

<Karendaki>

---

## Informe Proyecto

---

<Christian Alejandro Torres  
Hernandez,Hector Roman Robles Orantes>



Universidad Politécnica  
de Chiapas  
Tecnología para el bien común

## Tabla de contenido

<b>ÍNDICE DE SEGMENTOS DE CÓDIGO</b>	<b>2</b>
<b>1 EMPRESA O INSTITUCIÓN</b>	<b>3</b>
<b>2 DESCRIPCIÓN DEL PROYECTO</b>	<b>4</b>
<b>3 EXPECTATIVAS ACADÉMICAS</b>	<b>4</b>
<b>4 DIAGNÓSTICO SITUACIONAL</b>	<b>5</b>
<b>5 DESARROLLO DE PROYECTO</b>	<b>5</b>
<b>6 RESULTADOS</b>	<b>21</b>
<b>7 LECCIONES APRENDIDAS</b>	<b>22</b>
<b>8 COMPETENCIAS ADQUIRIDAS</b>	<b>22</b>
<b>9 CONCLUSIONES</b>	<b>23</b>

## 1 Empresa, Institución o Negocio

---

BashLashes, es un negocio del sector de Belleza, se dedican específicamente a colocar pestañas, se encuentra ubicado en Tuxtla Gutiérrez Chiapas.

En BashLashes, entendemos que tus pestañas son una parte esencial de tu estilo personal, por lo que nos esforzamos por brindarte opciones que se adapten a tus necesidades. Ya sea que deseas un aspecto natural y sutil o un estilo más dramático y atrevido



Imagen 1.1 Fachada del negocio BashLashes

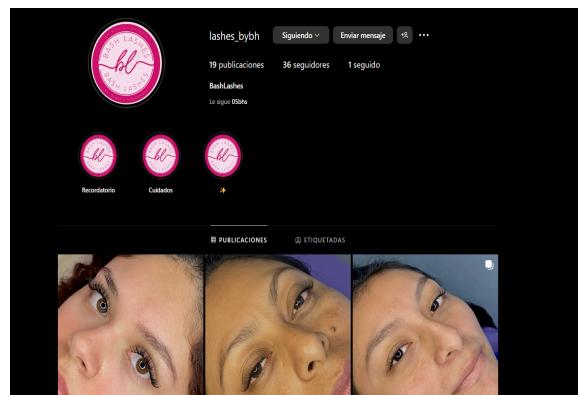


Imagen 1.2 Redes Sociales del negocio

## **2 Descripción del proyecto**

---

El crecimiento de los negocios implica la automatización de actividades relacionadas con citas y la satisfacción de las necesidades del cliente para facilitar el desarrollo de la empresa, desde los procesos más simples hasta los más complejos. Esta automatización se enfoca en optimizar la gestión de citas, garantizando una experiencia fructífera y satisfactoria para los clientes, al tiempo que libera recursos y tiempo para dedicarse a otras áreas de la empresa. Al implementar sistemas de gestión de citas y reportes de ventas óptimos, las empresas pueden mejorar la calidad de sus servicios, fortalecer su reputación, lo que contribuye significativamente a su crecimiento y éxito a largo plazo.

## **3 Expectativas Académicas**

---

- Antes de realizar el proyecto, pensaba que no iba a ser tan difícil el manejo de fechas como lo es un calendario.
- En la búsqueda de información para sacar adelante el proyecto no eran explícitos como creí que iba a ser al buscar información sobre la resolución de la problemática.
- Pensé que se podía realizar objetos de clases abstractas.
- Pensé que no nos quitaría nada de tiempo heredar.
- Pensé qué íbamos a usar solo dos arraylist.
- Pensé que no íbamos a usar el mismo código para todo.
- Llegar a comprender sobre los programas que podemos ofrecer.
- Conocer el formato de cómo se trabaja un proyecto profesional.

## **4 Diagnóstico Situacional**

---

No sabíamos que se podría realizar objetos de clases abstractas, se pensaba que si se podían hacer, pero al momento del desarrollo del proyecto conforme a los días, nos dimos cuenta que no se puede, y eso nos hizo perder tiempo.

Implementación de herencia en para el desarrollo del proyecto, esto se nos dificultó porque la idea original de herencia era diferente a la que al final si se realizó, así que consideramos que debemos de tomarnos el tiempo necesario para poder delimitar la idea de la herencia.

Nos faltó un poco el hecho de tener un guía en el cual poder basar nos en cómo podría llegar a hacer un programa más completo.

## **5 Desarrollo de Proyecto**

---

Al principio del proyecto nos basamos de la información que pudimos obtener del cliente a si de algunos proyectos similares que tuvimos anteriormente para a si poder ir a la parte del análisis, donde pudimos iniciar con las casos de uso donde vislumbramos el alcance que podría llegar a tener debido a que pudimos observar que tendría unas pocas vistas pero con bastantes funciones, después con los casos de uso pudimos observar que herramientas eran las que se requerían y a cuales deberíamos de enfocarnos mas, debido a que son los que mayor uso se les dará, siguiendo a si pudimos generar la estructura de como se vería el código creado los diagramas de clases donde pudimos, generar varias clases en las cuales dividimos las funciones y poder volver a agruparlas dentro de cada clase al igual que ponerle los atributos que cada clase debe de tener, lo cual resultó un poco difícil pero sin duda lo mas complicado fue la implementación de la herencia y las clases abstractas, sin duda alguna eso nos generó un problema debido a que no daba mucha libertad el hecho de elegir en donde ponerlo, no solo eso también

nos complicó más la cuestión que debería de ser abstracto por lo que tuvimos que modificar los diseños varias veces.

## D. Caso de Uso

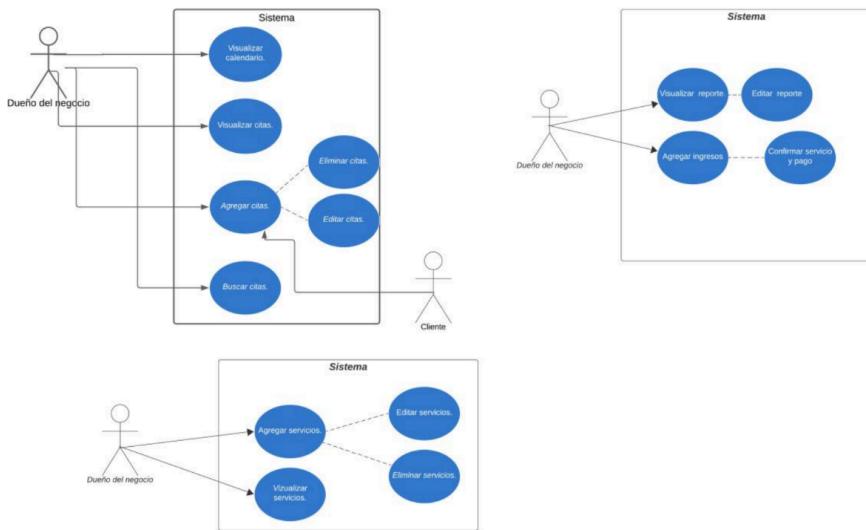


Imagen 2.0 Diagramas UML: Caso de uso.

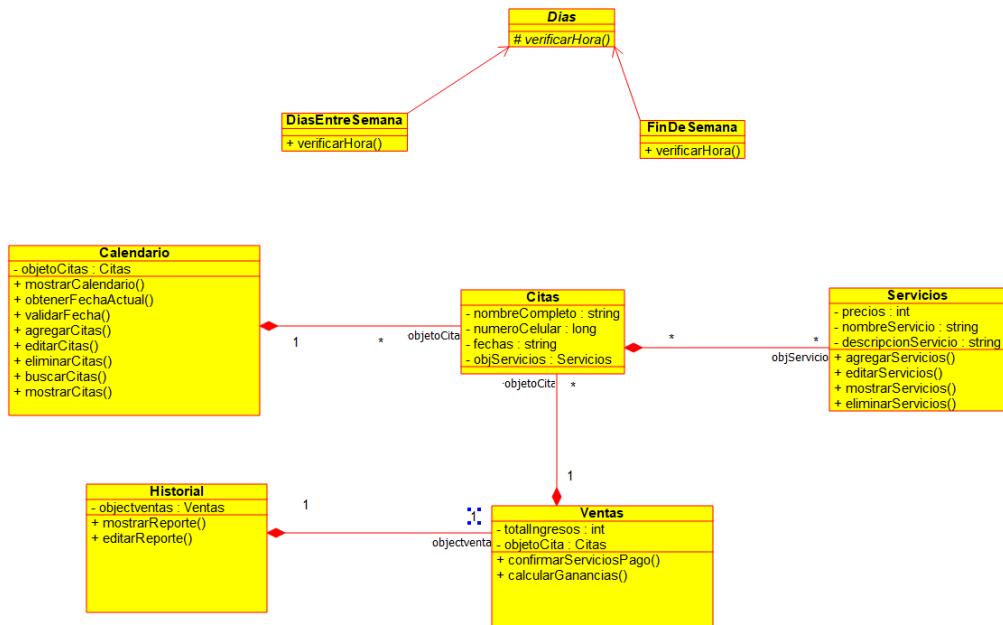


Imagen 2.1 Diagramas UML: Diagrama de clases

Generales	
Clave	<HU1>
Título	< Mostrar calendario >
Puntos	<Christian Alejandro Torres Hernández>

Descripción	
Como:	Administrador
Quiero:	que se pueda visualizar un calendario en el cual poder ubicar mis fechas.
Para:	tener en cuenta cuando asigne una cita, y así poder guiarme de las fechas.

Criterios de aceptación		
<b>Se deberá utilizar lenguaje Gherkin:</b>		
1.	<visualizar calendario>	<p><b>Dado que</b> el usuario ha accedido al programa  <b>Cuando</b> el usuario quiere checar la fecha actual, futuras y pasadas.  <b>Entonces</b> el usuario al ingresar podrá ver en un apartado el calendario el cual mostrará la fecha actual vista por medio de un mes, en el que también podrá visualizar sus citas donde ya registro alguna fecha.</p>

### Imagen 3.0 Historia de usuarios: Mostrar Calendario.

Generales	
Clave	<HU2>
Título	< Mostrar Fecha actual.>
Puntos	<Christian Alejandro Torres Hernández>

Descripción	
Como:	administrador.
Quiero:	poder visualizar la fecha actual.
Para:	poder visualizar las fechas en la que están las citas

Criterios de aceptación		
<b>Se deberá utilizar lenguaje Gherkin:</b>		
1.	<visualizar fecha actual>	<p><b>Dado que</b> el usuario ha accedido al programa  <b>Cuando</b> el usuario quiere checar la fecha en la que se ubica  <b>Entonces</b> el usuario podrá ver el día en el que se encuentra debido a que este será marcado de un color diferente al de los demás días.</p>

### Imagen 3.2 Historia de usuarios: Mostrar Fecha Actual.

Generales	
Clave	<HU3>
Título	< Mostrar citas.>
Puntos	<Christian Alejandro Torres Hernández>
Que se muestre las citas ya ingresadas.	
Descripción	
<p>Como: administrador.</p> <p>Quiero: poder visualizar todas las citas que ya se agregaron.</p> <p>Para: poder visualizar las fechas en las que ya hay citas.</p>	
Criterios de aceptación	
<p><b>Se deberá utilizar lenguaje Gherkin:</b></p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;visualizar citas&gt;</p> <p><b>Dado que</b> el usuario ha accedido al programa  <b>Cuando</b> el usuario quiere checar que citas tiene registradas  <b>Entonces</b> el usuario al ingresar podrá ver en el calendario todas las citas que ya tiene registra.</p>

Imagen 3.3 Historia de usuarios: Mostrar citas.

Generales	
Clave	<HU4>
Título	< Agregar citas.>
Puntos	<Christian Alejandro Torres Hernández>
Que pueda agregar nuevas citas.	
Descripción	
<p>Como: Administrador</p> <p>Quiero: agregar citas del negocio BashLashes.</p> <p>Para: tener un control de las citas y no perderlas</p>	
Criterios de aceptación	
<p><b>Se deberá utilizar lenguaje Gherkin:</b></p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;Agregar citas&gt;</p> <p><b>Dado que</b> el usuario quiere agregar una nueva cita  <b>Cuando</b> vaya a agregar una cita en apartado del programa  <b>Entonces</b> el usuario deberá ingresar los datos del cliente (nombre, numero) y podrá ver los servicios disponibles.</p>
2.	<p>&lt;costos vacíos y negativos&gt;</p> <p><b>Dado que</b> el usuario quiere agregar una nueva cita  <b>Cuando</b> ingrese los costos vacíos o negativos  <b>Entonces</b> se mostrará un mensaje de error correspondiente.</p>
3.	<p>&lt;sin servicios &gt;</p> <p><b>Dado que</b> el usuario quiere agregar una nueva cita  <b>Cuando</b> quiera registrar la cita sin servicios  <b>Entonces</b> se mostrará un mensaje de error correspondiente.</p>

Imagen 3.4 Historia de usuarios:Agregar citas.

Generales	
Clave	<HU5>
Título	< Editar citas.>
Puntos	<Christian Alejandro Torres Hernández>
Descripción	
Como: Administrador Quiero: editar las citas que ya fueron ingresadas. Para: si quiero aplazar la fecha de la cita o cambiar el servicio.	
Criterios de aceptación	
<b>Se deberá utilizar lenguaje Gherkin:</b> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;cambiar datos de las citas&gt;</p> <p><b>Dado que</b> el usuario quiera cambiar los datos de una cita  <b>Cuando</b> quiera cambiar la fecha de la cita o el servicio  <b>Entonces</b> el usuario seleccionara la fecha y el servicio a reemplazar para después cambiarla.</p>
2.	<p>&lt;cambiar una cita primero &gt;</p> <p><b>Dado que</b> el usuario quiera cambiar los datos de una cita y luego otra  <b>Cuando</b> el cliente estando en la edición intente editar otra  <b>Entonces</b> se mandará el mensaje de error correspondiente.</p>

Imagen 3.5 Historia de usuarios: Editar citas.

Generales	
Clave	<HU6>
Título	<Eliminar citas>
Puntos	< Christian Alejandro Torres Hernández >
Descripción	
Como: Administrador Quiero: que pueda eliminar las citas cuando, los clientes del negocio o cuando yo quiera cancelar la cita. Para: que cuando me cancelen una cita yo pueda eliminarla.	
Criterios de aceptación	
<b>Se deberá utilizar lenguaje Gherkin:</b> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;Eliminar citas&gt;</p> <p><b>Dado que</b> el usuario quiere eliminar una cita  <b>Cuando</b> el usuario quiere cancelar una cita ya registrada previamente  <b>Entonces</b> el usuario deberá buscar la cita indicada que ya fue registrada anteriormente y seleccionarla para así poder confirmar la eliminación.</p>
2.	<p>&lt;Tener citas previas&gt;</p> <p><b>Dado que</b> el usuario quiere eliminar una cita  <b>Cuando</b> el usuario quiere eliminar una cita y no hay citas registradas  <b>Entonces</b> el usuario deberá buscar la cita indicada, pero como no está la cita buscada o no hay citas registradas mandará el mensaje de error correspondiente.</p>

Imagen 3.6 Historia de usuarios: Eliminar citas.

Generales				
Clave	<HU8>			
Título	<Agregar servicios>			
Puntos	<Hector Roman Robles Orantes>			
Descripción				
<p>Como: administrador.</p> <p>Quiero: que se pueda agregar el servicio que necesito y el precio de dicho servicio.</p> <p>Para: tener un control de los servicios del negocio BashLashes.</p> <p>Instrucciones:</p> <p>Las historias de usuario deberán hacerse como narrativa:</p> <p><b>Por ejemplo:</b> Como <b>comensal</b> quiero <b>conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.</b></p>				
Criterios de aceptación				
<p><b>Se deberá utilizar lenguaje Gherkin:</b></p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>				
1.	<Datos a ingresar>	<p><b>Dado que</b> el usuario quiere agregar un nuevo servicio.</p> <p><b>Cuando</b> el usuario agregue el servicio.</p> <p><b>Entonces</b> le desplegará un formulario en el cual le pedirá ingresar el nombre del servicio y el precio de dicho servicio.</p>		
	<Servicio nuevo >	<p><b>Dado que</b> el usuario quiere agregar un nuevo servicio</p> <p><b>Cuando</b> el usuario quiera confirmar el servicio.</p> <p><b>Entonces</b> le aparecerá una alerta si está seguro de agregar dicho servicio.</p>		
2.	<Servicio repetido>	<p><b>Dado que</b> el usuario quiere agregar un servicio repetido</p> <p><b>Cuando</b> el usuario quiera confirmar un servicio que ya se había agregado.</p> <p><b>Entonces</b> le mandara una alerta que el servicio que quiere agregar se encuentra repetido.</p>		
	<datos vacíos>	<p><b>Dado que</b> el usuario quiere agregar un servicio y costo sin datos</p> <p><b>Cuando</b> el usuario intente agregar un servicio y un costo en el cual no ingresa ningún dato, dejando todos los espacios vacíos.</p> <p><b>Entonces</b> se mostrará un mensaje de error correspondiente indicándole que ingrese los datos faltantes.</p>		
3	<Costo>	<p><b>Dado que</b> el usuario quiera agregar un servicio debe de colocar el costo.</p> <p><b>Cuando</b> el usuario intente agregar un costo negativo o 0.</p> <p><b>Entonces</b> se le desplegará una alerta que dicho costo a agregar no se puede colocar en número negativo y tampoco el número 0.</p>		
	<Descripción>	<p><b>Dado que</b> el usuario quiera agregar un servicio puede decidir si quiere colocar descripción o no.</p> <p><b>Cuando</b> el usuario intente agregar una descripción.</p> <p><b>Entonces</b> podrá darle clic a un botón y se desplegará un formulario en el cual le pida agregar descripción.</p>		

Imagen 3.7 Historia de usuarios: Agregar servicios.

Generales							
Clave	<HU9>						
Título	<Editar servicios>						
Puntos	<Hector Roman Robles Orantes>						
Descripción							
<p>Como: Administrador.</p> <p>Quiero: que se pueda editar servicios ya establecidos para modificar su precio y nombre.</p> <p>Para: poder editar el servicio y precio del negocio de BashLashes.</p>							
Instrucciones:							
<p>Las historias de usuario deberán hacerse como narrativa:</p> <p><b>Por ejemplo:</b> Como comensal quiero conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.</p>							
Criterios de aceptación							
<p><b>Se deberá utilizar lenguaje Gherkin:</b></p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>							
1.	<table border="1"> <tr> <td>&lt;Datos a ingresar&gt;</td><td> <p><b>Dado que</b> el usuario quiera editar un servicio.</p> <p><b>Cuando</b> el usuario acepte editar el servicio.</p> <p><b>Entonces</b> le desplegará un formulario en el cual le pedirá ingresar el nuevo nombre del servicio y el precio de dicho servicio.</p> </td></tr> <tr> <td>&lt;Servicio editado&gt;</td><td> <p><b>Dado que</b> el usuario quiera editar un nuevo servicio.</p> <p><b>Cuando</b> el usuario quiera confirmar el servicio a editar.</p> <p><b>Entonces</b> le aparecerá una alerta si está seguro de editar</p> </td></tr> </table>	<Datos a ingresar>	<p><b>Dado que</b> el usuario quiera editar un servicio.</p> <p><b>Cuando</b> el usuario acepte editar el servicio.</p> <p><b>Entonces</b> le desplegará un formulario en el cual le pedirá ingresar el nuevo nombre del servicio y el precio de dicho servicio.</p>	<Servicio editado>	<p><b>Dado que</b> el usuario quiera editar un nuevo servicio.</p> <p><b>Cuando</b> el usuario quiera confirmar el servicio a editar.</p> <p><b>Entonces</b> le aparecerá una alerta si está seguro de editar</p>		
<Datos a ingresar>	<p><b>Dado que</b> el usuario quiera editar un servicio.</p> <p><b>Cuando</b> el usuario acepte editar el servicio.</p> <p><b>Entonces</b> le desplegará un formulario en el cual le pedirá ingresar el nuevo nombre del servicio y el precio de dicho servicio.</p>						
<Servicio editado>	<p><b>Dado que</b> el usuario quiera editar un nuevo servicio.</p> <p><b>Cuando</b> el usuario quiera confirmar el servicio a editar.</p> <p><b>Entonces</b> le aparecerá una alerta si está seguro de editar</p>						
2	<table border="1"> <tr> <td>&lt;datos vacíos&gt;</td><td> <p><b>Dado que</b> el usuario quiere editar un servicio y costo sin datos</p> <p><b>Cuando</b> el usuario intente agregar un servicio y un costo en el cual no ingresa ningún dato, dejando todos los espacios vacíos.</p> <p><b>Entonces</b> se mostrará un mensaje de error correspondiente indicándole que ingrese los datos faltantes</p> </td></tr> <tr> <td>&lt;Costo&gt;</td><td> <p><b>Dado que</b> el usuario quiera editar un servicio debe de colocar el costo.</p> <p><b>Cuando</b> el usuario intente agregar un costo negativo o 0.</p> <p><b>Entonces</b> se le desplegará una alerta que dicho costo a agregar no se puede colocar en número negativo y tampoco el numero 0.</p> </td></tr> <tr> <td>&lt;Descripción&gt;</td><td> <p><b>Dado que</b> el usuario quiera editar una descripción seleccionada.</p> <p><b>Cuando</b> el usuario intente editar una descripción.</p> <p><b>Entonces</b> podrá darle clic a un botón y se desplegará un formulario para editar la descripción.</p> </td></tr> </table>	<datos vacíos>	<p><b>Dado que</b> el usuario quiere editar un servicio y costo sin datos</p> <p><b>Cuando</b> el usuario intente agregar un servicio y un costo en el cual no ingresa ningún dato, dejando todos los espacios vacíos.</p> <p><b>Entonces</b> se mostrará un mensaje de error correspondiente indicándole que ingrese los datos faltantes</p>	<Costo>	<p><b>Dado que</b> el usuario quiera editar un servicio debe de colocar el costo.</p> <p><b>Cuando</b> el usuario intente agregar un costo negativo o 0.</p> <p><b>Entonces</b> se le desplegará una alerta que dicho costo a agregar no se puede colocar en número negativo y tampoco el numero 0.</p>	<Descripción>	<p><b>Dado que</b> el usuario quiera editar una descripción seleccionada.</p> <p><b>Cuando</b> el usuario intente editar una descripción.</p> <p><b>Entonces</b> podrá darle clic a un botón y se desplegará un formulario para editar la descripción.</p>
<datos vacíos>	<p><b>Dado que</b> el usuario quiere editar un servicio y costo sin datos</p> <p><b>Cuando</b> el usuario intente agregar un servicio y un costo en el cual no ingresa ningún dato, dejando todos los espacios vacíos.</p> <p><b>Entonces</b> se mostrará un mensaje de error correspondiente indicándole que ingrese los datos faltantes</p>						
<Costo>	<p><b>Dado que</b> el usuario quiera editar un servicio debe de colocar el costo.</p> <p><b>Cuando</b> el usuario intente agregar un costo negativo o 0.</p> <p><b>Entonces</b> se le desplegará una alerta que dicho costo a agregar no se puede colocar en número negativo y tampoco el numero 0.</p>						
<Descripción>	<p><b>Dado que</b> el usuario quiera editar una descripción seleccionada.</p> <p><b>Cuando</b> el usuario intente editar una descripción.</p> <p><b>Entonces</b> podrá darle clic a un botón y se desplegará un formulario para editar la descripción.</p>						

Imagen 3.8 Historia de usuarios: Editar servicios.

Generales	
Clave	<HU10>
Título	<Eliminar servicios>
Puntos	<Hector Roman Robles Orantes>

Descripción
<p>Como: administrador.</p> <p>Quiero: poder eliminar servicios.</p> <p>Para: mantener actualizado el negocio de BashLashes.</p>
<p>Instrucciones:</p> <p>Las historias de usuario deberán hacerse como narrativa:</p> <p><b>Por ejemplo:</b> Como comensal quiero conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.</p>

Criterios de aceptación
<p>Se deberá utilizar lenguaje Gherkin:</p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>

1.	<Datos a ingresar>	<p><b>Dado que</b> el usuario quiera editar un servicio.  <b>Cuando</b> el usuario acepte editar el servicio.  <b>Entonces</b> le desplegará un formulario en el cual le pedirá ingresar el nuevo nombre del servicio y el precio de dicho servicio.</p>
	<Servicio editado>	<p><b>Dado que</b> el usuario quiera editar un nuevo servicio.  <b>Cuando</b> el usuario quiera confirmar el servicio a editar.  <b>Entonces</b> le aparecerá una alerta si está seguro de editar</p>
2	<datos vacíos>	<p>dicho servicio.  <b>Dado que</b> el usuario quiere editar un servicio y costo sin datos  <b>Cuando</b> el usuario intente agregar un servicio y un costo en el cual no ingresa ningún dato, dejando todos los espacios vacíos.  <b>Entonces</b> se mostrará un mensaje de error correspondiente indicándole que ingrese los datos faltantes</p>
	<Costo>	<p><b>Dado que</b> el usuario quiera editar un servicio debe de colocar el costo.  <b>Cuando</b> el usuario intente agregar un costo negativo o 0.  <b>Entonces</b> se le desplegará una alerta que dicho costo a agregar no se puede colocar en número negativo y tampoco el numero 0.</p>
	<Descripción>	<p><b>Dado que</b> el usuario quiera editar una descripción seleccionada.  <b>Cuando</b> el usuario intente editar una descripción.  <b>Entonces</b> podrá darle clic a un botón y se desplegará un formulario para editar la descripción.</p>

Imagen 3.9 Historia de usuarios: Eliminar servicios.

Generales	
Clave	<HU11>
Título	<Visualizar servicios>
Puntos	<Hector Roman Robles Orantes>

Descripción	
Como: administrador.	Permitirá visualizar todos los servicios BashLashes.
Quiero: poder visualizar todos los servicios que ya se agregaron.	
Para: visualizar los servicios que se dan en el negocio de BashLashes.	
Instrucciones:	
Las historias de usuario deberán hacerse como narrativa:	
	<b>Por ejemplo:</b> Como comensal quiero conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.

Criterios de aceptación	
<b>Se deberá utilizar lenguaje Gherkin:</b>	
1.	<p><b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</p> <ul style="list-style-type: none"> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>
	<p><b>Dado que</b> el usuario quiere visualizar los servicios.  <b>Cuando</b> el usuario quiera ver todos los servicios.  <b>Entonces</b> se le desplegará una vista de todos los servicios</p> <p>que ya se han ingresado.</p>

Imagen 3.10 Historias de usuario:Visualizar servicios.

Generales	
Clave	—
Título	<Confirmar servicio y pago>
Puntos	<Hector Roman Robles Orantes>
Descripción	
<p>Como: administrador.          Quiero: poder confirmar servicios y pagos de las citas ya realizadas.          Para: tener una administración correcta de los ingresos de BashLashes.</p>	
Instrucciones:	
<p>Las historias de usuario deberán hacerse como narrativa:  <b>Por ejemplo:</b> Como comensal quiero conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.</p>	
Criterios de aceptación	
<p>Se deberá utilizar lenguaje Gherkin:</p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;Datos a ingresar&gt;</p> <p>Dado que el usuario quiere confirmar los servicios y pagos de citas ya realizadas.  <b>Cuando</b> el usuario  <b>Entonces</b> se le desplegará una vista de todos los servicios que ya se han ingresado.</p>

Imagen 3.11 Historias de usuario:Confirmar servicio y pago.

Generales	
Clave	—
Título	<Visualizar reporte>
Puntos	<Hector Roman Robles Orantes>
Descripción	
<p>Como: administrador.          Quiero: visualizar los ingresos de las citas.          Para: tener un mejor control de los ingresos del negocio.</p>	
Instrucciones:	
<p>Las historias de usuario deberán hacerse como narrativa:  <b>Por ejemplo:</b> Como comensal quiero conocer todas las opciones del menú para elegir el platillo que deseo comer en el momento, así como tener idea de las calorías que contiene.</p>	
Criterios de aceptación	
<p>Se deberá utilizar lenguaje Gherkin:</p> <ul style="list-style-type: none"> <li>• <b>Given:</b> Contexto inicial/precondiciones que se cumplen en el inicio del test</li> <li>• <b>When:</b> Describe un evento.</li> <li>• <b>Then:</b> Salida esperada tras la ejecución del evento.</li> </ul>	
1.	<p>&lt;Visualizar reporte&gt;</p> <p>Dado que el usuario quiera checar el reporte de ingresos.  <b>Cuando</b> el usuario este seleccionando la fecha del reporte.  <b>Entonces</b> le arrojara todos los ingresos de la fecha que este seleccionando.</p>

Imagen 3.12 Historias de usuario:Visualizar reporte

#### **Elección de paletas de colores:**

Para el diseño de la paleta de colores de Karendaki, nos hemos inspirado en la estética moderna. Optamos por una combinación de colores suaves, como tonos morados ,una sombra clara de azul-magenta, y tonos de azul, que proporcionan una apariencia agradable y confiable.

#### **Elección de vistas:**

En cuanto a las vistas del software Karendaki, nos hemos centrado en mantener una interfaz limpia y fácil de navegar. Hemos priorizado la simplicidad y la claridad en la disposición de las vistas, asegurando que cada elemento esté organizado de manera lógica y accesible para el usuario.

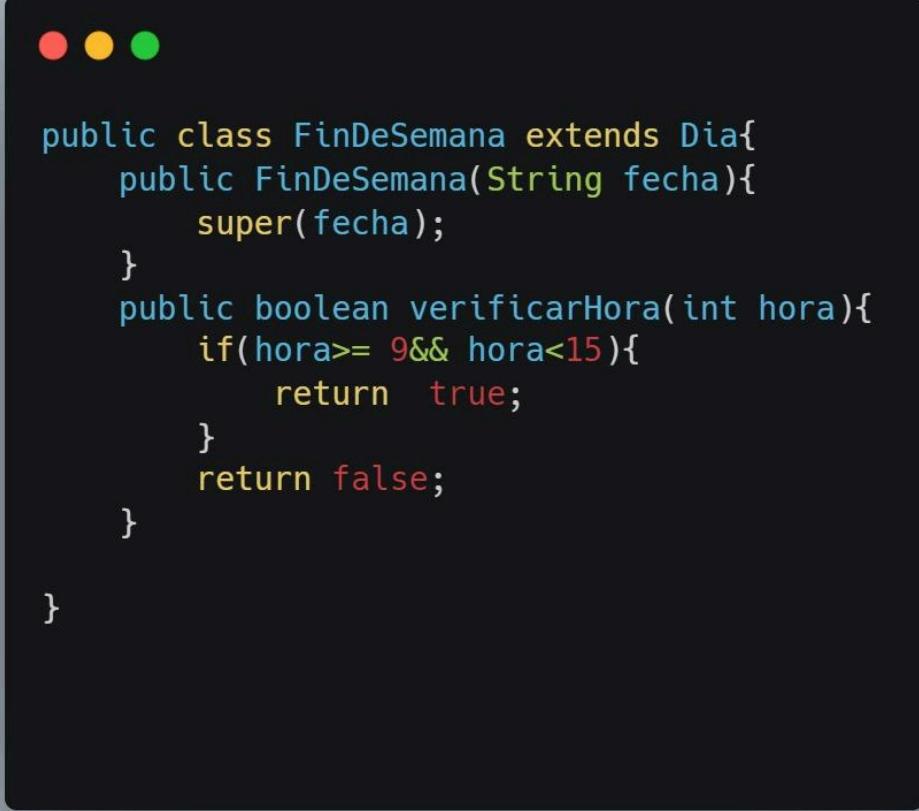
#### **Elección de elementos para combinar las funcionalidades:**

Al combinar las funcionalidades de Karendaki, nos hemos enfocado en la coherencia y la integración fluida entre los diferentes elementos. Hemos seleccionado cuidadosamente iconos, botones y otros elementos de diseño que complementan las funcionalidades del software y facilitan su uso intuitivo.

#### **Elección de formas:**

En cuanto a las formas utilizadas en el diseño de Karendaki, nos hemos inclinado hacia líneas limpias y geométricas que reflejan la estética minimalista. Hemos evitado el uso excesivo de formas complejas o adornos innecesarios, optando en su lugar por formas simples y definidas que contribuyen a la claridad.

-



The screenshot shows a Java code editor window with a dark theme. At the top left are three colored window control buttons: red, yellow, and green. The code itself is written in Java and defines a class named `FinDeSemana` which extends the `Dia` class. It includes a constructor that takes a `String fecha` and calls `super(fecha)`. A method `boolean verificarHora(int hora)` is implemented to check if a given hour is between 9 and 15, returning `true` if it is and `false` otherwise.

```
public class FinDeSemana extends Dia{
    public FinDeSemana(String fecha){
        super(fecha);
    }
    public boolean verificarHora(int hora){
        if(hora>= 9&& hora<15){
            return true;
        }
        return false;
    }
}
```

Imagen 4.0 Código:FinDeSemana

En esta parte del código de la clase `FinDeSemana` hijo de `Dia`, se implementa el método abstracto.



```
import java.util.ArrayList;
import java.util.Scanner;

public class Historial {
    private Ventas objectventas = new Ventas();

    public void setobjectVentas(Ventas objectventas) {
        this.objectventas = objectventas;
    }

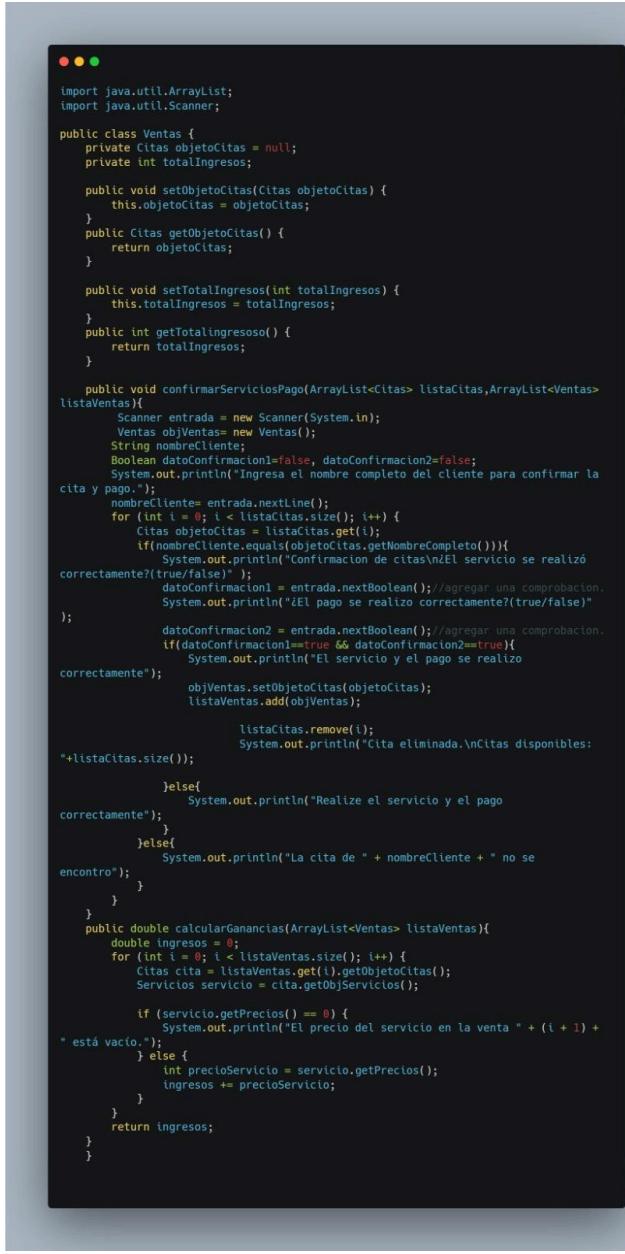
    public Ventas get() {
        return objectventas;
    }

    public void mostrarReporte(ArrayList<Ventas> listaVentas) {
        for (int i = 0; i < listaVentas.size(); i++) {
            Citas cita = listaVentas.get(i).getObjetosCitas();
            Servicios servicio = cita.getObjetosServicios();
            System.out.println("Cita " + (i + 1) + " - Cliente: " +
                cita.getNombreCompleto() + ", Servicio: " +
                servicio.getNombreServicios() + ", Precio: " +
                servicio.getPrecios());
        }
        double ingresosTotales = objectventas.calcularGanancias(listaVentas);
        System.out.println("La ganancia total fue: " + ingresosTotales);
    }

    public void editarReporte(ArrayList<Ventas> listaVentas){
        Scanner entrada = new Scanner(System.in);
        String nuevoNombre,nuevoServicio,nombre;
        int nuevoPrecio,r1;
        System.out.println("Ingrese el nombre del cliente");
        nombre= entrada.nextLine();
        for(int i=0; i<listaVentas.size();i++){
            Citas citas = listaVentas.get(i).getObjetosCitas();
            Servicios servicio = citas.getObjetosServicios();
            if(nombre.equalsIgnoreCase(citas.getNombreCompleto())){
                System.out.println("¿Qué quieres editar?(1 Nombre/2 Precios/3
Servicio.)");
                do {
                    r1 = entrada.nextInt();
                    if (r1 < 1 || r1 > 3) {
                        System.out.println("Error, ingrese una opcion entre el 1, 2 o
3");
                    }
                } while (r1 < 1 || r1 > 3);
                switch (r1) {
                    case 1:
                        System.out.println("Ingrese la correccion del nuevo
nombre:");
                        entrada.nextLine();
                        nuevoNombre = entrada.nextLine();
                        citas.setNombreCompleto(nuevoNombre);
                        break;
                    case 2:
                        System.out.println("Ingrese la correccion del nuevo
precio:");
                        nuevoPrecio = entrada.nextInt();
                        servicio.setPrecios(nuevoPrecio);
                        break;
                    case 3:
                        System.out.println("Ingrese la nueva correccion del
servicio:");
                        entrada.nextLine();
                        nuevoServicio= entrada.nextLine();
                        servicio.setNombreServicios(nuevoServicio);
                        break;
                }
            }
        }
    }
}
```

Imagen 4.1 código:Clase Historial

En la clase Historial se obtiene una instancia de ventas y muestra reporte y edita reporte.



```
import java.util.ArrayList;
import java.util.Scanner;

public class Ventas {
    private Citas objetoCitas = null;
    private int totalIngresos;

    public void setObjetoCitas(Citas objetoCitas) {
        this.objetoCitas = objetoCitas;
    }

    public Citas getObjetoCitas() {
        return objetoCitas;
    }

    public void setTotalIngresos(int totalIngresos) {
        this.totalIngresos = totalIngresos;
    }

    public int getTotalIngresos() {
        return totalIngresos;
    }

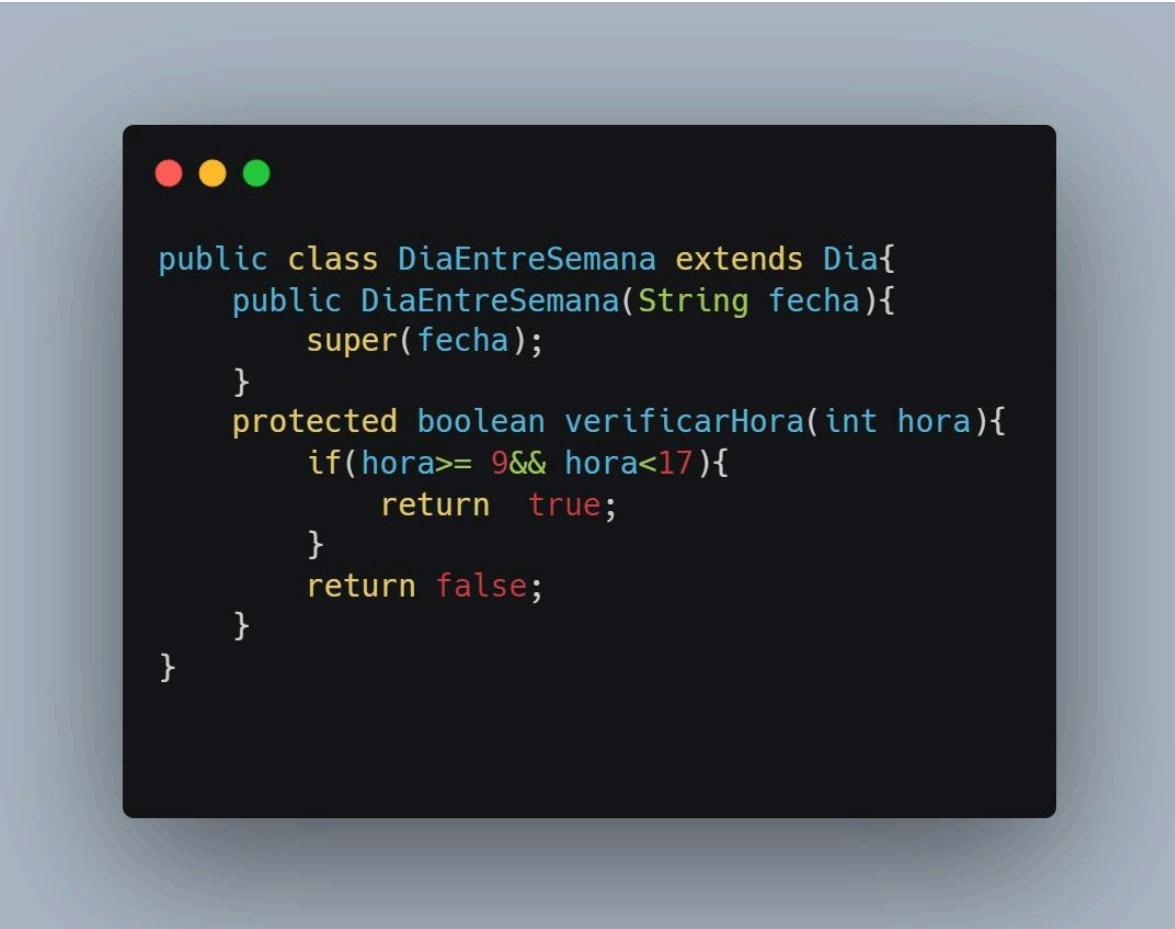
    public void confirmarServiciosPago(ArrayList<Citas> listaCitas, ArrayList<Ventas> listaVentas) {
        Scanner entrada = new Scanner(System.in);
        Ventas objVentas = new Ventas();
        String nombreCliente;
        Boolean datoConfirmacion1=false, datoConfirmacion2=false;
        System.out.println("Ingresa el nombre completo del cliente para confirmar la cita y pago.");
        nombreCliente= entrada.nextLine();
        for (int i = 0; i < listaCitas.size(); i++) {
            Citas objetoCitas = listaCitas.get(i);
            if(nombreCliente.equals(objetoCitas.getNombreCompleto())){
                System.out.println("Confirmación de citas\nEl servicio se realizó correctamente?(true/false)");
                datoConfirmacion1 = entrada.nextBoolean(); //agregar una comprobación.
                System.out.println("El pago se realizó correctamente?(true/false)");
            };
            datoConfirmacion2 = entrada.nextBoolean(); //agregar una comprobación.
            if(datoConfirmacion1==true && datoConfirmacion2==true){
                System.out.println("El servicio y el pago se realizaron correctamente");
                objetoCitas.setObjetoCitas(objetoCitas);
                listaVentas.add(objVentas);

                listaCitas.remove(i);
                System.out.println("Cita eliminada.\nCitas disponibles:" + listaCitas.size());
            }else{
                System.out.println("Realice el servicio y el pago correctamente");
            }
            else{
                System.out.println("La cita de " + nombreCliente + " no se encontró");
            }
        }
        public double calcularGanancias(ArrayList<Ventas> listaVentas){
            double ingresos = 0;
            for (int i = 0; i < listaVentas.size(); i++) {
                Citas cita = listaVentas.get(i).getObjetoCitas();
                Servicio servicio = cita.getObjServicios();

                if (servicio.getPrecios() == 0) {
                    System.out.println("El precio del servicio en la venta " + (i + 1) +
                        " está vacío.");
                } else {
                    int precioServicio = servicio.getPrecios();
                    ingresos += precioServicio;
                }
            }
            return ingresos;
        }
    }
}
```

Imagen 4.2 código:Clase Ventas

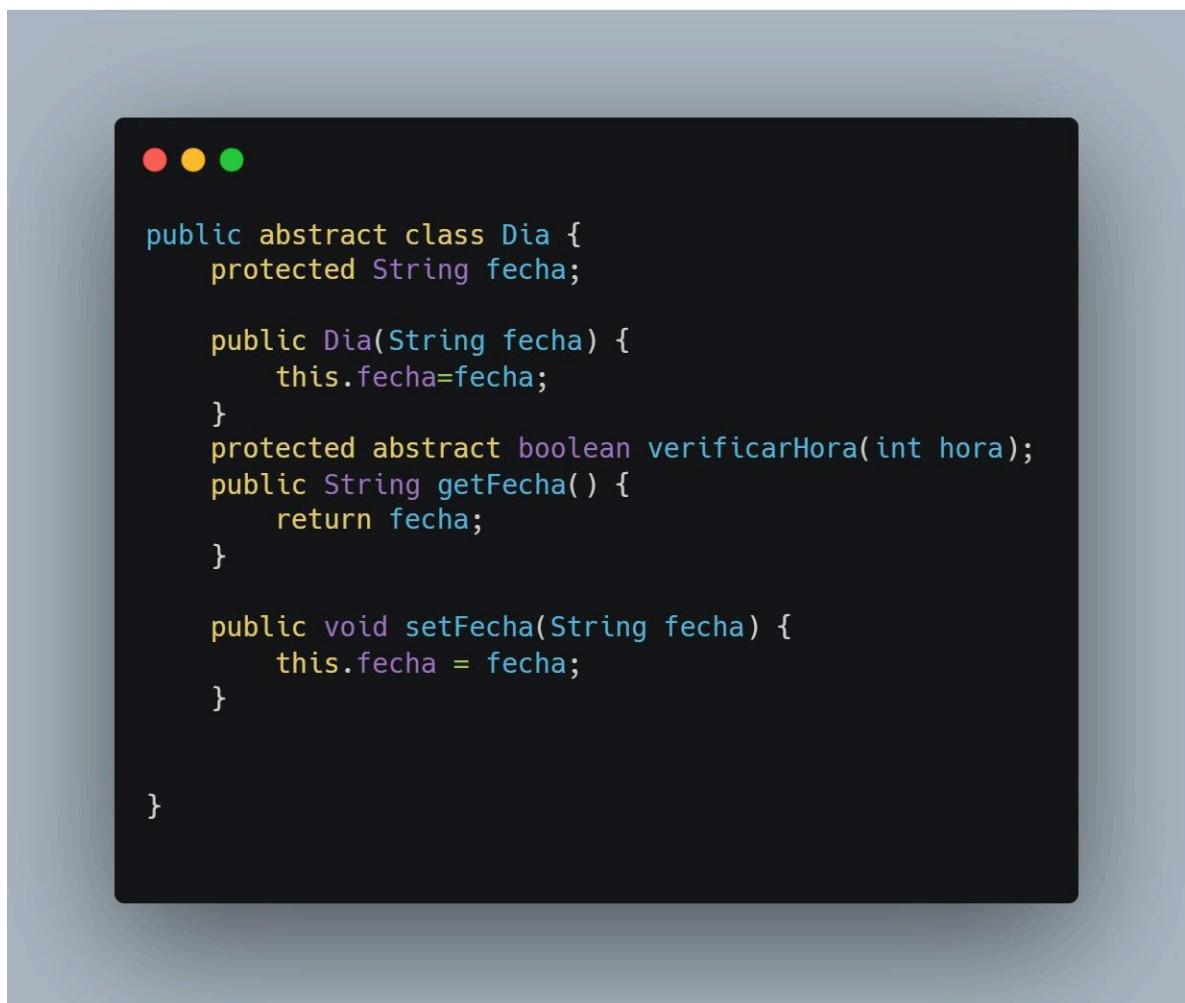
En esta clase hace el cálculo de ganancias y la confirmación de servicios que realmente se hayan pagado.



```
public class DiaEntreSemana extends Dia{
    public DiaEntreSemana(String fecha){
        super(fecha);
    }
    protected boolean verificarHora(int hora){
        if(hora>= 9&& hora<17){
            return true;
        }
        return false;
    }
}
```

Imagen 4.3 código: Clase DiaEntreSemana

En esta clase DiaEntreSemana es hijo de Dia y implementa a su manera el metodo abstracto.



The screenshot shows a dark-themed Java code editor window. At the top, there are three colored window control buttons (red, yellow, green). Below them is the Java code for the abstract class `Dia`. The code includes a protected attribute `String fecha`, a constructor that initializes `fecha`, and methods for getting and setting the date. It also includes an abstract method `verificarHora(int hora)`.

```
public abstract class Dia {  
    protected String fecha;  
  
    public Dia(String fecha) {  
        this.fecha=fecha;  
    }  
    protected abstract boolean verificarHora(int hora);  
    public String getFecha() {  
        return fecha;  
    }  
  
    public void setFecha(String fecha) {  
        this.fecha = fecha;  
    }  
}
```

Imagen 4.4 código:Clase Dia

La clase `Dia` es el padre de `Diaentresemana` y `Findesemana`, a su vez es la clase abstracta e implementa la declaración del método abstracto para que los hijos lo adeguen a su manera.

## 6 Resultados

---

```
File Edit Selection View Go Run PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS comprob
PS C:\Users\minis\OneDrive\Escritorio\comprob> & "C:\Program Files\Java\jdk-21\bin\java.exe" '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\minis\AppData\Roaming\Code\User\reduhat.java\jdt_ws\comprob_64bcd86\bin' 'Main'
Menú principal.
Seleccionar (1 Ver citas/2 Menú de ventas/3 Menú de citas/4 Menú de servicios/5 Salir).
4
Menú servicios.
Seleccionar (1 Agregar/2 Editar/3 Mostrar/4 Eliminar/5 Salir).
1
Ingrese el nombre del servicio.
pestanas largas.
Ingrese la descripción del servicio.
se usa pegamento
Ingrese el precio del servicio:
400
¿Quieres agregar otro servicio (1 Si/2 No)?
2
Menú servicios.
Seleccionar (1 Agregar/2 Editar/3 Mostrar/4 Eliminar/5 Salir).
3
Servicio 1
Nombre del servicio: pestanas
Precio del servicio: 500
Descripción del servicio: Pegamento
-----
Servicio 2
Nombre del servicio: unas
Precio del servicio: 200
Descripción del servicio: Pegamento
-----
Servicio 3
Nombre del servicio: pestanas largas
Precio del servicio: 400
Descripción del servicio: se usa pegamento
-----
Menú servicios.
Seleccionar (1 Agregar/2 Editar/3 Mostrar/4 Eliminar/5 Salir).
2
Ingrese el nombre del servicio a editar.
pestanas
¿Qué quieres editar?(1 Nombre/2 Precios/3 Descripción).
2
Ingrese el nuevo precio:
400
Menú servicios.
Seleccionar (1 Agregar/2 Editar/3 Mostrar/4 Eliminar/5 Salir).
4
Ingrese el nombre del servicio a eliminar.
pestanas largas.
Servicio eliminado.
Servicios disponibles: 2
Menú servicios.
Seleccionar (1 Agregar/2 Editar/3 Mostrar/4 Eliminar/5 Salir).
5
```

El proyecto se desplegó en consola y los resultados fueron adecuados a la implementación de las ideas originalmente establecidas, los CRUDS fueron exitosamente realizados.

## **7 Lecciones Aprendidas**

---

La falta de experiencia en la elaboración de proyectos grandes ha generado estrés durante el desarrollo de este proyecto. Durante la etapa de delimitación, descartar ideas fue desafiante, llevándonos a momentos de bloqueo al intentar definir qué aspectos incluir sin exceder los límites establecidos. Esta situación nos ha llevado a sentirnos perdidos y frustrados al no poder visualizar claramente qué elementos heredar en el proyecto sin salirnos de las restricciones establecidas. Sin embargo, esto lo resolvimos buscando muchos ejemplos de softwares similares y videos para que la idea que se implementó no estuviera fuera de los límites.

Al momento de tener que implementar las cosas que nos pedían que debería tener el proyecto pasamos por una falta de comprensión en los temas que se nos proporcionaron, ya que la idea fue errónea a la hora de su implementación, generó más problemas a la hora de usarlo por lo que tuvimos que investigar más logrando así la solución del problema a costa de robarnos tiempo.

## **8 Competencias adquiridas**

---

- >>Comprender mejor el lenguaje de Java.
- >>Adaptar los métodos necesarios para el proyecto.
- >>Organización de métodos.
- >>Crear algoritmos capaces de tener excepciones.
- >>Usar los recursos necesarios para solucionar los problemas del proyecto.
- >>Manipular métodos abstractos.
- >>Aceptar que no se pueden hacer objetos de clases abstractas.
- >>Manejo de excepciones.
- >>Diferenciar las clases normales con las abstractas.
- >>Investigar más sobre soluciones a problemas que tengamos.
- >>Organizar de mejor manera el código gracias a los diagramas.
- >>Planear la estructura del código gracias a los diagramas de clase.
- >>Localizar partes específicas del código.

- >>Solucionar problemas más rápidamente gracias a múltiples herramientas.
- >>Desarrollar un código más limpio y eficiente.
- >>Integrarse más en los trabajos en equipo.
- >>Colaborar de mejor manera con otras personas.
- >>Explicación de bloques de código.
- >>Adaptarse al cambio de ideas.
- >>Evaluar cual es la ruta más corta para sacar el proyecto.

## 9 Conclusiones

---

El programa resultó más difícil de lo esperado, debido a que pensamos que sería más sencillo pero fue escalando por todos los elementos que requería, entonces durante todo el cuatrimestre que tuvimos le agregamos mas y mas cosas que nos complicaron mucho mas el programa

por lo que decidimos orientarnos bien en cuestión a nuestros tiempos y trabajar en el código y en todos los documentos requeridos, debido a que todos ellos requieren un minucioso trabajo debido a que de ellos nos guiarímos o alguien más ser guiará para entender el programa, que elementos tiene y para qué sirve, por lo tanto durante el proceso de elaboración nuestro entendimiento sobre el lenguaje java mejora más de lo pensado debido a que ya podíamos hacer cosas que antes no se nos ocurrían, pero debido a tener más cosas también generó más problemas, porque había más confusión con cada elemento agregado, no solo eso tambien subio el grado de dificultad debido a que si quieres agregarles más funciones, algunas de ellas son más avanzadas por lo que quiere que tengas mayor conocimiento sobre el propio lenguaje y las librerías externas, al final concluimos en que nos sirvió demasiado este proyecto y nos gusto el hecho de como quedo, aunque aún faltan cosas para poder llamarlo una app de citas consideramos que es un gran avance y si en algún futuro con más conocimiento sobre programación como las bases de datos, esperamos poder actualizarlo y así crear un mejor código.