|  |
| --- |
| SRS  LOGROLLING |
| Alberto Almagro  Rubén Gómez  Juan Carlos Llamas  Jaime Martínez  Santiago Mourenza  Pedro Palacios  Adrián Sanjuán  Pablo Torre |





Índice

[1. Introducción 5](#_Toc26043048)

[1.1 Propósito del plan 5](#_Toc26043049)

[1.2 Definiciones 5](#_Toc26043050)

[1.3 Organización del documento 6](#_Toc26043051)

[2. Descripción general 7](#_Toc26043052)

[2.1 Perspectiva del producto 7](#_Toc26043053)

[2.2 Funciones del producto 8](#_Toc26043054)

[2.2.1 Usuarios 8](#_Toc26043055)

[2.2.2 Favores 9](#_Toc26043056)

[2.2.3 Premios y compras 10](#_Toc26043057)

[2.2.4 Buscador y geolocalizador 10](#_Toc26043058)

[2.3 Características del usuario 11](#_Toc26043059)

[2.4 Restricciones 11](#_Toc26043060)

[2.4.1 Restricciones legales 11](#_Toc26043061)

[2.4.2 Restricciones técnicas 11](#_Toc26043062)

[2.4.3 Restricciones de proyecto 12](#_Toc26043063)

[2.5 Supuestos y dependencias 12](#_Toc26043064)

[2.6 Requisitos futuros 12](#_Toc26043065)

[3. Requisitos específicos 13](#_Toc26043066)

[3.1 Interfaces externos 13](#_Toc26043067)

[3.1.1 Interfaces de usuario 13](#_Toc26043068)

[3.1.2 Interfaces Hardware 21](#_Toc26043069)

[3.1.3 Interfaces Software 21](#_Toc26043070)

[3.1.4 Interfaces de comunicación 21](#_Toc26043071)

[3.2 Requisitos funcionales 22](#_Toc26043072)

[3.2.1 Clases 22](#_Toc26043073)

[3.2.1.1 Subsistema de gestión de usuarios: 22](#_Toc26043074)

[3.2.1.2 Subsistema de favores 27](#_Toc26043075)

[3.2.1.3 Subsistema de premios y compras 28](#_Toc26043076)

[3.2.1.4 Subsistema de búsqueda y geolocalización 29](#_Toc26043077)

[3.2.2 Casos de uso: 31](#_Toc26043078)

[3.3 Requisitos de rendimiento: 64](#_Toc26043140)

[3.4 Requisitos lógicos de la base de datos: 64](#_Toc26043141)

[3.5 Restricciones de diseño 64](#_Toc26043142)

[3.6 Atributos del sistema software 65](#_Toc26043143)

# Introducción

## Propósito del plan

El principal objetivo de nuestra aplicación es que nuestros usuarios decidan en qué quieren **gastar su tiempo** ya que, como bien es sabido, **“el tiempo es oro”.** Logrolling es una aplicación de **intercambio de favores** en la que un usuario puede pedir y realizar favores.

Si el usuario quiere realizar un favor, la aplicación le proporcionará una lista ordenada por filtros a elección del usuario. Si lo que quiere el usuario es pedir un favor, este podrá subirlo a la lista de favores. De esa manera, las personas podrán ayudarse mutuamente en la medida del tiempo que tenga cada uno. La aplicación también incluye un sistema de geolocalización para buscar la mejor ruta desde la posición actual hasta el destino del pedido.

La aplicación utilizará una **moneda virtual**, denominada **grollies**, que irán aumentando y disminuyendo a través de varios factores. Para pedir un favor se **reducirá** el número de grollies del usuario y cuando se realiza un favor satisfactoriamente se **aumentará** en función de la recompensa ofrecida por el que pidió el favor. Además, estos grollies se podrán **intercambiar** por distintos **premios** dentro de la aplicación.

## Definiciones

|  |  |
| --- | --- |
| USUARIO | Cualquier individuo que utilice la aplicación. Hay varios tipos de usuario, entre ellos está el administrador y el usuario común. Además el usuario común puede ser demandante o realizador de favores |
| ADMINISTRADOR | Usuarios encargados de la gestión y supervisión de la aplicación |
| GROLLIES | Es la moneda virtual que utiliza la aplicación. Se ganarán realizando favores y se podrán cambiar por regalos. |

## Organización del documento

El documento ofrece en primer lugar una breve introducción a Logrolling, sus objetivos y a qué público está destinada. También explicamos las **restricciones del desarrollo** de la aplicación y los **requisitos del futuro**.

Después de esta breve introducción, se desarrollan los **requerimientos técnicos** del software de la aplicación. Aquí se concreta el diseño y la funcionalidad de las **interfaces** de usuario y se aclara la necesidad de software externo.

Por último, definimos la funcionalidad del software a través de **casos de uso y clases**. En esta sección podremos ver el diagrama de casos de uso de la aplicación y la especificación de cada caso en particular. Además, incluimos también unos cuantos **diagramas de actividad** de los casos de uso más importantes.

# Descripción general

*Nuestra aplicación permite intercambiar favores entre los distintos usuarios a través de nuestra moneda virtual, los grollies. Además, tiene todas las funcionalidades que se pueden esperar de una aplicación moderna y actual.*

## Perspectiva del producto

Nuestra aplicación Logrolling pretende **solucionar** una buena parte de los problemas cotidianos de la gran mayoría de la gente, en especial, aquellos relativos a la **falta de tiempo** para realizar determinadas tareas.

Mediante un sistema de **intercambio de favores** pretendemos estrechar lazos entre los miembros de una comunidad y recompensar a aquellas personas que se presten a ayudar a los demás. Nuestros usuarios podrán demandar o realizar favores a cambio de nuestra moneda de pago, los **grollies,** y obtener recompensas en forma de premios una vez alcanzada una determinada cantidad de los mismos.

Nuestra aplicación también permite a cualquier empresa que lo desee promocionarse mediante un sistema por el cuál ofrecerá recompensas a cambio de información útil como, por ejemplo, atención al cliente, seguimiento de directrices de la empresa o conocimiento de los empleados de los productos que venden.

## Funciones del producto

Nuestra aplicación se divide en **4 subsistemas** principales, a saber, Usuario, Favores, Premios y compras, y Buscador y geolocalizador.

Subsistema Favores

Subsistema Usuarios

Subsistema Premios y compras

Subsistema Buscador y geolocalizador

## Usuarios

Los usuarios pueden ser de dos tipos distintos. Distinguimos entre **Empresas** y **Personas**. En esta sección nos referiremos a los usuarios-persona como usuarios y los usuarios-empresa como empresa.





Empresas

Usuarios

Las funciones que pueden realizar ambos dentro de la app son ligeramente distintos. La principal diferencia es que las empresas solo pueden demandar favores y no realizarlos. Independientemente, eso queda para la sección de favores. Las funciones comunes de ambos tipos de usuarios son:

|  |  |
| --- | --- |
| Función | Descripción |
| Registro | En un primer instante se le dará la opción al usuario de registrarse en la aplicación. |
| Log-in y log-out | El usuario podrá iniciar y cerrar sesión cuando quiera (aunque esta se mantendrá abierta por defecto) |
| Cambiar Foto de perfil | El usuario pude modificar su foto de perfil |
| Cambiar e-mail | El usuario pude modificar la dirección de correo electrónico de contacto del usuario con la aplicación |
| Cambiar Idioma | El usuario puede modificar el idioma |
| Cambiar contraseña | El usuario podrá cambiar su contraseña o solicitar que le sea recordada |
| Leer normas de uso | El usuario tendrá acceso en todo momento a las normas de uso de la aplicación |
| Conectarse con Facebook | El usuario podrá conectar su cuenta en la app con su cuenta de Facebook |

## Favores

Los usuarios pueden demandar o realizar favores según sus necesidades o disponibilidad. Sus funciones son:

|  |  |
| --- | --- |
| Función | Descripción |
| Publicar un favor/favor múltiple | Un usuario podrá publicar un favor (una necesidad) ofreciendo una retribución (en grollies) estandarizada con el objetivo de que otro usuario lo realice. |
| Ofrecer un favor/favor múltiple | Un usuario puede ofrecer un favor esperando una compensación concreta (en grollies). A la hora de realizar un recado un usuario podrá establecer la opción de que no solo uno, sino varios, soliciten su ayuda. |
| Realizar favor | Un usuario puede adjudicarse un favor con la motivación de obtener una retribución (en grollies) tras su realización. |
| Atribuirse favor | Un usuario que necesite un favor ya ofrecido podrá reservarlo, para llevarlo a cabo. |
| Chat | Los usuarios se podrán comunicar a través de un chat privado para concretar detalles del favor a realizar y la forma. |
| Demandar favor (empresa) | Una empresa puede solicitar un favor a cambio de una compensación en grollies al igual que el resto de usuarios pero no puede realizar ningún favor |
| Poner tiempo de entrega | El usuario que demanda un favor puede establecer un límite de tiempo en el que quiere que se le haga el favor |
| Negociar Grollies | Cualquier usuario puede negociar el precio de un favor con la persona o empresa que lo demanda |

## Premios y compras

Los usuarios pueden adquirir paquetes de moneda virtual o intercambiarlos por premios según las siguientes funciones:

|  |  |
| --- | --- |
| Función | Descripción |
| Compra de grollies | Los usuarios podrán ingresar dinero en la aplicación para recibir grollies a cambio. |
| Intercambiar grollies por premios | Si el usuario tiene suficientes grollies podrá intercambiarlos por premios de Amazon. |

Ç

## Buscador y geolocalizador

Los usuarios podrán filtrar la búsqueda de favores de dos maneras:

|  |  |
| --- | --- |
| Función | Descripción |
| Buscar favores | Un usuario puede buscar favores por su nombre en un buscador de la app |
| Buscar favores cerca | Un usuario puede buscar favores filtrando según la distancia a su localización o a otra que selecciones |

## Características del usuario

Nuestro proyecto está orientado a la obtención de diferentes recompensas por la realización de distintos favores publicados por otros usuarios. Por ello la aplicación está destinada a un **público general**, en especial aquellas personas que puedan ofrecer parte de su tiempo para ayudar a otras personas, y a aquellas que necesitadas de tiempo por el exceso de actividades en su día a día.

En un primer momento pude ser muy beneficioso desarrollarla en un **ámbito universitario** donde abundan ambos tipos de usuarios prototípicos.

El manejo de la aplicación requerirá **nociones básicas** de uso de dispositivos electrónicos, pero, además, se desarrollará usando interfaces sencillas que no sean ningún obstáculo para cualquier tipo de colectivo no acostumbrado al uso de la tecnología.

## Restricciones

## Restricciones legales

Tras una consulta de carácter jurídico, supimos que es ilegal desarrollar una aplicación que acepte pagos a cambio de una moneda virtual con la intención de posteriormente retirar esta moneda virtual como dinero real. Es por ello por lo que ofrecemos la posibilidad de reembolsar el dinero invertido mediante **premios y regalos**.

## Restricciones técnicas

* Nuestra aplicación está destinada a usuarios **Android**, aunque no descartamos una versión iOS y otra de escritorio para el futuro.
* La aplicación debe estar programada en **Java**
* La **base de datos** debe soportar el acceso de una cantidad considerable de dispositivos que accedan a la aplicación de manera simultánea.

## Restricciones de proyecto

Nuestra aplicación deberá ser accesible para cualquier persona por lo que debe tener una **interfaz intuitiva**.

## Supuestos y dependencias

Algunos factores que pueden afectar los requerimientos del sistema son entre otros: agregar nuevas funcionalidades a las ya existentes o la utilización de un lenguaje de programación distinto a Java para la realización de los futuros requerimientos.

El usuario deberá tener un dispositivo con sistema operativo de **Android Lollipop 5.0** o posterior con los **servicios de Google**. También se va a necesitar conexión con internet y que el usuario tenga instalada una versión suficientemente actualizada de Google Maps.

## Requisitos futuros

En un futuro se nos podría demandar la ampliación de las funcionalidades de la aplicación, pero más a corto plazo sería preferible centrarse en poder lanzar la aplicación para **otros sistemas operativos**.

# Requisitos específicos

## Interfaces externos

## Interfaces de usuario

*En esta sección se especifican y muestran las diferentes pantallas que componen la aplicación. No se incluyen todas las pantallas, sino solamente las pantallas iniciales, las principales y algunas con especial relevancia.*

|  |
| --- |
| Pantalla de Carga |
| Descripción: |
| Pantalla de carga inicial en la que la aplicación se intenta conectar con el servidor. |
| Precondiciones: |
| El usuario ha abierto la aplicación. |
| Postcondiciones: |
| Éxito: El usuario es dirigido a la pantalla de inicio de sesión.  Fallo: Se muestra un mensaje de error y un botón para reintentar establecer la comunicación con el servidor. |
| Acciones a realizar: |
| Ninguna |

Solo se incluyen pantallas relacionadas con los usuarios comunes, siendo las pantallas para las empresas prácticamente idénticas con algunos matices no relevantes.

|  |
| --- |
| Pantalla de inicio de sesión |
| Descripción: |
| Pantalla de en la que el usuario ha de identificarse para poder acceder a todas las funcionalidades. |
| Precondiciones: |
| La aplicación se ha conectado con el servidor y ha detectado que no hay ningún usuario identificado. |
| Postcondiciones: |
| Éxito: El usuario es dirigido a la pantalla principal o a la pantalla de registro.  Fallo: Se muestra un mensaje de error y se permanece en esta pantalla. |
| Acciones a realizar: |
| 1: El usuario introduce los datos con los que se registró anteriormente.  2: El usuario accede a la pantalla de registro mediante el botón “Registrarse”. |

|  |
| --- |
| Pantalla de registro |
| Descripción: |
| Pantalla en la que los usuarios pueden crearse una nueva cuenta con la que poder iniciar sesión. |
| Precondiciones: |
| El usuario ha pulsado el botón “Registrarse” en la pantalla de inicio de sesión. |
| Postcondiciones: |
| Éxito: El usuario crea una cuenta satisfactoriamente y es enviado a la pantalla de inicio de sesión.  Fallo: Se muestra un mensaje de error y se permanece en esta pantalla. |
| Acciones a realizar: |
| 1: El usuario puede introducir los datos que aparecen en la pantalla.  2: El usuario se identifica a sí mismo como una empresa y elige la opción correspondiente.  3: El usuario puede acceder a los términos y condiciones o a la política de privacidad pulsando los correspondientes iconos.  4: El usuario puede volver a la pantalla de inicio de sesión pulsando el botón “Iniciar sesión”.  5: El usuario puede pulsar el botón de “Registrarse” para crear un nuevo usuario. |

|  |
| --- |
| Pantalla principal “Favores” |
| Descripción: |
| Pantalla principal en la que el usuario puede ver los favores que otra gente haya pedido. |
| Precondiciones: |
| El usuario ha iniciado sesión o ha pulsado el botón “Favores” de la parte inferior en cualquier pantalla en la que este aparezca. |
| Postcondiciones: |
| El usuario accede a los detalles de un favor específico, filtra los favores, accede a otra pantalla principal o accede a la pantalla de compra de “grollies”. |
| Acciones a realizar: |
| 1: El usuario puede seleccionar un favor específico de entre toda la lista.  2: El usuario puede pulsar en cualquiera de los botones de la parte inferior de la pantalla para acceder a las otras pantallas principales.  3: El usuario puede acceder al filtro de favores, donde podrá seleccionar el orden en el que quiere que se le muestren los favores disponibles.  4: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

|  |
| --- |
| Pantalla principal “Mis Favores” |
| Descripción: |
| Pantalla principal en la que el usuario puede ver los favores que haya pedido y pedir más favores. |
| Precondiciones: |
| El usuario ha pulsado el botón “Mis Favores” de la parte inferior en cualquier pantalla en la que este aparezca o ha terminado de pedir un nuevo favor. |
| Postcondiciones: |
| El usuario accede a los detalles de uno de los favores que hay pedido, accede a la pantalla para pedir un nuevo favor, accede a otra pantalla principal o accede a la pantalla de compra de grollies. |
| Acciones a realizar: |
| 1: El usuario puede seleccionar un favor específico de entre toda la lista.  2: El usuario puede pulsar en cualquiera de los botones de la parte inferior de la pantalla para acceder a las otras pantallas principales.  3: El usuario puede pedir un nuevo favor mediante el botón “Pedir un nuevo favor”.  4: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

|  |
| --- |
| Pantalla principal “Mensajes” |
| Descripción: |
| Pantalla principal en la que el usuario puede ver las conversaciones que mantenga con otros usuarios y acceder a ellas. |
| Precondiciones: |
| El usuario ha pulsado el botón “Mensajes” de la parte inferior en cualquier pantalla en la que este aparezca o ha vuelto de una de las conversaciones. |
| Postcondiciones: |
| El usuario accede a una conversación en concreto, accede a otra pantalla principal o accede a la pantalla de compra de “grollies”. |
| Acciones a realizar: |
| 1: El usuario puede seleccionar una conversación concreta de entre toda la lista.  2: El usuario puede pulsar en cualquiera de los botones de la parte inferior de la pantalla para acceder a las otras pantallas principales.  3: El usuario puede buscar una conversación en concreto mediante el botón con forma de lupa de la parte superior derecha.  4: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

|  |
| --- |
| Pantalla principal “Regalos” |
| Descripción: |
| Pantalla principal en la que el usuario puede ver los regalos disponibles y canjear un número determinado de “grollies” por uno de ellos. |
| Precondiciones: |
| El usuario ha pulsado el botón “Regalos” de la parte inferior en cualquier pantalla en la que este aparezca o ha terminado de comprar un regalo. |
| Postcondiciones: |
| El usuario accede a los detalles de un regalo en concreto, a otra pantalla principal o a la pantalla de compra de “grollies”. |
| Acciones a realizar: |
| 1: El usuario puede seleccionar un regalo específico de entre toda la lista.  2: El usuario puede pulsar en cualquiera de los botones de la parte inferior de la pantalla para acceder a las otras pantallas principales.  3: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

|  |
| --- |
| Pantalla principal “Configuración” |
| Descripción: |
| Pantalla principal en la que el usuario puede acceder a su perfil, a los ajustes de notificaciones, a la pantalla de ayuda, a su link de invitación, cambiar el idioma o cerrar sesión. |
| Precondiciones: |
| El usuario ha pulsado el botón “Configuración” de la parte inferior en cualquier pantalla en la que este aparezca o ha vuelto de cualquier pantalla a la que se puede acceder desde esta. |
| Postcondiciones: |
| El usuario accede a una de las opciones detalladas en la descripción o a la pantalla de inicio de sesión si selecciona cerrar sesión. |
| Acciones a realizar: |
| 1: El usuario puede acceder a su perfil para editarlo.  2: El usuario puede modificar los ajustes de notificaciones de la aplicación.  4: El usuario puede acceder a la pantalla de ayuda.  5: El usuario puede invitar a un amigo o conocido mediante un enlace de descarga.  6: El usuario puede cambiar el idioma de la aplicación.  7: El usuario puede cerrar sesión, lo que le llevará a la pantalla de inicio de sesión.  8: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

|  |
| --- |
| Pantalla “Pedir nuevo favor” |
| Descripción: |
| Pantalla en la que el usuario puede pedir un nuevo favor especificando el nombre del favor, su descripción, lugar y la fecha límite para realizarlo y la recompensa, así como realizar especificaciones adicionales como adjuntar fotos o añadir el lugar donde se debe entregar el favor. |
| Precondiciones: |
| El usuario ha pulsado el botón “Pedir nuevo favor” que se encuentra en la pantalla “Mis favores”. |
| Postcondiciones: |
| El usuario pide un nuevo favor o vuelve a la pantalla “Mis favores”. También puede ser que el usuario no tenga suficientes “grollies” como para pedir el favor o no haya completado todos los campos imprescindibles, lo que mostrará un mensaje de error y permanecerá en esta pantalla. |
| Acciones a realizar: |
| 1: El usuario puede completar los diferentes datos sobre el favor que quiere pedir.  2: El usuario puede buscar en un mapa las ubicaciones relacionadas con el favor a pedir pulsando en los botones con forma de mapa.  4: El usuario puede volver a la pantalla “Mis favores” pulsando el botón de la flecha que se encuentra en la parte superior izquierda de la pantalla.  5: El usuario puede adjuntar fotos que haga con la cámara del móvil o importe de la galería.  6: El usuario puede pedir el favor pulsando el botón “Pedir nuevo favor”.  7: El usuario puede acceder a la pantalla de compra de “grollies” pulsando en cualquier parte del recuadro donde aparecen sus “grollies” actuales. |

## Interfaces Hardware

## Interfaces Software

## Interfaces de comunicación

## Requisitos funcionales

*El proyecto se dividirá en cuatro subsistemas. La organización está basada en un sistema orientada a objetos, dónde se complementan las clases y los casos de uso. La base de datos, la seguridad y eficiencia juegan un rol muy importante.*

## Clases

El proyecto estará **dividido en clases**, cada una de la cual dispone de un prototipo de las **funciones y atributos** necesarios para que la aplicación funcione tal y como se planea, así mismo de **esquema general** y **tarjetas CRC** en el [anexo](Anexo) mostrando la interacción entre las clases y los tipos de dichas funciones y atributos.

## Subsistema de gestión de usuarios:

***DATA***

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Data |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| dataType | Tipo de dato que representa | DataType |
| name | Nombre del campo del dato |  |
| rawData | Bytes de los datos |  |
| Nombre | **Descripción método** |  |
| getType | Devuelve el tipo de datos | DataType |
| getName | Devuelve el nombre del campo |  |
| getRawBytes | Devuelve los datos |  |
| getAsInteger | Devuelve una representación de entero |  |
| getAsString | Devuelve una representación como texto |  |
| getAsFloat | Devuelve una representación como float |  |
| getAsDouble | Devuelve una representación como double |  |
| getAsBoolean | Devuelve una representación como boolean |  |

Representa un dato abstracto de una fila de resultados de una consulta a base de datos.

|  |  |  |
| --- | --- | --- |
| Nombre del enumerado: | | DataType |
| Superclase: | | - |
| Nombre | **Descripción literal** | **Colaboraciones** |
| Integer | Entero |  |
| Float | Número de coma flotante |  |
| Double | Número de coma flotante de doble precisión |  |
| Boolean | Valor de sí o no |  |
| String | Cadena de texto |  |
| File | Archivo |  |

***DATA TYPE***Tipo enumerado que representa un tipo de datos.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | DataRow |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| dataList | Lista de datos de la fila | Data |
| Nombre | **Descripción método** |  |
| getColumnCount | Devuelve el número de columnas |  |
| getColumnDataAtIndex | Devuelve el dato en el índice dado | Data |

***DATA ROW***Es una fila de datos devuelta por una consulta a una base de datos.

***DATABASE SERVICE***Clase abstracta que representa una conexión a una base de datos.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | DatabaseService |
| Superclase: | | - |
| Nombre | **Descripción método** |  |
| query | Devuelve un array de filas de la base de datos | DataRow |
| close | Cierra la conexión |  |

Al crear nuestras abstracciones para las bases de datos, podemos cambiar de servicio de base de datos sin tener que modificar el código, fomentando la **reusabilidad**, por ejemplo, teniendo una conexión a una base de datos MySQL y otra a SQLite. Se puede ver mejor la relación entre estas clases en el siguiente esquema:

***USER MANAGER***Clase con métodos estáticos que representa una lista de usuarios. Está conectada a una base de datos mediante un **DatabaseService.**

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | UserManager |
| Superclase: | | - |
| Nombre | **Descripción atributo estático** | **Colaboraciones** |
| dataType | Tipo de dato que representa | DatabaseService |
| Nombre | **Descripción método estático** |  |
| getUser | Devuelve un usuario buscado | Data |
| deleteUser | Borra un usuario dado | User |
| modifyUser | Modifica un usuario dado actualizándolo | Data, User |
| insertUser | Inserta un nuevo usuario | User |

***USER***Usuario que utiliza la aplicación.

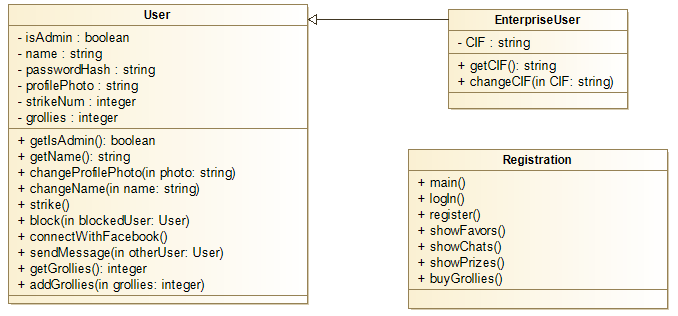
|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | User |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| isAdmin | ¿El usuario es administrador? |  |
| Name | Nombre del usuario |  |
| passwordHash | Hash seguro almacenado de la contraseña |  |
| profilePhoto | Ruta a la foto de perfil del usuario |  |
| strikeNum | Número de strikes que ha recibido |  |
| hrollies | Número de grollies que posee el usuario |  |
| Nombre | **Descripción método** |  |
| getIsAdmin | Devuelve si el usuario es administrador |  |
| getName | Devuelve el nombre del usuario |  |
| changeProfilePhoto | Cambia la foto de perfil del usuario |  |
| strike | Añade un strike al usuario |  |
| block | El usuario bloquea a otro usuario |  |
| connectWithFacebook | Conecta la cuenta con Facebook |  |
| sendMessage | Envía un mensaje a otro usuario |  |
| addGrollies | Añade (o resta) grollies |  |

***ENTERPRISE USER***Usuario especial que puede pagar por anunciar sus favores.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | EnterpriseUser |
| Superclase: | | User |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| CIF | CIF de la empresa |  |
| Nombre | **Descripción método** |  |
| getCIF | Devuelve el CIF de la empresa |  |
| changeCIF | Cambia el CIF de la empresa |  |

***REGISTRATION***Clase principal que da acceso a todas las funcionalidades.

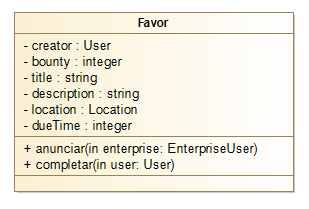
|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Registration |
| Superclase: | | - |
| Nombre | **Descripción método** |  |
| main | Método principal de la aplicación |  |
| logIn | Permite al usuario iniciar sesión |  |
| register | Permite al usuario registrarse |  |
| showFavors | Muestra los favores cercanos | FavorManager |
| showChats | Muestra los chats del usuario | UserManager |
| showPrizes | Muestra los premios canjeables | PrizeManager |
| buyGrollies | Muestra la opción de comprar grollies | Transaction |

  
Esquema de las relaciones entre clases:

## Subsistema de favores

***FAVOR***Representa un favor solicitado por un usuario.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Favor |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| creator | Usuario que ha creado el favor | User |
| bounty | Recompensa recibida por completarlo |  |
| title | Título del favor |  |
| description | Descripción del favor |  |
| location | Posición del favor | Location |
| dueTime | Tiempo máximo para completar el favor |  |
| Nombre | **Descripción método** |  |
| anunciar | El favor es anunciado por una empresa | EnterpriseUser |
| completar | El favor es completado por un usuario | User |

Diagrama de la clase en UML que muestra los tipos de cada método y atributo:

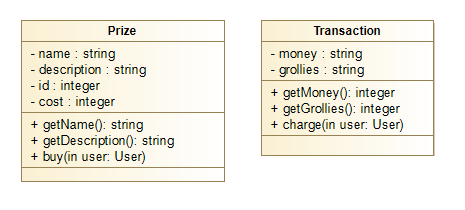
## Subsistema de premios y compras

***PRIZE***Premio que el usuario puede intercambiar si tiene suficientes grollies.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Prize |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| name | Nombre del premio |  |
| description | Descripción del premio |  |
| id | Identificación del premio (única) |  |
| cost | Coste en grollies del premio |  |
| Nombre | **Descripción método** |  |
| getName | Devuelve el nombre |  |
| getDescription | Devuelve la descripción |  |
| buy | El premio es comprado por un usuario | User |

***TRANSACTION***Transacción que representa un intercambio de dinero a cambio de grollies.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Transaction |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| money | Dinero a cobrar |  |
| grollies | Cantidad de grollies a ingresar |  |
| Nombre | **Descripción método** |  |
| getMoney | Devuelve el dinero real a cobrar |  |
| getGrollies | Devuelve los grollies a ingresar |  |
| charge | Realiza el cargo a un usuario | User |

Esquema UML de las clases:

## Subsistema de búsqueda y geolocalización

***FILTER***Filtro usado para restringir la búsqueda de favores.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Filter |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| keywords | Palabras clave |  |
| position | Posición del favor | Location |
| minGrollies | Mínima cantidad de grollies |  |
| Nombre | **Descripción método** |  |
| getKeywords | Devuelve las palabras clave |  |
| getPosition | Devuelve la posición del favor | Location |
| getMinGrollies | Devuelve la cantidad mínima de grollies |  |
| changeKeywords | Modifica las palabras clave |  |
| changePosition | Modifica la posición del favor | Location |
| changeMinGrollies | Modifica la cantidad mínima de grollies |  |

***LOCATION***Posición en el globo terráqueo.

|  |  |  |
| --- | --- | --- |
| Nombre de la clase: | | Location |
| Superclase: | | - |
| Nombre | **Descripción atributo** | **Colaboraciones** |
| latitude | Latitud |  |
| longitude | Longitud |  |
| Nombre | **Descripción método** |  |
| getLatitude | Devuelve la latitud |  |
| setLatitude | Cambia la latitud |  |
| getLongitude | Devuelve la longitud |  |
| setLongitude | Cambia la longitud |  |

Esquema UML con las relaciones entre las clases:



## Casos de uso:

|  |
| --- |
|  |
|  |

1. **Caso de uso: NEGOCIAR GROLLIES**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J1

**Objetivo en Contexto:**

El realizador selecciona negociar grollies para proponer un cambio en la recompensa

**Actor principal: Realizador de favor**

**Actores** **secundarios**: Pedidor de favor, Sistema, Base de datos

**Qué datos usa:** Los datos respectivos al favor seleccionado

**Precondiciones:**

El realizador de favor selecciona un favor en la lista de favores y después selecciona negociar grollies

**Postcondiciones:**

**Éxito:** Se modifica correctamente la recompensa propuesta por el realizador de favor

**Fallo:** Mensaje de error: no se ha podido modificar la recompensa

Flujo principal**:**

1. El realizador de favor selecciona un favor entre la lista de favores
2. El realizador de favor selecciona negociar grollies
3. El realizador de favor propone nueva recompensa
4. El pedidor de favor acepta la nueva recompensa
5. El sistema accede a la base de datos y cambia la recompensa que había.
6. El sistema notifica el correcto cambio de la recompensa

Flujos secundarios:

4.1 Si el pedidor de favor no acepta la nueva recompensa el sistema lo notificará y no se modificará la base de datos.

2.2 Si no es posible la comunicación entre el sistema y la base de datos se notificará y se volverá a la pantalla de búsqueda de favores

1. **Caso de uso: CAMBIAR E-MAIL**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J2

**Objetivo en Contexto:**

El objetivo es que el usuario desea cambiar su e-mail.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** e-mail antiguo y e-mail nuevo del usuario.

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente cambia su e-mail.

**Postcondiciones:**

**Éxito:** Se modifica correctamente su e-mail.

**Fallo:** No se modifica su e-mail.

Flujo principal**:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario reescribe su contraseña.
3. El usuario escribe su nuevo e-mal dos veces.
4. El usuario acepta el cambio del nuevo e-mail.
5. El sistema accede a la base de datos y cambia el e-mail.
6. El sistema notifica el correcto cambio de e-mail.

Flujos secundarios:

2.1 Si el usuario no escribe bien su contraseña, aparecerá un mensaje de error con otro posible intento.

3.2 Si los dos e-mails no coinciden entre sí no se podrá realizar el cambio.

1. **Caso de uso: CAMBIAR IDIOMA**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J3

**Objetivo en Contexto:**

El objetivo es que el usuario desea cambiar el idioma.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** Ninguno

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente “cambiar idioma”.

**Postcondiciones:**

**Éxito:** Se modifica correctamente el cambio de idioma.

**Fallo:** No se modifica el cambio de idioma.

Flujo principal**:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario selecciona “cambiar idioma”.
3. El usuario selecciona el idioma deseado entre los disponibles.
4. El usuario acepta el cambio de idioma.
5. El sistema accede a la base de datos y cambia el idioma.
6. El sistema notifica el correcto cambio de idioma.
7. **Caso de uso: CAMBIAR CONTRASEÑA**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J4

**Objetivo en Contexto:**

El objetivo es que el usuario desea cambiar la contraseña.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** Contraseña antigua y contraseña nueva.

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente “cambiar contraseña”.

**Postcondiciones:**

**Éxito:** Se modifica correctamente el cambio de contraseña.

**Fallo:** No se modifica el cambio de contraseña.

Flujo principal**:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario selecciona “cambiar contraseña”.
3. El usuario escribe su anterior contraseña.
4. El usuario escribe su nueva contraseña dos veces.
5. El sistema accede a la base de datos y cambia la contraseña.
6. El sistema notifica el correcto cambio de contraseña.

Flujos secundarios:

2.1 Si el usuario no escribe bien su anterior contraseña, aparecerá un mensaje de error con otro posible intento.

3.2 Si las dos contraseñas no coinciden entre sí, no se podrá realizar el cambio.

1. **Caso de uso: CAMBIAR FOTO DE PERFIL**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J5

**Objetivo en Contexto:**

El objetivo es que el usuario desea cambiar la foto de perfil.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** Imagen externa a la aplicación e imagen actual.

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente “cambiar imagen”.

**Postcondiciones:**

**Éxito:** Se modifica correctamente el cambio de imagen.

**Fallo:** No se modifica el cambio de imagen.

Flujo principal**:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario selecciona “cambiar imagen”.
3. El usuario accede a la galería de su teléfono y selecciona una imagen.
4. El sistema accede a la base de datos y cambia la imagen.
5. El sistema notifica el correcto cambio de imagen.
6. **Caso de uso: LEER NORMAS DE USO**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J6

**Objetivo en Contexto:**

El objetivo es que el usuario desea leer las normas de uso.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** Ninguna.

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente “leer normas de uso”.

**Postcondiciones:**

**Éxito:** Podrá leer las normas de uso.

**Fallo:** No podrá leer las normas de uso.

**Flujo principal:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario selecciona “leer normas de uso”.
3. El sistema accede a la base de datos y accede a las normas de uso.
4. **Caso de uso: CONECTAR CON FACEBOOK**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J7

**Objetivo en Contexto:**

El objetivo es que el usuario desea conectar la aplicación con Facebook.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos y sistema.

**Qué datos usa:** e-mail y contraseña de Facebook

**Precondiciones:**

El usuario selecciona “cambiar configuración” y posteriormente “conectar con Facebook”.

**Postcondiciones:**

**Éxito:** Se conecta con Facebook.

**Fallo:** No se conecta con Facebook.

Flujo principal**:**

1. El usuario selecciona “cambiar configuración”.
2. El usuario selecciona “conectar con Facebook”.
3. El usuario escribe su e-mail y contraseña de Facebook para poder acceder.
4. El sistema accede a la base de datos y accede con Facebook.
5. El sistema notifica la conexión con Facebook.

Flujos secundarios:

3.1 Si el usuario no escribe bien e-mail o contraseña, aparecerá un mensaje de error con otro posible intento.

1. **Caso de uso: BANEAR USUARIO**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J8

**Objetivo en Contexto:**

El usuario comete una infracción grave y es baneado.

Actorprincipal**: Administrador**

**Actores** **secundarios**: Base de datos, Sistema, Usuario.

**Qué datos usa:** La información que causa la infracción.

**Precondiciones:** El usuario ha cometido una infracción y el sistema lo detectó.

**Postcondiciones:**

**Éxito:** El usuario ha sido baneado con éxito.

**Fallo:** Se determina que la infracción no es tan grave.

Flujo principal**:**

1. El Sistema detecta una infracción de las normas de uso.
2. El Administrador revisa si la infracción es grave o no.
3. El Administrador determina que es grave y decide banear.

10.El Usuario es baneado.

Flujos secundarios:

2.1 Si la infracción no es grave el Administrador pone un strike Usuario en el caso de que la infracción sea leve.

2.2 Si la infracción no es grave el Administrador retira la disputa al Usuario en el caso de que la infracción fuese errónea.

1. **Caso de uso: PONER STRIKE A USUARIO**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J9

**Objetivo en Contexto:**

El usuario comete una infracción leve se le pone un strike.

Actorprincipal**: Administrador**

**Actores** **secundarios**: Base de datos, Sistema, Usuario.

**Qué datos usa:** La información que causa la infracción.

**Precondiciones:** El usuario ha cometido una infracción y el sistema lo detectó.

**Postcondiciones:**

**Éxito:** Al usuario le colocan un strike con éxito en el caso de haber 3 se banea al usuario.

**Fallo:** Se determina que la infracción no es tan grave.

Flujo principal**:**

1. El Sistema detecta una infracción de las normas de uso.
2. El Administrador revisa si la infracción es grave o no.
3. El Administrador determina que es leve y decide poner un strike.
4. El Usuario es puesto un strike.

Flujos secundarios:

2.1 Si la infracción no es grave el Administrador retira la disputa al Usuario en el caso de que la infracción fuese errónea.

4.1 Si hay tres strikes el Administrador pone en estado de baneo al Usuario.

1. **Caso de uso: ELIMINAR FAVOR NO ADECUADO.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J10

**Objetivo en Contexto:**

El usuario publica un favor no adecuado y el sistema procede a tratarlo.

Actorprincipal**: Administrador**

**Actores** **secundarios**: Base de datos, Sistema, Usuario.

**Qué datos usa:** La información que causa la infracción.

**Precondiciones:** El usuario ha cometido una infracción y el sistema lo detectó.

**Postcondiciones:**

**Éxito:** Se confirma que la infracción es grave y se borra.

**Fallo:** Se determina que la infracción no es tan grave y se reinstaura el favor.

Flujo principal**:**

1. El Administrador confirma que el usuario cometió una infracción.
2. El Administrador coloca strike o baneo.
3. El Administrador borra el favor no adecuado
4. **Caso de uso: BLOQUEAR CHAT .**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J11

**Objetivo en Contexto:**

Bloquear el chat de un usuario.

Actorprincipal**: Administrador**

**Actores** **secundarios**: Base de datos, Sistema, Usuario.

**Qué datos usa:** Los datos del chat a bloquear y/o los datos del que solicita bloquear.

**Precondiciones:** El usuario ha cometido una infracción y es denunciada por otro usuario o usuario decide bloquear directamente el chat sin denunciarlo.

**Postcondiciones:**

**Éxito:** Se confirma que la infracción y se bloquea o se bloquea directamente.

**Fallo:** El administrador no acepta la denuncia y la única manera de bloquear es por parte del usuario.

Flujo principal**:**

1. El Administrador confirma que el usuario cometió una infracción.
2. El Administrador elimina el chat y no permite al usuario hablar al solicitante.
3. El Administrador procede a ver el castigo necesario.

Flujos secundarios:

* 1. El Usuario procede a bloquear al otro usuario sin denunciar.

1. **Caso de uso: NOTIFICAR ACEPTACIÓN DE FAVOR.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J12

**Objetivo en Contexto:**

Notifica al usuario que su favor ha sido aceptado

Actorprincipal**: Sistema**

**Actores** **secundarios**: Base de datos, Usuario.

**Qué datos usa:** Los datos propios de los usuarios implicados.

**Precondiciones:** El favor ha sido solicitado por un usuario y tras ser aceptado el sistema envía una notificación al mismo.

**Postcondiciones:**

**Éxito:** Se confirma que se acepta la realización del favor y se le envía notificación.

**Fallo:** El dueño del favor niega al usuario la realización de este.

Flujo principal**:**

1. El Pedidor del Favor Acepta que el otro usuario realice su favor.
2. El Sistema recibe la información y formaliza una notificación.
3. El Sistema envía la notificación al realizador del favor.

Flujos secundarios:

1.1 El Pedidor del favor niega la posibilidad de hacer el favor terminando el flujo.

1. **Caso de uso: PEDIR FAVOR.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J13

**Objetivo en Contexto:**

Publicar un favor en la plataforma.

Actorprincipal**: Pedidor de Favor.**

**Actores** **secundarios**: Base de datos, Sistema.

**Qué datos usa:** Los datos proporcionados por el usuario.

**Precondiciones:** El pedidor del favor necesita estar registrado y tener grollies para dar en su cuenta.

**Postcondiciones:**

**Éxito:** El favor es publicado.

**Fallo:** El favor no ha podido ser publicado.

Flujo principal**:**

1. El Pedidor del Favor publica un favor poniendo toda la información.
2. El Sistema actualiza la feed con este nuevo favor.

Flujos secundarios:

* 1. El Sistema envía mensaje de error al usuario por no poderse publicar el favor.

1. **Caso de uso: PONER TIEMPO DE ENTREGA.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J14

**Objetivo en Contexto:**

Poner un tiempo máximo para entregar el favor.

Actorprincipal**: Pedidor de Favor.**

**Actores** **secundarios**: Base de datos, Sistema, realizador del favor.

**Qué datos usa:** Los datos proporcionados por el usuario.

**Precondiciones:** El pedidor del favor necesita estar registrado y tener un favor publicado.

**Postcondiciones:**

**Éxito:** El tiempo de favor es aplicado.

**Fallo:** El tiempo del favor no se pudo actualizar o se rechaza.

Flujo principal**:**

1. El Pedidor del Favor pone un tiempo para hacer el favor.
2. El Realizador del Favor acepta el tiempo y procede a realizar el favor.

Flujos secundarios:

* 1. El Realizador del Favor reniega del tiempo.

2.2.1 El Realizador del Favor negocia un nuevo tiempo.

2.2.2 El Realizador del Favor termina el favor y decide no hacerlo.

1. **Caso de uso: SOLICITAR PREMIOS.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J15

**Objetivo en Contexto:**

Cambiar Grollies por premios.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos, Sistema, Admin.

**Qué datos usa:** Los datos del catálogo de premios, del usuario y de su entidad bancaria.

**Precondiciones:** El usuario necesita haber hecho favores para poder conseguir regalos.

**Postcondiciones:**

**Éxito:** El premio tiene el coste en Grollies necesario al número de estos del usuario.

**Fallo:** El usuario no tiene suficientes Grollies para comprar el premio.

Flujo principal**:**

1. El Usuario Selecciona un premio del catálogo.

2. El Sistema Acepta la transacción y envía el premio reduciendo los Grollies correspondientes de la cuenta del usuario.

Flujos secundarios:

* 1. El Sistema Niega la transacción por falta de fondos o stock.

1. **Caso de uso: CAMBIAR CONFIGURACIÓN.**

**Nombre del AUTOR:** Santiago Mourenza

**Identificador**: J16

**Objetivo en Contexto:**

Cambiar alguna configuración de la app.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos, Sistema, Admin.

**Qué datos usa:** Ninguno.

**Precondiciones:** El usuario necesita haber iniciado sesión con su usuario y contraseña.

**Postcondiciones:**

**Éxito:** Entra en la configuración de la app.

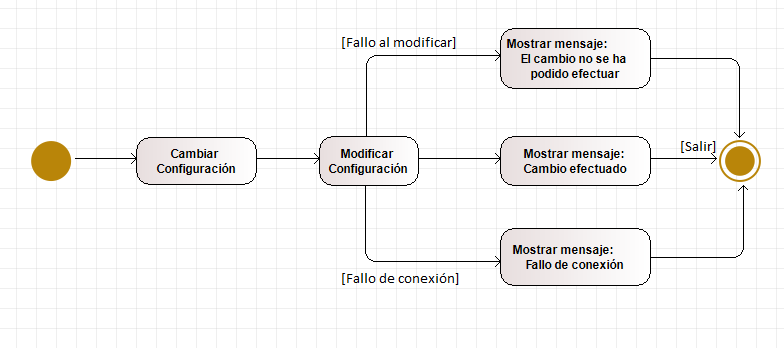
Flujo principal**:**

1. El Usuario Selecciona cambiar configuración.

2. Se cambia al menú de configuración y el usuario elije que cambio quiere hacer.

Flujos secundarios:

* 1. El usuario decide no cambiar nada y vuelve atrás.



1. **Caso de uso: SOLICITAR GROLLIES.**

**Nombre del AUTOR:** Jaime Martínez

**Identificador**: J17

**Objetivo en Contexto:**

El usuario quiere comprar Grollies en la aplicación

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos, Sistema, Admin.

**Qué datos usa:** Los datos del catálogo de Grollies, del usuario y de su entidad bancaria.

**Precondiciones:** El usuario quiere comprar Grollies en la aplicación.

**Postcondiciones:**

**Éxito:** El usuario compra Grollies

Flujo principal**:**

1. El Usuario selecciona comprar Grollies

2. El Usuario selecciona un método de pago

3. Rellena los datos de pago según el método elegido

4. El usuario compra Grollies

Flujos secundarios:

* 1. El usuario decide no comprar Grollies y volver atrás

1. **Caso de uso: PROPONER GROLLIES.**

**Nombre del AUTOR:** Jaime Martínez

**Identificador**: J18

**Objetivo en Contexto:**

Proponer una determinada cantidad de Grollies por obtener un favor.

Actorprincipal**: Usuario.**

**Actores** **secundarios**: Base de datos, Sistema, Admin.

**Qué datos usa:** Ninguno

**Precondiciones:** El usuario demandante quiere ofrecer una cantidad de Grollies por obtener un favor.

**Postcondiciones:**

**Éxito:** El usuario demandante acepta o rechaza la oferta

Flujo principal**:**

1. El Usuario ofrece un favor.

2. El Usuario propone una oferta en Grollies por obtener el favor

Flujos secundarios:

* 1. El usuario decide no realizar la oferta y volver atrás

1. **Caso de uso: GESTIONAR ALMACÉN DE DATOS**

**Nombre del AUTOR:** Pablo Torre

**Identificador**: J19

**Objetivo en Contexto:**

Tratar con los cambios en los datos del sistema

Actorprincipal**: Sistema**

**Actores** **secundarios**: Base de datos, Administrador, Usuario

**Qué datos usa:** Todos los cambios efectuados por el usuario/administrador que deben ser registrados además de toda la información almacenada previamente

**Precondiciones:**

El usuario solicita ver la Localización del Producto en el Plano.

**Postcondiciones:**

**Éxito:** Se guarda el cambio en la base de datos.

**Fallo:** Mensaje de error: no se ha podido efectuar el cambio

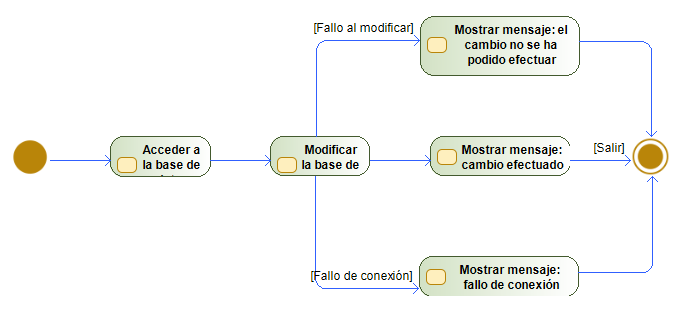
Flujo principal**:**

1. El sistema realiza uno de los siguientes cambios en la aplicación: añadir/eliminar usuario, … HACER
2. El sistema accede a la base de datos y la modifica.
3. El sistema muestra una notificación con el cambio efectuado
4. El sistema vuelve a la pantalla previa a este caso de uso

Flujos secundarios:

2.1 Si no se puede modificar la base de datos se notificará y se saltará al punto 4 del flujo principal. (Por ejemplo, si no se pone toda la información necesaria para añadir un usuario)

2.2 Si no es posible la comunicación entre el sistema y la base de datos se notificará y se saltará al punto 4 del flujo principal



1. **Caso de uso: BUSCAR UN FAVOR CON FILTRO**

**Nombre del AUTOR:** Pedro Palacios

**Identificador**: J20

**Objetivo en Contexto:**

Buscar entre la lista de favores publicados

Actorprincipal**: Usuario**

**Actores** **secundarios**: Base de datos, Usuario

**Qué datos usa:** La información almacenada de todos los favores que están disponibles en ese instante.

**Precondiciones:**

El usuario se encuentra en la pantalla de inicio de la aplicación

**Postcondiciones:**

**Éxito:** Se muestran todos los favores posibles

**Fallo:** Mensaje de error. Se posibilita realizar una nueva búsqueda

Flujo principal**:**

1. El usuario selecciona la opción buscar un favor.
2. El sistema muestra la pantalla de búsqueda con filtro.
3. El usuario introduce una recompensa mínima en grollies (opcional).
4. El usuario decide buscar por localización y selecciona una distancia máxima (opcional).
5. El usuario introduce una fecha límite máxima (opcional).
6. El usuario lanza la búsqueda.
7. El sistema selecciona los favores de base de datos que verifican los filtros introducidos.
8. La base de datos devuelve el resultado de esta selección.
9. El sistema muestra por pantalla los favores seleccionados.

Flujos secundarios:

6.1 Si no es posible la comunicación entre el sistema y la base de datos se le avisa al usuario de que el servicio no está disponible en ese momento.

8.1 En el caso en que la búsqueda no obtenga resultados se notifica al usuario y se le da la opción de modificar filtros de búsqueda

1. **Caso de uso: SELECCIONAR FAVOR**

**Nombre del AUTOR:** Alberto Almagro

**Identificador**: J21

**Objetivo en Contexto:**

El usuario selecciona un favor de los que aparecen en pantalla para acceder a más información sobre esta.

Actorprincipal**: Usuario**

**Actores** **secundarios**: Base de datos, Sistema

**Qué datos usa:** Información mostrada por pantalla por *buscar discoteca*. La información almacenada de todas las discotecas en la base de datos

**Precondiciones:**

El usuario ha utilizado *buscar favor* con éxito

**Postcondiciones:**

**Éxito:** Se accede al perfil con la información del favor seleccionado

**Fallo:** Mensaje de error al cargar el favor seleccionado

Flujo principal**:**

1. El usuario selecciona un favor mostrado en pantalla.
2. El sistema pide a la base de datos la información del favor.
3. El base de datos devuelve la información del favor solicitado
4. El sistema muestra en pantalla la información de la discoteca seleccionada

Flujos secundarios:

2.1 Si no es posible la comunicación entre el sistema y la base de datos se avisa al usuario de que el servicio no está disponible en ese momento.

1. **Caso de uso: CHATEAR CON USUARIO**

**Nombre del AUTOR:** Juan Carlos Llamas

**Identificador**: J22

**Objetivo en Contexto:**

El usuario selecciona el chat para hablar sobre el favor en cuestión

Actorprincipal**: Usuario**

**Actores** **secundarios**: Usuario, Sistema, Base de datos

**Qué datos usa:** Los datos respectivos al favor seleccionado

**Precondiciones:**

El usuario selecciona un favor en la lista de favores y después selecciona chatear.

**Postcondiciones:**

**Éxito:** Se abre el chat correctamente, permitiendo a ambos usuarios hablar.

**Fallo:** Mensaje de error: no se ha podido abrir chat

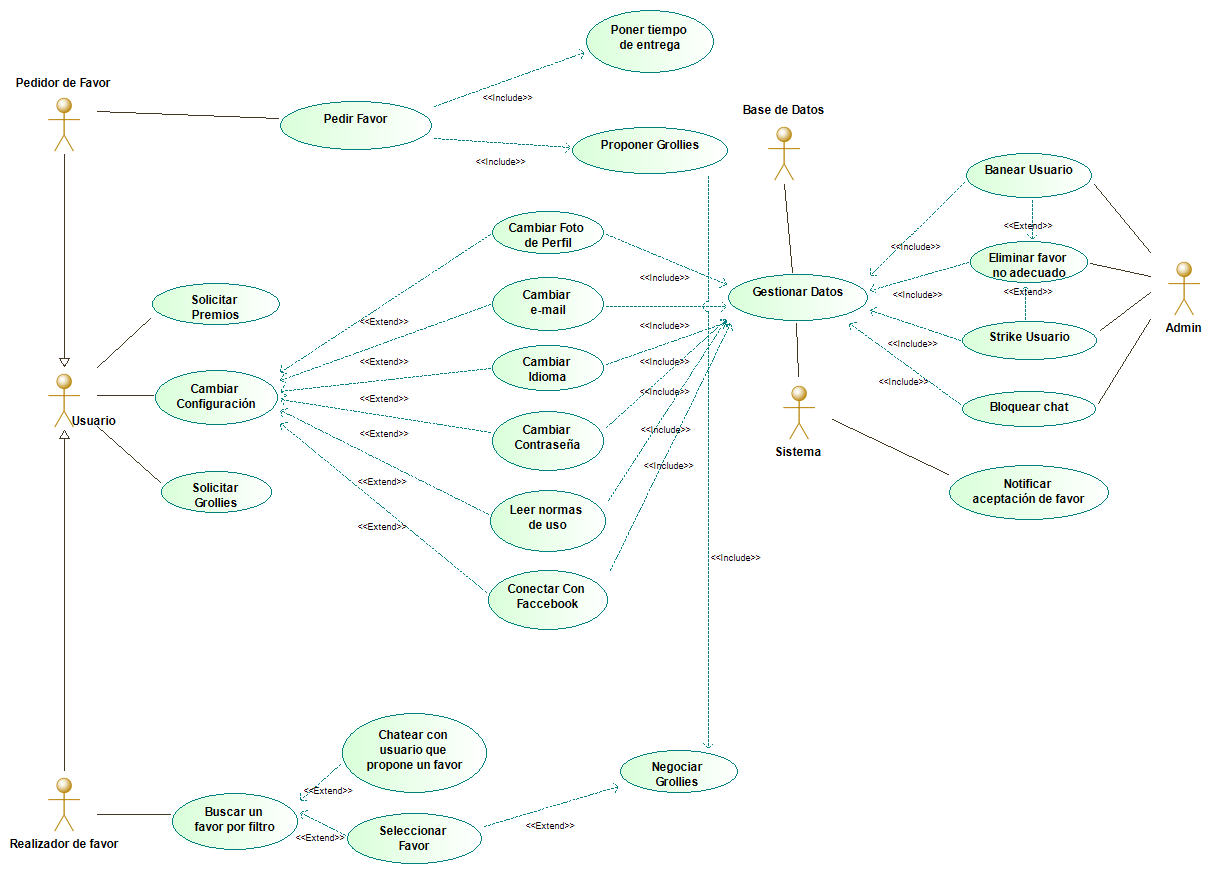
Flujo principal**:**

1. El usuario selecciona un favor entre la lista de favores
2. El usuario selecciona el chat
3. El sistema accede a la base de datos y almacena un nuevo chat.
4. El sistema lleva a usuario es a una nueva pantalla en la que chatear.

Flujos secundarios:

2.1 Si no se puede almacenar un nuevo chat en la base de datos se notificará al usuario, diciéndole que pruebe en otro momento.

* + 1. Si no es posible la comunicación entre el sistema y la base de datos se notificará y se volverá a la pantalla de búsqueda de favores



## Requisitos de rendimiento:

La aplicación está orientada **a un público amplio**, y hemos planeado un **gran número de usuarios**, por lo tanto, estimamos que el número de terminales conectados al servidor puede ser bastante alto.

La actividad de los usuarios estará por lo general **repartida entre los días de la semana**, pero sufrirá picos de carga en **períodos de exámenes**, al estar el público objetivo más ocupado y por tanto con mayor necesidad de pedir favores. Es necesario adaptar los servidores para soportar dicha carga.

## Requisitos lógicos de la base de datos:

Todos los datos estarán almacenados en una **base de datos relacional**, y es importante mantener su **consistencia** y coherencia en todo momento, sobre todo con las transacciones monetarias. Deberá permitir **acceso concurrente** a un gran número de personas (ver requisitos de rendimiento). Se mantendrá una **copia de seguridad** que se renovará periódicamente.

Las principales clases que se relacionan con la base de datos son:

* **UserManager:** Gestiona los usuarios, administradores y empresas.
* **FavorManager:** Gestiona los favores solicitados por usuarios.
* **PrizeManager:** Gestiona los premios disponibles a cambio de grollies.

## Restricciones de diseño

La aplicación será implementada en un lenguaje **orientado a objetos**, con soporte para **módulos** y **paquetes**.

El almacenamiento se realizará en una **base de datos relacional**. Debido al diseño modular, pueden usarse distintos sistemas, aunque principalmente se usará MySQL.

## Atributos del sistema software

**DISPONIBILIDAD**La aplicación debe estar **disponible en todo momento** para el acceso por parte de los usuarios. El mantenimiento debe realizarse de manera que no interfiera con el uso normal de la aplicación.   
  
Deberán realizarse **copias de seguridad frecuentes y redundantes**, para poder reestablecer los datos en otros servidores en caso de daño. Dicha responsabilidad recae sobre los administradores.

**SEGURIDAD**Los datos del usuario seguirán la normativa recogida por la **Agencia Española de Protección de Datos** y se usará en todo momento **conexión cifrada** entre la aplicación y el servidor, para asegurarse que los datos de los usuarios no se vean comprometidos.

Se seguirá el protocolo **PCI** (Payment Card Industry) no almacenando ningún dato bancario de los usuarios, gestionándolo todo a través de una API.  
  
Las **contraseñas** de los usuarios serán hasheadas adecuadamente y no se almacenará la contraseña en texto plano o cifrada.

**MANTENIBILIDAD**Los administradores se encargan de que el **servidor no se sature**, eliminando los **datos no necesarios** y copias de seguridad obsoletas. Los administradores también son los encargados de reiniciar el sistema en caso de error.

**ACCESIBILIDAD**La aplicación debe permitir que múltiples usuarios se conecten **concurrentemente** al servidor y a la base de datos. Cualquier usuario con la **aplicación descargada e identificado** con conexión a Internet debería ser capaz de acceder a la misma.