



UNIVERSIDAD DE LOS LAGOS

## Proyecto semestral- Sprint 2 Sistema de Reserva Restaurante

**Integrantes:** Roberto Azua, Nicolás Almuna,  
Felipe Delgado, Benjamín Miranda, Daniel  
Sepúlveda, Felipe Vera

**Asignatura:** Taller Programación aplicada

**Carrera:** Ingeniería Civil en Informática

**Docente:** Joel Torres

## Índice

|  |           |
|--|-----------|
| <b>Introducción.....</b>                         | <b>3</b>  |
| <b>Diagrama de clases.....</b>                   | <b>4</b>  |
| <b>Diseños de<br/>interfaces/Wireframes.....</b> | <b>5</b>  |
| <b>Funcionamiento Código.....</b>                | <b>8</b>  |
| <b>Conclusion.....</b>                           | <b>9</b>  |
| <b>Bibliografía.....</b>                         | <b>10</b> |

## Introducción

El desarrollo de cualquier programa requiere el diseño de una interfaz efectiva y fácil de usar. Esta etapa de diseño implica considerar tanto la apariencia visual como la usabilidad de la interfaz que acompañará a nuestro software. Aunque aún no hemos recibido una interfaz específica para trabajar en la funcionalidad asignada a este sprint, hemos adjuntado algunas propuestas de interfaces en las cuales podríamos implementar el código y las funcionalidades desarrolladas durante este periodo.

La pantalla principal de la interfaz propuesta se caracteriza por:

- Mostrar de manera clara los tres botones principales: “Agregar reserva”, “Modificar reserva” y “Confirmar disponibilidad”.
- Presentar información relevante en pantalla, como los tipos de comidas disponibles según la reserva, el plan de degustación o el plan seleccionado.
- Permitir a los usuarios acceder fácilmente a las acciones necesarias y visualizar los detalles de su reserva, así como las opciones de comida disponibles según sus preferencias y elecciones.

Al presionar el botón “Agregar reserva”, se desplegará una nueva ventana que permitirá seleccionar los datos necesarios para realizar una reserva, tales como:

- El tipo de reserva
- La modalidad de la reserva (solo disponible si se selecciona el tipo de reserva como “evento”)
- El plan de comida
- El plan de degustación
- Los sectores a reservar
- Las indicaciones proporcionadas por el cliente

Esta funcionalidad garantiza que las opciones presentadas sean pertinentes y relevantes según el tipo de reserva elegido, lo que asegura una experiencia intuitiva y sin confusiones para el usuario.

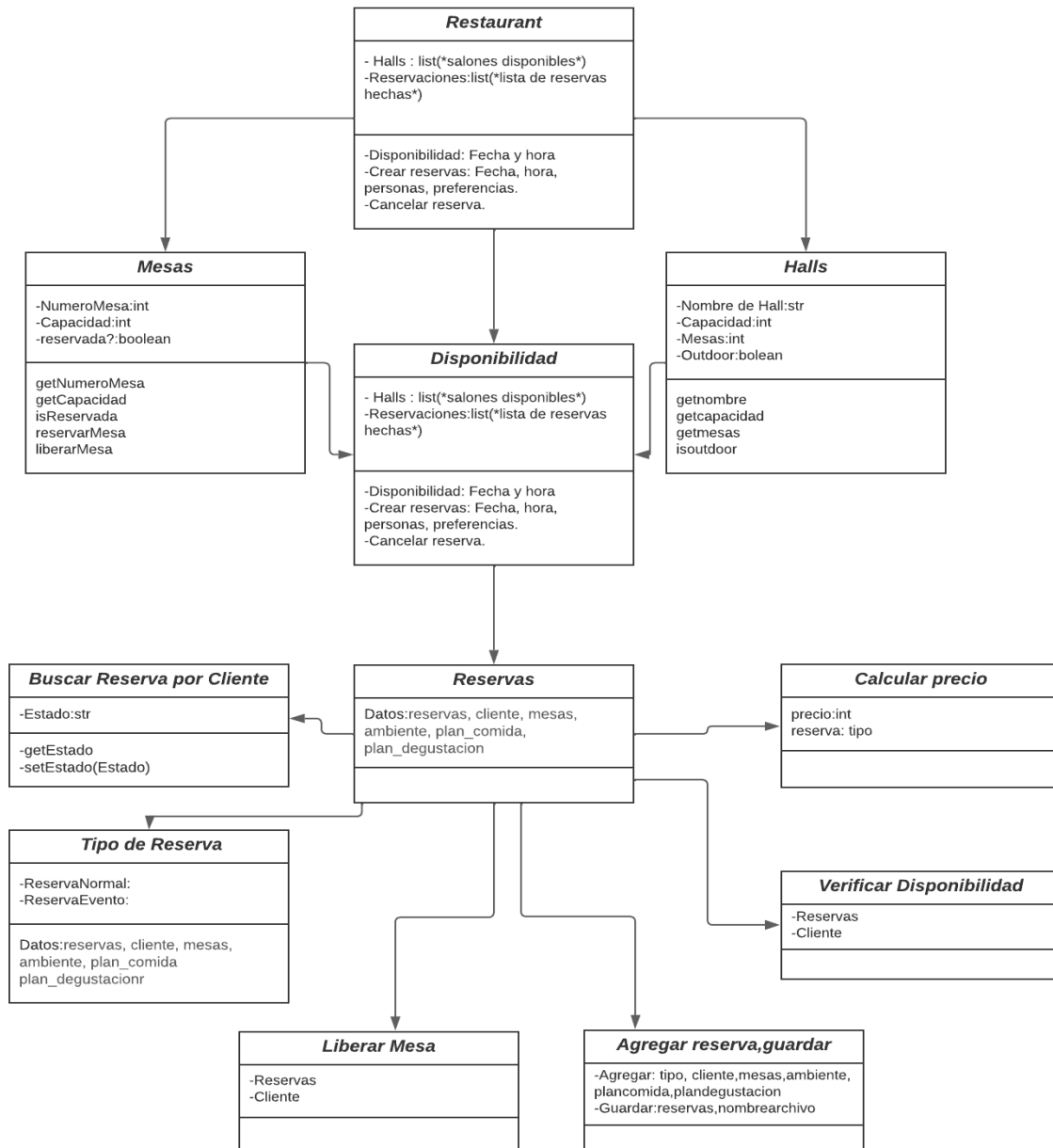
Al presionar el botón “Modificar reserva”, se abrirá una nueva ventana que mostrará las reservas registradas previamente, organizadas según el tipo de reserva. Al seleccionar una reserva disponible, se mostrarán los detalles correspondientes. A partir de esta selección, el usuario podrá elegir entre dos opciones: “Eliminar” y “Editar”.

Si el usuario hace clic en el botón “Eliminar”, se desplegará una ventana de alerta para confirmar si se desea eliminar la reserva o cancelar la acción.

Si el usuario selecciona el botón “Editar”, se abrirá una ventana prácticamente idéntica a la de agregar reserva, pero con los datos previamente completados y la posibilidad de editarlos. Una vez realizadas las modificaciones deseadas, al presionar el botón “Actualizar”, los nuevos datos se guardarán y la reserva se actualizará con los cambios correspondientes. La reserva seguirá siendo visible en la lista, pero con los datos actualizados.

## Diagrama de Clases

Al comenzar un proyecto, lo primero que se debe de realizar es un diagrama de clases para así comprender como va ser la estructura y las funcionalidades que queremos implementar en nuestro código, de este modo nos entrega una visión más gráfica sobre las clases que tendremos y como se relacionan entre ellas.



En el diagrama podemos ver la clase principal Restaurante, la cual es la que contiene la información principal sobre las clases Halls y mesas del restaurante, a la vez estas clases verifican su disponibilidad de mesas y halls con la clase Disponibilidad, donde si las hay, esto pasa a la clase Reserva la cual tiene distintas funcionalidades que estarán implementadas en el código, tales como agregar y guardar reserva, verificar disponibilidad, tipo de reserva, entre otros. De esta manera se asegura que la funcionalidad del programa de reservas esté clara, y esperamos pueda ser más comprensible para las siguientes personas que se adjunten a este proyecto.

## Diseño de interfaces /Wireframes

Como parte esencial de la etapa de diseño, resulta fundamental considerar la apariencia y la usabilidad de la interfaz de nuestro programa. Aunque no se haya recibido una interfaz específica para trabajar en la funcionalidad asignada a este sprint, adjuntamos algunas propuestas de interfaces donde se podría implementar el código y las funcionalidades desarrolladas en este periodo.

### -Pantalla principal:

La interfaz propuesta muestra de forma clara los tres botones principales que se utilizarán: "**Agregar reserva**", "**Modificar reserva**" y "**Confirmar disponibilidad**". Además, se presenta información relevante en pantalla, donde es posible mostrar los tipos de comidas según la reserva, el plan de degustación o el plan seleccionado. Esto permitirá a los usuarios acceder fácilmente a las acciones necesarias y visualizar los detalles de su reserva, así como las opciones de comida disponibles según sus preferencias y elecciones.

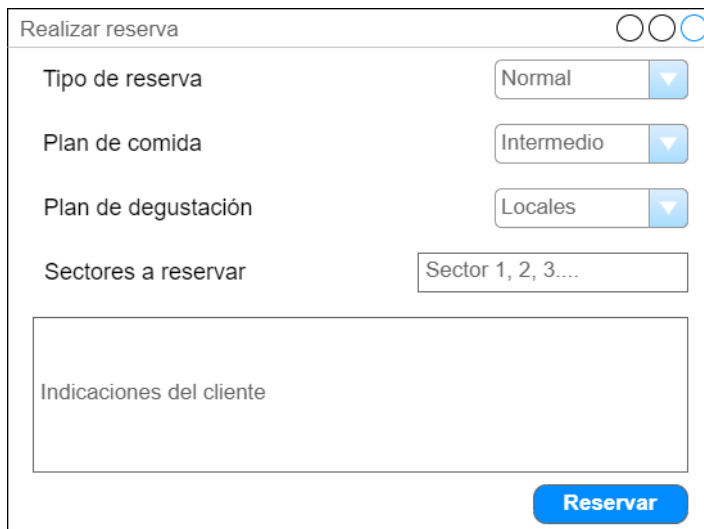


## -Botón de “Agregar reserva”

Cuando se presiona el botón "Agregar reserva", se desplegará una nueva ventana que permitirá seleccionar los datos necesarios para la reserva. Estos datos incluirán el **tipo de reserva**, la **modalidad de la reserva**, el **plan de comida**, el **plan de degustación**, los **sectores a reservar** y las **indicaciones proporcionadas por el cliente**.

Es importante destacar que la opción de **modalidad de reserva** solo estará disponible si se selecciona el tipo de reserva como "**evento**". En caso contrario, esta opción no se mostrará en la interfaz. Esta funcionalidad asegurará que las opciones presentadas sean pertinentes y relevantes según el tipo de reserva elegido.

De esta manera, la interfaz adaptará dinámicamente las opciones disponibles según las selecciones previas del usuario, lo que garantizará una reserva más intuitiva y libre de confusiones.



Realizar reserva

Tipo de reserva: Normal

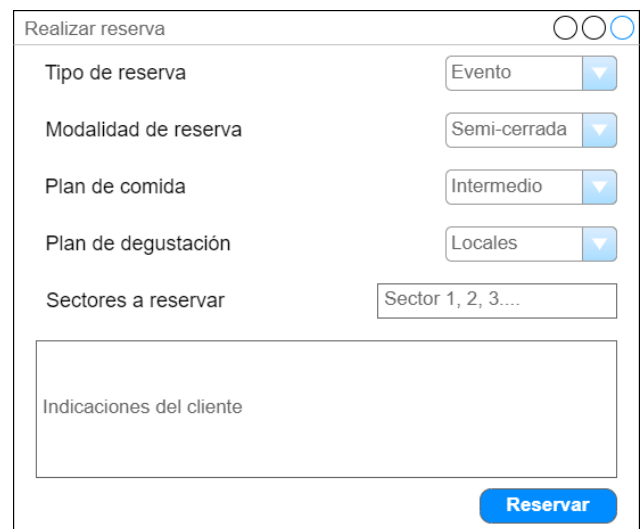
Plan de comida: Intermedio

Plan de degustación: Locales

Sectores a reservar: Sector 1, 2, 3....

Indicaciones del cliente

Reservar



Realizar reserva

Tipo de reserva: Evento

Modalidad de reserva: Semi-cerrada

Plan de comida: Intermedio

Plan de degustación: Locales

Sectores a reservar: Sector 1, 2, 3....

Indicaciones del cliente

Reservar

## -Botón de “Modificar reserva”

Al presionar el botón "Modificar reserva", se abrirá una nueva ventana que mostrará las reservas registradas previamente, organizadas según el tipo de reserva. Al seleccionar una reserva disponible, se mostrarán los detalles de dicha reserva.

Una vez seleccionada una reserva, se presentarán dos opciones: "Eliminar" y "Editar". Al hacer clic en el botón "Eliminar", se desplegará una ventana de alerta para confirmar si se desea eliminar la reserva o cancelar la acción.

Por otro lado, al seleccionar el botón "Editar", se abrirá una ventana prácticamente idéntica a la de agregar reserva, pero con los datos previamente completados y la posibilidad de editarlos. Una vez realizadas las modificaciones deseadas, al presionar el botón "Actualizar", los nuevos datos se guardarán y la reserva se actualizará con los cambios correspondientes. La reserva seguirá siendo visible en la lista, pero con los datos actualizados.

## Funcionamiento Código

```
1
2 def cargar_reservas(self):
3     reservas = []
4     with open(self.nombre_archivo, 'r') as archivo:
5         lector_csv = csv.reader(archivo)
6         for fila in lector_csv:
7             reserva = {
8                 'tipo': fila[0],
9                 'cliente': fila[1],
10                'mesas': int(fila[2]),
11                'ambiente': fila[3],
12                'plan_comida': fila[4],
13                'plan_degustacion': fila[5]
14            }
15            reservas.append(reserva)
16        return reservas
17
18 def guardar_reservas(self):
19     with open(self.nombre_archivo, 'w', newline='') as archivo:
20         escritor_csv = csv.writer(archivo)
21         for reserva in self.reservas:
22             fila = [
23                 reserva['tipo'],
24                 reserva['cliente'],
25                 str(reserva['mesas']),
26                 reserva['ambiente'],
27                 reserva['plan_comida'],
28                 reserva['plan_degustacion']
29             ]
30             escritor_csv.writerow(fila)
31
```

Para fines de este Sprint al código se le agregó las funciones para leer y escribir dentro del archivo .csv



## Conclusión

El diseño de la interfaz propuesta se enfoca en proporcionar una experiencia intuitiva y fácil de usar para los usuarios. A través de la disposición clara de botones y la adaptación dinámica de las opciones según las selecciones previas del usuario, se facilita la realización de reservas y la gestión de las mismas. La combinación de una apariencia atractiva y una interfaz funcional contribuirá a mejorar la usabilidad de nuestro programa y, en última instancia, la satisfacción de los usuarios.

## Bibliografía

- [Diagrama de clases,TPA GRUPO 5: Lucidchart](#)
- <https://drive.google.com/file/d/1B3llet5t-iMiEd9WbVGXKG61pfgrugS/view?usp=sharing>
- [https://github.com/feveja/tpa\\_grupo5.git](https://github.com/feveja/tpa_grupo5.git)