plt.figure(figsize=(20, 3))
plt.bar(alimentos, df_producto.select("cantidad").rdd.flatMap(lambda x: x).collect())
plt.title("Productos mas pedidos")
plt.xlabel("Productos")
plt.ylabel("Cantidad")
plt.xticks(rotation=90) #girar los nombres de los alimentos
plt.show()
```

✓ 24.8s Python



Una lista de los productos cancelados.

```python
# Filtrar por Estado Cancelado
dfEstadoEntregado = dfFechaADate.filter(dfFechaADate.Estado == 'Cancelado')
dfEstadoEntregado.show()
```

✓ 4.2s

```
+---+----------+--------------------+-----+---------+
| Id|     Fecha|    ListadoProductos|Total|   Estado|
+---+----------+--------------------+-----+---------+
|  3|2023-04-11|Hamburguesa Senci...| 9.98|Cancelado|
|  4|2023-04-11|Hamburguesa Senci...|19.96|Cancelado|
|  5|2023-04-11|Cerveza Alambra 2...|20.98|Cancelado|
| 31|2023-03-22|Coca Cola 1 2.0, ...|23.49|Cancelado|
| 32|2023-04-01|Cerveza Mahou Mae...|19.94|Cancelado|
| 33|2023-04-01|Cerveza Mahou Mae...|13.45|Cancelado|
| 35|2023-04-01|Coca Cola 1 2.0, ...|  4.0|Cancelado|
| 36|2023-04-01|Coca Cola 1 2.0, ...|  4.0|Cancelado|
| 37|2023-04-01|Agua Mineral 1 1....|  5.5|Cancelado|
| 38|2023-04-01|Aquarius 1 1.99, ...|17.97|Cancelado|
| 39|2023-04-01|Agua Mineral 1 1....|27.48|Cancelado|
| 46|2023-04-02|Coca Cola Zero 1 ...|17.98|Cancelado|
| 47|2023-04-02|Pizza Hawaiana 1 ...|31.96|Cancelado|
| 50|2023-04-02|Coca Cola 1 2.0, ...|27.98|Cancelado|
| 52|2023-04-02|Agua Mineral 1 1....|22.48|Cancelado|
| 56|2023-04-03|Lasagna 1 10.99, ...|23.47|Cancelado|
| 66|2023-04-03|Agua Mineral 1 1....|17.96|Cancelado|
| 69|2023-04-03|Cerveza Alambra 1...|27.41|Cancelado|
| 70|2023-04-03|Agua Mineral 1 1....|20.43|Cancelado|
| 71|2023-04-03|Agua Mineral 1 1....|20.43|Cancelado|
+---+----------+--------------------+-----+---------+
```

Los 5 pedidos que más se repiten.

```
#me quedo solo con lista de productos
dfListadoProductos = dfquitamosCantidadPrecio.select("ListadoProductos")

#5 pedidos que mas se repiten
dfListadoProductos.groupBy("ListadoProductos").count().orderBy("count", ascending=False).show(5, truncate=False)
```

```
[58]  ✓ 6.7s
```

```
+----------------------------------------------------+-----+
|ListadoProductos                                    |count|
+----------------------------------------------------+-----+
|[Agua Mineral, Coca Cola, Coca Cola Zero]           |6    |
|[Agua Mineral, Coca Cola, Coca Cola Zero, Fanta Naranja]|3    |
|[Cafe, Te]                                          |3    |
|[Coca Cola Zero, Fanta Naranja]                     |2    |
|[Cerveza Alambra, Cerveza Mahou Maestra]            |2    |
+----------------------------------------------------+-----+
only showing top 5 rows
```

Agrupamos por fecha y mostramos las 5 fechas con más pedidos y cuantos pedidos tiene.

```
#Agrupa por fecha y lista de productos
dfEstadoEntregado2 = dfquitamosCantidadPrecio.groupBy('Fecha').agg(collect_list('ListadoProductos').alias('ListadoProductos'))
dfEstadoEntregado2.show(truncate=False)
#Que fecha tiene mas pedidos, cada pedido esta separado por []
dfEstadoEntregado2.withColumn('ListadoProductos', explode('ListadoProductos')).groupBy('Fecha').count().orderBy("count", ascending=False).show(5, truncate=False)
```

```
[65]  ✓ 13.4s
```

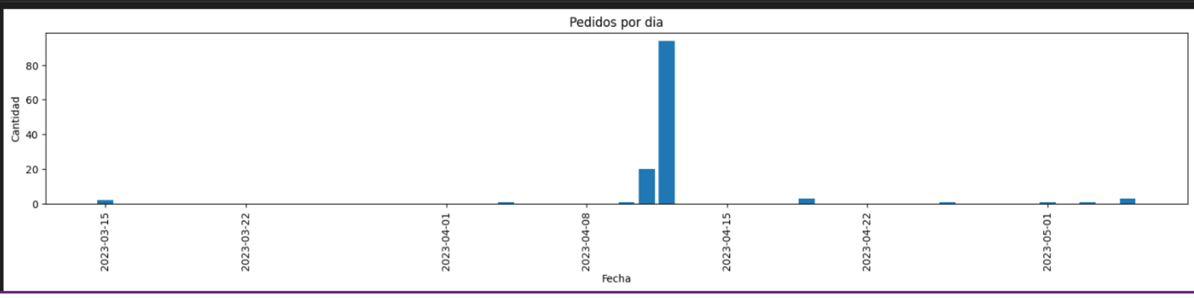
```
+----------+------------------------------------------------------------------------------------------------------------------------------------
|Fecha     |ListadoProductos
+----------+------------------------------------------------------------------------------------------------------------------------------------
|2023-04-12|[[Nestea, Nestea Zero, Spaghetti a la boloñesa, Fettuccine, Cafe, Flan de huevo], [Cerveza Mahou Maestra, Cerveza Alambra, Cerveza Mahou], [Coca Cola, Coca Cola Zero,
|2023-04-11|[[Cerveza Mahou Randler, Cerveza Alambra], [Aquarius, Aquarius Naranja, Spaghetti a la boloñesa, Penne all-arrabbiata], [Coca Cola, Coca Cola Zero, Fanta Naranja], [C
|2023-03-15|[[Hamburguesa Sencilla, Coca Cola Zero, Hamburguesa Sencilla, Hamburguesa Sencilla], [Cerveza Mahou Randler, Agua Mineral, Coca Cola]]
|2023-04-10|[[Cerveza Mahou Sin Gluten, Cerveza Alambra, Hamburguesa Doble, Hamburguesa Sencilla]]
|2023-05-03|[[Coca Cola]]
|2023-04-04|[[Cerveza Alambra, Cerveza Mahou Maestra, Cerveza Mahou, Nestea Zero, Nestea]]
|2023-05-01|[[Agua Mineral]]
|2023-05-05|[[Agua Mineral, Cerveza Alambra], [Agua Mineral, Coca Cola], [Agua Mineral, Coca Cola, Coca Cola Zero]]
|2023-04-19|[[Agua Mineral, Coca Cola, Penne all-arrabbiata], [Calamares a la romana, Patatas a lo pobre, Lasagna], [Coca Cola Zero, Hamburguesa Doble]]
|2023-04-26|[[Coca Cola, Coca Cola Zero]]
+----------+------------------------------------------------------------------------------------------------------------------------------------

+----------+-----+
|Fecha     |count|
+----------+-----+
|2023-04-12|94   |
|2023-04-11|20   |
|2023-05-05|3    |
|2023-04-19|3    |
|2023-03-15|2    |
+----------+-----+
only showing top 5 rows
```

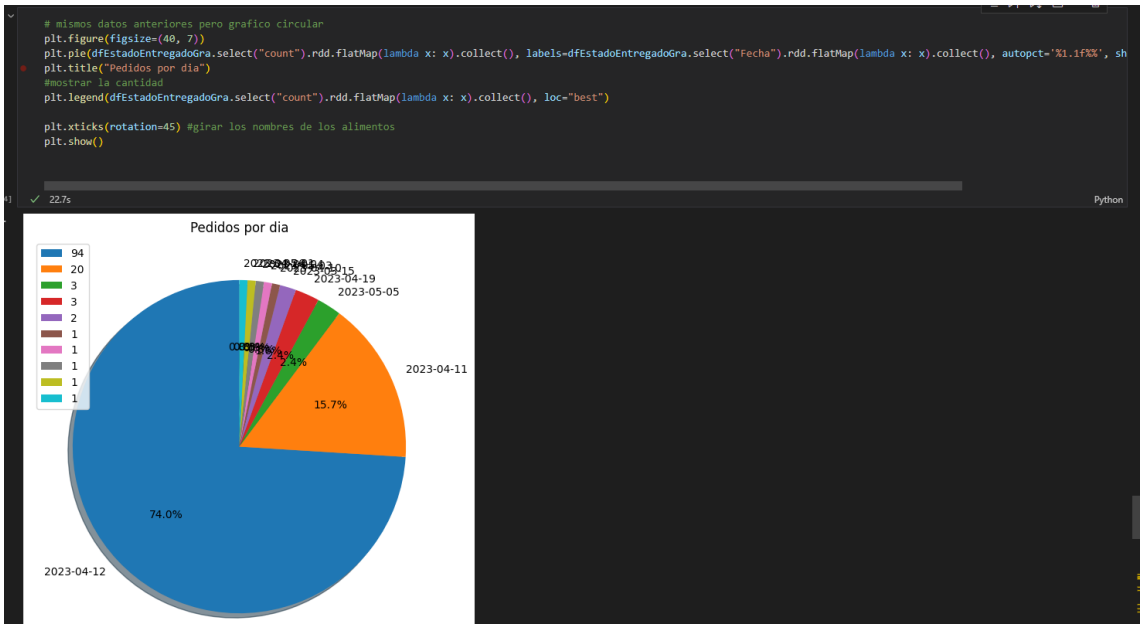
Grafica con la cantidad de pedidos por día (*El problema de mostrar este grafica es que no se han introducido pedidos diariamente, por lo que se ve un poco vacía*).

```
#Mostrar una grafica con dfEstadoEntregado2 con la cantidad de pedidos,pero solo muestra esos 5 dias
dfEstadoEntregadoGra = dfEstadoEntregado2.withColumn('ListadoProductos', explode('ListadoProductos')).groupBy('Fecha').count().orderBy("count", ascending=False)
#Crear grafico
plt.figure(figsize=(20, 3))
plt.bar(dfEstadoEntregadoGra.select("Fecha").rdd.flatMap(lambda x: x).collect(), dfEstadoEntregadoGra.select("count").rdd.flatMap(lambda x: x).collect())
plt.title("Pedidos por dia")
plt.xlabel("Fecha")
plt.ylabel("Cantidad")
plt.xticks(rotation=90) #girar los nombres de los alimentos
plt.show()
```

```
[76]  ✓ 14.5s                                                Python
```



Estos datos se pueden tener una mejor visualización en un gráfico circular.



Estos son algunos ejemplos de todos los datos que podríamos mostrar.