APP financiera para la gestión de las hojas de gasto en *ServiceNow*

Daniel de Vicente Garrote

Septiembre 2020

Resumen

La idea de este proyecto es crear una aplicación basada en *ServiceNow* que permita a sus usuarios poder gestionar gastos en su empresa de forma fácil y sencilla.
Las características principales de esta son la inclusión de gastos a mano, la importación de gastos a partir de hojas de cálculo, la búsqueda de facturas por periodo y usuario, y su manejo mediante una aplicación de *Smartphone* para ver y crear gastos a través de esta. Una característica importante es que el usuario sea capaz de poder adjuntar imágenes a los gastos para poder añadir copias de las facturas en estas, e incluso poder tomar una fotografía a través de su teléfono para añadirla directamente a la aplicación. Adicionalmente, los usuarios deben diferenciarse entre usuarios normales y usuarios mánager. Estos últimos pueden aprobar o rechazar los gastos de otros usuarios, exportar sus datos a hojas Excel y crear o importar gastos para otros usuarios.
Por otro lado, debe poder crearse clientes con sus proyectos, los cuales llevarán asociados un código de proyecto, y dichos proyectos deben asociarse a los gastos que los usuarios creen. Finalmente, ambos usuarios deberán tener acceso a reportes en tiempo real sobre la información de la aplicación, que variará dependiendo del tipo de usuario. Dicha aplicación tendrá que tener un énfasis en la facilidad de uso, por lo que no solo tendrá que cumplir con los objetivos propuestos, sino que el usuario debe poder familiarizarse rápidamente con su funcionamiento y que este le resulte cómodo.
La utilidad de esta aplicación reside en la capacidad de la empresa (en este caso *SilverStorm*) de poder administrar los gastos de los empleados durante viajes y reuniones, permitiendo a los usuarios añadirlas en la aplicación y a los empleados de contabilidad (los mánagers) poder determinar que gastos cubrirá la empresa o cuales no. Por otro lado la aplicación mostrará la información en distintos formatos, ya sea por tablas o por gráficos en tiempo real, permitiendo a los mánagers visualizar fácilmente donde se están cubriendo los gastos de los empleados de la empresa (los usuarios normales), así como los suyos propios.

Tabla de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La idea de este Trabajo de Fin de Grado viene a partir de varias ideas que se me ocurrieron a mi antes de comenzar este proceso. Por un lado estuve trabajando en *SilverStorm* durante un año aproximadamente, y se me ocurrió que podría buscar un TFG que utilizara la tecnología actual de *ServiceNow*, ya que es la que utilizo en mi empresa actualmente. Por otro lado la oportunidad de realizar el trabajo ofrecido por la empresa me enseñará los procedimientos de la empresa en cuestión de obtención de requisitos, creación de procesos y listado de elementos técnicos necesarios que se aplican en mi empresa actualmente.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado consisten en crear una aplicación basada en *ServiceNow* que permita a sus usuarios registrar gastos y comprobar si son admisibles como gastos de empresa. Para ello habrá que seguir una serie de pasos, acompañados de una metodología híbrida, utilizada en la empresa *SilverStorm*. Adicionalmente será necesario poder registrar las tarjetas de crédito que se asignen a los usuarios, ver los gastos que realicen con ellas, y generar registros de pagos para cubrir los gastos de la empresa.

1.3. Contexto

SilverStorm es una compañía española especializada en creación de productos y servicios software que mediante metodologías ágiles permite a los diferentes equipos de trabajo establecer que objetivos debe alcanzar sus productos, así como que elementos son necesarios para poder alcanzar los objetivos de dicha aplicación.
Dentro de la empresa se ofrece a los estudiantes la posibilidad de realizar las prácticas de empresa necesarias en la carrera de Ingeniería Informática de Valladolid, incluyendo una formación sobre las tecnologías empleadas. Adicionalmente se puede ofrecer a los estudiantes la opción de realizar Trabajos de Fin de Grado, basándose en aplicaciones desarrolladas sobre *ServiceNow*, cuyos requisitos son

ofrecidos por empleados de la propia compañía, de forma que las aplicaciones desarrolladas puedan ser usadas en un futuro.
ServiceNow, es una compañía americana fundada en 2004, que basa su trabajo en diferentes tecnologías como *ITSM*. Sin embargo con el paso del tiempo sus campos de trabajo fueron ampliándose a otros sectores como Recursos Humanos, Seguridad, Delivery y Atención a Cliente. Dicha compañía ofrece su propio servicio basado en nube, el cual ofrece a sus clientes servidores con diferentes aplicaciones que cubran los anteriores campos.
El software de *ServiceNow* se basa en el modelo *application Platform-as-a-Service(aPaaS)* [18], es decir, que es un servicio en la nube que ofrece herramientas de desarrollo y despliegue de aplicaciones. Su infraestructura es la de *multi-instance single-tenant*, la cual permite crear diferentes instancias para diferentes cliente, y permite aislar los datos de una instancia del resto de instancias. Dicha infraestructura ofrece la ventaja del aislamiento de datos, así como poder realizar upgrades a instancias sin afectar a otros clientes.
Por otro lado el desarrollo de aplicaciones es bastante intuitivo. por un lado es posible crear lógica de negocio con solo un poco de conocimiento de JavaScript, que es el lenguaje de programación utilizado en la plataforma para definir la lógica. Adicionalmente otros elementos son completamente creados a partir de interfaces y no requieren conocimiento en *scripting* para utilizarlos, como por ejemplo las *UI Policies*, que son elementos de lógica que permiten definir la consistencia de los campos cuando se cumplan ciertas condiciones definidas por el usuario.
Otra característica relevante de *ServiceNow* es que la creación de vistas para *SmartPhones* permite usar la aplicación móvil para acceder a los datos con dicho *SmartPhone*, lo que facilita el acceso a datos fuera de estaciones de trabajo.
Otras utilidades de *ServiceNow* incluyen: Manejo de roles de usuario, manejo de grupos de usuarios, flujos de trabajo, tests automatizados, gestión de seguridad mediante ACLs, creación de paginas personalizadas con *Angular* (Service Portal), y la posibilidad de incluir *plugins* para añadir mas funcionalidad a las instancias.
El desarrollo de esta aplicación va destinada a los empleados de la empresa, los cuales pueden registrar gastos asociados con su desempeño laboral, particularmente los relacionados con viajes de negocio. Entre los gastos realizados en los viajes de trabajo destacan principalmente los gastos de transporte (tren, coche, avión...), los gastos de comidas de empresa, los gastos de alojamiento y otro tipo de gastos que puedan surgir (accidentes, compra de material *in situ*...). Adicionalmente los empleados de la sección de contabilidad contarán con mas funcionalidad para determinar que gastos son admisibles como gastos de empresa y cuales no, así como validar la lista de gastos de un usuario asociados a un periodo de tiempo determinado (en nuestro caso de un mes, aunque con opción a cambiar). También permite a los empleados de contabilidad (mánagers de gastos de empresa) crear gastos a nombre de otros empleados si es necesario, así como añadir usuarios y crear o modificas listas de gasto para todos los usuarios.

1.4. Procedimiento

Como se ha descrito anteriormente, será necesario reunirse con el cliente, obtener los objetivos que debe cumplir la aplicación, listar los requisitos, definir los pasos del proceso de gastos, así como otros procesos propios de la aplicación, listar los elementos técnicos necesarios y realizar la implementación.

1.5. Tecnologías utilizadas

Las tecnologías usadas para este Trabajo de Fin de Grado son las siguientes:

- ServiceNow: Software donde se implementará la aplicación. Aunque nosotros crearemos elementos desde cero también trae otros elementos de caja, como control de usuarios y permisos de acceso.
- Microsoft Office: Para la creación de los documentos de la aplicación.
- draw.io: Para crear la línea de tiempo.
- Astah UML: Para crear los diagramas de proceso, y el diagrama de roles.

1.6. Estructura del documento

La memoria de este trabajo esta dividida en diferentes capítulos, junto a la bibliografía y una anexo de imágenes de la aplicación.
El primer capítulo está dedicado a la metodología de trabajo, que es la empleada por la empresa, y es una combinación de dos metodologías: Una ágil, y otra mas tradicional. EL segundo capítulo desarrolla los elementos de proceso, descritos en la plantilla de procesos. El tercer capítulo enumera los elementos técnicos que deben crearse en *ServiceNow*, así como las historias de usuario correspondientes a los elementos mencionados anteriormente.
El cuatro capítulo explica de forma concisa como se ha implementado los elementos obtenidos de las historias de usuario, junto a los pasos seguidos para crearlos. El último capítulo describe una serie de datos que permiten reproducir los procesos detallados anteriormente, junto a ejemplos correctos y erróneos de la ejecución de dichos procesos.

Capítulo 2

Metodología de trabajo

2.1. Metodología Agile/WaterFall

La metodología empleada en el trabajo es la misma que se emplea en la empresa *SilverStorm*, que consiste en una combinación entre las metodologías *Agile* y *WaterFall*. En nuestro caso tenemos dos documentos de plantilla que nos permitirán conocer tanto los elementos a nivel de proceso (ciclo básico de una lista de gastos, roles de los usuarios, responsabilidades, estados de una entidad), como que elementos a desarrollar a nivel técnico (tablas, formularios, scripts...). A partir de esos documentos podemos definir que historias de usuario tenemos, así como que elementos crear para llevarlas a cabo.

2.1.1. Metodología *Agile*

La metodología *Agile* es una metodología[1] basada en el desarrollo cíclico de una serie de fases con el objetivo de poder ir incrementando la funcionalidad del software desarrollado, gracias al constante flujo de información recibido por parte del cliente, que ayudan a ir cambiando el software a medida que se va desarrollando. Dicha característica la diferencia de la metodología de casos de uso, donde la funcionalidad especificada por el cliente se realiza de forma previa y no permite modificaciones posteriores. En *Agile* cada ciclo posee una duración fija de unas semanas, tras la cual se realiza una reunión para determinar que objetivos se han cumplido, y cuales quedan por cumplir.

2.1.2. Metodología *WaterFall*

La metodología *Waterfall*[2], a diferencia de *Agile*, es una metodología que sigue una serie de pasos en cascada, es decir, uno detrás de otro, y no realiza iteraciones. Es una metodología mas tradicional que no itera sobre sí misma, sino que sigue una serie de fases, las cuales se describen así:

- Requisitos: Se obtienen los requisitos del cliente, para poder ir definiendo las siguientes fases.
- Diseño: Se especifica los pasos que debe seguir el producto software, y los elementos que lo componen.
- Implementación: Se realiza la implementación de los elementos obtenidos.

- Verificación: El cliente verifica que todo está bien.

- Mantenimiento: El cliente usa de forma regular el software, con el objetivo de encontrar fallos y que el equipo de desarrollo los arregle.

2.1.3. Combinación *Agile-WaterFall*

A pesar de las diferentes características de cada una de las metodologías descritas, se ha optado por una mezcla de ambas, ya que es la habitualmente utilizada por *SilverStorm*, y además permite el uso de las características positivas de ambas. En el caso de *SilverStorm*, los procedimientos de obtención de requisitos y diseño de software siguen el procedimiento en cascada o *WaterFall*, sin embargo, la fase de implementación se realiza mediante procedimientos *Agile*. Dicho procedimiento consisten en implementar lo que se haya determinado en la fase anterior, mostrarle al cliente las implementaciones completas, y en base a su *Feedback*, desarrollar nuevos elementos, los cuales se volverán a implementar. Tras terminar el ciclo de implementaciones se procede al despliegue y mantenimiento de forma secuencial.

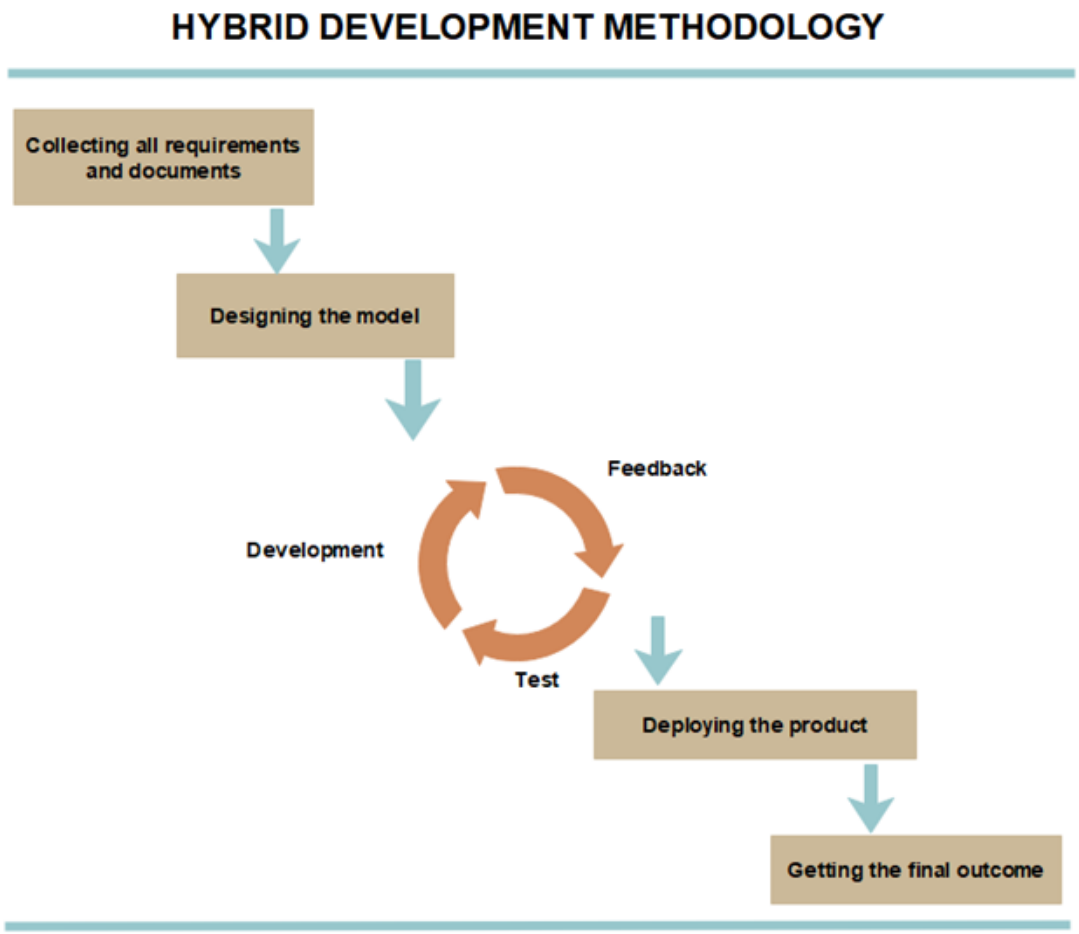


Figura 2.1: Procedimiento híbrido *Agile-Waterfall*, con iteraciones en Implementación

2.2. Obtención de requisitos

Lo primero de todo es conocer que objetivos debe cumplir nuestra aplicación. Para ello se ha concertado una primera reunión con el cliente, donde se listará que cosas debe poder hacer la aplicación, y que cosas debe evitar, como por ejemplo que un usuario sin rol de mánager pueda visualizar gastos de otros usuarios.

2.3. Desarrollo de los requisitos del cliente

Una vez hecha la primera reunión, se ha establecido una lista de requisitos funcionales que debe seguir la aplicación, junto a otros requisitos de carácter no funcional, relacionados con la seguridad, especificación de características de ciertos elementos, características de ciertos elementos como archivos y formatos de datos.

1. La aplicación debe poder generar aprobaciones de los usuarios, para decidir si son gastos válidos para la empresa o no.

2. La aplicación debe poder aceptar hojas en Excel con los gastos, y generar las líneas correspondientes a dicha hoja de cálculo.

3. La aplicación debe permitir a los mánagers de los usuarios poder decidir sobre las aprobaciones de los gastos, de forma que si se aprueban se gestiona su pago según la forma de este, y si se rechazan se quitan de la aplicación (marcar como rechazado).

4. La aplicación deberá realizar el registro de pago a los usuarios según los tipo de pago realizados. Dichos tipos solo pueden ser en efectivo o en tarjeta de crédito.

5. La aplicación albergará a los clientes a los que realiza pagos, junto a los proyectos a los que van dirigidos. El objetivo de esto es poder como se gestionan los gastos a cada proyecto.

6. La aplicación permitirá a los usuarios añadir los gastos a través de una hoja de cálculo, en vez de crear los gastos a mano en la aplicación.

7. La aplicación contará con una versión móvil para la creación de gastos. También debe permitir añadir imágenes a los gastos para por ejemplo mostrar las facturas generadas.

8. La aplicación permitirá a los mánagers exportar los datos de sus usuarios a hojas de cálculo.

9. La aplicación mostrará a los usuarios y mánagers reportes gráficos sobre los gastos de la aplicación, con la funcionalidad de poder configurar que tipo de datos ver en el reporte.

10. La aplicación permitirá enviar a los usuarios las listas de gasto para que un mánager los valide y permita que se creen los gastos correspondientes.

11. La aplicación aceptará como hojas Excel para cargar datos en formato .xlsx.

12. La aplicación será una aplicación web, alojada en una instancia de *ServiceNow* y ofrecida por la empresa *SilverStorm*.

13. La aplicación será accesible a través de todos los navegadores convencionales, como *Firefox*, *Chrome* u *Opera*, entre otros.

14. La aplicación permitirá el acceso a la aplicación móvil mediante la aplicación *Now Mobile*. Dicha aplicación será accesible a los sistemas operativos *Android* y *Apple*.

Una vez hecho este recopilatorio, se procede a buscar las posibles entidades que pueden albergar nuestra aplicación, los diferentes diagramas de actividad que describan los procesos necesarios, y los elementos que deben añadirse para que sea posible cumplirlos.

2.4. Desarrollo de procesos y elementos técnicos

Durante esta fase, se elaborarán los procesos que nuestra aplicación debe cumplir, así como los elementos que lo llevarán a cabo. Para ello contamos con dos plantillas que la empresa usa habitualmente para definir dichos elementos. Una es una plantilla para procesos, donde se definen que roles de usuario hay, los procesos que sigue la aplicación mediante diagramas y estados, responsabilidades de roles (RACI), y requisitos funcionales de la aplicación.
Por otro lado tenemos la plantilla de elementos técnicos, donde definimos elementos de la aplicación como tablas, formularios y otros elementos de interfaz. También se definen otros tipos de elementos como reportes, notificaciones, páginas de *Service Portal*...

2.5. Desarrollo de historias de usuario

Una vez conocidos los elementos desarrollados en las plantillas, podemos definir nuestras historias de usuario iniciales, que incluirán entre otras cosas: Definición de tablas, creación de permisos, elementos de interfaz y otros desarrollos especiales.

2.6. Implementación: Creación de elementos, Feedback y actualización

Una vez conocidos los elementos a implementar, es cuestión de realizar dicha implementación. Durante esta fase se preguntarán dudas al cliente para ver si las implementaciones son correctas y satisfacen sus requisitos. Si existieran objeciones se propondrían nuevos cambios, o el cliente decidiría que nuevos elementos deben añadirse o cambiarse. Una vez decidido esto se desarrollan los cambios propuestos durante la reunión.

2.7. Despliegue y mantenimiento

En este último paso sería desplegar la aplicación junto a otros elementos imprescindibles en la instancia objetivo, y dejar que el cliente la vaya usando, con el objetivo de ver si los procesos funcionan correctamente, y de encontrar defectos que se hayan pasado por alto. En este último caso simplemente sería desarrollar una solución y añadirla en la instancia con la aplicación funcional.
Aunque la aplicación está completada, se ha decidido no publicarla en producción aún, debido a si la empresa decide querer usarla o no, pero estará disponible en la instancia de desarrollo por si se considera necesaria. En todo caso se creará un clon de la instancia, purgado de información personal, que será usado en la defensa del trabajo.

2.8. Línea temporal del trabajo

Durante el desarrollo de este Trabajo de Fin de Grado, se han ido realizando diferentes tareas durante determinados días. El siguiente diagrama muestra un orden temporal de los eventos mas importantes durante el desarrollo de la aplicación, así como de todo el trabajo en conjunto. En cada fase destacan las siguientes acciones:

- Reunión de objetivos y listado de requisitos: Tras haber hecho la primera reunión con el cliente, se han anotado los objetivos que debe cumplir la aplicación. A continuación se han listado los requisitos funcionales y no funcionales que corresponden a al aplicación.

- Comienzo de la documentación de procesos: Durante los días siguientes se ha ido rellenando la plantilla de documentación de procesos, empezando por roles de usuario y primeros elementos del proceso de aprobaciones de líneas de gasto. Aquí se tuvo en cuenta inicialmente la posibilidad de usar *WorkFlows*, pero se descartó por ser innecesario.

- Comienzo de la documentación técnica: Una vez teniendo definidas las partes importantes del proceso, se ha comenzado a escribir los elementos técnicos en su correspondiente plantilla. Al principio solo estaban los diagramas del documento de procesos y los roles de usuario, pero conforme avanzaban los días se han ido añadiendo otros elementos como tablas, acciones de usuario...

- Versión 2.0 de la documentación: En esta etapa el cliente especifico que las lineas de gasto deben pasar por dos etapas de aprobación: Una previa que se debía de pasar de forma individual a cada gasto, y otra relacionada con todos los gastos del usuario en el periodo actual. Aquí es donde hubo grandes cambios en el proceso y los elementos técnicos, y por ello se tuvo que añadir una nueva versión completa de los documentos.

- Primeras implementaciones de la instancia: Tras haber definido lo suficiente la documentación técnica, se ha procedido a añadir los elementos mas básicos como tablas y formularios. Con el paso del tiempo y los retoques de la documentación, se ha ido añadiendo mas elementos, y cambiando otros.

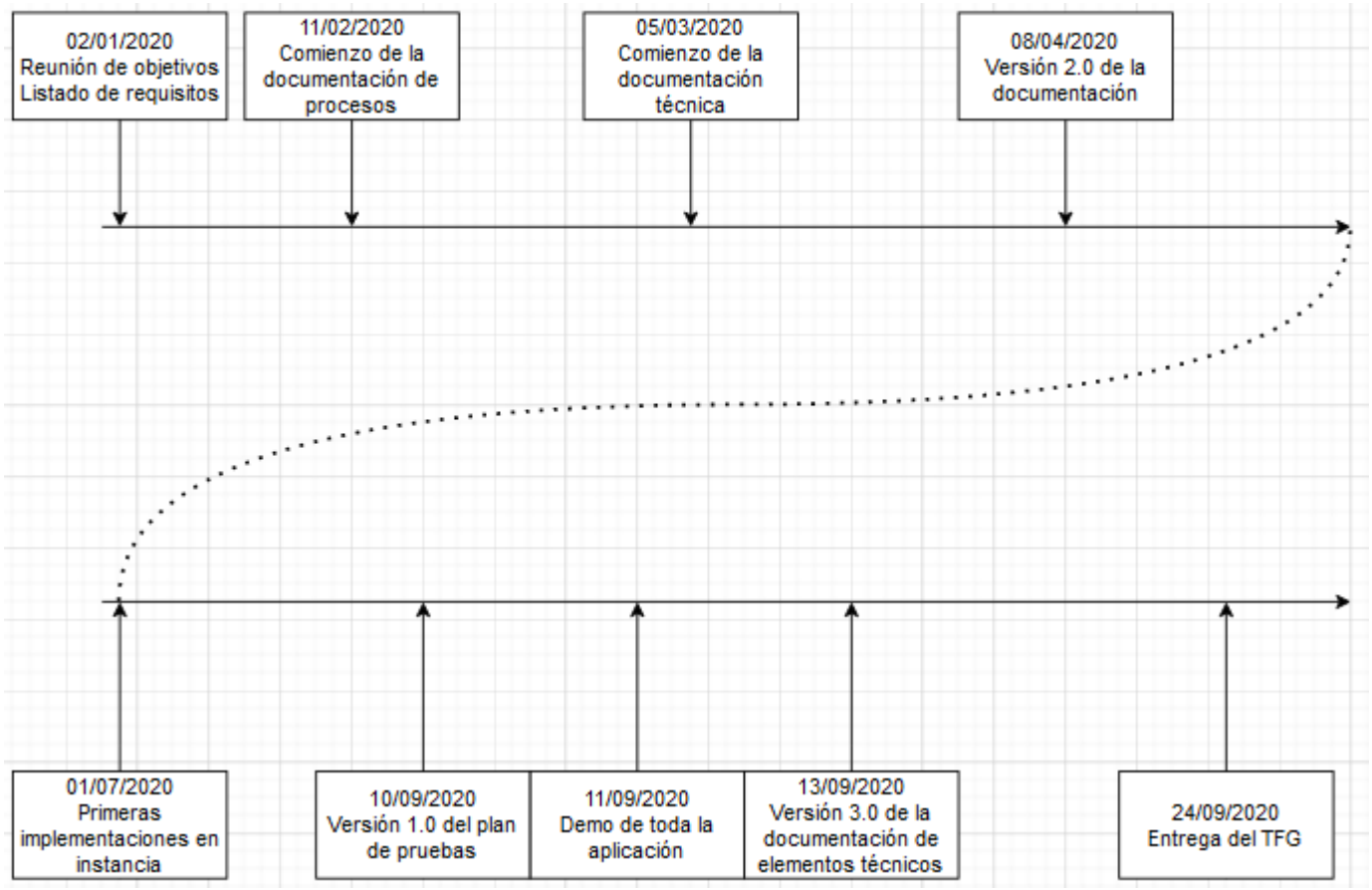


Figura 2.2: Diagrama con los eventos importantes del TFG

- Versión 1.0 del plan de pruebas: Se hizo un pequeño documento con pruebas que debía pasar la aplicación. Los siguientes días fueron dedicados a buscar los defectos que salieran y luego sería corregirlos.
- Demo de toda la aplicación: Se hizo una demostración al cliente, el cual pidió pequeños cambios, como la inclusión de un campo de coste total en los pagos, y la posibilidad de filtrar los reportes por un rango de fechas.
- Versión 3.0 de la documentación de elementos técnicos: Esta versión incluye los cambios del cliente durante la demo. Los siguientes días están dedicados al testeo y la corrección de defectos.
- Entrega del TFG: una vez arreglados los defectos, se revisa la memoria, y se entrega todo.

Capítulo 3

Elaboración de los elementos del proceso

3.1. Roles de usuario

Uno de los aspectos mas fundamentales de la aplicación es la existencia de diferentes tipos de usuario, cada unos con unas opciones disponibles dentro de la aplicación. En nuestro caso solo nos interesa distinguir dos tipos de usuarios nuevos:

- Empleado(bill_user): Sería un usuario sin privilegios especiales, que se dedica principalmente a registrar los gastos dedicados a la empresa por diferentes motivos como viajes o comidas entre otros. También puede realizar importaciones y exportaciones, pero solo puede ver y exportar sus propios gastos, así como el uso de la aplicación móvil para las actividades anteriormente descritas.

- Mánager(bill_manager): Sería un supervisor de empresa que controlara los gastos de esta (finanzas). Puede realizar las actividades de un usuario normal, pero adicionalmente puede decidir si aprobar o rechazar los gastos de otros usuarios, así como visualizar todos los gastos registrados en la aplicación. incluidos gastos de otros usuarios. Incluye otras funciones como crear, ver, modificar o borrar clientes, proyectos y tarjetas de crédito de la empresa.

Adicionalmente, *ServiceNow* incluye por defecto una serie de roles, en los que se encuentra el rol de administrador (admin)[6]. Dicho rol permite acceder a toda la funcionalidad de la instancia, incluyendo todas las aplicaciones *"scopeadas"*. Naturalmente siendo yo el usuario administrador tengo acceso a todas las opciones de configuración de *ServiceNow*, con la excepción de los *Interactive Filters*.

3.2. Ciclo básico de la aplicación

A continuación una vez definidos los tipos de usuario se procederá a explicar que ciclo sigue la aplicación en cuanto al manejo de gastos en la aplicación.
Lo primero que debemos saber es que los gastos, o **líneas de gasto**, definen que características tiene el gasto en cuestión, como por ejemplo la fecha en la que se hizo, el tipo de servicio que cubren, el valor del IVA, la divisa utilizada y que usuario la realizó. Sin embargo para la gestión periódica de

las líneas de gasto existe una forma de englobarlas en base a varios parámetros en común.
Para cada usuario y periodo, existe una serie de gastos sobre los que los mánagers deciden que líneas de gasto aprobar como gasto empresarial, y cuales no, y luego deciden si dicho conjunto de líneas de gasto es válido o no, y a partir de ahí se decide si se cubren dichos gastos o no. Dicho conjunto de líneas de gasto se definirá como **lista de gastos**.
El ciclo de la línea de gastos es definida en el siguiente diagrama:

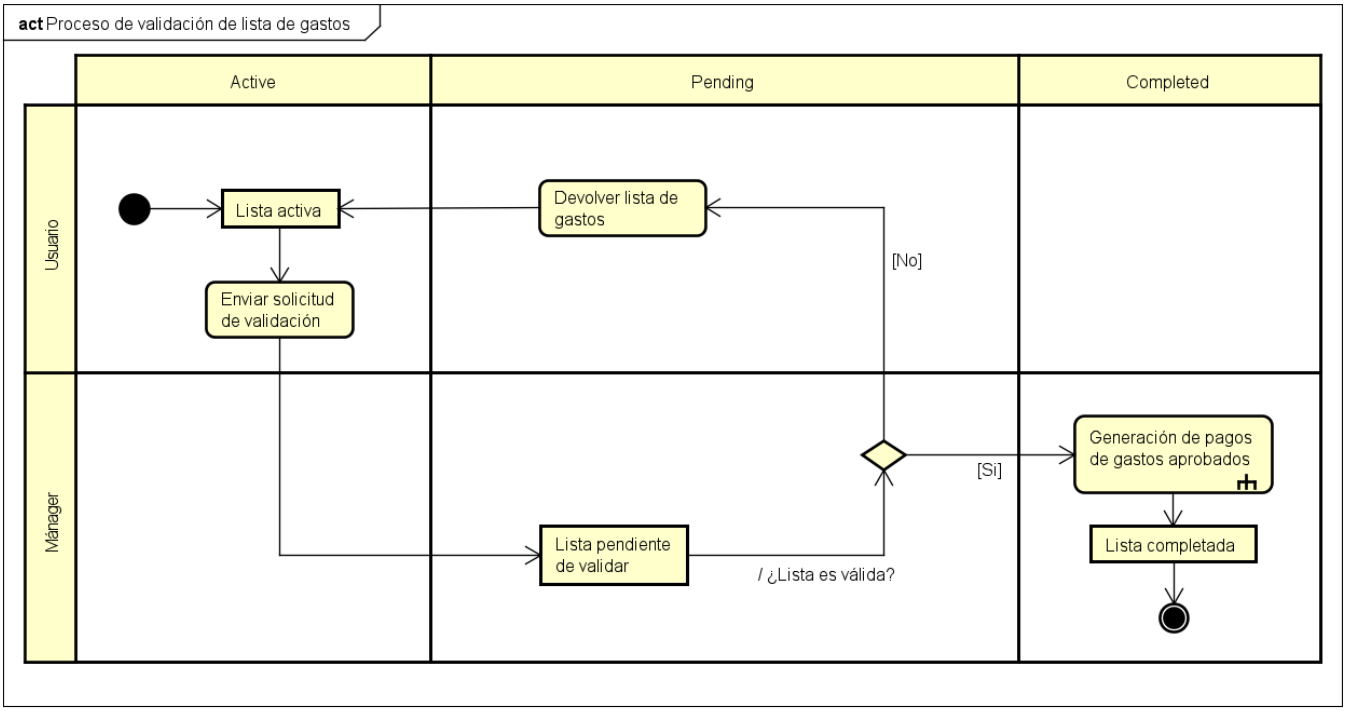


Figura 3.1: Proceso de validación de listas de gasto

Dentro del proceso de listas se distingue los siguientes estados:

- Active: Una lista se crea en este estado. Mientras esté activo, se permitirá añadir gastos al grupo.
- Pending: Sigue permitiendo añadir nuevas líneas de gasto, pero adicionalmente permite a los mánagers aprobar o rechazar las líneas de gasto en su interior.
- Completed: Se llega cuando es validado por un mánager.

El proceso es el siguiente:

- Una vez creada la lista de gastos (ya sea a mano por un mánager a automáticamente a principios de mes), el usuario o un mánager puede introducir en la lista de gastos tantas líneas como quiera.
- Una vez terminado el usuario puede enviar la lista a validar, y esta pasa al estado "Pending". En este estado un mánager pueda decidir que líneas de gasto son aceptables para aprobar o no, y decide si la lista es válida o no.

- Si el mánager rechaza la lista, esta vuelve al estado "Active", y las líneas que fuesen rechazadas por el mánager (no las canceladas) vuelven a estar pendientes, de forma que el usuario puede modificar los valores pertinentes o simplemente cancelarlas para que no se cuenten en el gasto.
- Si el mánager aprueba la lista, esta pasa al estado "Completed", y se crean los pagos correspondientes. Siempre se crearán un pago para las líneas de gasto inmediatas pagadas en metálico, mas un pago por cada línea de gasto inmediata pagada con tarjeta de crédito.

3.2.1. Gastos inmediatos y gastos planificados

Evidentemente, además de las listas de gasto, las propias líneas siguen un ciclo similar, con sus propios estados. Dentro de la aplicación cabe destacar que hay dos tipos de procesos en los gastos. Uno son los gastos inmediatos, que son aquellos que el empleado (usuario) realiza pagando en el momento de su adquisición. Un ejemplo sería el gasto al pedir un taxi, un billete para el metro, o o el pago de un refrigerio con clientes.
El otro tipo de gastos son los planificados, que son los que se negocian entre el usuario y el mánager de antemano, y siguen su propio ciclo, aunque también estén asociados a una lista de gastos. Un ejemplo sería la reserva de una habitación de un hotel, o una reserva de un vuelo a otro país. En el caso de las líneas de gasto de carácter inmediato, este sería su ciclo dentro de la aplicación, junto con los siguientes estados:

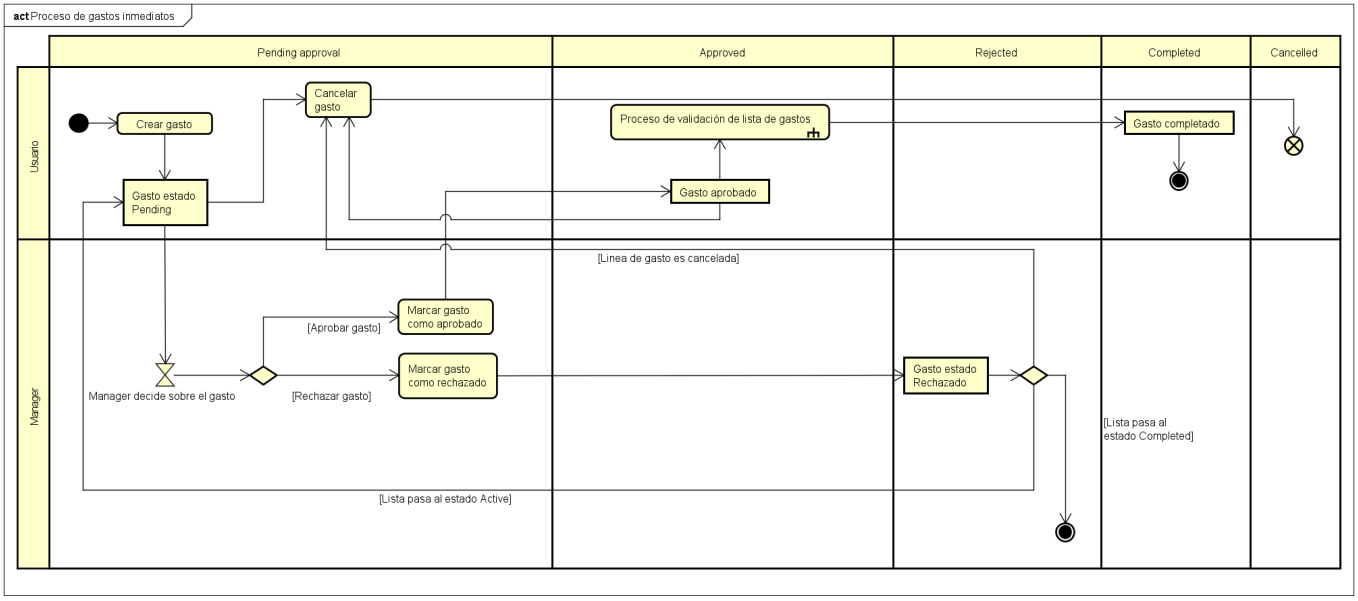


Figura 3.2: Proceso de gastos inmediatos

- Pending: Cuando se crea un nuevo estado, y está pendiente de aprobación.
- Approved: El gasto está aprobado como gasto de empresa.
- Rejected: El gasto ha sido rechazado como gasto de empresa.
- Completed: El gasto ya ha sido abonado al usuario , y se completa.

- Cancelled: El gasto ha sido cancelado por el mánager o el usuario, y se descarta el proceso de aprobación.

Donde el ciclo se describe como sigue:

1. El usuario crea un nuevo gasto, introduciendo los datos necesarios para este, incluyendo como es el proceso de gasto. En ambos casos en la creación de pone el gasto en estado "Pending".
2. La línea de gasto permanece en es estado hasta que la lista correspondiente pase al estado "Pending", es entonces donde el mánager decide que hacer con las líneas de gasto.
3. Si la línea es aprobada, pasa al estado "Approved", hasta que se valide la lista de gastos.
4. Si la lista correspondiente pasa al estado "Completed", la línea de gasto también lo hará.
5. Si la línea es rechazada, pasa al estado "Rejected". En caso de que la lista entera sea rechazada la línea vuelve al estado "Pending", pero si la lista es validada no cambiará el estado de la línea de gasto.
6. Mientras la linea no esté en estado "Completed", si se cancela la línea pasará al estado "Cancelled".

Por otro lado las líneas de gasto planificadas siguen este otro proceso:

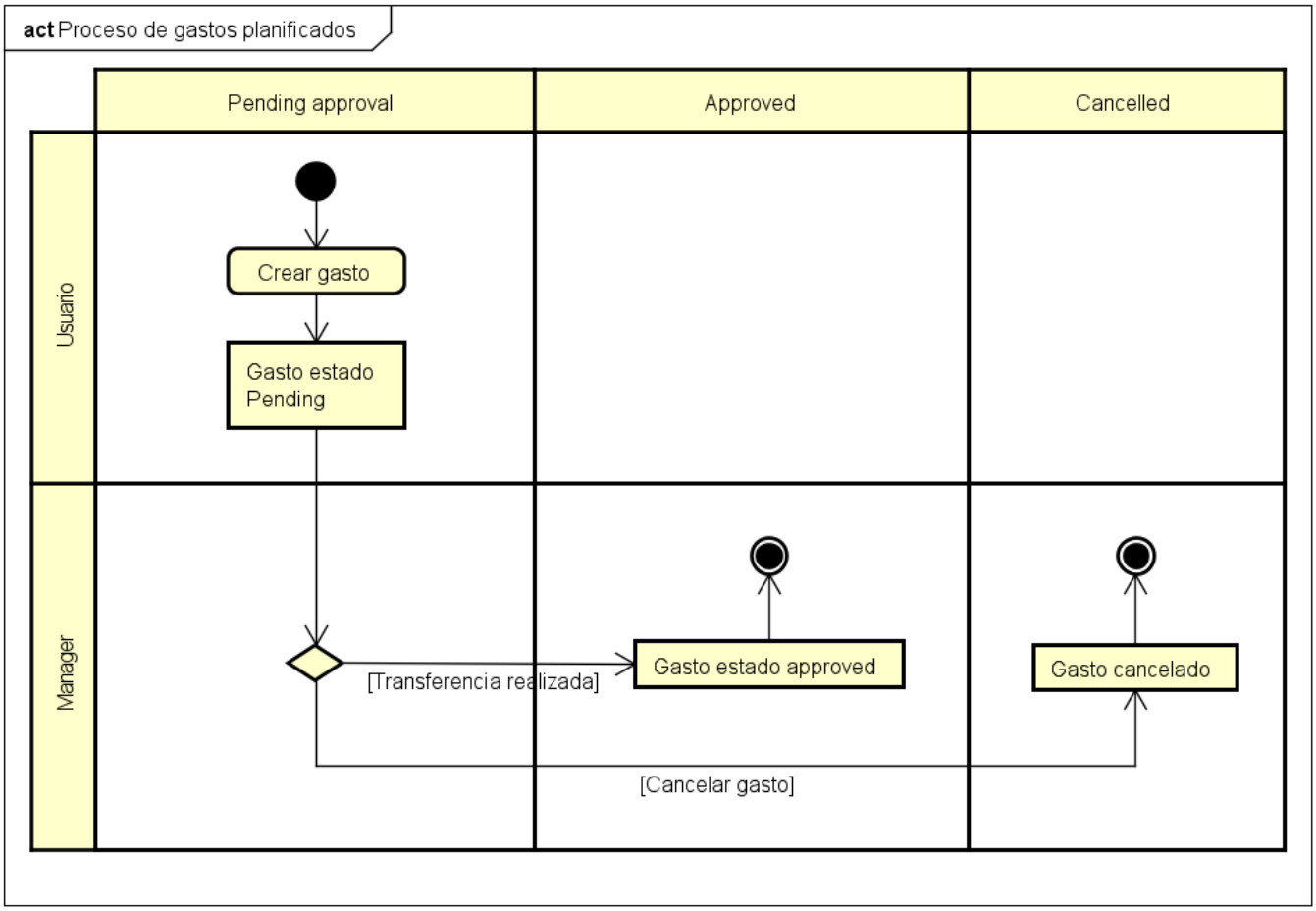


Figura 3.3: Proceso de gastos planificados

Los estados son similares al de las líneas de gasto planificadas:

- Pending: Cuando se crea un nuevo estado, y está pendiente de aprobación.
- Completed: El gasto ya ha sido abonado al usuario Empleado, y se completa.
- Cancelled: El gasto ha sido cancelado por el mánager o el usuario, y se descarta el proceso de aprobación.

Y siguen este proceso:

1. Tras una reunión, el usuario Empleado crea una línea de gasto rellenando los datos necesarios, con el método de proceso a "Planning". Tras crearlo, el estado pasa a ser "Pending".
2. Luego el usuario Mánager puede hacer dos cosas:
3. Si decide aprobar la línea de gasto, puede crear la transferencia en el sistema, donde se crearía el pago de dicha línea de gasto, y pasaría directamente al estado "Completed".
4. Puede cancelar la línea de gasto, pasando al estado "Cancelled". El propio usuario puede también cancelar la línea gasto si lo considera necesario.

3.3. Otros procesos

3.3.1. Creación mensual de listas de gasto automatizadas

Una de las cosas importantes para la aplicación es que haya una forma de tener hechas listas de gastos para todos los usuarios que usen la aplicación. Para ello sería necesario tener que automatizar el proceso de creación de listas mensual. Dicho proceso sigue este diagrama:

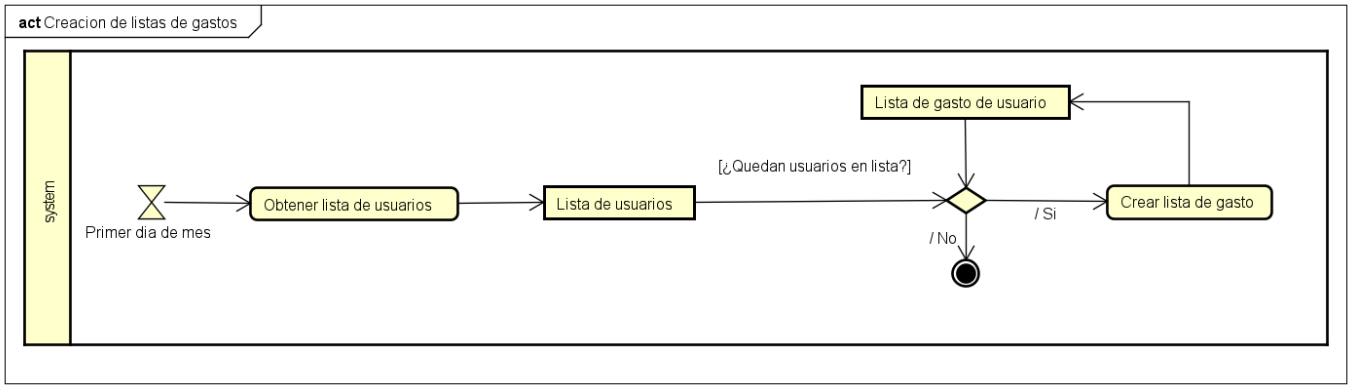


Figura 3.4: Proceso de creación de listas de gasto

El sistema simplemente miraría los usuarios con rol de la aplicación, y por cada un crearía una lista de gastos vacía, con periodo definido en el nuevo mes. Dicho proceso se ejecutará al primer día de cada mes.

3.3.2. Importación y exportación de lineas de gasto

El siguiente diagrama describe los procedimientos de importación y exportación de gastos de la aplicación:

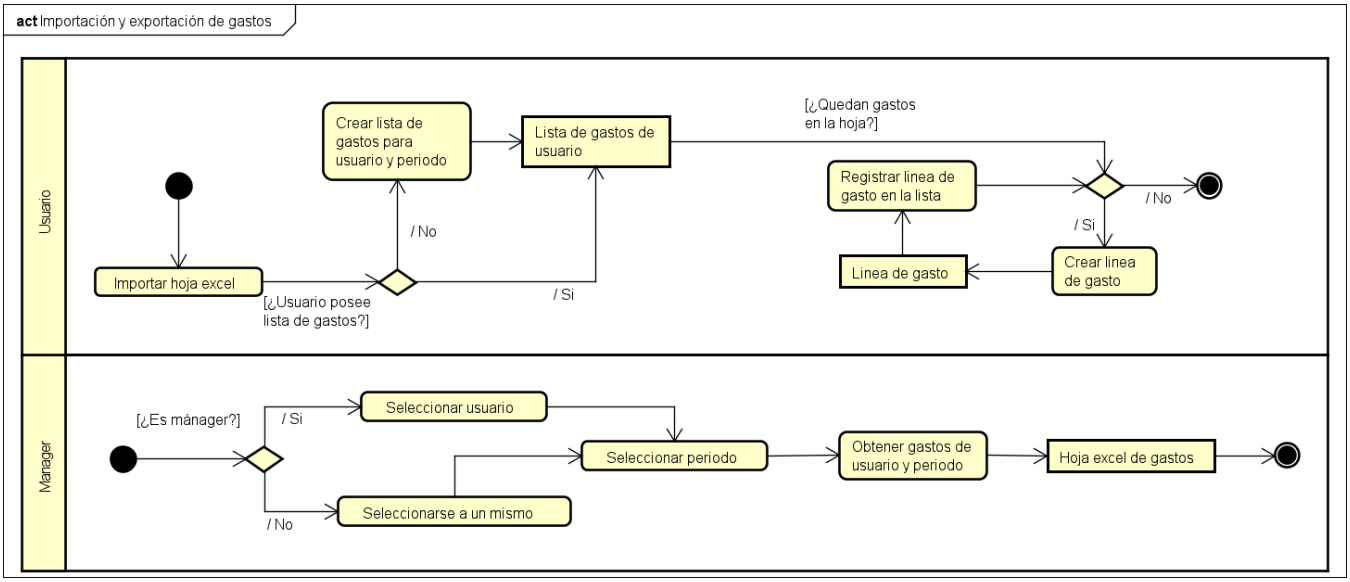


Figura 3.5: Proceso de importación y exportación

En el caso de la importaciones, si el usuario en un Empleado, solo puede importar sus gastos, pero los mánagers pueden importar los gastos de cualquier otro usuario. Para que sea posible la importación, tanto el nombre como el periodo deben ser correctos, y debe de haber al menos un gasto en la tabla. Los gastos se incluirán en la lista indicada por el periodo del usuario, pero denegará la inserción de gastos en una lista ya completada. Este sería un ejemplo de hoja Excel a importar:

SILVER STORM SOLUTIONS

HOJA DE GASTOS

Nombre: Daniel de Vicente Garrote — Periodo: abr-20

			Otras Divisas				Euros									Efectivo	Tarjeta
Fecha	Descripción	Código Proyecto	Tipo *	Moneda	Importe	€	nª KM	€/KM	KM	Transporte	Hotel	Comidas	Resto	IVA	TOTAL		empresa
09-nov-19	Ida-Vuelta Zamora	ABC					100	0,17	17,00						17,00	17,00	
09-nov-19	Autopista VA7	ABC					0	0,17	0,00	5,25					5,25		5,25
09-nov-19	Comida	ABC					0	0,17	0,00			26,49			26,49		26,49
09-nov-19	Cena 4 personas	ABC					0	0,17	0,00			84,50			84,50		84,5
10-nov-19	Desayuno	ABC					0	0,17	0,00			12,25			12,25		12,25
10-nov-19	Comida	ABC					0	0,17	0,00			24,45			24,45		24,45
10-nov-19	Hotel	ABC					0	0,17	0,00		75,24				75,24		75,24
10-nov-19	Autopista VA7	ABC					0	0,17	0,00	5,25					5,25		5,25
27-nov-19	Ida-Vuelta Zamora	ABC					100	0,17	17,00						17,00	17,00	
27-nov-19	Autopista VA7	ABC					0	0,17	0,00	5,25					5,25		5,25
					Total	0			34,00	15,75	75,24	147,69	0,00	0,00	272,68	34,00	238,68

* Tipo: T: Transporte H: Hotel C: Comidas O: Otros

A pagar al trabajador: 34,00

Recepción

Control

Contabilización

1 de 1

Figura 3.6: Ejemplo de Excel de gastos

Para las exportaciones, un usuario Empleado solo puede exportar sus propios gastos, y un mánager puede exportar los de cualquiera. Sin embargo los dos puede realizar exportaciones en cualquier pe-

riodo. La exportación generará un archivo Excel con los registros seleccionados.

3.3.3. Creación de tarjetas de crédito

Debido a la necesidad de usar tarjetas de crédito otorgadas por la empresa *SilverStorm*, es necesario tenerlas registradas en la aplicación, de forma que se pueda saber que gastos se van a cubrir con ella, así como ver si están activas, si están caducadas, y poder crear pagos a dichas tarjetas. El proceso es exclusivo de los mánagers, pues solo ellos pueden asignar las tarjetas a los usuarios. El diagrama para la creación de tarjetas de crédito sería el siguiente:

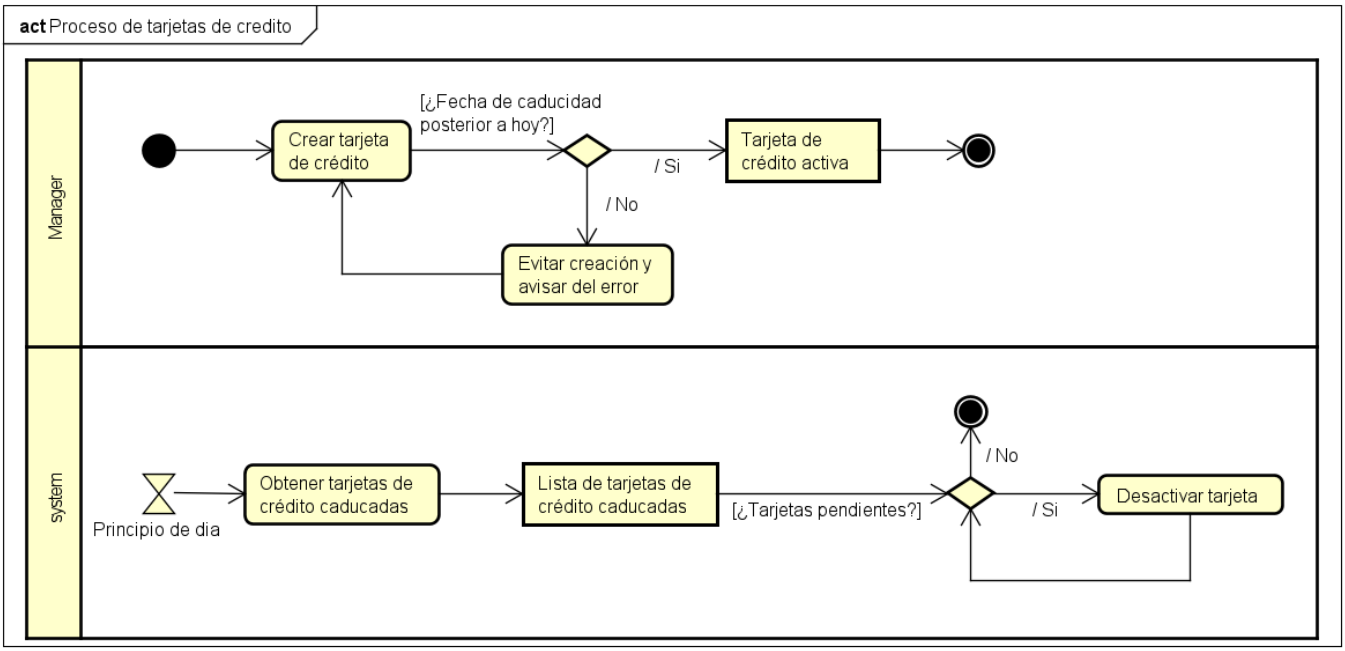


Figura 3.7: Proceso de tarjetas de crédito

El ciclo para su creación es el siguiente:

1. El mánager accede al módulo de tarjetas de crédito.
2. El sistema muestra la lista con las tarjetas de crédito.
3. El mánager pulsa en el botón "New".
4. El sistema muestra el formulario de la tabla de tarjetas de crédito.
5. El mánager introduce los datos de la tarjeta: Número de tarjeta, número de cuenta, fecha de caducidad y usuario a la que pertenece. Tras esto elige guardar el registro.
6. El sistema comprueba que la fecha de caducidad es posterior a la fecha actual, si no lo es cancela el guardado. Si es válida la fecha guarda la tarjeta, poniendo el marcador de Activo a True.

La aplicación comprueba cada día las tarjetas de crédito cuya fecha de caducidad sea la fecha actual y que aún estén activas, y las desactiva. El sistema comprueba cada día las tarjeta para ver si se han caducado, y si alguna se caduca, el sistema pone el valor del campo "Active" a false, impidiendo usar la tarjeta en futuros gastos. La aplicación seguirá creando pago a las tarjetas desactivadas si dichas líneas de gasto fueron hechas antes de su caducidad.

3.3.4. Creación de un cliente y sus proyectos

Todos las líneas de gasto siempre van asociadas a un proyecto registrado en la empresa, que a su ver pertenece a un cliente. La aplicación debe poder registrar que líneas de gasto están relacionadas para ver como se dirigen los gastos de la empresa. El ciclo básico es el siguiente:

1. El mánager accede al módulo de clientes.
2. El sistema muestra la lista de clientes.
3. El mánager pulsa el botón "New".
4. El sistema muestra el formulario de la tabla de clientes.
5. El mánager introduce los datos necesarios del cliente. Ningún campo es obligatorio pero se pueden introducir valores como el nombre, datos de una dirección como calle, provincia, dirección postal... Una vez introducidos el usuario decide guardar el registro.
6. El sistema guarda al cliente en la aplicación.
7. El mánager vuelve al menú principal, y elige el módulo de proyectos.
8. El sistema muestra la lista de proyectos.
9. El mánager pulsa el botón "New".
10. El sistema muestra el formulario de la tabla de proyectos.
11. El mánager introduce los valores en los campos. Los campos obligatorios son: Nombre del proyecto, complejidad del proyecto, tipo de proyecto y cliente al que va asociado (creado anteriormente). Después seleccionar guardar el proyecto.
12. El sistema guarda el registro del proyecto, que se asocia al cliente con el campo de tipo Referencia.

3.3.5. Creación de nuevos usuarios

Los administradores puede crear nuevos usuarios en la aplicación, así como asignarles los roles de la aplicación (usuario o mánager). El ciclo básico es el siguiente:

1. El administrador accede al módulo de usuarios.
2. El sistema muestra la lista de usuarios.
3. El administrador pulsa el botón "New".
4. El sistema muestra el formulario de la tabla de usuarios(sys_user).
5. El administrador introduce los siguientes datos: ID de usuario, nombre y apellidos.
6. El sistema registra al usuario, creando una lista de gastos para él en el periodo actual.

Una vez creado el usuario, el administrador puede asignar los roles de aplicación a dicho usuario. Para ello solo tiene que ir a la lista de roles, buscar el rol a asignar y añadir al usuario en la lista de usuarios con el rol.

3.4. Matriz de responsabilidades RACI

A continuación es necesario definir que actividades concretas debe hacer cada tipo de usuario. Para ello definimos la matriz de responsabilidades, o matriz RACI. En la tabla se define que tipo de usuario y actividad son posibles. Tras haber definido tipos de usuarios y actividades, las opciones se describen tal y como salen debajo:

Actividad	User	Manager
Creación de lineas de gasto externas o planificadas	X	X
Creación de líneas de gasto inmediatas	X	X
Creación de pagos en líneas de gasto planificadas		X
Visualización de listas y líneas de gasto propias	X	X
Visualización de todas las listas de gastos de la aplicación		X
Modificación de líneas de gasto propias	X	X
Modificación de todas las líneas de gasto de la aplicación		X
Borrado de líneas de gasto propias	X	X
Borrado de todas las líneas de gastos de la aplicación		X
Importación de listas de gastos propias	X	X
Importación de listas de gastos ajenas		X
Exportación de líneas de gasto propias	X	X
Exportación de todas las líneas de gasto		X
Aprobación de líneas de gastos		X
Envío de lista de gastos a validación	X	X
Validación de listas de gastos		X

Tabla 3.1: Matriz RACI

Se puede apreciar que algunas tareas no son posibles para el usuario normal, pues no pueden ver gastos creados por otros usuarios, sino solo los suyos propios.

Capítulo 4

Historias de usuario y elemento técnicos de la aplicación

4.1. Definición de roles de usuario

Nuestra primera historia de usuario comienza añadiendo los roles de usuario que usaremos en nuestra instancia de *ServiceNow*. Como vimos en el desarrollo del proceso de aprobación de listas de gasto, existen dos tipos de usuario: Usuarios normales que solo registran gastos, y mánagers que además de poder añadir también sus gastos administran los gastos de otros. En este caso solo hay que crear un rol para cada tipo de usuario:

Rol *ServiceNow*	Usuario Proceso
bill_user	Usuario normal o empleado
bill_mánager	Usuario mánager o supervisor

Tabla 4.1: Tabla de asignación de roles

Dentro de cada rol existen una serie de acciones que cada tipo de usuario puede realizar. Dichas acciones se describen en la siguiente tabla:

Rol *ServiceNow*	Resumen de funcionalidad
bill_user	Podrá crear lineas de gastos así como nuevas listas de gastos para sí mismo. Podrá ver, modificar o borrar solo las lineas de gasto creadas por él mismo. Podrá ver, modificar o borrar solo las listas de gastos creadas por él mismo. Podrá importar una hoja de cálculo con una serie de gastos, relacionados con su usuario y un periodo concreto. Dichos gastos se asociarán al usuario que las importó. Podrá exportar sus propios gastos, en cualquier periodo (mes y año).
bill_mánager	Podrá crear lineas de gastos así como nuevas hojas de gastos para el usuario mismo. Podrá ver, modificar o borrar los gastos de todos los usuarios, incluyendo los suyos propios. Podrá importar una hoja de cálculo con una serie de gastos, relacionados con un usuario y un periodo concretos. Podrá exportar los gastos de cualquier usuario en cualquier periodo (mes y año). Podrá crear y modificar listas de gasto para cualquier usuario.

Tabla 4.2: Funciones de los roles

Una cosa que tenemos que tener en cuenta en cuanto a acciones de cada tipo de usuario, es que los mánagers pueden también actuar como usuarios normales. En este caso *ServiceNow* ofrece la posibilidad de que un rol contenga a otro. Eso hace que los usuario que tengan asignado el rol que contenga al otro también tengan asignado ese mismo rol en ellos, facilitando la obtención de permisos en la aplicación. En nuestro caso el rol de mánager contiene el rol de usuario, ya que los mánagers también tienen acceso a las acciones de los usuarios normales, mas las suyas propias.

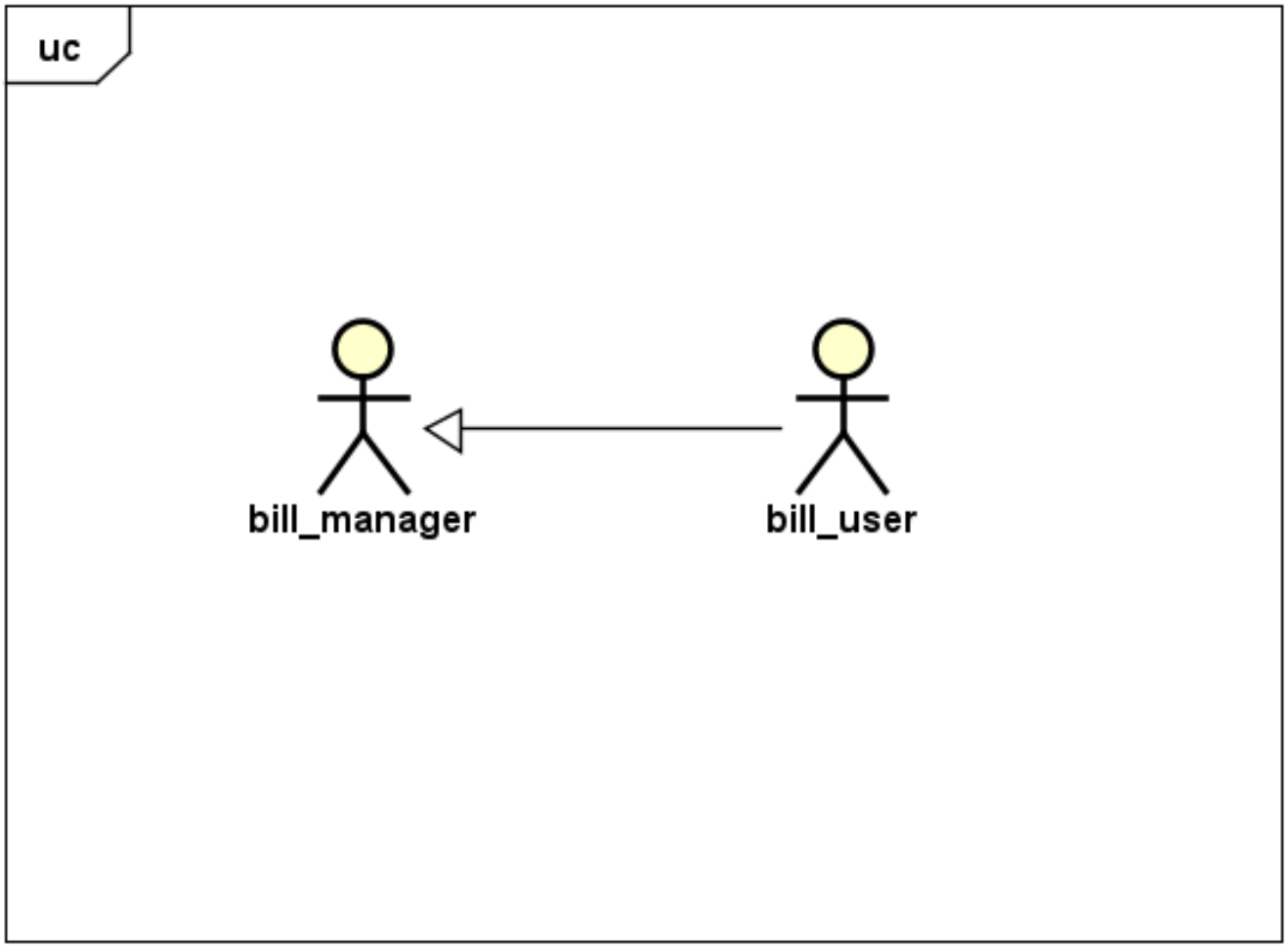


Figura 4.1: Jerarquía de roles de la aplicación

4.2. Diagrama de clases

Una vez conocidos los procesos, sabemos cuales son las entidades importantes de nuestra aplicación. En nuestro caso se crearán cuatro entidades principales, mas unas cuantas secundarias que servirán de apoyo a otras funcionalidades de la aplicación. En el siguiente diagrama se muestran las entidades de la aplicación: Cabe destacar como se relacionan las siguientes entidades dentro del diagrama de la aplicación: Cada usuario puede pertenecer a varios grupos o a ninguno, de los cuales cada grupo puede tener o no asignados roles. Dicho funcionamiento permite asignar de forma masiva los roles de la aplicación a usuarios, con solo añadirles a los grupos que tengan asignados un rol.
Luego cada usuario tendrá una serie de listas de gasto, las cuales a su vez pueden contener lineas de gasto, o simplemente no tener lineas de gasto. Luego cada linea de gasto puede tener asignado un pago o no, dependiendo de si se ha creado un pago para ella o no, es decir, si se ha completado la línea o no. De esta forma se pueden agrupar las líneas de gasto de un usuario para un periodo de tiempo concreto, que en nuestro caso será un mes entero.
Para cada línea de gasto se le debe asociar su correspondiente proyecto, que pertenecerá a un cliente. Así mismo los clientes de la aplicación contarán con varios proyectos dentro de la instancia para ser

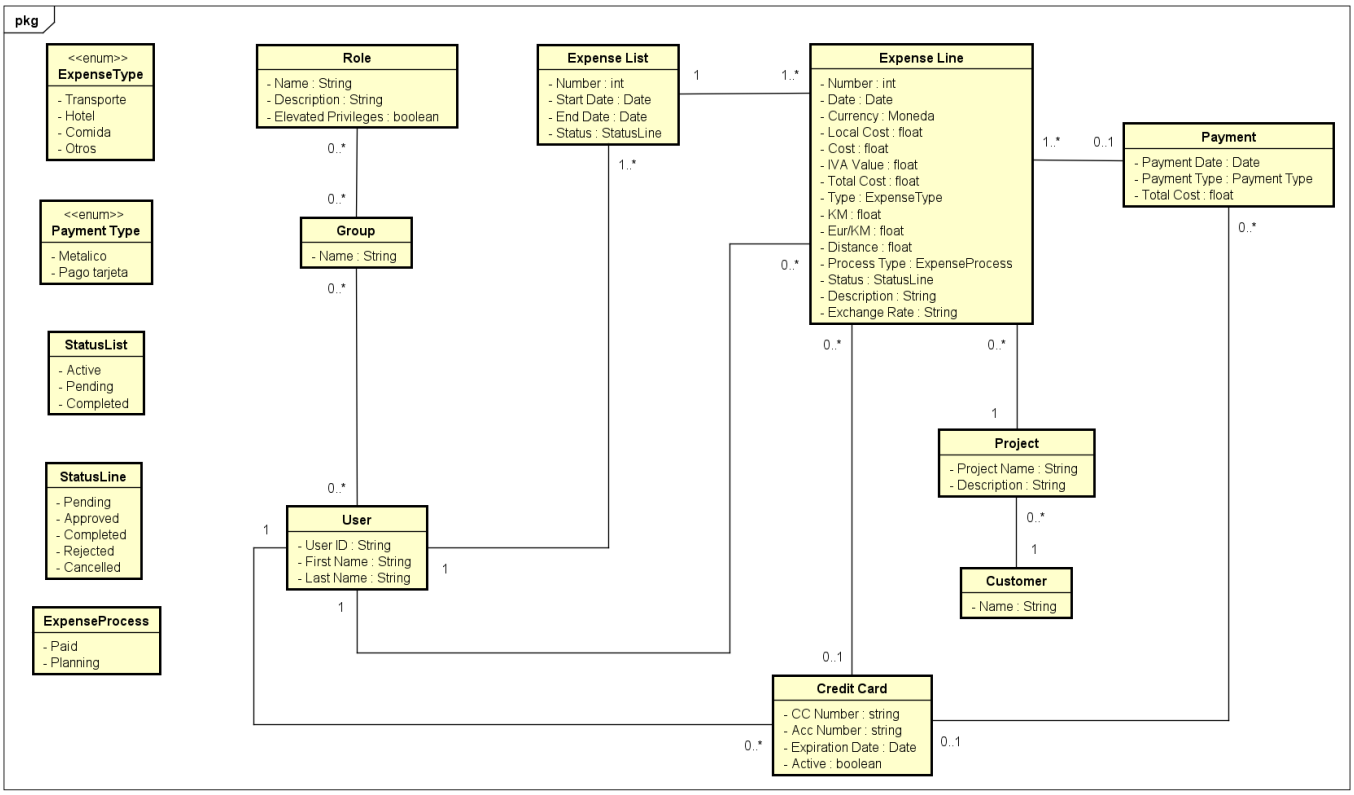


Figura 4.2: Diagrama de entidades de la aplicación

referenciados por las líneas de gasto, así como usados por otras aplicaciones de la instancia.
Por último los usuarios pueden tener asignadas tarjetas de crédito de forma opcional, que luego son referenciadas en las líneas de gasto si se han pagado con dicha tarjeta, de forma que se pueda identificar fácilmente si una línea de gasto ha sido pagada con tarjeta o en efectivo.
Hay que tener en cuenta que las entidades de usuario, roles, grupos, clientes y proyectos son usadas por las aplicaciones de la instancia, lo cual implica que su funcionamiento interno no debe ser afectado por la creación de las nuevas entidades. La idea es simplemente asociar los gastos a los proyectos mediante simples referencias.

4.3. Entidad *Expense List*

Una de las primeras entidades (o tablas) que hay que crear en la aplicación es la de listas de gasto, llamada **Expense List**. Dicha tabla contendrá los datos de la lista y englobará una serie de líneas de gasto mediante referencias a la lista. Dicha tabla contendrá los siguientes campos:

- Number: Auto-numérico de SerivceNow, añadido durante su creación.
- User: Usuario al que pertenece la lista de gastos, es una referencia en un registro de la tabla de usuarios (sys_user).
- Total cost: Coste total de las líneas de gasto dentro de la lista. Se actualiza cada vez que se añada, borre o modifique una línea de gasto dentro de la lista.

- Start Date: Fecha de comienzo del periodo de gastos de la lista. Aunque es modificable por los mánagers, por defecto la creación automática la sitúa en el primer día del nuevo mes.
- End Date: Fecha de finalización del periodo de gastos de la lista. Aunque es modificable por los mánagers, por defecto la creación automática la sitúa en el último día del nuevo mes.
- Status: Estado en el que se encuentra la lista. Por defecto en su creación está en "Active" (activa), pero al solicitar validación por parte del usuario, cambia al estado "Pending" (el mánager decide aquí si la lista cumple con los requisitos para remunerar los gastos del usuario). Una vez validada el estado pasa a "Completed" (la lista se ha validado y se crean los pagos de los gastos aprobados).

La entidad tendrá los campos del formulario y de la lista de una forma determinada. En nuestro caso la lista quedará en este orden de columnas de izquierda a derecha: Number, Total Cost, User. En el caso del formulario los elementos se colocarán en esta disposición:

Number	User
Total cost	Status
Start Date	End Date

Tabla 4.3: Orden del formulario de la tabla de listas de gastos

Por último tendrá que mostrar una lista con las líneas de gasto asociadas a la lista, en nuestro caso simplemente tendremos que añadir una *Related List*, o lista relacionada, que coja las líneas de gasto que hagan referencia a la lista. Dicha lista relacionada permitirá ver las líneas de gasto, y acceder al formulario de línea de gasto con el botón "New" visible junto a la lista relacionada.

4.4. Entidad *Credit Card*

La siguiente entidad o tabla a crear será la de las tarjetas de crédito, llamada **Credit Card**, las cuales albergarán las tarjetas que los usuarios usarán para registrar gastos en ella. En este caso es similar a otras entidades, creamos la tabla con su nombre, y cada uno de los campos a usar. Dichos campos serán los siguientes:

- CC Number: Número de la tarjeta. Es obligatorio rellenarla.
- Acc Number: Número de la cuenta asociada a la tarjeta. Es obligatorio rellenarla.
- User: Usuario al que pertenece la tarjeta: Es obligatorio rellenarla.
- Expiration Date: Fecha de caducidad de la tarjeta. Es obligatorio rellenarla, y debe situarse en el futuro con respecto al momento en el que se esté creando o actualizando.
- Active: Indica si la tarjeta esta activa. Esta a True por defecto, y se desactiva automáticamente al caducarse.

Para dicha entidad la lista de tarjetas de crédito tendrá las columnas en el siguiente orden: CC Number, User, Active. El formulario se colocará de la siguiente forma:

CC Number	Acc Number
User	Active
Expiration Date	

Tabla 4.4: Orden del formulario de la tabla de tarjetas de crédito

Por último se tendrá que añadir una *Related List* para hacer referencia los pagos hechos con la tarjeta. Esta lista relacionada no permitirá crear nuevos pagos a mano, ya que el sistema tendrá que crearlos todos automáticamente al aprobar los gastos.

4.5. Entidad *Expense Line*

La siguiente entidad es la tabla de líneas de gasto, o tabla **Expense Line**. Dicha tabla contendrá los datos de los gastos generados por los usuarios y será la que contenga mas información. Dicha tabla contendrá los siguientes campos:

- Date: Fecha en la que se realizó el gasto. Es un campo obligatorio, y debe representar una fecha dentro del periodo de la lista en la que está asignada.
- Currency: Divisa con la que se realizó el pago. Es obligatorio ponerla, y usa una tabla existente por defecto en *ServiceNow* (*fx_currency*), de la cual se obtiene el ratio mas reciente para la conversión.
- Local cost: Coste en moneda local en donde se realizo el gasto. Es obligatorio y debe ser un valor numérico.
- Cost: Coste convertido a Euros del gasto. No es editable pero se obtiene con la conversión de Local Cost.
- IVA Value: Coste del valor del IVA del gasto. Es obligatorio, y debe reflejar su valor en Euros.
- Total cost: Coste total del gasto. No es editable, pero se obtiene a partir del coste convertido mas el IVA.
- Distance: Distancia de viaje. Solo es usable cuando el gasto es de tipo Transporte.
- Cost per KM: Coste en euros por kilómetro viajado. Solo es usable cuando el gasto es de tipo Transporte.
- Project: Proyecto asociado a la línea de gasto. Usa la tabla pm_project de *ServiceNow* (por defecto con un plugin), y es obligatoria en cada línea.

- User: Usuario al que pertenece la lista. Es obligatorio, aunque los usuarios normales solo puede seleccionarse a si mismos.

- Expense List: Lista de gastos a la que pertenece la linea. Es obligatorio, pero solo se puede seleccionar listas del usuario del campo User que estén activas.

- Number: Auto-numérico de *ServiceNow*.

- Credit Card: Tarjeta de crédito con la que se realizó el pago. Solo es obligatoria si el método de pago es con tarjeta, y solo permite usar tarjetas activas del usuario.

- Payment Method: Representa el método de pago de la línea de gasto. Puede ser efectivo o con tarjeta de empresa.

- Type: Tipo de línea de gasto que se ha registrado. Los tipos que permite registrar son Transporte, Hotel, Comida u Otros.

- Description: Descripción breve del gasto registrado. Es obligatorio rellenar.

- Additional comments: Campo de bitácora usado para anotar comentarios sobre la línea de gasto. Necesario cuando se apruebe o rechace la línea de gasto.

- Process Type: Proceso que se ha seguido en la línea de gasto. El proceso puede ser *Paid* (el usuario ya realizó el pago en un momento concreto), o *Planning* (el gasto se planifica para un futuro cercano, útil para diferentes tipos de reservas).

- Manager: Campo oculto que define al usuario mánager que hizo una aprobación o rechazo de la línea de gasto. Usado como referencia en las notificaciones.

- Status: Estado de la línea de gasto. No es editable desde el formulario y tiene varios estados: "Pending", "Approved", "Completed", "Rejected" y "Cancelled".

- Payment: Pago asociado al gasto. De solo lectura, y se rellena cuando un pago cubre dicha línea de gasto, ya sea al aprobar unja línea de gasto con tarjeta, al crear un pago para un gasto planificado, o al validar una lista de gastos.

- Exchange Rate: Muestra el ratio de conversión de la moneda local (Local cost) a Euros. Se rellena cuando se obtiene el valor de *Currency* y *Date*.

La lista de la tabla tendrá visible los siguientes campos en orden: Number, User, Description, Total Cost, Type, Payment Method. El formulario contendrá los campos en la siguiente distribución: La sección principal con estos campos:

Number	Date
User	Expense List
Process Type	Project
Type	Status
Distance	Cost per KM
Total Cost	IVA Value
Payment Method	Credit Card
Description	
Additional Comments	

Tabla 4.5: Orden del formulario de la tabla de lineas de gasto (sección principal)

Y una sección para la conversión de divisa con estos otros campos:

Local Cost	Currency
Cost	Exchange Rate

Tabla 4.6: Orden del formulario de la tabla de lineas de gasto (sección conversión)

4.6. Entidad *Payments*

La última entidad nueva a crear es la entidad de pagos, llamada **Payment**. En dicha entidad se almacenan los datos del pagos a usuarios de gastos aprobados, incluyendo cuando se realizó, la forma de pago del gasto, y que gastos concretos cubre. La lista de la tabla contendrá los campos en el siguiente orden: User, Payment Date, Payment Type, Total Cost. En cuanto al formulario se colocarán los campos en el siguiente orden:

Payment Date	Payment Type
Credit Card	User
Total Cost	

Tabla 4.7: Orden del formulario de la tabla de pagos

Esta entidad también tendrá asociados una serie de líneas de gasto que cubrirá, para ello crearemos una *Related List que las mostrará*, pero no permitirá añadir nuevos gastos, ya que los pagos se crearán de forma automática por el sistema.

4.7. Entidades predefinidas

Aparte de las tablas ya creadas por nosotros, haremos uso de otras tablas base de *ServiceNow*, las cuales complementarán la funcionalidad de nuestras tablas para permitir que los procesos de la aplicación funcionen. La ventaja es que aunque son usadas por otras aplicaciones de la instancia, no

es necesario configurar nada sobre ellas, ya que nos ofrecen los necesario. Dichas tablas son la tabla de usuarios **sus_user**, la tabla de roles **sys_user_role**, la tabla de compañías **core_company** y la tabla de proyectos **pm_project** disponible con el plugin PPM Standard[15].
Dichas entidades nos ofrecen de base la posibilidad de crear nuevos usuarios[8], gestionar roles[7] y añadirlos a usuario[4], y poder usar proyectos y compañías de forma fácil. Adicionalmente la tabla de grupos permite agrupar a los usuarios y poder ponerles un rol a todos los miembros del grupo, asignando dicho rol al propio grupo[3]

4.8. Creación de permisos ACL

Aun teniendo las tablas necesarias creadas, es necesario poder regular que tipo de acciones puede hacer cada tipo de usuario sobre las nuevas entidades, así como el acceso a las entidades predefinidas. Para ello *ServiceNow* trae por defecto un sistema de control de accesos, conocido como *Access Control List*. Dicho sistema se compone de una tabla base del sistema, llamada **sys_security_acl**, en la cual se registra entre otras cosas el tipo de elemento a tratar (normalmente registros de tablas) y el tipo de operación a tratar (lectura, escritura, creación, borrado, edición de celda de tabla, derecho para crear reportes...). Luego hay que definir si la ACL es a nivel de tabla (restringe el acceso a toda la tabla), o a nivel del campo (restringe el acceso a un campo de la tabla, o a todos si se selecciona la *wildcard*).
Tras definir que elementos vamos a controlar y el tipo de operación, podemos definir que criterios usar para ello: Posesión de un rol, uso de un filtro de condiciones, o el uso de un script. Es posible usar los tres a la vez en caso de que haya varias condiciones.
En nuestro caso no interesa principalmente crear ACLs para nuestras cuatro tablas customizadas, pero además añadiremos otra serie de ACLs a las tablas de proyectos y compañías, ya que es necesario que los clientes y los mánagers puedan ver los datos de ambas tablas y puedan usarlas en los campos de tipo *Reference* de las tablas customizadas.
Para la tabla de listas de gasto, los usuarios normales solo podrán visualizar sus propias tablas de gasto, y no podrán ni crear editar listas de gastos. Por otro lado los mánagers y administradores podrán ver todas las listas de gasto de la aplicación, y podrá crear, borrar y editar listas.
Por otro lado en líneas de gasto todos los usuario podrá crear líneas de gasto, pero los usuarios normales solo podrán crear líneas para uno mismo, mientras por otro lado los mánagers y administradores podrán crear líneas de gasto para cualquier usuario de la aplicación con roles. En cuanto al tema de visibilidad un usuario normal solo podrá ver sus líneas pero los mánagers podrán ver cualquier línea de gasto. En el caso de las tarjetas de crédito es similar: Los usuarios normales podrán ver solo sus tarjetas y no podrán crear nuevas ni editar las existente. Sin embargo, los mánagers podrán ver y editar todas las tarjetas, así como crear unas nuevas. Por último la tabla de pagos no permitirá a ningún usuario no administrador crear ni editar pagos, sin embargo los usuarios normales solo podrán visualizar los pagos para ellos, mientras que los mánagers podrán ver los de todos los usuarios. En cuanto a las ACL de tablas base, será necesario crearlas en la scope Global, de forma que ambos tipo de usuarios puedan ver los datos de las tablas, así como poder seleccionar proyectos en los campos de

referencia a la hora de crear líneas de gasto, además de permitir a los mánagers crear y editar clientes y proyectos.

4.9. Creación de menús de aplicación y módulos de aplicación

Antes de poder crear los elementos de interacción con las tablas es necesario crear accesos a ellas. Es nuestro caso la interfaz de ServiceNow en vista por defecto nos ofrece los siguientes elementos en pantalla: Una barra superior con opciones de usuario y personalización llamada **Banner Frame**, la ventana con el contenido de formularios listas y otros elementos de UI llamada **Content Frame**, y por último una barra lateral con los menús de la aplicación llamada **Application Navigator**. Nos interesa que los usuarios puedan ver diferentes opciones según su rol, lo que limita que opciones tienen disponibles.
Para los usuarios normales, solo habría que crear un menú de aplicación llamada **User Expenses**, que contiene las opciones específicas para usuarios sin privilegio de mánager. Dicho menú de aplicaciones contiene los siguientes módulos y su funcionalidad:

- My Credit Cards: Muestra las tarjetas de crédito del usuario actual.
- My Payments: Muestra los pagos creados para el usuario actual.
- Import Expense Sheet: Muestra una página para cargar un Excel con los gastos.
- Expenses: Modulo que contiene elementos sobre las líneas de gasto.
- Create New: Muestra el formulario de Expense List para crea una nueva linea de gasto.
- My Expenses: Muestras las líneas de gasto del usuario actual.
- My Pending Expenses: Muestra las líneas de gasto pendientes del usuario actual.
- My Approved Expenses: Muestra las líneas de gasto aprobadas del usuario actual.
- My Paid Expenses: Muestra las líneas de gasto completadas del usuario actual.
- My Expense Lists: Muestras las listas de gasto del usuario actual.

En el caso de los usuarios mánagers, además de tener visible el menú de User Expenses, tendrán visible otro menú de aplicación llamado Manager Expenses. Dicho menú contiene opciones similares al de User Expenses, mas algún módulo adicional. Dichos módulos son:

- All credit cards: Muestra todas las tarjetas de crédito de la aplicación.
- All payments: Muestra todos los pagos de la aplicación.
- Import Expense Sheet: Muestra una página para cargar un Excel con los gastos.

- Companies: Muestra la lista de clientes de la aplicación.
- Projects: Muestra la lista de proyectos de la aplicación.
- Expenses: Modulo que contiene elementos sobre las líneas de gasto.
- Create New: Muestra el formulario de Expense List para crea una nueva linea de gasto.
- All Expenses: Muestras las líneas de gasto de la aplicación.
- All Pending Expenses: Muestra las líneas de gasto pendientes de la aplicación.
- All Approved Expenses: Muestra las líneas de gasto aprobadas de la aplicación.
- All Paid Expenses: Muestra las líneas de gasto completadas de la aplicación.
- All Expense Lists: Muestras las listas de gasto de la aplicación.

4.10. Creación del ciclo básico: *UI Actions*

Para poder permitir que los usuarios puedan realizar las funcionalidades importantes de la aplicación no solo basta con que puedan crear registros en los formularios, sino que también necesitan elementos de interfaz que reaccionen al pulsar sobre ellos y envíen los eventos necesarios para permitir que los usuarios puedan gestionar las listas de gasto y sus líneas. La forma de permitir a un usuario poder interactuar con sus gastos se puede realizar fácilmente con una serie de elementos de interfaz de listas y formularios llamados **UI Actions**. Las *UI Actions* son elementos que al ser seleccionados permiten realizar acciones concretas sobre el sistema en base a scripts en lenguaje JavaScript, como por ejemplo editar el estado de un campo en otro registro, o alterar elementos del propio registro.
Lo que nos interesa a nosotros en nuestro caso es que un usuario normal pueda enviar la lista a validar por parte de un mánager. Para ello tenemos que crear una serie de *UI Actions* sobre la tabla de *Expense List*, así como otra serie de *UI Actions* sobre la tabla de *Expense Line*.
Para la tabla de *Expense Line* tendremos las siguientes *UI Actions*:

- Cancel: Pone un registro en el estado "Cancelled". Accesible cuando al usuario de la lista y cualquier mánager, siempre y cuando la línea no esté completada o cancelada.
- Approve: Pone un registro en el estado "Approved". Accesible a gastos inmediatos en estado "Pending" y solo para mánagers.
- Reject: Pone un registro en el estado "Rejected". Accesible a gastos inmediatos en estado "Pending" y solo para mánagers.
- Create Payment: Pone un registro en el estado "Completed", creando el pago correspondiente para el gasto. Accesible a gastos planificados en estado "Pending" y solo para mánagers.

En cuanto a la tabla de *Expense List* existe otra serie de *UI Actions* que nos ofrecen otras opciones necesarias para poder gestionar las listas de gasto. Las *UI Actions* son:

- Cancel All: Cancela todas las líneas de gasto pendientes de la lista. Visible con la lista sin completar y al menos una línea de gasto en un estado distinto de "Completed" o "Cancelled".

- Approve All: Aprueba todas las líneas de gasto inmediatas pendientes de la lista. Visible con la lista en estado "Pending" y al menos una línea de gasto inmediata en estado "Pending". Solo visible a mánagers.

- Reject All: Aprueba todas las líneas de gasto inmediatas pendientes de la lista. Visible con la lista en estado "Pending" y al menos una línea de gasto inmediata en estado "Pending". Solo visible a mánagers.

- Create Payments for All: Aprueba todas las líneas de gasto planificadas pendientes de la lista. Visible con la lista en estado "Pending" y al menos una línea de gasto planificada en estado "Pending". Solo visible a mánagers.

- Send to Validation: Pone la lista en estado "Pending". Solo visible a mánagers y con la lista en estado "Active".

- Validate: Pone la lista y todas las líneas aprobadas en estado "Completed", generando los pagos necesarios. Solo visible a mánagers y con la lista en estado "Pending". Si quedan líneas de pago pendientes avisará al usuario.

- Reject validation: Pone la lista en estado "Active", y todas las líneas de gasto rechazadas en estado "Pending". Solo visible a mánagers y con la lista en estado "Pending".

Una vez hechas las *UI Actions* con la lógica necesaria, será posible a los usuarios iniciar el proceso de aprobación o validación de listas de gasto, así como a los mánagers poder decidir sobre estas.

4.11. Notificaciones de correo

Una utilidad interesante que nos ofrece ServiceNow es la posibilidad de poder crear notificaciones de correo online o de SMS cuando se desencadenen eventos concretos. Dichas notificaciones permitirán al usuario recibir avisos de cuando sus gastos hayan sufrido algún cambio, o vayan a recibir pagos de sus listas de gastos. En este caso las notificaciones necesarias para la aplicación son:

- Approved Paid Expense: Avisa al usuario de que una línea de gasto inmediata ha sido aprobada.

- Approved Paid Expense: Avisa al usuario de que una línea de gasto inmediata ha sido aprobada.

- Rejected Expense: Avisa al usuario de que una línea de gasto ha sido rechazada.

- Created Payment: Avisa al usuario cuando se crean los pagos al validar una lista.

4.12. Importador de gastos

Una característica notoria en *ServiceNow* es la capacidad de poder importar distintos tipos de archivos y crear registros a partir de ellos. Dicha capacidad se logra mediante dos herramientas muy útiles: *Data Sources* y *Transform Maps*. Los *Data Sources* son mecanismos que obtiene datos de un archivo y crea un *Import Set* con ellos[10]. Dichos *Import Sets* se registran en una tabla creada para el *Data Source*, denominada *Import Set Table*.
Por otro lado los *Transform Maps* son un conjunto de mapeos de campos o *Field Maps* que mueven los datos desde la tabla del *Import Set* a la tabla donde queremos crear los registros[20].
Uno de los requisitos de la aplicación consiste en poder insertar archivos Excel en la aplicación, y crear líneas de gasto que irán asociadas a una lista de gasto, definida por un usuario y un periodo de tiempo de mes y año. El principal inconveniente de este mecanismo es exclusivo para uso directo de los administradores, y por lo tanto los usuarios sin rol de administrador no pueden acceder a ella.
La solución a este inconveniente pasa por adjuntar el archivo desde otro sitio y copiar el archivo adjunto al *Data Source* mediante el uso de la *API GlideSysAttachment*[5].
Para lograr este "workaround" del importador es necesario crear los siguientes elementos en la instancia:

- Un *Data Source*, que reciba archivos de tipo Excel.
- Una tabla para el *Import Set*, con tantos campos de tipo *String* como haya en cada fila del Excel.
- Un *Record Producer*, donde se insertará el Excel.
- Un *Catalog Client Script* para comprobar que solo se adjunte un archivo Excel.
- Un *Scheduled Data Import* que ejecute el *Data Source*.
- Una tabla auxiliar llamada *Expense Data Uploads*. Contendrá los siguientes campos:
  - Number: Auto-numérico de *ServiceNow*.
  - User: Campo de tipo referencia con el usuario del Excel.
  - Period: Fecha del periodo en el Excel.
  - Status: Campo de tipo Choice con los valores "Pending", "In Process", "Completed" y "Failed".
  - Expense Lines Issues: Campo de tipo referencia a otra tabla auxiliar.
  - *Record Producer*: Referencia al *Record Producer* anterior
- Otra tabla auxiliar llamada *Log Imports*, que contendrá un varios campos de tipo *String*:
  - External ID: Numero del *Import Set*.
  - Message: Mensaje de log.

    - Row: Fila del Excel.

- Un *Transform Map*, que contenga varios *Field Maps* para los campos, y dos *Transform Scripts*: Uno para la validación del Excel, y otro para tareas posteriores a la importación como el borrado del Excel del Data Source y la actualización del registro creado en el *Record Producer*, indicando si la importación ha sido correcta o no.

4.13. Exportador de gastos

Este es otro de los requisitos de la aplicación, poder exportar los gastos de la aplicación a formato Excel. En este caso el propio *ServiceNow* ya ofrece la opción de exportar datos de tablas a Excel o CSV, por lo que no es necesario crear ningún elemento de configuración[12].

4.14. Aplicación móvil sobre *Now Mobile*

ServiceNow posee la opción de acceder a las instancias mediante varias aplicaciones móviles. Dichas aplicaciones se resumen de la siguiente forma(a partir de la versión de *New York* en adelante[19]):

- Now Mobile: Esta aplicación de ServiceNow está pensada para agentes de la aplicación, y se centra en las tareas de incidentas, eventos, seguridad y otro tipo de solicitudes.
- Mobile Agent: Aplicación pensada para los empleados. Ofrece servicios que incluyen el registro de solicitudes, servicio de búsqueda global, subida de imágenes y vídeos a registros de la instancia, y la opción de usar *Virtual Agent*.
- Mobile OnBoarding: Aplicación destinada a empleados de Recursos Humanos o empleados de IT. Ofrece diferentes vistas para manejar datos relacionados con dichos cargos empresariales.
- SerivceNow Mobile Classic: Aplicación clásica de ServiceNow. Actualmente desaconsejada por su obsolescencia, y se retirará en breve.

Una vez resumida la utilidad de las diferentes aplicaciones en *SerivceNow*, escogeremos la que se adapte mejor a nuestro tipo de cliente. Si bien es usado por empleados del departamento de finanzas, otros empleados de la compañía *SilverStorm* también la usará para registrar sus gasto, por lo que en nuestro caso optaremos por *Now Mobile*, ya que es la aplicación móvil que está pensada para nuestros usuarios finales.
Dentro de nuestra aplicación móvil se comprenden los siguientes objetivos:

- Visualizar listas de gasto.
- Visualizar lineas de gasto.
- Crear lineas de gasto.
- Editar lineas de gasto sin aprobar.

Para nuestra aplicación móvil, la instancia contará con diferentes elementos que permitirán conseguir estos objetivos. Al crearlos el usuario se logueará con sus credenciales en la aplicación móvil, y mostrará la interfaz con elementos de aplicación móvil. En nuestra aplicación harán falta los siguientes elementos:

- Un Applet Launcher, llamado *Expense Application*.
- Dos Applets:
  - Uno llamado Expense Lists, cuya lista muestre para cada lista de gastos el Number, Total Cost, Status y User. La vista de formulario mostrará todos los campos creados anteriormente.
  - Otro llamado Expense Lines, cuya lista muestre para cada linea de gasto su Number, Project, Description, Total Cost, Type y Payment Method. La vista de formulario mostrará todos los campos excepto Exchange Rate y Manager. Para el formulario debe estar visible la opción de *Attachments*.
- Un *Top Menu Function* llamado *New*, que será visible en la lista de lineas de gasto.
- Un *Top Menu Function* llamado *Edit*, que será visible en el formulario de lineas de gasto.
- Varias *UI Policies* de *Mobile* que imiten el funcionamiento de la vista del formulario del escritorio, o permitan estructurar los formularios para que los datos lleguen a la instancia de forma correcta.

4.15. Dashboards, reportes y filtros interactivos

Uno de los requisitos de la aplicación consiste en la visualización de gráficos relacionados con los gastos que hay actualmente en la aplicación. Para ello *ServiceNow* ofrece una serie de elementos de representación visual de datos llamadas reportes.
Los reportes[16] ofrecen representaciones visuales de diferentes tipos sobre los datos en tiempo real. Dichos reporte se pueden adjuntar en diferentes elementos de la aplicación, entre los que destacan los Dashboards. Los Dashboards son elementos de inicio que pueden mostrar diferentes tipos de objetos, como reportes, filtros interactivos, barras de búsqueda y otros.
En nuestro caso crearemos dos *Dashboards*: Manager Dashboards para los mánagers, y User Dashboards para los usuarios normales. Manager Dashboards contendrá los siguientes elementos:

- Total cost of all expenses in a period: Reporte que muestra un número equivalente al valor total de todas las líneas de gasto de la aplicación en el periodo actual.
- Total cost of all expenses per project: Reporte que muestra un gráfico de barras con los gastos totales de la aplicación en cada proyecto.
- Total cost of all expenses over time: Reporte que muestra un gráfico de barras con los gastos totales de la aplicación para cada día del año.

- Filter Expense By Project: Filtro interactivo que clasifica los gastos de los reportes por el proyecto al que están asociados.
- Filter Expense By Status: Filtro interactivo que clasifica los gastos de los reportes por el estado en el que se encuentran.
- Filter Expenses in a period of time: Filtro interactivo personalizado que clasifica los gastos de los reportes en base al rango determinado por las fechas escritas.
- Filter Expenses by User: Filtro interactivo que clasifica los gastos de los reportes por el usuario al que pertenecen.

Y User Dashboard contendrá los siguientes elementos:

- Total cost of user expenses in a period: Reporte que muestra un número equivalente al valor total de las líneas de gasto del usuario actual en el periodo actual.
- Total cost of user expenses per project: Reporte que muestra un gráfico de barras con los gastos totales del usuario actual en cada proyecto.
- Total cost of user expenses over time: Reporte que muestra un gráfico de barras con los gastos totales del usuario actual para cada día del año.
- Filter Expense By Project: Filtro interactivo que clasifica los gastos de los reportes por el proyecto al que están asociados.
- Filter Expense By Status: Filtro interactivo que clasifica los gastos de los reportes por el estado en el que se encuentran.
- Filter Expenses in a period of time: Filtro interactivo personalizado que clasifica los gastos de los reportes en base al rango determinado por las fechas escritas.

Lo que nos deja con un total de 6 reportes, 3 filtros interactivos y un filtro interactivo customizado, o *Custom Interactive Filter*[9]

4.16. Generador automático de listas de gasto

Para facilitar la tarea de creación de listas de gasto en cada periodo, se debe crear una *Scheduled Job*[17] que automatice este proceso. Dicho *Scheduled Job* debe buscar todos los usuarios con rol de aplicación, y por cada uno crear una lista que abarque todo el nuevo mes, entre el primer y el último día del mes.

4.17. Desactivador de tarjetas de crédito caducadas

Esta otra tarea automatizada también se realizará mediante una *Scheduled Job*. Dicho *Scheduled Job* comprobará a diario y a medianoche las tarjetas de crédito de la aplicación, y si encuentra tarjetas con la misma fecha de caducidad que la fecha actual o posterior, las desactiva.

Capítulo 5

Implementación de la aplicación en la instancia

Una vez definidos los objetivos de la aplicación, los elementos que nos permitirán cumplirlos, y los tipos de usuarios, podemos empezar a implementar nuestra aplicación de *ServiceNow*. En nuestro caso utilizaremos una instancia de desarrollo proporcionada por la empresa en la que trabajo. Dicha instancia alberga otros elementos para otros propósitos, así como las tablas predeterminadas necesarias para poder implementar toda la aplicación. Lo bueno en este caso es que al usar una *scope* privada, en principio es posible controlar el acceso de archivos de aplicación entre la *scope* nuestra y los archivos en *scope* global (prohíbe el acceso si no hay permiso explícito).

5.1. Creación guiada de la aplicación

Para empezar con la implementación, lo primero es crear la *scoped* application, que nos creará una *scope* privada, y las tablas base de nuestra aplicación. Para ello nos dirigimos al Navegador de aplicaciones y vamos a **System Applications** > **Studio**. Allí con la ventana nueva del navegador se muestra una lista de las aplicaciones ya creadas, pero en nuestro caso nosotros crearemos una nueva aplicación. Durante la creación guiada nos aparecerán una serie de ventanas que nos indicarán que datos meter (ver Apéndice A) De primeras introducimos el nombre de la aplicación, el nombre interno (empieza por *x_* y sigue con los caracteres definitorios de la instancia y el nombre de la aplicación), y una descripción opcional.A continuación debemos crear los roles que tendrá nuestra aplicación. Ya teniendo definidos los roles en los pasos anteriores, simplemente añadimos los dos roles: Usuario normal, y mánager. Hay que asegurarse luego de que el rol de mánager contenga el rol de usuario, de forma que los usuarios con el rol de mánager tengan de forma automática el de usuario.
A continuación hay que determinar que tipos de vista vamos a definir en la aplicación. *ServiceNow* actualmente ofrece tres vistas: *WorkSpace, Mobile y Classic*. En nuestro caso solo elegiremos *Classic* (el navegador web), y *Mobile* (para la aplicación móvil), ya que *WorkSpace* no será necesario.
Después habrá que definir el modelos de datos, que vendría a ser las tablas de la *scoped application*. Como las tablas de Proyectos, Usuarios y Compañías ya existen en la *scope* **Global** de la aplicación, solo es necesarios crear las tablas siguiente: Linea de gasto, Lista de gastos, Pagos y Tarjetas de Crédito. Dichas tablas no dependen ni extienden de otras tablas, así se se crearán a partir de cero. En nuestro caso la ventana nos ofrece crear nuevos campos para la tabla, así que creamos los campos solo definiendo su nombre, su tipo y su longitud máxima si se pide. Tras crear los campos nos pide le nombre de la tabla, y otras características como roles, extensibilidad... En nuestro caso solo ponemos nombre, auto-numérico de los registros si es necesario, y quitamos la creación de roles específicos para la tabla. Finalmente al crear las tablas necesarias se procede a definir el nombre, las tablas y roles que cada vista usará. Dado que el funcionamiento es casi el mismo en ambas vistas, las dos tendrán el mismo nombre, roles y tablas.
Tras esto, la creación guiada de una *scoped application* queda terminada. Lo único creado hasta ahora son las tablas con sus campos necesarios, sin embargo no existen ninguna lógica implementada hasta ahora. A continuación se procederá a crear los elementos de lógica que definirán el proceso de creación de gastos en la aplicación.

5.2. Lógica de las listas de gasto

La primera tabla a rellenar, será la de listas de gasto (**Expense List**). En este caso añadiremos una *Related List*, que mostrará una lista con las líneas de gasto asociadas a la lista.
Para configurar la tabla volvemos a ir a **System Definition** > **Tables**, y buscamos la tabla de listas de gasto por su *Label*. Una vez allí ya teniendo creados los campos del asistente guiado, procedemos a configurar los campos para ajustarlos a la lógica deseada:

Nombre del campo	Type	Mandatory	Read Only	Referenced Table
Number	String	No	Si	-
User	Reference	Si	No	*Users (sys_user)*
Total Cost	Decimal	No	Si	-
Start Date	Date	Si	No	-
End Date	Date	Si	No	-
Status	Choice	No	Si	-

Tabla 5.1: Configuración de los campos en la tabla **Expense List**

Una vez definidos los atributos, colocamos los campos mediante **Configure** > **Form Layout** para dejarlos colocados según el documento técnico (ver imagen A.7). Tras esto accedemos con el menú contextual del formulario a **Configure** > **Related Lists**, y a partir de las slushbuckets seleccionamos la de **Expense Line**. Eso nos deja visible la lista relacionada con las líneas de gasto que pertenecen a la lista actual.
Tras dejar hecho el formulario, procedemos a configurar la lógica de la tabla. Para esta tabla no hace falta ni *UI Actions* ni *Client Scripts*, pero si harán falta dos *Business Rules*, que se crearán accediendo desde el formulario o la lista con menú contextual a **Configure** > **Business Rules**. Las *Business Rules* son:

- Validate dates for the list: Aborta la creación o actualización de la lista de gasto si el nuevo periodo excluye una línea de gasto existente, o si la fecha de inicio es igual o posterior a la fecha de fin.
- Create payment for validated lists: Cuando una lista pasa al estado "Completed", se crean los pagos correspondientes a la lista actual.

Luego de configurar las reglas de negocio se crean las *UI Actions*. En nuestra aplicación casi todas se restringen a *UI Actions* de tipo *Form Button* (botón de formulario). Las *UI Actions* creadas son las siguientes:

- Approve All: Aprueba todas las líneas de gasto aun pendientes de la lista.
- Cancel All: Cancela todas las líneas de gasto aun pendientes de la lista.
- Create Payments for All: Completa las líneas de gasto planificadas aun pendientes de la lista.
- Reject All: Rechaza todas las líneas de gasto aun pendientes de la lista.
- Reject Validation: Devuelve la lista de gastos al estado "Active", revirtiendo las líneas de gasto rechazadas al estado "Pending".
- Send to Validation: Pone la lista de gastos en el estado "Pending". Esta *UI Action* habilita los botones que aprueban o rechazan las líneas de forma conjunta (Approve All, Reject All, Create Payments for All).

- Validate: Pasa la lista de gastos al estado "Completed". Muestra una advertencia si hay lineas de gasto en estado "Pending", e impide su aprobación en ese caso.

Una vez completado el formulario, solo queda colocar las líneas de la lista de la tabla. Para ello volvemos a la lista y vamos a **Configure** > **List Layout**. Dentro de los *slushbuckets* movemos los campos que nos interesa mostrar en la lista. Esta opción se puede alterar para un usuario individual mediante la opción "Personalize List", pudiendo modificar que campos muestra la lista solo para ese usuario (ver imagen A.8).

5.3. Lógica de las tarjetas de crédito

La siguiente tabla es la de las tarjetas de crédito. En este caso hay poco que hacer, ya que solo hay que comprobar la fecha de expiración de las tarjetas y ordenar la lista y el formulario (imágnenes A.9 y A.10).
Lo primero es colocar el formulario con **Configure** > **Form Layout**, luego la lista con **Configure** > **List Layout**, y luego configurar la lógica de los campos:

Nombre del campo	Type	Mandatory	Read Only	Referenced Table
CC Number	String	Si	No	-
Acc Number	String	Si	No	-
User	Reference	Si	No	*Users (sys_user)*
Expiration Date	Date	Si	No	-
Active	True/False	No	No	-

Tabla 5.2: Configuración de los campos en la tabla **Credit Card**

En esta tabla no existe ningún script de cliente, pero si una regla de negocio. Dicha regla simplemente comprueba que la fecha de una tarjeta no pueda ser anterior o igual a la fecha actual. Por otro lado existe una *Scheduled Job* que desactiva las tarjetas que tiene como fecha de caducidad el día actual, y se ejecuta cada día a medianoche.
Por último añadimos una lista relacionada con los pagos que se hayan realizado con la tarjeta. Para ello en el formulario vamos a **Configure** > **Related Lists** y añadimos la de *Payments*.

5.4. Lógica de los pagos

La tabla de pagos es la mas sencilla de las cuatro, ya que solo el sistema los crea, no necesitan ninguna modificación posterior, todos sus campos son de solo lectura, y no posee scripts sobre la propia tabla. Adicionalmente el campo *User* hará referencia a la tabla de usuarios, y el campo *Credit Card* hará referencia a la tabla de tarjetas de crédito (**Credit Card**). Lo único necesario es colocar el formulario y la lista con sus campos correspondientes. Como siempre nos iremos a **Configure** > **List Layout** en la lista, y a **Configure** > **Form Layout** en el formulario (imágnenes A.11 y A.12).

5.5. Lógica de las lineas de gasto

La tabla de lineas de gasto es la que mas campos tiene y la que mas elementos requiere para funcionar correctamente. Lo primero es definir que campos son obligatorios, cuales son solo en modo lectura, y adaptar el formulario según que valores se vayan introduciéndose. Adicionalmente hay que definir reglas de negocio que controlen como la información es enviada desde formulario, y que pasos deben hacerse al crear, actualizar o borrar lineas de gasto.
Para la gestión del formulario, en el navegador de aplicaciones vamos a **System Definition** > **Tables**, y buscamos el registro de la tabla de lineas de gasto que hemos creado en el paso anterior (*Expense Line*). A partir del formulario de definición de la propia tabla tendremos una lista de los campos creados, mas otros campos por defecto como *Created, Updated...* En cada campo de la lista existe una serie de checks que definen si el campo es obligatorio, si es solo lectura, o si se usa como *Display* para campos de tipo *Reference*. Para esta tabla nos interesa dejar marcado los siguientes valores para cada campo.

Nombre del campo	Type	Mandatory	Read Only	Referenced Table
Date	Date	Si	No	-
Currency	Reference	Si	No	*Currency (fx_currency)*
Local Cost	Decimal	Si	No	-
Cost	Decimal	No	Si	-
IVA Value	Decimal	Si	No	-
Total Cost	Decimal	No	Si	-
Distance	Decimal	No	No	-
Cost per KM	Decimal	No	No	-
Project	Reference	Si	No	*Project (pm_project)*
User	Reference	Si	No	*User (sys_user)*
Expense List	Reference	Si	No	*Expense List (x_ssssl_expenseapp_expenselist)*
Number	String	No	Si	-
Credit Card	Reference	Si	No	*Credit Card (x_ssssl_expenseapp_credit_card)*
Payment Method	Choice	Si	No	-
Type	Choice	Si	No	-
Description	String	Si	No	-
Additional comments	Journal	No	No	-
Process Type	Choice	Si	No	-
Manager	Reference	No	Si	*Users (sys_user)*
Status	Choice	No	Si	-
Payment	Reference	No	Si	-
Exchange Rate	String	No	No	-

Tabla 5.3: Configuración de los campos en la tabla **Expense Line**

Con esto definimos que campos serán obligatorios o de solo lectura por defecto, así como que valores

por defecto deben tener al crearse el registro, o que campos hacen referencia en el caso de los campos de tipo *Reference*.
Lo siguiente es definir elementos de configuración que manipulen el propio formulario, pero antes nos iremos al formulario de la tabla de **Expense Line** (New Record), y en el menú contextual del formulario nos iremos a **Configure** > **Form Layout**, ya partir de ahí podemos colocar los campos en el formulario tal y como se han definido en la documentación de diseño (ver imágen A.13). Una vez colocado los campos, procedemos a definir la lógica de cliente de este. Lo primero y mas sencillo son las **UI Policies**, que alterarán propiedades de uno o varios campos según que condiciones haya actualmente. Para definir lo que hacen cada uno se describirán que condiciones la ejecutan, y que campos alteran al ejecutarse. Las *UI Policies* creadas para la tabla son las siguientes:

- Hide "Distance", "Cost per KM" and "Transport Cost" if *Type* is not *Transport*: Si el tipo de gasto no es de *Transport*, oculta los campos *Distance*, *Cost per KM* y *Transport Cost*.
- Hide "Credit Card" field if Payment Method is Money: Oculta el campo *Credit Card* si el gasto es se ha abonado en metálico, es decir, *Payment Method* es igual a *Money*.
- Clear and make read only on "Expense List" if User is empty: Si se borra el contenido del campo *User*, se borra el contenido del campo **Expense List**, y se pasa a solo lectura.
- Change "Local Cost" to autofilled if "Type" is Transport: Si el gasto es de tipo *Transport*, se ocultan los campos *Local Cost* y *Currency*.
- Lock "Process Type"when its not Pending: Pone el campo *Process Type* en solo lectura si el gasto es inmediato y se encuentra aprobado.

Luego es necesario que se recalcule el coste total y la conversión de divisas al cambiar los valores de coste local o del IVA. Para ello se tiene que tratar desde el lado del cliente, y desde el lado del servidor para que funcione desde la aplicación móvil y otras reglas de negocio. Para ello sera necesario crear los siguientes scripts de cliente:

- Calculate Cost: Si uno de estos campos (Currency, Local Cost, Date) cambia o es borrado, actualiza los valores de *Currency* y *Cost*. Borra dichos campos si uno de los tres esta vacío. Esta función realiza conversión de divisas para obtener el resultado.
- Calculate Cost from Transport: Si se rellenan los campos *Distance* y *Cost Per KM*, se calcula el coste convertido a Euros. Si uno de los dos campos de vacía el campo *Local Cost* también lo hará.
- Calculate Total Cost: Calcula el valor del campo *Total Cost* si se rellenan los campos *Cost* e *IVA Value*. Lo borra si alguno de estos también se borra.
- Set Credit card properties with Method: Quita o pone la obligatoriedad y solo lectura del campo *Credit Card* dependiendo del método de pago establecido.

- Make Local Cost read only Transport Type: Hace el campo *Local Cost* de solo lectura para los gastos de transporte.
- Modify Credit Card when User changes: Borra y pone en modo lectura el campo *Credit Card* si no hay usuario en el campo *User*.
- Set Transport fields mutually: Hace que el campo *Transport Cost* se limpie y sea modo solo lectura si el campo *Distance* o *Cost per KM* están escritos, o los deja en modo solo lectura si *Transport Cost* está relleno. Hace accesible el campo *Currency* si solo *Transport Cost* es editable.

En cuanto a reglas de negocio *(Business Rules)*, necesarias para mantener la lógica de la aplicación fuera del formulario, se crearon las siguientes:

- Cancel Save if Date is in invalid range: Aborta el guardado de la línea de gasto si la fecha introducida no está dentro del periodo de la lista de gastos seleccionada en el formulario.
- Create payment for Credit Card Expense: Crea un pago para las líneas de gasto aprobadas que hayan sido pagadas con tarjeta de crédito.
- Create Payment for Planned Expense: Crea un pago para las líneas de gasto aprobadas que hayan sido pagadas siendo gastos planificados.
- Recalculate Total Cost in the List: Recalcula el valor de *Total Cost* de la lista de gastos a la que la línea pertenezca. Se activa al actualizar algún valor de coste, crear una línea o borrarla.
- Recalculate total costs: Calcula el valor de los campos *Cost* y *Total Cost* al cambiar el valor de *Local Cost, Currency, IVA Value*, o cuando *Cost* o *Total Cost* están vacíos. Esto recalcula los valores de solo lectura desde el lado del servidor, y es utilizado para recalcular fuera del formulario de escritorio.
- Delete Credit Card for Money Expenses: Borra el contenido del campo *Credit Card* si el método de pago es en efectivo.
- Set Manager when last modified: Rellena el campo interno *Manager* cuando un usuario mánager aprueba o rechaza la línea de gasto.

Para terminar con el formulario, se procede a crear las *UI Actions*. En este caso las *UI Actions* permiten al usuario manipular el registro de forma intuitiva. Las *UI Actions* para esta tabla tiene dos versiones: Una que aparece como botón o parte del menú contextual del formulario, y otra que aparece en el desplegable de la lista de lineas de gasto. La idea de hacer estas dos versiones se debe a como las *UI Actions* recogen los datos entre el formulario y la lista de la misma tabla. Las *UI actions* creadas son:

- Approve: Visible con la linea en estado "Pending" inmediatas. Pone la línea en estado "Approved". Es necesario escribir un comentario para usarla desde el formulario.

- Cancel: Visible con la línea sin completar ni aprobar. Pone la línea en estado "Cancelled". Es necesario escribir un comentario para usarla. Es necesario escribir un comentario para usarla desde el formulario.

- Create Payment: Visible para las líneas de gasto planificadas en estado "Pending". Pone la línea en estado "Completed". Es necesario escribir un comentario para usarla desde el formulario.

- Reject: Visible con la linea en estado "Pending" inmediatas. Pone la línea en estado "Rejected". Es necesario escribir un comentario para usarla. Es necesario escribir un comentario para usarla desde el formulario.

Al haber terminado de configurar el formulario, solo queda reordenar la lista de líneas de gasto. Para ello vamos a la lista y en el menú contextual de lista vamos a **Configure** > **List Layout**, y colocamos que campos vamos a mostrar y en que orden (imagen **??**).

5.5.1. Conversión de divisas con *fx_currency*

Cuando se cambia un valor numérico que afecte al campo Cost, el sistema deberá obtener dicho valor a partir del mecanismo que exista según los datos proporcionados. Si es basado en distancia el sistema interpretará el valor en Euros directamente, pero si se rellena con el coste directamente el sistema debe tener en cuenta la divisa con la que se rellena, y hacer la conversión adecuadamente.
En este caso, el sistema trae por defecto dos tablas de sistema con las que se pueden realizar conversiones de divisas. La primera tabla es la de divisas o tabla **Currency**, que almacena una lista de las divisas existentes con su símbolo, su nombre, y si están activas. Luego existe otra tabla que contiene los ratios de conversión, llamada **Exchange Rates**, que contiene una lista de registros con su divisa, ratio y fecha de creación. Dicha tabla de ratios se actualiza en base a la página web del Banco Central Europeo (BCE)[11].
El funcionamiento de la conversión de la aplicación es el siguiente: Cuando se altere un campo relacionado con el coste del gasto, ya sea *Local Cost*, *Currency* o incluso *Date*, el sistema con un script de cliente obtiene mediante una función en un *Script Include* el ratio de conversión de Euros a la moneda seleccionada en el campo *Currency* para la fecha descrita en el campo *Date*, redondeando la cifra a 4 decimales. Luego el sistema simplemente divide el coste con moneda local entre el ratio y lo añade al campo Cost, redondeando con dos cifras decimales.
Existe una *Business Rule* que calcula los costes en el lado de servidor, y realiza las mismas operaciones fuera del formulario, permitiendo calcular la conversión cuando se importar gastos en el importador de Excel o creando los gastos en la aplicación móvil.

5.6. Otras tablas

El resto de tablas relacionadas con la aplicación que no están metidas en la scope de la aplicación son usadas por otras aplicaciones, y al solo ser referenciadas en campos de la tablas del scope, no hace falta hacer ninguna modificación sobre ellas. Dichas tablas son las de proyectos (*pm_project*), las de

usuarios (*sys_user*) y las de clientes (*core_company*). Sin embargo para el importador se han creado dos tablas adicionales, una para almacenar los archivos y llevarlos al data source correspondiente, y otra para registrar los logs del importador.
La primera tabla auxiliar se llama *Expense Data Upload* (*x_ssssl_expenseapp_expense_data_upload*). Dicha tabla es donde los registros del *Record Producer* usados para insertar los archivos Excel se guardan los archivos Excel correspondientes, y se copian posteriormente al *Data Source*. La tabla no requiere configuración de lista, pero se crean unos pocos campos para mostrar la información de la importación al usuario. Dichos campos son:

- Number: Auto-numérico de *ServiceNow*.
- Date: Fecha que está en el Excel para el periodo y compararlo con las fechas de las líneas de gasto del Excel.
- User: Usuario escrito dentro del Excel. Se usa para comprobar si puede insertar el Excel o no en base a sus permisos, y para crear las líneas bajo su nombre si existe en la aplicación.
- *Record Producer*: Contiene el *Record Producer* que crea el registro. siempre sera *Import Expense Sheet*.
- State: Estado del registro con respecto a la importación. Por defecto empieza en In Process, pero al moverse el usuario y el periodo al registro se pone en "Pending", lo que pone en marcha una *Business Rule* que procede a la creación de registro en la *Import Set* Table. Al terminar se pone el estado a "Completed" o "Failed" según si ha habido errores de validación o no.
- Expense Line Issues: Contiene el *Import Set* creado para los registros, y se rellena cuando hay alguna linea en la tabla de logs del importador.

La segunda tabla se llama *Log Import* (*x_ssssl_expenseapp_log_import*), y simplemente contiene los logs del importador, sobre todo los relacionados con errores de validación de datos. Como la otra tabla al ser auxiliar no se configura ni la lista ni el formulario, pero se crean tres campos: Uno con el número del *Import Set*, otro con el mensaje de log a mostrar, y otro con la fila del Excel donde se encuentra el error (la fila representa una línea de gasto concreta).

5.7. Creación de aplicaciones en el navegador y sus módulos

La creación de aplicaciones y módulos para el explorador de aplicaciones de la instancia se basa en la creación de registros en dos tablas core de la instancia: Application Menu (*sys_app_application*), y Modules (*sys_app_module*). Lo primero es ir a **System Definition > Application Menus**, y crear dos registros. Uno se llamará *User Expenses* y el otro *Manager Expenses*. Ambos tendrán como requisito tener roles para poder visualizarlos: *User Expenses* requiere el rol *bill_user*, y *Manager Expenses* requiere *bill_manager*.
Para los módulos tenemos que irnos a **System Definition > Modules**, y crear tantos registro como módulos existan en los dos menús de aplicaciones con las siguientes especificaciones:

Name	Application Menu	Order	Role	Link Type	Filter
All Credit Cards	Manager Expenses	100	bill_manager	List of Records	N/A
All Payments	Manager Expenses	200	bill_manager	List of Records	N/A
Import Expense Sheet	Manager Expenses	300	bill_manager	URL	N/A
Companies	Manager Expenses	400	bill_manager	List of Records	N/A
Projects	Manager Expenses	500	bill_manager	List of Records	N/A
Expenses	Manager Expenses	600	bill_manager	Separator	N/A
Create New	Manager Expenses	700	bill_manager	New Record	N/A
All Expenses	Manager Expenses	800	bill_manager	List of Records	N/A
All Pending Expenses	Manager Expenses	900	bill_manager	List of Records	Status is Pending
All Approved Expenses	Manager Expenses	1000	bill_manager	List of Records	Status is Approved
All Paid Expenses	Manager Expenses	1100	bill_manager	List of Records	Status is Completed
All Expense Lists	Manager Expenses	1200	bill_manager	List of Records	N/A

Tabla 5.4: Tabla de configuración de módulos para Manager Expenses

My Credit Cards	Manager Expenses	100	bill_user	List of Records	User is(dynamic) Me
My Payments	Manager Expenses	150	bill_user	List of Records	User is(dynamic) Me
Import Expense Sheet	Manager Expenses	200	bill_user	URL	N/A
Expenses	Manager Expenses	300	bill_user	Separator	N/A
Create New	Manager Expenses	400	bill_user	New Record	N/A
My Expenses	Manager Expenses	500	bill_user	List of Records	User is(dynamic) Me
My Pending Expenses	Manager Expenses	600	bill_user	List of Records	User is(dynamic) Me and Status is Pending
My Approved Expenses	Manager Expenses	700	bill_user	List of Records	User is(dynamic) Me and Status is Approved
My Paid Expenses	Manager Expenses	800	bill_user	List of Records	User is(dynamic) Me and Status is Completed
My Expense Lists	Manager Expenses	900	bill_user	List of Records	User is(dynamic) Me

Tabla 5.5: Tabla de configuración de módulos para User Expenses

Los resultados de la definición de menús y módulos se muestran en imágenes en el apéndice A, siendo las imágenes A.15a y A.15b.

5.8. Script Includes

Aunque *ServiceNow* permite la creación de elementos sin necesidad de usar scripting para ello, normalmente es necesario tener que usar scripts para poder configurar nuestras aplicaciones. Uno de las cosas a tener en cuenta es la necesidad de poder tener acceso a funciones desde varios scripts de la aplicación, y para ello *ServiceNow* dispone de unos scripts de servidor llamados *Script Includes*.
Los *Script Include* son scripts de servidor que permiten escribir funciones sobre ellas, y poder luego invocar dichos *Script Includes* desde otros sitios como *Client Scripts, Business Rules, UI Macros* e incluso desde otros *Script Include*. Los *Script Includes* como los objetos en programación pueden

extender de otros *Script Includes*, heredando los constructores y funciones que estos tienen, pero implementando nuevas funciones o haciendo override a dichas funciones heredadas. Adicionalmente se pueden hacer que los *Scripts Includes* sean llamables desde scripts de cliente (*Client Callable*), lo que permite a los scripts de lado de cliente llamar a dichas funciones mediante el uso de *GlideAjax*.
El objetivo de los *Script Includes* en la aplicación es precisamente mover toda la lógica de servidor a estos scripts, de forma que se puedan reutilizar funciones en diferentes sitios, así como facilitar la lectura de los elementos de customización creados.
El primer *Script Include* creado se llama **ExpenseUtils**. Dicho script almacena funciones usadas por *Business Rules* y otros elementos de la aplicación. Las funciones creadas son:

- selectableUsersForExpenseLine: Devuelve la query con usuarios seleccionables para la creación o modificación de líneas de gasto en base al rol del usuario actual.
- getExchangeRate: Devuelve el ratio de conversión a Euros de un divisa en base a la divisa y al fecha pasadas.
- selectableListsForExpenseLine: Devuelve la query con las listas seleccionables en base al ID de un usuario concreto de la aplicación.
- canCancelAll: Comprueba si un usuario tiene acceso a la *UI Action* "Cancel All".
- canSendToValidation: Comprueba que un usuario pueda enviar a validación un lista suya.
- canMakeApprovalonAll: Comprueba si un usuario tiene acceso a las *UI Action* "Approve All" y "Reject All".
- canMakePaymentAll: Comprueba si un usuario tiene acceso a la *UI Action* "Create Payment for All".
- canValidate: Comprueba si un usuario tiene acceso a las *UI Action* "Validate" y "Reject Validation".
- canMakeApproval: Comprueba si un usuario tiene acceso a las *UI Action* "Approve" y "Reject".
- canCancel: Comprueba si un usuario tiene acceso a las *UI Action* "Cancel".
- onlyRelatedUserOrAllManagers: Comprueba si un usuario concreto es un mánager, o el usuario es el actual.
- hasPendingExpenses: Devuelve true o false si hay lineas de gasto pendientes de aprobar o no en una lista de gastos.
- getUserLists: Devuelve las listas de gasto de un usuario en base a su user ID.
- createListsForNewPeriod: Crea las nuevas listas de gasto para los usuarios con rol activo en el nuevo periodo. Usado en la *Scheduled Job* cada primer día del mes.
- needsCalculateTotalCost: Comprueba si una lista necesita recalcular su coste total o no.

- checkPeriodConflicts: Comprueba si el cambio de un periodo definido por dos fechas de una lista excluye una línea de gasto existente.
- getActiveCreditCards: Devuelve las tarjetas de crédito activas para un usuario concreto.
- checkUserAndList: Comprueba y devuelve true o false si para una lista concreta el usuario pertenece a ella.
- getUserWithRoles: Devuelve a los usuarios con roles de la aplicación.

El segundo script se llama *ExpenseUtilsAjax*, y contiene funciones usadas en Client Script mediante GlideAjax. Las funciones que contiene son las siguientes:

- dateIsInPeriodRange: Comprueba que una fecha esta dentro del periodo comprendido entre dos fechas, en base a una fecha y una lista de gastos.
- getExchangeRate: Devuelve el ratio de conversión a Euros en base a la divisa y la fecha proporcionadas.
- hasPendingExpenses: Devuelve true o false si hay lineas de gasto pendientes de aprobar o no en una lista de gastos. Esta función simplemente llama a la función del mismo nombre del *Script Include ExpenseUtils*.

El tercer script se llama *ImportExpenseUtils*, y se usa para almacenar funciones usadas en la importación de lineas de gasto a través de Excel. Las funciones se describen mas adelante en el importador de gastos de la aplicación.
Debido a que no es posible usar *SncTriggerSynchronizer.executeNow()* dentro de un *Script Include* de la *scope*, se ha creado una copia del mismo *Script Include* en la *scope* **Global**, que contiene solo la función que llama a *SncTriggerSynchronizer.executeNow()*.

5.9. Importador de gastos

La inserción de registros en *ServiceNow* es posible gracias al uso de los *Data Sources*. Los *Data Sources* son elementos de la instancia que toman como *Attachments* archivos en varios formatos (Excel, CSV, JDBC) y crean registros temporales en una tabla denominada *Import Set Table*. Luego con otros elementos denominados *Transform Maps* es posible redirigir de forma controlada la copia de datos de los registros de la *Import Set Table* a los registros de la tabla donde se quieren crear o actualizar los registros.
El mayor inconveniente a la hora de realizar el importador es que los *Data Source* solo son accesibles a los usuarios administradores, por lo que es necesario crear una interfaz que permita a un usuario no administrador insertar su archivo Excel, y que este llegue al *Data Source* correspondiente para que el propio sistema ejecute un *Scheduled Import Set* que permita al *Data Source* correspondiente extraer los datos del Excel a la *Import Set Table*. El proceso para obtener este resultado involucra varios elementos de configuración y customización que permitan usar el *Data Source* de manera indirecta

a los usuarios de la aplicación, que valide que los datos sean correctos en el Excel, que compruebe permisos sobre el usuario del Excel y el que haga la importación, y que la creación de los registros de líneas de gasto se realice correctamente.
Lo primero son las tablas de *Expense Data Upload* y *Log Import*, que se pueden crear en **System Definition > Tables** con los campos mencionados anteriormente.
La *Import Set Table* puede crearse fácilmente mediante la opción **System Import Sets > Load Data**, subiendo el archivo con el formato correcto y diciéndole que se cree una nueva *Import Set Table*. De esa forma vendrá configurada correctamente con todos los campos necesarios, sin embargo se deberán añadir dos campos adicionales: User y Period para el usuario y el periodo obtenidos a través del *Record Producer*.
Luego se crea el *Record Producer* para que el usuario pueda insertar sus gastos. Para ello vamos a **Service Catalog > Record Producers** y lo creamos con el botón "New". En *Table* seleccionamos *Expense Data Upload*, el nombre será *Import Expense Sheet*, y en el script del *Record Producer* hacemos que coja el registro del archivo Excel, generado en la tabla *sys_attachment*, y mediante la *API GlideExcelParser* obtenemos el usuario y el periodo de las celdas concretas y los rellenamos sobre los campos correspondientes de la tabla *Expense Data Upload*.
Una vez creado el *Record Producer*, es necesario usar una *Business Rule* que colocará el Attachment sobre la *Data Source* y la ejecutará, obteniendo los registros para la *Import Set Table*. La *Business Rule* se creará sobre la tabla *Expense Data Upload*, con nombre "Exp Import - Trigger Scheduled Import", de tipo after (después de que el registro pase al estado In Process). Dicha **Business Rule** ejecutará varias funciones del *Script Include ImportExpenseUtils*, tanto de la versión del scope como de la versión global. Dichas funciones son:

- copyAttachments: Copia el Attachment del registro creado en *Expense Data Upload* a la *Data Source* correspondiente (*Import Expense Sheet*). Para ello *ServiceNow* ofrece la *API GlideSysAttachment* que permite hacer la copia de forma automática de un registro a otro.
- setRunAs: Modifica el valor del campo *Run As* del *Scheduled Import Set* con el del usuario actual.
- triggerScheduledImport: Ejecuta la función *SncTriggerSynchronizer.executeNow()* sobre el Scheduled Import Set. Esta llamada se realiza sobre la versión en el *scope* **Global**.

Lo siguiente será crear dicho Scheduled *Expense Data Upload*, lo que permitirá ejecutar el *Data Source*. Para ello vamos a System Import Sets > Scheduled Imports y le damos a "New". El *Scheduled Import Set* de nombre "Exp Import - Trigger Scheduled Import" contendrá el *Data Source* a usar, una condición definida en una función del *Script Include*, y llamará a otras dos funciones al acabar. La función de la condición con nombre checkPendingDU comprobará que el registro de la tabla *Expense Data Upload* esté en estado In Process, y si lo cumple se ejecuta el *Data Source* asociado. Al acabar llama a otras dos funciones: deleteAttachments que borra los Attachments del *Data Source* para evitar que se lean otra vez, y updateDataUpload que pone el registro del a tabla *Expense Data Upload* a "Completed" o "Failed" dependiendo se si hay logs de error o no. Una vez creados los elementos que permiten al *Data Source* obtener los registros temporales del *Import Set Table*, es necesario crear y

configurar un Transform Map para validar los datos y transformarlos correctamente a la tabla de líneas de gasto.
En la Related List de *Transform Maps* dentro del *Data Source*, se puede crear uno con el botón "New". Como nombre ponemos "Expense Transform Map", marcamos como True lo de *Run Business Rules*, de forma que las BR se ejecuten al transformar los registros de la *Import Set Table* a la tabla de líneas de gasto, y finalmente definimos las tablas origen (**Import Set Expense**) y destino (**Expense Line**).
Una vez creado el *Transform Map* definimos los *Field Maps*, que determinan que campo de la tabla origen se empareja con que campo de la tabla destino. Debido a que todos los campos de la tabla del *Import Set* son de tipo String, los valores numéricos no pueden pasarse tal y como están ahora. Es necesario poner un script dentro de los *Map Fields* de los campos numéricos para ajustar el formato de los números correctamente, eso es, asegurándose de que el punto flotante sea una coma, y no un punto.
Otros campos como *Credit Card* o *Project* necesitan también el uso de scripts en el *Field Map*, ya que no están guardados los sys_id en la *Import Set Table*, pero si los nombres, por lo que es necesario hacer un GlideRecord para obtener el sys_id correcto para la tarjeta de crédito y el proyecto. Otros *Field Map* como el de la Description son solo texto y se pueden mapear directamente.
Una vez definidos los *Field Map* es necesario hacer el paso de validación, y otros pasos posteriores al mapeo de datos entre las dos tablas. Lo primero es crear un *Transform Script*, que se puede ejecutar en cualquier momento del mapeo para realizar actividades concretas, como parar el mapeo, ignorar una línea bajo ciertos requisitos, o simplemente poner logs.
En la Related List de *Transform Maps* (junto a la de *Field Maps*), creamos dos *Transform Scripts*. El primero de tipo onStart (antes de comenzar todo el mapeo), simplemente copia las valores de User y Period del *Expense Data Upload* a los campos creados para los registros de la *Import Set Table*. Luego se crea otro *Transform Script* de tipo onStart que procederá a la validación de los datos en la *Import Set Table*.
Dicho script borra el attachment del *Data Source*, ya que no nos hace falta ahora tenerlo ahí, y luego procede a la fase de validación. La validación consiste en llamar a una función del *Script Include ImportExpenseUtils* llamada validate. Dicha función devuelve true o false según si los datos son correctos o no, y ese valor booleano se asigna a la variable ignore del *Transform Script*, que determina si los *Map Fields* se ejecutarán o no. Los criterios usados para determinar si los datos de un Excel son válidos son:

- Que existan una fecha y un nombre de usuarios en la primera fila, junto a las etiquetas correspondientes. El usuario debe existir en la aplicación y la fecha debe ser válida.
- Cada fecha de la línea de gasto debe ser válida, y debe tener el mismo mes y año que la fecha de periodo de la parte superior.
- Cada línea de gasto debe tener el campo Proyecto relleno, y debe existir en la aplicación.
- El campo de Descripción del Excel para todas las líneas no puede estar vacío.
- El tipo de cada línea debe estar relleno con uno de estos cuatro valores: T, C, H, O.

- Los valores de Moneda, Importe y Conversión deben o estar relleno los tres o estar vacíos. No pueden solo estar rellenos algunos y otros no. Adicionalmente los valores de Importe y Conversión deben ser valores numéricos, y el valor en Moneda debe ser una divisa existente.

- Los siguientes valores deben ser numéricos o vacíos:

  - num KM
  - Eur/KM
  - KM
  - Transporte
  - Hotel
  - Comida
  - Otros
  - IVA
  - TOTAL
  - Tarjeta Empresa
  - Efectivo

  Adicionalmente, solo uno de estos cuatro (Hotel, Comida, Transporte, Otros) puede tener un valor distinto de cero, que debe coincidir con el tipo descrito anteriormente.

- Los valores de Tarjeta Empresa y Efectivo son mutuamente exclusivos, es decir, uno debe tener el mismo valor que TOTAL, y el otro debe ser cero o vacío.

Una vez definido el *Transform Maps* solo queda definir la URL del *Record Producer* sobre el módulo "Import Expense Sheet" para que se muestre al pulsar sobre él. Para cada módulo en los dos menús de aplicación creados, se copia y pega la URL del Record Producer en el campo URL, sobre la pestaña "Link Type".

5.10. Dashboard, reportes y filtros interactivos

Para la página principal, es necesario crear los *Dashboards*, que contendrán los reportes y sus filtros interactivos. Para los *Dashboards* vamos a **Self-Service** > **Dashboards** y le damos a "New". Dentro del formulario solo tendremos que rellenar los campos de nombre y roles requeridos. En nuestro caso *User Dashboard* tendrá de rol requerido el de usuario de gastos (*bill_user*), y el de *Manager Dashboard* el de mánager de gastos (*bill_manager*).
La creación de un *Dashboard* por cada rol nos permitirá que en caso de necesitar mas reportes específicos para un rol este solo se adjunte al *Dashboard* para ese usuario, aunque hay que configurar los dos aparte.

Para los reportes tendremos que ir al módulo **Reports** > **View\Run**, y crear los reportes siguiendo las siguientes características:

Report Name	Source Type	Table	Type	Configurations
Total cost of all expenses per project	Table	Expense Line	Bar	Group By: Project, Aggregation: Sum (Total Cost), Show Other: checked, Default Filter: N/A.
Total cost of all expenses over time	Table	Expense Line	Bar	Group By: None, Trend By: Date per Date, Aggregation: Sum (Total Cost), Percentage Calculation: Use Aggregation, Default Filter: N/A.
Total cost of all expenses of an user in a month	Table	Expense Line	Single Score	Aggregation: Sum (Total Cost), Default Filter: "Date" on "This Month".
Total cost of user expenses per project	Table	Expense Line	Bar	Group By: Project, Aggregation: Sum (Total Cost), Show Other: checked, Default Filter: "User" is(dynamic) "Me".
Total cost of user expenses over time	Table	Expense Line	Bar	Group By: None, Trend By: Date per Date, Aggregation: Sum (Total Cost), Percentage Calculation: Use Aggregation, Default Filter: "User" is(dynamic) "Me".
Total cost of user expenses of an user in a month	Table	Expense Line	Single Score	Aggregation: Sum (Total Cost), Default Filter: "User" is(dynamic) "Me" and "Date" on "This Month".

Tabla 5.6: Configuración de reportes en *ServiceNow*

Luego quedarían de añadir los filtros interactivos. Estos han sido mas complicados debido a la necesidad de autorizarme por parte de los empleados de la empresa, ya que los filtros interactivos no son accesibles de base y necesitan de una licencia de *Performance Analytics*[14].
Los filtros interactivos se encuentran en **Reports** > **Interactive Filters**, donde se pueden crear bajo permiso del equipo encargado de la instancia. Para filtrar por estado, usuario y proyectos en las líneas de gasto, será necesario crear un filtro interactivo por cada uno, mas crear un *Dynamic Block*

para el filtro de rango por fechas.
El primer filtro es el de proyecto. Este filtro será de tipo *Reference*, con un seleccionador de un solo input, que cogerá la tabla de proyecto, y tendrá una *Interactive Filter Reference* apuntando al campo *Project* de la tabla de líneas de gasto.
El segundo filtro es el de usuario. Este filtro será de tipo *Reference*, con un seleccionador de un solo input, que cogerá la tabla de usuarios, y tendrá una *Interactive Filter Reference* apuntando al campo *User* de la tabla de líneas de gasto.
El tercer filtro es el de estado de línea de gasto. Este filtro será de tipo *Choice List*, con un seleccionador de un solo input, que apuntará al campo *Status* de la tabla de líneas de gasto.
Para el último filtro no existe nada de caja, por lo que tendremos que diseñar un filtro customizado, llamada comúnmente un **Custom Interactive Filter**[9]. En este caso simplemente vamos a **Content Management** > **Blocks** > **Dynamic** y creamos el contenido en base a la documentación de *ServiceNow*. En nuestro caso crearemos dos campo de input de tipo "Date", que al cambiar su contenido llamen a una función dentro del XML. Dicha función creará una query de tablas en base a los valores de los campos, definirá los valores en *SNC.canvas.interactiveFilters.setDefaultValue()*, y llamará a la función predefinida *publishFilter()* para filtrar los reportes en el *Dashboards*, o a *removeFilter()* si ninguno de los campos está relleno, y por tanto eliminando las condiciones de filtrado de los reportes.
Una vez creado todo nos vamos a la pantalla principal y la configuramos para que muestre *Dashboards*, y colocamos los elementos de forma visualmente cómoda.

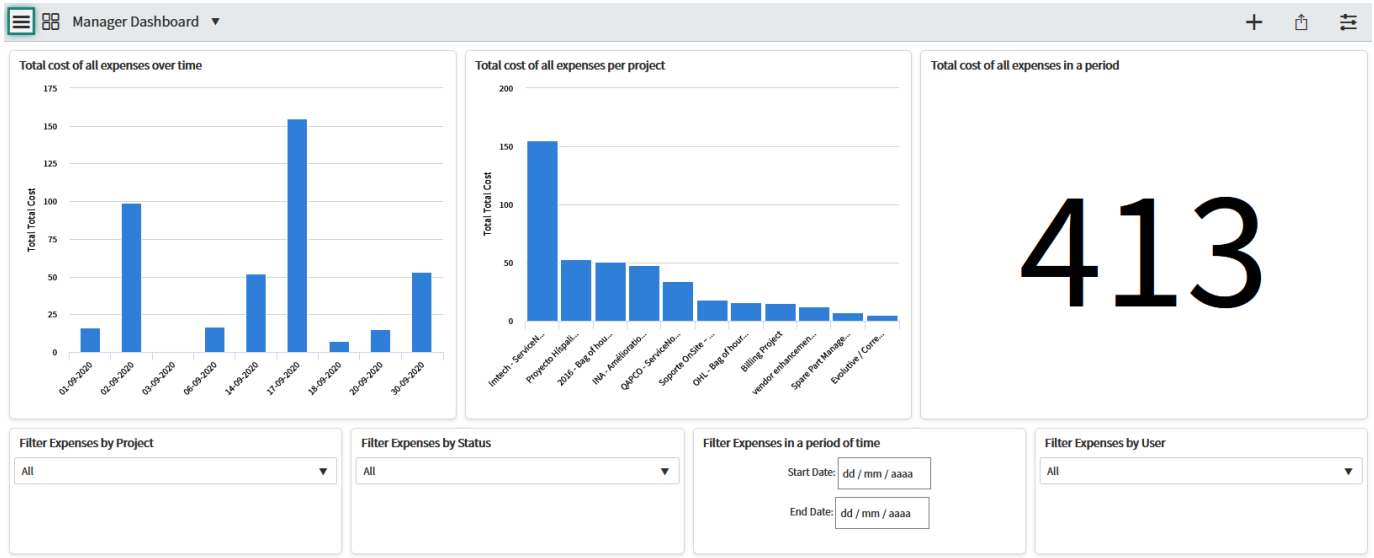


Figura 5.1: Visualización del *Dashboard* de inicio

Por último, abrimos la pestaña de configuración de cada *Dashboard* (signo + arriba a la derecha), y en cada reporte de gráfico de barras le damos al botón de configuración (engranaje que sale en la esquina superior derecha del reporte), y marcamos las opciones *"Follow Interactive Filter"* y *"Show when following filter"*, de esta forma los reportes mostrarán los datos de las líneas de gasto filtradas por los reportes.

5.11. Aplicación móvil

Para el propósito de este trabajo, se va a utilizar la aplicación *Now Mobile* para el desarrollo de la aplicación móvil, ya que es la que se ajusta al tipo de usuario que se menciona en la sección 4.14 de la memoria. Lo necesario para está aplicación se centra en poder visualizar las listas y líneas de gasto, así como poder crear o modificar nuevas líneas de gasto. Lo primero es ir a **System Applications > Studio**, el cual nos lleva a la IDE interna de *ServiceNow*, útil para el desarrollo de aplicaciones móviles que no sean la versión clásica para móviles de *ServiceNow*, ya que esta obsoleta. A partir de ahí se nos muestra una lista con las Scoped Applications existentes en la instancia, simplemente seleccionamos al nuestra (Expense Application).

Lo primero es definir un *Expense Application*, que será un apartado de Now Mobile donde se mostrarán los elementos de nuestra aplicación. Para ello en Studio nos vamos a **Mobile Studio > Applet Launcher** y creamos uno de tipo Generic, llamado *Expense Application*. Dentro del Launcher creamos una UI Section de nombre Expense List, que contendrá dos Applets, una para las listas de gasto, y otra para las líneas de gasto. A continuación vamos a **Mobile Studio > Applets** y creamos los dos Applets que añadiremos a la UI Section.

El primer Applet se llamará Expense Lists, el cual mostrará una lista con las listas de gasto que el usuario puede visualizar (afectado por las ACL). Dicha *Applet* definirá los registros de la listas de gasto mediante un Data Item, creado en **Mobile Studio > Data Items**. Luego entre las varias opciones existentes se colocarán los datos a visualizar en la pestañas List Screen y Form Screen del Applet. En los dos se visualizarán el número, las fechas de inicio y fin, el coste total de las líneas que la contienen, el usuario al que pertenecen y el estado actual.

El otro Applet llamado *Expense Lines*, mostrará la información de las líneas de gasto (limitada también por las ACL de la aplicación), con su Data Item propio, y mostrando en la List Screen los siguientes campos: Number, Project, Description, Total Cost, Type y Payment.

En la vista de formulario se mostrarán todos los campos excepto Manager y los campos predefinidos del sistema. Luego es necesario definir que se puedan añadir Attachments desde la aplicación móvil. Para ello en la Applet vamos a **Form Screen > Body**, y marcamos el check de "Show Attachment List"[13], de forma que el usuario pueda adjuntar archivos, o incluso sacar fotos con el móvil y adjuntarlas directamente en la línea de gasto a crear o modificar.

Como es necesario poder crear y modificar líneas de gasto, es necesario definir dos *Functions*, una para crear líneas y otro para modificarlas. Las Functions se pueden crear en **Mobile Studio > Functions > Actions**, y también es necesario crear la *Action Item* correspondiente para cada *Function* creada, ya que es necesario definir los campos en la vista móvil, y que se copien los datos de los campos al registro en la aplicación. El primer *Function*, *New*, será en Contexto Global, y usará la *Action Item New* creada al mismo tiempo, además se marcará la opción "Allow Images Upload" para permitir que el usuario pueda subir las imágenes en los registros de la aplicación. Luego por cada campo que se vaya a rellenar en la aplicación móvil debe crearse los siguientes elementos: *UI Parameter* en el *Function*, *Item Parameter* en el *Action Item* y *Action Parameters Mapping* en el *Function*.

Item Parameter Name	Item Parameter Type	UI Parameter Input Type	UI Parameter Order
Date	Date	Text	100
User	String	Text	200
Expense List	String	List (*Expense List*)	300
Project	String	List (*Project*)	400
Payment Method	String	List (*Expense Line* → *Payment Method*)	500
Credit Card	String	List (*Credit Card*)	600
Local Cost	Decimal	Text	700
Currency	String	List (*Currency*)	800
Description	String	Text	900
IVA Value	Decimal	Text	1000
Type	String	List (*Expense Line* → *Type*)	1100
Distance	Decimal	Text	1200
Cost per KM	Decimal	Text	1300
Transport Cost	Decimal	Text	1400

Tabla 5.7: Lista de mapeos de campos realizados entre *Now Mobile* y la instancia

Como la aplicación calculará automáticamente los valores de *Cost* y *Total Cost*, no es necesario añadirlos en la aplicación móvil, pero si será necesario replicar la lógica de cliente en la aplicación móvil, y para ello es necesario crear varias *UI Policies* de *Mobile*, denominadas **Screen UI Policies**. Dichas *UI Policies* de definen en **Mobile Studio** > **UI Policies**, y funcionan de forma similar a las *UI Policies* en la vista de escritorio (la única diferencia es que no permiten manipular la escritura de los campos). Las *Screen UI* son las siguientes:

- Make transport fields mandatory: Muestra y hace obligatorios los campos *Distance*, *Cost per KM* y *Transport Cost* cuando el tipo es Transporte.

- Exclude Distance and Cost per KM: Oculta los campos *Distance* y *Cost per KM* si el campo *Transport Cost* está relleno.

- Hide Currency for Distance Expenses: Muestra el campo *Currency* solo si el tipo de la línea no es Transporte, o siendo el tipo de línea Transporte no se hayan rellenado los campos *Distance* o *Cost per KM*.

- Hide Expense List if User is empty: Oculta el campo *Expense List* si el campo de usuario está vacío.
- Exclude Transport Cost: Oculta el campo *Transport Cost* si están rellenos el campo *Distance* o el campo *Cost per KM*.
- Remove Local Cost for Transport Expenses: Oculta el campo *Local Cost* si el tipo de la línea es Transporte.
- Distance and Cost per KM needs to be filled: Asegura que los campos de distancia y coste por KM sean obligatorios si no están rellenos. Evita que un usuario pueda enviar cambios con uno de los campos vacío.
- Hide Credit Card field for Money Expenses: Oculta el campo de tarjeta de crédito si el método de pago es en efectivo.

Con estas *Screen UI Policies* conseguimos que los gastos que no sean de tipo Transporte tengan visible el campo de coste local y el campo de divisa, permitiendo añadir costes convencionales. Por otro lado también permite añadir gastos de tipo Transporte que usen el método de conversión normal pero con el campo Transport Cost, o crear líneas de gasto con el coste basado en distancias. La *Function Edit* se comporta de forma bastante similar a la de *New*, pero se añaden una *UI Parameter* y un *Item Paramenter* para el *Sys ID* de la línea de gasto, además el contexto del *Function* será Record, y no Global. Para mostrar las Functions, es necesario añadir la de *New* en la Applet, en **List Screen** > **Functions**, y la de *Edit* en **Form Screen** > **Overall Settings** > **Functions**. Tanto los botones *New* como *Edit* aparecerán como Top Menu Functions (arriba a la derecha de la pantalla móvil, donde los tres puntos).

Capítulo 6

Plan de pruebas

Una vez implementado todo, es cuestión de probar que lo que hay implementado funciona correctamente, y se adapta a los requisitos del cliente. Para ello simplemente habrá que reproducir los pasos necesarios para obtener las líneas de gastos, realizar la petición de validación de listas, y ver que se crean los pagos correspondientes. Adicionalmente será necesario simular otras operaciones relacionadas con otros elementos de la aplicación como tarjetas de crédito.

6.1. Datos de ejemplo

Para probar las siguientes funcionalidades, se han añadido los siguientes elementos de prueba:

Nombre	Rol
Expense Manager Test A	bill_manager
Expense Manager Test B	bill_manager
Expense User Test A	bill_user
Expense User Test B	bill_user

Tabla 6.1: Usuarios de ejemplo

Number	User	Start Date	End Date
EXP0001783	Expense Manager Test A	01/09/2020	30/09/2020
EXP0001784	Expense Manager Test B	01/09/2020	30/09/2020
EXP0001791	Expense User Test A	01/08/2020	30/08/2020
EXP0001782	Expense User Test A	01/09/2020	30/09/2020
EXP0001786	Expense User Test B	01/09/2020	30/09/2020

Tabla 6.2: Listas de gasto de ejemplo

User	CC Number	Acc Number	Expiration Date
Expense Manager Test A	4929870583835950	4929870583835950	10/10/2020
Expense Manager Test B	5437544715716006	5437544715716006	07/11/2020
Expense User Test A	4716453556813010	4716453556813010	10/10/2020
Expense User Test B	5246337721649787	5246337721649787	07/11/2020

Tabla 6.3: Tarjetas de credito de ejemplo

6.2. Creación de una lista de gastos

Si no existe una lista de gastos para el usuario, es necesario crear una. Lo primero es ir a la lista de listas de gastos, ubicada en **Manager Expenses** > **All Expense Lists**, y le damos al botón "New". Ello nos llevará al formulario de listas de gasto, en donde tendremos que rellenar el usuario al que pertenece, y las fechas que comprenden el periodo de tiempo para esta lista. Hay que tener en cuenta que si la lista tiene las fechas mal colocadas (*Start Date* es igual o posterior a *End Date*), el sistema abortará la creación y avisará al usuario de que el rango de fechas esta mal puesto. Una vez creada la lista esta será modificable por los mánagers, pero deberá tenerse en cuenta que si el nuevo rango de fechas excluye una línea de gasto existente (su fecha queda fuera del periodo), el sistema impedirá la actualización de esta.

Como ejemplo concreto, el mánager puede ir la la lista, ir al botón de "New", y rellenar los siguientes valores:

- User: Expense User Test A
- Start Date: 01/09/2020
- End Date: 30/09/2020

Al guardar, el sistema muestra la lista creada con los valores introducidos, mas los valores de *Number*, *Total Cost* y *Status* por defecto. Por otro lado el sistema abortará la creación y guardado de una lista con estos valores:

- User: Expense User Test A
- Start Date: 01/09/2020
- End Date: 30/08/2020

Ya que la fecha de final esta antes que la fecha de inicio. Adicionalmente si se intenta modificar la lista con estos valores descritos abajo, con una línea de gasto creada en la fecha 12/09/2020, el sistema también abortará el guardado:

- User: Expense User Test A
- Start Date: 01/09/2020
- End Date: 15/08/2020

6.3. Creación de una línea de gasto manual

Tras tener una lista creada para el usuario, este puede crear lineas desde la opción **User Expenses > Expenses > Create New**. Dicho módulo abrirá un formulario para líneas de gasto, que deberá rellenar con los campos obligatorios. El usuario deberá tener en cuenta las siguientes condiciones al crear una linea de gasto:

- Al elegir de tipo el valor Transporte, el sistema mostrará los campos *Distance* y *Cost per KM*, que indicarán el coste en base a una distancia recorrida. Sin embargo también permitirá rellenar el campo *Transport Cost* para añadir un coste que no sea basado en distancia. El sistema impide rellenar una lista con ambas opciones.
- Si el tipo no es Transporte, el usuario podrá añadir el coste con la divisa local del gasto y la divisa utilizada. El sistema calculará automáticamente el valor del gasto en Euros. Si no hay cambio de divisa el sistema usará la divisa de Euros, con ratio de conversión 1.
- Si el gasto se paga con tarjeta de crédito, se debe añadir una tarjeta activa que pertenezca al usuario.
- El tipo de proceso determina como se aprobará la línea de gasto.
- Si quien rellena la línea es un usuario normal, solo podrá crearlas a sí mismo, pero los mánagers y administradores podrán crear líneas a cualquier usuario con roles en la aplicación.
- El sistema impide crear líneas de gasto con una fecha fuera del periodo de la lista a la que se va a asignar.

Una vez creada, estará en estado "Pending", pero no se podrá realizar el proceso de aprobación de líneas hasta que la lista no sea enviada para su validación.

Un ejemplo concreto, sería que el usuario *Expense User Test A* decidiera crear una línea con estos datos

- User: Expense User Test A
- Expense List: EXP0001791
- Date: 13-08-2020
- Process Type: Paid
- Type: Alojamiento
- Project: Expense Project
- Payment Method: Money
- Description: Habitación Berlín

- Local Cost: 107,56
- IVA Value: 3,55
- Currency: EUR

Al ir introduciendo los datos relativos a la fecha, el coste local y la divisa, el sistema calcula el coste total. Com ya está en euros el ratio de conversión es 1 y simplemente añaden el el campo *Cost* el valor de *Local Cost*, y añadiendo en el coste total la suma del coste en euros con el coste del IVA. En este caso *Total Cost* quedaría igual a 107,56 + 3,55 = 111,11 €.
El sistema siempre tendrá en cuenta tanto que los campos necesarios estén rellenos, como que la fecha de la línea de gasto sea válida y esté dentro del periodo de la lista de gastos, como el formato de los datos en cada campo. Si el usuario intenta crear una línea con estos gastos:

- User: Expense User Test A
- Expense List: EXP0001791
- Date: 31-11-2020
- Process Type: Paid
- Type: Alojamiento
- Project: Expense Project
- Payment Method: Money
- Description: Habitación Berlín
- Local Cost: 107,56
- IVA Value: 3,55
- Currency: EUR

El sistema abortará el guardado, y avisará al usuario de que la fecha no está comprendida en el periodo de la línea de gasto, adicionalmente si intenta meter caracteres alfabéticos en un campo numérico, como por ejemplo el campo *Local Cost*, el sistema impide que se registre el resultado en el campo y lo reemplaza automáticamente por un cero.

6.4. Creación de una línea de gasto mediante aplicación móvil

Aparte de crear lineas de gasto en escritorio, la aplicación permite crearlas desde la aplicación móvil. El procedimiento es similar al de escritorio. El usuario deberá ir a *Expense Application* > **Expense Lines**, y en el botón superior derecho saldrá un desplegable con la opción "New". Aparecerá un pequeño formulario con los campos necesarios para rellenar una línea de gasto. Dicho formulario

también mostrara ciertos campos según las características de este (ver sección 5.11). Adicionalmente se pueden adjuntar archivos abajo del formulario, ya sea añadiendo archivos del teléfono, o incluso adjuntando una foto que puede tomarse inmediatamente.
Una vez creada la línea el sistema calculará los campos restante (conversión de divisa, cálculo de coste por distancias), y creará la línea en el sistema.

La forma de introducir es similar al caso anterior, el usuario *Expense User Test B* accede a la aplicación *Now Mobile*, se loguea y con el botón "New" mencionado anteriormente accede al formulario y pone los siguientes valores:

- User: Expense User Test B
- Expense List: EXP0001786
- Date: 06-09-2020
- Process Type: Paid
- Type: Other
- Project: Expense Project
- Payment Method: Credit Card
- Credit Card: 5246337721649787
- Description: Compra cliente americano
- Local Cost: 21,5
- IVA Value: 0,28
- Currency: USD

En este caso se ha usado dólares americanos, asi que el ratio de conversión a Euros para el dia 6 de septiembre es igual a 0,844452, por lo que el valor de *Cost* equivale a 21,5 * 0,844452 = 18,16 €. Luego en el campo *Total Cost* quedaría en total 18,16 + 0,28 = 18,44 €.

6.5. Importación de líneas de gasto

Una tercera opción para añadir líneas de gasto a la aplicación es mediante el importador. La opción se encuentra en **User Expenses** > **Import Expense Sheet**, y dentro de *Content Frame* el usuario puede descargar la plantilla del Excel, insertar los datos de los gastos y adjuntarlo en la opción de *Attachments*. Si existe un solo archivo y el formato del Excel y las líneas es correcto, el sistema mostrará que la importación habrá sido exitosa, creando las líneas de gasto en la correspondiente lista.
En caso de que haya cualquier fallo, ya sea por una lista inexistente o completa, un usuario inexistente

o un campo vacío o mal escrito. El importador avisará mediante logs de que ha habido fallo y no creará ninguna línea de gasto. En la lista relacionada del formulario saldrá que fallos ha habido y en que fila del Excel habrá ocurrido.

El usuario *Expense Manager* Test B accede a la ventana de importación, e introduce el Excel con los siguientes datos: Al guardar el archivo, se crearán los siguientes registros en la lista de gastos: Sin embargo el importador cancelará la importación de lineas de gasto si alguno de los datos está mal formateado. Si por ejemplo faltara de rellenar el usuario, o una fecha tiene el formato incorrecto (por ejemplo 31/2/2020), el sistema logueará el fallo en la lista y cancelará la improtación.

Nombre:	Expense User Test A													Periodo	01/08/2020		
Fecha	Descripción	Código Proyecto	Tipo	Moneda	Importe	**Conversion**	num KM	Eur/KM	KM	Transporte	Hotel	Comidas	Resto	IVA	TOTAL	Efectivo	Tarjeta empresa
01/08/2020	Gasto habitación	Expense Project	H					0	0,00	0,00	15,45	0,00		0,28	15,73	15,73	
06/08/2020	Viaje AP6	Expense Project	T				20	0,17	3,40					1,90	5,30		5,30
30/08/2020	Ticket Metro	Expense Project	T				0	0		53,20				0,00	53,20	53,20	
14/08/2020	Cena empresarial	Expense Project	C	USD	21,5	18,17 €						18,17		0,00	18,17		18,17
17/08/2020	Propinas	Expense Project	O										151,32	3,27	154,59		154,59
14/08/2020	Gasto hostal	Expense Project	H	CHF	36,55	33,89 €					33,90			0,11	34,00	34,00	
02/08/2020	Taxi	Expense Project	T				217	0,22	47,74					0,00	47,74	47,74	

Figura 6.1: Excel para el importador

Number ▲	Description	Total Cost
EXP0006901	Gasto habitación	15,73
EXP0006902	Viaje AP6	5,3
EXP0006903	Ticket Metro	53,2
EXP0006904	Cena empresarial	18,17
EXP0006905	Propinas	154,59
EXP0006906	Gasto hostal	34
EXP0006907	Taxi	47,74

Figura 6.2: Lineas de gasto creadas a través del importador

6.6. Enviar la lista para validación

Una vez creadas las líneas necesarias, el usuario puede ir a la lista de gastos y pulsar en el botón *Send to Validation*. El usuario aún podrá adjuntar gastos mientras la lista no esté completada.
En este ejemplo el usuario *Expense User Test A* puede validar su lista de gastos EXP0001791.

6.7. Proceso de aprobación de listas de gastos

Los mánagers de la aplicación podrán validar (aprobar) listas de gasto cuando estás hayan sido enviadas por el usuario. Para ello tienen que ir a Manager Expenses > All Expense Lists, y abrir la lista de gasto a validar. Dentro de la lista pueden ver todas las líneas de gasto asociadas, y pueden abrirlas una a una para aprobarlas o rechazarlas, e incluso la propia lista relacionada ofrece la opción de seleccionar las líneas para aprobarlas o rechazarlas. Adicionalmente existen botones que deciden sobre todas la líneas de gasto aun pendientes (Approve All, Reject All, Create Payments for All, Cancel All). Si la lista es rechazada (*Reject Validation*), la lista vuelve al estado "Active", poniendo las líneas de gasto rechazadas en "Pending". Por el contrario si se valida la lista esta se pone en estado "Completed", y se crean los pagos de la forma especificada anteriormente.

En el ejemplo de la lista EXP0001791, el usuario *Expense Manager Test A* decide aprobar todas las líneas de gasto, excepto las líneas EXP0006905 y EXP0006903. Luego a continuación decide dejarlas rechazadas y pulsa sobre el botón Validate. A partir de ahí la lista esta completada y no se le pueden asignar nas líneas de gasto. Por otro lado se habrán creado los pagos correspondientes a la lista: Uno para los gastos en metálico con valor total de 208,58 €, y dos pagos para los gastos con tarjeta de crédito, con valores 5,3 y 18,17 € respectivamente.

6.8. Completar una línea de gasto planificada

Si aun no se ha validado la lista, el mánager puede completar una línea de gasto planificada dentro de la propia línea. La lista de líneas de gasto pendientes se encuentra en **Manager Expenses > All Pending Expenses**, y buscar una con proceso Planning. Dentro tendrá dos botones: El de *Cancel* y el de *Create Payment*. Si pulsa sobre *Create Payment*, la línea se completará independientemente de la lista de gasto, y se generará el pago correspondiente a la línea. Para probar esta debemos crear una línea de gasto planificada. Para ello el usuario *Expense User Test A* crea una línea con las siguientes características:

- User: Expense User Test A
- Expense List: EXP0001782
- Date: 20-09-2020
- Process Type: Planning
- Type: Comida
- Project: Expense Project
- Payment Method: Money
- Description: Planificado
- Local Cost: 15,02
- IVA Value: 0
- Currency: EUR

6.9. Cancelar una línea de gasto

Si una línea resulta injustificable o ha sido creada por error, se puede cancelar para que no se tenga en cuenta a la hora de crear pagos. El usuario o mánager puede abrir una línea de gasto sin aprobar o completar y pulsar sobre el botón *Cancel*. Dicha opción junto a las otras de aprobación también son seleccionables en una lista desplegable de la vista de lista (*List View*).

6.10. Creación de una tarjeta de crédito

Un mánager puede crear tarjetas de crédito que hayan sido entregada a usuario para pagar sus gastos con ella. Para ello debe ir a **Manager Expenses > All Credit Cards**, y pulsar en el botón "New".

En el formulario debe rellenar los campo obligatorios, teniendo en cuenta que la tarjeta no debe estar caducada, y por lo tanto la fecha de caducidad no debe ser igual o posterior al día de hoy.

Como ejemplo, el usuario Expense Manager Test B puede ir al módulo de tarjetas de crédito, y mediante el botón "New" muestra el formulario para crear una tarjeta, y la guarda con los siguientes datos´:

- CC Number: 5437544715716006
- Acc Number: 5437544715716006
- User: Expense Manager Test B
- Expiration Date: 07-11-2020

Otro ejemplo sería que el mismo usuario fuera a crear una tarjeta de crédito con estos datos:

- CC Number: 5246337721649787
- Acc Number: 5246337721649787
- User: Expense User Test B
- Expiration Date: 07-08-2020

En este caso el sistema detecta que la tarjeta de crédito ya está caducada, y aborta el guardado de esta, mostrando un mensaje que indica que la fecha de caducidad debe ser posterior al día actual.

Capítulo 7

Conclusiones

El desarrollo de esta aplicación ha sido útil no solo para afianzar conocimientos sobre las diferentes herramientas de *ServiceNow*, e incluso sobre las utilidades que ofrecen las aplicaciones móviles de *ServiceNow*. Adicionalmente ha sido útil ya que me ha brindado la oportunidad de poder trabajar el desarrollo de requisitos y la elaboración de documentos de diseño tal y como se haría en la empresa en la que trabajo actualmente. Además dicho trabajo me ha dado experiencia con el trato directo con los clientes que han reclamado la creación de esta aplicación, anotando los requisitos de esta, resolviendo dudas que surgían, ofreciendo alternativas a problemas que surgían y ofrecer una aplicación que no solo funciones correctamente sino que esté hecha de forma óptima y sea ampliable en un futuro. Después del Trabajo de Fin de Grado, espero poder participar en el diseño de aplicaciones para otros proyectos de la compañía en la que trabajo, así como profundizar mi conocimiento sobre la herramienta de *ServiceNow*, lo cual me permitirá realizar tareas de implementación mas complejas.

Bibliografía

[1] Charles G. Cobb. *Making Sense of Agile Project Management: Balancing Control and Agility*. 1st Edition. Hoboken, New Jersey: John Wiley & Sons, 2011.

[2] Joey F. George Jeffrey A. Hoffer y Joseph S. Valacich. *Modern Systems Analysis and Design, Global Edition*. 7th Edition. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2013.

[3] ServiceNow. *Add roles to groups in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/roles/task/t_AssignRoleToGroup.html (visitado 01-07-2020).

[4] ServiceNow. *Add roles to users in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/users-and-groups/task/t_AssignARoleToAUser.html (visitado 01-07-2020).

[5] ServiceNow. *Attachment functions*. URL: https://docs.servicenow.com/bundle/orlando-servicenow-platform/page/script/useful-scripts/reference/r_UsefulAttachmentScripts.html (visitado 30-07-2020).

[6] ServiceNow. *Base System Roles*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/roles/reference/r_BaseSystemRoles.html (visitado 01-07-2020).

[7] ServiceNow. *Create roles in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/roles/task/t_CreateARole.html (visitado 01-07-2020).

[8] ServiceNow. *Create users in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/users-and-groups/task/t_CreateAUser.html (visitado 01-07-2020).

[9] ServiceNow. *Custom interactive filters*. URL: https://docs.servicenow.com/bundle/orlando-performance-analytics-and-reporting/page/use/dashboards/concept/c_CustomPublishers.html (visitado 14-08-2020).

[10] ServiceNow. *Data Sources*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/import-sets/concept/c_DataSources.html (visitado 30-07-2020).

[11] ServiceNow. *Default currency conversions*. URL: https://docs.servicenow.com/bundle/paris-platform-administration/page/administer/currency/concept/currency-conversions.html (visitado 14-08-2020).

[12] ServiceNow. *Export List in ServiceNow*. URL: https://docs.servicenow.com/bundle/madrid-platform-administration/page/administer/exporting-data/concept/c_ExportListData.html (visitado 20-06-2020).

[13] ServiceNow. *Manage attachments on details screens*. URL: https://docs.servicenow.com/bundle/paris-mobile/page/administer/tablet-mobile-ui/concept/sg-mobile-attachments.html (visitado 19-08-2020).

[14] ServiceNow. *Performance Analytics*. URL: https://docs.servicenow.com/bundle/orlando-performance-analytics-and-reporting/page/use/performance-analytics/reference/r_PALandingPage.html (visitado 14-08-2020).

[15] ServiceNow. *PPM Standard Plugin*. URL: https://docs.servicenow.com/bundle/orlando-it-business-management/page/product/project-portfolio-suite-with-financials/concept/c_ProjectPortfolioSuiteWithFinancials.html (visitado 28-06-2020).

[16] ServiceNow. *Reporting in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-performance-analytics-and-reporting/page/use/reporting/reference/reporting-landing-page.html (visitado 04-08-2020).

[17] ServiceNow. *Scheduled Jobs in ServiceNow*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/administer/reference-pages/concept/c_ScheduledJobs.html (visitado 30-07-2020).

[18] ServiceNow. *ServiceNow Fundamentals*. 1st Edition. Santa Clara, CA: ServiceNow, 2019.

[19] ServiceNow. *ServiceNow Mobile*. URL: https://docs.servicenow.com/bundle/paris-mobile/page/administer/tablet-mobile-ui/concept/mobile-config-navigation.html (visitado 02-09-2020).

[20] ServiceNow. *Transform Maps*. URL: https://docs.servicenow.com/bundle/orlando-platform-administration/page/script/server-scripting/concept/c_CreatingNewTransformMaps.html (visitado 30-07-2020).

Apéndice A

Imágenes de la implementación

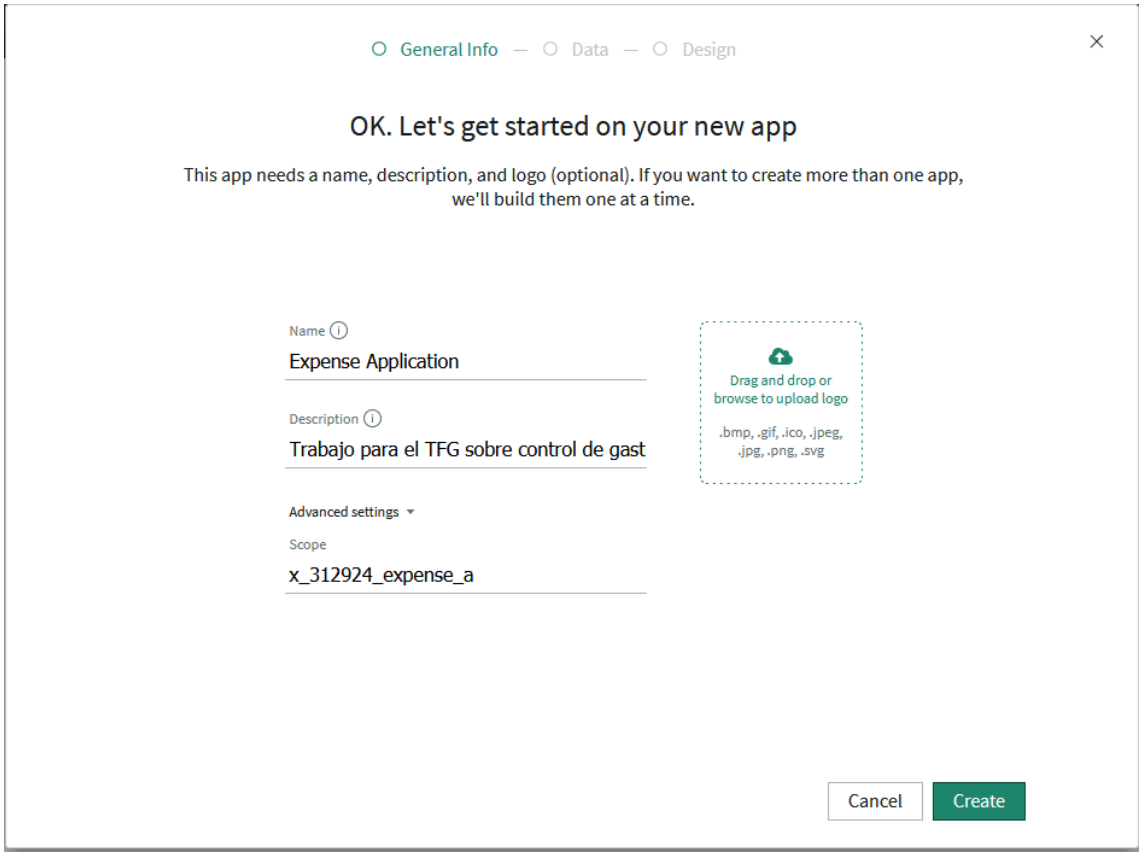


Figura A.1: Ventana inicial de creación de *scoped applications*

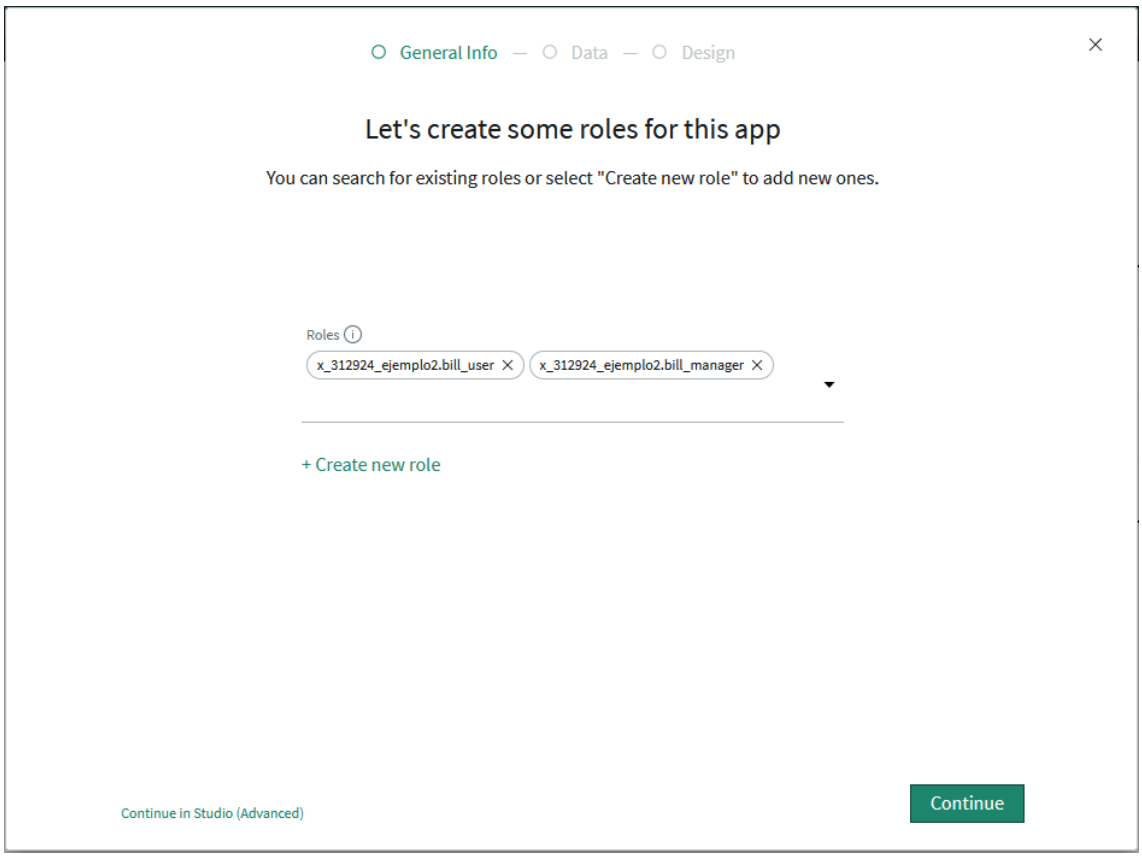


Figura A.2: Ventana para la introducción de roles en la aplicación

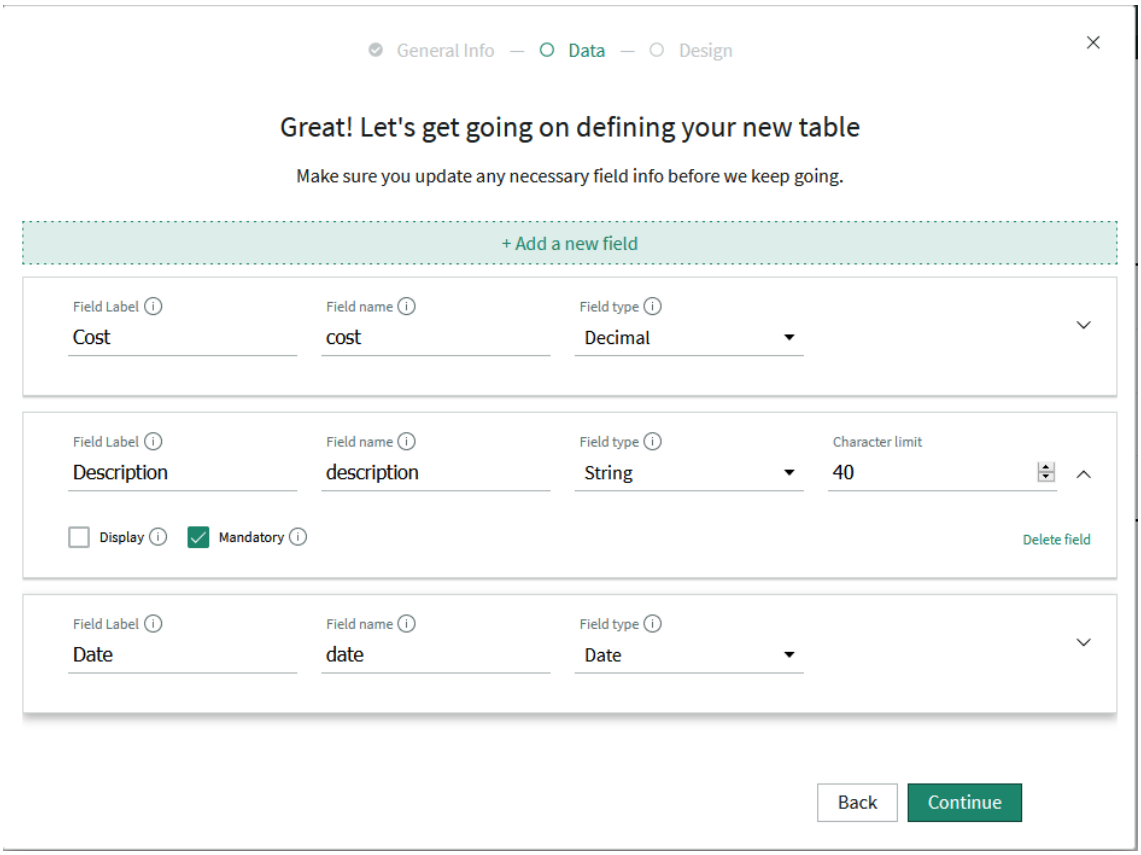


Figura A.3: Ventana para la creación de tablas y campos en la aplicación

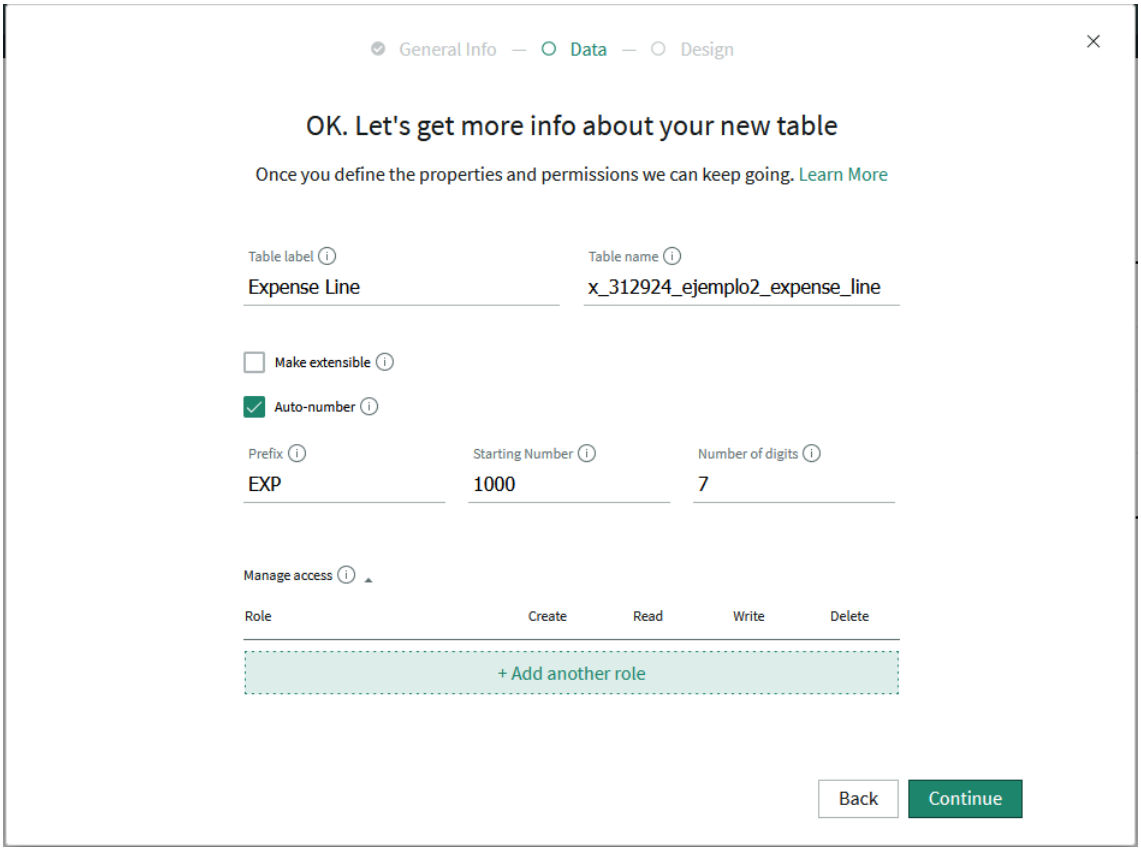


Figura A.4: Ventana para el nombre de la tabla y otras propiedades

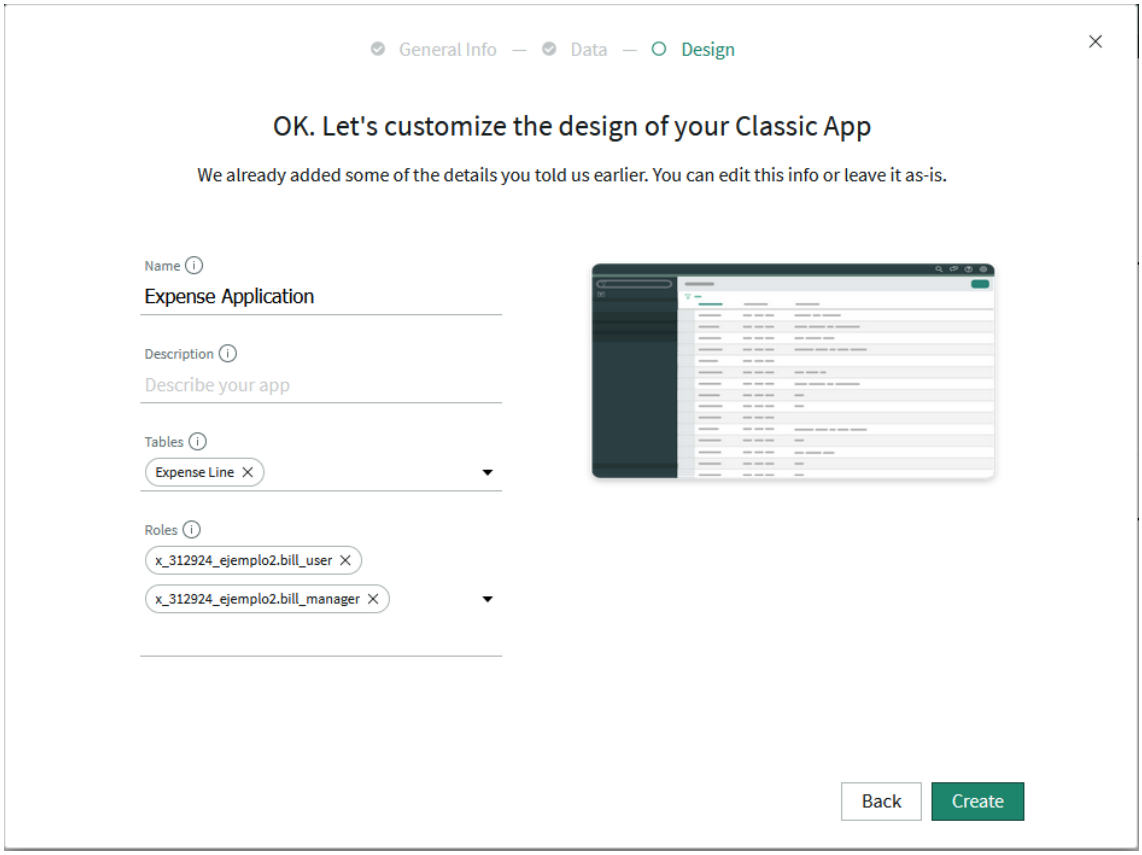


Figura A.5: Ventana para definir nombre y elementos de la vista de escritorio

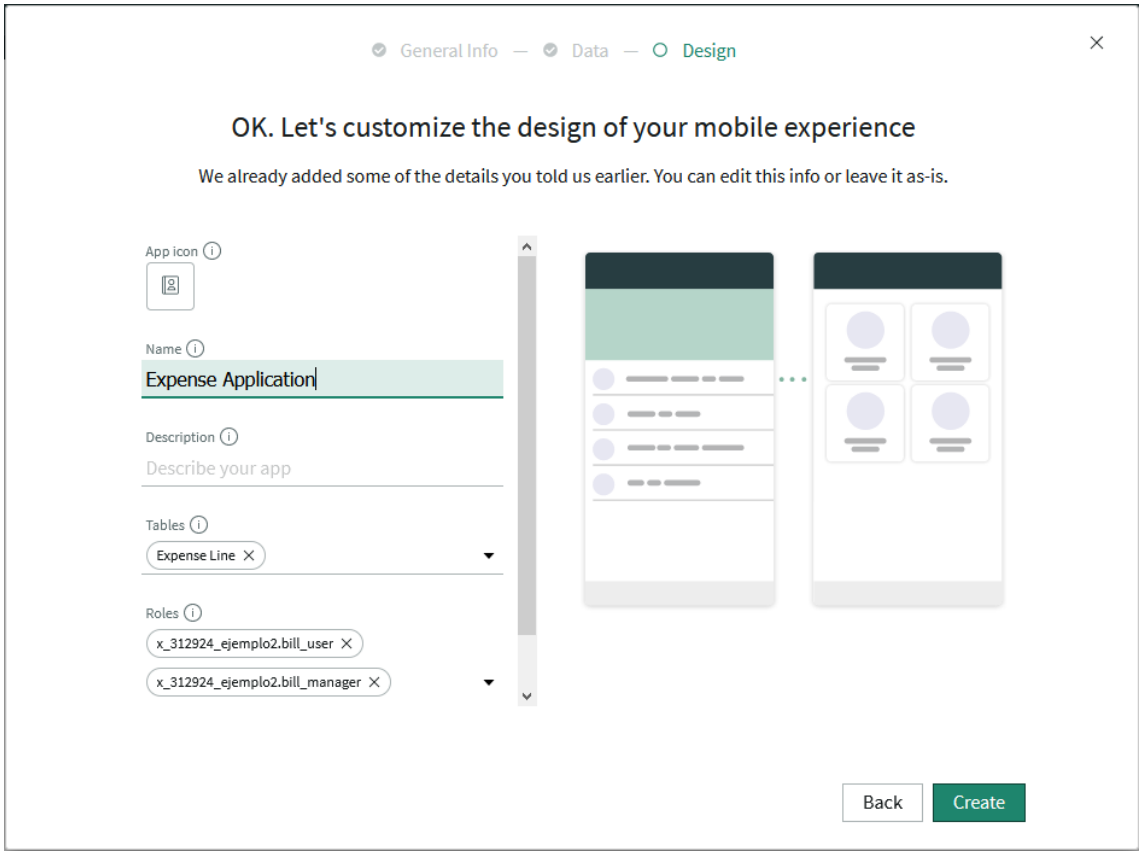


Figura A.6: Ventana para definir nombre y elementos de la vista móvil

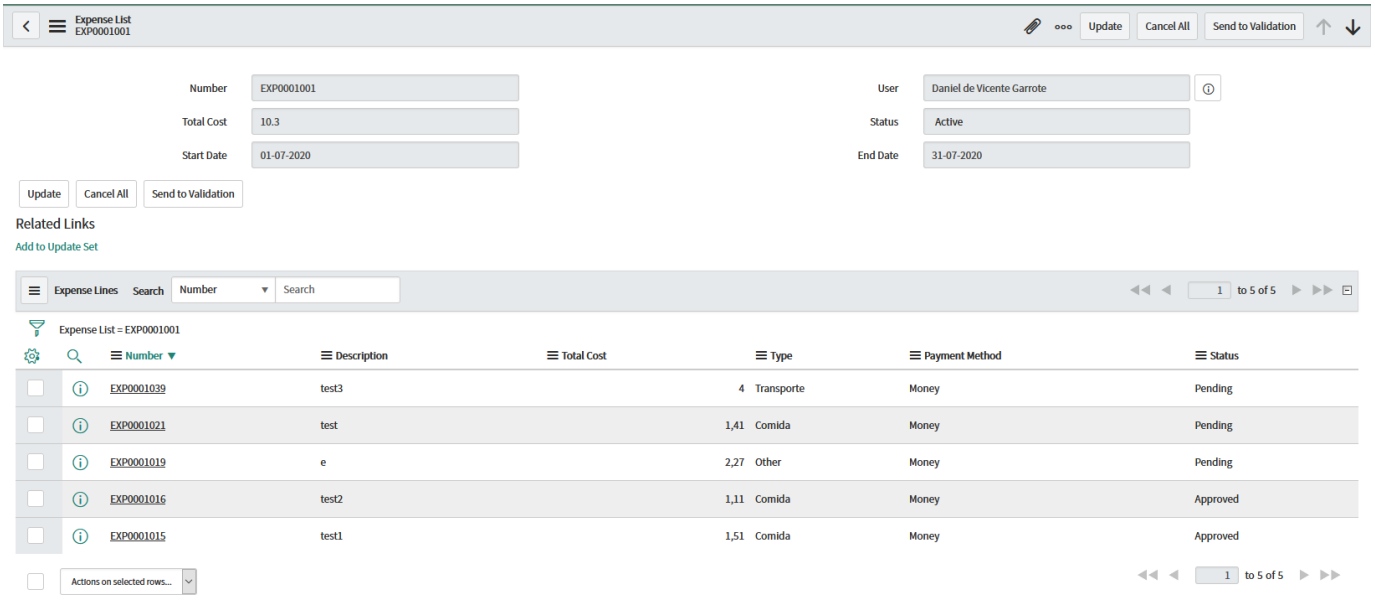


Figura A.7: Formulario de listas de gasto por defecto

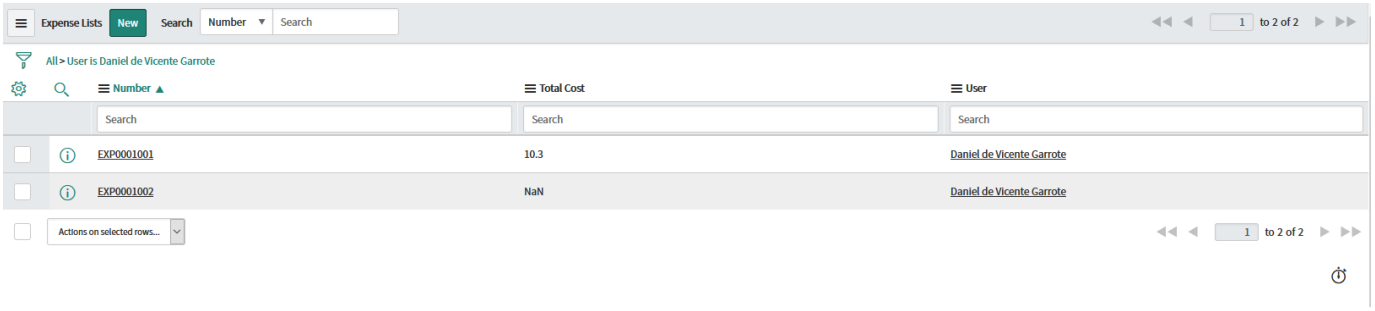


Figura A.8: Visualización de la lista de listas de gasto por defecto

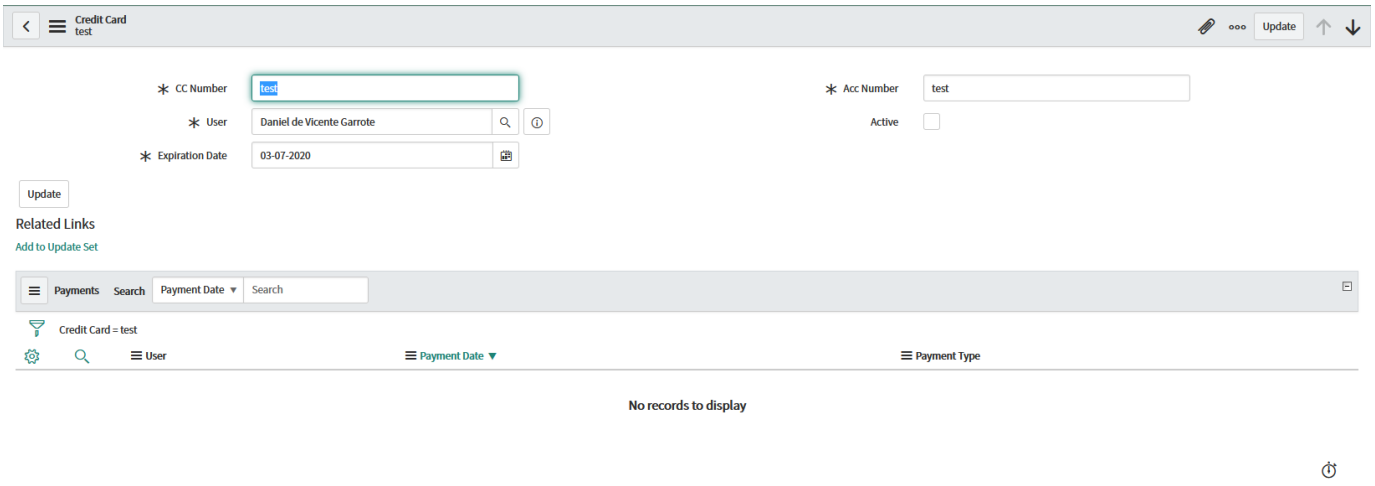


Figura A.9: Formulario de tarjetas de crédito por defecto

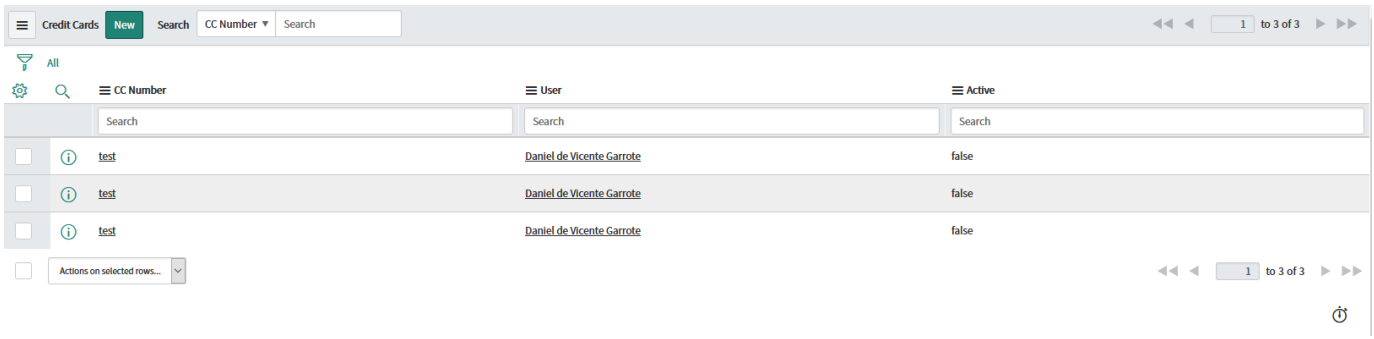


Figura A.10: Visualización de la lista de tarjetas de crédito por defecto

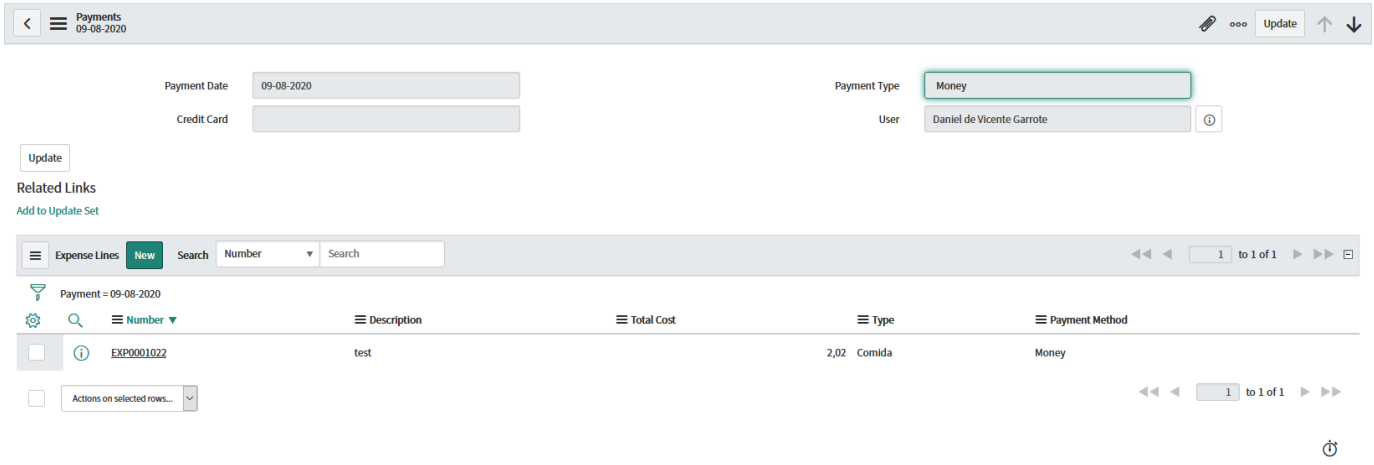


Figura A.11: Formulario de pagos por defecto

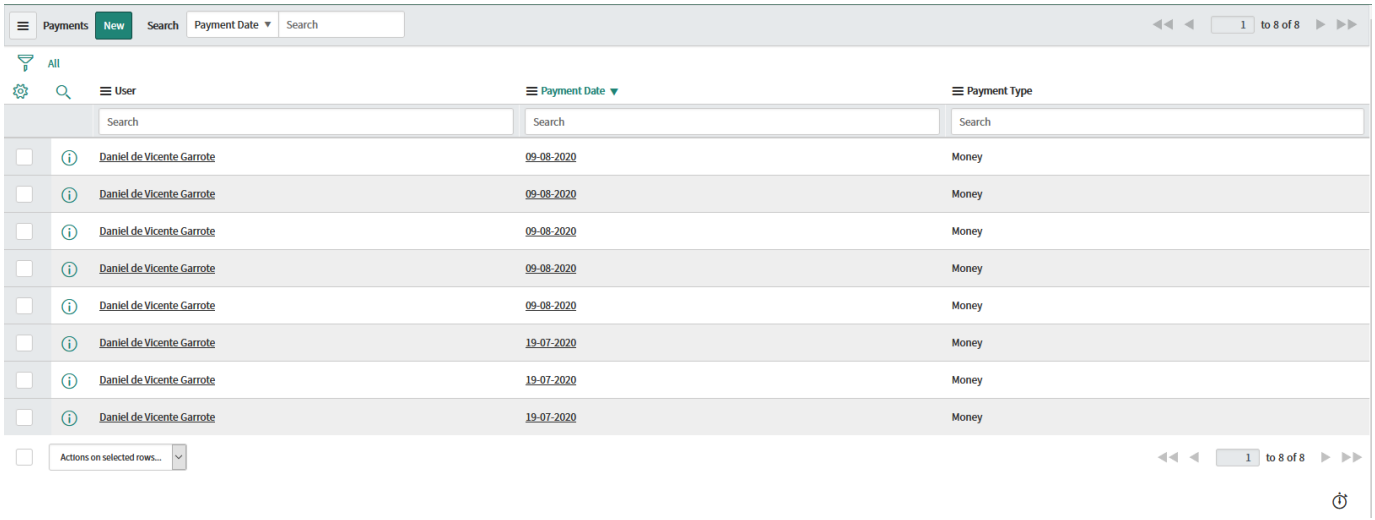


Figura A.12: Visualización de la lista de pagos por defecto

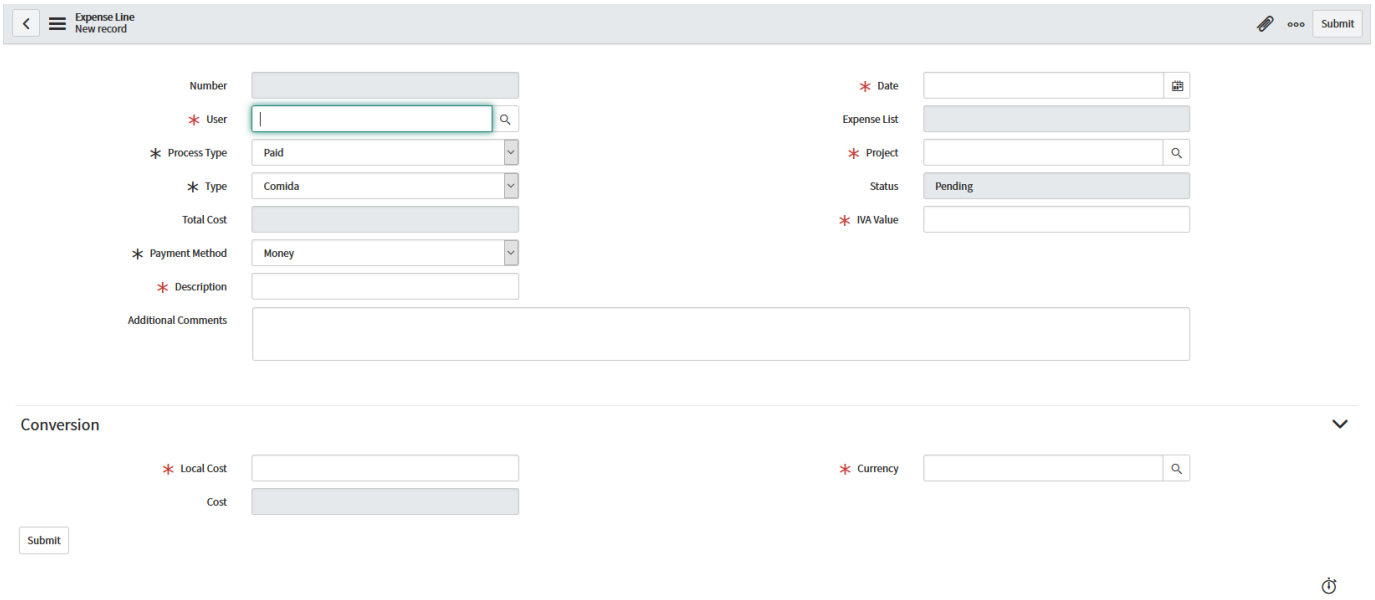


Figura A.13: Formulario de lineas de gasto por defecto

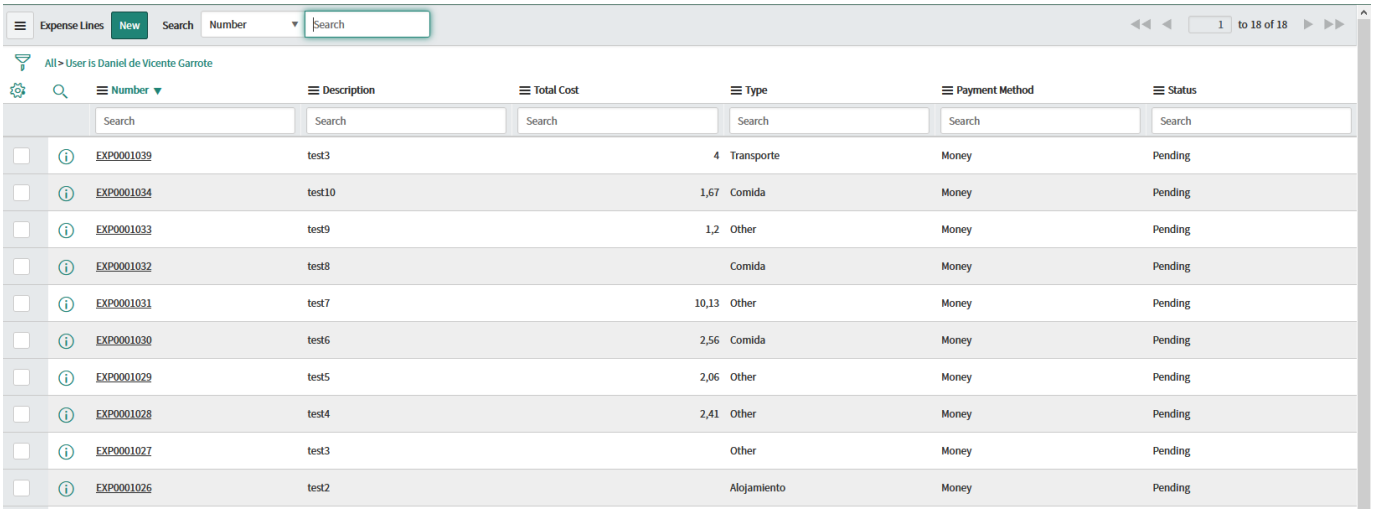
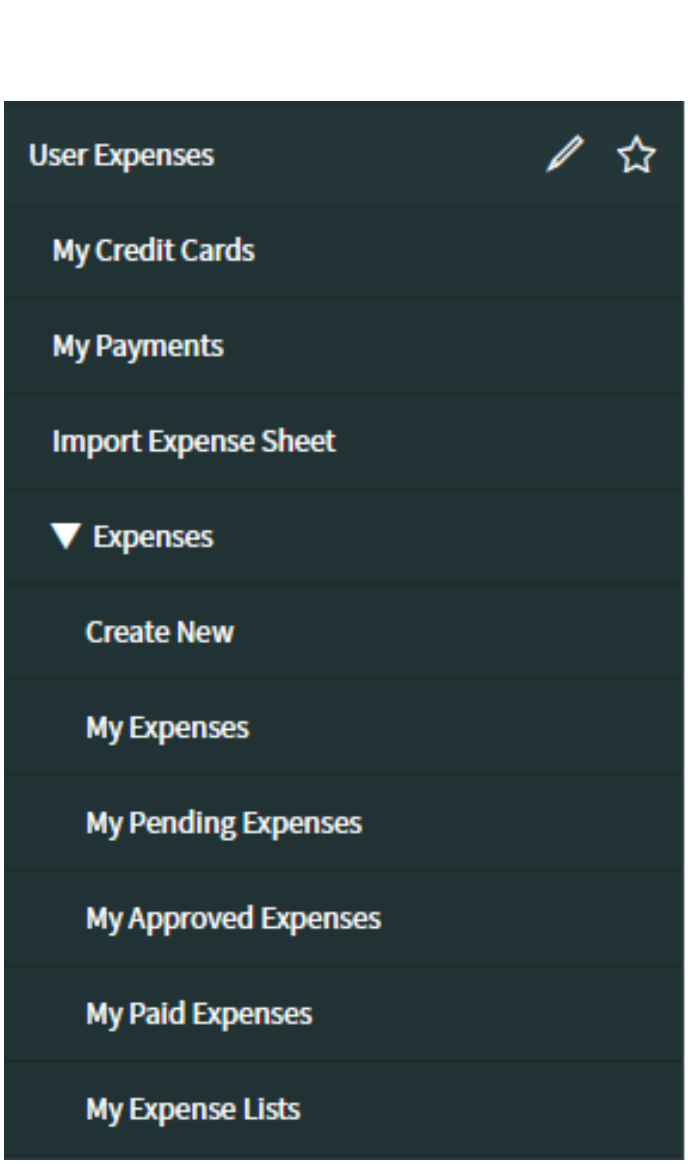
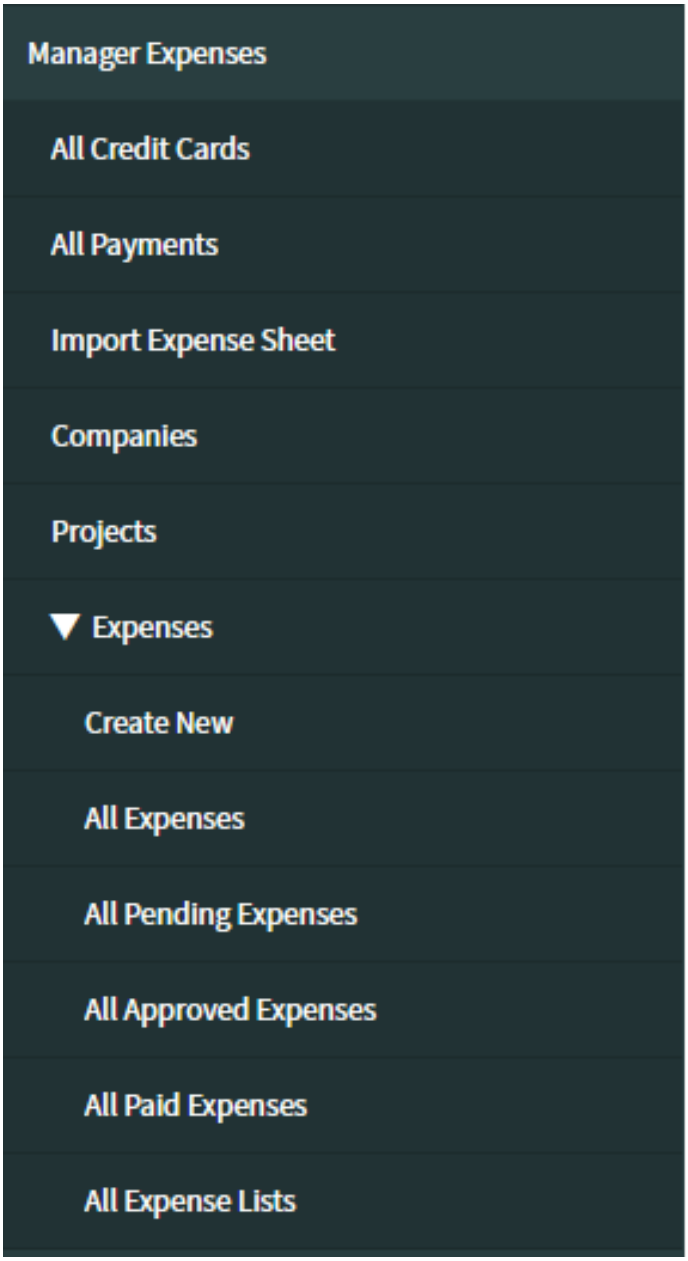


Figura A.14: Visualización de la lista de lineas de gasto por defecto



(a) Menú de aplicaciones para usuarios normales



(b) Menú de aplicaciones para usuarios mánager

Figura A.15: Lista de los menús de aplicación para la aplicación