# PROYECTO FINAL

# COCINA EXPLOSIVA

*GRUPO N° 20*

*Braian Urra, Sebastian Vega, Cristobal Gonzalez*

# ÍNDICE:

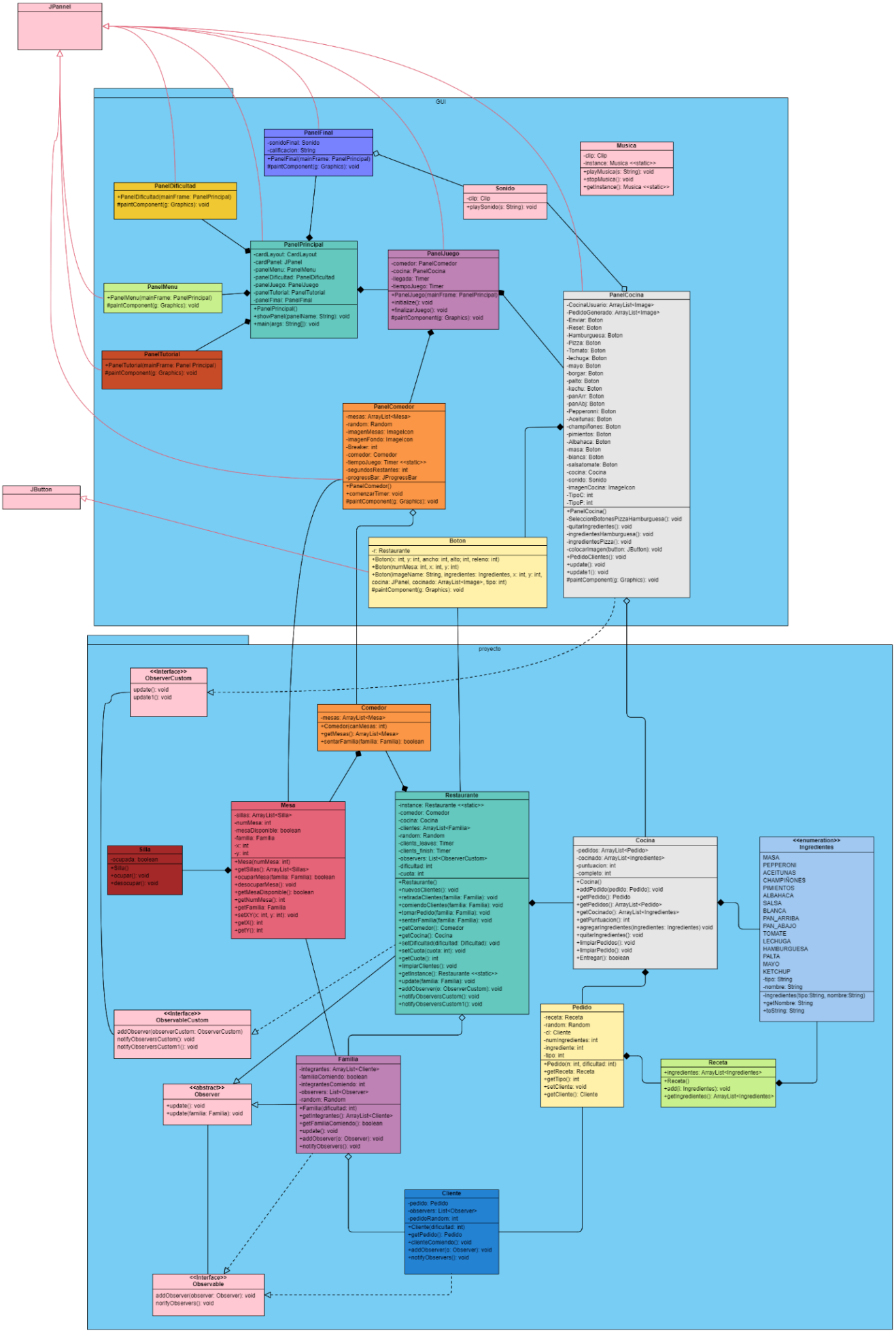
# INTRODUCCIÓN

Como nuestro proyecto final, decidimos crear un restaurante estilo Simulador de Cocina. decidimos desarrollar esta idea por nuestro interés por los videojuegos y los juegos de simulación. El objetivo principal de este proyecto es utilizar e integrar el conocimiento que obtuvimos en el curso para la creación de un proyecto que demuestra nuestra capacidad para crear e implementar lo aprendido.

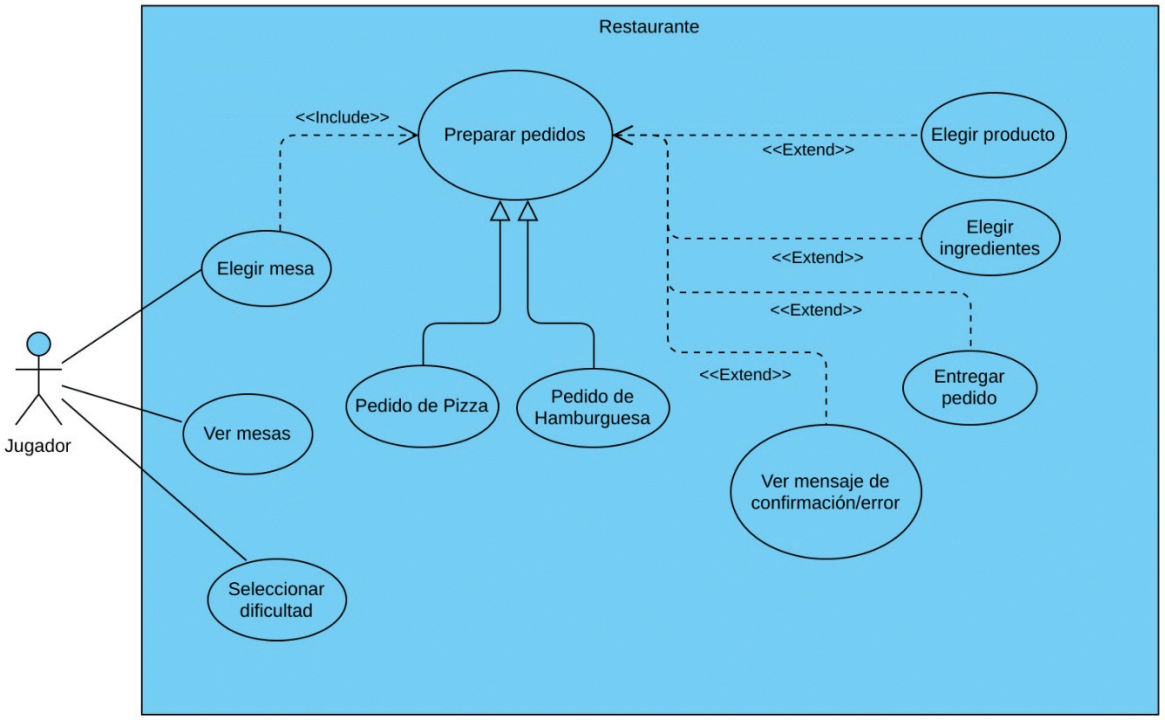
Durante el desarrollo de este proyecto, nos encontramos con diversos desafíos que nos permitieron seguir aprendiendo y creciendo como diseñadores. Desde la planificación y el diseño iniciales hasta la codificación y la resolución de problemas, cada sesión brinda la oportunidad de aplicar los conocimientos aprendidos en el aula.

Este informe recopila el proceso de desarrollo de nuestro proyecto, incluidos diagramas UML, casos de uso, diagramas de interfaz y presenta las decisiones tomadas y los problemas involucrados.

# DIAGRAMA UML



# DIAGRAMA CASOS DE USO



# PATRONES DE DISEÑO UTILIZADOS

En el proyecto se hizo uso de 3 patrones de diseños distintos, los cuales fueron

- Observers
- Singleton
- Refactoring

**Observer:** Este patrón fue esencial para manejar las dependencias entre los diferentes objetos en el simulador de restaurante. Permite que los cambios en el estado de los clientes y familias se propagen automáticamente a las clases correspondientes, facilitando la actualización de la interfaz y la lógica de negocio, esto nos evitó bastantes problemas a la hora de los pedidos.

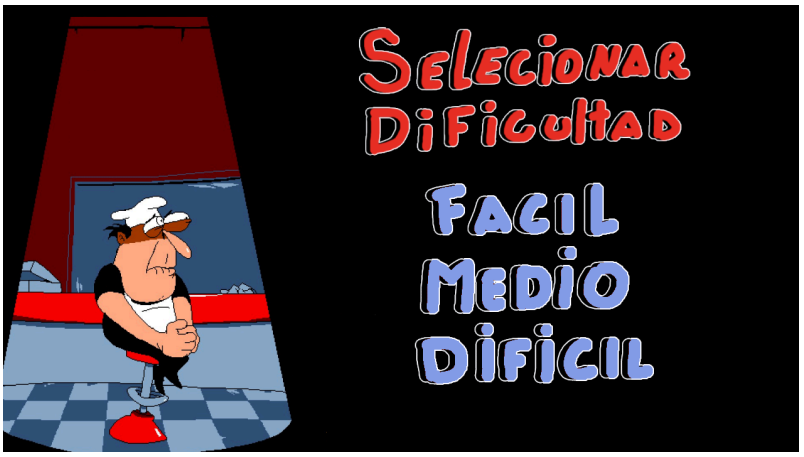
**Singleton:** Implementar Restaurante como un Singleton era bastante necesario para el código ya que, al ser una única instancia el poder acceder a él desde varias partes del código sin necesidad de pasarlo como parámetro ayudó mucho a evitar confusiones y complicaciones con el código. Con la Música ocurre algo similar, lo más conveniente en el momento fue hacerlo un Singleton, ya que como solo suena una única música de fondo, el poder cambiarla desde cualquier parte igualmente ayudó a evitar cambiar esta clase y ahorrarnos tiempo.

Al igual, se usó el patrón de Refactoring cuando se llegó alrededor del 50% del código ya que se generaron errores al implementar métodos en la GUI que deberían estar en la lógica del código, esto generó cambios en la mayoría de clases. Luego se volvió a implementar al finalizar el proyecto para entender mucho mejor el código y hacer un seguimiento correcto.

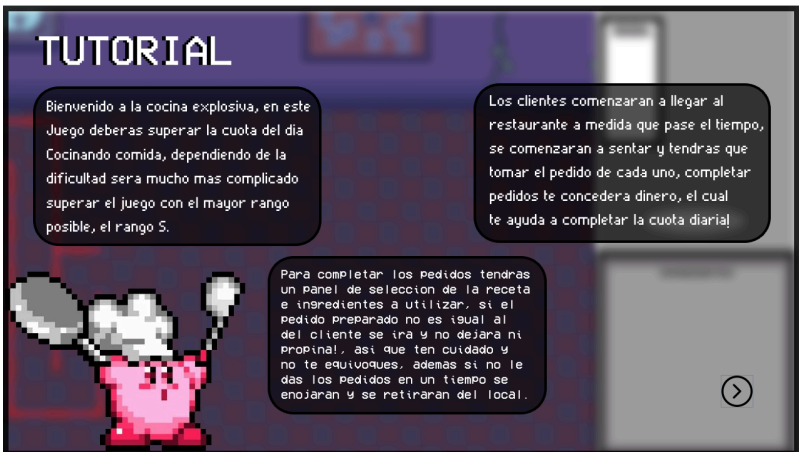
# DEMOSTRACIÓN INTERFAZ

## Panel de Menú


## Panel de dificultad

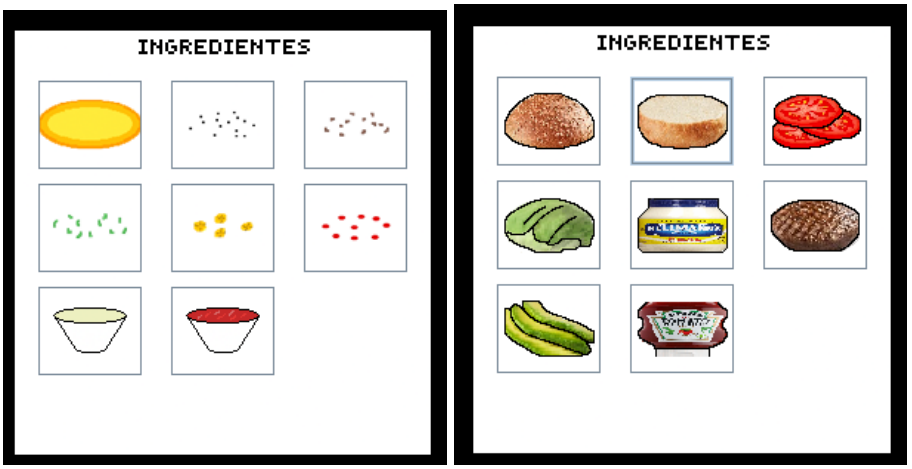



## Panel de tutorial



## Panel de Juego


## Panel de Juego 2




*(Panel de distintos ingredientes)*

## Panel Final


*(La calificación se crea con swing, al igual que la puntuación, no están incluidas en el fondo)*

# DECISIONES TOMADAS

***Decision 1#:***

Al principio del código se tenía pensado que uno pudiera sentar a los clientes individualmente pero se nos complicó demasiado sentar clientes individualmente ya que no logramos trabajar bien el arraylist de clientes con la cocina, entonces uno solo apretar un botón y se sentaba.

***Decisión 2#:***

Eliminamos la opción de sentar a los clientes por problemas con tiempo ya que no habíamos definido muy bien los tiempos entonces habían muchos involucrados lo que generaba que había actualizar muchas cosas en el código entonces ahora los clientes en cuanto llegaron se sentaron.

***Decision 3#:***

Se cambio los clientes por familias ya que no tenía sentido que se sentaran extraños con extraños por lo cual ahora familia sería un arraylist con los clientes

***Decisión 4#:***

Por la decisión 2 como ya no se interactuaba de ninguna forma con ellos decidimos que los pedidos se tomarán en la mesa

# PROBLEMA Y AUTOCRÍTICA

Uno de los grandes problemas que tuvimos al principio fue el tiempo. A pesar de que luego se pudo solucionar un poco gracias a alargar el plazo, la procrastinación fue perjudicial para el trabajo. Además, los certámenes nos tuvieron un poco preocupados, lo que hizo que dejáramos el trabajo de lado. Esta situación nos hizo reflexionar sobre la importancia de una mejor gestión del tiempo y la necesidad de evitar la procrastinación para asegurar un progreso constante y efectivo en nuestros proyectos.

Otra dificultad fue nuestra inexperiencia en la creación de juegos. Esto, sumado a lo menos permisivo que es Swing para este tipo de desarrollo, complicó aún más nuestro trabajo. La búsqueda de herramientas más adecuadas para el desarrollo de juegos es fundamental, eso nos hubiera asegurado un progreso constante y efectivo en nuestros proyectos.