Universidad de Valladolid

E.U. DE INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

GESTIÓN DE UN BANCO DE TIEMPO ENTRE LOS MIEMBROS DE UNA COMUNIDAD UNIVERSITARIA

Alumno: Emeterio Galán Álvarez
Tutora: Amelia García Garrosa

" Poner al dinero como bien supremo nos conduce a la catástrofe "

José Luis Sampedro (1917 – 2013)

"If time be of all things the most precious,
wasting time must be the greatest prodigality."

“Si el tiempo es lo más valioso,
la pérdida de tiempo es el mayor de los derroches.”

Benjamin Franklin (1706-1790)

Agradecimientos:

Me gustaría agradecer a mi tutora Dña. Amelia García Garrosa la dedicación y el tiempo dedicado al trabajo, haciendo todo lo posible para que el trabajo se terminará a tiempo a pesar de las circunstancias.

También quisiera agradecer la ayuda de D. Aníbal Bregón Bregón y D. Miguel Ángel Martínez Prieto, por su ayuda durante el curso en las diferentes fases del trabajo.

Y no puedo olvidarme de Marta, mis amigos y familia, que han aguantado las malas caras, los cabreos y los monólogos relacionados con el trabajo, una y otra vez, además de darme ánimo y fuerza para que el trabajo llegara a su fin.

Índice de contenido

1.- Introducción

1.1. - Motivación

En la actualidad estamos pasando momentos bastante difíciles, nuestra situación económica es complicada y hay que potenciar la solidaridad entre la gente. Hoy en día, ayudarnos unos a otros es sobre todo una satisfacción personal. Esos valores deben potenciarse en lugar de seguir en una etapa en la que predomine el dinero. En este contexto económico puede ser buena idea poner en marcha un banco de tiempo.

Un banco de tiempo es un sistema de intercambio de servicios por tiempo. En él, la unidad de intercambio no es el dinero habitual sino una medida de tiempo, por ejemplo, el trabajo por hora. Es un sistema de intercambio de servicios por servicios o favores por favores. Propone la ventaja de fomentar las relaciones sociales y la igualdad entre distintos estratos económicos.

El banco de tiempo en el que nos centraremos permitirá a cualquier miembro de la comunidad universitaria (alumnos, profesores y personal de administración y servicios) la posibilidad de consumir u ofertar un servicio.

El objeto de este TFG será la implementación de una aplicación web y la implementación de una aplicación móvil (a modo de demostración) , para una gestión ágil de nuestro banco de tiempo.

El sistema permitirá a los usuarios ofertar servicios que podrán ser consumidos, buscar entre los servicios existentes y en caso de no encontrar un servicio que le parezca adecuado, hacer una propuesta para satisfacer su necesidad. Los usuarios que estén registrados podrán ver las propuestas pendientes y si lo desean crear un servicio que satisfaga una propuesta.

Los servicios tienen un precio en horas que se sumarán o restarán del saldo de los usuarios en el momento que se realice una consumición.

Existirán moderadores que velarán por el buen funcionamiento de la aplicación y que podrá retirar propuestas y servicios, como controlar las cuentas de usuario y su saldo. También consultará todos los datos la base de datos.

1.2. - Dominio del problema

El software a implementar tiene por objeto gestionar un banco de tiempo que permita a los usuarios de la Universidad de Valladolid el consumo de servicios que se encuentren ofertados. La Universidad de Valladolid, según nuestros datos, cuenta con más de 30.000 alumnos, más de 2.000 profesores y 1.000 PAS, que se reparten en 25 centros.

El conocimiento informático que se espera de los usuarios habituales es muy bajo, pero se requiere de personas con mayor conocimiento para poder realizar las labores de mantenimiento y moderación.

Se quiere facilitar el acceso al mayor número de miembros de la comunidad universitaria. Por ello se ha optado por la creación de una aplicación web, ya que la mayoría de las personas que

utilizan un ordenador esta familiarizado con el uso de los navegadores web. Además de que éstos se encuentran instalados en la mayoría de los sistemas operativos por defecto consiguiendo que los requisitos de la aplicación sean fácilmente alcanzables.

No se dispondrá, por temas de confidencialidad y seguridad, de los datos de las personas de la Universidad y la aplicación se ejecutará en un servidor independiente de los recursos de la Universidad por lo que el sistema puede ser implantado en cualquier lugar.

Los usuarios accederán al sistema mediante una cuenta de usuario, ésta será un correo de la Universidad y una contraseña. La contraseña será generada por el sistema y será temporal, en el caso de que el usuario no se quiera registrar en el sistema, o será la deseada por el usuario en caso registrarse en la aplicación.

Una cuenta de usuario sólo será válida mientras la persona sea miembro de la universidad, y esta característica puede cambiar cada cuatrimestre.

Las cuentas de los usuarios pueden ser atacadas, ya que todos los usuarios cumplen un patrón que es el correo de la universidad, que fácilmente se puede intuir o conseguir.

Se debe tener en cuenta, que las personas también pueden olvidar la contraseña de su usuario y realizar varios intentos fallidos de acceso, sin ser éstos un ataque a la cuenta.

La aplicación móvil, correrá sobre dispositivos con sistema operativo Android, en la API 17 que proporciona Google.

1.3. - Objetivos y desarrollo de la idea

Se desea crear una aplicación para una comunidad universitaria en la que todos los miembros puedan participar.

La aplicación permitirá a un usuario registrado realizar servicios a cambio de “horas” y después éstas podrán usarse para que el usuario en cuestión pueda consumir otros servicios ofertados por otro miembro de la universidad registrado en la aplicación.

Esto puede ser un paso previo al registro como UsuarioBT que le permitiría tener un primer contacto con la aplicación para ver qué servicios están ofertados y proponer alguno que le interese. Si finalmente decide participar en el Banco de Tiempo para ofertar y consumir servicios será necesario que se registre como UsuarioBT

Los miembros de la universidad podrán registrarse en la aplicación, pasando a ser usuarios del banco de tiempo (UsuarioBT), ésto les permitirá acceder a todas las funcionalidades. Si por el contrario un miembro de la universidad no quiere registrarse en la aplicación, podrá acceder a ella solicitando una contraseña temporal (UsuarioUVa), pero con funcionalidades muy limitadas, como la consulta de los servicios y la opción de poder añadir propuestas.

Para garantizar que los usuarios son miembros de la universidad, se necesitará obligatoriamente la posesión de una cuenta de correo con formato “*****@*****.uva.es”. Ya que los miembros de la universidad cuando dejan de de serlo, pierden su cuenta de correo a los seis

meses, las cuentas de usuario del banco de tiempo también serán bloqueadas y se necesitará de una comprobación mediante correo de que la cuenta sigue activa.

El control de la aplicación la realizarán los moderadores. Durante la instalación se creará uno por defecto que dará permisos de moderación a un miembro real e inmediatamente el primero deberá ser borrado.

Los usuarios registrados (UsuarioBT) y los usuarios no registrados (UsuarioUVa), podrán usar un buscador para buscar entre los servicios que se encuentran en el sistema,añadir propuestas y ver aquellas que ha propuesto, además de ver un listado de las propuestas que hay pendientes y de los servicios.

El usuario registrado podrá añadir servicios individualmente o podrá crear servicios que satisfagan una propuesta que se encuentre publicada en la aplicación.

En caso de crear un servicio que satisfaga una propuesta éste no tiene que ser exclusivo de ésta por eso el servicio es independiente de la propuesta concreta, porque un mismo servicio puede ser más general que la descripción de una propuesta. En el momento en que se crea un servicio que satisfaga una propuesta, automáticamente se creará una consumición aceptada.

Una vez creados los servicios, el UsuarioBT podrá consultarlos, y modificarlos, y ver las peticiones que tiene sin contestar de ese servicio.

Además de la aplicación web, existirá una aplicación para móviles android que permitirá a los usuarios la gestión rápida de las peticiones que se hagan a sus servicios.

El UsuarioBT podrá ver el detalle de los servicios y pedir aquéllos que desee, siempre que tenga saldo suficiente.

Las peticiones crean una consumición en estado pendiente, y en el momento que el dueño del servicio acepte la consumición se producirá el cobro y abono de las horas.

Los usuarios podrán ver el histórico de sus consumiciones, tanto las que han solicitado y han sido aceptadas (servicio comprado) como las que han realizado y han aceptado (servicio vendido).

Los usuarios podrán ver y actualizar algunos de sus datos personales, como la contraseña, el campus o la residencia.

Para la moderación del sistema se requerirá de un usuario registrado (UsuarioBT) con permisos de moderación (Moderador).

El moderador podrá cambiar las características de un usuario (UsuarioBT) o el tipo de miembro (alumno, PAS o PDI) .

También puede gestionar el estado de su cuenta (activada, desactivada, bloqueada o cancelada). Sólo se podrá acceder al sistema si la cuenta está activada, los otros tres estados indican la importancia de impedir el acceso.

La cuenta puede estar desactivada por la necesidad de comprobar el correo o circunstancias menores, y éste estado puede ser cambiado automáticamente por el sistema.

El estado de cuenta bloqueada o cancelada, da información al resto de los moderadores sobre la situación de la cuenta, ya que estos estados fueron asignados explícitamente por uno de los moderadores.

Una cuenta bloqueada indica que el usuario está teniendo un comportamiento en la aplicación que no es el adecuado y ha sido penalizado o se están estudiando las medidas a tomar.

Una cuenta cancelada indica que ese usuario ha dejado la aplicación o ha sido expulsado de ella.

Si un usuario ha realizado tres intentos fallidos de acceso al sistema no dejará acceder con esa cuenta, por lo que es necesario resetear el número de intentos mediante una petición a un moderador.

Los moderadores podrán modificar el saldo de los usuarios, estableciendo abonos o penalizaciones , y ajustando el saldo en caso de que un servicio no se haya realizado, o hubiera un fallo en el sistema.

Los moderadores podrán descargar todos los datos de la aplicación, a excepción de las contraseñas. Dichas contraseñas se encuentran encriptadas en el sistema y sería innecesario y no contienen ningún tipo de información valiosa para el moderador y sería innecesario tener acceso a ellas.

Los datos se descargan en hojas excel para poder hacer un tratamiento de datos si es necesario, permitiendo consultar la aplicación y los datos simultáneamente.

1.4. - Estado del arte y herramientas

1.4.1. - Estado del arte

La Plataforma Software Empresarial elegida, para la realización ha sido WAMP, la versión para el sistema operativo Windows de LAMP, y la versión utilizada la número 2.4.

La Plataforma consta de los siguientes componentes.

- Windows, como sistema operativo;
- Apache, como servidor web;
- MySQL, como gestor de bases de datos;
- PHP (generalmente), Perl, o Python, como lenguajes de programación
- jQuery, como framework Javascript.

Esta plataforma usa Windows 7 como sistema operativo, en el que montaremos los demás componentes de la plataforma.

Apache (Versión 2.4.4) , como servidor HTTP para poder acceder a la aplicación a través de un navegador desde cualquier dispositivo.

MySQL (Versión 5.6.12) como sistema gestor de base de datos, en el que guardaremos todos los datos referentes a los usuarios y los servicios que oferta la aplicación.

Y como lenguaje de programación PHP, que nos permite crear páginas dinámicas de la que se puede recuperar información almacenada en bases de datos.

La plataforma interpreta y da soporte a la versión de PHP 5.4.12.

La librería de Javascript jQuery permite interactuar de manera sencilla con los elementos HTML, en su versión 2.1.0.

1.4.2. - Herramientas utilizadas

Las herramientas utilizadas para la creación de la aplicación fueron:

1. Windows: Sistema operativo sobre el que se montará la aplicación.

2. Apache: Servidor HTTP que nos proporciona el acceso mediante un navegador.

3. MySQL: Sistema gestor de bases de datos para almacenar todos los datos necesarios para el funcionamiento de la aplicación

4. NotePad ++ : Entorno de desarrollo para programar en el lenguaje PHP, creación de estilos CSS y programación de JavaScript.

5. Evolus Pencil : Desarrollo de gráficos de la documentación.
6. StarUml: Creación de los diagramas de análisis y de diseño.
7. DIA: Creación de el diagrama de Entidad-Relación.
8. Gantt Project: Creación de los diagramas de Gantt.
9. Google Chrome: Como navegador y depurador de la aplicación.
10. Mozilla Firefox: Como navegador y depurador de la aplicación.
11. Eclipse ADT: Para el desarrollo de la aplicación móvil.

1.3. - Metodología

Modelo Espiral: es una metodología que combina las principales ventajas de los modelos en cascada e iterativo y permite obtener prototipos de la aplicación cada vez más completos.

Para el proyecto se ha decidido realizar dos vueltas a la espiral. Cada una de estas iteraciones constará de las siguientes fases:

1. Análisis
2. Diseño
3. Implementación
4. Pruebas

Además, la iteración uno constará con una fase de planificación.

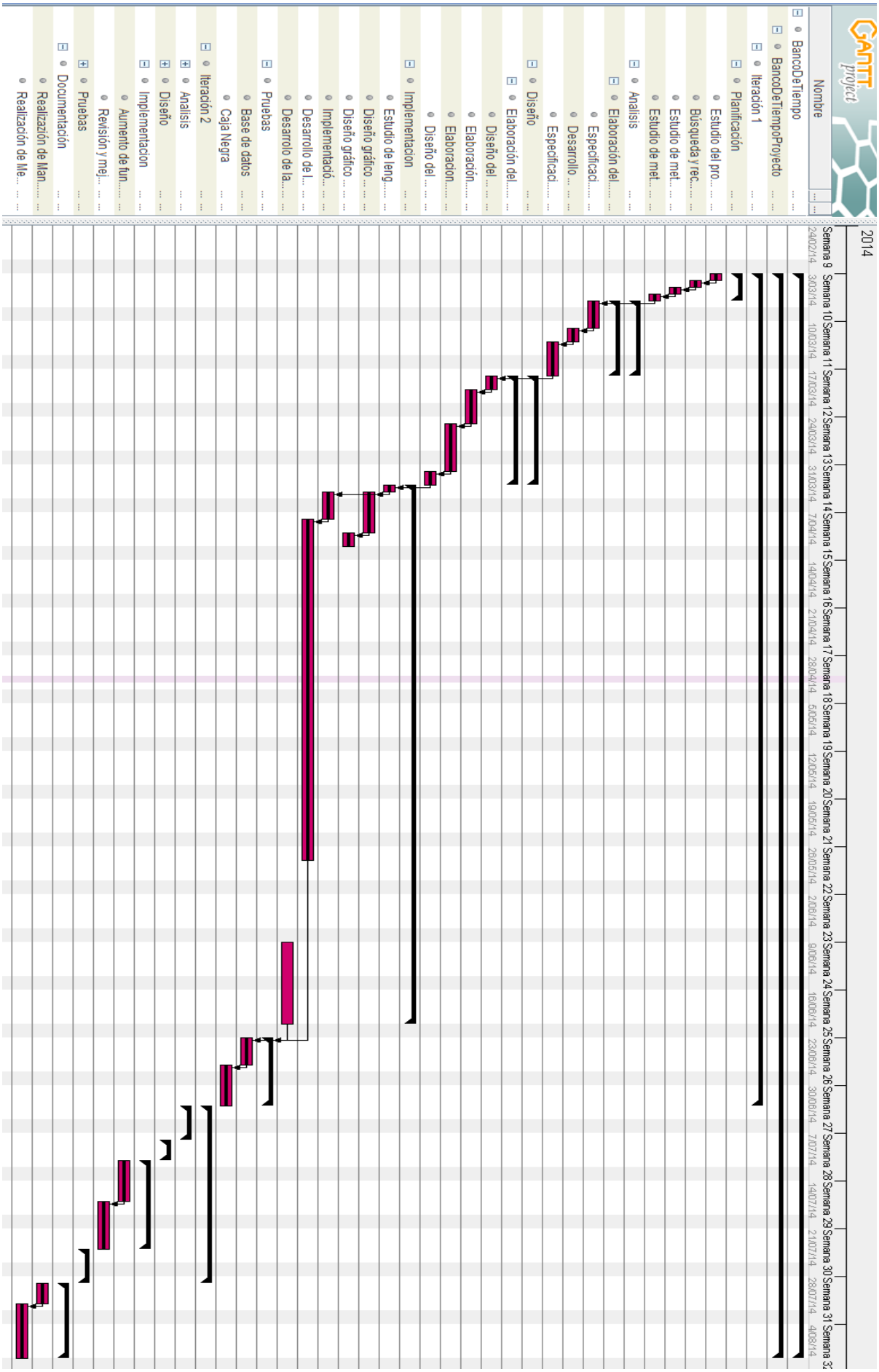
Por último, después de obtener el producto tras las dos iteraciones, se realiza una fase de documentación.

2.- Planificación y Presupuesto

2.1. - Planificación (Diagrama de Gantt)

GANTT project

Nombre	Fecha de inicio	Fecha de fin
BancoDeTiempo	3/03/14	8/08/14
BancoDeTiempoProyecto	3/03/14	8/08/14
Iteración 1	3/03/14	2/07/14
Planificación	3/03/14	6/03/14
Estudio del proyecto a desarrollar	3/03/14	3/03/14
Búsqueda y recopilación de información	4/03/14	4/03/14
Estudio de metodologías y patrones a utilizar	5/03/14	5/03/14
Estudio de metodologías y patrones a utilizar	6/03/14	6/03/14
Analisis	7/03/14	17/03/14
Elaboración del DRS	7/03/14	17/03/14
Especificación de requisitos y restricciones	7/03/14	10/03/14
Desarrollo de Casos de Uso	11/03/14	12/03/14
Especificación de Casos de Uso	13/03/14	17/03/14
Diseño	18/03/14	2/04/14
Elaboración del DAS	18/03/14	2/04/14
Diseño del diagrama de clases del sistema	18/03/14	19/03/14
Elaboración de diagramas de comportamiento (secuencia y estados)	20/03/14	24/03/14
Elaboracion de diagramas de diseño del sistema (objetos y secuencia)	25/03/14	31/03/14
Diseño del diagrama Entidad-Relación	1/04/14	2/04/14
Implementacion	3/04/14	20/06/14
Estudio de lenguajes y tecnologías a aplicar	3/04/14	3/04/14
Diseño gráfico de la interfaz de la aplicación	4/04/14	9/04/14
Diseño gráfico de elementos a utilizar con la vista	10/04/14	11/04/14
Implementación de la Base de Datos y sus funciones de control	4/04/14	7/04/14
Desarrollo de la aplicación	8/04/14	27/05/14
Desarrolo de la Aplicación Móvil	9/06/14	20/06/14
Pruebas	23/06/14	2/07/14
Base de datos	23/06/14	26/06/14
Caja Negra	27/06/14	2/07/14
Iteración 2	3/07/14	28/07/14
Analisis	3/07/14	7/07/14
Ampliación y revisión del documento DRS	3/07/14	7/07/14
Diseño	8/07/14	10/07/14
Ampliación y revisión del documento DAS	8/07/14	10/07/14
Implementacion	11/07/14	23/07/14
Aumento de funcionalidades en la aplicación	11/07/14	16/07/14
Revisión y mejora de código	17/07/14	23/07/14
Pruebas	24/07/14	28/07/14
Caja Negra	24/07/14	28/07/14
Documentación	29/07/14	8/08/14
Realizazión de Manual de Usuario	29/07/14	31/07/14
Realización de Memoria	1/08/14	8/08/14


GANTT project
Nombre
BancoDeTiempo
BancoDeTiempoProyecto
Iteración 1
Planificación
Estudio del pro...
Búsqueda y rec...
Estudio de met...
Estudio de met...
Análisis
Elaboración del...
Especificaci...
Desarrollo ...
Especificaci...
Diseño
Elaboración del...
Diseño del...
Elaboración...
Elaboración...
Diseño del...
Implementacion
Estudio de leng...
Diseño gráfico ...
Diseño gráfico ...
Implementació...
Desarrollo de l...
Desarrollo de la...
Pruebas
Base de datos
Caja Negra
Iteración 2
Analisis
Diseño
Implementacion
Aumento de fun...
Revisión y mej...
Pruebas
Documentación
Realización de Man...
Realización de Me...
2014
Semana 9 24/02/14
Semana 10 03/03/14
Semana 11 10/03/14
Semana 12 17/03/14
Semana 13 24/03/14
Semana 14 31/03/14
Semana 15 7/04/14
Semana 16 14/04/14
Semana 17 21/04/14
Semana 18 28/04/14
Semana 19 5/05/14
Semana 20 12/05/14
Semana 21 19/05/14
Semana 22 26/05/14
Semana 23 2/06/14
Semana 24 9/06/14
Semana 25 16/06/14
Semana 26 23/06/14
Semana 27 30/06/14
Semana 28 7/07/14
Semana 29 14/07/14
Semana 30 21/07/14
Semana 31 28/07/14
Semana 32 4/08/14

2.2. - Estimación y presupuesto.

Entradas: 6	Salidas:0	Consultas:10	Ficheros Externos:1	Ficheros Internos:1
Usuario		Buscar Usuario	Copia de Seguridad	Base de Datos
Moderador		Buscar Servicio		
Servicio		Buscar Propuesta		
Consumición		Consultar Servicio		
Mensaje		Consultar Propuesta		
Propuesta		Consultar Oferta		
		Consultar Base de Datos		
		Gestionar Propuesta		
		Gestionar Servicio		
		Gestionar Usuario		

Calculo de los Puntos de Función No Ajustados (PFNA):

	Bajo	Medio	Alto	Total
Entradas	1*3=3	2*4=8	3*6=18	29
Salidas	0*4	0*5	0*7	0
Consultas	3*3=9	4*4=16	3*6=18	43
Ficheros Internos	0*7	0*10	1*15=15	15
Ficheros Externos	0*5	1*7	0*10=15	7
				94

Después de obtener los PFNA ,éstos deben ser ajustados mediante un Factor de Ajuste (FA), que se calcula sobre 14 características generales de los sistemas. A cada característica se le atribuye un peso de 0 a 5 según la complejidad o influencia.

Factores de ajuste	Complejidad
1.- Comunicación de datos	4
2.- Funciones distribuidas	2
3.- Rendimiento	5
4.- Gran carga de trabajo	5
5.- Frecuencia de transacciones	4

6.- Entrada on-line de datos	4
7.- Requisito de manejo del usuario final	4
8.- Actualizaciones on-line	0
9.- Procesos complejos	3
10.- Utilización de otros sistemas	0
11.- Facilidad de mantenimiento	2
12.- Facilidad de operación.	2
13.- Instalación en múltiples lugares	0
14.- Facilidad de cambio	3
TOTAL:	38

FA = 0,65+(0,01*38) = 0,65+0,38= 1,03

PF = FA · PNFA= 1,03· 94 = 96,82

Cada punto de función equivale a 30 líneas de código en PHP. Luego el total de líneas de código resultantes será de:

96,82 pf · 30 ldc/pf= 2094,6 LDC → 2,1 KLDC

2.2.1. - COCOMO

Como es una aplicación con menos de 50 KLC, con pocas presiones de tiempo y está desarrollada en un entorno estable, se considera un modelo orgánico.

Esfuerzo Nominal

$$EN = 3,2 * (2,1)^{1,05} = 6,98 \text{ personas-mes (PM)}$$

Factores	Valor de los factores
Fiabilidad requerida	1,15 (Alto)
Tamaño de la base de datos	1,16 (Muy Alto)
Complejidad del software	0,85 (Bajo)
Restricciones de tiempo de ejecución	1 (Medio)
Restricciones de memoria volatilidad del hardware	1 (Medio)
Restricciones de tiempo de respuesta	1 (Medio)
Calidad de los analistas	1,07 (Alto)
Experiencia con el tipo de aplicación	1 (Medio)
Fiabilidad requerida	0,82 (Muy Alto)

Experiencia con el hardware	1 (Medio)
Experiencia con el lenguaje de programación	0,95 (Alto)
Calidad de los programadores	0,86 (Alto)
Técnicas modernas de programación	0,91 (Alto)
Empleo de herramientas	1 (Medio)
Restricciones a la duración del proyecto	1 (Medio)

Esfuerzo:

E = EN * Valor Factores

E= 6,98 * (1,15*1,16*0,85*1*1*1*1,07*1*0,82*1*0,95*0,86*0,91*1*1)

E= 6,98 * 0,74 = 5,1652 Persona-Mes (PM)

Tiempo de desarrollo:

$TD=2,5 * (5,1652)^{0,38}$ = 4,66 meses

Coste:

Considerar el sueldo medio de 1350 € para una persona.

C= 5,1652 * 1350 = 6973,02 €

Número medio de personas

5,1652 / 4,66 = 1,108 personas

2.2.2. - Recursos

Software

Producto	Precio	Uso	Coste
NotePad ++	0,00 €	60,00%	0,00 €
Microsoft Windows 7 Home Premium w/SP1 -1 PC	69,00 €	20,00%	13,80 €
WAMP	0,00 €	70,00%	0,00 €
Evolus Pencil	0,00 €	10,00%	0,00 €
Dia	0,00 €	10,00%	0,00 €
StarUml	0,00 €	40,00%	0,00 €
GanttProject	0,00 €	15,00%	0,00 €
Google Chrome	0,00 €	10,00%	0,00 €
Mozilla Firefox	0,00 €	10,00%	0,00 €
Eclipse ADT (Android Development Tools)	0,00 €	20,00%	0,00 €
Total	**13,80 €**		

Hardware

Producto	Precio	Uso	Coste
PC	640,00 €	33,00%	211,20 €
Internet	30,00 €/mes	200,00%	60,00 €
Impresora Multifunción	110,00 €	15,00%	16,50 €
Total	**287,70 €**		

Personal

Producto	Coste
Ingeniero	**6.973,02 €**
Total	**6.973,02 €**

PRESUPUESTO TOTAL : 13,80 + 287,70 + 6.973,02 = **7274,52 €**

3.- Análisis

3.1. - Identificación de los actores

Hay 3 tipos de actores principales, partiendo de la premisa de que todos los usuarios son miembros de la Universidad.

ACT-01	UsuarioUva
Descripción	Persona que es miembro de la universidad, pero que no está registrada en nuestro sistema. Puede proponer servicios que requiera, pero no puede ofertar servicios ni solicitarles.
Comentario	

Tabla 001

ACT-02	UsuarioBt
Descripción	Persona miembro de la universidad, y que está registrada en el sistema. Puede proponer servicios que requiera. Puede ofertar servicios. Puede consumir servicios de otros usuarios. Además puede comunicarse con otros usuarios.
Comentario	

Tabla 002

ACT-03	Moderador
Descripción	Persona miembro de la universidad, que está registrada en el sistema. Tiene privilegios y deberes sobre el funcionamiento de la aplicación, no se requiere un conocimiento amplio de informática. Pero debe de conocer las funciones que tiene que desempeñar y método de aplicación de éstas.
Comentario	

Tabla 003

3.2. - Requisitos Funcionales

RF-1. Permitir el ingreso al sistema de los miembros de la universidad.
RF-2. Los miembros de la universidad pueden hacer propuestas.
RF-3. Los miembros de la universidad pueden ver los servicios ofertados.
RF-4. Los miembros de la universidad no registrados podrán solicitar una contraseña. temporal que les permita usar el sistema durante un tiempo predefinido.
RF-5. Los miembros de la universidad podrán buscar servicios en la aplicación.
RF-6. Los miembros de la universidad podrán filtrar los servicio por tipo de servicio.
RF-7. Los miembros de la universidad podrán visualizar las propuestas no satisfechas que se encuentren en el sistema.
RF-8. Los miembros de la universidad podrán gestionar (editar y eliminar) sus propuestas.
RF-9. Los usuarios registrados en el sistema deberán indicar el campus al que pertenecen.
RF-10. Los usuarios registrados en el sistema podrán consumir servicios.
RF-11. Los usuarios registrados en el sistema podrán satisfacer las propuestas.
RF-12. Los usuarios registrados en el sistema podrán ofertar servicios.
RF-13. Los usuarios registrados en el sistema podrán eliminar sus servicios.
RF-14. Los usuarios registrados en el sistema podrán editar sus servicios.
RF-15. Los usuarios registrados en el sistema podrán rechazar la realización de consumiciones.
RF-16. Los usuarios podrán activar su cuenta de manera sencilla y rápida.
RF-17. Los usuarios registrados recibirán las horas cuando acepten el servicio.
RF-18. Los usuarios registrados pueden enviarse mensajes entre ellos.
RF-19. Los moderadores tendrán secciones propias.
RF-20. Los moderadores podrán dar bonificaciones.
RF-21. Los moderadores podrán consultar todos los datos de la base de datos.
RF-22. Los moderadores podrán imponer sanciones.
RF-23. Los moderadores podrán bloquear, desbloquear y cancelar usuarios.
RF-24. El sistema permitirá que los usuarios registrados tengan una cantidad de saldo negativo
RF-25. La interfaz mostrará en todo momento el saldo disponible y el nombre de usuario.
RF-26. La interfaz debe permitir acceder en todo momento a la cuenta del usuario.
RF-27. El sistema debe proporcionar un método para registrar un usuario de manera sencilla.

3.3. - Requisitos No Funcionales

RNF-1. Al sistema debe poder accederse a través de un ambiente Web y desde un dispositivo móvil.
RNF-2. Disponer de seguridad de autentificación de usuarios.
RNF-3. El sistema debe manejar la información de manera segura, es decir por medio de un método eficiente de encriptación.
RNF-4. El sistema debe tener métodos de bloqueo para los ataques de fuerza bruta.
RNF-5. El sistema debe ser altamente escalable, es decir debe poderse agregar nueva funcionalidad sin perder la calidad y el funcionamiento que ya se ha alcanzado.
RNF-6. El sistema debe ser confiable, asegurando un funcionamiento adecuado.
RNF-7. El sistema debe permitir su uso las 24 horas del día 7 días de la semana.
RNF-8. El sistema debe restringir que el usuario solo realice las opciones permitidas.
RNF-9. El sistema debe ser capaz de recuperarse fácilmente de cualquier error que pudiera

sucederse.

RNF-10. El sistema debe ser compatible con la mayoría de los navegadores Web.

RNF-11. El sistema comprobará cada 6 meses si el usuario sigue siendo miembro de la Uva.

RNF-12. El sistema generará una contraseña temporal para los usuarios no registrados que quieran proponer servicios.

RNF-13. Los usuarios deben pertenecer a la universidad.

RNF-14. El sistema debe soportar simultáneamente 5000 usuarios.

RNF-15. El sistema debe garantizar que los usuarios sean miembros de la universidad.

RNF-16. Los requisitos de hardware deben ser fácilmente alcanzables.

RNF-17. Los datos de los usuarios serán independientes de los datos que dispone la universidad.

RNF-18. Los servidores en los que correrá la aplicación serán independientes de los servidores de datos de la universidad.

RNF-19. La aplicación móvil debe ejecutarse sin problemas sobre la API 17 de android.

RNF-20. No se permitirá la eliminación de registros de las tablas bajo ninguna circunstancia.

3.3.1. - Requisitos de información

RINF-1. De los usuarios que soliciten una contraseña temporal se debe guardar su correo electrónico y la contraseña que se genera.

RINF-2. De los usuarios que se registren en el sistema debe guardarse su correo, contraseña, numero de intentos fallidos de acceso al sistema, el estado de la cuenta, el tipo de miembro dentro de la universidad, el tipo de usuario, su residencia, el campus donde estudia, el saldo actual y la fecha de último acceso.

RINF-3. De los servicios debe guardarse el título, el precio, la descripción, el tipo de servicio, el correo de la persona que oferta el servicio y la fecha de creación del servicio.

RINF-4. De las propuestas debe guardarse la descripción, el tipo de servicio que se cree que satisfará la propuesta, el correo de la persona que realiza la propuesta, la fecha en que se realizó la propuesta y el identificador del servicio que satisface la propuesta.

RINF-5. De las consumiciones debe guardarse la fecha en que se realizó la consumición, el identificador del servicio que se consume, el correo de la persona que ha solicitado el servicio, el correo de la persona que ofrece el servicio y el estado en el que se encuentra la consumición

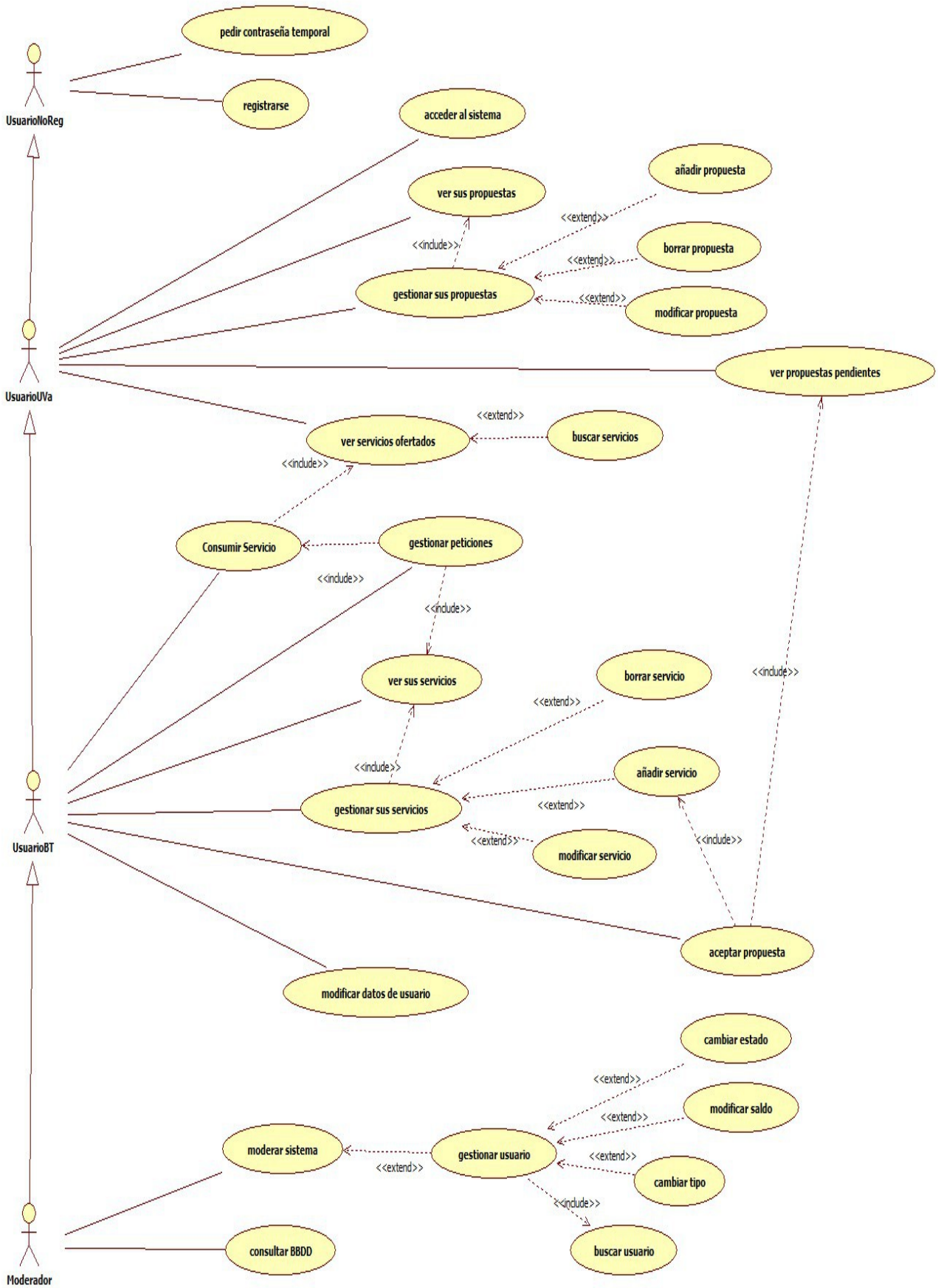
3.4. - Diagramas y especificaciones de casos de uso

3.4.1. - Diagrama de casos de uso

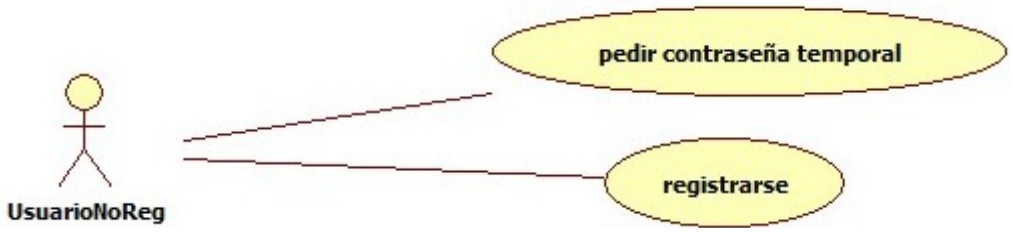
Este diagrama de casos de uso muestra la funcionalidad existente en la aplicación. Aparecen representados los tres tipos de actores que se contemplan, así como todas la acciones que pueden realizar y la interconexión que existen entre los diferentes casos.

Después se detallará en profundidad cada uno de los casos de uso representados en el diagrama.

Se puede apreciar, por ejemplo, como un *UsuarioUVa* utiliza el caso de uso *ver servicios* y éste a la vez es requerido por el caso de uso *consumir servicio.* Con este ejemplo se puede observar como los casos de uso están relacionados entre sí y la funcionalidad que proporciona la aplicación.


pedir contraseña temporal
registrarse
UsuarioNoReg
acceder al sistema
ver sus propuestas
añadir propuesta
borrar propuesta
gestionar sus propuestas
modificar propuesta
<<include>>
<<extend>>
<<extend>>
<<extend>>
UsuarioUVa
ver propuestas pendientes
ver servicios ofertados
buscar servicios
<<extend>>
<<include>>
Consumir Servicio
gestionar peticiones
<<include>>
<<include>>
ver sus servicios
borrar servicio
<<extend>>
<<include>>
añadir servicio
gestionar sus servicios
<<extend>>
<<extend>>
modificar servicio
<<include>>
<<include>>
UsuarioBT
aceptar propuesta
modificar datos de usuario
cambiar estado
<<extend>>
modificar saldo
<<extend>>
moderar sistema
gestionar usuario
<<extend>>
<<extend>>
cambiar tipo
<<include>>
buscar usuario
consultar BBDD
Moderador

3.4.2. - Especificación de Casos de Uso

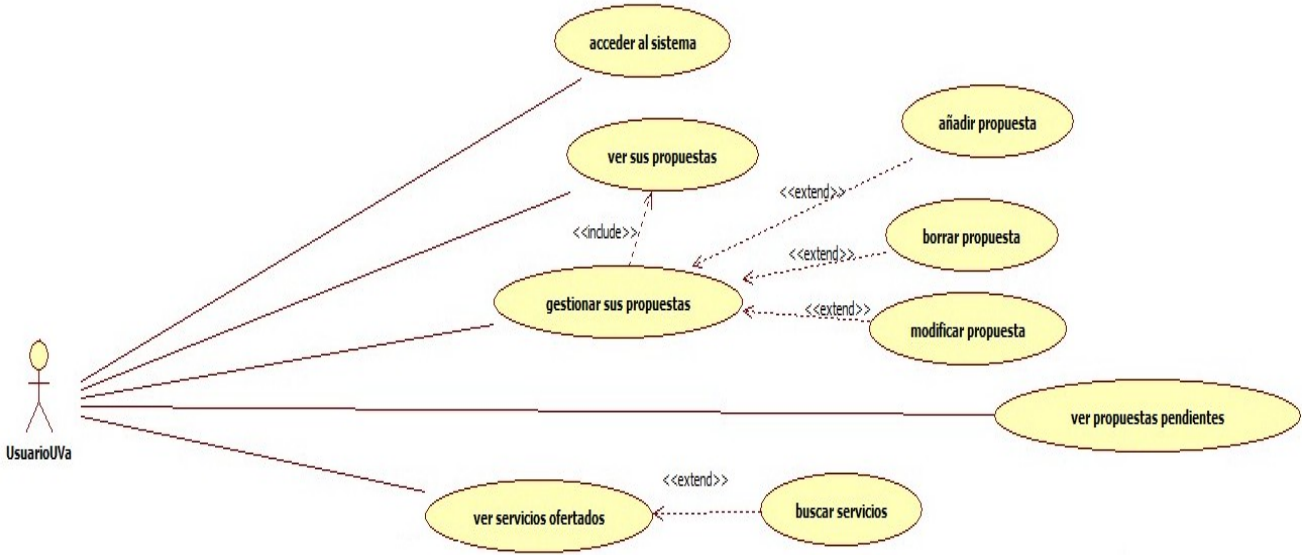


Identificador	**UC-01 Pedir contraseña temporal**	
Descripción	Permite que cualquier miembro de la Universidad pueda acceder a unas funciones limitadas del sistema por un periodo determinado.	
Precondiciones	- El usuario debe ser miembro de la Universidad. - El sistema debe estar activo y preparado para recibir los datos del usuario.	
Casos de uso relacionados		
Secuencia normal	Paso	Acción
	1	El usuario introduce su correo de la universidad en el campo del formulario habilitado a tal efecto.
	2	Hace click en el botón enviar.
	3	El sistema comprueba que el formato del mail es el adecuado.
	4	El sistema genera una contraseña temporal
	5	El sistema guarda el correo introducido y la contraseña temporal encriptada en la base de datos.
	6	El sistema manda un correo electrónico con la contraseña generada.
Postcondición	El usuario tiene acceso a la aplicación con la contraseña generada durante un periodo determinado.	
Excepciones	Paso	Acción
	3	Si el sistema detecta que el formato introducido no corresponde con un formato válido, coloreará la caja en color rojo.
Frecuencia	Una vez por periodo.	
Importancia	Vital	
Comentario		

Tabla 004

Identificador	**UC-02 Registrarse.**	
Descripción	Permite que cualquier miembro de la universidad pueda registrarse en el sistema para poder acceder a todas las funcionalidades de la aplicación (exceptuando la moderación)	
Precondiciones	- El usuario debe ser miembro de la universidad.	
Casos de uso relacionados		
Secuencia normal	Paso	Acción
	1	El usuario desde la página principal hace click en el enlace para registrarse.
	2	El sistema muestra el formulario que debe rellenar para poder registrarse.
	3	El usuario rellena el campo de correo.
	4	El usuario rellena los campos restantes.
	5	El usuario pulsa sobre el botón enviar.
	6	El sistema comprueba que se han introducido todos los datos correctamente.
	7	El sistema comprueba que no exista ya el usuario.
	8	El sistema almacena los datos en la base de datos.
Postcondición	El usuario se registra en la aplicación.	
Excepciones	Paso	Acción
	4	Si el sistema detecta que el formato introducido no corresponde con un formato válido, coloreará la caja de color rojo.
	6.1	Si el campo correo no tiene el formato correcto, la caja se colorea en rojo y se muestra un mensaje indicando que no es correcto.
	6.2	Si el campo residencia no se ha rellenado, la caja se colorea en color rojo y se muestra un mensaje y una alerta indicando que no es correcto.
	6.3	Si no se ha seleccionado un tipo de miembro, la caja se colorea en color rojo y se muestra un mensaje y una alerta indicando que no es correcto.
	6.4	Si no se ha seleccionado un campus, la caja se colorea en color rojo y se muestra un mensaje y una alerta indicando que no es correcto.
Frecuencia	Una vez por periodo.	
Importancia	Vital	
Comentario	-	

Tabla 005



Identificador	**UC-03 Acceder al sistema**	
Descripción	Permite que cualquier miembro de la universidad, que previamente se haya registrado o solicitado una contraseña temporal pueda acceder al sistema.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal.	
Casos de uso relacionados	- UC-01 Pedir contraseña temporal - UC-02 Registrarse.	
Secuencia normal	Paso	Acción
	1	El usuario introduce su correo de la universidad en el campo del formulario habilitado a tal efecto.
	2	El usuario introduce su contraseña de la aplicación en el campo del formulario habilitado a tal efecto.
	2	Hace click en el botón enviar.
	3	El sistema comprueba el estado de la cuenta.
	4	El sistema comprueba que el correo y la contraseña son correctos.
	5	El sistema muestra la pantalla principal de la aplicación.
	6	El sistema carga el menú dependiendo del tipo de usuario que se ha autenticado.
Postcondición	El usuario ha entrado en la aplicación.	
Excepciones	Paso	Acción
	4	No se puede conectar a la base de datos
	5	La cuenta está desactivada

	6.1	El sistema incrementa el número de intentos si la contraseña es errónea y muestra un mensaje con el número de intentos fallidos.
	6.1.1	Si el numero de intentos sobrepasa el establecido se bloquea la cuenta
	6.2	Si el usuario no existe, el sistema muestra un mensaje indicándolo.
Frecuencia	Una vez por sesión.	
Importancia	Vital	
Comentario	-	

Tabla 006

Identificador	**UC-04 Ver propuestas pendientes**	
Descripción	Permite a cualquier miembro de la universidad que pueda acceder al sistema ver las propuestas pendientes que hay en el sistema.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga las propuestas en la página.
	4	El usuario visualiza las propuestas pendientes.
Postcondición	El usuario puede ver las propuestas pendientes, ordenar por diferentes parámetros y pasar de página para poder ver todas.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar las propuestas y aparece vacía.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Las propuestas se cargan en AJAX por lo que el usuario tiene la opción de pasar página sin necesidad de refrescar ésta.	

Tabla 007

Identificador	**UC-05 Ver sus propuestas.**
Descripción	El usuario ve las propuestas que ha realizado.

Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de propuestas.	
Casos de uso relacionados	-	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de propuestas.
	2	El usuario entra en su perfil.
	3	El sistema carga el listado de propuestas.
	4	El usuario *cliquea* sobre el desplegable para poder visualizar sus propuestas.
Postcondición	El sistema muestra un listado con todas las propuestas que realizó.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de propuestas.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 008

Identificador	**UC-06 Gestionar propuestas.**	
Descripción	Permite que el usuario pueda gestionar todas sus propuestas. Añadir propuestas nuevas, modificar las propuestas ya añadidas y borrar propuestas.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema. - UC-05 Ver sus propuestas	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga las propuestas en la página.
	4	El usuario visualiza todas sus propuestas y las diferentes opciones
Postcondición	El usuario puede elegir la acción que quiere realizar.	
Excepciones	Paso	Acción
	2	El sistema no puede cargar la sección de propuestas.

	3	El sistema no puede cargar las propuestas y la lista aparece vacía.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 009

Identificador	**UC-07 Añadir propuesta.**	
Descripción	Permite que el usuario pueda añadir propuestas.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de propuestas.	
Casos de uso relacionados	- UC-05 Gestionar propuestas	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga el formulario para añadir propuestas.
	4	El usuario rellena los campos necesarios.
	5	El usuario pulsa en el botón de enviar datos.
	6	El sistema comprueba que los datos introducidos cumple las reglas.
	7	El sistema escribe en la base de datos la propuesta.
Postcondición	La propuesta se guarda en la base de datos	
Excepciones	Paso	Acción
	2	El sistema no puede cargar la sección de propuestas.
	7	No se puede conectar a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 010

Identificador	**UC-08 Borrar propuesta.**
Descripción	Permite que el usuario pueda borrar propuestas.
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de propuestas.

Casos de uso relacionados	- UC-05 Gestionar propuestas - UC-06 Ver sus Propuestas	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga el listado de propuestas.
	4	El usuario cliquea sobre el desplegable para poder visualizar sus propuestas.
	5	El usuario cliquea sobre el icono de borrado para eliminar una propuesta.
	6	El sistema pide confirmación para el borrado de la propuesta seleccionada.
	7	El usuario acepta el borrado.
	8	El sistema borra la propuesta de la base de datos.
	9	El sistema carga de nuevo el listado de las propuestas.
Postcondición	La propuesta seleccionada no existe en la base de datos.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de propuestas.
	7	El usuario no acepta el borrado
	8	Falla la conexión con la base de datos
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	La eliminación se realiza con AJAX, por lo que se observa el cambio al momento, sin necesidad de recargar la página completa.	

Tabla 011

Identificador	**UC-09 Modificar propuesta.**	
Descripción	Permite que el usuario pueda cambiar las características de la propuesta.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de propuestas.	
Casos de uso relacionados	- UC-05 Gestionar propuestas - UC-06 Ver sus Propuestas	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga el listado de propuestas.
	4	El usuario cliquea sobre el desplegable para poder visualizar

		sus propuestas.
	5	El usuario cliquea sobre el icono de edición para modificar una propuesta.
	6	El usuario cambia el tipo de propuesta.
	7	El sistema actualiza la base de datos con el cambio
	8	El sistema muestra el tipo de propuesta cambiado.
	9	El usuario cambia la descripción de la propuesta.
	10	El sistema actualiza la base de datos con el cambio
	11	El sistema muestra la descripción de la propuesta cambiada.
Postcondición	La propuesta se queda modificada en la base de datos.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de propuestas.
	7	No se puede acceder a la base de datos.
	8	El sistema no puede cargar la sección de propuestas.
	10	No se puede acceder a la base de datos.
	11	El sistema no puede cargar la sección de propuestas.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Las modificaciones se realizan con AJAX, por lo que se observa el cambio al momento, sin necesidad de recargar la página completa.	

Tabla 012

Identificador	**UC-10 Ver servicios ofertados**	
Descripción	Permite a cualquier miembro de la universidad que pueda acceder al sistema ver los servicios que hay en el sistema.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El sistema carga los servicios en la página principal.
	3	El usuario entra en la página principal.
	3	El usuario entra en la página de servicios
	4	El sistema carga los servicios en la página.
	5	El usuario visualiza los servicios.
Postcondición	El usuario puede ver los servicios, ordenar por diferentes parámetros y	

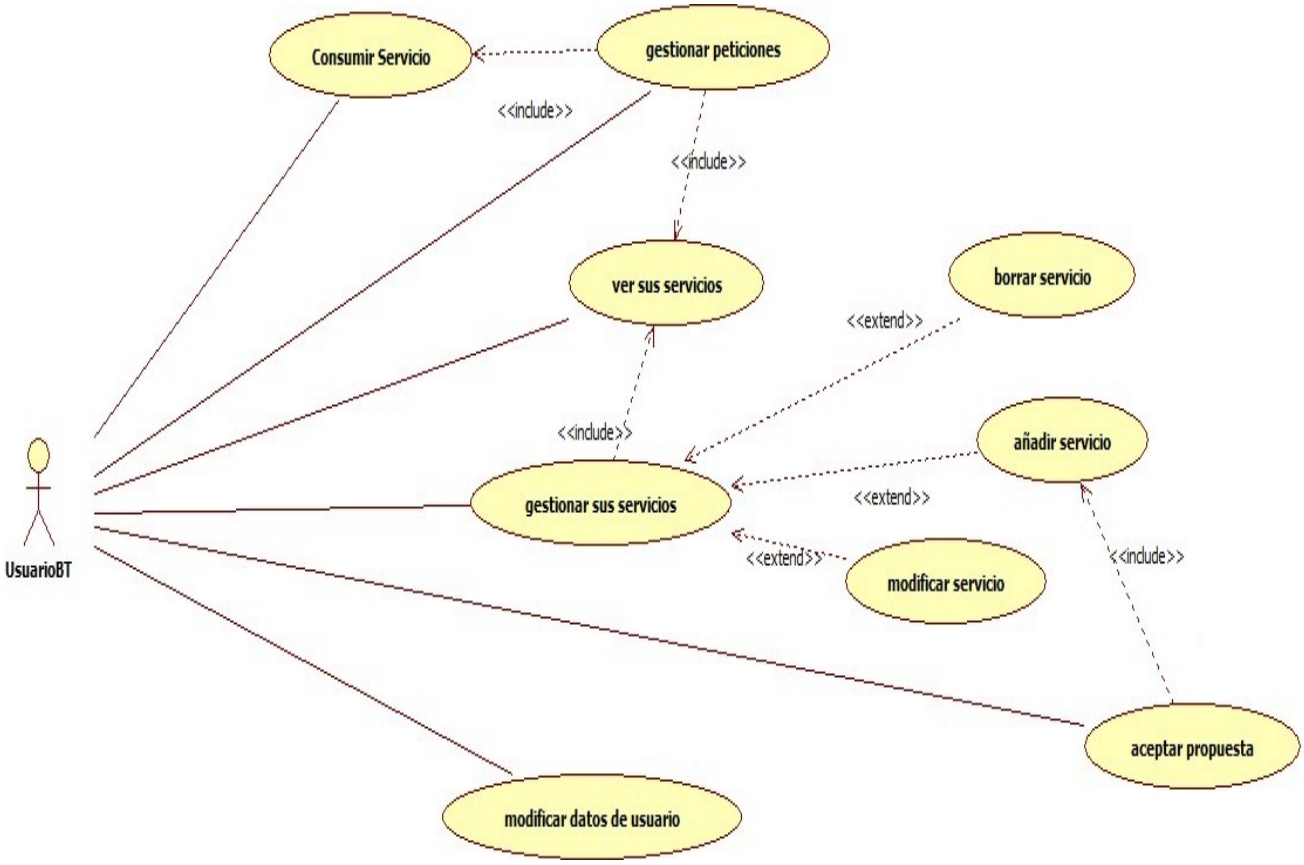
	pasar de página para poder ver todos.	
Excepciones	Paso	Acción
	2	El sistema no puede cargar los servicios y aparece vacía.
	6	El sistema no puede cargar los servicios y aparece vacía.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Las propuestas se cargan en AJAX por lo que el usuario tiene la opción de pasar página sin necesidad de refrescar ésta.	

Tabla 013

Identificador	**UC-11 Buscar servicios**	
Descripción	Permite a cualquier miembro de la universidad que pueda acceder al sistema, buscar entre los servicios que se ofertan en el sistema.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema o haber solicitado una contraseña temporal. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema. - UC-10 Ver servicios ofertados	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El sistema carga el formulario de búsqueda.
	3	El usuario rellena el campo titulo.
	4	El usuario pulsa el botón de buscar.
	5	El sistema valida el titulo introducido.
	6	El sistema busca los servicios.
	7	El sistema muestra los servicios.
	3	El usuario rellena el campo descripción.
	4	El usuario pulsa el botón de buscar.
	5	El sistema valida la descripción introducida.
	6	El sistema busca los servicios.
	7	El sistema muestra los servicios.
	3	El usuario selecciona el tipo de servicio.
	4	El usuario pulsa el botón de buscar.
	5	El sistema busca los servicios.
	6	El sistema muestra los servicios.
	3	El usuario selecciona el precio.
	4	El usuario pulsa el botón de buscar.
	5	El sistema busca los servicios.

	6	El sistema muestra los servicios.
Postcondición	El sistema muestra los servicios filtrados por la condición, o condiciones de búsqueda que el usuario desea.	
Excepciones	Paso	Acción
	6	El sistema no puede cargar los servicios y aparece vacía.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 015



Identificador	**UC-12 Aceptar propuesta**	
Descripción	Permite que un usuario registrado, al ver las propuestas pendientes, genere un servicio que satisfaga esa propuesta.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema. - UC-02 Registrarse	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	2	El usuario entra en la página de propuestas.
	3	El sistema carga las propuestas.
	4	El usuario visualiza las propuestas pendientes.
	5	El usuario selecciona la propuesta que quiere satisfacer.
	6	El sistema carga la página para añadir un nuevo servicio.
	7	El usuario rellena los datos del servicio que satisfacen la propuesta seleccionada.
	8	El sistema almacena el nuevo servicio, y crea una

		consumición con la propuesta.
Postcondición	La propuesta se marca como aceptada. Se crea un nuevo servicio. Se crea una consumición.	
Excepciones	Paso	Acción
	3	No se puede acceder a la base de datos.
	6	No se puede acceder a la base de datos.
	8	No se puede acceder a la base de datos.
Frecuencia	Frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 016

Identificador	**UC-13 Modificar datos de usuario.**	
Descripción	Permite que el usuario pueda cambiar su información.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la página de su perfil.	
Casos de uso relacionados	- UC-03 Acceder al sistema. - UC-02 Registrarse	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de su perfil.
	3	El usuario cliquea sobre el desplegable para mostrar el formulario de actualizar los datos
	4.1	El usuario rellena el campo de contraseña, para cambiar su contraseña.
	4.2	El usuario repite la contraseña introducida, para asegurarse que no se ha confundido.
	4	El usuario selecciona el campus que quiere modificar.
	4	El usuario modifica el campo residencia.
	5	El usuario introduce la contraseña actual para comprobar que es el usuario el que modifica los datos
	6	El usuario pulsa sobre el botón actualizar.
	7	El sistema valida los datos introducidos.
	8	El sistema actualiza la base de datos con el/los cambio/s
Postcondición	Los datos de usuario se modifican en la base de datos.	
Excepciones	Paso	Acción
	5	La contraseña actual introducida no es correcta.

	7	Los datos introducidos no son válidos.
	8	No se puede acceder a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Las modificaciones se realizan con AJAX, por lo que se observa el cambio al momento, sin necesidad de recargar la página completa.	

Tabla 017

Identificador	**UC-14 Ver sus servicios**	
Descripción	El usuario ve los servicios que ha ofertado.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de servicios o la de perfil	
Casos de uso relacionados	-	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de servicios.
	2	El usuario entra en su perfil.
	3	El sistema carga el listado de servicios.
	4	El usuario cliquea sobre el desplegable para poder visualizar sus servicios.
Postcondición	El sistema muestra un listado con todas los servicios que ha ofertado.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar el listado de servicios.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 018

Identificador	**UC-15 Gestionar sus servicios.**	
Descripción	Permite que el usuario pueda gestionar todas sus servicios. Añadir nuevos servicios, modificar servicios ya añadidos y borrar servicios.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema.	
Casos de uso relacionados	- UC-03 Acceder al sistema. - UC-14 Ver sus servicios	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.

	2	El usuario entra en la página de servicios.
	3	El sistema carga los servicios que el usuario a ofertado en la página.
	4	El usuario visualiza todos sus servicios y las diferentes opciones
Postcondición	El usuario puede elegir la acción que quiere realizar.	
Excepciones	Paso	Acción
	2	El sistema no puede cargar la sección de servicios.
	3	El sistema no puede cargar los servicios y la lista aparece vacía.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 019

Identificador	**UC-16 Añadir servicio.**	
Descripción	Permite que el usuario pueda ofertar.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de servicios.	
Casos de uso relacionados	- UC-15 Gestionar sus servicios - UC-14 Ver sus servicios	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de servicios.
	3	El sistema carga el formulario para añadir servicios.
	4	El usuario rellena los campos necesarios.
	5	El usuario pulsa en el botón de enviar datos.
	6	El sistema comprueba que los datos introducidos cumplen las reglas.
	7	El sistema escribe en la base de datos el servicio.
Postcondición	La propuesta se guarda en la base de datos	
Excepciones	Paso	Acción
	2	El sistema no puede cargar la sección de servicio.
	7	No se puede conectar a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	-	

Tabla 020

Identificador	**UC-17 Borrar servicio.**	
Descripción	Permite que el usuario pueda borrar servicios ofertados.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de propuestas.	
Casos de uso relacionados	- UC-15 Gestionar sus servicios - UC-14 Ver sus servicios	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de servicios.
	3	El sistema carga el listado de servicios.
	4	El usuario cliquea sobre el desplegable para poder visualizar sus servicios.
	5	El usuario cliquea sobre el icono de borrado para eliminar una propuesta.
	6	El sistema pide confirmación para el borrado del servicio seleccionado.
	7	El usuario acepta el borrado.
	8	El sistema borra el servicio de la base de datos.
	9	El sistema carga de nuevo el listado de los servicios.
Postcondición	El servicio seleccionada no existe en la base de datos.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de servicios.
	7	El usuario no acepta el borrado
	8	Falla la conexión con la base de datos
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	La eliminación se realiza con AJAX, por lo que se observa el cambio al momento, sin necesidad de recargar la página completa.	

Tabla 021

Identificador	**UC-18 Modificar servicio.**
Descripción	Permite que el usuario pueda cambiar las características de los servicios que oferta.
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de servicios.

Casos de uso relacionados	- UC-15 Gestionar sus servicios - UC-14 Ver sus servicios	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página de servicios.
	3	El sistema carga el listado de servicios.
	4	El usuario cliquea sobre el desplegable para poder visualizar sus servicios.
	5	El usuario cliquea sobre el icono de edición para modificar un servicio.
	6	El usuario cambia el tipo de servicio.
	7	El sistema actualiza la base de datos con el cambio
	8	El sistema muestra el tipo de propuesta cambiado.
	9	El usuario cambia la descripción del servicio.
	10	El sistema actualiza la base de datos con el cambio
	11	El sistema muestra la descripción del servicio cambiado.
Postcondición	El servicio se modifica en la base de datos.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de servicios.
	7	No se puede acceder a la base de datos.
	8	El sistema no puede cargar la sección de servicios.
	10	No se puede acceder a la base de datos.
	11	El sistema no puede cargar la sección de servicios.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Las modificaciones se realizan con AJAX, por lo que se observa el cambio al momento, sin necesidad de recargar la página completa.	

Tabla 022

Identificador	**UC-19 Consumir servicio.**	
Descripción	Permite que un usuario registrado en el sistema pueda consumir los servicios que hay ofertados.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de servicios.	
Casos de uso relacionados	- UC-10 Ver servicios ofertados	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.

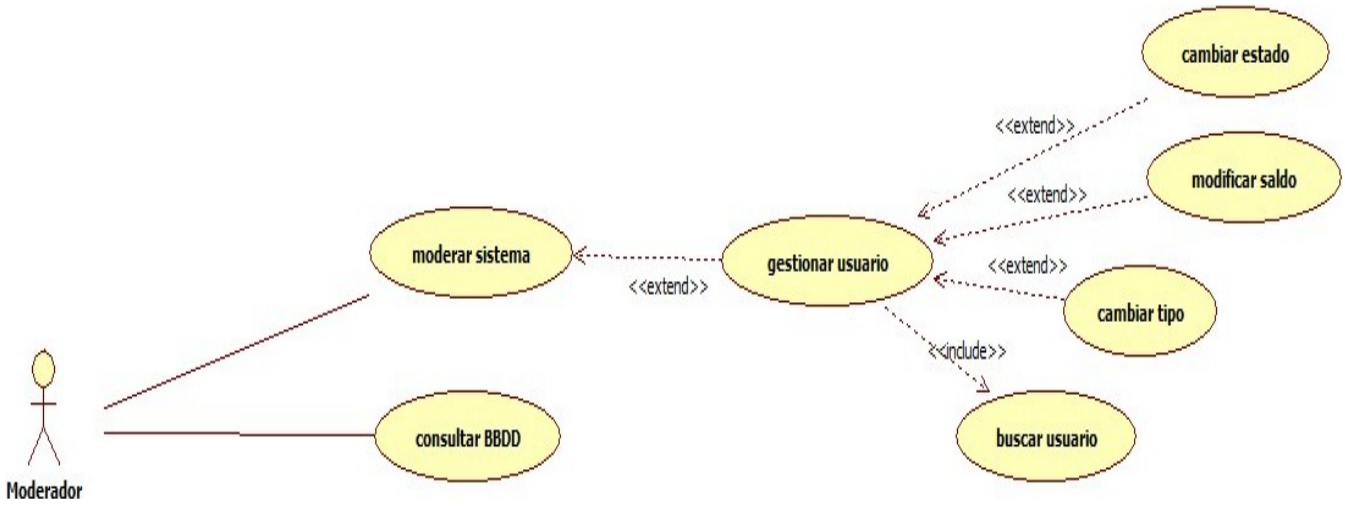
	2	El usuario entra en la página principal.
	2	El usuario entra en la página de servicios.
	3	El sistema carga los servicios en la página.
	4	El usuario visualiza los servicios ofertados.
	5	El usuario ordena los servicios por cualquiera de los parámetros que se ofrecen.
	5	El usuario restringe la búsqueda a un tipo de servicio.
	6	El sistema muestra los servicios con las características que el usuario desea, paginando los resultados para que sea mas cómodo.
	7	El usuario elige un servicio, y pulsa en pedir consumición.
	8	El sistema comprueba que el usuario tiene saldo suficiente para realizar la consumición.
	9	El sistema crea la consumición y la deja en estado pendiente, hasta que la petición la acepte el usuario que oferta el servicio.
	10	El usuario que oferta el servicio acepta realizar la consumición.
	11	El sistema cambia el estado de la consumición de pendiente a confirmar que se ha realizado la consumición.
	12	El usuario que realizó la petición confirma la realización del servicio.
	13	El sistema descuenta el saldo del usuario que pidió el servicio y se le añade al usuario que realizó el servicio.
Postcondición	Se crea una consumición y se modifica el saldo de los usuarios implicados.	
Excepciones	Paso	Acción
	3	El sistema no puede cargar la sección de servicios.
	8	El usuario no tiene saldo suficiente.
	10	El usuario no acepta realizar el servicio.
	12	El usuario no confirma la realización del servicio.
	14	El usuario no vota a la persona que realizó el servicio.
	15	El usuario no vota a la persona que pidió el servicio.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario		

Tabla 023

Identificador	**UC-20 Gestionar peticiones.**
Descripción	Permite que un usuario registrado al que le han realizado la petición de una consumición pueda aceptarla o rechazarla.

Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe acceder a la sección de servicios o a la de perfil.	
Casos de uso relacionados	- UC-14 Ver sus servicios - UC-19 Consumir servicio.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	2	El usuario entra en la página de servicios.
	3	El sistema carga los servicios en la página.
	4	El usuario visualiza los servicios que oferta.
	5	El usuario pulsa sobre una de las opciones, “aceptar” o “rechazar”
	6	El sistema marca la petición con la opción elegida.
Postcondición	Se crea una consumición y se modifica el saldo de los usuarios implicados.	
Excepciones	Paso	Acción
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario		

Tabla 024



Identificador	**UC-21 Moderar sistema.**	
Descripción	Permite a los usuarios con permiso de moderación realizar acciones sobre los usuarios, propuestas y servicios.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador que otro moderador previamente le habrá dado.	
Casos de uso relacionados	-	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
Postcondición	El sistema proporciona al moderador las herramientas para poder realizar la moderación.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario		

Tabla 025

Identificador	**UC-22 Gestionar usuario.**
Descripción	El sistema proporciona al moderador herramientas para gestionar a los usuarios.

Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
	5	El usuario elige gestionar un usuario
	6	El sistema carga las opciones que se pueden realizar sobre un usuario del sistema.
Postcondición	El sistema proporciona al moderador las herramientas para poder realizar la moderación sobre los usuarios.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario		

Tabla 026

Identificador	**UC-23 Cambiar estado.**	
Descripción	El sistema proporciona al moderador herramientas para cambiar el estado de los usuarios entre los estados establecidos.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema. -UC-22 Gestionar usuario. -UC-25 Buscar usuario.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.

	5	El moderador elige gestionar un usuario
	6	El sistema carga un buscador de usuarios para buscar el usuario sobre el que quiere cambiar el estado.
	7	El moderador rellena el campo de búsqueda y busca el usuario
	8	El sistema comprueba los datos introducidos
	9	El moderador selecciona el usuario.
	10	El sistema carga las opciones que se pueden realizar sobre un usuario del sistema.
	11	El moderador cambia el estado del usuario buscado.
	12	El sistema modifica el estado del usuario en la base de datos
Postcondición	El usuario seleccionado tiene un estado diferente.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
	7	No existe el usuario
	12	No se puede acceder a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Los estados pueden ser activada, desactivada, bloqueada o cancelada.	

Tabla 027

Identificador	**UC-24 Cambiar tipo.**	
Descripción	El sistema proporciona al moderador herramientas para cambiar el tipo de usuario de un usuario seleccionado.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema. -UC-22 Gestionar usuario. -UC-25 Buscar usuario.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
	5	El moderador elige gestionar un usuario
	6	El sistema carga un buscador de usuarios para buscar el usuario sobre el que quiere cambiar el estado.

	7	El moderador rellena el campo de búsqueda y busca el usuario
	8	El sistema comprueba los datos introducidos
	9	El moderador selecciona el usuario.
	10	El sistema carga las opciones que se pueden realizar sobre un usuario del sistema.
	11	El moderador cambia el tipo del usuario buscado.
	12	El sistema modifica el tipo del usuario en la base de datos
Postcondición	El usuario seleccionado es un tipo de usuario diferente.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
	7	No existe el usuario
	12	No se puede acceder a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	Los usuarios solamente pueden ser usuarios normales, o moderadores.	

Tabla 028

Identificador	**UC-25 Buscar usuario.**	
Descripción	El sistema proporciona al moderador la herramienta para poder buscar a un usuario concreto.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema. -UC-22 Gestionar usuario.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
	5	El moderador elige gestionar un usuario
	6	El sistema carga un formulario con un campo para introducir el correo del usuario.
	7	El sistema comprueba los datos introducidos
	8	El sistema busca en la base de datos los usuarios que contienen la cadena introducida
	9	El sistema carga los resultados en la página.

Postcondición	Se muestra un listado con el usuario que cumple la condición y todas las opciones que se pueden realizar sobre ellos.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
	8	No existe ningún usuario con la cadena introducida.
	8	No se puede acceder a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	La carga y la búsqueda se realizará a través de AJAX, sin refrescar la página.	

Tabla 029

Identificador	**UC-26 Modificar saldo.**	
Descripción	El sistema proporciona al moderador herramientas para modificar el saldo de un usuario, pudiendo aplicarle bonificaciones o penalizaciones.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema. -UC-22 Gestionar usuario. -UC-25 Buscar usuario.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
	5	El moderador elige gestionar un usuario
	6	El sistema carga un buscador de usuarios para buscar el usuario sobre el que quiere cambiar el estado.
	7	El moderador rellena el campo de búsqueda y busca el usuario
	8	El sistema comprueba los datos introducidos
	9	El moderador selecciona el usuario.
	10	El sistema carga las opciones que se pueden realizar sobre un usuario del sistema.
	11	El moderador rellena el campo habilitado e introduce la cantidad.
	12	El sistema modifica el saldo del usuario en la base de datos
Postcondición	El usuario seleccionado es un tipo de usuario diferente.	

Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
	7	No existe el usuario
	12	No se puede acceder a la base de datos.
Frecuencia	Muy frecuente.	
Importancia	Vital	
Comentario	El saldo se introducirá a través de un spinbox que sólo permita la introducción de números.	

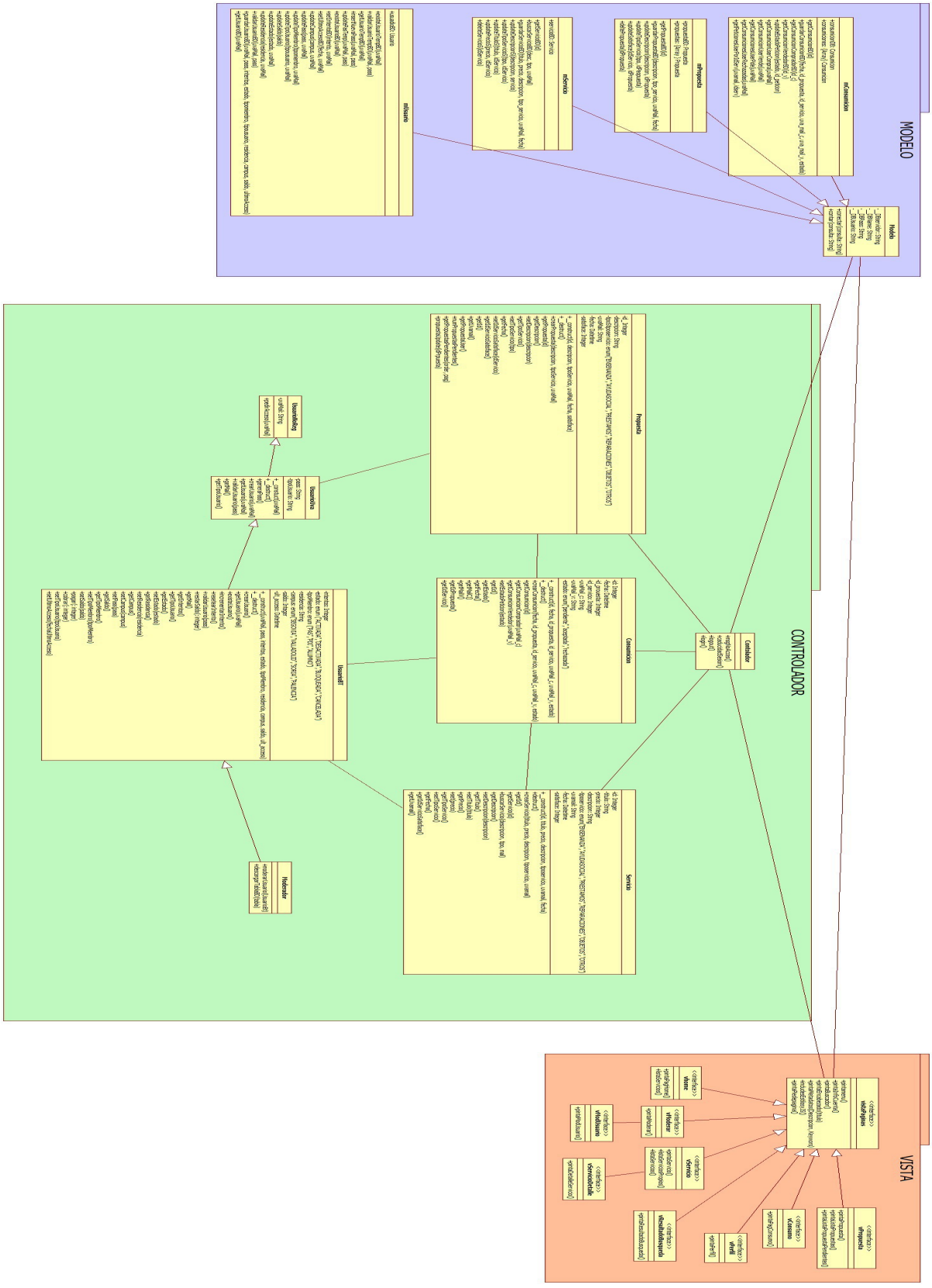
Tabla 030

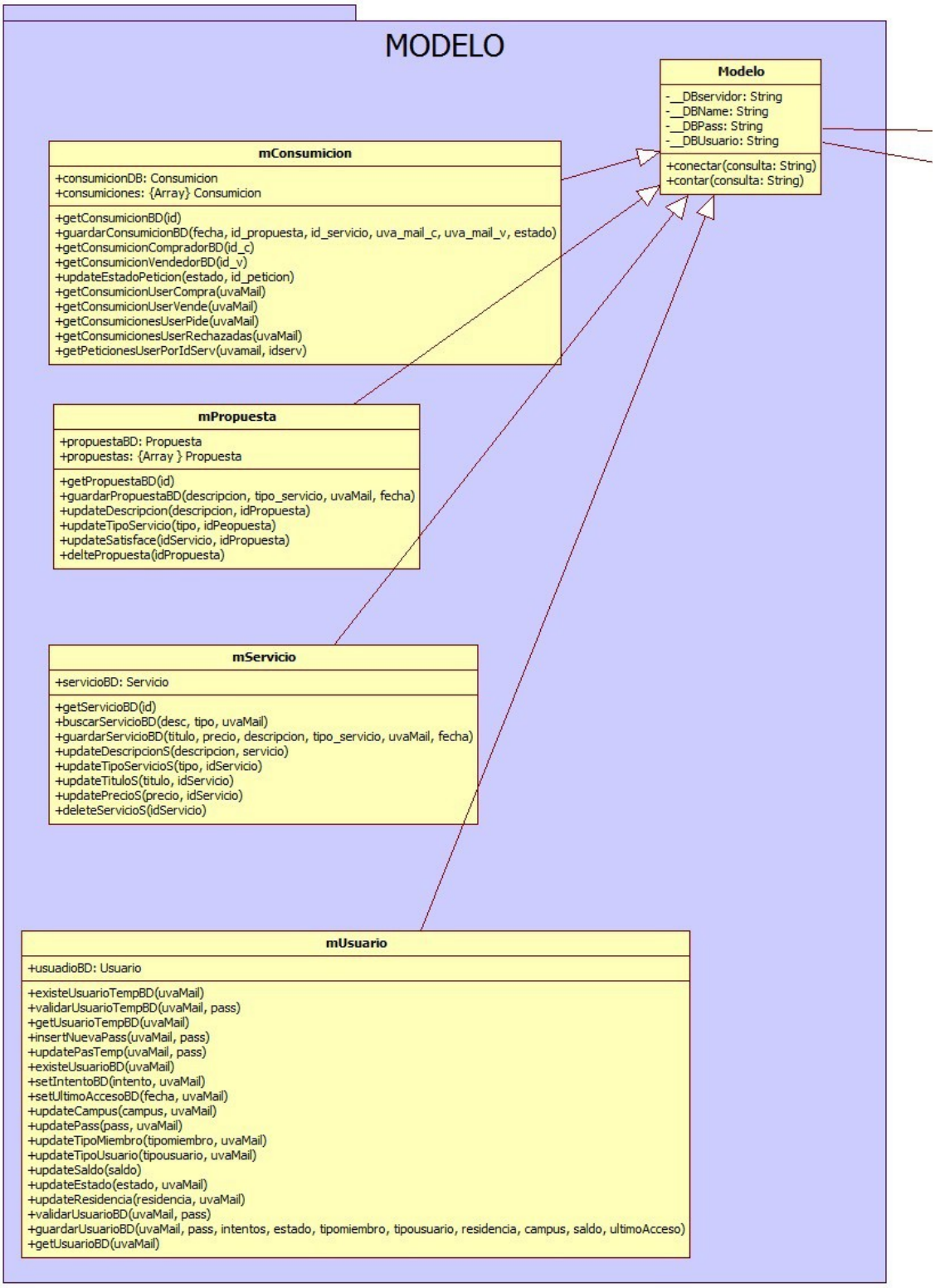
Identificador	**UC-27 Consultar BBDD.**	
Descripción	El sistema proporciona al moderador herramientas para poder consultar, sólo consultar, cualquier tabla de la base de datos.	
Precondiciones	- El usuario debe ser miembro de la universidad. - El usuario previamente debe haberse registrado en el sistema. - El usuario debe acceder al sistema. - El usuario debe tener permisos de moderador.	
Casos de uso relacionados	-UC-21 Moderar sistema.	
Secuencia normal	Paso	Acción
	1	El usuario accede al sistema.
	2	El usuario entra en la página principal.
	3	El sistema comprueba el tipo de usuario y carga el menú de moderación.
	4	El sistema carga la página de moderación con las opciones que puede realizar el moderador.
	5	El moderador elige consultar base de datos.
	6	El sistema carga un buscador en el que se elegirá la tabla de la base de datos y el rango de fechas a consultar.
	8	El moderador pulsa el botón enviar.
	9	El sistema devuelve una lista de los datos de la tabla introducida.
Postcondición	La propuesta es eliminada de la base de datos.	
Excepciones	Paso	Acción
	3	Fallo en la comprobación del tipo de usuario.
	9	No se puede acceder a la base de datos.
Frecuencia	Raramente.	
Importancia	Vital	
Comentario	-	

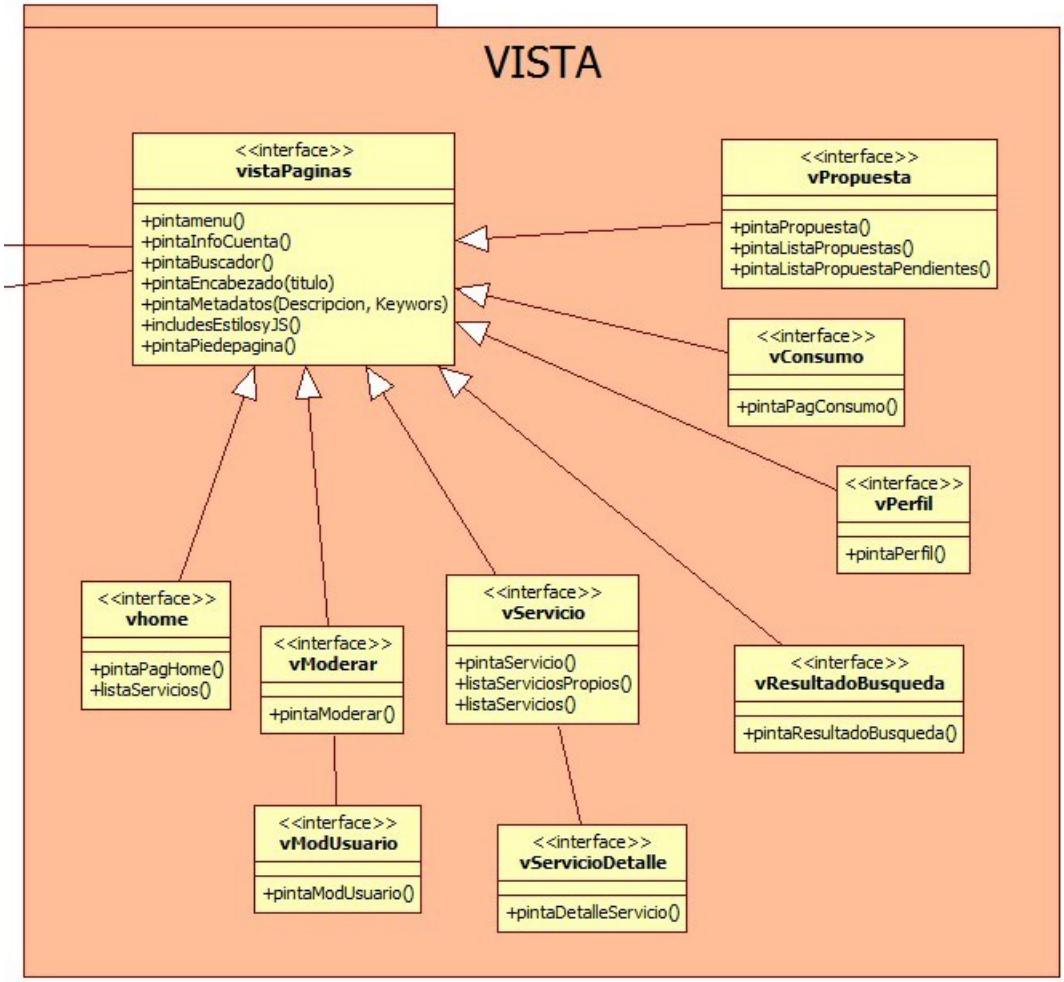
Tabla 031

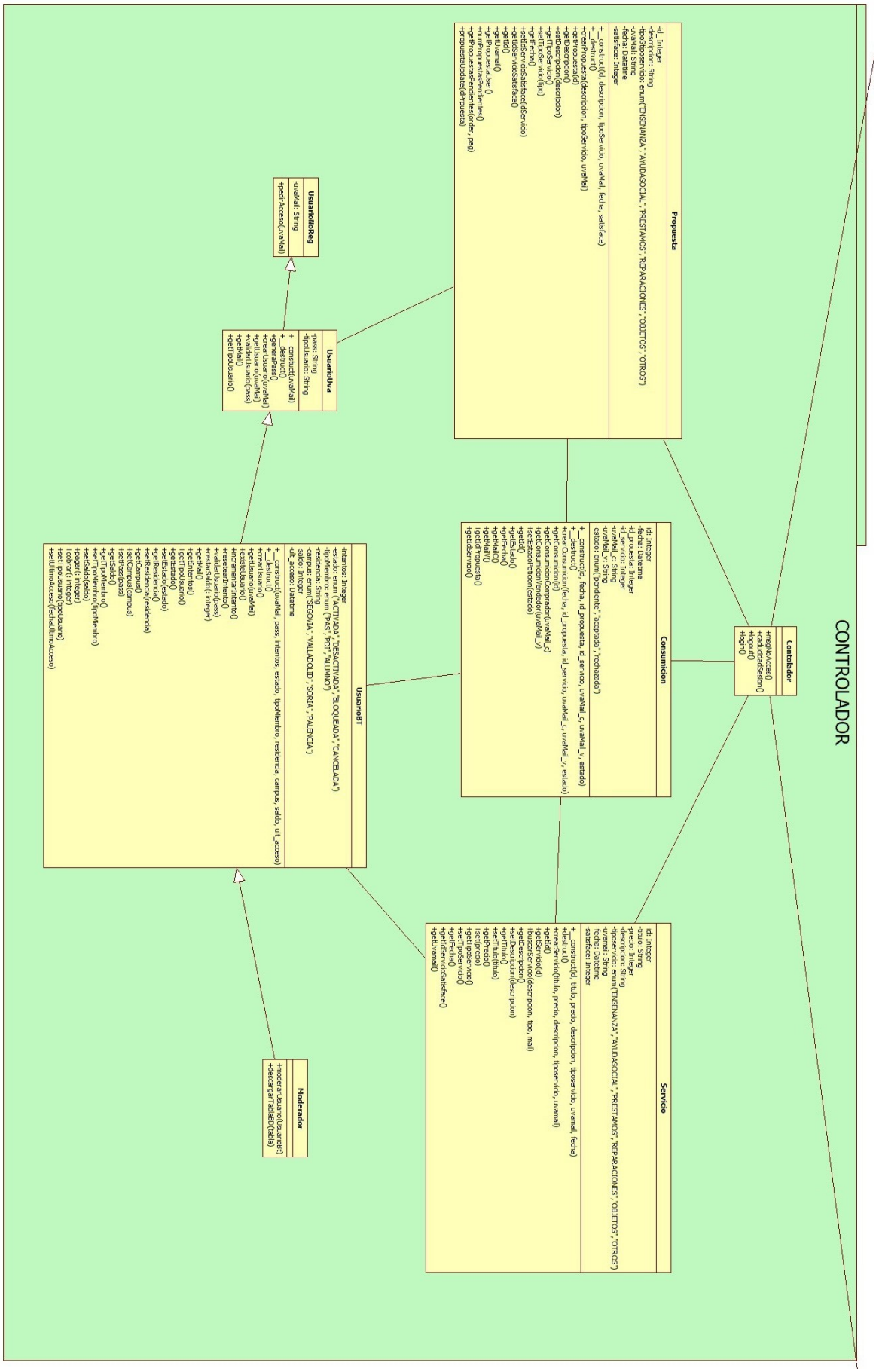
3.5. - Diagrama de clases de análisis

En el diagrama de clases, se puede observar la utilización de el patrón modelo-vista-controlador y los objetos que se crean para satisfacer las necesidades de la aplicación, así como los métodos que se utilizan y los parámetros que necesitan.




MODELO
Modelo
-__DBservidor: String
-__DBName: String
-__DBPass: String
-__DBUsuario: String
+conectar(consulta: String)
+contar(consulta: String)
mConsumicion
+consumicionDB: Consumicion
+consumiciones: {Array} Consumicion
+getConsumicionBD(id)
+guardarConsumicionBD(fecha, id_propuesta, id_servicio, uva_mail_c, uva_mail_v, estado)
+getConsumicionCompradorBD(id_c)
+getConsumicionVendedorBD(id_v)
+updateEstadoPeticion(estado, id_peticion)
+getConsumicionUserCompra(uvaMail)
+getConsumicionUserVende(uvaMail)
+getConsumicionesUserPide(uvaMail)
+getConsumicionesUserRechazadas(uvaMail)
+getPeticionesUserPorIdServ(uvamail, idserv)
mPropuesta
+propuestaBD: Propuesta
+propuestas: {Array } Propuesta
+getPropuestaBD(id)
+guardarPropuestaBD(descripcion, tipo_servicio, uvaMail, fecha)
+updateDescripcion(descripcion, idPropuesta)
+updateTipoServicio(tipo, idPeopuesta)
+updateSatisface(idServicio, idPropuesta)
+deltePropuesta(idPropuesta)
mServicio
+servicioBD: Servicio
+getServicioBD(id)
+buscarServicioBD(desc, tipo, uvaMail)
+guardarServicioBD(titulo, precio, descripcion, tipo_servicio, uvaMail, fecha)
+updateDescripcionS(descripcion, servicio)
+updateTipoServicioS(tipo, idServicio)
+updateTituloS(titulo, idServicio)
+updatePrecioS(precio, idServicio)
+deleteServicioS(idServicio)
mUsuario
+usuadioBD: Usuario
+existeUsuarioTempBD(uvaMail)
+validarUsuarioTempBD(uvaMail, pass)
+getUsuarioTempBD(uvaMail)
+insertNuevaPass(uvaMail, pass)
+updatePasTemp(uvaMail, pass)
+existeUsuarioBD(uvaMail)
+setIntentoBD(intento, uvaMail)
+setUltimoAccesoBD(fecha, uvaMail)
+updateCampus(campus, uvaMail)
+updatePass(pass, uvaMail)
+updateTipoMiembro(tipomiembro, uvaMail)
+updateTipoUsuario(tipousuario, uvaMail)
+updateSaldo(saldo)
+updateEstado(estado, uvaMail)
+updateResidencia(residencia, uvaMail)
+validarUsuarioBD(uvaMail, pass)
+guardarUsuarioBD(uvaMail, pass, intentos, estado, tipomiembro, tipousuario, residencia, campus, saldo, ultimoAcceso)
+getUsuarioBD(uvaMail)


VISTA
<<interface>>
vistaPaginas
+pintamenu()
+pintaInfoCuenta()
+pintaBuscador()
+pintaEncabezado(titulo)
+pintaMetadatos(Descripcion, Keywors)
+includesEstilosyJS()
+pintaPiedepagina()
<<interface>>
vPropuesta
+pintaPropuesta()
+pintaListaPropuestas()
+pintaListaPropuestaPendientes()
<<interface>>
vConsumo
+pintaPagConsumo()
<<interface>>
vPerfil
+pintaPerfil()
<<interface>>
vhome
+pintaPagHome()
+listaServicios()
<<interface>>
vModerar
+pintaModerar()
<<interface>>
vServicio
+pintaServicio()
+listaServiciosPropios()
+listaServicios()
<<interface>>
vResultadoBusqueda
+pintaResultadoBusqueda()
<<interface>>
vModUsuario
+pintaModUsuario()
<<interface>>
vServicioDetalle
+pintaDetalleServicio()

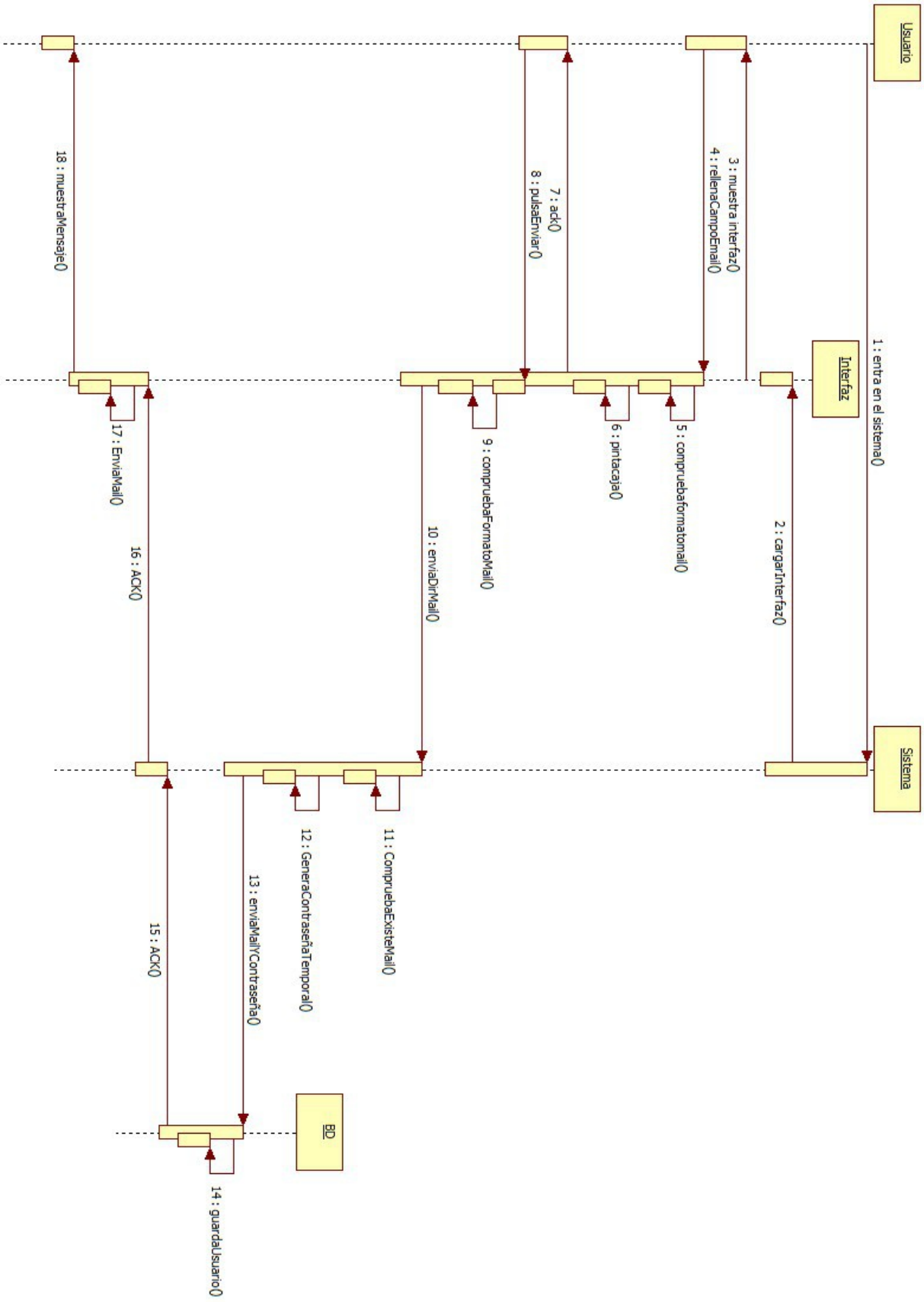

CONTROLADOR
Controlador
+msgNoAcceso()
+cerrarSesion()
+logout()
+login()
Propuesta
-id_: Integer
-descripcion: String
-tipoServicio: enum("ENSENANZA","AYUDASOCIAL","PRESTAMOS","REPARACIONES","OBJETOS","OTROS")
-uvaMail: String
-fecha: Datetime
-satisface: Integer
+__construct(id, descripcion, tipoServicio, uvaMail, fecha, satisface)
+__destruct()
+crearPropuesta(descripcion, tipoServicio, uvaMail)
+getPropuesta(id)
+getDescripcion()
+setDescripcion(descripcion)
+getTipoServicio()
+setTipoServicio(tipo)
+getFecha()
+setIdServicioSatisface(idServicio)
+getIdServicioSatisface()
+getId()
+getPropuestasUser()
+numPropuestasPendientes()
+getPropuestasPendientes(order, pag)
+getPropuestaUpdate(idPropuesta)
UsuarioNoReg
-uvaMail: String
+pedirAcceso(uvaMail)
UsuarioUva
-pass: String
-tipoUsuario: String
+__construct(uvaMail)
+__destruct()
+crearUsuario(uvaMail)
+getUsuario(uvaMail)
+validar(usuario/pass)
+getMail()
+getTipoUsuario()
Consumición
-id: Integer
-fecha: Datetime
-id_propuesta: Integer
-id_servicio: Integer
-uvaMail_c: String
-uvaMail_v: String
-estado: enum("pendiente","aceptado","rechazado")
+__construct(id, fecha, id_propuesta, id_servicio, uvaMail_c, uvaMail_v, estado)
+__destruct()
+crearConsumicion(fecha, id_propuesta, id_servicio, uvaMail_c, uvaMail_v, estado)
+getConsumicion(id)
+getConsumicionComprador(uvaMail_c)
+getConsumicionVendedor(uvaMail_v)
+setEstadoPeticion(estado)
+getId()
+getEstado()
+getFecha()
+getMail()
+getIdPropuesta()
+getIdServicio()
UsuarioBT
-intentos: Integer
-estado: enum ("ACTIVADA","DESACTIVADA","BLOQUEADA","CANCELADA")
-tipoMiembro: enum ("PAS","PDI","ALUMNO")
-residencia: String
-campus: enum("SEGOVIA","VALLADOLID","SORIA","PALENCIA")
-saldo: Integer
-ult_acceso: Datetime
+__construct(uvaMail, pass, intentos, estado, tipoMiembro, residencia, campus, saldo, ult_acceso)
+__destruct()
+getUsuario(uvaMail)
+crearUsuario()
+getIntentos()
+incrementarIntento()
+resetearIntento()
+validarUsuario(pass)
+restarSaldo(integer)
+getMail()
+getTipoUsuario()
+getEstado()
+getTipoUsuario()
+setEstado(estado)
+getResidencia()
+setResidencia()
+getCampus()
+setCampus(campus)
+getPass(pass)
+getSaldo()
+setSaldo()
+getTipoMiembro()
+setTipoMiembro()
+getUltimoAcceso()
+setUltimoAcceso()
+pagar(integer)
+cobrar(integer)
+setSaldo(saldo)
+setTipoUsuario(tipoUsuario)
+setUltimoAcceso(fechaUltimoAcceso)
Servicio
-id: Integer
-titulo: String
-precio: Integer
-descripcion: String
-tiposervicio: String
-uvamail: String
-fecha: Datetime
-satisface: Integer
+__construct(id, titulo, precio, descripcion, tiposervicio, uvamail, fecha)
+destruct()
+crearServicio(titulo, precio, descripcion, tiposervicio, uvamail)
+getId()
+buscarServicio(id)
+buscarServicio(descripcion, tipo, mail)
+getDescripcion()
+setDescripcion(descripcion)
+getTitulo()
+setTitulo(titulo)
+getPrecio()
+setPrecio()
+getTipoServicio()
+setTipoServicio()
+getFecha()
+getIdServicioSatisface()
+getUvamail()
Moderador
+moderarUsuario(UsuarioBt)
+descargarTablaBt(tabla)

3.6. - Diagrama de secuencia

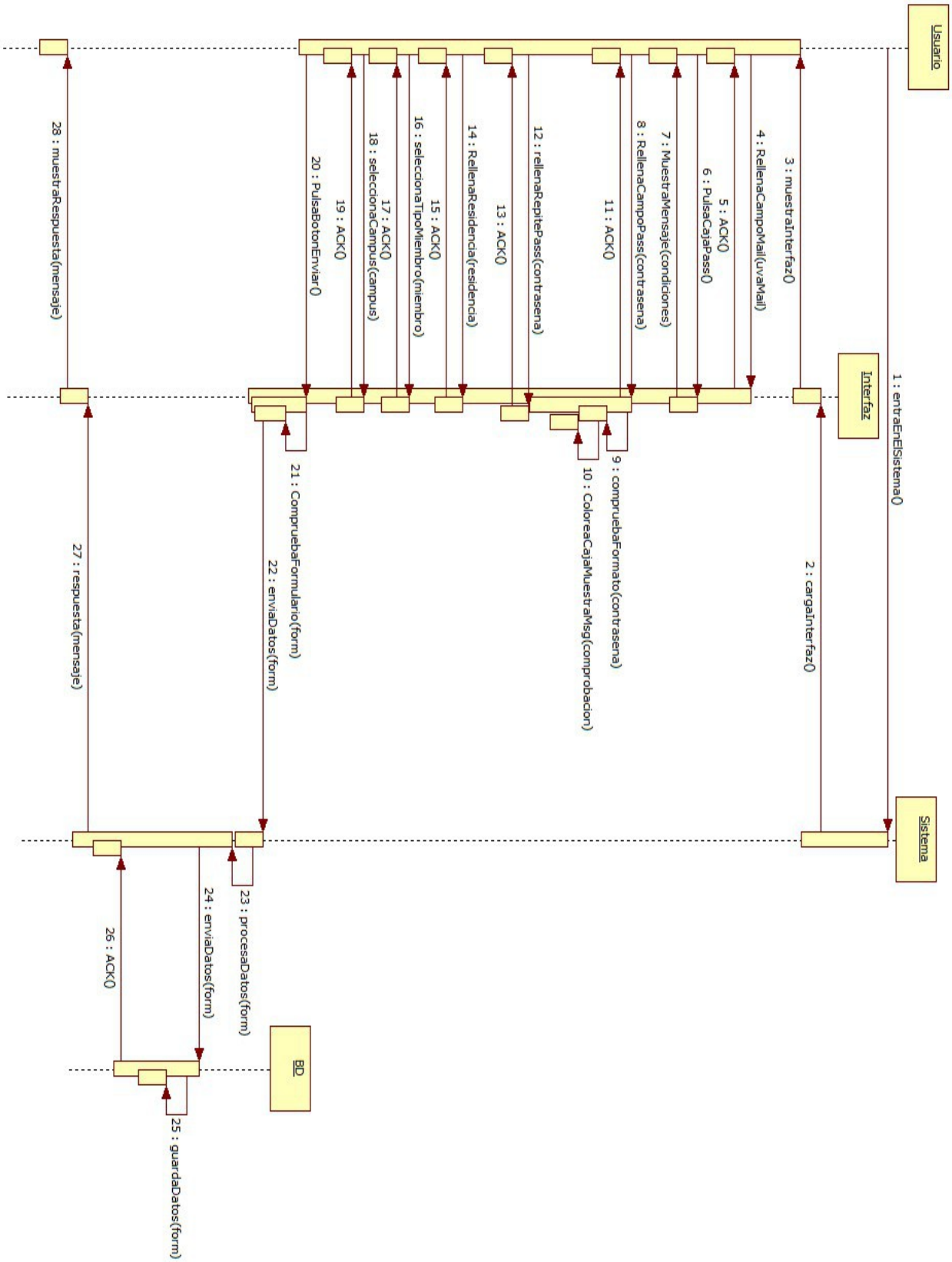
Los diagramas de secuencia, muestra la interacción que existe entre los objetos de la aplicación, según se va desarrollando en el tiempo, se modela un diagrama de secuencia para cada caso de uso existente en la aplicación.

Se plasmará en la memoria aquellos más significativos, ya que muchos de ellos son equivalentes en la mayoría de sus llamadas y flujo de ejecución.

3.6.1. - Pedir Contraseña

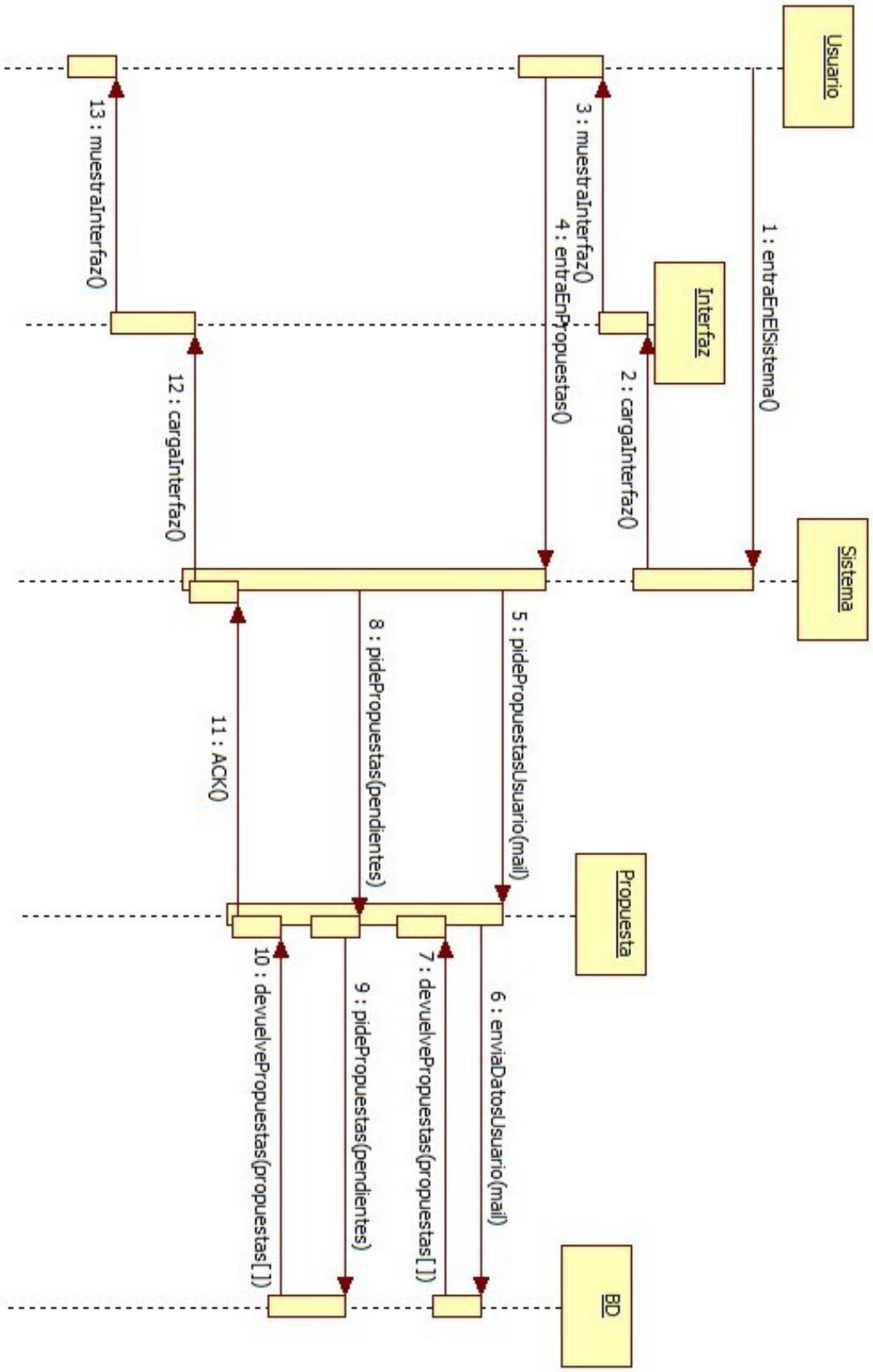


3.6.2. - Registrarse

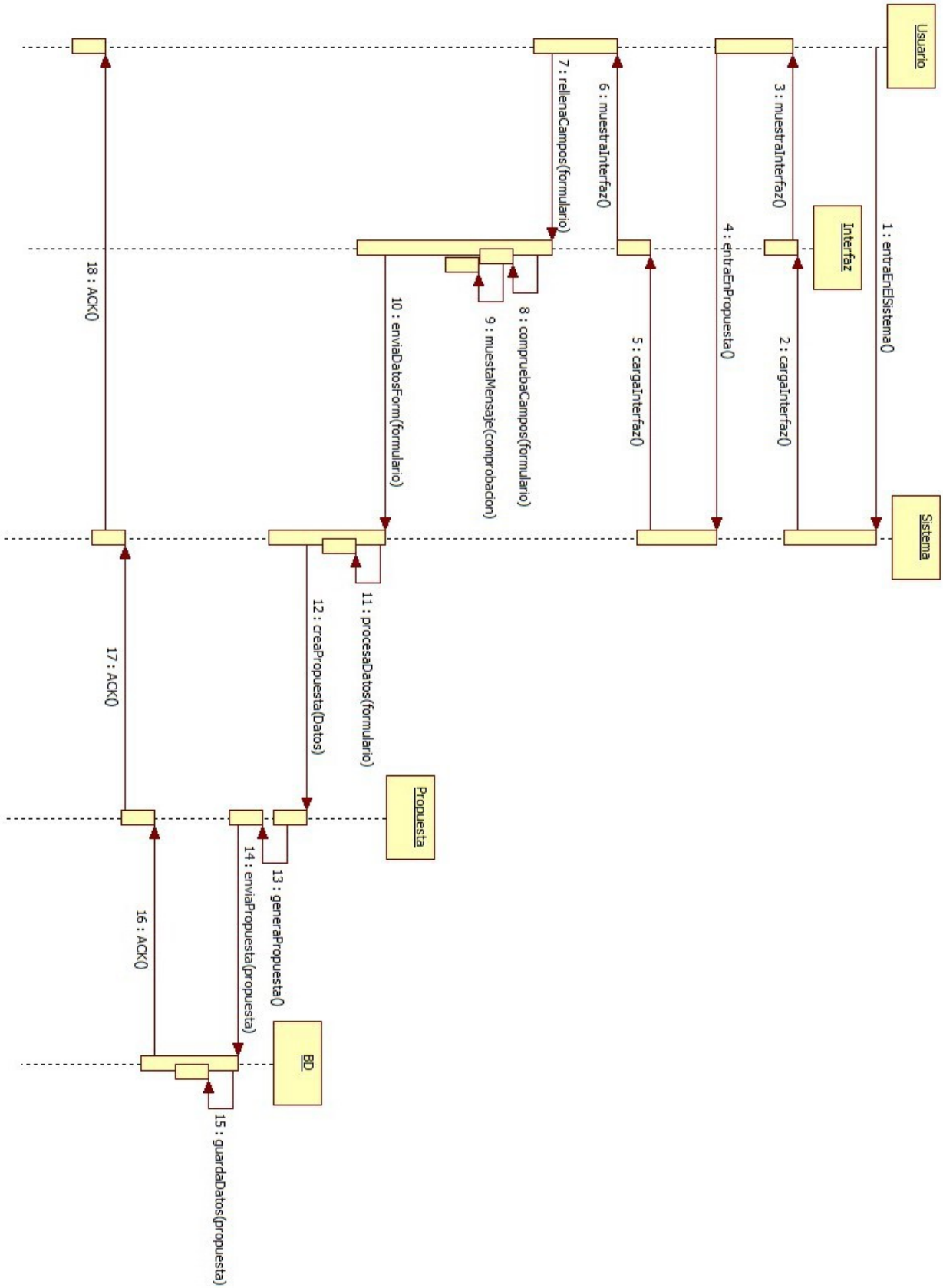


3.6.3. - Ver propuestas

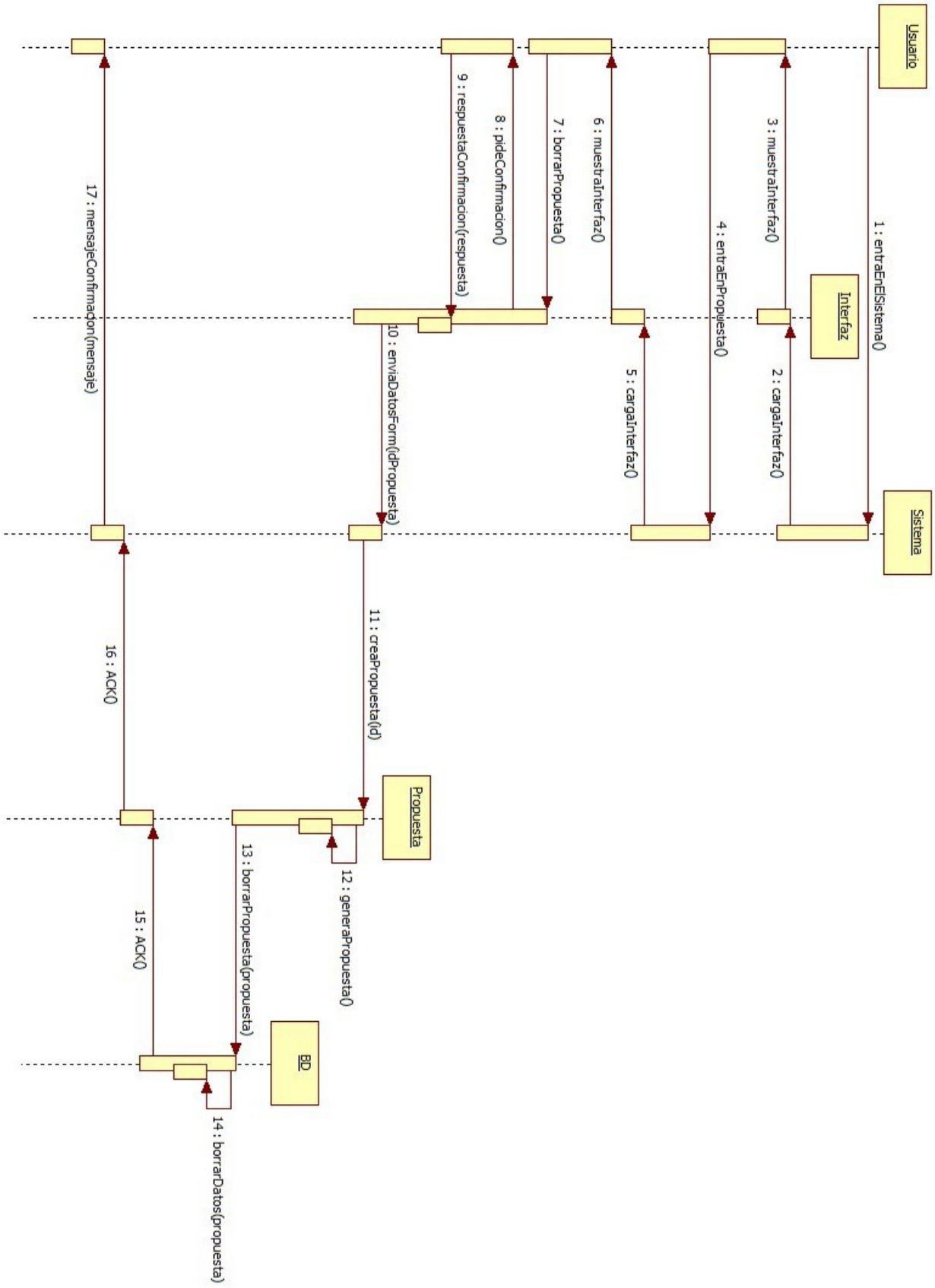
En este diagrama se representa los casos de uso de *ver sus propuestas* y de *ver propuestas pendientes*.



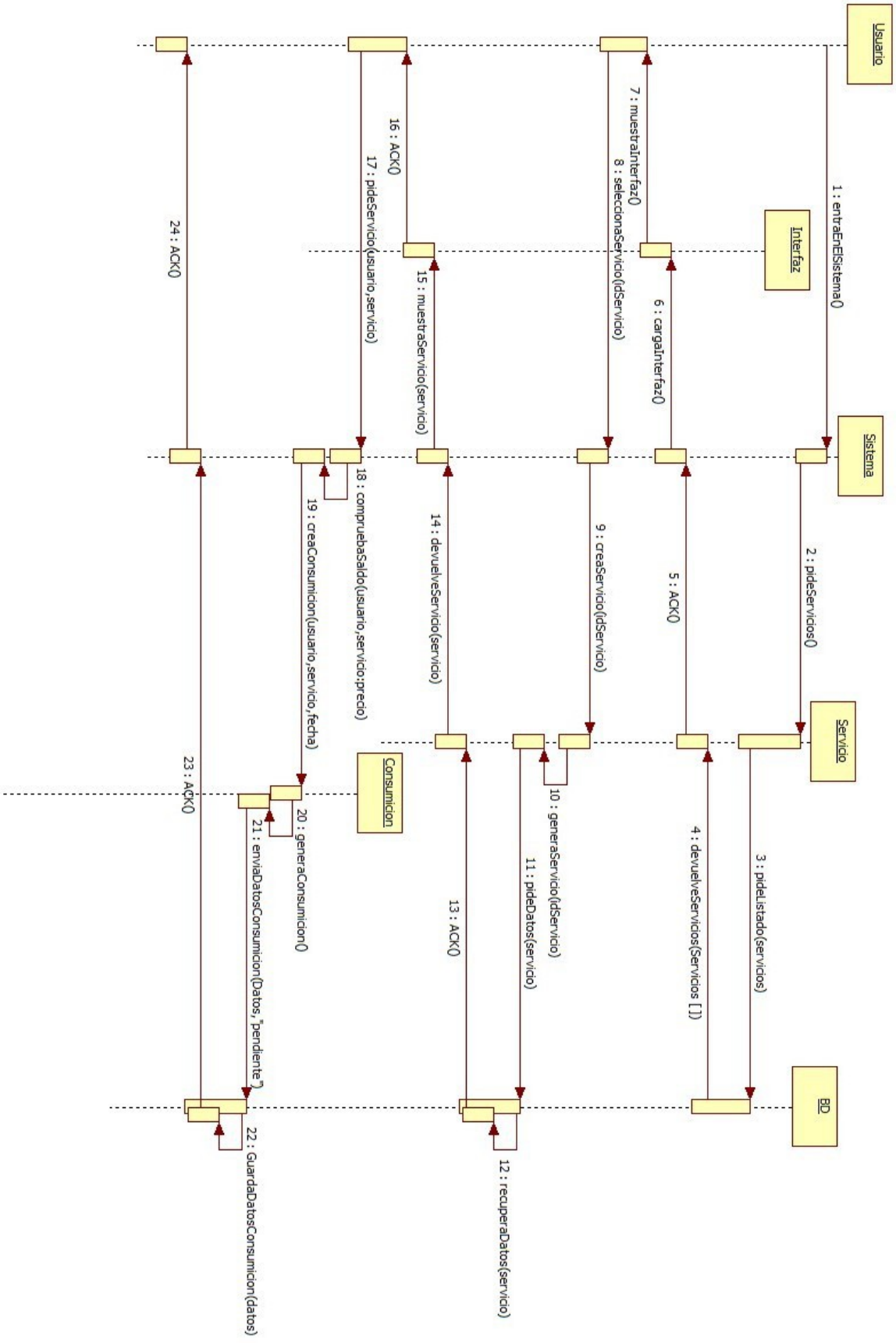
3.6.4. - Añadir propuestas / Modificar propuestas



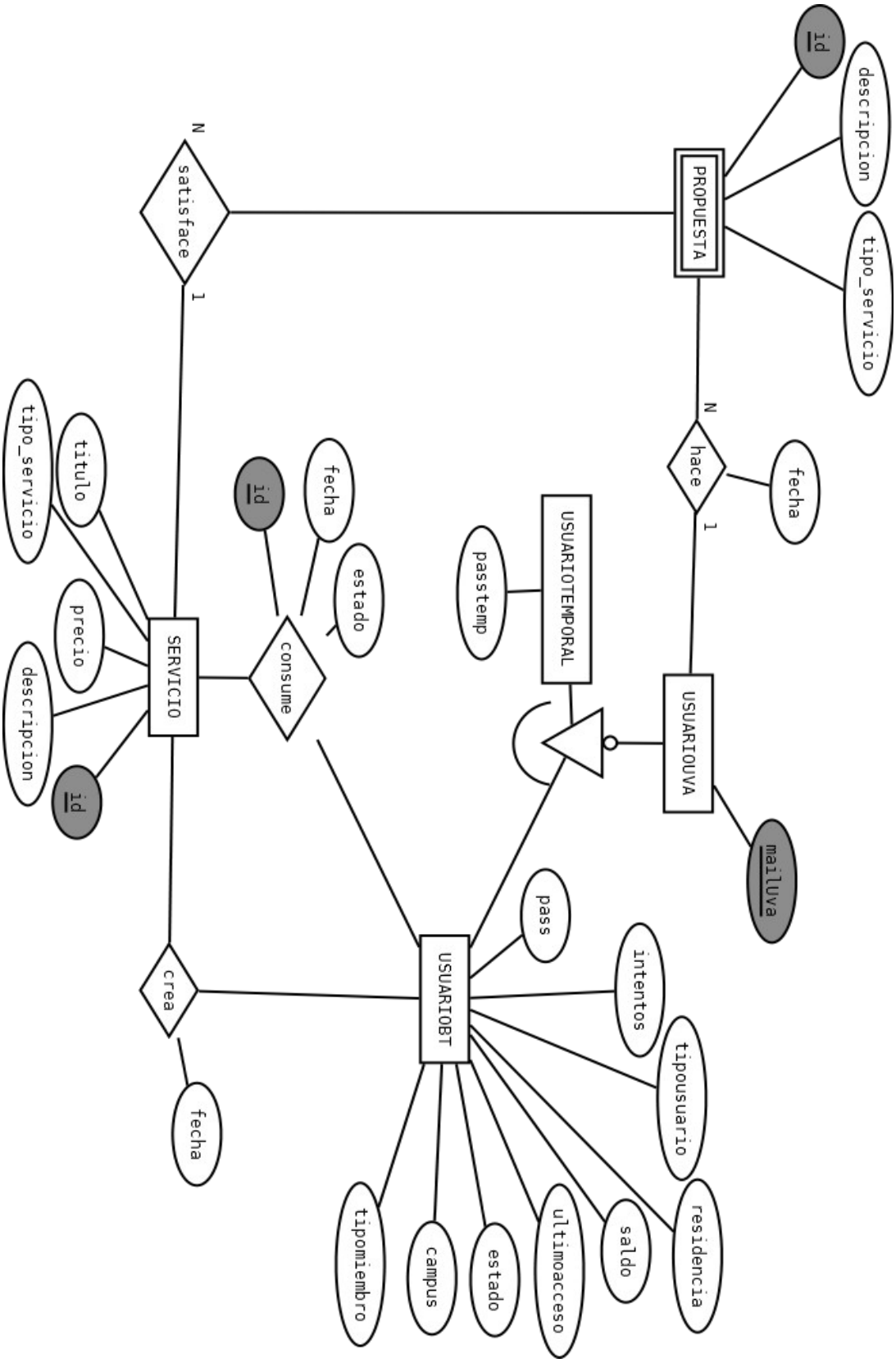
3.6.5. - Borrar propuestas



3.6.6. - Consumir servicio



3.7. - Diagrama Entidad-Relación



3.8. - Diccionario de datos

Entidad	Atributo	Valor	Descripción
usuario_temporal	mailuva	Cadena de caracteres	Email de la persona que solicito la contraseña.
	pass	Cadena de caracteres	Cadena de 128 caracteres (codificación SHA-512

Tabla 032

Entidad	Atributo	Valor	Descripción
consumición	id	Entero	Identificador autonumérico para cada consumición.
	fecha	Fecha y hora	Fecha y hora de cuando se realizó la consumición.
	id_servicio	Cadena de caracteres	Identificador del servicio al que hace referencia la consumición.
	id_uvamail_c	Cadena de caracteres	Email de la persona que ha solicitado la consumición.
	id_uvamail_v	Cadena de caracteres	Email de la persona que realizará el servicio.
	estado	Enumeración.	Estado en el que se encuentra la consumición.

Tabla 033

Entidad	Atributo	Valor	Descripción
propuesta	id	Entero	Identificador autonumérico para cada propuesta.
	descripción	Área de texto	Descripción de la propuesta.
	tipo_servicio	Enumeración	Tipo de la propuesta.
	uvamail	Cadena de caracteres	Email de la persona que hizo la propuesta.
	fecha	Fecha y hora	Fecha y hora de cuando se realizó la propuesta.
	idSatisface	Entero	Identificador del servicio que satisface la propuesta.

Tabla 034

Entidad	Atributo	Valor	Descripción
servicio	id	Entero	Identificador autonumérico para cada servicio.
	titulo	Cadena de caracteres	Título del servicio.
	precio	Flotante	Precio del servicio.
	descripción	Área de texto	Descripción del servicio.
	tipo_servicio	Enumeración	Tipo del servicio.
	uvamail	Cadena de caracteres	Email de la persona que dio de alta el servicio.
	fecha	Fecha y hora	Fecha y hora de cuando se dio de alta el servicio.

Tabla 035

Entidad	Atributo	Valor	Descripción
servicio	id	Entero	Identificador autonumérico para cada servicio.
	titulo	Cadena de caracteres	Título del servicio.
	precio	Flotante	Precio del servicio.
	descripción	Área de texto	Descripción del servicio.
	tipo_servicio	Enumeración	Tipo del servicio.
	uvamail	Cadena de caracteres	Email de la persona que dio de alta el servicio.
	fecha	Fecha y hora	Fecha y hora de cuando se dio de alta el servicio.

Tabla 036

4.- Seguridad

4.1. - Motivación

La seguridad es uno de los aspectos fundamentales de cualquier sistema de información, ya sea un sistema informático u otro sistema cualquiera. Por ello creo conveniente que tenga un apartado propio en el que se analicen los aspectos de seguridad tratados y cómo fueron afrontados, asumiendo siempre que no se puede garantizar la seguridad completamente pero si reducirla en la mayor medida posible.

Se debe de cuidar al máximo la privacidad e intentar que los datos que se envían y circulan por la red sean lo más difíciles de interceptar e interpretar por gente ajena.

4.2. - Contraseñas encriptadas en la base de datos.

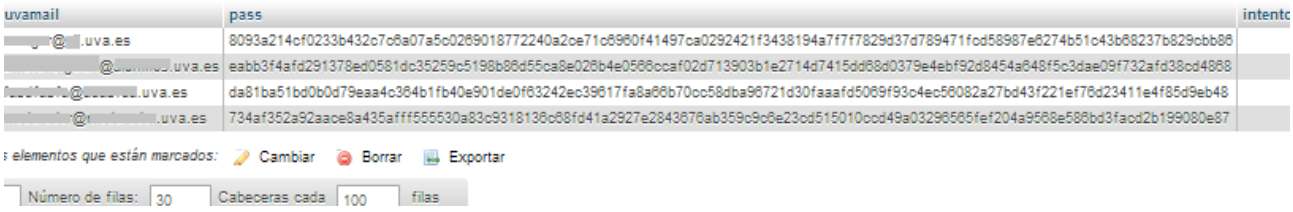
Las contraseñas son uno de los puntos mas sensibles de la seguridad, ya que éstas, no sólo pueden darte acceso al sistema, sino que muchas personas utilizan la misma contraseña para varias aplicaciones, por lo que ésta debe ser encriptada a la hora de almacenarla en la base de datos.

Para la encriptación de la contraseña se utilizará *SHA-2* , que es el algoritmo que más seguridad ofrece. Actualmente sabemos gracias a Hans Dobbertin , que el protocolo *MD5* presenta problemas de seguridad.

Todas las contraseñas utilizan ésta cadena para ser almacenadas.

```
$sha512=openssl_digest($pass, 'sha512'); // encriptar la contraseña para almacenarla.
```

De esta manera se almacena automáticamente encriptada y nunca se tiene la contraseña original.



4.3. - Caducidad de sesión

En ocasiones, el usuario se identifica en una aplicación y se va del lugar en el que está dejando la sesión abierta, haciendo que cualquier persona pueda utilizar nuestro usuario sin ninguna dificultad.

Para evitar esto, se ha creado una función a la que se pasa como parámetro el tiempo que se desea que una sesión se mantenga. Mientras no haya inactividad en el uso de la aplicación, la sesión se mantendrá, en caso contrario, el usuario deberá de identificarse de nuevo.

```
/*
	Función: caducidadSesion()
	Descripción:
	Si el tiempo de la sesión, es menor que el tiempo actual,
	borramos las variables de sesión e indicamos que debe loguearse de nuevo
	en caso contrario, se actualiza el tiempo de la sesión al actual
*/

function caducidadSesion(){
	$sg = 300;// $sg es el tiempo en segundos que queremos que este activa la sesion
	if (($_SESSION['timeSession'] + $sg) < time())
	{
		$_SESSION[u4s3r]='';
	}
	else
	{
		$_SESSION['timeSession']=time();
	}
}
```

4.4. - SQLInjection

Aunque en Android se pueden ejecutar consultas *SQL* en bruto, para evitar el *SQL Injection*, se usarán las funciones que esta arquitectura nos proporciona. Ya que vienen preparadas para evitar *SQL Injection* además de otros tipos de ataques.

4.5. - Acceso / Control de usuario a través del mail (Uva)

Para asegurar que las personas que acceden al sistema son personas de la Universidad de Valladolid, el registro se hará unicamente mediante el correo electrónico que la Universidad de Valladolid proporciona a todos los miembros; alumnos, profesores y PAS.

4.6. - Control de estado de cuentas

También en cualquier momento un moderador, podrá regenerar las contraseñas, mandando la nueva contraseña al correo proporcionado por los usuarios. Así se consigue tener la certeza de que las cuentas de las personas que utilizan la aplicación, son realmente personas físicas a las que se las puede atribuir cualquier acción que haya realizado su usuario.

4.7. - Generación de contraseñas

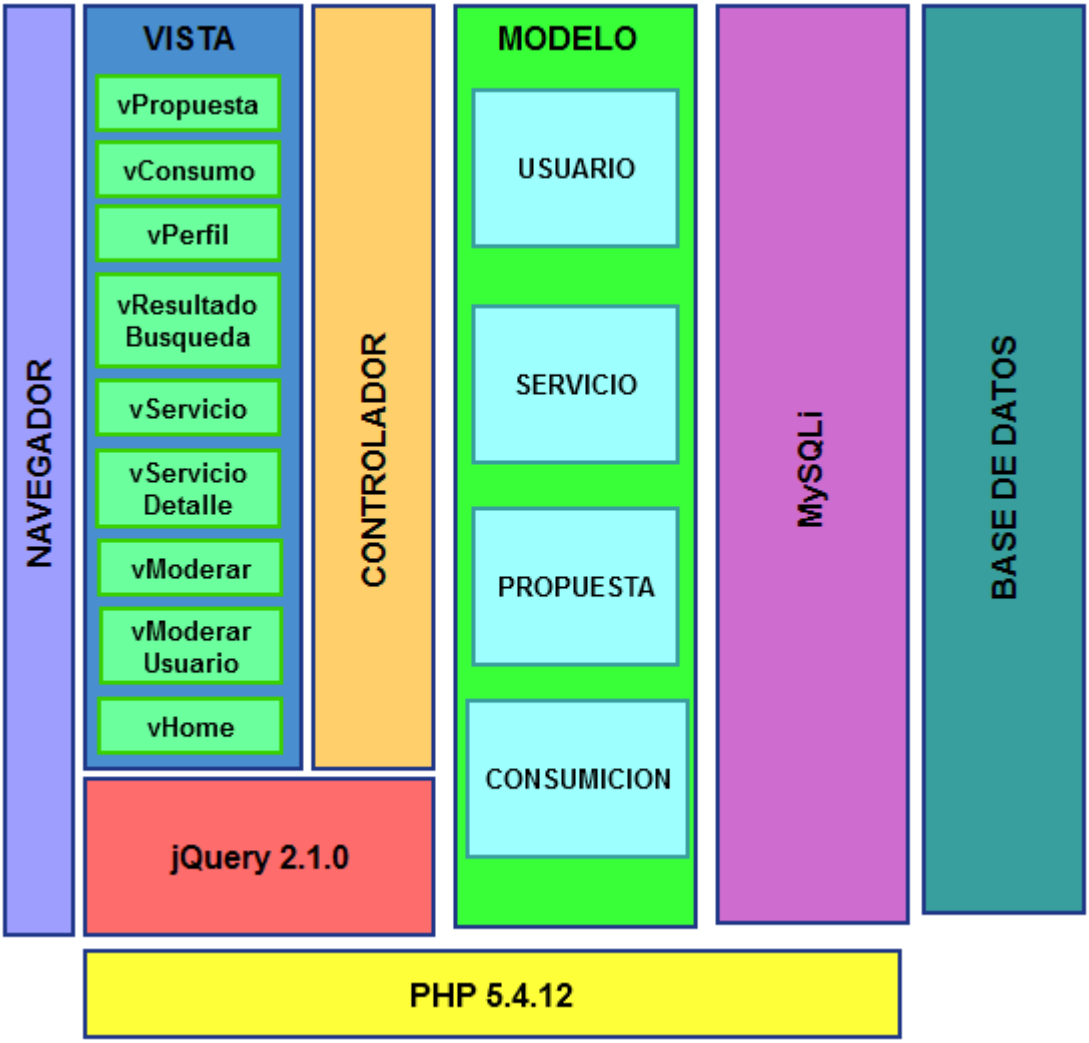
Se ha creado una función que permite generar contraseñas aleatoriamente, en función de la fecha y hora del servidor (evitando así intentos por parte del cliente modificando su fecha y hora) , un array de letras y una semilla aleatoria para aumentar la exclusividad.

```
function generaPass(){
		$fecha = new DateTime();
		$fecha=$fecha->getTimestamp();
```

```
                // semilla aleatoria que se suma al tiempo
                $semilla=rand();

                $resultado=$fecha + $semilla;

                $letras                                                                 =
array("A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V"
,"W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v
","x","y","z","1","2","3","4","5","6","7","8","9","0","_","$");

                //mezcla el array
                shuffle($letras);

                //Convertir el numero en array de numeros
                $num_array  = array_map('intval', str_split($resultado));

                $pass="";
                for ($i=0;$i<sizeof($num_array);$i++){
                        $num= 28 - $num_array[$i];
                        $pass.=$letras[$num];
                }
                return $pass;
        }
```

5. - Diseño

5.1. - Arquitectura lógica



La arquitectura lógica, se basa en un modelo de tres niveles, en la que se separa, la interfaz de usuario , que son las vistas, la lógica de negocio y la capa de persistencia donde se encuentran los datos.

Se usará el patrón arquitectónico Modelo-Vista-Controlador, permitiendo separar la funcionalidad. Realizando en la parte de la vista, únicamente la funcionalidad de recuperar los datos necesarios y mostrarlos de manera adecuada; en la parte del controlador la ejecución de las funciones y métodos propios para realizar todas las operaciones y la parte del modelo que se encarga de proporcionar los métodos y de hacer las consultas a la base de datos.

Se usa un modelo de tres niveles, dado que para la parte de la aplicación que se desea implementar, no es necesario un nivel de abstracción mayor. La separación en estos tres niveles separa las diferentes partes con su funcionalidad común, y no se precisa de la separación de las capas de la vista y la lógica de negocio.

Esto permite la reutilización de la lógica de negocio para diferentes sistemas consiguiendo que los cambios que se realicen en la lógica de negocio no afecten a los datos.

Los clientes son clientes ligeros, se conectan a través de su navegador a la interfaz web que ofrece el sistema favoreciendo su uso y actualización. Ya que una aplicación en la que se requiera de una instalación en los dispositivos sería mas complejo y menos funcional, dado que las actualizaciones las deberá realizar el usuario en su dispositivo.

Como se observa toda la lógica de negocio está creada bajo *PHP* 5.4.12. Las tres vistas implementadas en *PHP* que se conectan con el controlador que nos proporciona los servicios, y el modelo que maneja todos los datos referentes con los usuarios, los servicios, las propuestas y las consumiciones.

Disponemos de nueve vistas principales que el usuario consultará y utilizará para acceder e interactuar con el sistema para poder realizar todas las operaciones necesarias. A través de ella podrá darse de alta, consultar servicio, consumir, realizar peticiones y actualizaciones de los datos.

La librería *jQuery* nos permite utilizar el código HTML de manera mas sencilla , utilizando Javascript de manera fácil y rápida, pudiendo utilizar *AJAX* para cargar los resultados que devuelve el controlador en las vistas.

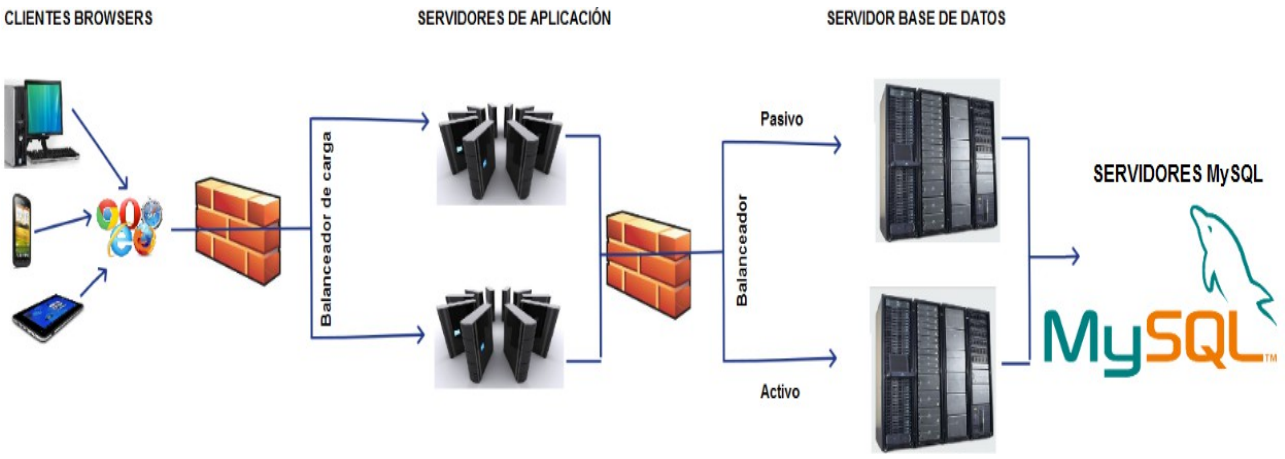
El controlador se encargará de las peticiones que realice el usuario contra nuestro sistema. En relación con los servicios nos permitirá recoger los servicios para que sea seleccionados, éste conectará con el modelo de servicio para realizar la consulta contra la base de datos y nos devolverá un array con los servicios que existen que será enviado a la capa de la vista de seleccionar servicios.
.

Disponemos de cuatro modelos, los correspondientes con el usuario, servicio, propuesta y consumición.

Por ejemplo en el modelo de servicios, nos da las funcionalidades que requieren de la base de datos, como son la de solicitar los servicios que se encuentran en la base de datos, asignar ese servicio a una persona etcétera.

El modelo se encarga de realizar las operaciones con la base de datos a través de el driver *MySQLi* que es el driver que permite conectarse a bases de datos *MySQL* a través de paginas *PHP*.

5.2. - Arquitectura física



La arquitectura física es una arquitectura de tres capas, en la que se diferencian los tres componentes principales.

Estimaremos, según nuestros datos, que la Universidad de Valladolid cuenta con alrededor de 33.000 miembros, entre alumnos, PAS y PDI; que serán los usuarios del sistema.

1. Usuarios: Estos clientes no contienen nada de la lógica de negocio (usuarios ligeros), simplemente disponen de un navegador web que se conectará a los servidores de aplicaciones que poseen toda la lógica de negocio, y estos les proporcionarán la interfaz.

Los usuarios se conectarán a través de un *firewall*, a un balanceador que tiene una IP virtual para aumentar la seguridad entre las capas. Este balanceador permite redirigir el tráfico hacía los diversos servidores de aplicación, consiguiendo que un servidor no se sature y reciba todo el tráfico.

Además se consigue que las actualizaciones se puedan realizar sin parar el servicio, ya que mientras se actualiza un servidor de aplicación, el balanceador de carga redireccionaría todo el tráfico a otro, y una vez actualizado, se le indicaría al balanceador que apuntará al actualizado; se actualizaría el primero y una vez actualizado volvería a repartir el tráfico entre todos los servidores de aplicación.

2. Servidores de aplicación: Poseen toda la lógica de negocio. Dado el número de personas del que dispone la Universidad de Valladolid, y estimando un uso simultáneo de un 10% de los usuarios, se prevé que el número de usuarios conectados al mismo tiempo en nuestro sistema será de 3300 usuarios, y que cada uno realizará una media de 2 transacciones, lo que crea una carga de trabajo de 6600 transacciones.

Así que dicha carga se repartirá en 3 servidores sabiendo que cada uno puede soportar eficientemente una carga de trabajo de 3000 transacciones, teniendo soporte efectivo de 9000 transacciones simultáneas.

El reparto de la carga de trabajo entre los servidores será administrado por un balanceador de carga.

Estos servidores serán replicados en caso de que el sistema crezca y se ralentice o deteriore el servicio que ofrece la aplicación, consiguiendo una mayor escalabilidad, al poder distribuir la carga gracias a un balanceador de carga, consiguiendo que no se sobrecarguen los servidores de aplicación y se mantenga de manera efectiva el servicio.

3. Servidores de bases de datos: En ellos se encuentran los datos de la aplicación.

Como se dijo anteriormente, se prevé una carga máxima de 6600 transacciones simultáneas, y los 3 servidores de aplicación que colocaremos nos permitirán hasta 9000. Por lo que para evitar posibles cuellos de botella, calcularemos la cantidad de servidores de bases de datos que hacen falta sobre la capacidad que proporcionan los servidores de aplicación.

Los servidores que utilizamos son de uso exclusivo para el procesamientos de datos, y éstos son capaces de soportar 11000 transacciones por lo que usaremos 2 servidores de bases de datos.

Uno de estos servidores de bases de datos se usará de forma activa para la consulta y el tratamiento de los datos. Añadiendo un servidor adicional pasivo, que se encargará de realizar la replica de los datos que se encuentran en el servidor principal, para de esta manera conseguir mayor disponibilidad y la posibilidad de copia de seguridad, en caso de que fallase el servidor activo.

A través de el balanceador se configurará un servidor principal que será el activo, y otro que actuará sólo en caso de que el servidor principal tenga problemas, consiguiendo que la disponibilidad sea mayor. En caso de querer aumentar la seguridad, o incluso repartir la carga como en los servidores de aplicación, se podrá realizar mas de una replicación del servidor de bases de datos.

Además para intentar disminuir los ataques externos se protegerá cada uno de los componentes de las diferentes capas con *firewall*, para que nos proporcione mayor seguridad.

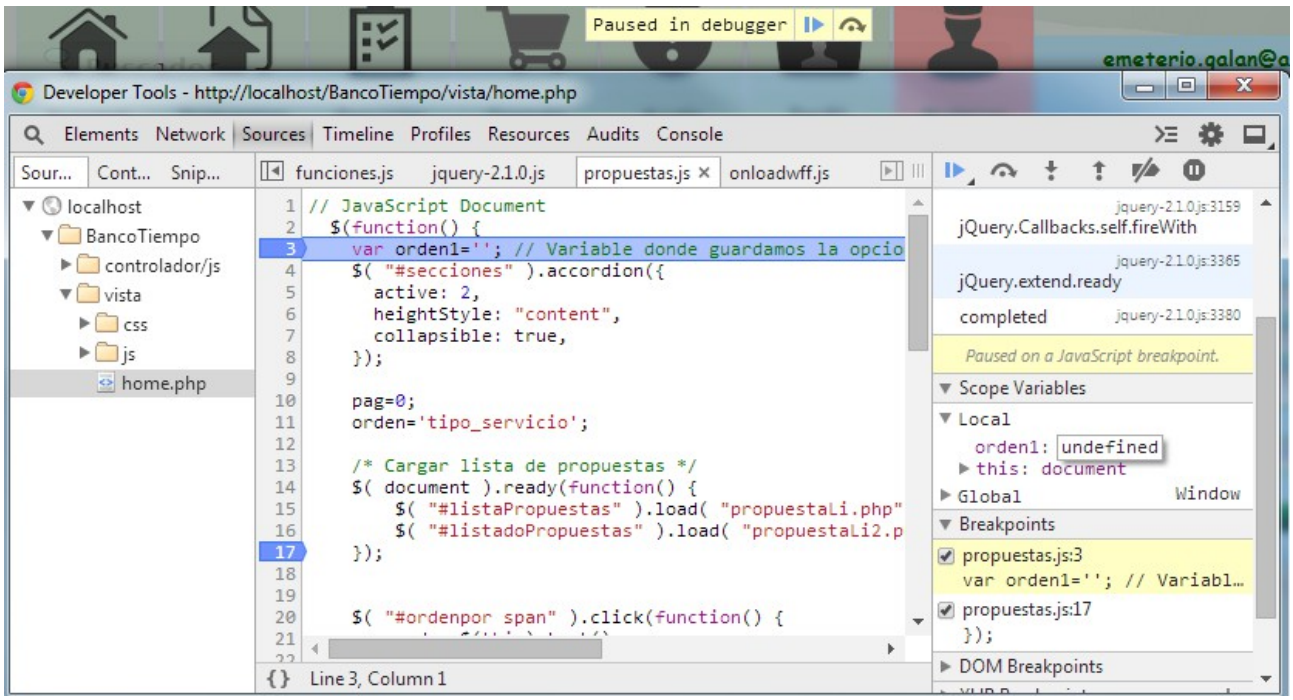
6. - Pruebas

6.1. - Pruebas de caja blanca

Durante el desarrollo de la aplicación se han realizado todas las pruebas necesarias de caja blanca, para la realización de ellas se ha utilizado como depurador y visor la herramienta para desarrolladores que proporciona el navegador *Google Chrome* en el cual, se puede cargar el javascript, interrumpir su ejecución en un punto y a través de la consola introducir los valores deseados.

También a través de la consola se pueden hacer llamadas a las funciones individualmente permitiendo saber su comportamiento sin la influencia de otros factores.

Gracias a la pestaña de *Red* o *Network* se pueden ver todos los recursos a los que se ha intentado acceder, así como el método empleado. Ésto ha resultado muy útil, ya que ha permitido ver el valor que devuelven algunas páginas (previamente preparadas para devolver algún valor) que son cargadas por *AJAX* y no se puede apreciar su comportamiento en la ejecución normal.



Para la realización de las pruebas de caja blanca en la aplicación móvil se ha utilizado el depurador propio del entorno de desarrollo.

6.2. - Pruebas de caja negra

Aunque se han realizado numerosas pruebas de caja negra durante la implementación de la aplicación, sólo se mostrarán un conjunto limitado de ellas.

ID	Entrada	Acción Esperada	Resultado
001	Añadir servicio, rellenando todos los campos.	Aparece en el listado de Servicios	OK
002	Eliminar servicio	No aparece en el listado de servicios	OK
003	Se rechaza una petición	La petición no aparece asociada al servicio	OK
004	Se rechaza una petición	La petición cambia de estado en la base de datos	OK
005	Se rechaza una petición	Aparece la petición en consumiciones rechazadas	OK
006	Se cambia el estado de un usuario	No se le permite el acceso	OK
007	Regenerar contraseñas	Se cambia las contraseñas de todos los usuarios que no son moderadores	OK
008	Descargar Tabla consumicion.	Formato correcto entre las cabeceras y los resultados obtenidos.	OK
009	Descargar Tabla usuario.	Formato correcto entre las cabeceras y los resultados obtenidos.	OK
010	Descargar Tabla contraseñas temporales.	Formato correcto entre las cabeceras y los resultados obtenidos.	OK
011	Descargar Tabla propuesta.	Formato correcto entre las cabeceras y los resultados obtenidos.	OK
012	Descargar Tabla servicio.	Formato correcto entre las cabeceras y los resultados obtenidos.	OK
013	Se realiza una consumición.	Aparece la consumición en la lista de consumiciones vendidas.	OK
014	Se realiza una consumición.	Aparece la consumición en la lista de consumiciones compradas.	OK
015	Se hace una petición.	Aparece la petición en la lista de consumiciones pendientes.	OK
016	Se realiza una búsqueda por título	Muestra el servicio con dicho título.	OK
017	Se realiza una búsqueda por precio máximo.	Muestra los servicio inferiores a ese precio.	OK
018	Una persona no logueada intenta acceder a la aplicación.	El sistema evita la entrada.	OK
019	Una persona se desloguea.	El sistema evita la carga de páginas.	OK

Tabla 037

7. - Aplicación Android

7.1. - Introducción

Motivación

Como complemento a un banco de tiempo implementado y funcionando sobre un servidor web, se quiere que los usuarios que han ofrecido servicios puedan, no sólo a través del navegador, consultar las peticiones de ofertas que les han hecho, y desde cualquier lugar poder aceptar, consultar o rechazar las ofertas.

Aunque desde los dispositivos móviles se permite utilizar el navegador, y con ello, la aplicación web, se desea crear la aplicación móvil, ya que permite que se adapte totalmente a los dispositivos además de una mayor rapidez en su uso, ya que no se deben de cargar páginas alojadas en un servidor, sólo las diferentes conexiones con la base de datos.

Alcance del sistema

La aplicación que se desea desarrollar permite complementar la funcionalidad ya existente en una aplicación web.

La aplicación permitirá el acceso a los servicios de cada usuario y poder ver las peticiones realizadas sobre esos servicios, y desde ellas, consultarlas, aceptarlas o denegarlas.

Queda fuera del alcance del sistema la creación , eliminación o modificación de los servicios, así como la petición de ofertas o comentarios sobre ellas.

Estas funcionalidades que quedan fuera del alcance actual del sistema, deja abierto la posibilidad de posibles ampliaciones.

Identificación del entorno tecnológico

La aplicación se desarrollará para terminales Android , debido a su gran extensión y el costo económico de desarrollo.

Se desarrollará para ser eficiente en Android 4.1 con API 17, aunque esto no limita su uso en otras versiones de Android.

7.2. - Desarrollo de las clases

MainActivity.class

Es la clase que se encarga del login.

En un principio se intentó hacer que se cargará la actividad de servicios en caso de que se tuvieran las preferencias compartidas almacenadas. Ésto funcionaba bien, pero como se da la opción de poder acceder sin guardar las preferencias, se debe de cargar primero la página de login (MainActivity.class) para poder crear el intent que mandara el bundle a la siguiente activity, por lo que se hará la comprobación directamente desde la página de login (MainActivity.class) y en caso de que existan preferencias guardadas, cargará automáticamente la página de servicios.

Se han creado dos funciones para una mejor compresión del código, estas funciones son la de "comprueba", que comprueba si existen un usuario y una contraseña dada en la base de datos. Y "recordarPref", que almacena el nombre de usuario y la contraseña si se ha decidido que se guarden para no tener que introducir de nuevo los datos.

A continuación se mostrarán el código mas relevante de la clase.

1.- Comprobamos que existen preferencias compartidas, en caso de que existan, recogemos las preferencias compartidas, y las cargamos en el bundle para almacenarlos en el intent y abrir el activity de servicios, y una vez que le hemos lanzado cerramos la clase actual.

```
// Si no hay preferencias compartidas definidas
if (TBapp.getPrefs().getString("username", null) == null) {
      .
      .
      //(2)
      .
}
// Si existen preferencias compartidas las recogemos
else {
      // Cargamos el bundle con la preferencia compartida
      b.putString("nick", TBapp.getPrefs().getString("username", ""));
      intentServicios.putExtras(b);// Cargamos el intent
      startActivity(intentServicios);// Iniciamos la nueva actividad, la de Servicios.
      finish(); // Cerramos la actividad de login (Main)
}
```

2.- Si no existen preferencias compartidas, implementamos el listener de click del botón y recogemos los datos introducidos.

Comprobamos que el usuario y la contraseña son correctos, utilizando la función "comprueba" que nos devuelve un booleano. Si es correcto miraremos si se ha marcado, o no, la opción de recordar los datos de usuario. Si no es correcto se mostrará un mensaje de error.

```
// Si no hay preferencias compartidas definidas
if (TBapp.getPrefs().getString("username", null) == null) {
  // Recogemos los parámetros que hemos introducido.
```

```
  final EditText txtNombre = (EditText) findViewById(R.id.editTextUser);
  final EditText txtPass = (EditText) findViewById(R.id.editTextPassword);
  final CheckBox chkRecordar = (CheckBox) findViewById(R.id.checkBoxRecordar);
  final Button btnEnviar1 = (Button) findViewById(R.id.botonEnviar1);

  // Implementamos el evento "click" del botón
  btnEnviar1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    // Comprobamos el usuario y miramos si quiere recordar las preferencias.
      if (comprueba(txtNombre.getText().toString(),txtPass.getText().toString())) {
            .
            .
           (3)
            .

      }
       else {
         // ERROR EN EL USUARIO
      Toast.makeText(MainActivity.this,R.string.usuarioNoEncontrado,
Toast.LENGTH_LONG).show();
      }
    }
  });
}
```

3.- Una vez comprobado que el usuario y la contraseña son correctos , comprobamos si ha seleccionado la opción de recordar las preferencias, en caso de que la haya marcado, se utilizará la función "recordarPref" cuya función es recordar el nombre de usuario y contraseña introducidos.

Si eligió recordar las preferencias, las recogemos para almacenarlas en el bundle que utilizaremos después al final de la comprobación para lanzar el intent.

Sin embargo, si eligió no recordar las preferencias, recogeremos el nombre de usuario directamente del campo del TextView para cargar el bundle.

```
if (chkRecordar.isChecked()) {// RECORDAR
  // Guardamos la preferencias
  recordarPref(txtNombre.getText().toString(),txtPass.getText().toString());
  // Cargamos el bundle con la preferencia compartida
  b.putString("nick", TBapp.getPrefs().getString("username", ""));
Toast.makeText(MainActivity.this,R.string.recordarDatos,Toast.LENGTH_LONG).show();
}
else {// NO RECORDAR
   // Mandamos el usuario en caso de que no haya elegido recordar.
   b.putString("nick", txtNombre.getText().toString());
   Toast.makeText(MainActivity.this,R.string.noRecordar, Toast.LENGTH_LONG).show();
}

intentServicios.putExtras(b);// Cargamos el intent
startActivity(intentServicios);// Iniciamos la nueva actividad, servicios
finish();// Cerramos la actividad de login (Main)
```

4.- Se ha creado una función que nos permite comprobar, pasados el nombre de usuario y la contraseña, si existe y es correcto en la base de datos, con lo que se consigue mayor legibilidad del código.

```
@SuppressWarnings("deprecation")
```

```
public boolean comprueba(String nom, String pass) {
	// Recuperamos el contenido de la base de datos
	Cursor cursor;
	String whereClause = "nick =? AND pass =?"; // Consulta where
	String[] whereArgs = new String[] { nom, pass };// Parámetros que se pasan a la
consulta where
	// Descargamos los datos para consultarlos, aplicando las restricciones.
	cursor=TBapp.lee_de_bbdd("USUARIO_REG",whereClause, whereArgs);// Leer de la base
de datos (Tabla,Sentencia Where , campos del where)
	startManagingCursor(cursor);

	if (cursor.getCount() != 0) { // Hacemos un count para saber cuantas filas
tenemos de la consulta anterior.
		TBapp.getDb().close();
		return true;
	} else {
		TBapp.getDb().close();
		return false;
	}
}
```

5.- Se ha creado una función que nos permite recordar las preferencia, pasados el nombre de usuario y la contraseña. Se crea un objeto Editor, que nos permite editar, o crear, las preferencias existentes.

```
private void recordarPref(String nom, String pass) {
	Editor editor = TBapp.getPrefs().edit();
	editor.putString("username", nom);
	editor.putString("password", pass);
	editor.commit();
}
```

BaseActivity.class

Esta clase, que extiende de Activity, nos permite compartir funcionalidades entre varias activities, en ella implementaremos el menú, en el que habrá el menú de preferencias y el de cerrar sesión, también implementará la función "borrarPreferencias" que será utilizada cuando se seleccione la opción de "cerrar sesión" en el menú.

Creamos el menú de opciones, y la acción que se debe realizar cuando se pulse cada una de las opciones del menú.

```
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuInflater inflater = getMenuInflater(); //
		inflater.inflate(R.menu.menu, menu); //
		return true; // obligatorio retornar true
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		switch (item.getItemId()) { //
		case R.id.itemPreferencias:
			startActivity(new Intent(this, PreferenciasActivity.class));
			break;
		case R.id.itemLogout:
			borrarPreferencias();
			break;
		}
		return true;
	}

	// Fin Menu
```

Creamos una función que nos permitirá borrar las preferencias compartidas y cerrará la aplicación. Nos mostrará además un mensaje diciendo que se han borrado las preferencias compartidas.

```
	private void borrarPreferencias(){
		Editor editor = TBapp.getPrefs().edit();
		editor.putString("username", null);
		editor.putString("password", null);
		editor.commit();
		Toast.makeText(this,R.string.borrarPreferencias,
Toast.LENGTH_LONG).show();
		finish(); // Cerramos la aplicación
	}
```

DbHelper.class

Esta clase se encarga de la conexión con la base de datos, ésta extiende de la clase SQLiteOpenHelper.

En este caso hemos creado unas funciones para agregar datos aleatorios a la base de datos (cargarUsuariosEnDb, cargarServiciosEnDb, cargarOfertasEnDb), la funcionalidad sería mas real, si la carga se hiciera desde una base de datos.

En primer lugar vamos a definir dos variables, como son el nombre que daremos a la base de datos, y la versión de ésta, para poder llamar al constructor. De ésta manera si cambiamos algo en la estructura de la base de datos, con cambiar el numero de la versión de la base de datos eliminará la existente y cargará la nueva.

```
public class DbHelper extends SQLiteOpenHelper {
static final int VERSION = 1;
static final String DATABASE = "timebank.db";

// Constructor
public DbHelper(Context context) {
      super(context, DATABASE, null, VERSION);
}
.
.
.
```

En la primera ejecución se creará la base de datos, después (en las siguientes ejecuciones), ésta parte no la ejecutará.

```
public void onCreate(SQLiteDatabase db) {
      String sql;

      // Creamos la tabla de USUARIO
      sql = "create table USUARIO_REG  (_id INTEGER PRIMARY KEY AUTOINCREMENT,nick
VARCHAR (20),mail VARCHAR (30), pass VARCHAR (12), nombre VARCHAR (30))";
      db.execSQL(sql);

      /* Creamos la tabla de SERVICIO, dado que descargará la información de una pagina
no están creados las claves foráneas,ya que eso es gestionado en la base de datos del
servidor */
      sql = "create table SERVICIO  (_id INTEGER PRIMARY KEY AUTOINCREMENT,n_ofertante
VARCHAR (20), fecha DATETIME , gasto_economico FLOAT, estado INT,tipo VARCHAR (20),
descripcion TEXT)";
      db.execSQL(sql);

      /* Creamos la tabla de CONSUMICION, dado que descargará la información de una
pagina no están creados las claves foráneas, ya que eso es gestionado en la base de
datos del servidor */
      sql = "create table CONSUMICION  (_id INTEGER PRIMARY KEY
AUTOINCREMENT,n_consumidor VARCHAR (20), id_servicio INT, fecha DATETIME , comentario
TEXT, rechaza BOOLEAN, motivo TEXT)";db.execSQL(sql);

      cargarUsuariosEnDb(db); // Cargamos los usuario en la base de datos
      cargarServiciosEnDb(db);// Cargamos los servicios en la base de datos
      cargarOfertasEnDb(db);// Cargamos las consumiciones en la base de datos
}
```

Dado que si existe una base de datos, ésta no es creada, comprobamos la versión de la base de datos, y si ésta es menor que la que estamos ejecutando, borrará todas las tablas que existen, y creará la nueva base de datos

```
// Llamada siempre que tengamos una nueva version
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
	// Aqui van las sentencias del tipo ALTER TABLE, de momento lo hacemos
	// mas sencillo:
	db.execSQL("drop table if exists USUARIO_REG "); // borra la vieja base de datos
	db.execSQL("drop table if exists SERVICIO"); // borra la vieja base de datos
	db.execSQL("drop table if exists CONSUMICION"); // borra la vieja basede datos
	onCreate(db); // crea una base de datos nueva
}
```

TBApplication.class

Esta clase contiene el estado común de la aplicación , si existe alguna parte de la aplicación funcionando, el objeto Application es creado, y se podrá instanciar desde cualquiera de las clases de la aplicación.

En ella vamos a colocar la funcionalidad para poder leer y escribir de la base de datos, abrir la base de datos en modo lectura o escritura, devolvernos la base de datos o recoger las preferencias, por lo que simplemente usando el objeto application podremos usar estas funciones.

```
public class TBApplication extends Application implements
OnSharedPreferenceChangeListener {

	private SharedPreferences prefs;
	private DbHelper dbHelper;
	private SQLiteDatabase db;

.


.


.
```

Al utilizar variables privadas, solamente se podrá acceder a ellas a través de la clase TBAplication.

En la creación se recuperan las preferencias, y se hace una instancia de DbHelper, para pode acceder a la base de datos.

```
@Override
public void onCreate() {
	super.onCreate();
	this.prefs = PreferenceManager.getDefaultSharedPreferences(this);
	this.prefs.registerOnSharedPreferenceChangeListener(this);
	dbHelper = new DbHelper(this);
}
```

Utilizamos una función para abrir la base de datos en los diferentes modos, ya que la apertura de la base de datos siempre en modo escritura, es peligroso además de que consume recursos innecesariamente.

```
public void abrirDb(char modo) {
  if (modo == 'W') {
    db = this.dbHelper.getWritableDatabase();// Abrimos nuestra base de datos en modo
escritura
  }
  else {
   db = this.dbHelper.getReadableDatabase();// Abrimos nuestra base de datos en modo
lectura
   }
}
```

La función lee_de_bbdd nos devuelve un cursor con los datos, pasándole como parámetros la

tabla a la que se hace la consulta, la sentencia Where para la consulta y los datos que se utilizarán en la sentencia where.

```
public Cursor lee_de_bbdd(String tabla, String sentenciaWhere,String[] camposWhere) {
	// Cogemos los datos de la base de datos
	Cursor cursor = null; // Cursor que devolveremos
	abrirDb('R');
	try {
		cursor = db.query(true, tabla, null, sentenciaWhere, camposWhere,null,
null, null, null);
	}
	finally {
	}
	 return cursor;
}
```

La función escribe_en_bbdd es utilizada para cambiar el estado de rechazar o aceptar una oferta, por lo que se le pasa la tabla, la sentencia where, los datos para el where en el que se pasa el identificador de la consumición, y el nuevo estado que se va a guardar.

```
public void escribe_en_bbdd(String tabla, String sentenciaWhere,String[] camposWhere,
String campo, String dato) {

	ContentValues values = new ContentValues();
	values.put("rechaza", dato);
	abrirDb('W');// Abrimos la base de datos en modo escritura
	try {
		db.update(tabla, values, sentenciaWhere, camposWhere);
	}finally {
		 db.close();
	}
}
```

Tenemos otras tres funciones, que nos devuelven la referencia a las variables que tenemos como private.

```
public SQLiteDatabase getDb() {
	return db;
}

public SharedPreferences getPrefs() {
	return prefs;
}

public DbHelper getDBHelper() {
	return dbHelper;
}
```

PreferenciasActivity.class

Esta clase es la que nos permite utilizar y usar las preferencias, extiende de PreferenceActivity, es la activity a la que llamamos desde el menú, la que nos muestra las preferencias compartidas y donde podemos editarlas.

```
public class PreferenciasActivity extends PreferenceActivity { //
	@SuppressWarnings("deprecation")
	@Override
	protected void onCreate(Bundle savedInstanceState) { //
		super.onCreate(savedInstanceState);
		addPreferencesFromResource(R.xml.preferencias);
	}
}
```

ServiciosActivity.class

Es la clase que gestiona los servicios, nos muestra los servicios de cada usuario, y nos permite seleccionar uno de ellos para que nos muestre sus peticiones.

Se utiliza una clase de tareas asíncronas (AsyncTask) para la consulta con la base de datos, recuperaServiciosAsync, que nos devolverá un cursor con los datos recogido de la base de datos.

```
////////////////////////////////////////////////////////////////////////////
/// Clase asyncTask, para conectarse y cargar el cursor con los datos de la base de
datos,
//  El cursor es devuelto y tratado en el método principal
////////////////////////////////////////////////////////////////////////////////

public class recuperaServiciosAsync extends AsyncTask<String,Integer,Cursor>{
      @Override
      protected Cursor doInBackground(String... params) {
            Cursor cursor;
          // Leer de la base de datos (Tabla,Sentencia Where , campos del where)
            cursor=TBapp.lee_de_bbdd("SERVICIO","n_ofertante =?", params);
            return cursor;
      }
}
```

En el flujo principal, creamos dos variables que utilizaremos en toda la clase, el cursor donde almacenaremos los resultados de las consultas a la base de datos, y el listview donde mostraremos los servicios.

En la creación llamamos al recurso que tiene el diseño visual del activity. Y asignamos la referencia del listview para utilizarle.

```
public class ServiciosActivity extends BaseActivity {

      Cursor cursor;
      ListView listServicios;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_servicios);
            // Cogemos la referencia al ListView para escribir en él
            listServicios = (ListView) findViewById(R.id.listaServicios);
      }
.
.
.
```

1.- Recogemos el nick que recibimos de la clase Main, y lo almacenamos en la variable nom.

Instanciamos el TextView y le asignamos la referencia a su recurso, en la instancia del TextView le asignamos como texto el nick del usuario que hemos almacenado en la variable “nom”.

Creamos un cursor en el que recibiremos la ejecución de la AsyncTask. Después ejecutamos la AsyncTask a la que le pasamos como parámetro la variable nom, que contiene el usuario.

Una vez que tenemos el cursor, creamos las variables que pasaremos al adapter para poder asignar los valores a la referencia correspondiente del archivo XML de la fila, para ser mostrados.

Después creamos el adapter pasándole todos los parámetros mencionados y lo cargamos en el listview.

```
@SuppressWarnings("deprecation")
@Override
protected void onResume() {
	super.onResume();
	String nom = ""; // Variable para almacenar el nick
	// Recogemos los datos que nos pasa la anterior activity.
	Bundle bundle = this.getIntent().getExtras();
	nom = bundle.getString("nick"); // Recogemos el nick del usuario
	// Instanciamos el textview en el que pondremos el nombre de usuario
	TextView usuario= (TextView) findViewById(R.id.txtNickUserService);
	usuario.setText(nom); // Ponemos el nombre de usuario

	// Cursor en el que almacenaremos lo devuelto por la AsyncTask !!!
	Cursor cursor = null;
	try{
		try {
			// Recogemos la ejecución de la asynctask
			cursor = new recuperaServiciosAsync().execute(nom).get();
		} catch (Exception e) {
			e.printStackTrace();
		}

	String[] from = { "_id", "descripcion", "n_ofertante"};
	int[] to = { R.id.txtIdService, R.id.txtDescripcionService,
R.id.txtNickUserService};

	SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,R.layout.row_servicio,
cursor, from, to);

	listServicios.setAdapter(adapter); // Cargamos el list view
.
.
(2)
.
	}
	finally{
		TBapp.getDb().close();
	}
}
```

2.- Creamos un Listener para recoger cuando hacemos click sobre un elemento del listview.

Creamos el intent para lanzar la Actividad de Ofertas, creamos un contenedor en el que recogemos el id del servicio en el que hemos pulsado, se le carga en el intent y le ejecutamos, lanzando el activity de ofertas con las ofertas del servicio pulsado.

```
listServicios.setOnItemClickListener(new
android.widget.AdapterView.OnItemClickListener() {
	@Override
	public void onItemClick(AdapterView<?> arg0, View arg1,int arg2, long arg3) {
		// TODO Auto-generated method stub
		Intent intent = new Intent(ServiciosActivity.this,OfertasActivity.class);
		// Contenedor para mandar el id del servicio y poder consultar las ofertas.
		Bundle b = new Bundle();
		b.putLong("idServicio", arg3);// Devuelve el campo "_id" que utiliza el
ListViewer que es el mismo que utilizamos para saber el id del servicio
		intent.putExtras(b);
		startActivity(intent);
		}
	});
```

OfertasActivity.class

Es la clase que gestiona las peticiones de los usuario, nos muestra las ofertas que cada usuario a hecho para un determinado servicio, nos permite seleccionar una de las ofertas, y desde la misma activity consultarla, además de poder aceptarla o rechazarla.

Se utiliza una clase de tareas asíncronas (AsyncTask) para la consulta con la base de datos, recuperaOfertasAsync, que nos devolverá un cursor con los datos recogido de la base de datos.

```
////////////////////////////////////////////////////////////////////////////
Clase asyncTask, para conectarse y cargar el cursor con los datos de la base de datos,
////////////////////////////////////////////////////////////////////////////////

public class recuperaOfertasAsync extends AsyncTask<String, Integer, Cursor> {
	@SuppressWarnings("deprecation")
	@Override
	protected Cursor doInBackground(String... params) {
		Cursor cursor;
		// Descargamos los datos para consultarlos, aplicando las restricciones.
		// Leer de la base de datos (Tabla,SentenciaWhere,campos del where)
		cursor = TBapp.lee_de_bbdd("CONSUMICION", "id_servicio =?", params);
		startManagingCursor(cursor);
		return cursor;
	}
}
```

Creamos una clase que nos devolverá un ListAdapter, ya con la información de las ofertas, que asignaremos directamente después en el flujo normal.

Se crea un cursor en el que se almacena la ejecución de la AsyncTask, se crean las variables para hacer la consulta y restringirla únicamente al servicio seleccionado.

Ejecutamos la AsyncTask (recuperaOfertasAsync) y almacenamos el resultado en la variable de tipo Cursor.

Asignamos los campos que queremos d ela consulta con sus referencias del archivo rowOferta.xml para crear el adapter. Creamos el adapter y lo devolvemos.

```
// ////////////////////////////////////
// / Devolver La lista en un adpater
// ////////////////////////////////////////

public ListAdapter devuelveLista(String idServ) {

	// Obtenemos los datos de la base de datos
	// Cursor en el que almacenaremos lo devuelto por la AsyncTask !!!
	Cursor cursor = null;

	// Parámetros que se pasan a la consulta where
	String[] whereArgs = new String[] { (String) idServ };

	try {
		cursor = new recuperaOfertasAsync().execute(whereArgs).get(); // Recogemos
la ejecución de la asynctask
```

```
		} catch (Exception e) {
			e.printStackTrace();
		}

		String[] from = { "_id", "comentario", "n_consumidor", "rechaza" };

		int[] to = { R.id.txtIdOferta, R.id.txtComentarioOferta,R.id.txtNickUserOferta,
R.id.txtRechazaOferta };

		@SuppressWarnings("deprecation")
		SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,R.layout.row_oferta,
cursor, from, to);

		return adapter;
}
```

Creamos las variables que utilizaremos en el resto de la clase. Un cursor para almacenar el resultado de la conexión a la base de datos, el ListView donde se cargarán las ofertas que descarguemos y un flag para saber si se ha realizado algún cambio y tenemos que vovler a cargar las lista de ofertas.

Llamamos a la referencia del activity que contiene el aspecto visual (R.layout.activity_ofertas) y se le asigna al activity. También se asigna la referencia al listview.

Recogemos la identificación del servicio que hemos recibido del activity anterior, y creamos una variable transformando este identificador a una variable de tipo String.

Asignamos al listView de ofertas el adapter que nos devuelve la funcion devuelveLista, a la que le pasamos el identificador del servicio, devolviéndonos todas las ofertas de ese servicio.

```
public class OfertasActivity extends BaseActivity {

		Cursor cursor;
		ListView listOfertas;
		boolean cambio=false;

		@Override
		protected void onCreate(Bundle savedInstanceState) {
			super.onCreate(savedInstanceState);
			setContentView(R.layout.activity_ofertas);

			// Cogemos la referencia al ListView para escribir en él
			listOfertas = (ListView) findViewById(R.id.listaOfertas);

			// Recogemos los datos de la otra ventana
			Bundle bundle = this.getIntent().getExtras();
			final String idServicio = String.valueOf(bundle.getLong("idServicio"));

			try {
				// Cargamos la lista con el adaptador
				listOfertas.setAdapter(devuelveLista(idServicio));
.
.
(2)
(3)
.
.
		}
```

```
		finally {
		}
}
```

2.- Se ha creado un listener para cuando se hace click sobre un elemento de la lista.

Se crea una instancia, para poder realizar operaciones, del ToggleButton del elemento al que se ha hecho click, así como de un TextView donde recogemos el estado que tiene esa oferta en ese momento, y dependiendo de este estado se realizan diferentes operaciones.

En primer lugar si el ToggleButton no está visible, se hace visible.

Después se coloca una imagen (y color, en caso de error de imagen), el texto oportuno según el estado de la oferta y se marca pulsado o no.

De esta manera podemos ver el estado en el que se encuentra la oferta.

Una vez que es mostrado el ToggleButton, éste se puede pulsar, por lo que creamos un listener para saber cuando se pulsa el ToggleButton.

En el momento que pulsamos sobre el ToggleButton , se llama a la función aceptaOferta, en el que se le pasa el identificador de la oferta y el nuevo estado según corresponda, esta función cambia el estado en la base de datos, y se cambia el estado del ToggleButton como hemos indicado anteriormente. Además se mostrará un mensaje con el cambio realizado.

Como se realiza un cambio, el flag declarado como "cambio" se pone a true , que será utilizado en ScrollListenter para volver a cargar la lista con los cambios realizados.

```
listOfertas.setOnItemClickListener(new android.widget.AdapterView.OnItemClickListener()
{ // Listener para el listview
	@Override
	public void onItemClick(AdapterView<?> arg0, View arg1,int arg2, long arg3) {

		final ToggleButton toggRechaza = (ToggleButton)
arg1.findViewById(R.id.toggleRechazarOferta);
		final long id_oferta = arg3;

		TextView rec = (TextView) arg1.findViewById(R.id.txtRechazaOferta);
		// Estado actual en el momento de cargar el list view, por lo que sólo se
debe mostrar la primera vez
		if (toggRechaza.getVisibility() != View.VISIBLE) {

			toggRechaza.setVisibility(View.VISIBLE);
			if (rec.getText().toString().equalsIgnoreCase("0")) {
				toggRechaza.setBackgroundColor(0xFF00FF00);
				toggRechaza.setBackgroundResource(R.drawable.aceptada);
				toggRechaza.setText(R.string.txtAceptarOferta);
				toggRechaza.setChecked(true);// Lo marcamos como aceptada
			}
			else if (rec.getText().toString().equalsIgnoreCase("1")) {
				toggRechaza.setBackgroundColor(0xFFFF0000);
				toggRechaza.setBackgroundResource(R.drawable.rechazada);
				toggRechaza.setText(R.string.txtRechazarOferta);
				toggRechaza.setChecked(false);// Lo marcamos como rechazada
			}
			else {
```

```
                        toggRechaza.setBackgroundColor(0xFFFFFF00);
                        toggRechaza.setBackgroundResource(R.drawable.pendiente);
                        toggRechaza.setText(R.string.txtPendienteOferta);
                  }
            }
            // Listener para el botón toggle
            toggRechaza.setOnClickListener(new OnClickListener() {
                  public void onClick(View arg1) {
                  // ToggleButton toggRechaza =(ToggleButton) arg1 ;

                  cambio=true;// estamos haciendo un cambio en el estado

                  if (toggRechaza.isChecked()) {// Se acepta la oferta
                        aceptaOferta(id_oferta, true);
                        toggRechaza.setBackgroundColor(0xFF00FF00);
                        toggRechaza.setBackgroundResource(R.drawable.aceptada);

                        Toast toast =
Toast.makeText(OfertasActivity.this,getResources().getString(R.string.txtAceptadaLaOfer
ta).toString()+id_oferta,Toast.LENGTH_LONG);
                        toast.show();

                  }
                  else {// Se rechaza la oferta
                        aceptaOferta(id_oferta, false);
                        toggRechaza.setBackgroundColor(0xFFFF0000);
                        toggRechaza.setBackgroundResource(R.drawable.rechazada);
                        Toast toast =
Toast.makeText(OfertasActivity.this,getResources().getString(R.string.txtRechazadaLaOfe
rta).toString()+ id_oferta,Toast.LENGTH_LONG);
                        toast.show();

                  }
            }
});
```

3.- Creamos un listener de Scroll, para controlar los cambios realizados, si se ha realizado un cambio y nos movemos por la lista, en ese momento, se vuelve a cargar la lista con los datos nuevos actualizados.

Para saber si se ha realizado algún cambio, utilizamos la variable cambio que hemos declarado al principio.

```
// Vamos a utilizar el listener de Scroll para actualizar la lista cuando se haga algún
cambio en ella
listOfertas.setOnScrollListener(new OnScrollListener() {
      @Override
      public void onScroll(AbsListView view, int firstVisibleItem,int visibleItemCount,
int totalItemCount) {
      // TODO Auto-generated method stub

      }

      @Override
      public void onScrollStateChanged(AbsListView view,  int scrollState) {
      // TODO Auto-generated method stub
            if(cambio==true){// Se ha hecho un cambio
                  // Cargamos la lista con los cambios nuevos
                  listOfertas.setAdapter(devuelveLista(idServicio));
```

```
                    cambio=false;
                }
                toggRechaza.setVisibility(View.GONE); // Cerramos el toggle si nos movemos
        }
});
```

Se utiliza una función auxiliar para modificar el estado de la oferta, se ha decido utilizar una función aparte, por que es utilizada en varios puntos del código.

```
// ////////////////////////////////
// Rechazar oferta
// //////
public void aceptaOferta(long id_oferta, boolean acepta) {

        String[] idOferta = { String.valueOf(id_oferta) };

        if (acepta) {
                TBapp.escribe_en_bbdd("consumicion", "_id=?", idOferta, "rechaza","0");
        }
        else {
                TBapp.escribe_en_bbdd("consumicion", "_id=?", idOferta, "rechaza","1");
        }

}
```

7.3. - Recursos (/res)

Drawable

En estas carpetas (drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xhdpi, drawable xxhdpi)se guardan las imágenes que se utilizan en la aplicación dependiendo de la resolución del dispositivo en el que se ejecute.

Al igual que en values, se pueden crear carpetas para cada idioma en el que se mostrarán la imágenes según el idioma, drawable-es-hdpi. Se planteó el uso de esto para los diferentes idiomas, pero se optó, por que se pensó que era un gasto innecesario ya que sólo varía en tres imágenes y habría que duplicar todas , por utilizar las mismas imágenes y sobreponer, en su caso, el texto en su correspondiente idioma.

Imagen	Descripción
	Icono de la aplicación.
	Oferta aceptada (sobre ésta se pondrá el texto en el idioma correspondiente)
	Oferta pendiente (sobre ésta se pondrá el texto en el idioma correspondiente)
	Oferta Rechazada (sobre ésta se pondrá el texto en el idioma correspondiente)
	Icono de las preferencias
 .	Imagen que se utiliza en la acitivity principal.
	Imagen que se utiliza en el encabezado de la lista de ofertas

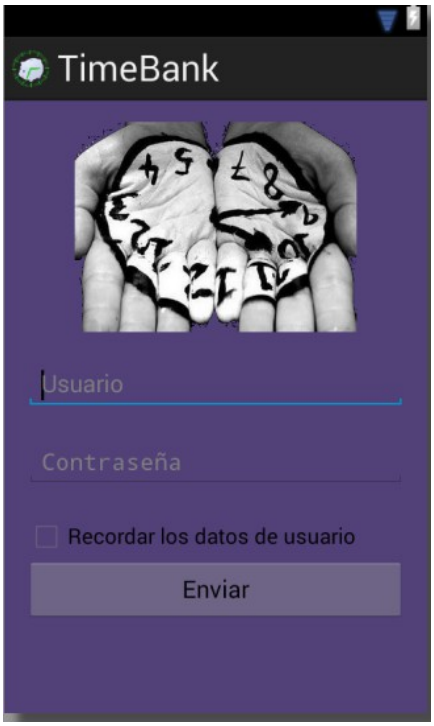
	Imagen que se utiliza en el encabezado de la lista de servicios
	Icono de desconexión
	Icono de usuario que aparece en el activity de ofertas.

Tabla 038

Layout

Son la parte visual de los diferentes componentes, se explicarán sólo los atributos que se consideren que no son autoexplicativos.

activity_main.xml



Es la parte visual del mainactivity.class , está formado por:

```
<RelativeLayout
    .
    .
    android:background="#524179" >
```

Fondo morado de toda la activity.

```
    <ImageView
        .
        .
        android:src="@drawable/img_principal" />
```

Imagen que se muestra encima de los botones

```
    <EditText
        android:id="@+id/editTextUser"
        .
        .
        android:hint="@string/usuario" >
```

```
        <requestFocus />
    </EditText>
```

Caja de texto para introducir el usuario, muestra el texto "@string/usuario" cuando está vacía

```
    <EditText
        .
        .
        android:hint="@string/contrasena"
        android:inputType="textPassword" />
```

Caja de texto para introducir la contraseña, de tipo password que oculta el texto introducido

```
    <CheckBox
        android:id="@+id/checkBoxRecordar"
        .
        .
        android:text="@string/recordarDatos" />
```

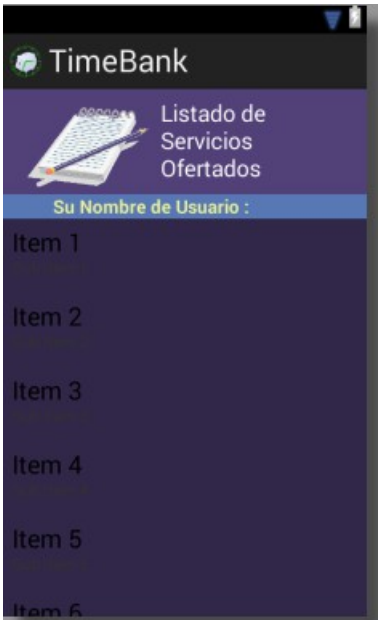
Checkbox para indicar si se quieren recordar los datos.

```
    <Button
        .
        .
        android:text="@string/enviar" />
```

Botón enviar

```
</RelativeLayout>
```

activity_servicios.xml



Es la parte visual del serviciosactivity.class , está formado por:

```
<LinearLayout  >

    <LinearLayout >

        <ImageView
            android:src="@drawable/listservicio" />
```

Imagen que se utiliza en el titulo.

```
        <TextView
            android:textColor="#FFF"
            android:textSize="20sp" />
```

Texto del titulo con el color indicado y el tamño indicado.

```
    </LinearLayout>

    <LinearLayout
        android:background="#5877B5" >
```

Fondo del linear layout donde se pondra el nombre de usuario del que son los servicios.

```
        <TextView
            android:id="@+id/labelNickUserService"

            android:textColor="#ECF296"
            android:textSize="16sp"
            android:textStyle="bold" />
```

Texto de "su nombre de usuario", con el estilo indicado.

```
        <TextView

            android:minWidth="120dp"
            android:text=""
            android:textAlignment="inherit"
            android:textColor="#ECF296"
            android:textSize="17sp" />
```

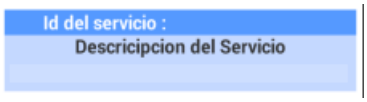
Texto donde se mostrará el nombre del usuario, con el estilo indicado.

```
    </LinearLayout>

    <ListView
        android:id="@+id/listaServicios"
        android:background="#6000" />
```

Lista donde se cargarán los servicios.

```
</LinearLayout>
```

row_servicio.xml



Aspecto de las filas que mostrará cada servicio.

```
<LinearLayout  >

    <LinearLayout >

        <TextView
            android:id="@+id/labelIdService"

            android:textStyle="bold"
            android:textSize="16sp"
            android:paddingRight="12dp"/>
```

Texto donde se mostrará el texto de "id del servicio", con el estilo indicado.

```
        <TextView
            android:id="@+id/txtIdService"
            android:textSize="17sp"/>
    </LinearLayout>
```

Texto donde se mostrará el identificador del servicio, con el estilo indicado.

```
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#C4D8FF" >

        <TextView
            android:id="@+id/labelDescripcionService"
            android:text="@string/labelDescripcionService"
            android:textAlignment="center"
            android:textStyle="bold"
            android:textSize="16sp" />
```

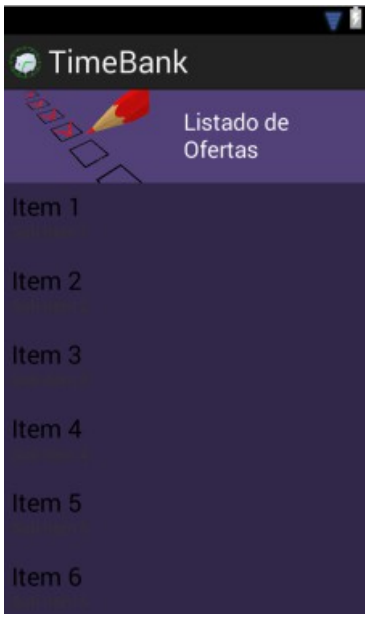
Texto donde se mostrará el texto de "Descripcion del servicio", con el estilo indicado.

```
    </LinearLayout>

    <LinearLayout
        android:padding="5dip">

        <TextView
            android:id="@+id/txtDescripcionService"
             />
```

Texto donde se mostrará la descripcion del servicio, con el estilo indicado.

```
    </LinearLayout>
</LinearLayout>
```

activity_ofertas.xml



Es la parte visual del ofertasactivity.class , está formado por:

```
<LinearLayout  >

    <LinearLayout >

        <ImageView
            android:id="@+id/imageView1"
            android:contentDescription="@string/tituloOfertas"
            android:src="@drawable/listofertas" />
```

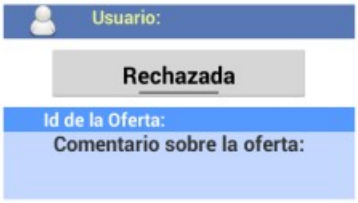
Imagen que se utiliza en el titulo.

```
        <TextView

            android:text="@string/tituloOfertas" />
```

Texto del titulo.

```
    </LinearLayout>

    <ListView
        />

</LinearLayout>
```

row_oferta.xml




```
<LinearLayout >

    <LinearLayout>

        <ImageView
            android:id="@+id/imageView1"
            android:src="@drawable/usuario" />
```

Imagen que se utiliza para indicar el usuario.

```
        <TextView
            android:id="@+id/LabelUsuarioOferta"
             />
```

Texto usuario.

```
        <TextView
            android:id="@+id/txtNickUserOferta"
            />
```

Texto donde se muestra el usuario que hace la peticion.

```
    </LinearLayout>

    <LinearLayout >

        <ToggleButton
            android:id="@+id/toggleRechazarOferta"
            android:onClick="onToggleClicked"
            android:visibility="gone" />
```

Función que se llama al hacer click, y la visibilidad gone, hace que el toggle este oculto y desplace hacia abajo el resto de componentes cuando se hace visible.

```
    </LinearLayout>

    <LinearLayout >

        <TextView
            android:id="@+id/labelIdOferta"
          />
```

Texto de "Id de la oferta".

```
        <TextView
            android:id="@+id/txtIdOferta" />
```

Texto donde se muestra el identidicador de la oferta.

```
    </LinearLayout>

    <LinearLayout>

        <TextView
            android:id="@+id/LabelComentarioOferta"
             />
```

Texto de "comentario sobre la oferta"

```
        <TextView
            android:id="@+id/txtComentarioOferta"
             />
```

Texto con el comentario de la oferta

```
        <TextView
            android:id="@+id/txtRechazaOferta"
            android:visibility="invisible" />
```

Texto donde se almacena el estado de la oferta, que se utiliza para colorear el toggle en la clase, por lo que está invisible.

```
    </LinearLayout>

</LinearLayout>
```

Menu

Contiene los archivos necesarios para la configuración del menú donde se pueden añadir los diferentes items.

Values

En esta carpeta están los archivo donde se predefinen los valores de la aplicación. Nos centraremos en el archivo strings.xml que es en el que se guardan los textos que se utilizarán en la aplicación.

Para poder utilizar varios idiomas se crean copias de esta carpeta con su sufijo correspondiente, por lo que si el dispositivo está configurado con otro idioma utilizará ese archivo en lugar de el que existe por defecto.

Este archivo es muy extenso, por lo que se pondrán y comentarán sólo algunas lineas.

values y values-es:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      .
      .
      .
    <string name="txtAceptarOferta">Aceptada</string>
    <string name="txtRechazarOferta">Rechazada</string>
    <string name="txtPendienteOferta"> Pendiente</string>
    <string name="txtAceptadaLaOferta">Aceptada la oferta: </string>
    <string name="txtRechazadaLaOferta">Rechazada la oferta: </string>
</resources>
```

values-en-rUS:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      .
      .
      .
    <string name="txtAceptarOferta">Accepted</string>
    <string name="txtRechazarOferta"> Rejected</string>
    <string name="txtPendienteOferta"> Pending</string>
    <string name="txtAceptadaLaOferta">Accepted the offer: </string>
    <string name="txtRechazadaLaOferta">Rejected the offer: </string>

</resources>
```

Estas lineas se usan para mostrar el mensaje de aceptada y rechazada, y para poner encima de la imagen correspondiente como queda cada oferta, si utilizamos el lenguaje castellano utilizará los textos que se muestran en primer lugar, y si el idioma es ingles de Estados Unidos, utilizará los textos del segundo.

Xml

preferencias.xml

Es el xml donde se guardarán las preferencias, podemos ver que se han creado sólo dos claves que son las de username y password.

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <EditTextPreference
        android:key="username"
        android:summary="@string/resumenUsuario"
        android:title="@string/usuario" />
    <EditTextPreference
        android:key="password"
        android:summary="@string/resumenContrasena"
        android:title="@string/contrasena"
        android:inputType="textPassword"/>

</PreferenceScreen>
```

8. - Manuales

8.1. - Manual de usuario aplicación web

Índice

1-Introducción

La aplicación de Banco de Tiempo, permite que los usuarios de una comunidad universitaria servicios, dando valor no sólo a las bienes materiales si no a la ayuda mutua.

2-Registrarse

Para poder acceder a la aplicación debe garantizarse que el usuario es un usuario de la universidad, por lo que es necesario que instroduzca una dirección de correo de la universidad.

Si no desea ser un usuario registrado en el sistema y sólo quiere consultar los servicios que existen y hacer alguna propuesta, puede solicitar una contraseña temporal pulsando sobre "pide tu contraseña temporal", que mostrará un campo para que introduzca el correo (de la universidad) al que se le enviará la contraseña generada.

Si decide que quiere registrarse en el sistema y tener acceso a todos los servicios que ofrece,

pulse sobre “Registrate” y cargará un formulario de registro para que lo rellene.

Una vez que haya rellenado todos los campos , se le enviará una contraseña al correo con la que pueda acceder al sistema y después desde la configuración de su perfil cambiar la contraseña por la que usted desee.

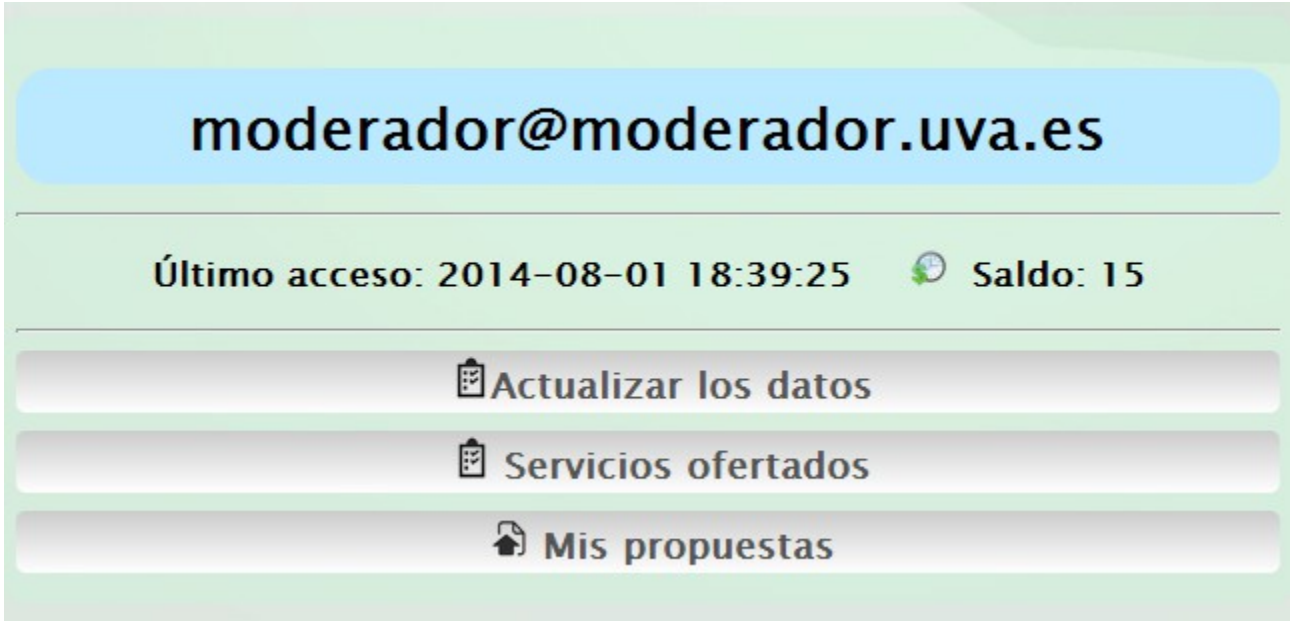
3-Página Principal

La página principal muestra el menú de navegación dependiendo del tipo de usuario que seas, en la imagen se muestran en el caso de que el usuario tenga permisos de moderación, ya que tendría todas las opciones. En caso de que el usuario sea un usuario temporal, sólo dispondría de las opciones “*Principal”* y *“Propuesta”*. Si fuera un usuario registrado tendría todas excepto la de moderación.

La página principal nos muestra el buscador y el listado de propuestas pendientes y servicios que hay en la aplicación. Estas características se verán mas adelante en sus respectivos apartados.

4- Perfil

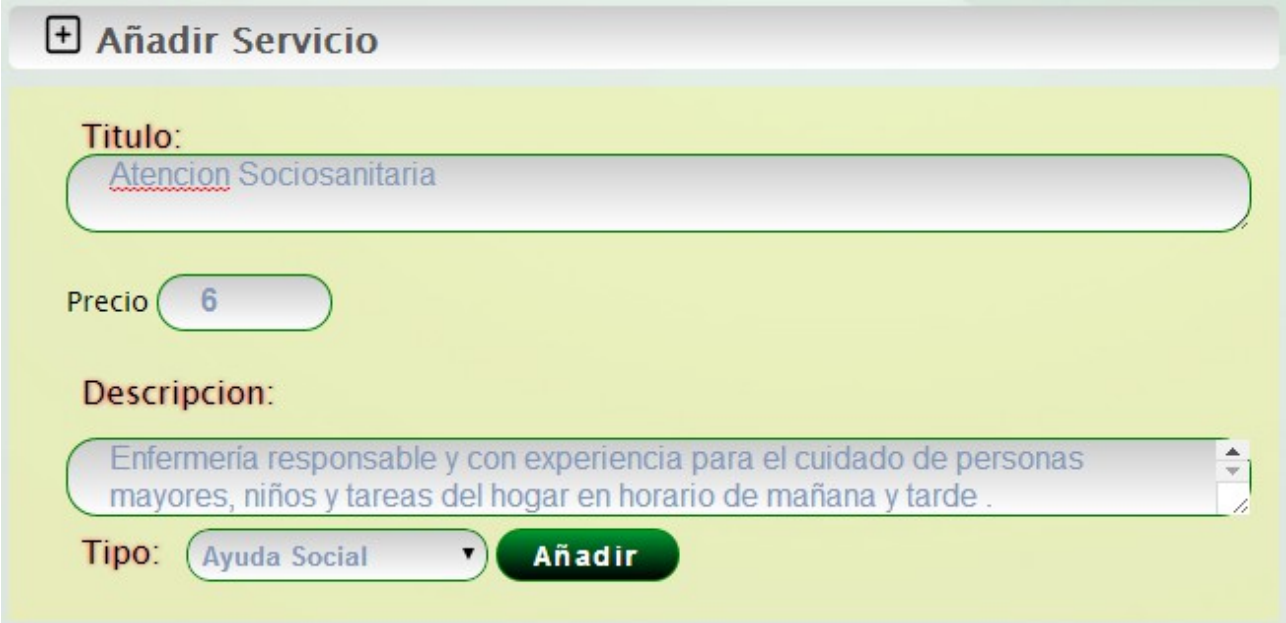
Desde su página de perfil, podrá actualizar sus datos, consultar su saldo, consultar los servicios ofertados por usted y el estado de las propuestas que ha realizado. Es similar a las opciones que se observan en las pestañas de "*servicios*" y "*propuestas*".



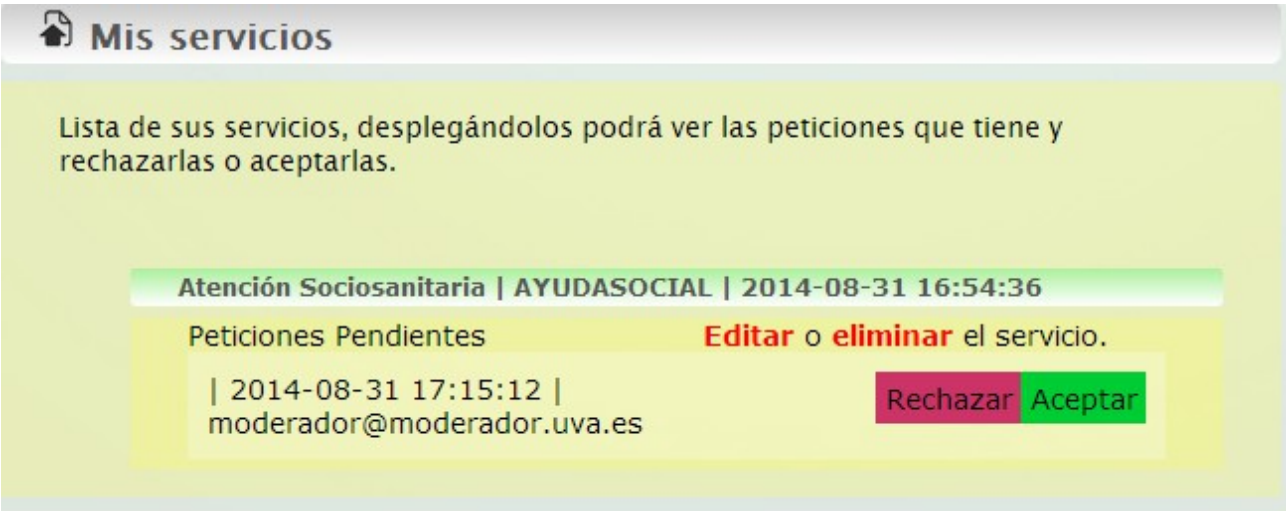
Desde la opción de actualizar los datos, usted podrá cambiar sus datos de usuario a los que tiene acceso.

5- Servicio

Puede añadir un servicio rellenando los campos del formulario que se despliega al pulsar sobre *"añadir servicio"*.



Desplegando las lisa de sus servicios se mostrarán las peticiones que tiene ese servicio y usted podrá rechazarlas o aceptarlas.

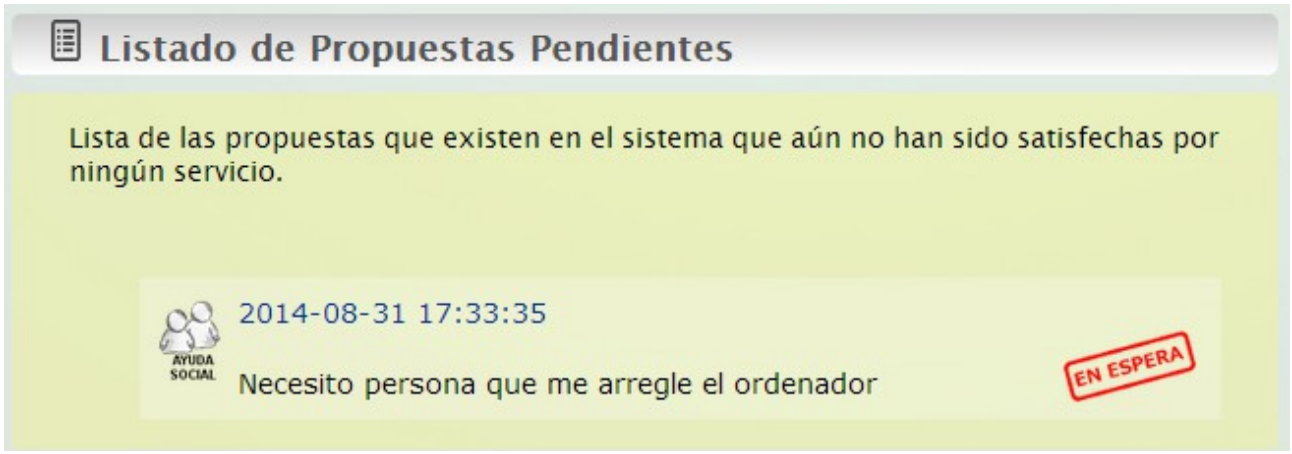


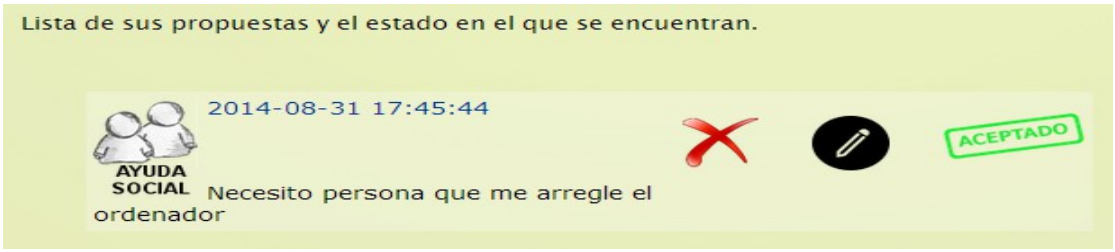
También puede ver las opciones a realizar sobre el servicio, como son editar o eliminar servicio, si pulsa sobre editar se desplegará un formulario para editar su servicio, y si pulsa sobre eliminar se pedirá confirmación para eliminar el servicio.

6- Propuesta

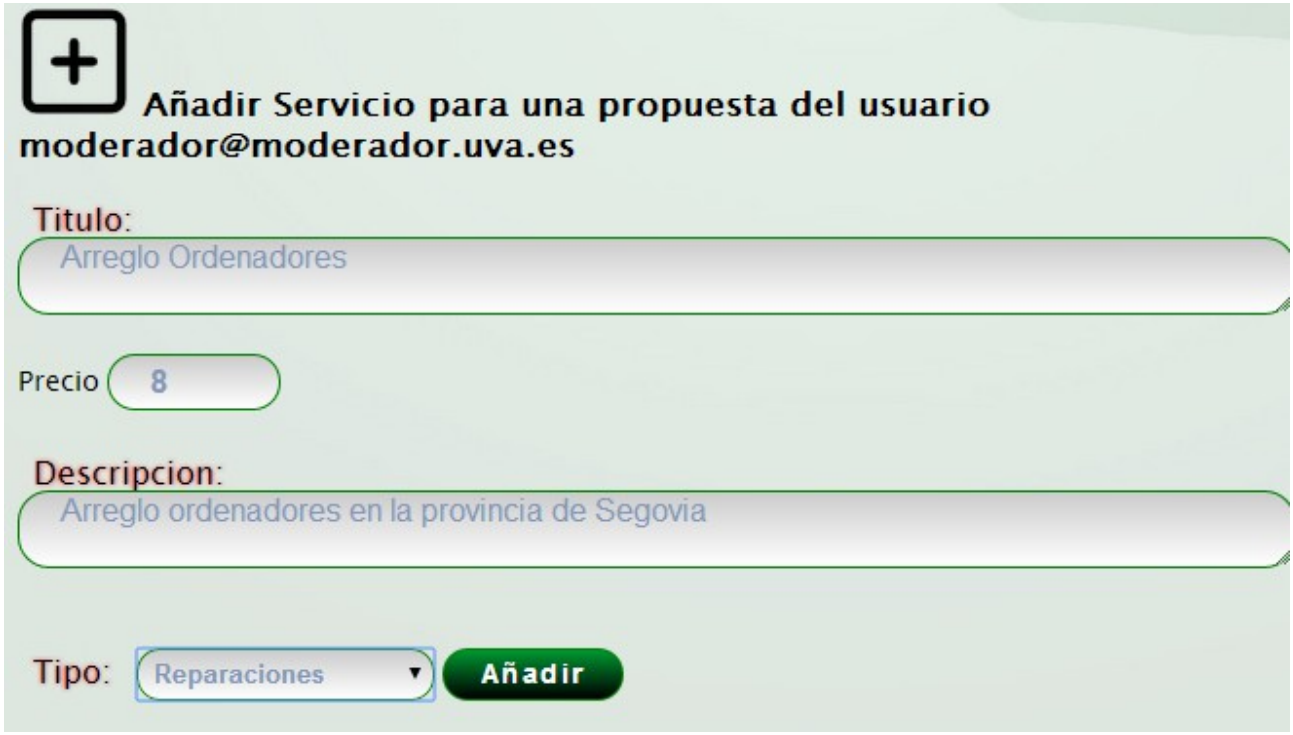
Puede añadir una propuesta rellenando los campos del formulario que se despliega al pulsar sobre *"añadir propuesta"*.

La propuesta quedará en estado pendiente, y se mostrará en el listado de propuestas pendientes con el icono de *"en espera"* mientras no se haya creado un servicio que la satisfaga, en el momento que alguien cree un servicio que la satisfaga el icono cambiará a "*aceptada"* y haciendo click sobre él veremos el detalle del servicio.





Si somos un usuario registrado que quiere satisfacer una propuesta, podemos hacerlo haciendo click sobre el icono de "en espera" y rellenando los campos para crear un nuevo servicio.

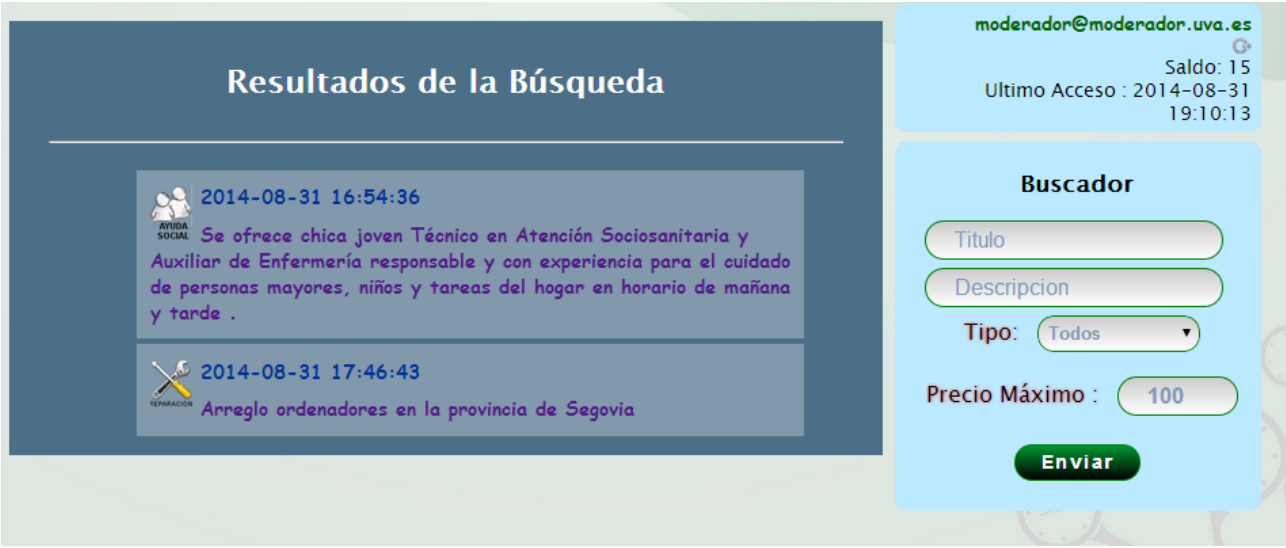


Ese servicio pasará a formar parte de la aplicación y cualquier usuario puede hacer peticiones sobre él.

Para eliminar una propuesta simplemente hay que pulsar sobre el icono de eliminar, y para editar sobre el icono de edición, que desplegará un formulario para realizar los cambios.

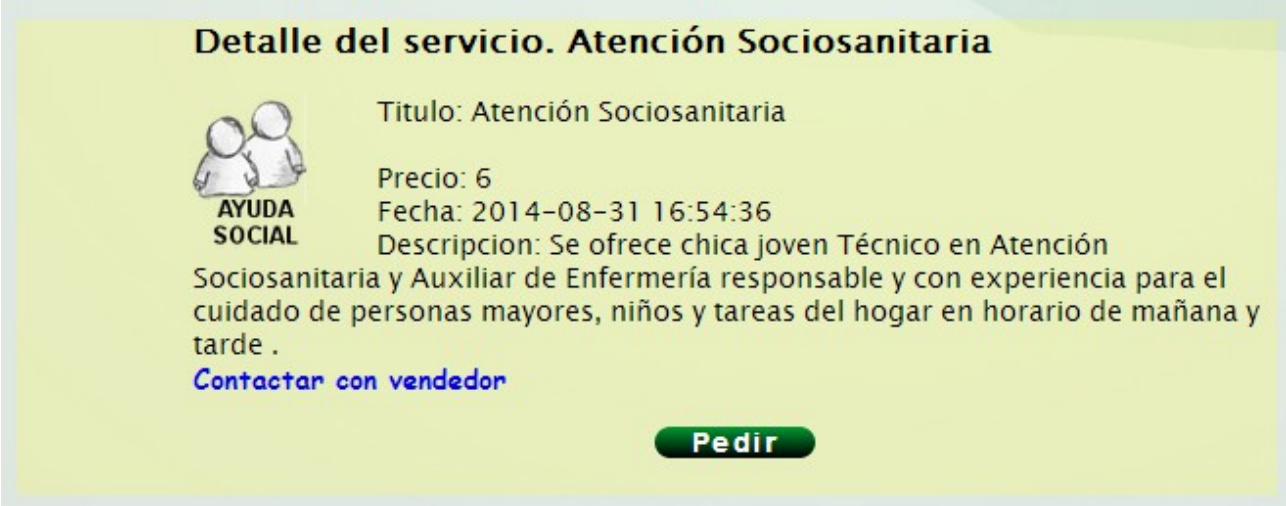
7- Buscar y pedir servicio

Podemos utilizar cualquiera de los dos buscadores que existen para buscar servicios con características concretas, podemos buscar por título, descripción, por tipo de servicio y por precio máximo, o sólo con alguna de las características según se desee.



Para poder pedir un servicio debemos acceder a la vista de detalle de el servicio que queremos solicitar.

La manera de acceder a la vista detalle es pulsando sobre un servicio, ya sea desde la lista de servicios o desde los resultados de una búsqueda.



8- Consumición

En la pestaña de consumo podemos ver el historial de todas las consumiciones que nos afectan, las pendientes que aún no nos han confirmado, las que hemos vendido nosotros, las que hemos comprado y las que nos han rechazado.

Podemos ver la fecha en la que se realizó la consumición, y podremos ver si el servicio aún sigue activo o ha sido retirado, también nos da la opción de mandar un Email al comprador o vendedor dependiendo del caso.

9- Moderar

Las personas con permisos de moderación podrán realizar tres acciones de moderación fundamentalmente, moderar usuario, descargar los datos de las tablas y regenerar contraseñas.

Moderar usuario: El moderador introduce el correo electrónico del usuario que desea moderar y pulsa en enviar, que cargará la página de moderación de usuario.

En el que podrá cambiar algunos datos sobre él, como el tipo de miembro, el estado de la cuenta, hacerle moderador, resetear sus intentos o modificar su saldo.

Con la opción de descargar tabla, descargará todos los datos de la tabla seleccionada para poder hacer consultas y poder tratar los datos en su ordenador.

La opción de regenerar contraseñas, genera contraseñas para todos los usuario que no son moderadores de la aplicación. Ésto puede usarse para comprobar que los usuarios del banco de tiempo siguen siendo miembros de la universidad.

8.2. - Manual de usuario aplicación Android

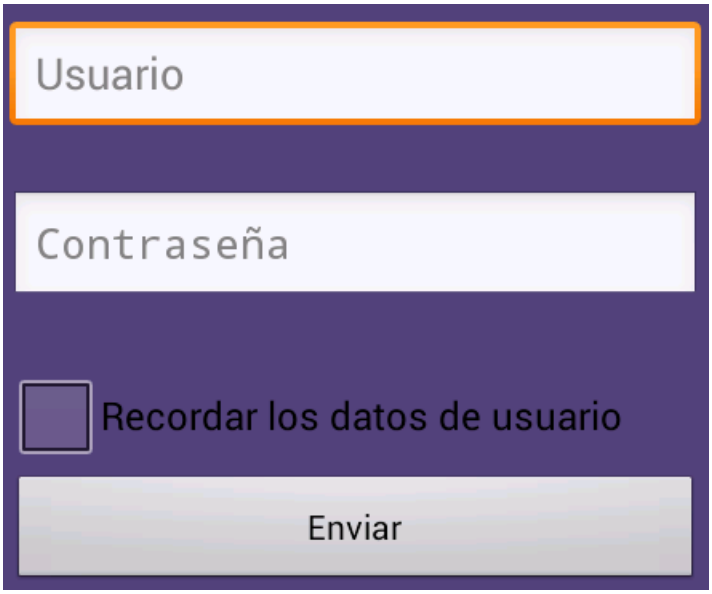
Índice

1.- Introducción

TimeBank es una aplicación android multilenguaje, que complementa la usabilidad de "El Banco de Tiempo" , que te permitirá aceptar y rechazar las peticiones sobre tus servicios, de forma rápida y simple.

2.- Acceso / Salida

Acceso:



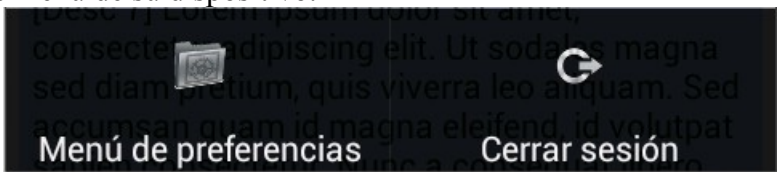
La primer vez accedas al sistema simplemente tienes que introducir tus datos de acceso en los campos habilitados para ello.

Si seleccionas “Recordar los datos de usuario” tus datos se guardarán y no se volverán a pedir a no ser que pulses la opción de “Cerrar sesión” , con lo que la siguiente vez que acceda al sistema, éste le pedirá de nuevo sus credenciales.

Salida:

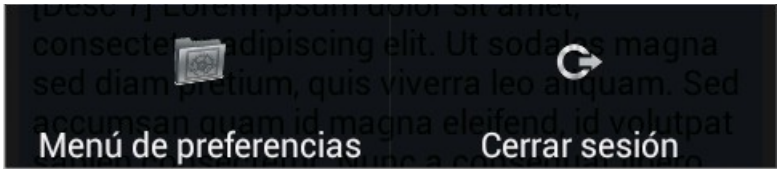
Para salir, simplemente cierre la aplicación con el teléfono.

Si además quiere eliminar su usuario, puede pulsar el botón de cerrar sesión que se muestra al pulsar el botón de menú de su dispositivo.



3.- Modificación de usuario

Si desea modificar el usuario o la contraseña introducida, puede pulsar el botón de preferencias que se muestra al pulsar el botón de menú de su dispositivo.



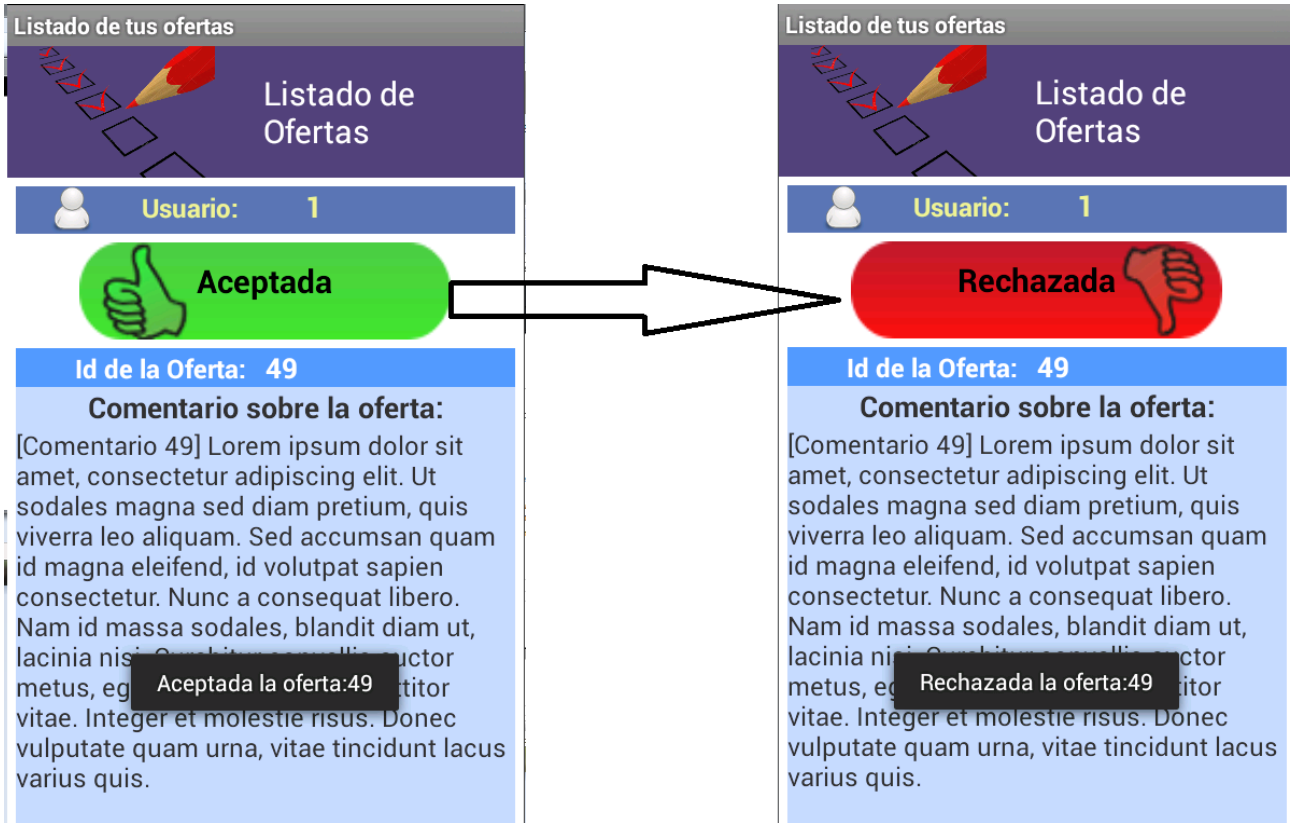
4.- Selección de un servicio

Una vez que se ha mostrado la lista de todos sus servicios, puede seleccionar cualquiera pulsando sobre él.

Después se le mostrará todas las peticiones que se han realizado a su servicio.

5.- Aceptar / Rechazar ofertas

Para ver el estado de cada oferta, simplemente pulse sobre la oferta y se mostrará el estado actual de esa oferta.



Para aceptar una oferta pulse sobre la oferta, y después pulse sobre el botón hasta que éste cambie al estado "Aceptada". Se le avisará con un mensaje emergente.

Para rechazar una oferta pulse sobre la oferta, y después pulse sobre el botón hasta que éste cambie al estado "Rechazada". Se le avisará con un mensaje emergente.

6.- F.A.Q

P- Aparece un mensaje de pendiente en el botón de las ofertas.
R- Quiere decir que esa oferta no ha sido atendida aún, y no se ha decidido si se aceptará o rechazará

P- Quiero volver al estado pendiente.
R- No es posible, en el momento que decides rechazar una oferta o aceptarla, significa que ya has decidido tomar una decisión sobre ella.

8.3. - Manual de instalación aplicación web

Índice

1.- Introducción

La aplicación web de Banco de Tiempo permite a los miembros de una comunidad universitaria intercambiar servicios.

A continuación se explicará el proceso para el despliegue de la aplicación en un servidor.

Para el correcto funcionamiento de la aplicación el servidor debe soportar PHP 5.4, y la Base de datos una versión de MySQL no inferior a 5.6.10

2.- Instalación Base de Datos

Para instalar la Base de Datos simplemente debemos ejecutar el Script SQL sobre la base de datos, éste automáticamente generará las tablas y toda la estructura necesaria.

Después de crear la estructura de la base de datos, debe crearse un usuario que tenga acceso a modificar las tablas, éste se configurará mas adelante en la aplicación.

Por defecto se crea un usuario con permisos de moderador que deberá ser sustituido por otro a través de una sentencias SQL si el administrador de la base de datos lo considera a oportuno, o a través de el panel de administración de la aplicación una vez desplegada.

USUARIO: moderador@moderador.uva.es
PASS: moderador

Una vez creada la base de datos, la estructura de la tabla y el usuario de la base de datos para que pueda acceder a ella, debe configurarse en el archivo **/modelo/bd.php , en la variable $link de la función conectar,** los parámetros de la base de datos que hemos creado.

```
$link = mysqli_connect("servidor", "usuario","contraseña", "base_de_datos");
```

3.- Instalación del servidor

Para instalar la aplicación simplemente deben copiarse todos los archivos y carpetas en la carpeta del servidor configurada para ser accesible desde internet. Y después configurar los parámetros de la base de datos (véase punto 2).

9. - Conclusiones

Con este trabajo de fin de grado, se quería conseguir la implementación de una aplicación web con un componente social, no buscando el enriquecimiento de personas o uso del dinero, si no como se ha explicado en los objetivos y la motivación, potenciar la ayuda entre las personas y valorar las cosas no sólo por su valor material si no por la satisfacción que genera el poder ayudar a otros.

Se ha querido restringir el uso a una comunidad universitaria, ya que es un ámbito en el que todos los miembros permanecen en el mismo lugar durante largo periodo de tiempo, pudiendo fomentar además las relaciones entre todos los estratos de la comunidad.

Durante la elaboración del trabajo he podido demostrar mis conocimientos en lo que se refiere a la ingeniería de software, la programación orientada a objetos, el uso del patrón modelo-vista-controlador y una ligera incursión en la programación en el lenguaje android para dispositivos móviles.

He ampliado mis conocimientos sobre el lenguaje PHP y me ha sorprendido el potencial de los frameworks, concretamente jQuery, que hace que las aplicaciones web tengan un comportamiento muy agradable para el usuario y abre un mundo enorme de posibilidades a la hora de programar y tratar eventos que utilizando simplemente Javascript sería prácticamente impensable.

La aplicación se ha pensado para su posible implementación en cualquier servidor, por lo que aunque está pensado para una comunidad universitaria, no habría ningún problema en poderse implementar en comunidades de vecinos, urbanizaciones o barrios.

Una de mis grandes inquietudes es la seguridad y las bases de datos, pero por diversas circunstancias no he podido implementar y desarrollar toda la parte de seguridad que me hubiera gustado. Pienso que los datos por insignificantes que parezcan pueden ser de gran valor en manos (in)adecuadas.

Después de todo el trabajo realizado estoy satisfecho con el resultado obtenido, tanto en la aplicación web como en la aplicación móvil.

10. - Propuestas para futuras ampliaciones

Existen varias ampliaciones que se pueden aplicar tanto a la aplicación web como a la aplicación Android.

Referente a las aplicaciones.

10.1. - Aumentar la funcionalidad de la aplicación móvil

La principal ampliación se basa en crear la aplicación funcional conectando las bases de datos entre la aplicación móvil y la aplicación web.

Ampliar los dispositivos que pudieran utilizar la aplicación como iPhone y Windows Phone, actualmente sólo puede ser ejecutada en dispositivos móviles que utilicen Android, que aunque son muy numerosos no abarcan todo el mercado.

Ampliar las opciones que se pueden realizar desde la aplicación móvil, como la creación de servicios, petición de servicios y acercarse a la funcionalidad completa que ofrece la version web.

10.2. - Crear un sistema de puntuación

Crear un sistema de puntuación y de comentarios sobre las consumiciones y transacciones que se realizan similar al implementado en eBay y otras plataformas de compra y venta.

De esta manera se conseguiría mayor interacción con el usuario y la calidad de la aplicación crecería considerablemente.

10.3. - Crear un sistema de comunicación interno y la posibilidad de abrir disputas

La posibilidad de poder comunicarse dentro de la aplicación con los usuarios que existen en ella, haría que los usuarios permanecieran mas tiempo en la página creando una plataforma de uso diario y con una imagen totalmente independiente y autosuficiente.

La posibilidad de abrir disputas conseguiría que la aplicación fuera mas seria y creíble, dando mayor grado de confiabilidad para los usuario que deciden usarla. Sabiendo que a través de las disputas no perderán su saldo.

En el apartado de seguridad sería conveniente mejorar y tener en consideración las siguientes prácticas.

10.4. - Keylogger

Para evitar que aplicaciones puedan leer las pulsaciones de teclas en el momento de acceder al sistema, sería conveniente usar un keylogger, que permita la introducción de las contraseñas de maneras diferentes como usando el ratón en un teclado que se moviese por la pantalla.

10.5 - XSS. Cross Site Scripting

El *XSS* permite a inyectar en páginas web vistas por el usuario, código JavaScript o cualquier otro lenguaje script similar.

Para evitar este tipo de ataques, se procederá al uso de dos funciones que nos permiten convertir las entradas enviadas, para que éstas no puedan ser ejecutadas.

Una de ellas viene implementada con *PHP* 5, como es *htmlentities()* que convierte el código insertado (<) a su correspondiente código html (<), por lo que no será ejecutado como se ejecuta el html.

10.6. - SQL Injection - Aplicación web

Para evitar el SQL Injection en la aplicación web debe crearse una función que preprocese cualquier cadena que se inserte en la página, evitando que se puedan ejecutar sentencias SQL a la hora de realizar una consulta.

10.7. - Logs

Creando un sistema de logs a través de triggers de todas las acciones que se realicen el sistema, por lo que es ejecutado desde la base de datos y no desde el código. Aunque se creara una aplicación espejo en la que se intentara saltarse el registro de las acciones, no sé podría ya que está implementado sobre la base de datos y no sobre la aplicación web.

En el log, se registrará la fecha y hora de la acción, la consulta ejecutada, el usuario que ejecuta la acción, y el tipo de acción ejecutada. Si se diera el caso de ejecutar una acción sin estar registrado en el sistema, aparecería el campo de usuario vacío, entonces el administrador podría ver que existe un fallo y qué operaciones ha realizado.

El acceso a la tabla de logs, será únicamente por por el usuario *"moderador_web"* de la base de datos (diferente del usuario de la aplicación) y únicamente podrá ejecutar acciones de consulta (SELECT) , por lo que las tablas que contienen los logs podrán ser solamente consultadas evitando de esta forma la modificación o borrado de la tabla para eliminar huellas.

10.8. - Encriptación SSL

La encriptación SSL permite que los mensajes enviados entre el servidor y el cliente se mantengan cifrados a través de la clave pública que el servidor ofrece.

Esta característica debe configurarse en el servidor que ofrece el servicio, en este caso se aplicará sobre el servidor apache instalado en el servidor.

11. - Glosario

Término	Definición
AJAX	Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.
Spinbox	Caja de introducción de valores numérico con botones para incrementar el valor que se introduce.
SHA-2	SHA-2 es un conjunto de funciones hash criptográficas (SHA-224, SHA-256, SHA-384, SHA-512) diseñadas por la Agencia de Seguridad Nacional (NSA) y publicada en 2001 por el Instituto Nacional de Estándares y Tecnología (NIST) como un Estándar Federal de Procesamiento de la Información (FIPS).
MD5	MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts)
SQL	Lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ellas.
SQL Injection	Método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar consultas a una base de datos.
PHP	Lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.
Firewall	Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios.
MySQL	MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario.
MySQLi	La extensión MySQLi (MySQL Mejorado) es un driver de base de datos relacional utilizado en el lenguaje de programación PHP para proporcionar una interfaz las bases de datos MySQL.
jQuery	Es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

12. - Bibliografía

Bibliografía

1. Hacking de aplicaciones web: SQL Injection. Enrique Rando, Chema Alonso. Informática 64. 2012.
2. Comparing Hash Algorithms: Md5, Sha1 or Sha2? : http://www.not-implemented.com/comparing-hash-algorithms-md5-sha1-sha2/
3. jQuery API Documentation: http://api.jquery.com/
4. Android. Guía para desarrolladores. W. Frank Ableson, Robi Sen, Chris King. Anaya
5. Ingeniería del Software. Ian Sommerville. Pearson
6. Aprenda Programación Orientada a Objetos (POO) en PHP. Stefan Mischook . 2007
7. Stackoverflow : http://stackoverflow.com/