MEMBRESÍA

Proyecto JavaEE de Fin de Carrera

Julio M. Fernández Jiménez

A mi marido y a la música (en ese orden).
Sin ninguno de ellos podría haber culminado mi viaje.

Tabla de contenido

Descripción del proyecto

El proyecto cuyo análisis se presenta a continuación consiste de una aplicación online para el servicio de la gestión de socios de la **Agrupación Astronómica de Terrassa**, aunque su funcionalidad puede ser aplicable a cualquier asociación independientemente de su actividad.

La aplicación, denominada "Membresía", permite gestionar el cobro de las cuotas de los socios de una agrupación así como la información personal de éstos. Mediante el componente de subscripciones, el gestor del sistema puede crear una o más subscripciones y asociarlas a los socios participantes. Los socios pueden realizar los pagos de forma parcial o de una sola vez, e incluso aplazar los pagos si así lo desean. El sistema permite también el envío de correos por parte del administrador del sistema a sus socios así como el envío automático de avisos y alertas de pago.

Objetivos del proyecto

Se puede decir, que a nivel profesional este proyecto representa la migración de mis conocimientos en desarrollo web en PHP a un entorno de trabajo basado en Java. Ésta, y la consolidación de las destrezas adquiridas durante mis años de carrera, son sin duda alguna las principales motivaciones para la creación de este trabajo. Dentro del ámbito profesional, se busca alcanzar un nivel de conocimiento del lenguaje de programación Java, y de la tecnología JavaEE, lo suficientemente alto como para poder competir en el mercado laboral actual.

Play Framework

Play es un framework de desarrollo basado en programación reactiva para desarrollos en JAVA y Scala. Este framework ha sido desarrollado pensando en la productividad del desarrollador y en en el rendimiento del hardware, de forma que promueve un modelo de hardware que escala horizontalmente en vez del tradicional escalamiento vertical de los tradicionales sistema JavaEE.

Play permite la edición de código en tiempo real sin necesidad de reiniciar el contenedor de aplicaciones. Con la finalidad de proveer un desarrollo continuo, **Play** detecta los cambios realizados en el código, lo compila y carga en el JMV sin necesidad de interacción por parte del usuario ni de reiniciar la maquina virtual.

La capacidad reactiva de **Play** lo hace único en su género al permitir múltiples llamadas asincrónicas simultáneas al servidor de forma que se pueden actualizar segmentos concretos de la aplicación. Por otro lado, **Play** es RESTful por naturaleza gracias a su sistema de *routing* y su soporte nativo para datos JSON.

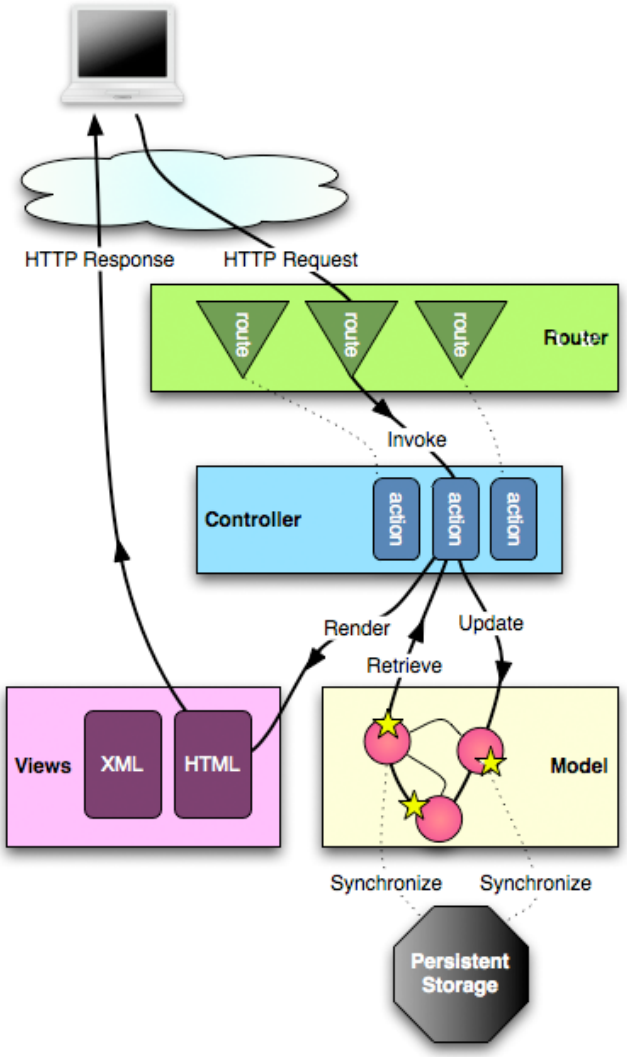
Aunque **Play** puede ser utilizado para desarrollar aplicaciones a nivel *entreprise*, éste difiere del sistema tradicional JavaEE al carecer de *servlets*. La ausencia de *servlets* hace de **Play** un framework *stateless*, es decir, sin persistencia de información entre sesiones.

Con **Play** el desarrollador pueda trabajar con el lenguaje de programación JAVA o Scala de forma dedicada o en conjunto, aunque esta última opción esta desaconsejada. Scala es un lenguaje de programación orientado a objetos puro en el sentido de que cada valor es un objeto dentro de la declaración del lenguaje.

A pesar de las innovaciones y ventajas que **Play** ofrece, éste no deja de ser un sistema basado en el modelo estructural MVC. En **Play**, todas las peticiones HTTP pasan por el sistema de enrutamiento del *framework* que traduce la URL y dirige la solicitud al

controlador adecuado. El controlador interacciona con el modelo y manipula el sistema persistente de datos para generar la estructura de datos solicitada. Acto y seguido, el controlador, a través de las vistas, genera el HTML solicitado y lo envía como respuesta.

La capa de modelo de **Play** hace uso de Ebean como sistema ORM para la interacción con los sistemas de persistencia de datos. Ebean ORM esta basado en el modelo JPA de Java, lo que hace que su curva de aprendizaje sea mínima para programadores familiarizados con el entorno JavaEE. En adición, Ebean permite que pueda hacerse uso de cualquier sistema de gestión de base de datos.



Arquitectura y Diseño

La aplicación está compuesta por una serie de módulos o componentes, cada uno de ellos encargado de una función concreta en el sistema. Los cinco componentes principales del sistema de escriben a continuación.

Componente de usuarios

El componente de usuarios se encarga de gestionar aquellos usuarios registrados de ámbito administrativos así como la gestión y el acceso al sistema a través de la autenticación de usuarios. La autenticación se realizan mediante el uso del email y la contraseña. Este componente permite también solicitar la restauración de la clave de acceso en caso de haberla olvidado.

En adición, también permite crear y modificar usuarios existentes, realizar cambios en el perfil personal del usuario que está autenticado y eliminar usuarios existentes. Desde este componente también es posible listar todos los usuarios actualmente registrados, listado desde donde se puede acceder a acciones concretas sobre los objetos enumerados, tales como las de editar o eliminar usuario.

Esta componente también permite la gestión de los socios de la asociación y del ciclo de vida de cada uno de los socios.

El componente permite listar todos los socios y tomar acción sobre cada uno de ellos desde el interface. Estas acciones pueden ser las de modificar los datos personales, eliminar el socio o crea uno nuevo. En adición es posible añadir un socio a una subscripción concreta, eliminarlo de dicha subscripción y realizar pagos pendientes de forma integra o parcial.

Componente de subscripciones

La gestión de subscripciones permite realizar todas las gestiones relacionadas con el manejo de subscripciones. A través de este componente se podrán crear, modificar y eliminar diferentes tipos de subscripciones así como definir su periodicidad y visualizar gráficamente los ingresos devengados.

Este componente cuenta como actor secundario el mismo sistema, que crea periodos de cobro para las subscripciones según se haya definido su periodicidad. Estos periodos de cobro son generados de forma automática cada vez que el periodo anterior vence.

El concepto detrás de la gestión de más de una subscripción está motivado por la necesidad de crear líneas de cobros alternativas a la principal según sea necesario. Existen una gran cantidad de casos en los que pueda ser necesario tener más de una subscripción vigente para con un grupo como por el ejemplo la recaudación de fondos para una actividad especial, un viaje o una celebración en particular.

Es también a través de este componente que los socios podrán realizar pagos, mediante Paypal, de las subscripciones pendientes de abonar y de forma parcial o íntegra haciendo uso de una URL encriptada que recibirán en su correo electrónico a principio de cada periodo de recaudación de cuotas. De igual forma, los administradores del sistema podrán manualmente registrar un pago sin necesidad de la URL encriptada.

Componente de comunicaciones

El componente de gestión de mensajes facilita el envío de notificaciones o mensajes a los socios. Estas notificaciones pueden ser personalizadas, a partir de una plantilla o automatizadas. En adición, el componente también habilita la capacidad de creación, edición y eliminación de dichas plantillas.

Componente de estadísticas

El componente de estadísticas analiza la recaudación de las cuotas a través del tiempo. Existen básicamente dos funciones principales de este componente. La primera de ellas muestra gráficamente (gráfico de barras) los ingresos mensuales por subscripción según la recaudación de las cuotas. Este gráfico puede también mostrar los ingresos desde una perspectiva global (todas las subscripciones).

La segunda funcionalidad de este componente es la de mostrar un listado de los socios ha pagado alguna cuota recientemente y mostrar los periodos por cobrar mas recientes.

Planificación del Trabajo Realizado

En todo trabajo planificado pueden existir discrepancias entre el plan estimado inicial de trabajo y el resultante. A continuación de muestran las diferencias entre ambas planificaciones donde podremos comparar tanto el tiempo invertido en cada tarea como el coste de cada una de ellas.

Para calcular los costes, se han utilizado tres tarifas diferentes según el tipo de tarea realizada. La siguiente tabla resume brevemente los roles de tareas y la tarifa por hora así como su código de color correspondiente.

Role	Precio/hora
Gestión/Administración	10€
Diseño	20€
Programación	25€

Planificación Inicial

A continuación se muestra la agenda de trabajo propuesta para el proyecto desglosada en sub-tareas de carácter general. El tiempo estimado para cada tarea se ha calculado en base a una dedicación de 3 horas al día.

Tarea	Hito	Horas	Fecha entrega	Coste
Plan de trabajo inicial	**PEC1**	**30**	**5/10/15**	300,00 €
Diseño del sistema	**PEC2**		**4/11/15**	
Especificaciones por componente	PEC2	15	10/10/15	225,00 €
Definición de diagramas de casos de uso	PEC2	12	14/10/15	180,00 €
Diagrama de clases	PEC2	15	19/10/15	225,00 €
Diagrama de componentes	PEC2	18	25/10/15	270,00 €
Implementación	**PEC3**		**18/12/15**	
Diseño UI (Adaptacion)	PEC2	24	7/11/15	480,00 €
Definición de modelos y funciones de interacción con DB	PEC2	30	17/11/15	600,00 €
Programación controladores	PEC2	30	27/11/15	600,00 €
Programación de servicios y business logic	PEC2	45	12/12/15	900,00 €
Testing	PEC2	15	17/12/15	300,00 €
Memoria y presentación	**PEC4**		**11/1/16**	

Preparación memoria escrita	PEC4	30	27/12/15	300,00 €
Preparación material audiovisual	PEC4	30	6/1/16	300,00 €
				4.680,00 €

Planificación Real

Para la planificación real de realizaron jornadas de cinco horas cada una y de diez los fines de semanas y festivos. El cuadro de tiempos de ejecución y presupuesto real para el sistema es como se muestra a continuación. Las tareas del área de desarrollo e implementación carecen de una fecha de finalización concreta al haber sido realizadas de forma intercala.

Tarea	Hito	Horas	Fecha entrega	Coste
Plan de trabajo inicial	**PEC1**	**30**	**5/10/15**	300,00 €
Diseño del sistema	**PEC2**		**7/11/15**	
Especificaciones por componente	PEC2	10	20/10/15	150,00 €
Definición de diagramas de casos de uso	PEC2	20	25/10/15	300,00 €
Diagrama de clases	PEC2	10	28/10/15	150,00 €
Diagrama de componentes	PEC2	45	7/10/15	675,00 €
Implementación	**PEC3**		**18/12/15**	
Diseño UI (Adaptacion)	PEC2	35		700,00 €
Definición de modelos y funciones de interacción con DB	PEC2	40		800,00 €
Programación controladores	PEC2	45		900,00 €
Programación de servicios y business logic	PEC2	80		1.600,00 €
Testing	PEC2	20		400,00 €
Memoria y presentación	**PEC4**		**11/1/16**	
Preparación memoria escrita	PEC4	30	7/1/16	300,00 €
Preparación material audiovisual	PEC4	30	11/1/16	300,00 €
				6.575,00 €

Casos de uso

A continuación se describen los diferentes casos de uso según el componente al que pertenecen.

Usuarios administrativos

CU_1	**Login**
Actor principal	Usuario administrador.
Objetivos	Acceder al sistema como usuario registrado.
Precondiciones	No estar autenticado y el usuario debe estar registrado previamente.
Postcondiciones	El usuario queda autenticado en el sistema y tiene acceso al menú y la gestión del sistema.
Escenario principal	El usuario entra su email y contraseña. El usuario es autenticado y llevado al panel principal del sistema desde el que se pueden acceder a todas las funcionalidades del mismo.
Escenario alternativo	1. El usuario introduce un email o contraseña errónea. 2. El sistema avisa al usuario de que ha introducido una secuencia de datos de autenticación incorrecta.

CU_2	**Recover password**
Actor principal	Usuario administrador.
Objetivos	Restaurar la clave o contraseña de acceso.
Precondiciones	No estar autenticado y el usuario debe estar registrado previamente.
Postcondiciones	El usuario adquiere una nueva clave de acceso a través de su correo electrónico.
Escenario principal	Desde el panel o formulario de recuperación de contraseña, el usuario entra su email y pulsa en “recuperar clave” para iniciar el proceso. El usuario recibe por correo electrónico una nueva clave generada de forma automática.
Escenario alternativo	1. El email introducido es incorrecto o no pertenece a ningún usuario. 2. El sistema avisa el usuario con un mensaje de error y le invita a proceder de nuevo.

CU_3	**Logout**
Actor principal	Usuario administrador.
Objetivos	Salir del sistema como usuario autenticado.

Precondiciones	Estar autenticado en el sistema.
Postcondiciones	El usuario es expulsado del sistema y debe volver a autenticarse para poder acceder.
Escenario principal	1. El usuario hace clic sobre el enlace de “salir” o “logout”.

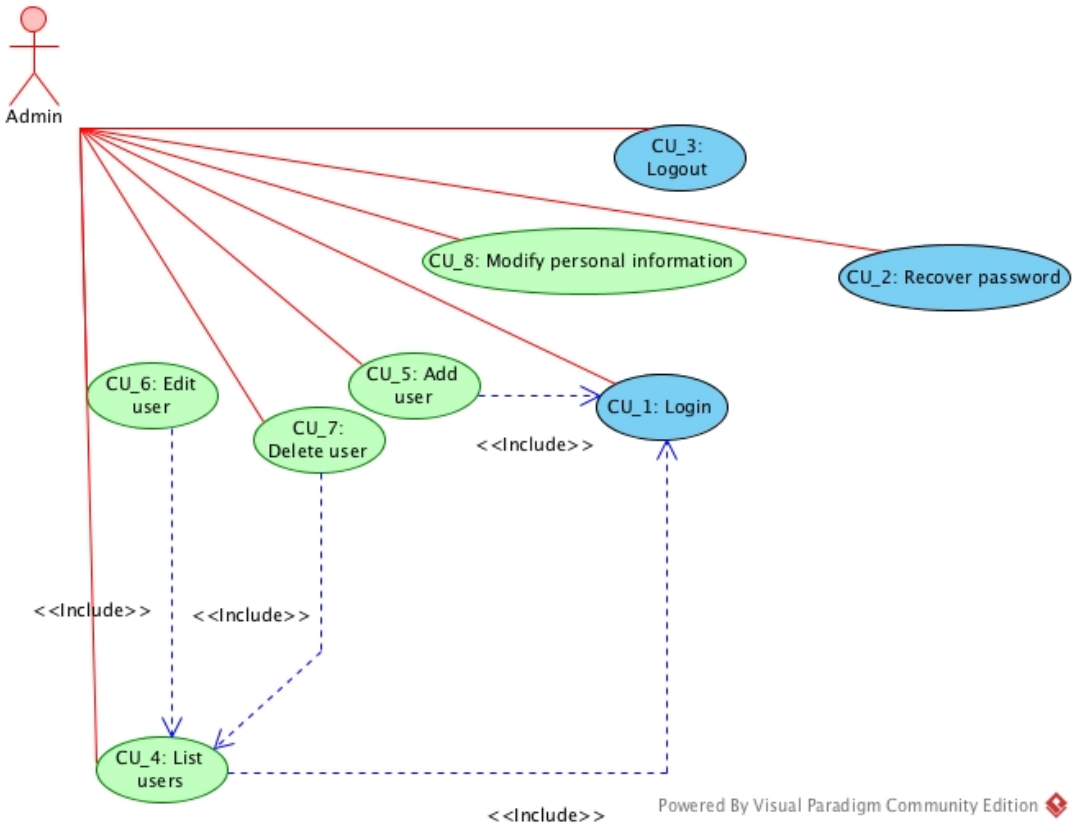
CU_4	**List users**
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de todos los usuarios registrados.
Precondiciones	Estar autenticado en el sistema.
Postcondiciones	Se muestra en pantalla todos los usuarios administradores del sistema actualmente registrados.
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Listar usuarios”. 2. El sistema muestra una tabla con todos los usuarios administradores registrados.

CU_5	**Add user**
Actor principal	Usuario administrador.
Objetivos	Añadir un nuevo usuario a la base de datos
Precondiciones	Estar autenticado en el sistema. Que el correo electrónico del nuevo usuario no este ya registrado bajo algún otro usuario.
Postcondiciones	El usuario queda registrado en el sistema. Se le envía un email al nuevo usuario con sus datos de acceso.
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Nuevo usuario” o sobre el botón en la esquina superior derecha del listado de usuarios. 2. El sistema muestra el formulario de alta que el usuario debe completar con los campos marcados como obligatorios. 3. El usuario completa el formulario y pulsa guardar 4. El usuario queda registrado y se le envía un email notificándole de la acción así como de sus datos de acceso. 5. Se retorna a la lista de usuarios administrativos.
Escenario alternativo	1. Se pulsa el botón guardar tras introducir un correo electrónico ya registrado para otro usuario. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error. 3. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 4. El sistema retorna al usuario al formulario y le pide que rectifique el error.

CU_6	Edit user
Actor principal	Usuario administrador.
Objetivos	Modificar los datos personales de un usuario registrado.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios administrativos.
Postcondiciones	1. Los datos del usuario quedan modificados.
Escenario principal	1. El usuario hace clic sobre el icono de “modificar usuario” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de usuarios administrativos. 2. Se carga el formulario de datos de usuario con los datos de éste. 3. Se modifican los datos y se pulsa el botón de guardar. 4. Los datos quedan guardados. 5. Se retorna a la lista de usuarios administrativos.
Escenario alternativo	1. Se pulsa el botón guardar tras modificar el correo electrónico por uno ya registrado para otro usuario. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error. 3. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 4. El sistema retorna al usuario al formulario y le pide que rectifique el error.

CU_7	Delete user
Actor principal	Usuario administrador.
Objetivos	Eliminar un usuario del sistema.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios administrativos.
Postcondiciones	Los datos del usuario quedan modificados.
Escenario principal	1. El usuario hace clic sobre el icono de “eliminar usuario” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de usuarios administrativos. 2. Se le presenta al usuario con una alerta de confirmación de acción. 3. El usuario confirma la acción. 4. Se elimina el usuario. 5. Se retorna a la lista de usuarios administrativos.
Escenario alternativo	1. El usuario no confirma la acción de eliminar el usuario 2. Se retorna a la lista de usuarios administrativos.

CU_8	Modify personal profile
Actor principal	Usuario administrador.
Objetivos	Permite modificar los datos personales del usuario actualmente autenticado.
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. Los datos del usuario quedan modificados.
Escenario principal	1. El usuario hace clic sobre el enlace “perfil” ubicado en la barra de información del usuario en la zona superior derecha de la pantalla. 2. Se carga el formulario de datos de usuario con los datos del usuario actual. 3. Se modifican los datos y se pulsa el botón de guardar. 4. Los datos quedan guardados. 5. Se retorna a la lista de usuarios administrativos.
Escenario alternativo	1. Se pulsa el botón guardar tras modificar el correo electrónico por uno ya registrado para otro usuario. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error. 3. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 4. El sistema retorna al usuario al formulario y le pide que rectifique el error.



Gestión de socios

CU_9	List members
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de todos los usuarios socios o miembros.
Precondiciones	Estar autenticado en el sistema.
Postcondiciones	Se muestra en pantalla todos los usuarios socios del sistema actualmente registrados.
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Listar socios”. 2. El sistema muestra una tabla con todos los usuarios socios registrados.

CU_10	Add member
Actor principal	Usuario administrador.
Objetivos	Añadir un nuevo socio a la base de datos
Precondiciones	Estar autenticado en el sistema. Que el correo electrónico del nuevo socio no esté ya registrado bajo algún otro usuario socio o usuario administrador.
Postcondiciones	1. El socio queda registrado en el sistema.

Escenario principal	2. El usuario hace clic sobre el enlace del menú “Nuevo socio” o sobre el botón en la esquina superior derecha del listado de socios. 3. El sistema muestra el formulario de alta que el usuario debe completar con los campos marcados como obligatorios. 4. El usuario completa el formulario y pulsa guardar. 5. El socio queda registrado. 6. Se le envía al nuevo miembro un email de bienvenida. 7. Se retorna a la lista de socios.
Escenario alternativo	1. Se pulsa el botón guardar tras introducir un correo electrónico ya registrado para otro socio o usuario administrador. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error. 3. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 4. El sistema retorna al usuario al formulario y le pide que rectifique el error.
Extensiones	CU_15, CU_16

CU_11	**Edit member**
Actor principal	Usuario administrador.
Objetivos	Modificar los datos personales de un usuario socio.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de socios.
Postcondiciones	1. Los datos del usuario quedan modificados.
Escenario principal	1. El usuario hace clic sobre el icono de “modificar socio” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de usuarios socios. 2. Se carga el formulario de datos de socios con los datos de éste. 3. Se modifican los datos y se pulsa el botón de guardar. 4. Los datos quedan guardados. 5. Se retorna a la lista de usuarios socios.
Escenario alternativo	1. Se pulsa el botón guardar tras modificar el correo electrónico por uno ya registrado para otro usuario. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error. 3. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria.

	4. El sistema retorna al usuario al formulario y le pide que rectifique el error.
Extensiones	CU_15, CU_16

CU_12	**Delete member**
Actor principal	Usuario administrador.
Objetivos	Eliminar un socio del sistema.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios administrativos.
Postcondiciones	1. El socio queda eliminado
Escenario principal	1. El usuario hace clic sobre el icono de “eliminar socio” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de socio. 2. Se le presenta al usuario con una alerta de confirmación de acción. 3. El usuario confirma la acción. 4. Se elimina el socio. 5. Se retorna a la lista de usuarios socio.
Escenario alternativo	1. El usuario no confirma la acción de eliminar el socio 2. Se retorna a la lista de usuarios socio. 1. El socio tiene pagos asociados 2. Se elimina la información personal del socio pero se mantiene la entrada en la base de datos con el fin de conservar la información de los pagos. 3. Se retorna al listado de socios.

CU_13	**Show member**
Actor principal	Usuario administrador.
Objetivos	Ver la ficha de un socio
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios administrativos.
Postcondiciones	1. La información del usuario se muestra en pantalla
Escenario principal	1. El usuario hace clic sobre el icono de “ver socio” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de socio. 2. Se le presenta al usuario con la ficha del socio, desde donde podrá ver los detalles de contacto del socio, sus subscripciones activas y su historial de pago por subscripción.
Extensiones	CU_17

CU_14a	Register Payment
Actor principal	Usuario administrador.
Objetivos	Registrar el pago de un socio a una subscripción concreta
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios socios.
Postcondiciones	1. El pago del socio queda registrado para un periodo concreto de la subscripción seleccionada.
Escenario principal	1. El usuario hace clic sobre el icono de “registrar pago” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla o listado de socio. 2. Se le presenta al usuario con un listado de las subscripciones a las cuales el socio esta inscrito. 3. El usuario selecciona la subscripción a la que desea aplicar el pago. 4. El sistema muestra aquellos periodos de la subscripción actuales o pasados para los que el socio tiene un balance pendiente. 5. El usuario introduce la cantidad que desea abonar 6. El sistema registra el pago. 7. Se retorna al usuario al listado de socios.
Escenario alternativo	1. El usuario tiene todos los pagos al día. 2. El sistema no permite registra ninguna pago 3. Retorna al listado de socios 1. La cantidad introducida es superior al balance actual para el periodo de cobro en curso 2. El sistema cancela el pago y retorna al formulario de pago.
Extensiones	CU_17

CU_14b	Register Payment
Actor principal	Usuario socio
Objetivos	Registrar el pago de un socio a una subscripción concreta
Precondiciones	1. Ser poseedor de la URL encriptada para realizar pagos
Postcondiciones	1. El pago del socio queda registrado para un periodo concreto de la subscripción seleccionada.
Escenario principal	1. El socio hace clic sobre el enlace de pago enviado previamente por email 2. El usuario es dirigido a un formulario de pago 3. El usuario define la cantidad a abonar del total pendiente 4. El usuario es dirigido a Paypal para completar su pago 5. El pago se realiza con éxito 6. Se retorna al portal inicial 7. Se confirma el pago

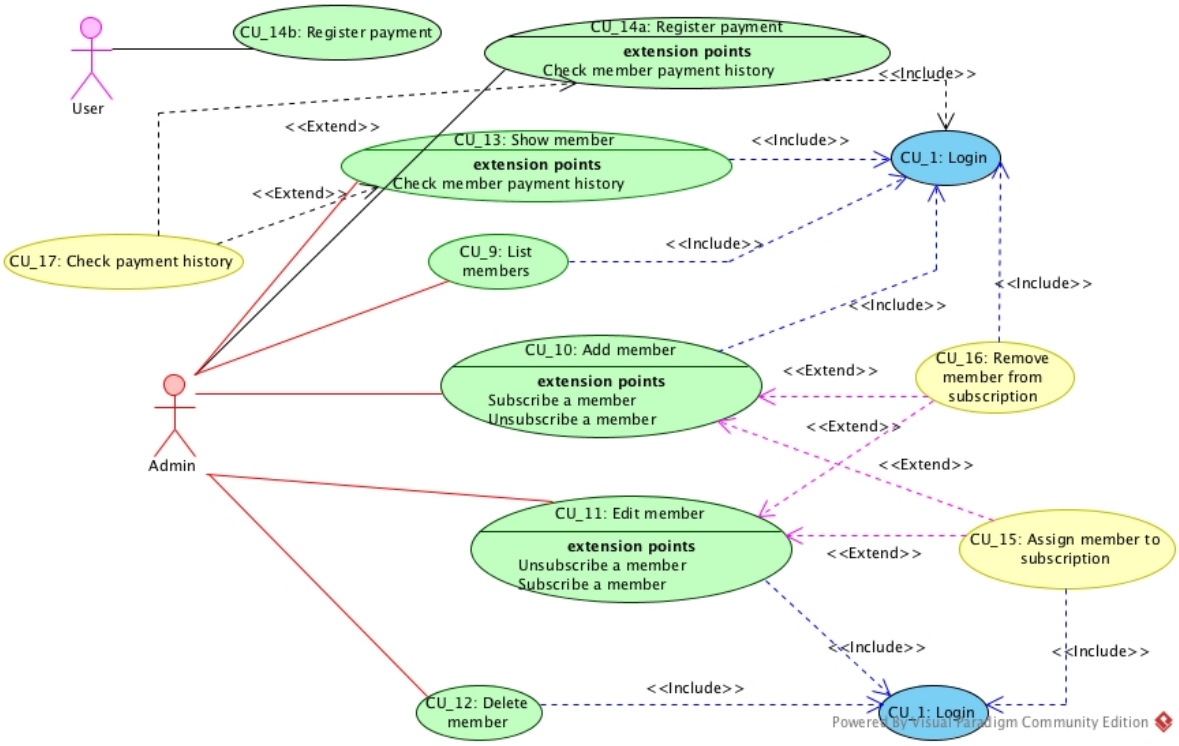
Escenario alternativo	1. La pasarela de pago no puede completar el pago 2. El pago no se registra 3. Se retorna al portal 4. Se informa del error de pago al socio

CU_15	**Assign member to subscription**
Actor principal	Usuario administrador.
Objetivos	Registra a un usuario en una subscripción concreta
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios socios.
Postcondiciones	El usuario socio queda registrado en una subscripción. Queda pendiente el balance del periodo de cobro en curso.
Escenario principal	1. El usuario hace clic sobre el icono de “editar socio” desde el listado de socios. 2. En la sección de subscripciones, el usuario selecciona la subscripción a la cual desea subscribir al socio 3. Un email es enviado al socio con la información de la nueva subscripción.

CU_16	**Remove member from subscription**
Actor principal	Usuario administrador.
Objetivos	Elimina a un usuario en una subscripción concreta
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios socios.
Postcondiciones	El usuario socio queda removido de la subscripción Queda cancelado el balance pendiente del periodo en curso.
Escenario principal	1. El usuario hace click sobre el icono de “editar socio” desde el listado de socios. 2. En la sección de subscripciones, el usuario deselecciona la subscripción a la cual se desea dejar de ser subscriptor.

CU_17	**Check payment history**
Actor principal	Usuario administrador.
Objetivos	Muestra el historial de pago del usuario
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios socios.
Postcondiciones	1. Muestra un listado de todos los pagos realizados por el socio en orden cronológico así como la subscripción y el periodo al que corresponde
Escenario principal	1. El usuario accede a la ficha de detalles del socios.

	2. Al final de la ficha de socio puede visualizarse el historial de pago del socio.



Gestión de subscripciones

CU_18	List subscriptions
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de todas las subscripciones activas
Precondiciones	Estar autenticado en el sistema.
Postcondiciones	Se muestra en pantalla todas las subscripciones actualmente existentes.
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Listar subscripciones”. 2. El sistema muestra una tabla con todas las subscripciones registradas.

CU_19	Add subscription
Actor principal	Usuario administrador.
Objetivos	Añadir una nueva subscripción al sistema
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. La subscripción queda registrada en el sistema

Escenario principal	1. El usuario hace clic sobre el enlace del menú “Nueva subscripción” o sobre el botón en la esquina superior derecha del listado de subscripciones. 2. El sistema muestra el formulario de alta que el usuario debe completar con los campos marcados como obligatorios. 3. El usuario completa el formulario y pulsa guardar. 4. La subscripción queda registrada. 5. Se crea el primer periodo de cobro para la nueva subscripción. 6. Se retorna a la lista de subscripciones.
Extensiones	CU_25
Escenario alternativo	1. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error.

CU_20 Edit subscription	
Actor principal	Usuario administrador.
Objetivos	Modificar los datos de una subscripción existente.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de subscripciones.
Postcondiciones	1. Los datos de la subscripción quedan modificados.
Escenario principal	1. El usuario hace clic sobre el icono de “modificar subscripción” ubicado en la barra de herramientas a la derecha de los datos de la subscripción en la tabla o listado de subscripciones. 2. Se carga el formulario de datos de subscripciones con los datos de ésta. 3. Se modifican los datos y se pulsa el botón de guardar. 4. Los datos quedan guardados. 5. Se retorna a la lista de subscripciones.

CU_21	**Remove subscription**
Actor principal	Usuario administrador.
Objetivos	Eliminar una subscripción del sistema.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de usuarios subscripciones.
Postcondiciones	1. La subscripción es eliminada
Escenario principal	1. El usuario hace clic sobre el icono de “eliminar subscripción” ubicado en la barra de herramientas a la derecha de los datos de la subscripción en la tabla o listado de subscripciones. 2. Se le presenta al usuario con una alerta de confirmación de acción.

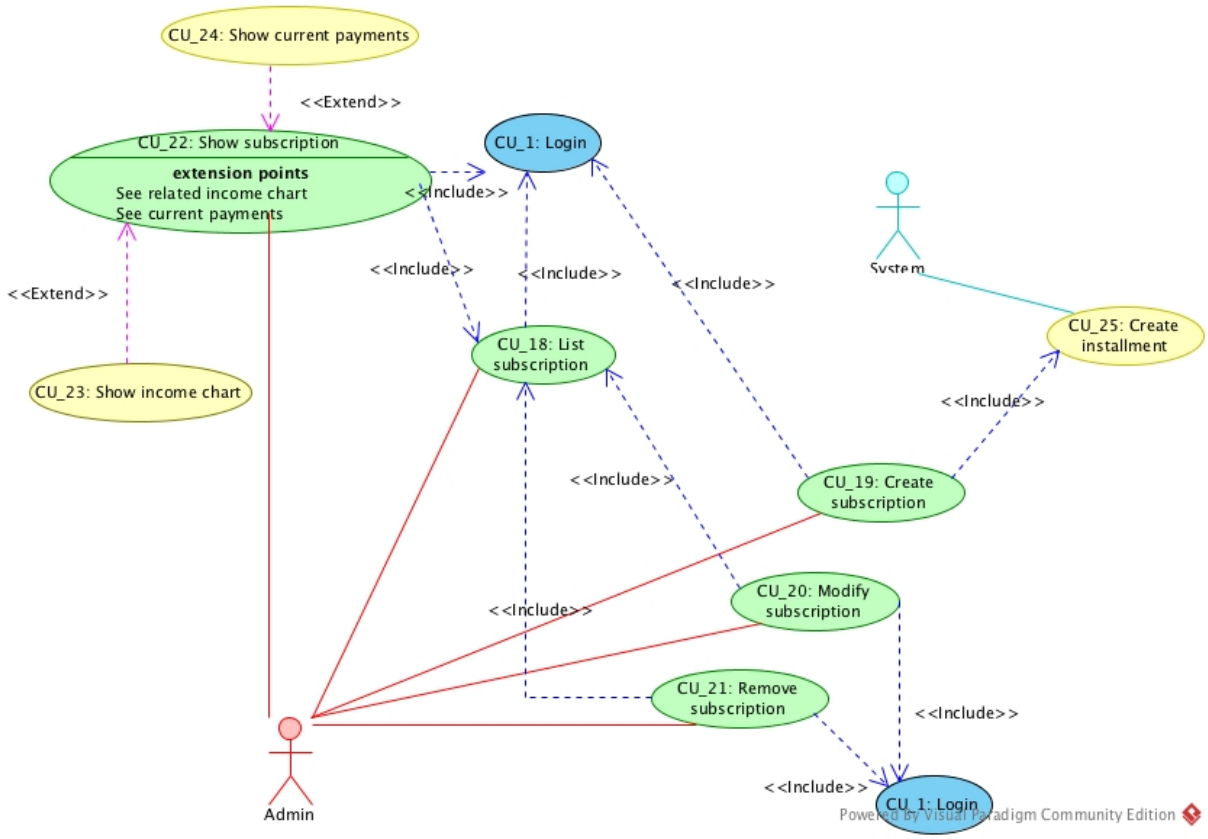
	3. El usuario confirma la acción. 4. Se elimina la subscripción. 5. Se retorna a la lista de subscripciones.

CU_22	**Show subscription**
Actor principal	Usuario administrador.
Objetivos	Ver la ficha de una subscripción
Precondiciones	Estar autenticado en el sistema. Acceder al listado de subscripciones.
Postcondiciones	1. La información de la subscripción se muestra en pantalla
Escenario principal	1. El usuario hace clic sobre el icono de "ver subscripción" ubicado en la barra de herramientas a la derecha de los datos de la subscripción en la tabla o listado de subscripciones. 2. Se le presenta al usuario con la ficha de la subscripción, desde donde podrá ver los detalles de la misma, los ingresos por mes.
Extensiones	CU_23, CU_24

CU_23	**Show income chart**
Actor principal	Usuario administrador.
Objetivos	Ver los ingresos de una subscripción por mes
Precondiciones	Estar autenticado en el sistema. Acceder al detalle de una subscripción
Postcondiciones	Un gráfico de barras muestra los ingresos devengados de los últimos 12 meses.
Escenario principal	1. El usuario accede a la ficha de detalle de la subscripción. 2. Accede a la gráfica

CU_24	**Show current payments**
Actor principal	Usuario administrador.
Objetivos	Ver pagos realizados de una subscripción
Precondiciones	Estar autenticado en el sistema. Acceder al detalle de una subscripción
Postcondiciones	Se muestra un listado de periodos de pagos y los pagos registrados.
Escenario principal	1. El usuario accede a la ficha de detalle de la subscripción. 2. Accede al listado
Escenario alternativo	1. Usuario administrador.

CU_25	Create installment
Actor principal	Sistema
Objetivos	Crear un periodo de cobro para una subscripción.
Precondiciones	Estar autenticado en el sistema. Haber creado una nueva subscripción o haber caducado el periodo anterior de cobro.
Postcondiciones	1. Un nuevo periodo de cobro se ha creado y notificado a los socios inscritos.
Escenario principal	1. Se crea una nueva subscripción 2. El sistema genera un nuevo periodo
Escenario alternativo	2. El cron job corre. 3. Se detectan aquellas subscripciones cuyo ciclo de periodicidad/recursividad esté por cumplirse. 4. Se genera un nuevo periodo. 5. Se notifican a los socios inscrito en la subscripción de que hay un nuevo periodo por cobrar.



Gestión de mensajería y comunicación

CU_28	Create notification template
Actor principal	Usuario administrador.
Objetivos	Crear un nueva plantilla de mensajes
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. La nueva plantilla queda registrada en la base de datos
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Nueva plantilla” o sobre el botón en la esquina superior derecha del listado de plantillas. 2. El sistema muestra el formulario de creación de plantillas que el usuario debe completar con los campos marcados como obligatorios. 3. El usuario completa el formulario y pulsa guardar. 4. La plantilla queda registrada.. 5. Se retorna a la lista de plantillas.
Escenario alternativo	1. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error.

CU_29	Edit notification template
Actor principal	Usuario administrador.
Objetivos	Modificar los datos de una plantilla
Precondiciones	Estar autenticado en el sistema. Acceder al listado de plantillas.
Postcondiciones	1. Los datos de la plantillas quedan modificados.
Escenario principal	1. El usuario hace clic sobre el icono de “modificar plantillas” ubicado en la barra de herramientas a la derecha de los datos de la plantillas en la tabla o listado de plantillas. 2. Se carga el formulario de plantillas con los datos de ésta. 3. Se modifican los datos y se pulsa el botón de guardar. 4. Los datos quedan guardados. 5. Se retorna a la lista de plantillas.
Escenario alternativo	1. Se pulsa el botón guardar tras no proveer toda la información solicitada como obligatoria. 2. El sistema retorna al usuario al formulario y le pide que rectifique el error.

CU_30	List notification template

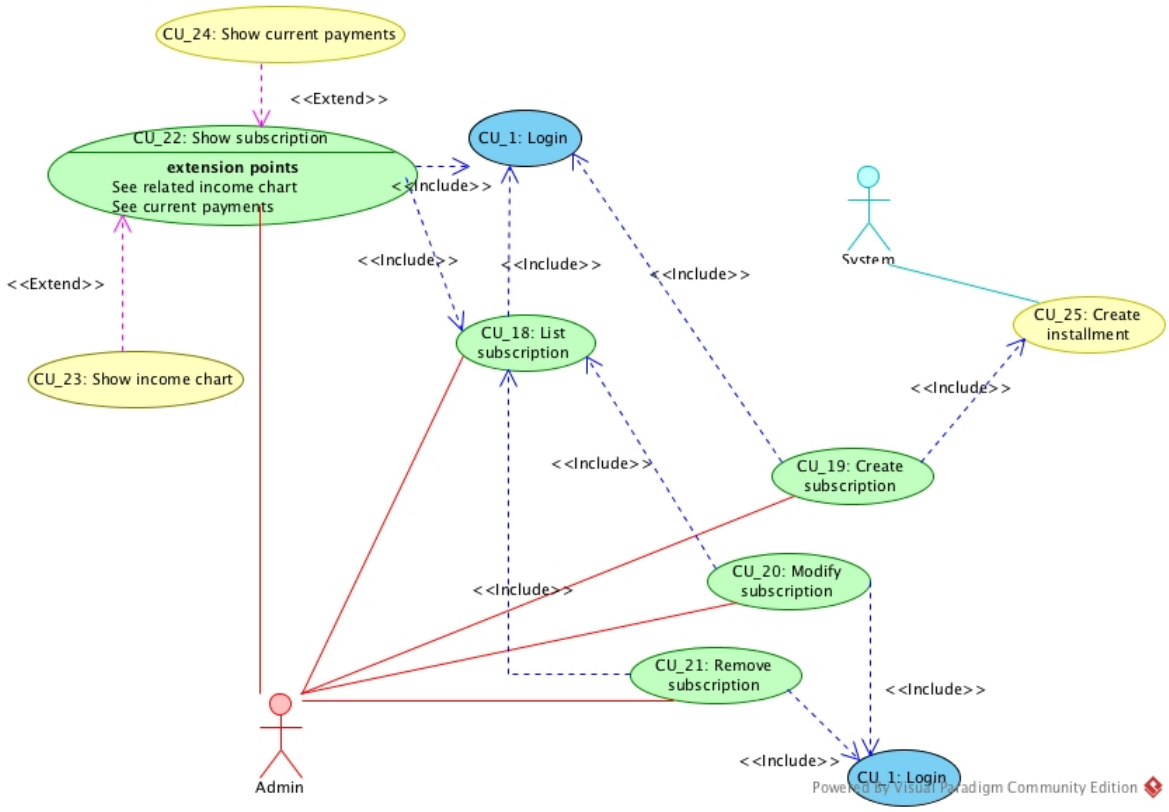
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de todas las plantillas de mensajes
Precondiciones	Estar autenticado en el sistema.
Postcondiciones	Se muestra en pantalla todas las plantillas de comunicaciones del sistema actualmente registradas.
Escenario principal	1. El usuario hace clic sobre el enlace del menú “Listar plantillas”. 2. El sistema muestra una tabla con todas las plantillas registradas.

CU_31	**Delete notification template**
Actor principal	Usuario administrador.
Objetivos	Eliminar una plantilla de comunicaciones.
Precondiciones	Estar autenticado en el sistema. Acceder al listado de plantillas.
Postcondiciones	Las plantilla queda eliminada
Escenario principal	1. El usuario hace clic sobre el icono de “eliminar plantilla” ubicado en la barra de herramientas a la derecha de los datos del usuario en la tabla del listado de plantillas. 2. Se le presenta al usuario con una alerta de confirmación de acción. 3. El usuario confirma la acción. 4. Se elimina la plantilla. 5. Se retorna a la lista de plantilla.
Escenario alternativo	El usuario no confirma la acción de eliminar la plantilla. Se retorna a la lista de plantilla.

CU_32	**Send mail from template**
Actor principal	Usuario administrador.
Objetivos	Enviar un email usando una plantilla de mensajes
Precondiciones	Estar autenticado en el sistema. Acceder al listado de plantillas de mensajes
Postcondiciones	1. El mensaje queda enviado con la plantilla de mensaje
Escenario principal	1. El usuario accede al listado de plantillas 2. El usuario hace clic en el botón de enviar correo
Extensiones	CU_33

CU_33	**Send notification**
Actor principal	Usuario administrador.
Objetivos	Enviar un mensaje a uno o mas socios
Precondiciones	1. Estar autenticado en el sistema.

Postcondiciones	1. Se envía un mensaje a uno o mas miembros
Escenario principal	1. El usuario hace clic sobre el enlace de “Enviar comunicado” ubicado en el menú de la izquierda del interface. 2. Se despliega el formulario de creación de notificaciones. 3. El usuario completa el formulario 4. El usuario selecciona los o el destinatario 5. El usuario hace clic en el botón de enviar 6. El correo se envía 7. Se retorna a la pagina previa desde donde se accedió al formulario de envío de mensajes.



Estadísticas

CU_34	Show payment by periods
Actor principal	Usuario administrador.
Objetivos	Mostrar un gráfico de ingresos por mes
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. Se muestra el listado de periodos
Escenario principal	1. El usuario accede al *dashboard* desde el menú principal 2. El gráfico de imgresos aparece en pantalla

CU_34	Show latest payments
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de los últimos pagos realizados
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. Se muestra el listado de periodos
Escenario principal	1. El usuario accede al *dashboard* desde el menú principal 2. El listado de pagos activos aparece en pantalla

CU_36	Show latest installments
Actor principal	Usuario administrador.
Objetivos	Mostrar un listado de periodos de pago
Precondiciones	1. Estar autenticado en el sistema.
Postcondiciones	1. Se muestra el listado de periodos
Escenario principal	1. El usuario accede al *dashboard* desde el menú principal 2. El listado de periodos activos aparace en pantalla

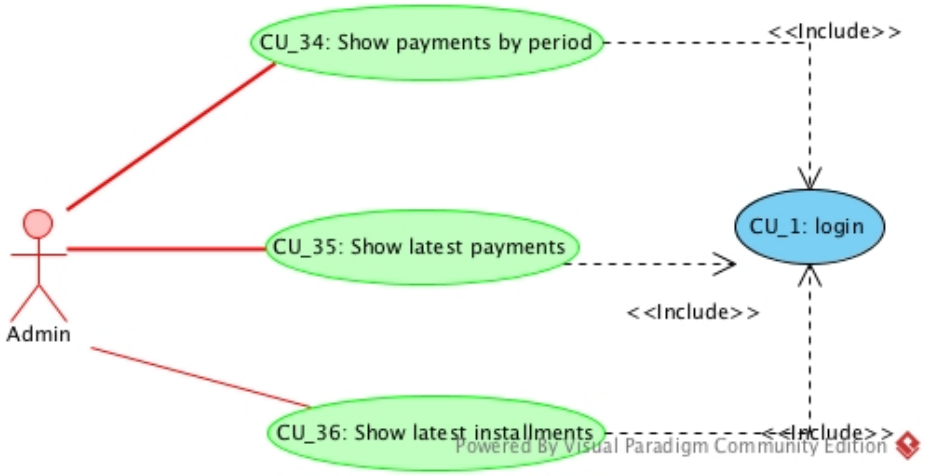


Diagrama de Clases

La clase *User* representa la entidad usuario de la que heredan las clases *UserAdmin* y *UserMember*. La idea de relacionar ambas entidades mediante una herencia es la poder, en un futuro, escalar el sistema permitiendo que los socios puedan acceder a la plataforma para hacer uso de servicios y prestaciones extras.

El sistema central del diagrama está compuesto por la clase *Subscription*, los periodos de pagos o *Installment,* las cuotas de miembros o *MemberInstallments* y los pagos o *Payment*. La clase *MemberInstallment* es una clase de tipo asociativa que contiene información de las cuotas de cada usuario por subscripción y periodo de pago. Los usuarios socios podrán realizar pagos múltiples, o un sólo pago, a un periodo concreto hasta que éste quede cubierto. En este diagrama también puede apreciarse que un socio puede estar subscrito a más de una subscripción.

Por ultimo tenemos el sistema de mensajería. Este sistema está representado por la clase asociativa *Message* que comprende el mensaje a enviar. Esta clase no necesita persistencia de datos ya que el sistema no almacena los mensajes enviados; si embargo, si es posible en un futuro escalar el sistema a niveles donde la gestión y el envío de mensajes sea más completa y gestionable. La clase *MessageTemplate* contiene textos predeterminados utilizados como plantillas para la generación de los mensajes.

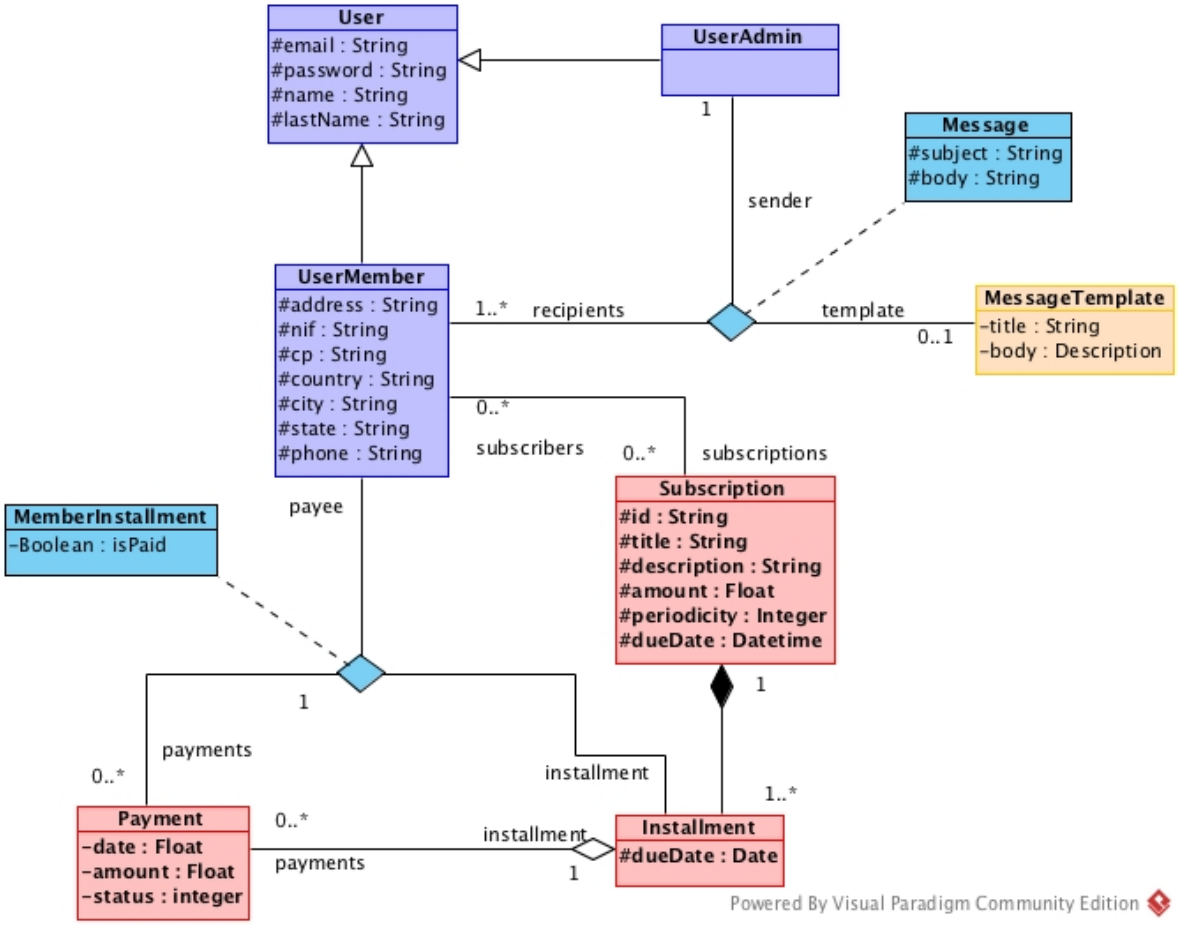

User
#email : String
#password : String
#name : String
#lastName : String
UserAdmin
1
Message
#subject : String
#body : String
sender
UserMember
#address : String
#nif : String
#cp : String
#country : String
#city : String
#state : String
#phone : String
1..*
recipients
template
0..1
MessageTemplate
-title : String
-body : Description
0..*
subscribers
0..*
subscriptions
payee
MemberInstallment
-Boolean : isPaid
Subscription
#id : String
#title : String
#description : String
#amount : Float
#periodicity : Integer
#dueDate : Datetime
1
1
payments
0..*
installment
1..*
Payment
-date : Float
-amount : Float
-status : integer
0..*
payments
installment
1
Installment
#dueDate : Date
Powered By Visual Paradigm Community Edition

Diagrama de Componentes

A continuación de definen los diagramas de componentes basados en la descripción de componentes descrita en la descripción general del sistema. Los diagramas siguen el patrón de diseño estructural Modelo-Vista-Controlador para aplicaciones JavaEE.

Los diagramas presentados muestran el sistema en diferentes niveles de granuralidad, comenzando por una vista global donde puede apreciarse la integración de las tres capas del modelo MVC ínteractuando entre sí a través de sus componentes.

Con la finalidad de encapsular y delegar la responsabilidad de las acciones en servicios o clases concretas, cada componente puede contener uno o mas interfaces de acceso según la naturaleza de las funciones que contenidas.

Diagrama general de componentes

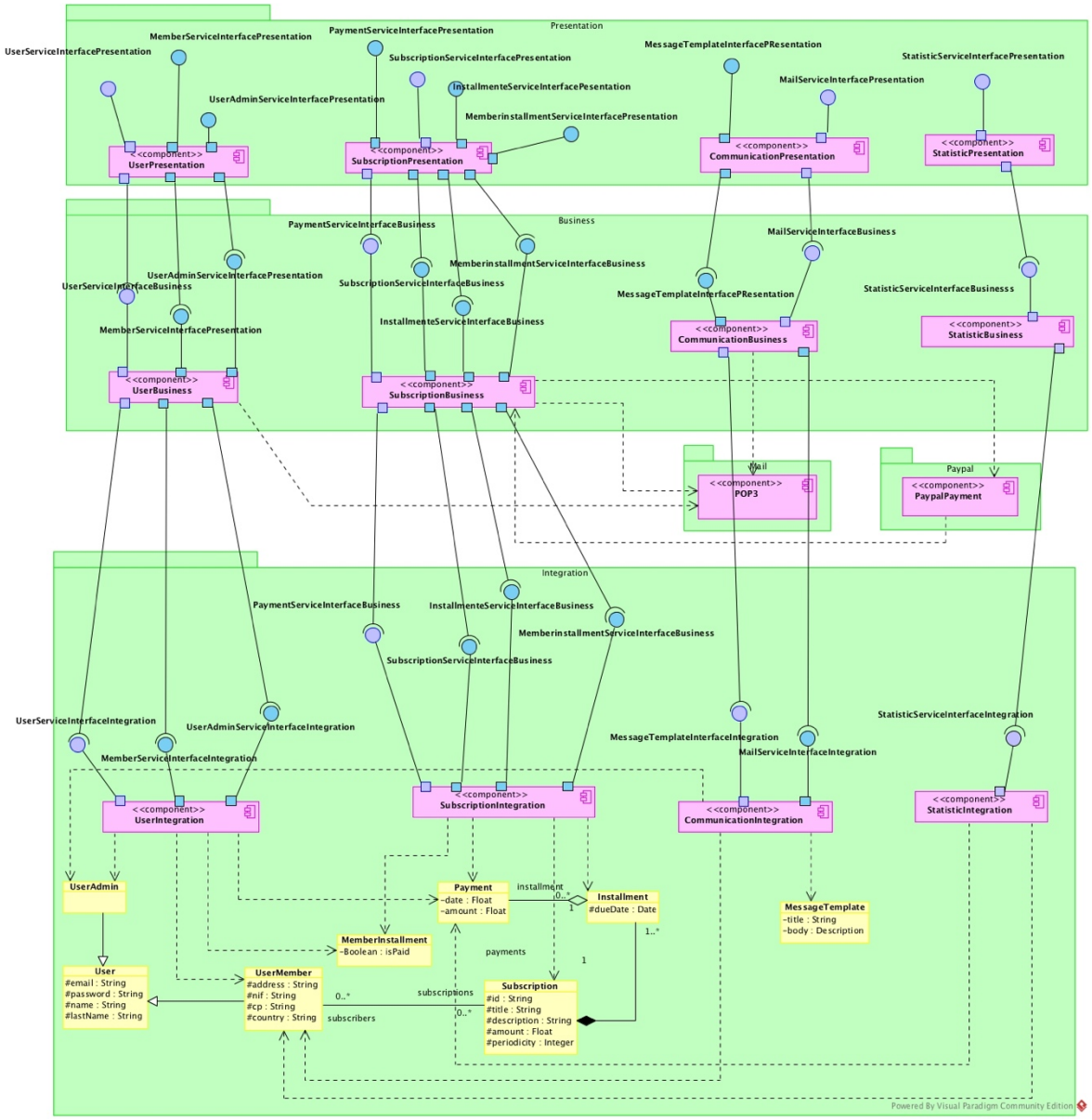
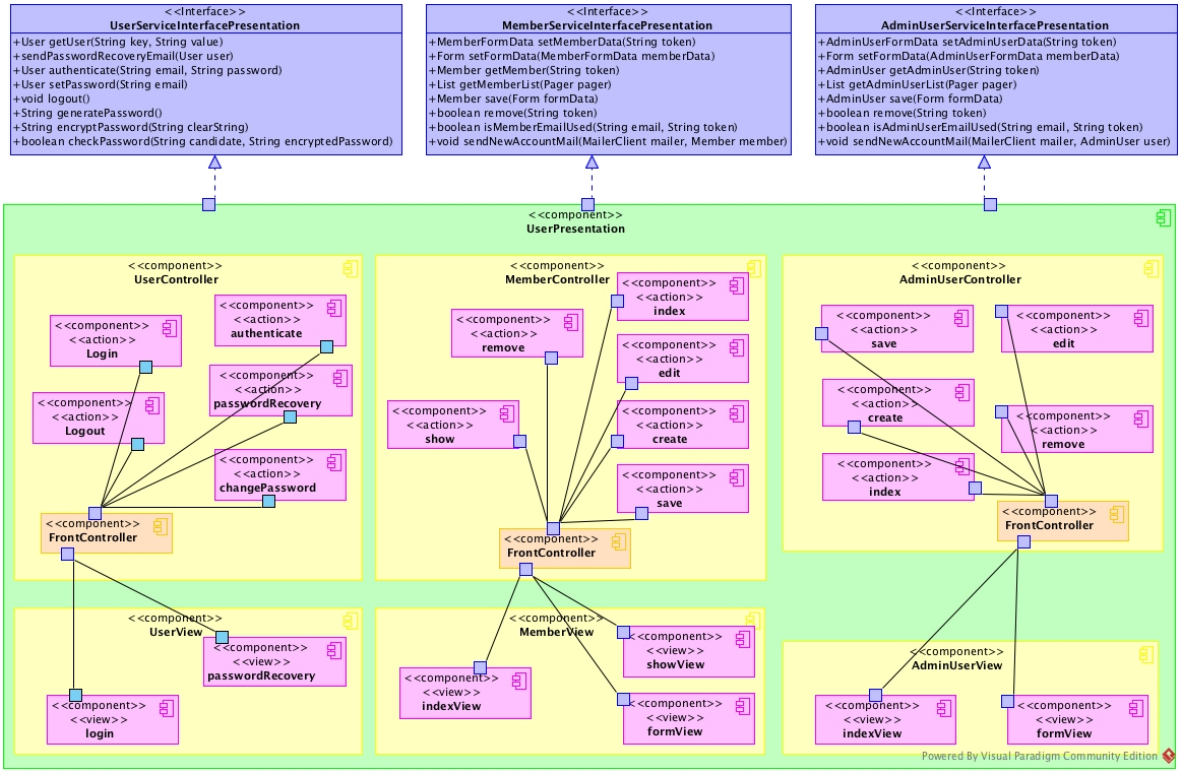
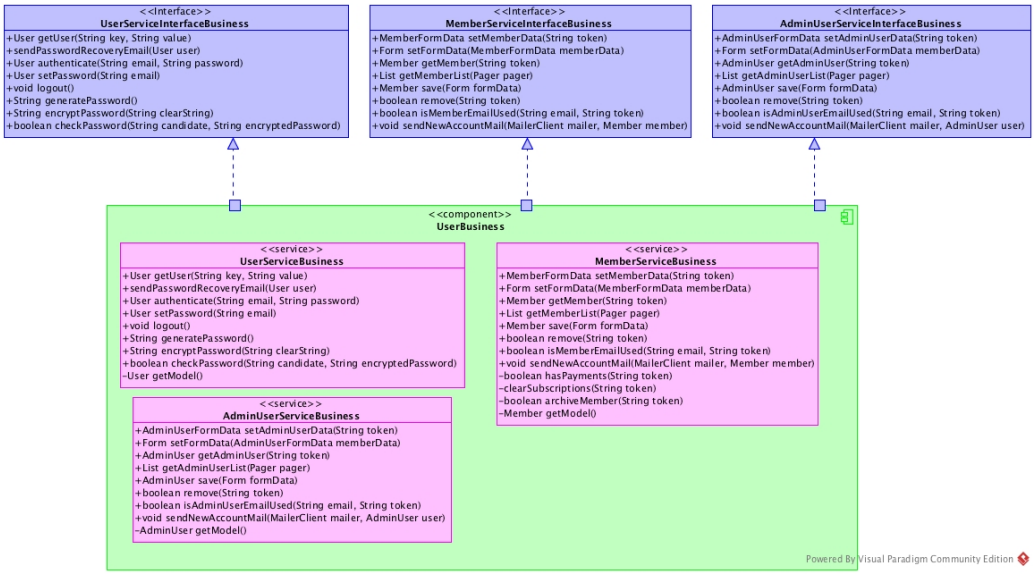


Diagrama de componentes usuarios

Capa de presentación



Capa de negocios



Capa de integración

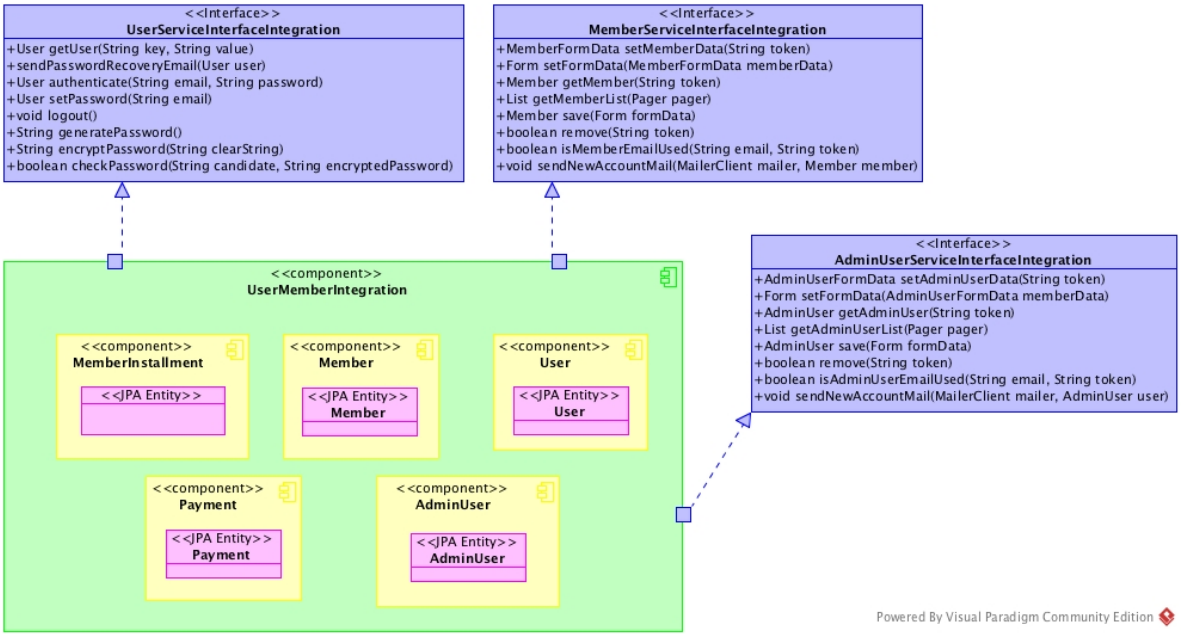
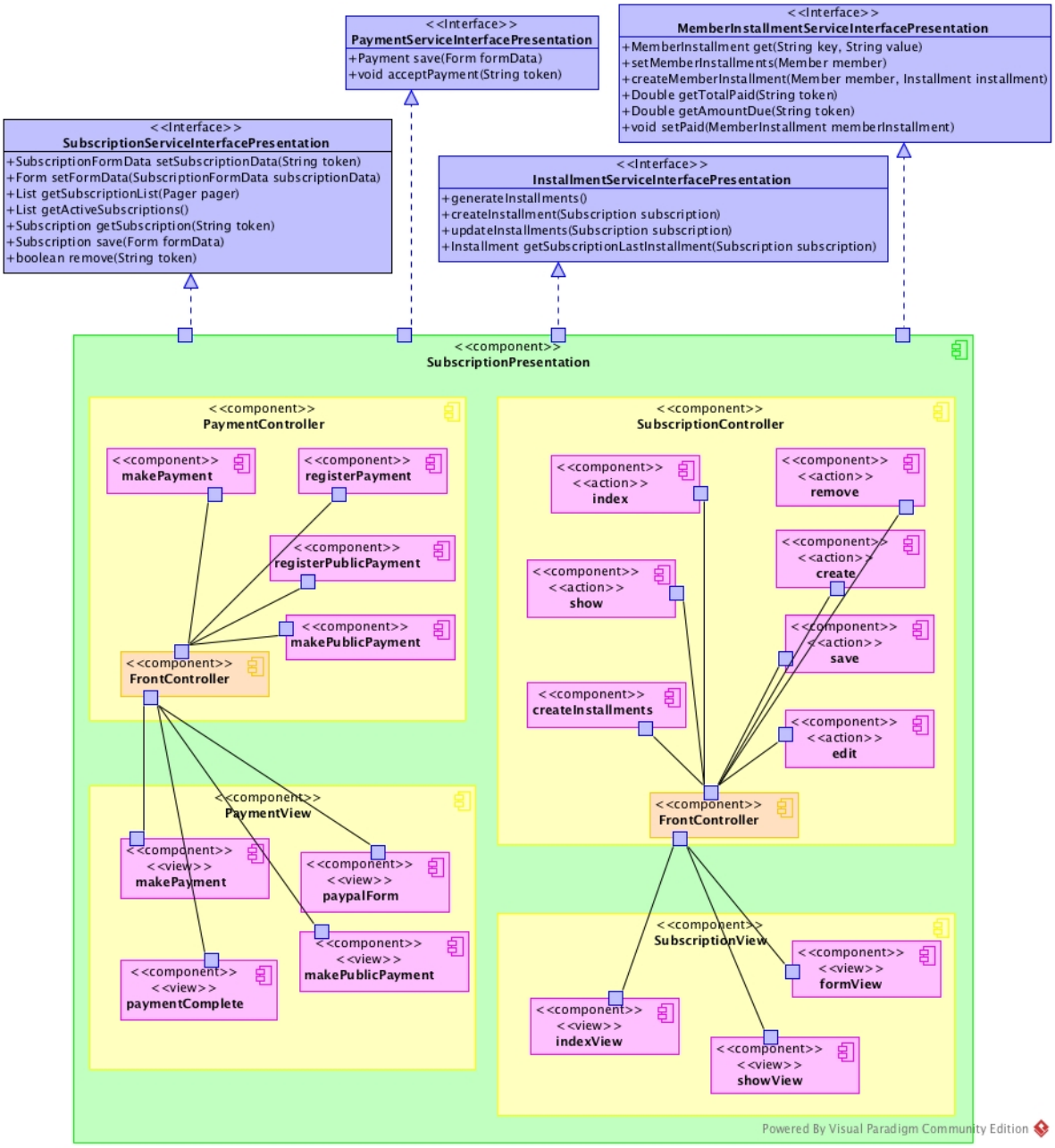
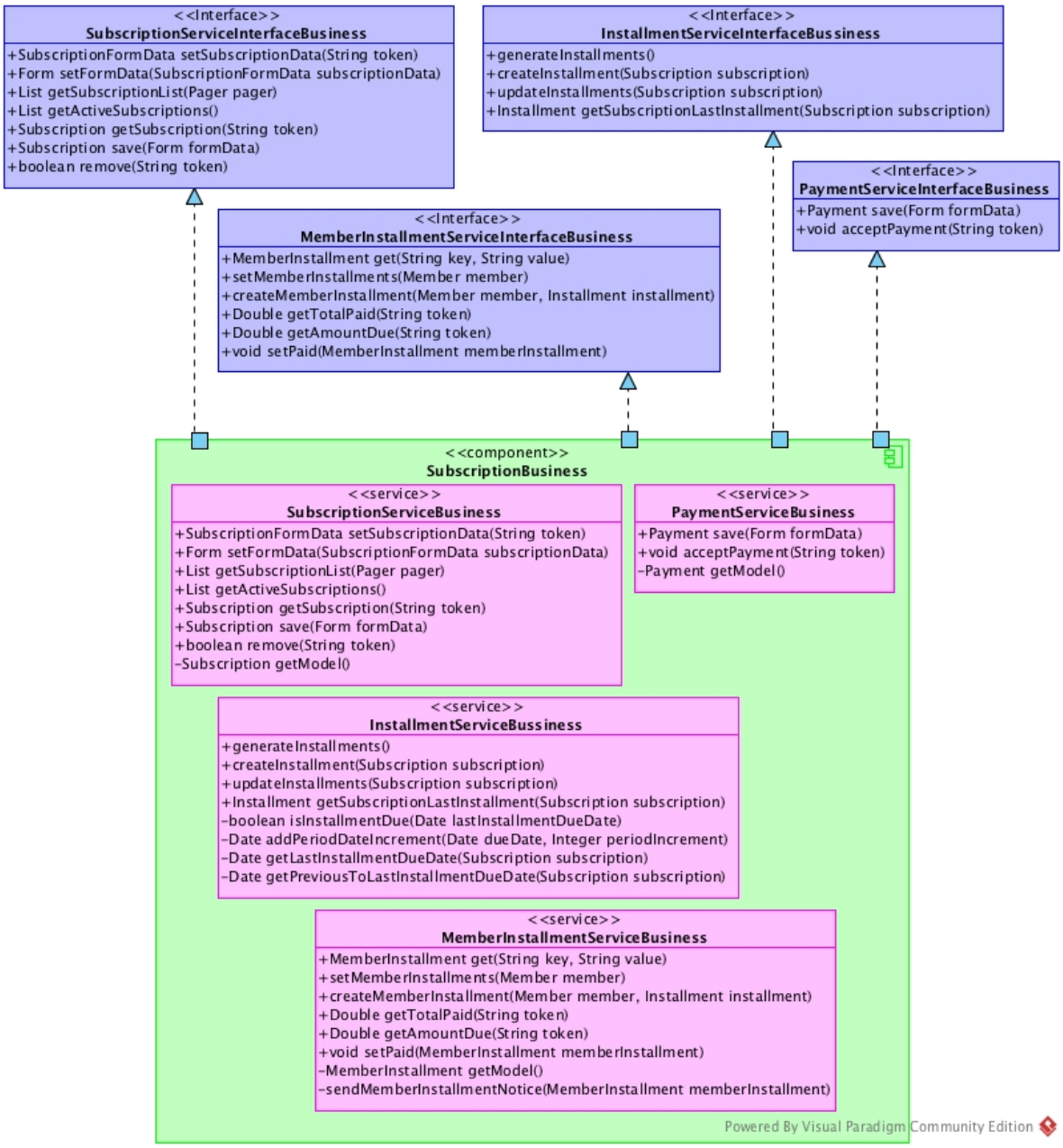


Diagrama de componentes subscripción

Capa de presentación



Capa de negocios



Capa de integración

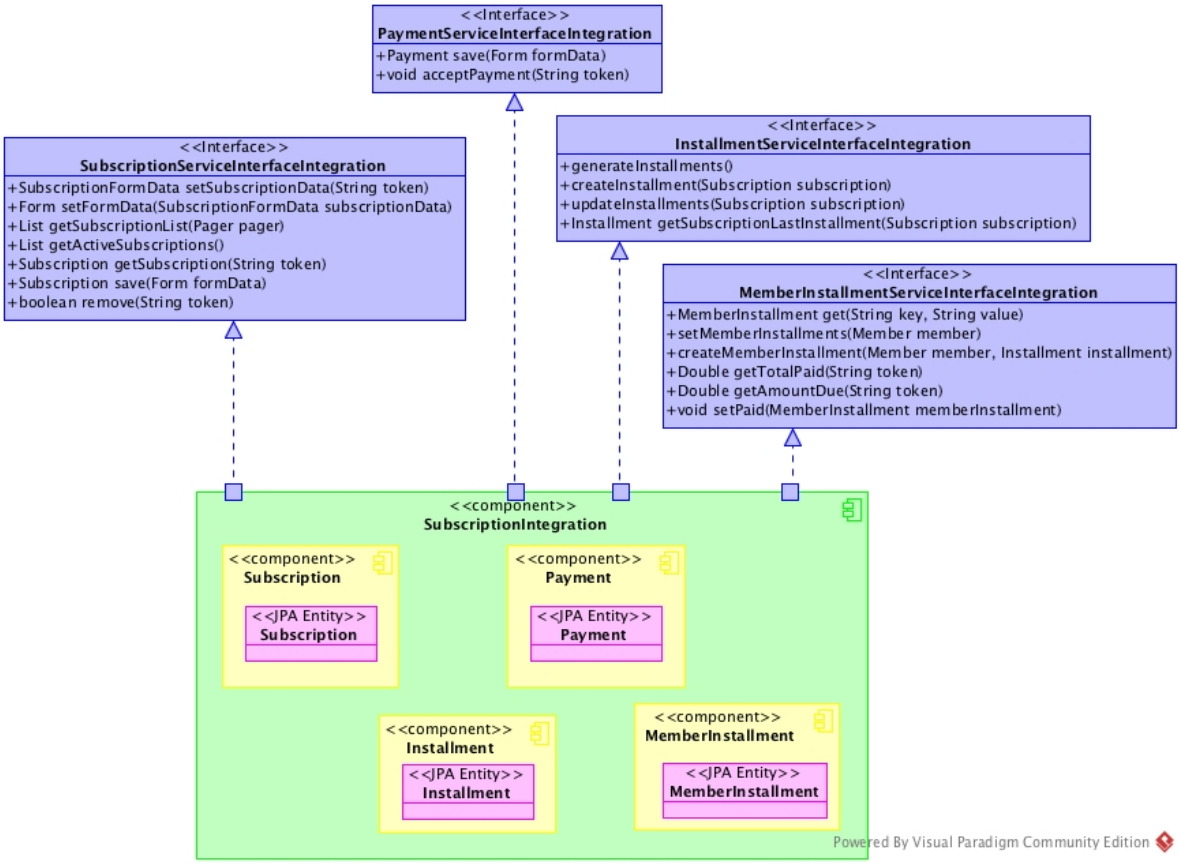
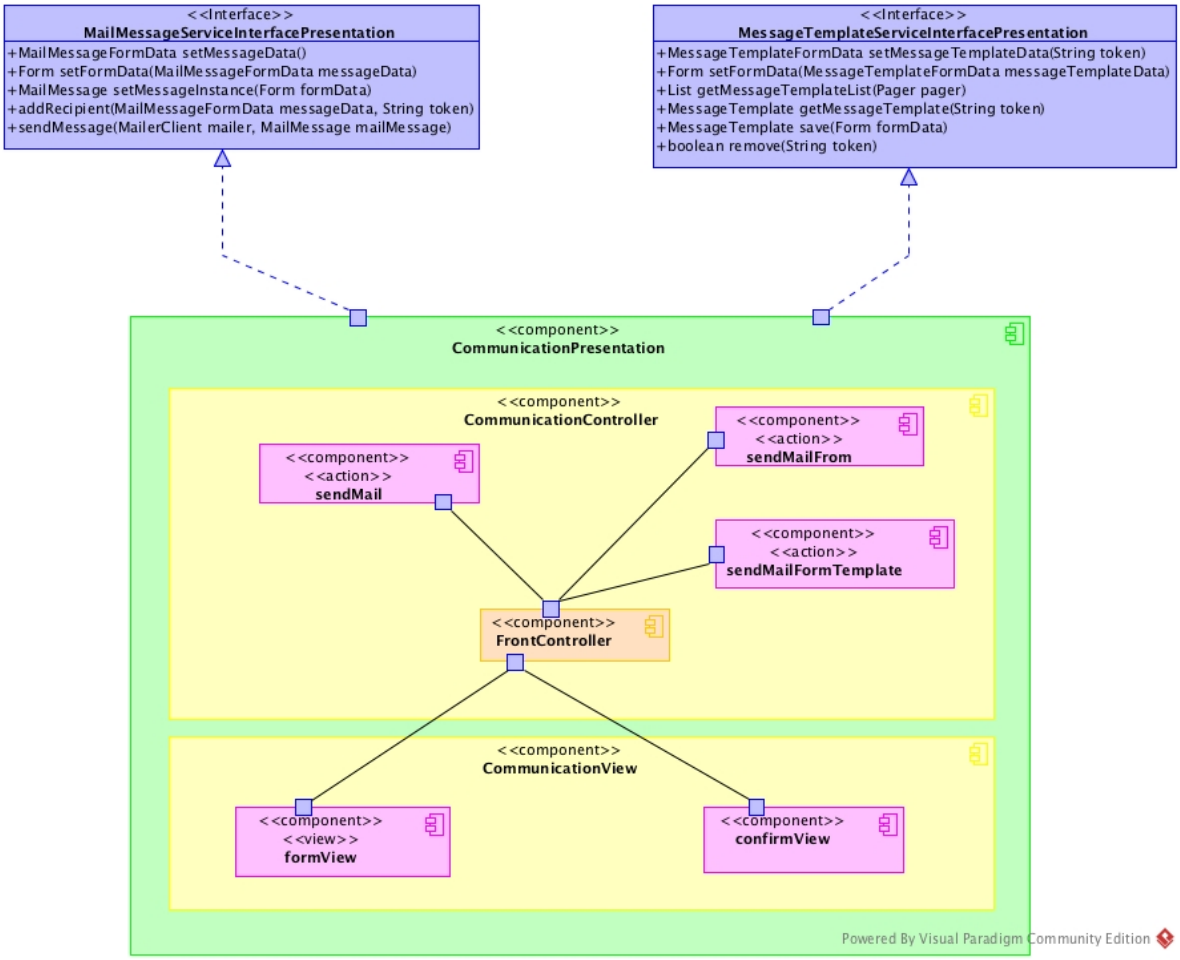
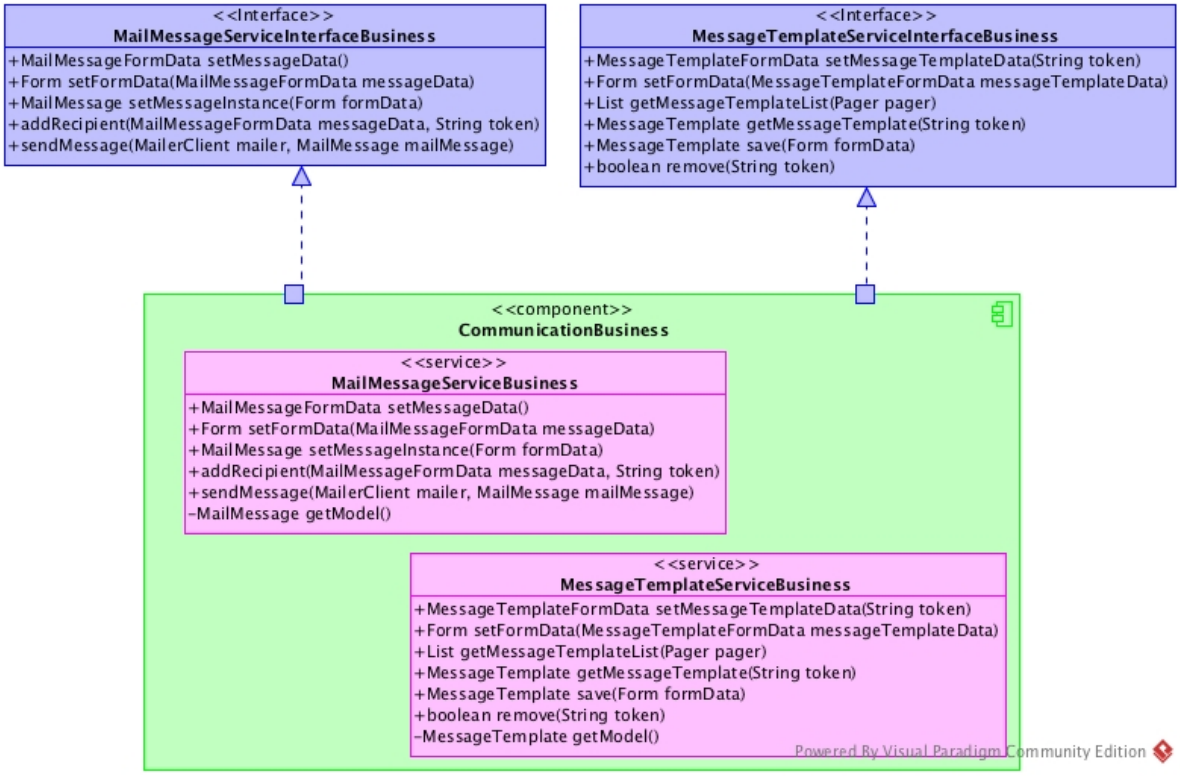


Diagrama de componentes comunicación

Capa de presentación



Capa de negocios



Capa de integración

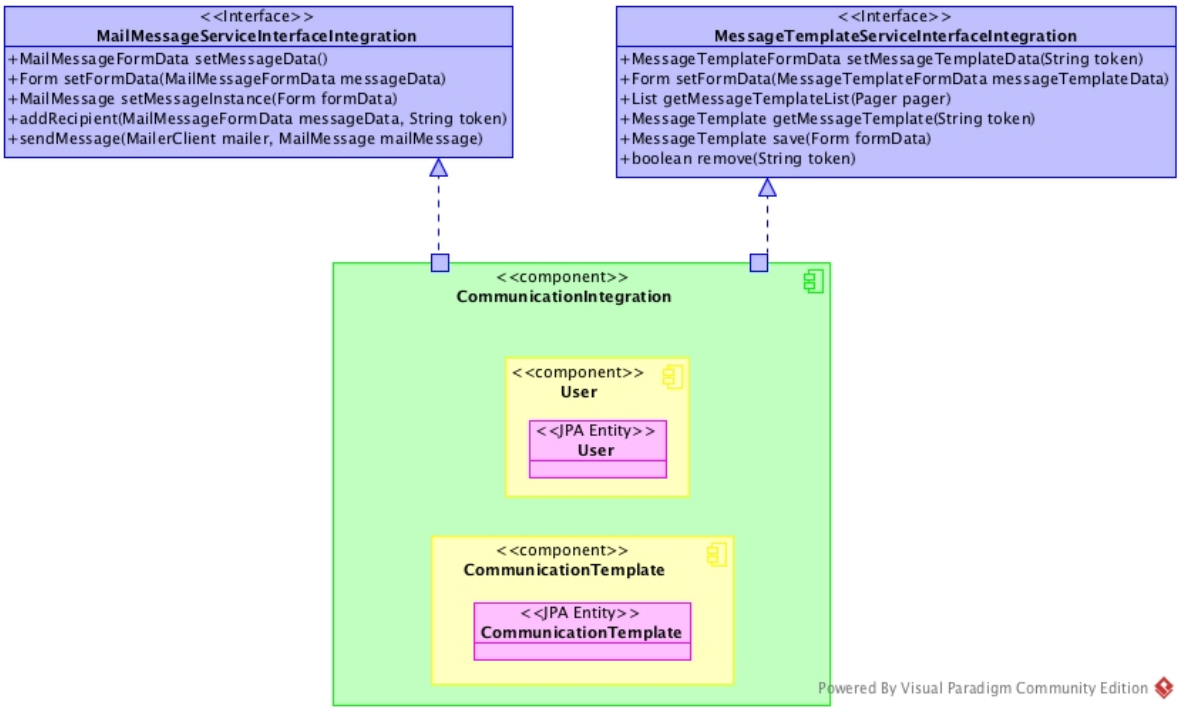
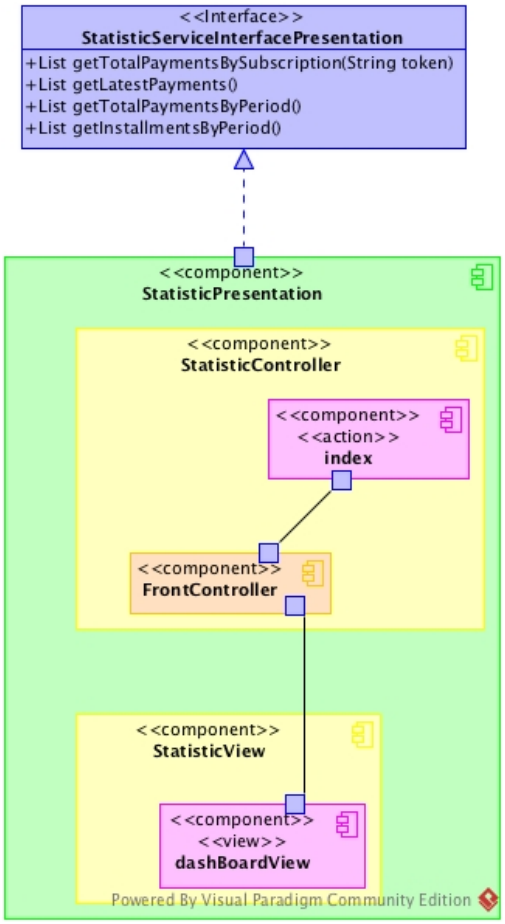
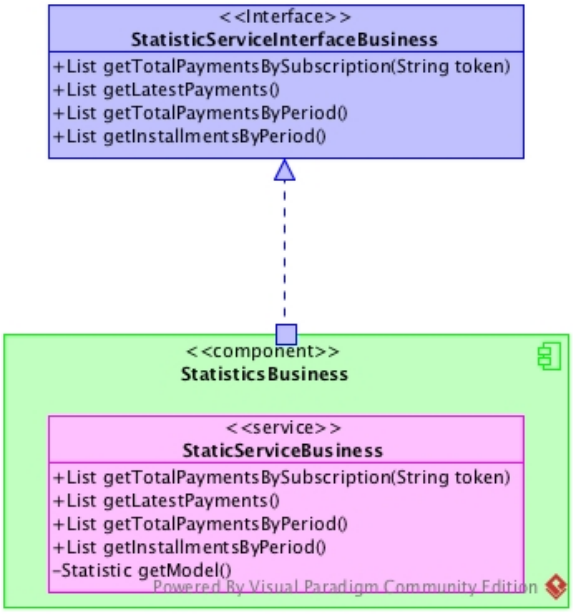


Diagrama de componentes estadísticas

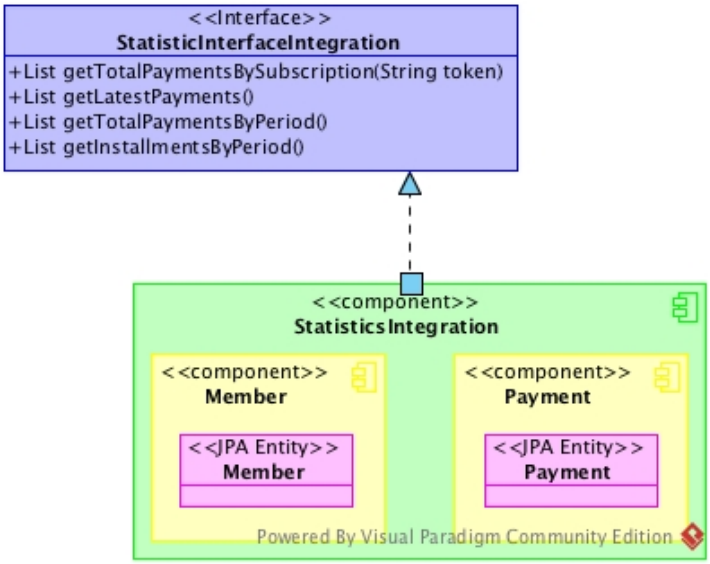
Capa de presentación



Capa de negocios



Capa de integración



La interfaz de Usuario

La interfaz de usuario está implementada haciendo uso de la plantilla **AdminLTE Control Panel Template** basada en el framework CSS Bootstrap para proveer al sistema de adaptabilidad y capacidad de respuesta ante dispositivos móviles.

La interfaz esta compuesta por las siguientes vistas según el componente:

1. Usuarios
    a. Listado de usuarios administrativos
    b. Listado de usuarios miembros
    c. Formulario de usuarios administrativos
    d. Formulario de usuarios miembros
    e. Detalle de usuario miembro
2. Subscripciones
    a. Listado de subscripciones
    b. Formulario de subscripciones
    c. Formulario de pago
    d. Formulario de pago por enlace
    e. Detalle de subscripción
3. Comunicación
    a. Listado de plantillas de mensajes
    b. Formulario de plantilla de mensaje
    c. Formulario de envío de mensajes
    d. Plantilla de correo

A continuación se describirán e ilustrarán algunos de los elementos de la interface de usuario mas relevantes.

Plantilla general de la interfaz de usuario

Todas las vistas internas, o que requieren de autenticación para acceder a ellas, están montadas sobre la misma plantilla o diseño. Esta plantilla se divide en dos áreas principales. En la izquierda se ubican los elementos comunes tales como las señas de identidad del sistema, el menú de navegación y el resumen del perfil de usuario. En la zona derecha (o área de trabajo) se encuentran los elementos específicos para cada vista; estos elementos, por lo general, pueden ser listados, formulario u hojas de detalles.
Las vistas externas no tienen elementos comunes de plantillas y toda la plantilla es área de trabajo.

Portal Principal

Una vez dentro del sistema, el usuario puede acceder al portal principal de la aplicación. Este portal esta compuesto por dos zonas principales. La primera de ellas es el panel de control, que es común en todas las secciones de la aplicación, y está ubicado a la izquierda y parte superior de la interface de usuario.

En la sección del panel de control de la izquierda podemos localizar, comenzado por arriba, la información del usuario actualmente identificado en el sistema. Aquí podemos ver si nombre y su foto **Gravatar** (en el caso de que tuviese). Acto y seguidos tenemos el buscador de contenidos y por ultimo el menú de navegación.

El menú esta organizado por componentes y es del tipo desplegable. Cada elemento del menú de navegación contiene enlaces a **los** principales puntos de entrada de cada componente.

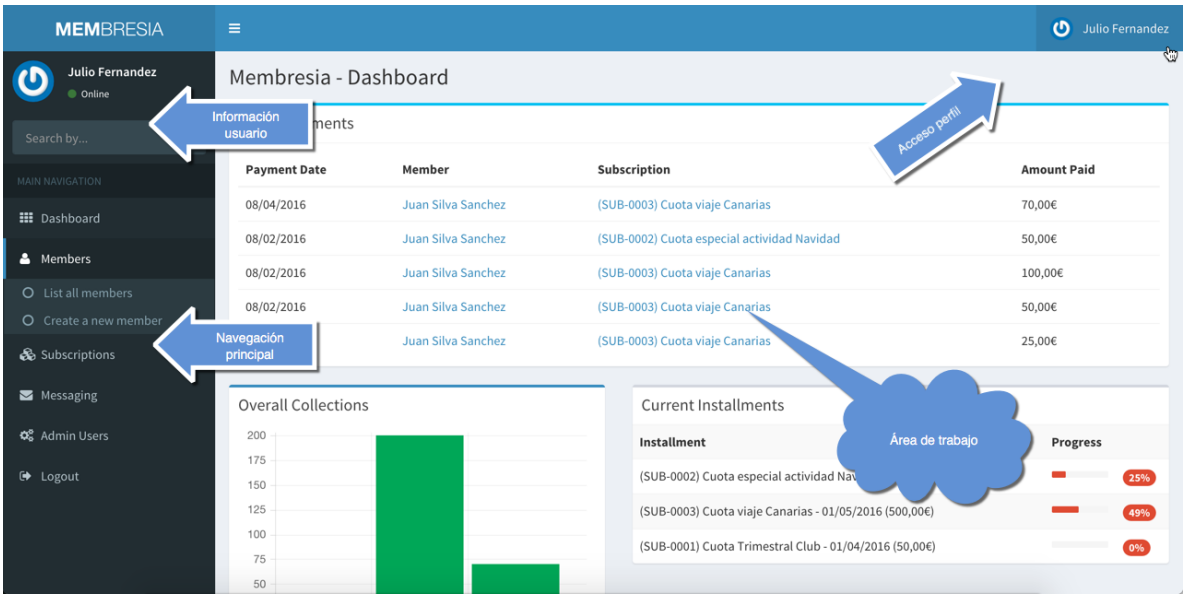
En la barra superior derecha tenemos el acceso al perfil de usuario. Desde aquí el usuario puede modificar su perfil o salir del sistema.

Por ultimo, cabe mencionar que el menú de navegación, así como el panel de control que lo contiene, puede contraerse para maximizar el área de trabajo. Esta funcionalidad es un indicativo de la capacidad de adaptación del sistema a dispositivos móviles de cualquier tipo, tamaño y resolución.

En al área central o de trabajo, también denominada DASHBOARD, tenemos una serie cajas que contienen información analítica del sistema y de su estado actual. La primera de ellas es un listado de los diez últimos pagos registrados. Cada entrada está ilustrada por el socio que ha realizado el pago, la subscripción y la cantidad abonada. Tanto el nombre del socio como el de la subscripción proveen un enlace que permiten acceder a la hoja de detalles de cada uno de ellos.

La siguiente caja es un gráfico de barras que ilustra la cantidad global recaudada por mes durante los últimos doce meses. Si pasamos el ratón por encima, podemos acceder a la información detallada del mes en cuestión.

Y por ultimo los periodos de cobro activos actualmente. Cada entrada esta acompañada de una barra de progreso, y un porcentaje, que muestra el total recaudado para dicho periodo de cobro del total esperado. Esta información es sumamente útil si queremos indagar en el progreso de un pago en concreto.

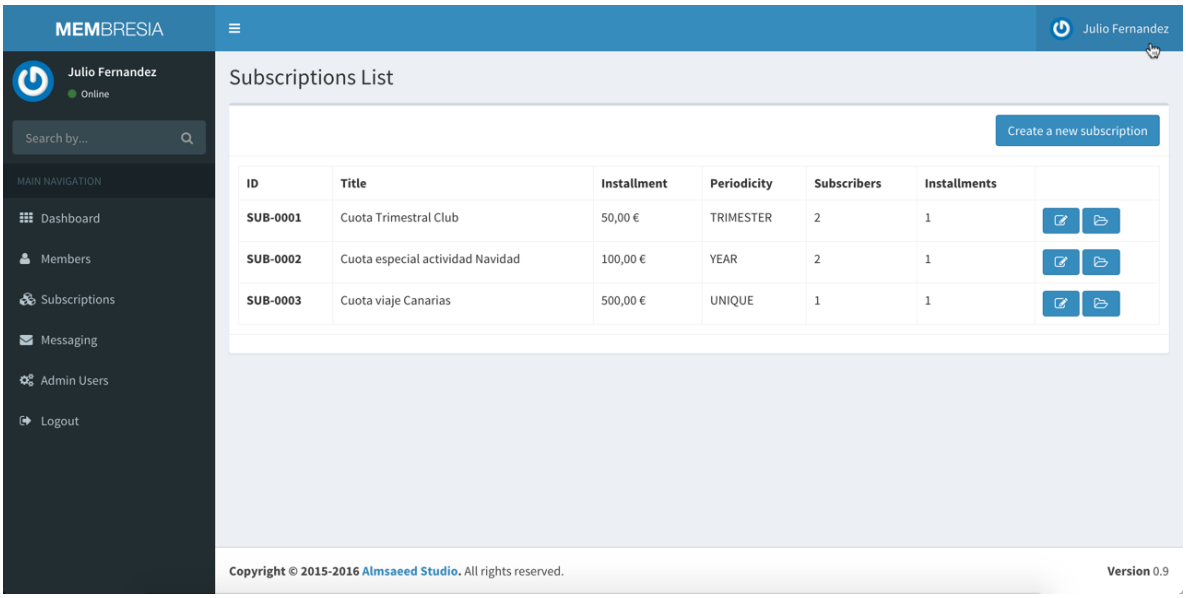


Vistas de listados

Las listas son el punto de partida para comenzar a operar en cualquiera de los componentes. El área de trabajo de una lista esta compuesta por el buscador en la parte superior, que puede variar según la naturaleza de los contenidos; y el listado, ubicado debajo del buscador. El listado a su vez se divide en dos zonas, que son la información del elemento listado y la barra de herramientas destinada a realizar acciones sobre el elemento al que esta relacionada.

La barra de herramientas tiene dos elementos comunes a todos los listados: editar y elimina elemento. El componente de socios, sin embargo, tiene asociada dos acciones adicionales a su barra de herramientas y que son de gran relevancia para este componente: realizar pago y enviar mensaje.

Otro elemento importante de la vista de listados es en el botón de crear un nuevo elemento. Esta acción está también accesible desde del panel de navegación de la derecha. Este elemento permite acceder al formulario correspondiente al componente del listado en modo creación.



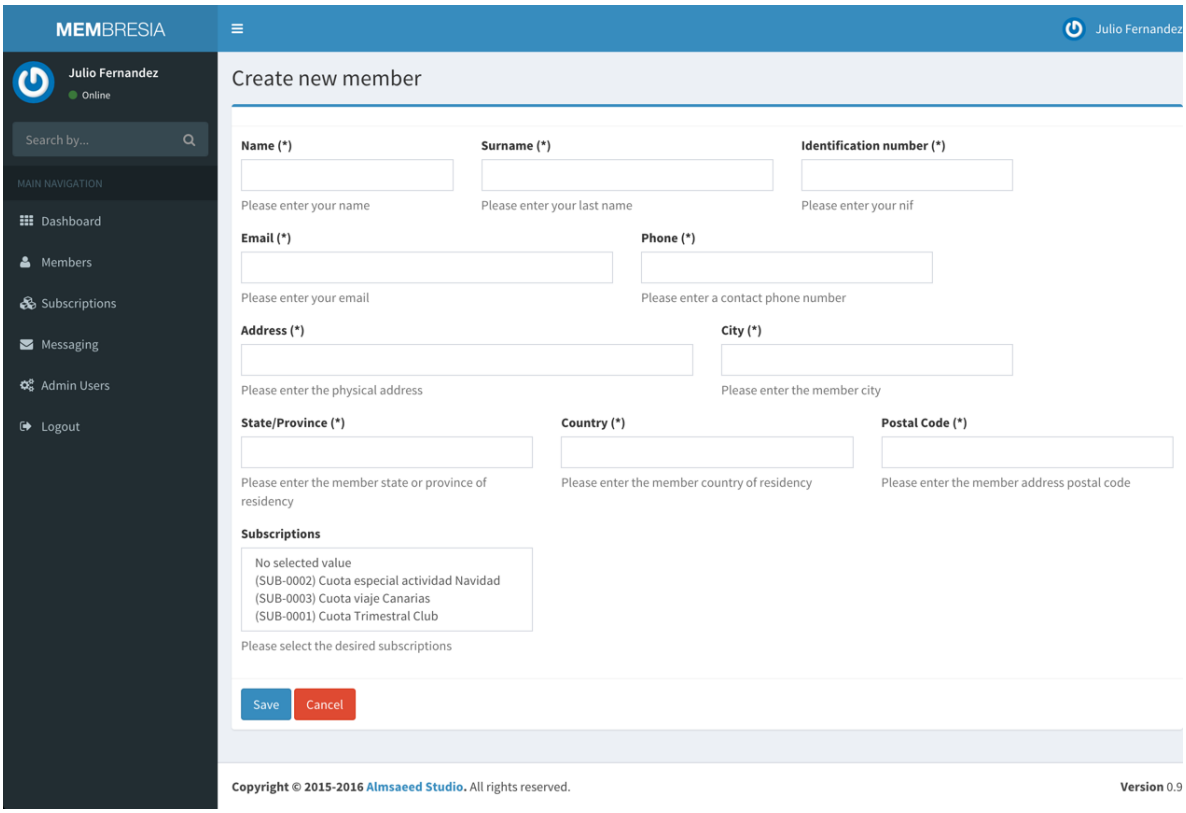
Vistas de formulario

Al igual que los listados, los formularios también hacen uso de la plantilla general de contenidos, aunque su composición es más variada y depende de la naturaleza del componente. Los formularios pueden servir tanto para crear elementos nuevos como para modificarlos. No existe una vista de formulario diferente para cada acción, una misma vista sirve para ambos propósitos y es la acción quien define su función.

Formulario de Socios

El formulario de socios está compuestos por dos secciones claramente diferenciadas por su funcionalidad en términos del tipo de información que gestiona. La primera de ellas es el formulario que recoge los datos personales y de contacto del socio tales como nombre y dirección o número de teléfono. La segunda gestiona las subscripciones a las que el usuario esta inscrito.

El selector de subscripciones es un control de selección múltiple desde donde se seleccionan las subscripciones a las que el socio deberá de abonar la cuota indicada.



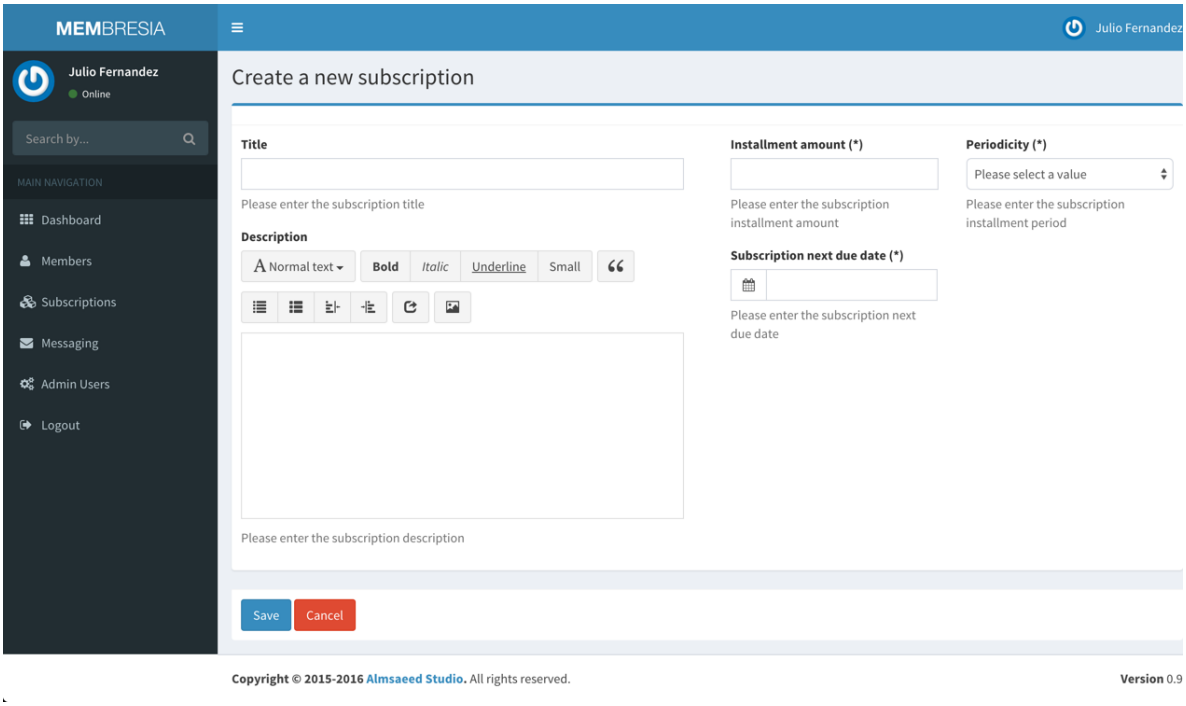
Formulario de subscripción

El modulo de subscripciones es sin duda alguna del corazón de la aplicación. Es este modulo el que crea las dependencias necesarias que representan los periodos de pago, los cobros a los socios subscriptores y los pagos.

La entidad subscripción es de carácter recurrente, es decir, que su ciclo de vida cíclico. La subscripción tiene periodicidad, cada vez que un periodo se vence se crea uno nuevo y se informa a los subscriptores de que existe un nuevo pago por realizar.

El formulario de subscripciones recolecta los datos necesarios para generar una subscripción. Los datos requeridos son el nombre o titulo , la cantidad a abonar en cada periodo de pago y la periodicidad. Esta última es de suma importancia ya que definirá cuando deben crearse los periodos de pagos y avisar a los socios con un nuevo email.

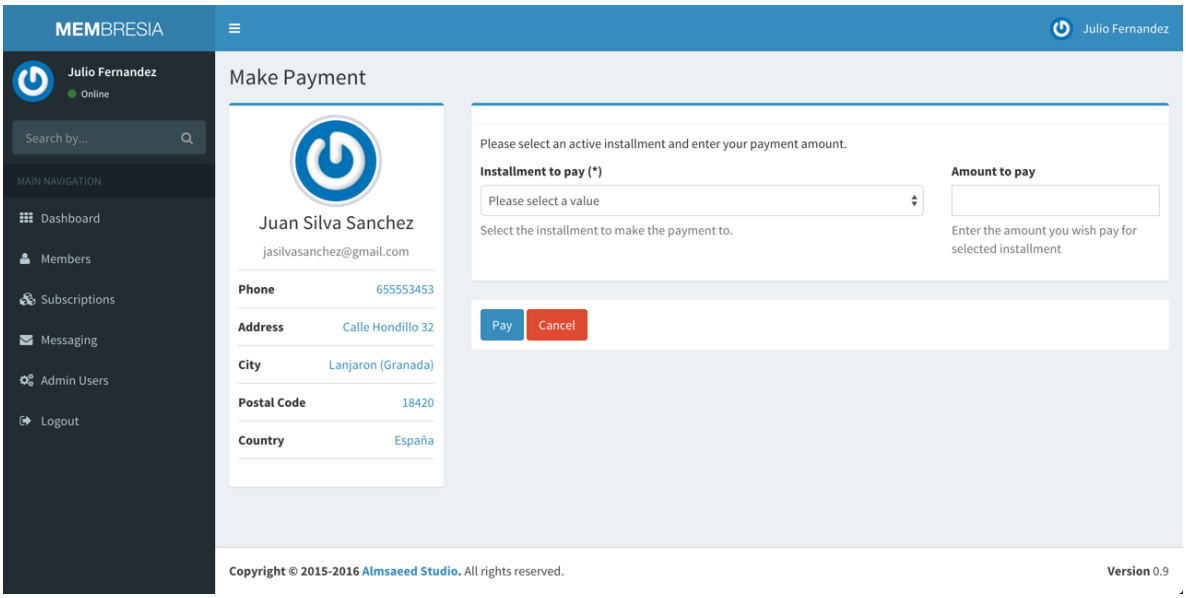
También es necesario indicar cuando se vence el próximo pago. Esta fecha se toma como referencia para crear los pagos y avisos futuros.



Formulario de registro de pago

El formulario de pagos permite realizar un pago parcial o competo sobre la cuota de una subscripción concreta. Para registrar un pago de forma manual es necesario indicar la subscripción y el periodo al cual se desea aplicar la cuota e introducir el total del importe a pagar.

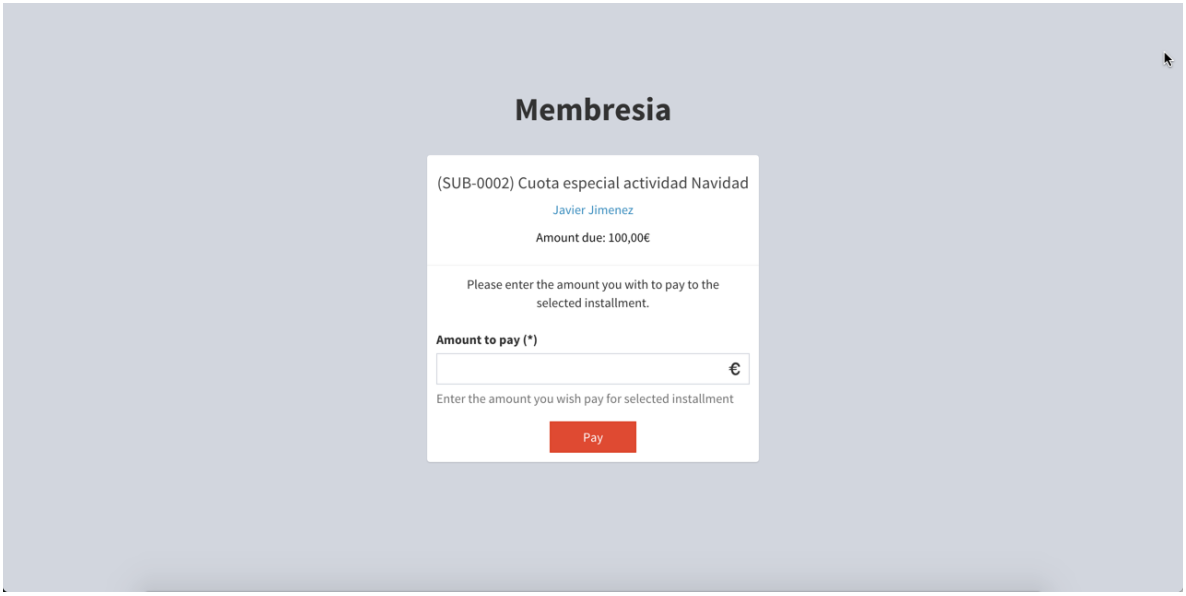
Para facilitar el proceso, este formulario está provisto con un resumen de los últimos pagos realizados por el socio.



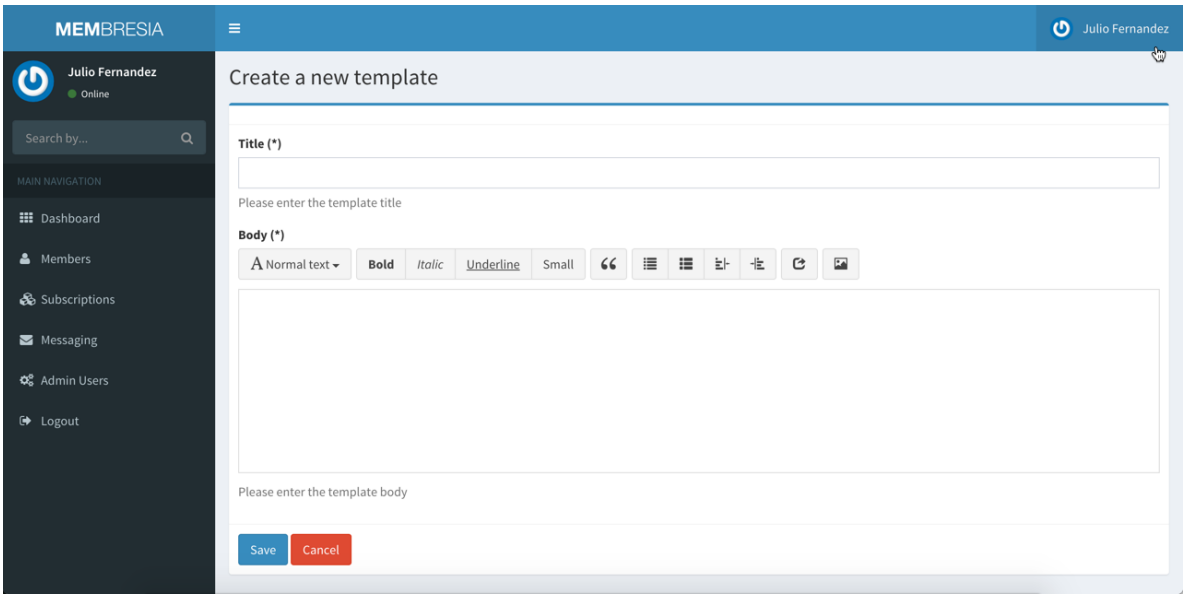
Formulario de registro de pago por correo

El formulario de pago a través de email, es idéntico al del pago manual sólo que no es posible seleccionar la subscripción y el periodo, ya que éstos viene definidos en la URL encriptada.

Al tratarse de una vista publica (no requiere autenticación explicita por parte del usuario), este formulario carece de plantilla con elementos navegables.



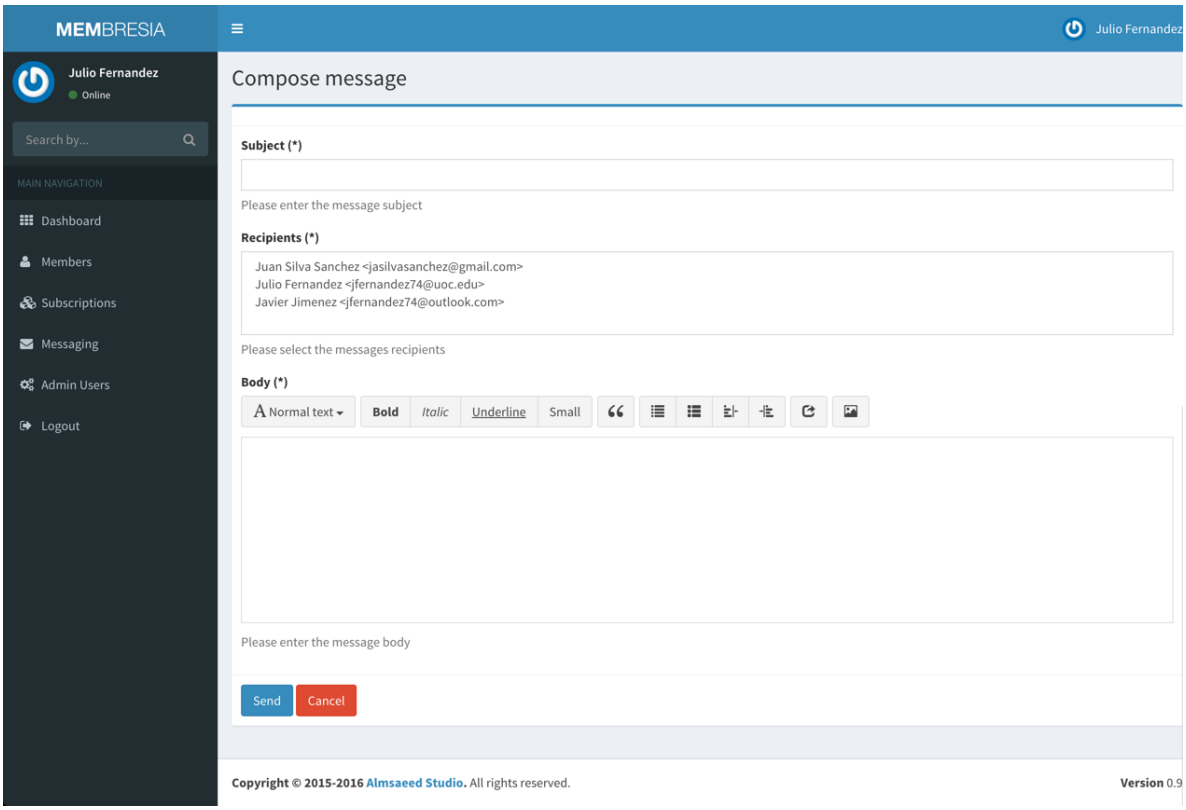
Formulario de plantilla de mensajes



Formulario de mensajes

La función del formulario de mensajes es la de proveer las herramientas para componer un mensaje de correo electrónico. Este formulario permite comenzar la composición desde un formulario en blanco o haciendo uso de una plantilla. Para utilizar una plantilla sólo es necesario seleccionarla desde el menú desplegable.

Para que el envío sea exitoso, es necesario que se seleccione uno o mas usuarios socios como destinatarios del mensaje. Esta acción puede realizarse desde el panel inferior del formulario.


MEMBRESIA
Julio Fernandez
Online
Search by...
MAIN NAVIGATION
Dashboard
Members
Subscriptions
Messaging
Admin Users
Logout
Julio Fernandez
Compose message
Subject (*)
Please enter the message subject
Recipients (*)
Juan Silva Sanchez <jasilvasanchez@gmail.com>
Julio Fernandez <jfernandez74@uoc.edu>
Javier Jimenez <jfernandez74@outlook.com>
Please select the messages recipients
Body (*)
Normal text
Bold
Italic
Underline
Small
Please enter the message body
Send
Cancel
Copyright © 2015-2016 Almsaeed Studio. All rights reserved.
Version 0.9

Implementación

La implementación del sistema se realizó haciendo uso del framework de desarrollo JavaEE **Play2**. A pesar de ser un framework muy popular, los inconvenientes no tardaron en aparecer durante las primeras fases del desarrollo. Los factores mas críticos que surgieron durante esta etapa inicial del desarrollo fueron como se indican a continuación:

1. **Sistema inestable**: Play es un framework relativamente nuevo (año 2012 proximadamente) que actualmente se encuentra en su versión 2.4. Este framework ha pasado por diversas etapa de desarrollo adaptándose a las necesidades de los usuarios y sus demandas, lo que ha causado que las distintas versiones existentes sean muy disparejas las unas de una de las otras. Esta diferencia en la operatividad del framework es tan marcada que en internet ya se habla de **Play1** y **Play2** como si de dos diferentes frameworks se tratasen. Existe tanta disparidad en las versiones, que incluso pueden percibirse diferencias marcadas entre las distintas versión de **Play2**.
2. **Falta de documentación**: Al tratarse de un sistema vivo y en continuo desarrollo, existe poca, ninguna o información contradictoria en internet referente al desarrollo en esta plataforma. La documentación oficial es escueta y poco clara, por lo que la única fuente de información fiable disponibles es aquella extraída desde foros tales como *StackOverflow* o *Google Forums*.
3. **Poco soporte IDE**: Existe muy pocos IDE con capacidad para desarrollar en **Play2**, y aquellos que ya lo hace carecen de un soporte 100% fiable.
4. **Lenguaje de programación**: Aunque Play soporta el uso de Scala y/o Java como lenguaje de programación, el uso de ambos de forma simultánea no es recomendado. A diferencia de Groovy and Grails, Play no opera bien cuando se intercalan ambos lenguajes es un mismo proyecto. El problema reside principalmente en que existen dos paquetes diferentes de librerías según se haga uso de Scala o Java, y cada una de ellos con las mismas clases pero adaptadas al lenguaje concreto. Este ambigüedad provoca problemas de interpretación en el

IDE y de compilación. Aunque se seleccionó Java puro como lenguaje de programación principal, las vistas están todas programadas en **Scala**. Esta implementación dual fue posible dado que las librerías relacionadas con las vistas no creaban conflicto alguno con aquellas de la lógica de negocio.

A pasar de los inconvenientes ya mencionados, Play resulto ser un framework muy fácil de trabajar y altamente fiable una vez controlados los factores de riesgo. Algunas de las funcionalidades que Play ofrece son:

1. **Sistema de gestión de aplicaciones**: La versión utilizada, **Play 2.4**, incluye un sistema denominado **Activator** que permite la inicialización de plantillas de proyectos, según sea lenguaje de programación seleccionado y tipo de proyecto. **Activator** también hace las veces de servidor o contenedor de aplicaciones, que permite realizar el desarrollo en un ambiente de trabajo especialmente diseñado para **Play** e independiente de la configuración de sistemas tales como Tomcat o JBoss (WildFly). **Activator** también se encarga de la compilación y mantenimiento del código.
2. **Soporte de librerías externas**: A pesar de su corta edad, Play tiene un gran soporte de librerías externas mediante Maven y/o Typesafe. Estos sistemas de gestión de librerías permiten integrar nuevas funcionalidades en la aplicación con poco esfuerzo.
3. **MVC**: **Play2** es puro MVC. Su infraestructura de tres capas basada en el patrón de diseño MVC permite un desarrollo estructurado y organizado que hace del proceso de desarrollo uno de fácil elaboración. El escalamiento y la localización de errores es sencillo y eficaz.
4. **Compilación automática en tiempo real**: Gracias a **Activator**, los cambios realizados en el código son compilados y reemplazados en el contenedor en tiempo real.

5. **Evolution**: Play hace uso de **Evolution** para sincronizar el esquema de base de datos con el modelo JPA programado.

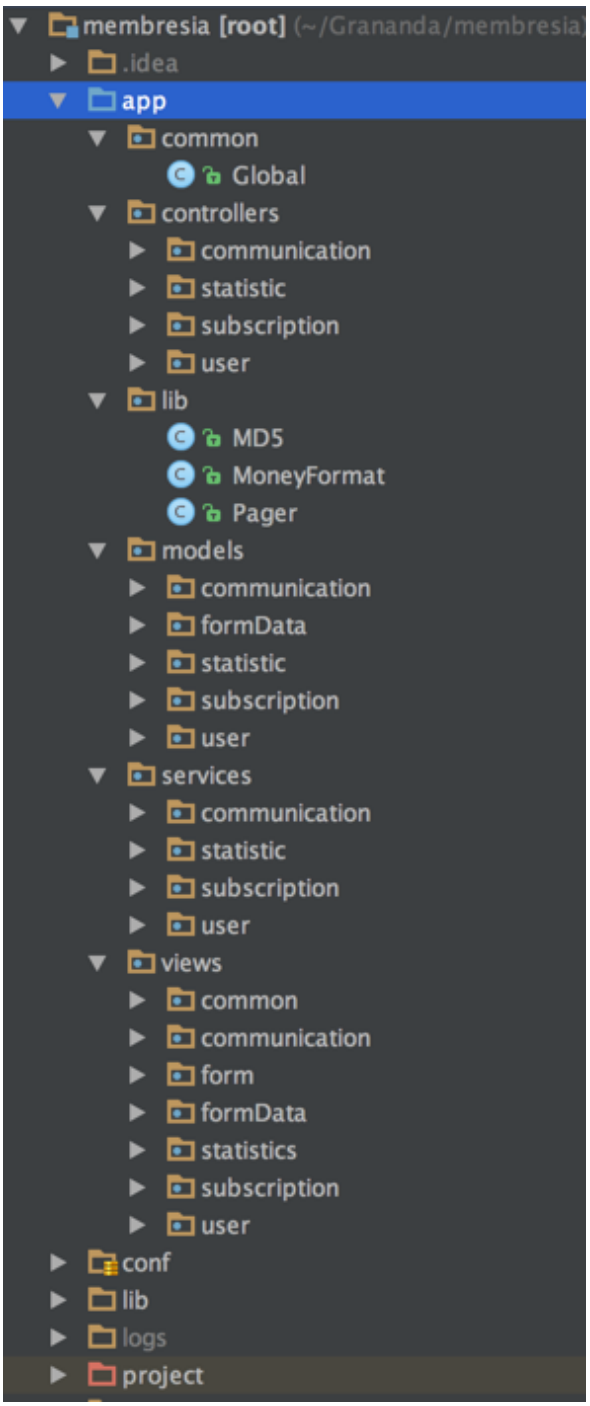
En general puede decirse que el período de implementación fue muy satisfactorio y eficiente convirtiendo a **Play2** en uno de mis framework de desarrollo JavaEE favoritos.

Software Utilizado.

Para el desarrollo del sistema se hizo uso del siguiente software de terceros:

1. **Intellij IDEA** para el desarrollo y codificación del sistema. Este ha resultado ser el único IDE con soporte global para **Play2**. A pesar de ello, al equipo de IDEA no ha podido mantener su producto a la par con el polimórfico Play, especialmente en la gestión de las vistas y la integración de Java como lenguaje principal de desarrollo.
2. **Postgresql** y **Mysql** como sistema de gestión de base de datos. Aunque inicialmente el sistema fue desarrollado para trabajar con **MySql**, el servidor de desarrollo en **Heroku**, utiliza **Postgresql**. La migración de un sistema a otro es muy sencilla gracias a que Evolution se encarga de generar las tablas según el SGBD seleccionado.
3. **Heroku** como servidor de desarrollo para aplicaciones web en la nube. Uno de los grandes inconvenientes del desarrollo en JavaEE ha sido siempre el encontrar un alojamiento escalable y fiable para JavaEE con herramientas eficaces de control de despliegue. Este era una tarea prácticamente imposible hasta hace poco. **Heroku** es precisamente la solución perfecta para aplicaciones basadas en JavaEE. Este servidor en la nube permite el despliegue de sistemas desde la consola de comando a su propio sistema de versión de gestiones o **Git-Hub** entre otros. **Heroku** provee un plan gratuito para aplicaciones en desarrollo que ha resultado muy eficaz para este proyecto. En adición, este servicio provee soporte especial para aplicaciones **Play2** que hace del proceso de despliegue uno altamente eficaz.

4. **DataGrip**: Este gestor de escritorio de base de datos de IDEA ha sido de gran utilidad para corroborar la eficacia de las entradas en la base de datos y el comportamiento de los datos como resultado de las acciones ejecutadas en el código y durante la ejecución del sistema.
5. **Docker**: Sistema de virtualización que permite correr aplicaciones como módulos añadidos sobre una maquina virtual Linux. **Docker** provee la herramienta **Kinematic**, un gestor de módulos **Docker** que facilita la instalación, puesta en marcha y ejecución de éstos. **Docker** ha sido utilizado para correr las base de datos utilizadas durante el desarrollo, tanto para **MySQL** como para **Postgresql**.
6. **AdminLTE Control Panel Template**: Aunque no es una aplicación en si, esta plantilla de diseño para gestores de contenidos ha sido de gran utilidad. AdminLTE es un sistema *responsive* y adaptable con herramientas y plug-ins en javascript, que han resultado altamente adaptables al sistema desarrollado.
7. **Git**: **Git-Hub** ha sido el repositorio y sistemas de control de versiones oficial de este proyecto. El código fuente esta disponible al público en general en https://github.com/grananda/membresia



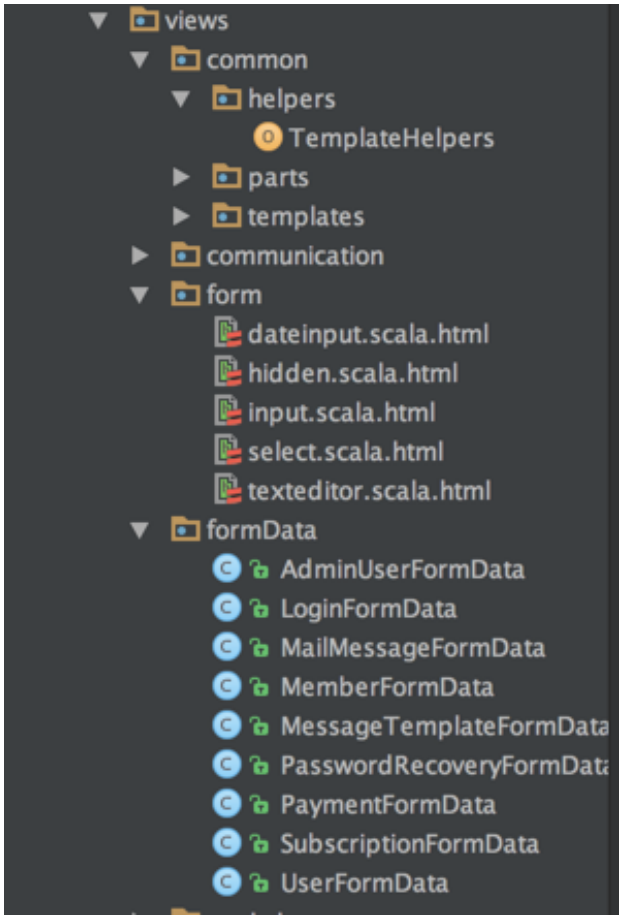
Estructura del Código

La carpeta principal de la aplicación Play es app. Es aquí donde reside todo el código. Esta carpeta esta dividida en:

1. **Common**: Carpeta donde almacenamos el archivo *bootstrap* de inicio. Aquí se ubica la clase *Global.java* donde se encuentran las funciones que necesitamos que se ejecuten antes y después de poner el sistema a correr o cuando detenemos la ejecución del mismo. Aquí también se encuentran los *cron-jobs* utilizados en el sistema, como por ejemplo la creación de periodos de pagos.
2. **Controllers**: Aquí se encuentran los controladores organizados en paquetes por componente según el diseño de componentes previamente descrito.
3. **Lib**: Archivos clases de librerías de terceros utilizadas en el código.
4. **Models**: Contiene las clases modelos para la definición **JPA** en **Ebean**.
5. **Services**: Esta carpeta contiene la capa de negocios.
6. **Views**: Aquí se encuentran las vistas en html **Scala**.
7. **Conf**: Contiene los archivos de configuración global del sistema, parámetros de conexión de bases de datos y traducciones de los textos. Esta carpeta también

almacena lo archivos regionales que traducen la web en catalán, castellano o ingles.

8. **Project**: Entre otras cosas, es aquí donde se almacena el archivo configuración para las dependencias o librerías de Play.



Capa de Presentación

La capa de presentación ha sido montada sobre **AdminLTE Control Panel Template** utilizando plantillas en **Scala** y **Twirl** (framework de **Play2** para ejecución de plantillas HMTL).

Las vistas están organizadas en paquetes según el componente al que pertenecen. En adición a estos paquetes, la carpeta de las vistas contiene dos carpetas especiales. En **common** tenemos archivos de ámbito común, como por ejemplo la plantilla principal y la de correos (templates); el archivo de navegación; la paginación de listados y el perfil de usuario (parts). Estos son todos archivos comunes que se encuentran en todas las paginas.

Pero de todos los archivos de las vistas, quizás los más relevantes son los contenidos en **formData**; esta carpeta contiene clases que permiten la validación de los modelos desde el servidor. Estas clases también se encargan de transferir los objetos desde el modelo a la vista y de recolectar los datos que provienen de éstas a través de POST o GET.

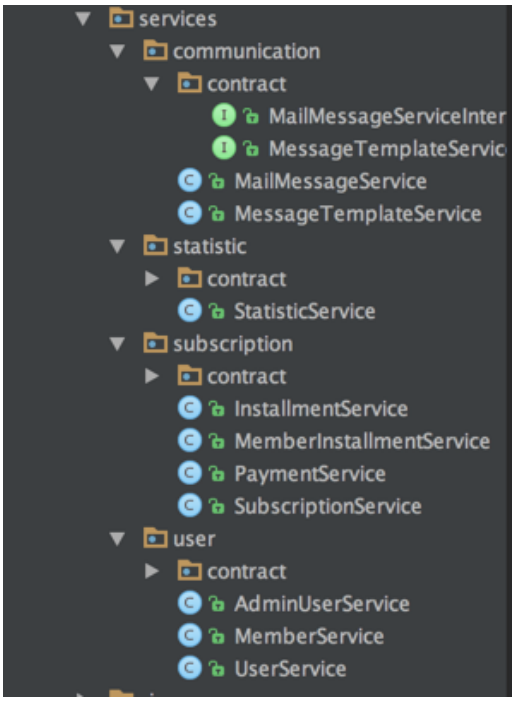
En adición, las carpeta **form** contiene fragmentos **Scala** que generan componentes de formularios HTML en formato **Twitter Bootstrap.**

Los controladores también forman parte de la capa de presentación y se ubican en **controllers**, donde también se organizan por componente. Un dato a resaltar sobre los controladores es que hacen uso anotaciones para crear *inyecciones de dependencias* para el uso de servicios de la capa de negocios y la seguridad.

```
/**
 * Controller class for AdminUser entity
 */
@With(SecuredAction.class)
public class AdminUserController extends Controller {

    @Inject
    private MailerClient mailer;

    @Inject
    private AdminUserService adminUserService;
```

La seguridad esta delegada en una clase o acción especial dentro del controlador *SecuredAction* que se ejecuta cada vez que una acción la solicita y que comprueba la existencia de una variable de sesión concreta.

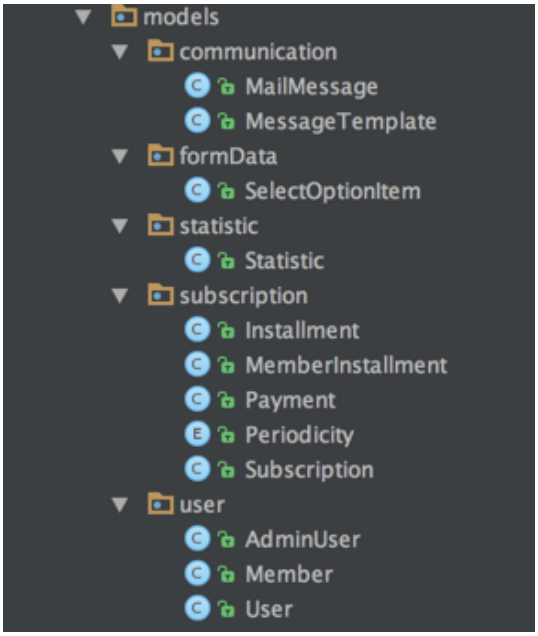
```
package controllers.user;

import ...

/**
 * Class for security check on action access
 */
public class SecuredAction extends Action.Simple {

    /**
     * Checks for auth session and redirects user if not authenticated.
     *
     * @param ctx Http context
     * @return F.Promise<Result>
     * @throws Throwable
     */
    public F.Promise<Result> call(Http.Context ctx) throws Throwable {
        String token = session("X-AUTH-TOKEN");
        if (token != null) {

            return delegate.call(ctx);
        }
        Result unauthorized = Results.redirect(controllers.user.routes.UserController.login());

        return F.Promise.pure(unauthorized);
    }
}
```



Capa de Negocios

La capa de negocios es algo inexistente en **Play**. Aunque **Play** sigue el patrón MVC, éste hace uso de los archivos para los modelos como medio para implementar la capa de negocio.

Con la finalidad de crear una abstracción y separación de responsabilidades mas adecuada a las necesidades del sistema, se optó por crear una capa de negocios aislada a modo de servicios. Esta capa contiene los archivos que se encargan de procesar los datos provenientes del controlador y prepararlos, si cabe, para ser transferidos al modelo. Es aquí donde encontramos los interfaces y donde se implantaría el acceso **RESTful** como parte de un servicio web.



La Capa de Integración

Esta capa esta representada por los modelos. Los archivos clases contenidos en esta capa son los encargados de modelar las entidades que componen el diagrama UML. Estos modelos hace uso de **Ebean** como sistema de persistencia de datos.

En adición, estas clases contiene toda la lógica que interacciona entre el objeto y **Ebean** para garantizar la persistencia con la base de datos. Es en esta capa también desde donde extraemos los datos necesarios desde nuestra base de datos para mostrarlos en las vistas.

Pero no todos los modelos en esta capa son persistentes. Algunos de ellos son virtuales, es decir, que carecen de persistencia en la base de datos. Uno de los modelos virtuales mas

destacados es el de estadísticas. Este modelo no tiene persistencia de datos ya que su función es la de recolectar la información necesaria para construir los informes y gráficos del sistema.

```
public class Statistic {

    /**
     * Returns a list of total money collected per month period and subscription
     *
     * @param token Unique subscription token
     * @return List
     */
    public List<SqlRow> getTotalPaymentsBySubscriptionAndPeriod(String token) {
        String sqlMySql =
                "SELECT" +
                        " SUM(payment.amount) AS amount," +
                        " DATE_FORMAT(payment.created_at, '%m/%Y') AS period" +
                        " FROM payment" +
                        " LEFT JOIN installment ON installment_id = installment.id" +
                        " LEFT JOIN subscription ON installment.subscription_id = subscription.id" +
                        " WHERE subscription.token = :token" +
                        " GROUP BY period" +
                        " ORDER BY period" +
                        " LIMIT :limit";

        String sqlPostgreSQL =
                "SELECT" +
                        " SUM(payment.amount) AS amount," +
                        " to_char(payment.created_at, 'MM/YYYY') AS period" +
                        " FROM payment" +
                        " LEFT JOIN installment ON installment_id = installment.id" +
                        " LEFT JOIN subscription ON installment.subscription_id = subscription.id" +
                        " WHERE subscription.token = :token" +
                        " GROUP BY period" +
                        " ORDER BY period" +
                        " LIMIT :limit";

        return Ebean.createSqlQuery(sqlPostgreSQL)
                .setParameter("token", token)
                .setParameter("limit", 12)
                .findList();
    }
```

Es importante comentar que este es el único modelo que no utiliza **JPA** y hace uso de **SQL** puro para su función de extracción de datos. Al evadir el sistema de persistencia de datos, el **SQL** es dependiente del **SGBD** y por ello se han dejado en el código los **SQL** correspondiente a **Postgresql** y **MySQL**, aunque el actualmente utilizado es el que corresponde a **Postgresql**. Si se hacer uso de un **SGDB** diferente a los mencionados, seria necesario escribir el SQL correspondiente a este nuevo sistema.

Trabajo Futuro

A medida que avanzaba el desarrollo del proyecto nuevas ideas sobre como mejorar el producto fueron surgiendo, así como otras pocas que se fueron quedando atrás por ser consideradas poco útiles o demasiado laboriosas como para cumplimentar con la fecha de entrega acordada en el calendario de trabajo. Algunas de estas funcionalidades en vistas de ser producidas en un futuro son:

- **Listado de pagos pendientes**: Aunque originalmente prevista, esta funcionalidad no pudo ser incorporada en el sistema. Sin embargo se considera que es de suma importancia para la gestión de los cobros de las cuotas y será implementada en un futuro.
- **Servicios Web:** La implementación de servicios web remotos para funciones básicas de sistema, como por ejemplo ejecutar un pago o crear una instancia, son funcionalidades que están prevista a ser incorporadas como servicios web en una futura actualización del sistema.
- **Ajax:** La experiencia del usuario es importante y por eso una de las mejoras a implementar en un futuro son servicios AJAX que permitan agilizar la forma en que el usuario visualiza la información. Esta implementación esta prevista inicialmente en la vista de detalle de las subscripciones, desde donde el usuario gestor podrá ver los pagos realizados por periodos y miembro según desee en vez de un listado global de todos los pagos pendientes que puede resultar poco viable una vez se haya recolectado una gran cantidad de pagos.
- **Portal Miembros:** Una de las futuras implementaciones mas ambiciosas es un portal publico para miembros. Desde este portal un miembro podrá darse de alta e inscribirse a la subscripciones que desee.
- **Sistema Global:** Por ultimo, y como complemento al punto anterior, se espera que *Membresia* pueda convertir en un portal de membresías global en el que cualquier asociación pueda darse de alta y gestionar sus pagos y socios desde un mismo

lugar en común con diversos métodos de pagos y servicios adicionales como la gestión de actividades.

Conclusiones

Debo decir que al principio tenia mis dudas sobre el uso de **Play** en el desarrollo de este proyecto, pero una vez en marcha pude descubrir un framework eficaz capaz de hacer el trabajo fácil, eficiente y práctico. Sencillamente, se puede decir que Play funciona y no dudo que en par de años se convierta en el framework de desarrollo por excelencia para desarrollos en JavaEE.

Como antiguo programador de sistema basados en **Groovy and Grails**, puedo decir que **Play** a mejorado el desarrollo en JavaEE de una forma notable. Su robustez, simplicidad y alto rendimiento superan con creces a **Grails** como framework de desarrollo para JavaEE. **Play** es un buen ejemplo de cómo una tecnología tan robusta como JavaEE puede estar al alcance de todos.

Ha sido un trabajo duro. Más de tres meses de trabajo continuo y de aprendizaje constante de nuevas tecnologías. Por otro lado ha sido la culminación de un proyecto de vida de años de carrera, un viaje de crecimiento intelectual y personal. Mientras trabajaba en el proyecto he podido recordar mis primeros años de carrera cuando apenas podía escribir dos líneas en Java sin buscar referencias en la web; nueve años mas tarde puedo decir que el desarrollo ha sido fluido y metódico, señal inequívoca de mi dominio sobre el lenguaje escogido para el desarrollo de la practica.

Sobre mis intereses futuros profesionales he podido meditar y definir hacia donde quiero (y no quiero) ir, que destrezas quiero desarrollar y cuales delegar. Este proyecto me ha servido de vehículo para definir mis metas a medio y largo plazo dentro del ámbito profesional, y esto se resume en una dedicación completa al desarrollo de sistema puramente enfocados al *back-end*. Programar es lo que sé hacer y lo que más disfruto dentro de todas las facetas que la ingeniería en computadores ofrece.

Fuentes

Recursos

Java Oracle
http://www.oracle.com/

Ebean
https://ebean-orm.github.io/

Play Framework
https://www.playframework.com/

MySQL
https://www.mysql.com/

Postgresql
http://www.postgresql.org.es/

Docker
https://www.docker.com/

Vitual Box
https://www.virtualbox.org/

Git-Hub
https://github.com/

Axure
http://www.axure.com/

Fortawesome
https://fortawesome.github.io

Play mailer
https://github.com/playframework/play-mailer

Artículos

Ebean ORM
https://ebean-orm.github.io/docs/

Java Servlets
http://users.dcc.uchile.cl/~jbarrios/servlets/general.html

Scala tour
http://docs.scala-lang.org/tutorials/tour/tour-of-scala.html

Play, the main concepts
https://www.playframework.com/documentation/1.0/main#lifecycle

Day 30: Play Framework–A Java Developer Dream Framework
https://blog.openshift.com/day-30-play-framework-a-java-developer-dream-framework/

What are the pros and cons of the Play Framework 2, for a Java developer?
https://www.quora.com/What-are-the-pros-and-cons-of-the-Play-Framework-2-for-a-Java-developer

How do I generate a random alpha-numeric string?
http://kodejava.org/how-do-i-generate-a-random-alpha-numeric-string/

Responsive HTML email template
https://github.com/leemunroe/responsive-html-email-template

https://gist.github.com/monteiro/4353448

Authentication in Play Framework using Java
http://alexgaribay.com/2014/06/16/authentication-in-play-framework-using-java/

Play Authetication
https://github.com/alexgaribay/play-authentication

Adding authentication
https://www.playframework.com/documentation/2.2.x/JavaGuide4

SOLVING @INJECT NULL POINTER EXCEPTION
http://www.javablog.be/inject-nullpointerexception/

Java EE CDI Dependency Injection (@Inject) tutorial
http://www.javacodegeeks.com/2013/05/java-ee-cdi-dependency-injection-inject-tutorial.html

Scheduling Jobs In Play 2
http://brainstep.blogspot.nl/2013/10/scheduling-jobs-in-play-2.html