8 Modelo de diseño.

8.1 Diagrama de clases de diseño.

Representaremos de forma estática las clases de trabajo de nuestro sistema. Ya que este se compone de cuatro aplicaciones claramente diferenciadas crearemos un diagrama de sistema propio para cada una de ellas. Destacamos que, aunque algunas clases puedan estar solo en uno de los diagramas pues serán propias y exclusivas de él otras serán comunes a todos o a varios de ellos.

Las clases destacadas que serán comunes a todos los diagramas serán la de **Sistema** y la de **Fachada** dichas clases implementan el patrón fachada que separa a la clase sistema del control de la base de datos reduciendo la visibilidad sobre ella.

La clase sistema admite distintos tipos de control, ya sea por parte de los usuarios mediante Clases Interfaz o mediante callbacks a los que el propio sistema puede subscribirse utilizando las Clase Timer.

Las Clases Interfaz aunque dependientes de cada una de las aplicaciones tienen en común que implementan un patrón vista controlador sobre la interfaz separando las clases de trabajo de las vistas de la interfaz.

En estas clases para reducir la dependencia no se guardarán instancias de las clases de trabajo cuando se estén mostrando sus datos, sino que solo se almacenarán como integer o string según corresponda en los identificadores únicos en el modero ER de cada una de las entidades a las que cada clase represente. Mediante estos identificadores, cuando un usuario seleccione, por ejemplo, una pieza podremos saber que pieza era, ya que sabremos su identificador único, el cual podremos proporcionar a la clase sistema.

Debido a esto los datos de las clases de trabajo se almacenarán en estructuras de tipo hash, cuya clave de búsqueda será el identificador único en el modero ER de cada una de las entidades a las que cada clase represente.

8.1.1 Diagrama de escritorio.

Esta aplicación será utilizada por los **responsables del Almacén**, los **Coordinadores Técnicos**, los **ayudantes de Almacén** y los **ayudantes de coordinador**. Recordamos como dijimos en el apartado visión que la aplicación sería diseñada en java en la versión de bajo coste y en JavaScript en las dos versiones de más alto coste.

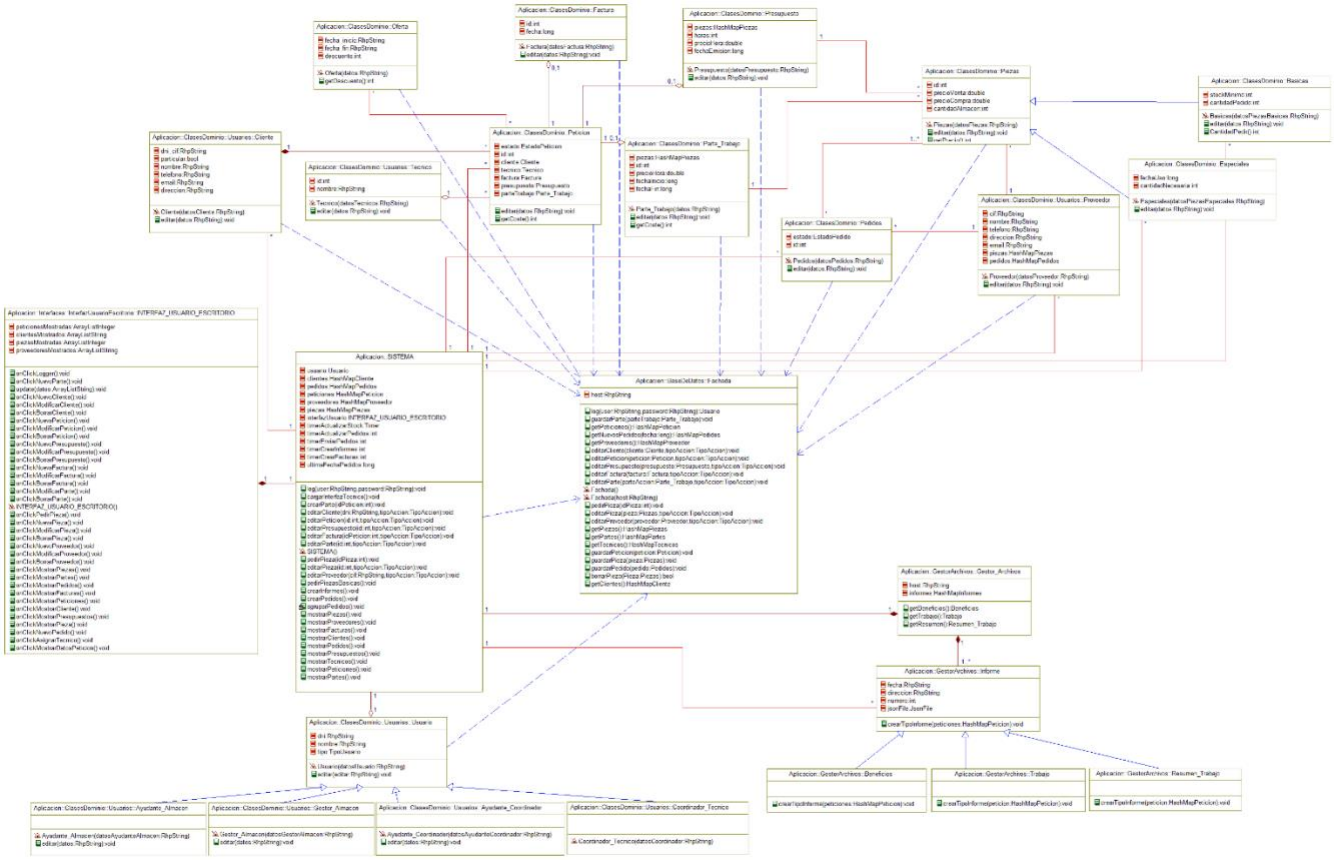
En este diagrama utilizaremos una clase interfaz que hará de manejador separando las clases de trabajo de las vistas.

Se utilizará también como se ha mencionado anteriormente una clase fachada para comunicarnos con la base de datos separando la complejidad de dichas comunicaciones de las clases de trabajo.

Existe una clase Gestor de Archivos que nos permite publicar y ver archivos de tipo json sobre el servidor PHP de nuestro sistema. Para poder utilizar estos archivos, necesitamos la clase informe con sus respectivas hijas.

Como clases de trabajo principales, tendremos peticiones podrán estar asignadas a un único técnico y podrán tener o no un presupuesto, una factura o un parte de trabajo. Tanto el parte de trabajo como el presupuesto podrán tener piezas.

Existirán clases proveedor que nos indicarán quien nos trae las piezas.
Para implementar el log in en el sistema, utilizamos una clase usuario cuyas hijas son los tipos de usuario que podrán utilizar esta aplicación (los responsables del Almacén, los Coordinadores Técnicos, los ayudantes de Almacén y los ayudantes de coordinador).
Cuando un usuario haya logrado conectarse adecuadamente al sistema proporcionando su usuario y contraseña de cuya verificación se encarga la base de datos, la clase sistema obtendrá una instancia de una clase usuario para saber en todo momento quien está utilizando la aplicación.
Cuando queramos modificar/crear/borrar algún dato del sistema, utilizaremos las clases de trabajo y cuando las modificaciones realizadas sean muchas o cuando haya pasado mucho tiempo desde la última actualización de la base de datos, éstas se materializarán en ella.
Cabe destacar que para comunicarnos entre el sistema y la interfaz se utilizará un tipo definido como enumeración el cuál es obligatorio poner cómo parámetro en los métodos de tipo **editar()**. Dicha enumeración contiene los valores crear, modificar y eliminar.
Para comunicarnos con la interfaz de usuario utilizaremos el método update() al que le pasaremos un vector de strings formateado en json de modo que a la entrada de los datos existirá un parser que decodifique la información para poder actualizar la interfaz eliminando la visión de esta respecto de las clases de trabajo. Esto mismo se utiliza también en los métodos editar() de las clases de trabajo y de sus constructores.
Trabajar con archivos json para realizar estas acciones aporta grandes beneficios, especialmente a la hora de debuggear el código, pués el paso de información entre clases es fácilmente formateable a texto entendible por el programador.



8.1.2 Diagrama de la aplicación móvil.

Esta aplicación será utilizada por los técnicos informaticos cuando estén trabajando. Recordamos como dijimos en el apartado de visión, que esta aplicación será realizada en javasrcipt mediante el framework ionic2 que nos permitirá crear aplicaciones multiplataforma.

Por el contrario para la versión de más alto coste se utilizará los lenguajes de programación soportados de forma nativa en cada plataforma móvil (java, swift).

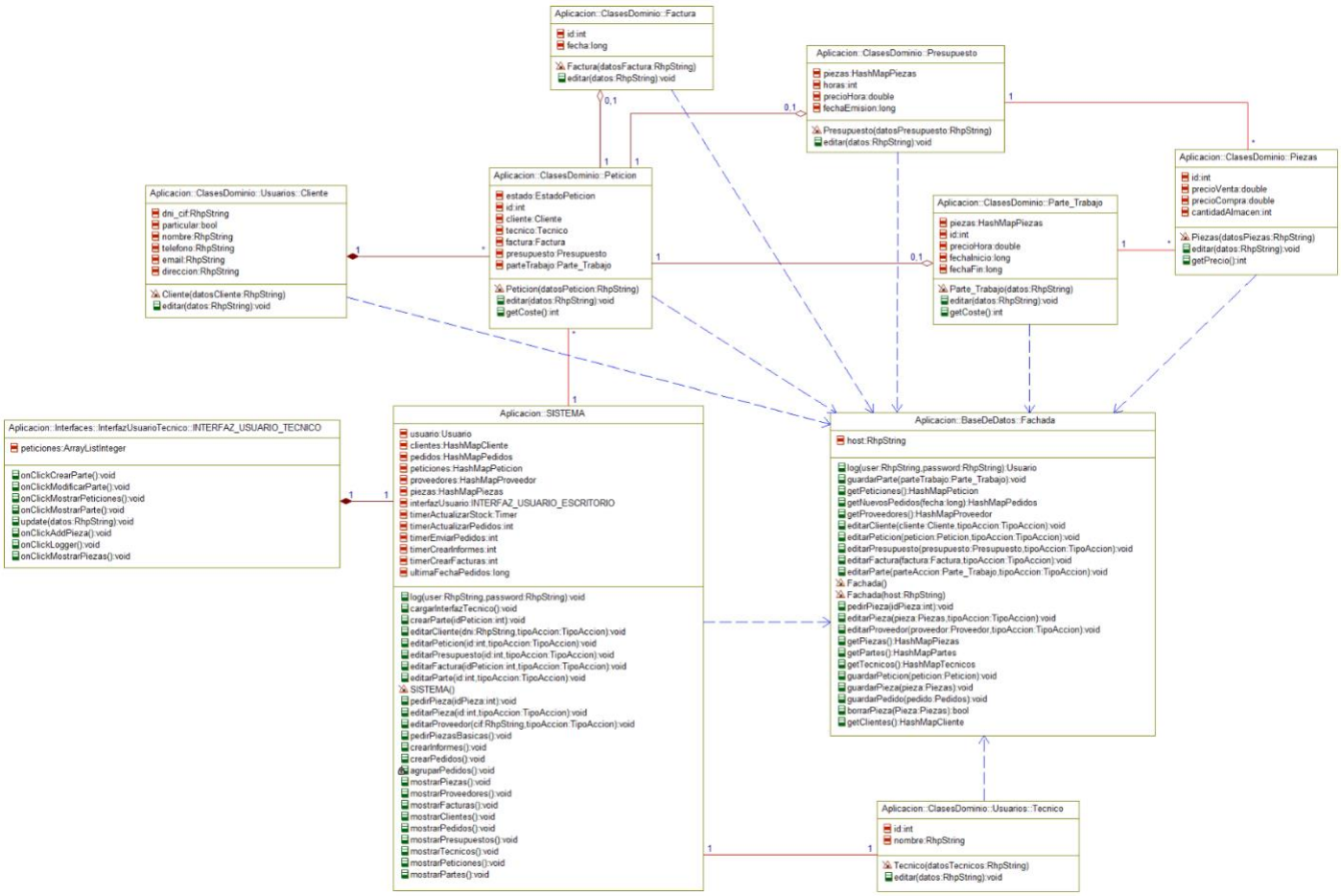
Tendremos una interfaz exclusiva para esta aplicación que se comunicará con el sistema del mismo modo que las otras mediante el patrón controlador.

También tendremos una fachada de la base de datos cuya funcionalidad ya ha sido explicada previamente.

De forma similar a las clases usuario en la aplicación de escritorio, en la aplicación movil cuando realicemos el log in a través de la fachada obtendremos una clase técnico que nos representará dentro del sistema y que servirá a este para saber, por ejemplo, qué peticiones o partes mostrar cuando estas sean solicitadas, ya que recordemos que cada técnico puede ver las suyas.

En las clases de trabajo el técnico tendrá acceso a las peticiones que tendrán asignadas un cliente y que podrán tener asignadas tanto una factura como un presupuesto.

El parte de trabajo y el presupuesto podrán tener piezas. Destacamos que en esta aplicación no es necesario saber el proveedor de cada pieza, por lo tanto, no tendremos dicha clase.



8.1.3 Diagrama del cliente.

Esta aplicación será utilizada por los clientes en las versiones de coste medio y coste alto tal y cómo explicamos en el artefacto de visión.

Tendremos una clase interfaz con la que el cliente podrá comunicarse con el sistema y una clase fachada con la que el sistema podrá comunicarse con la base de datos.

Cuando el cliente realice el log in en el sistema siendo este verificado por la base datos, obtendremos una clase cliente que le representa de modo que podamos saber de que cliente se trata.

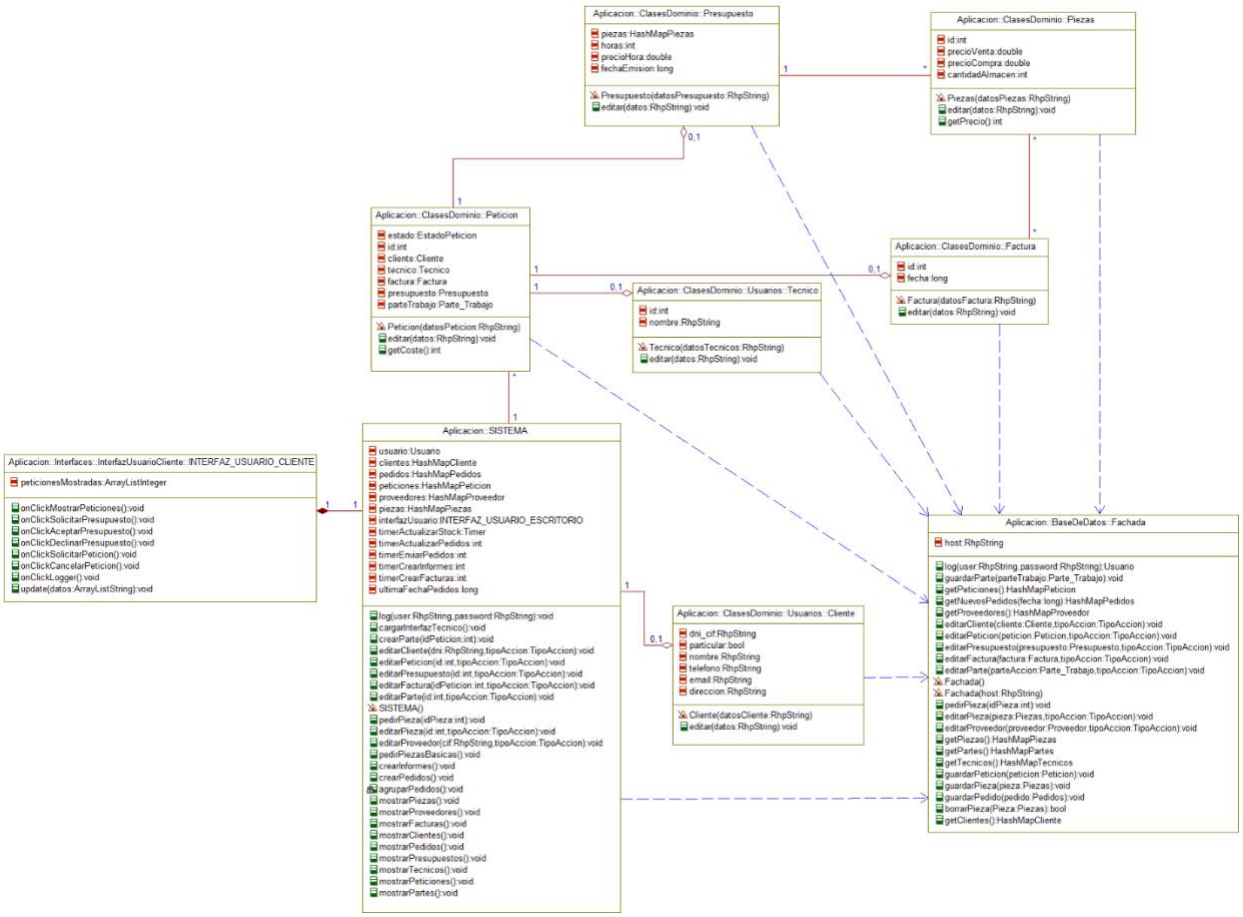
En esta aplicación el sistema tendrá acceso a clases petición que podrán tener asignadas un técnico, un presupuesto o una factura.

Tanto el presupuesto como la factura podrán tener piezas. Cabe destacar que en esta aplicación no nos interesa el tipo de pieza que esté en los presupuestos o facturas, por lo que sólo tendremos la clase padre y no las clases hijas (piezas especiales y piezas básicas).

Tampoco nos interesan datos como el parte de trabajo que es un documento interno del sistema ni los proveedores.

Las facturas que en ninguno de los otros modelos de diseño ni en el modelo de dominio ni en el de datos se relacionaban directamente con las piezas, sino que esta relación se realizaba a través del parte de trabajo, en esta aplicación si se hará directamente ya que la cardinalidad máxima nos lo permite, siendo en el caso de la cardinalidad mínima imposible ver las piezas que contiene una factura, ya que esta ni siquiera existiría.

Siempre que exista la factura podremos ver las piezas que contenga la petición de trabajo, pues la existencia de esta es previa y necesaria respecto de la de la factura.



8.1.4 Diagrama del Sistema.

Esta aplicación no tendrá interfaz. La clase sistema se controlará a si misma suscribiéndose a llamadas de callback mediante las clases timer.

El sistema creará un timer proporcionándole una frecuencia, una referencia a sí misma y un método al que llamar. Con la frecuencia indicada el timer llamará al método a través de la referencia a la instancia del sistema.

Tendremos una clase fachada con la que nos comunicaremos a la base de datos, una clase gestor de archivos con la que subiremos archivos json al servidor php que contendrán formateados los informes.

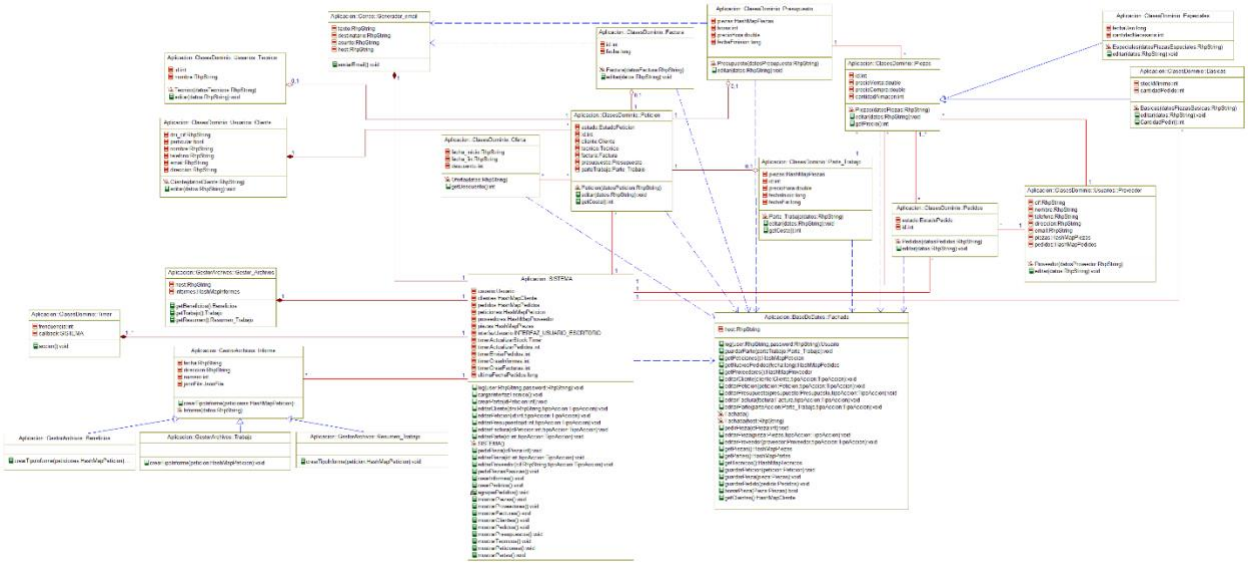
Como clases de trabajo tendremos los informes, las peticiones, las piezas, los proveedores, las facturas, presupuestos, partes de trabajo, técnico y cliente.

Destaca la clase gestor de email la cual es capaz de establecer una conexión a un servidor de correo electrónico y publicar nuevos correos en él. Esto es utilizado por ejemplo para solicitar pedidos a los proveedores o enviar las facturas a los clientes.

Las principales acciones que la clase sistema realizará en esta aplicación (serán lanzadas como se ha explicado mediante los timers) serán:

- Solicitar a la base de datos los pedidos que haya en la tabla pedidos cuya fecha sea superior a la fecha de la última vez que se realizó esta acción. Dichos pedidos serán agrupados por el proveedor y a dicho proveedor se le enviará un email en el que se le pedirán las piezas que contenían los pedidos.

- Diaría o semanalmente según como corresponda se crearán los informes en formato json pidiendo a la base de datos los datos necesarios para hacerlos y usando el gestor de archivos para publicarlos en el servidor php.
- Periódicamente y de forma similar a como se hace con los pedidos se obtendrá de la base de datos el listado de facturas nuevas y se enviarán estas a los clientes mediante un email.
- Cada cierto tiempo el stock de las piezas será actualizado en función de los nuevos pedidos con estado recibido y de los partes de trabajo.



8.2 Diagramas de secuencia de Sistema.

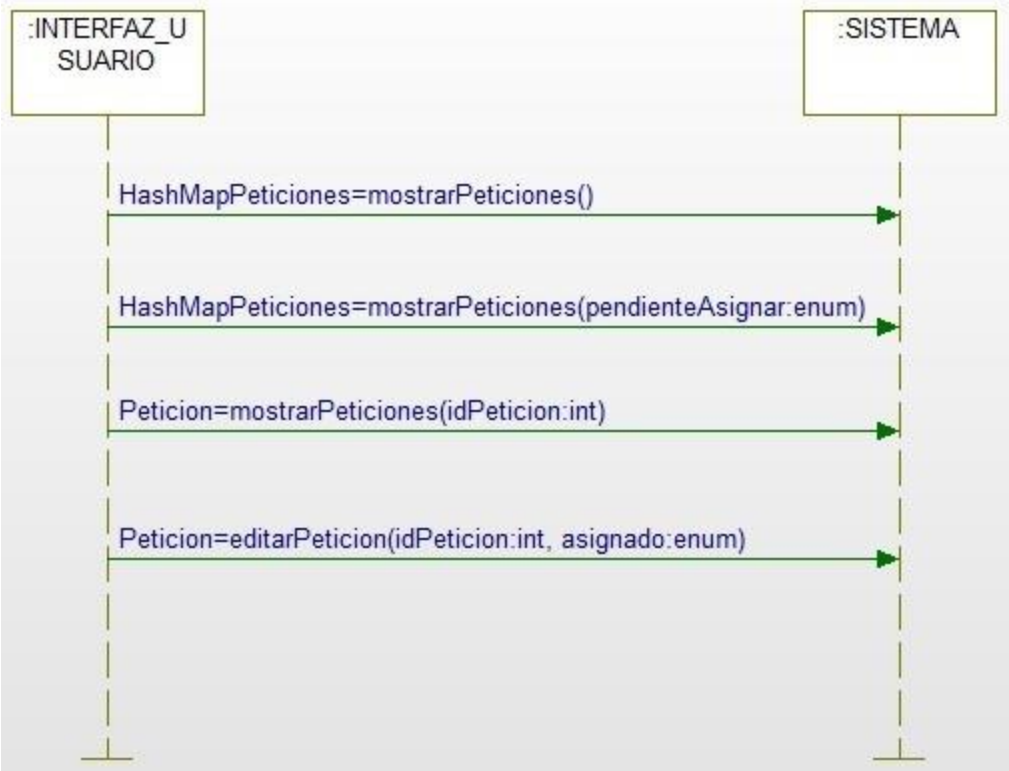
Un diagrama de secuencia del sistema consiste en un modelado que muestra, para cada escenario definido para un caso de uso concreto, los eventos que generan los actores participes del evento, indicando también el orden y los eventos entre los sistemas.

La forma de realizar estos diagramas es común para todos, partimos del escenario principal definido en el artefacto 5 (el modelado de los casos de uso) y a partir de ahí establecemos las comunicaciones que tiene el actor (en nuestro caso utilizaremos como actor las interfaces, ya que es la que se comunica con nuestro sistema).

Los diagramas de secuencia creados para esta iteración son:

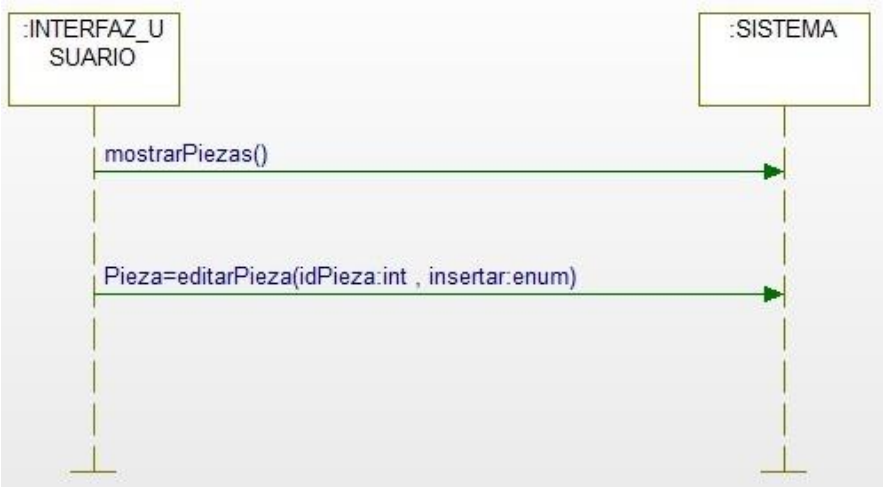
- Asignar petición de trabajo:

El actor por medio de la interfaz solicitar ver el listado completo de las peticiones de trabajo, con lo que obtendría el HashMap al completo. A continuación, desearía obtener un HashMap con las peticiones cuyo estado se encuentra en pendiente de asignación, la seleccionaría para obtener todos los datos y seleccionaría un técnico del listado que será el encargado de realizar ese trabajo.



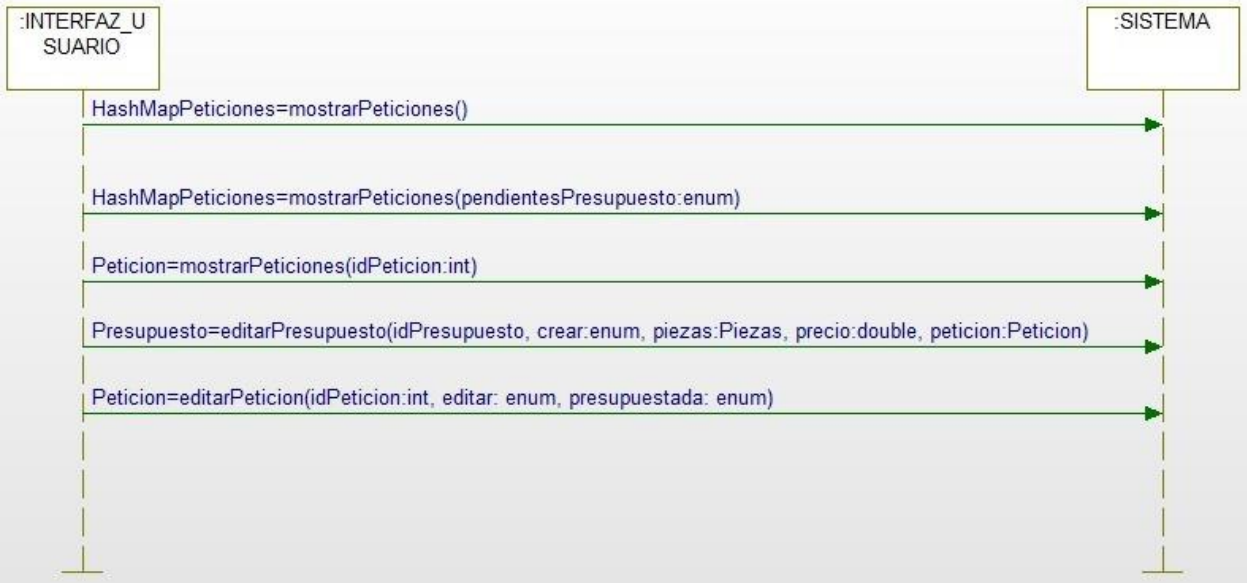
- Añadir pieza

Utilizando la interfaz de usuario, el responsable de almacén obtendría el hashMap de todas las piezas que se encuentran registradas en el sistema, luego quiere obtener un listado con únicamente aquellas peticiones que no cuentan con ningún técnico, accedemos a la información de la petición en concreto a la cual añadimos un técnico encargado.



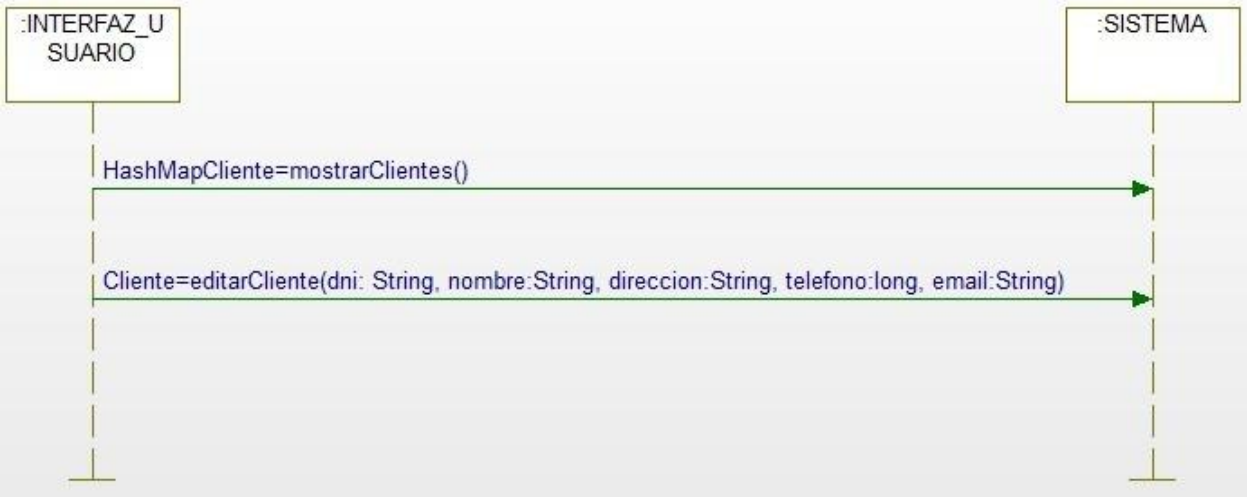
- Crear un presupuesto a una petición de trabajo

El usuario desea visualizar el menú en el que se muestra todo el listado de las peticiones registradas en el sistema, después únicamente desea ver las que están pendientes de presupuestar, muestra una petición en concreto que es la que se desea presupuestar, se introducen los datos y se cambia el estado de la petición.



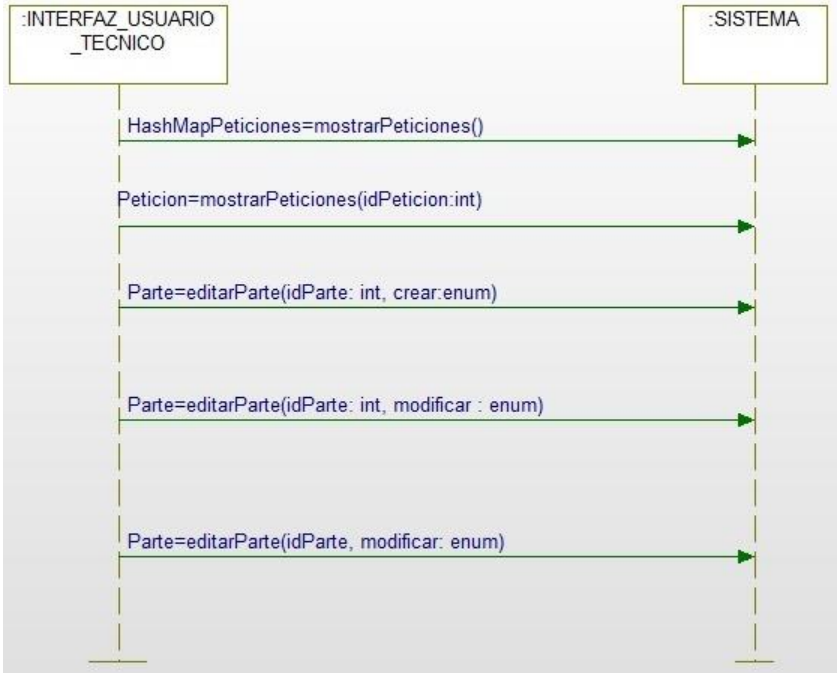
- Dar de alta cliente

Para registrar un cliente nuevo, primero se muestra el listado de todos los clientes, después se pulsaría sobre



- Crear parte de trabajo

Nuestro técnico cuenta con una interfaz diferente, ya que accede al sistema con un dispositivo móvil. Para crear un nuevo parte de trabajo, primero visualizamos todas las peticiones (ya que no podemos crear un parte sin asociarlo a una petición) por lo que el sistema nos devuelve un HashMap con todas las peticiones, después seleccionamos una y el sistema nos muestra la información de esa petición en particular. Pasamos a editarla, y se crea un parte con una fecha de inicio (la actual en ese momento en el dispositivo), luego se añaden las piezas necesarias y finalmente se finalizaría el parte (por lo que se añade la fecha actual en el dispositivo como fecha de finalización del parte de trabajo).



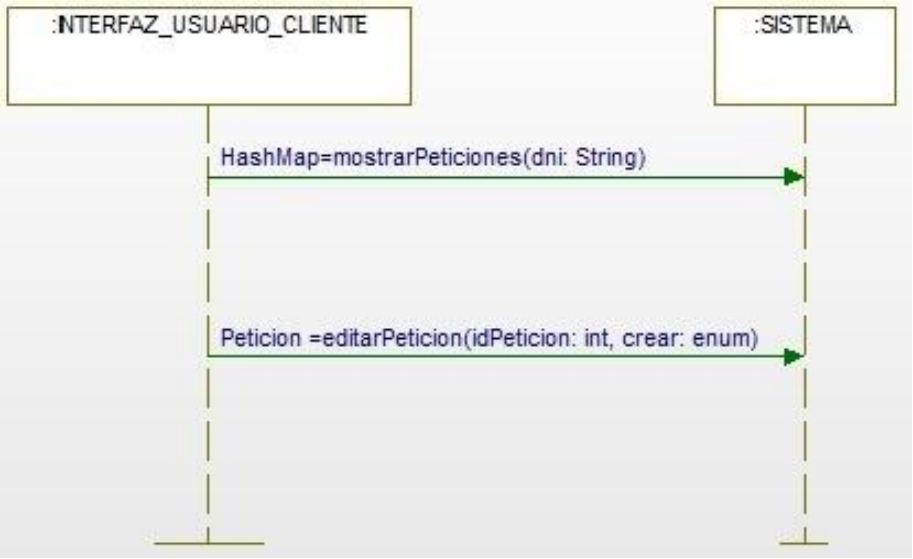
- Crear pedidos de piezas especiales

El responsable de almacén accede al menú de piezas, donde se desplegaría todo el listado de piezas (el sistema retorna un HashMap con las piezas), después se selecciona una (el sistema

muestra los datos de esa pieza) y finalmente se crea un pedido con el número de piezas necesarias y el identificador de las piezas que necesitamos.

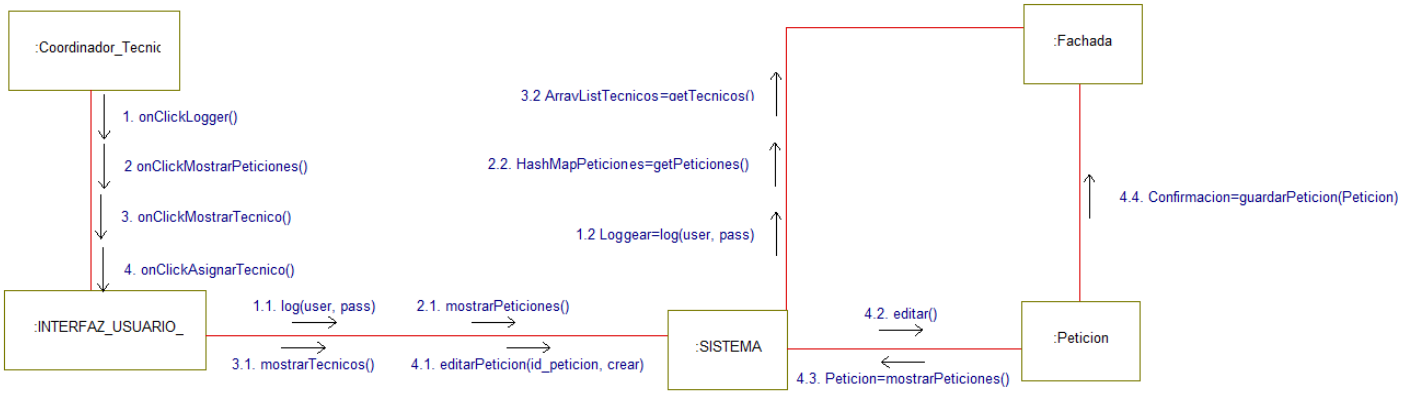
- Solicitar Peticiones trabajo

La interfaz del cliente es diferente, ya que no deberá tener acceso a mucha información. El cliente tras acceder al sistema visualizaría el listado de las peticiones de trabajo que ha solicitado (ya que el sistema le retorna un HashMap con todas las peticiones asociadas al dni del cliente), por ultimo desearía añadir una nueva solicitud con los datos correspondientes, a lo que el sistema respondería creando una petición con dichos datos.

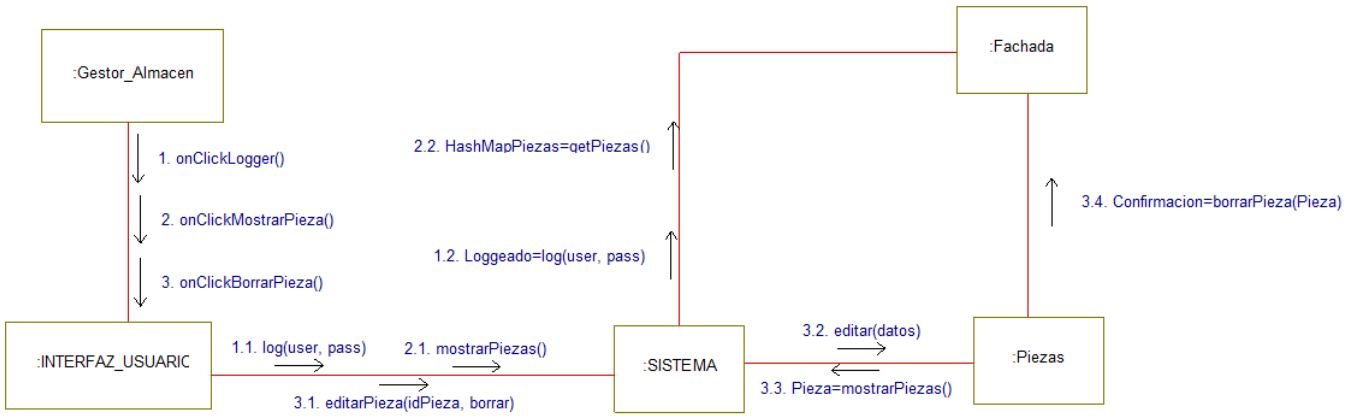


8.3 Diagramas de Colaboración.

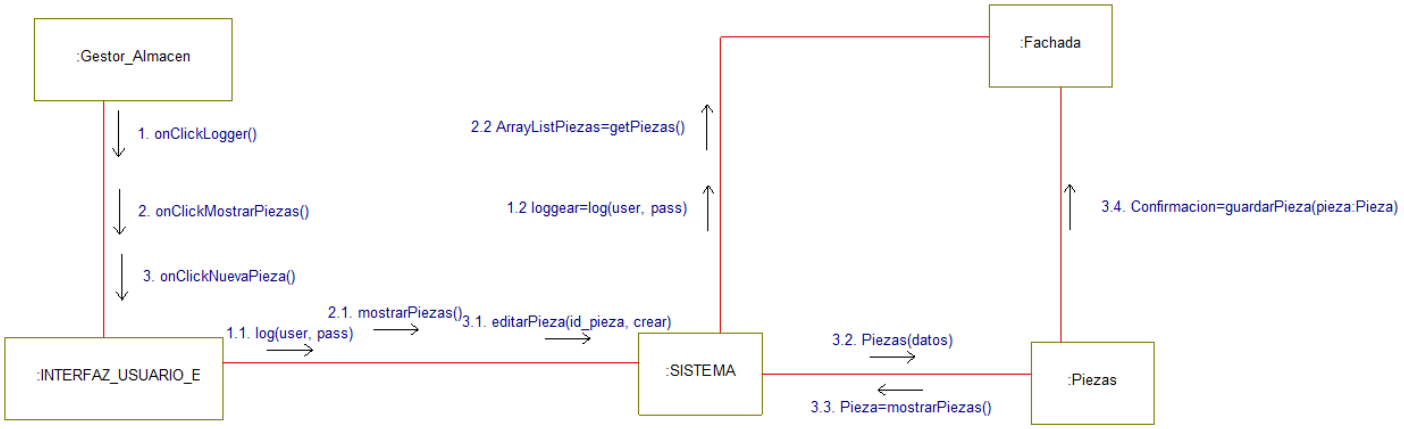
- Asignar Petición Trabajo



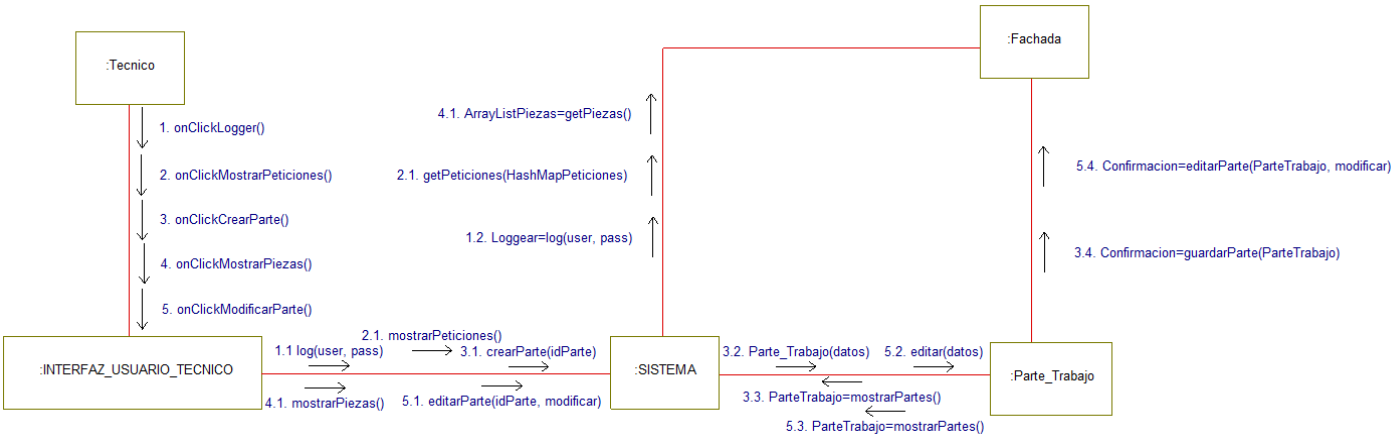
- Borrar Piezas



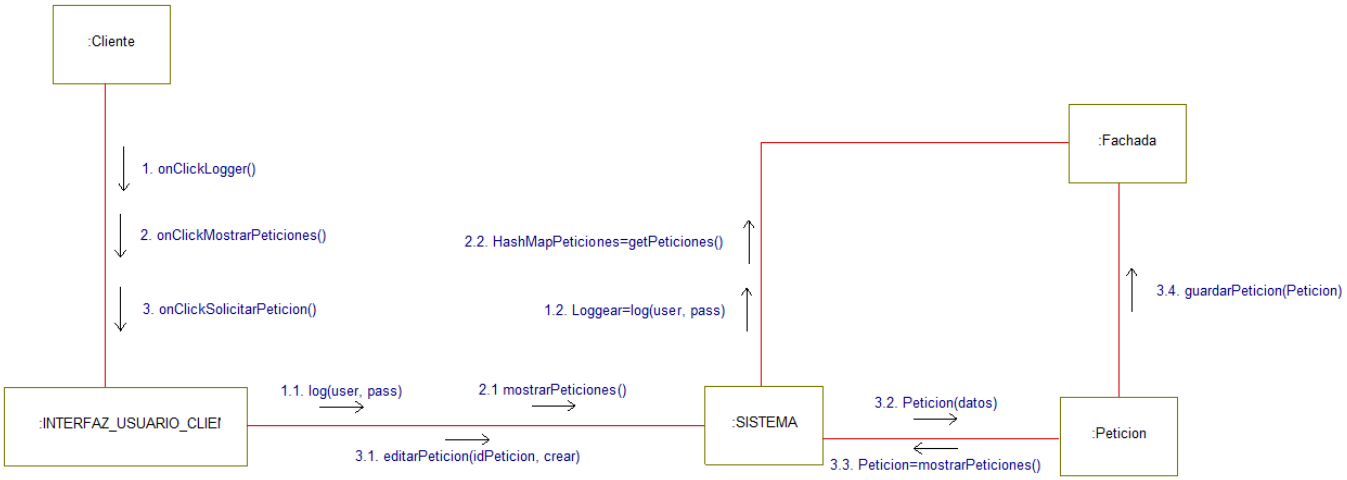
- Add Pieza



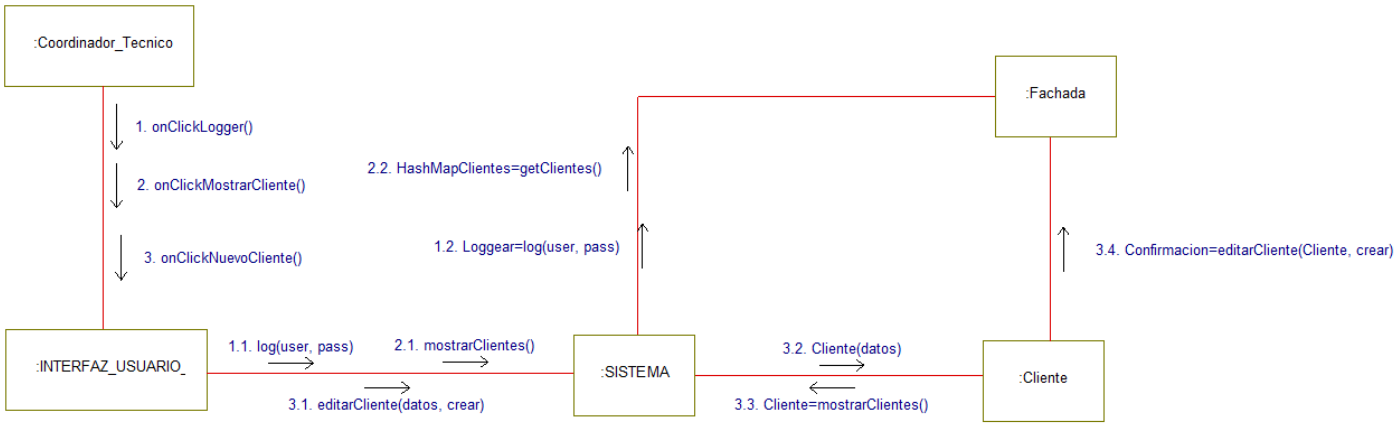
- Crear Parte Trabajo



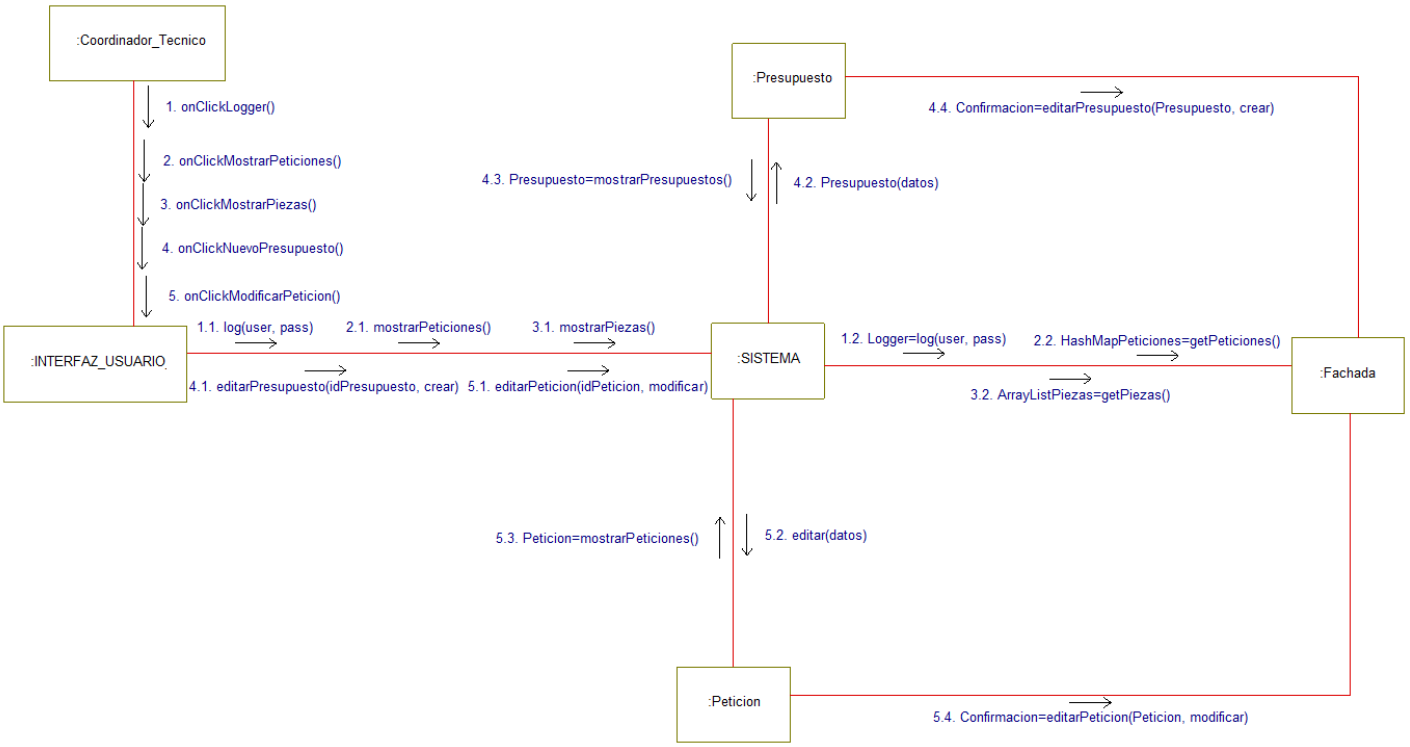
- Solicitar Petición Trabajo



- Dar Alta Cliente



- Crear Presupuesto



- Crear Pedido Piezas Especiales