Proyecto PFC

Iago Morales Camacho
DAW2

Índice

1.Descripción del proyecto y ámbito de implantación

1.1.Introducción

Durante mi estancia en la empresa de prácticas tuve la oportunidad de adentrarme en el mundo de SAP.

Desde el primer momento, quedé cautivado por este framework y me di cuenta de su enorme potencial para automatizar y optimizar procesos empresariales. Esto despertó en mí un gran interés por profundizar en SAP, y como resultado, decidí utilizarlo como base para mi proyecto.

Antes de continuar me parece interesante que entendamos mejor cómo funciona SAP.

Una empresa que desee optimizar una parte de sus operaciones empresariales, como en el caso propuesto, recurrirá a la implementación de SAP, una herramienta altamente completa que ofrece una amplia gama de funcionalidades superiores a las de la competencia.

En la opción de utilizar SAP, la empresa puede optar por desarrollar la aplicación internamente, siempre y cuando cuente con personal capacitado en su plantilla, o bien contratar especialistas en SAP para llevar a cabo el proyecto. Si se elige la segunda opción, la empresa cliente establecerá contacto con una empresa desarrolladora, a la cual comunicará los requisitos específicos de la aplicación. La empresa desarrolladora, a su vez, analizará la viabilidad del proyecto y presentará un presupuesto. Si ambas partes llegan a un acuerdo, se procederá al desarrollo de la aplicación requerida, trabajando en conjunto para lograr los mejores resultados.

Es importante tener en cuenta que una "aplicación" SAP se compone de múltiples aplicaciones más pequeñas. En otras palabras, se busca crear una aplicación individual para cada función. Por ejemplo, si se desea crear una aplicación para gestionar el alta y visualización de clientes, se desarrollarán dos aplicaciones separadas: una para dar de alta clientes y otra para visualizarlos. Estas aplicaciones se integrarán para formar una única aplicación funcional.

El diseño de SAP se distingue por su enfoque en la simplicidad y la moderación a través de diseños minimalistas. Este enfoque busca simplificar la interfaz y eliminar elementos innecesarios, lo que crea una apariencia visual limpia y equilibrada en las aplicaciones.

1.2.Tecnologías usadas: SAP

- **SAP** es una tecnología empresarial desarrollada por la empresa alemana SAP SE. Proporciona soluciones informáticas modulares y escalables que ayudan a las

organizaciones a gestionar y optimizar sus operaciones empresariales en áreas como finanzas, recursos humanos y cadena de suministro.

- **SAP BTP** (Business Technology Platform): Plataforma empresarial integral que ofrece servicios y herramientas para desarrollar, integrar y extender aplicaciones empresariales de SAP en la nube. Incluye capacidades de desarrollo, integración, gestión de datos, análisis y seguridad.

- **SAP BAS** (Business Application Studio): Entorno de desarrollo en la nube que permite a los desarrolladores crear y extender aplicaciones empresariales utilizando diferentes tecnologías de SAP. Proporciona herramientas y servicios para el ciclo de vida completo de las aplicaciones, desde el desarrollo hasta la implementación y administración.

- **SAP HANA Database Explore**: Herramienta para explorar y analizar datos almacenados en la base de datos SAP HANA. Ofrece una interfaz gráfica intuitiva para consultar, visualizar y analizar datos en tiempo real, así como para crear informes interactivos

- **SAP HANA Cloud Central:** Consola de administración centralizada para gestionar y monitorizar las instancias de SAP HANA en la nube. Permite la configuración, supervisión del rendimiento, aplicación de actualizaciones y gestión de la seguridad de las bases de datos SAP HANA en entornos en la nube.

- **SAP Fiori**: Una plataforma de diseño y pautas de experiencia de usuario para desarrollar aplicaciones empresariales intuitivas y centradas en el usuario. Ofrece aplicaciones preconstruidas y plantillas que siguen principios de diseño modernos, proporcionando una experiencia de usuario coherente y optimizada en dispositivos móviles y de escritorio.

- **SAP HANA:** Plataforma de base de datos y computación en memoria desarrollada por SAP. Permite el procesamiento rápido de grandes volúmenes de datos en tiempo real, utilizando tecnología de almacenamiento en memoria y ofreciendo capacidades para análisis empresarial, procesamiento de transacciones en tiempo real y planificación, entre otros.

- **OData:** Protocolo web utilizado por SAP HANA para exponer sus datos y servicios a través de interfaces web. Permite el acceso y la manipulación de datos utilizando protocolos y formatos estándar ampliamente adoptados, facilitando la integración e interoperabilidad entre diferentes sistemas.

1.3.Proyecto

Mi proyecto tiene como objetivo el desarrollo de dos aplicaciones que buscan optimizar la gestión de envíos de paquetes. La primera aplicación permitirá la creación de envíos desde

los almacenes, mientras que la segunda aplicación se encargará de recibir los envíos en las tiendas correspondientes.

Aplicación de creación de Envíos:
Esta aplicación permitirá crear envíos y los paquetes asociados al envío. Podemos configurar los siguientes datos de los envíos:

- Paquetes a enviar.
- Lugar de origen.
- Lugar de destino.
- Fecha de salida.
- Fecha de llegada.
- ID del conductor asignado al vehículo encargado del transporte (la asignación se realizará exclusivamente a conductores disponibles).
- Matrícula del vehículo encargado de llevar los paquetes (la asignación se realizará únicamente a vehículos no ocupados).

Aplicación para recibir los envíos:
Esta aplicación proporciona la funcionalidad de recepción de envíos, permitiendo visualizar tanto los paquetes asociados a cada envío como los muebles contenidos en dichos paquetes.

Página Principal:

- Por defecto, se mostrará una tabla que visualizará los envíos no entregados.
- Se incluirá un botón para acceder a la tabla de envíos entregados.
- Se añadirá un botón que permita visualizar una tabla con los envíos no entregados.
- Se implementará una función de búsqueda que permita buscar los pedidos por su ID.
- Se incluirá un botón para refrescar la tabla.
- Se agregará un botón que permita filtrar los pedidos por fecha de destino y fecha de salida,
- Se agrega un botón para ordenar los envíos por orden ascendente o descendente tanto por fecha de origen como de destino.
- Se incluirá un footer con un botón que abrirá un diálogo para verificar el ID del envío

Página de Detalles del Envío:

- Esta página se abrirá al hacer clic en un envío específico.
- Permitirá visualizar los detalles del envío, así como los paquetes asignados al mismo.

Página de Detalles del Mueble:

- Esta página se accede al hacer clic en un tipo de mueble en particular.
- Proporcionará la visualización de los datos detallados de los muebles.

Además, se desarrollarán las aplicaciones de manera que sean compatibles con dispositivos móviles, garantizando una experiencia de uso óptima para los usuarios.

En cuanto al idioma de la aplicación, se establecerá automáticamente según la configuración regional del dispositivo. Por ejemplo, si la región seleccionada es España, la aplicación se presentará en español. En caso de que la región sea diferente, se utilizará el idioma inglés.

La empresa cliente es la que contempla los cambios que puede tener esta aplicación.
Pero mejoras a futuro podrían ser:

- Añadir usuarios, de tal manera que solo puedan acceder a la aplicación trabajadores autorizados.

- También, en un futuro, y dependiendo de la expansión de la empresa a nivel internacional, se puede plantear la utilización de otros idiomas.

2.Temporalización del proyecto y fases de desarrollo

En esta estancia en las prácticas, estoy trabajando con SAP Fiori, ya que el equipo se especializa en el desarrollo con esta tecnología. SAP Fiori se encarga de desarrollar la parte de "view y controller" de la arquitectura MVC. Al principio, me resultó complicado comprenderlo, pero a medida que pasa el tiempo, se vuelve más ameno y comprensible.

Si bien tenía una idea de cómo abordar esta parte, aún me faltaba desarrollar el aspecto del "model", que implica crear el servicio oData. Esto representa uno de los mayores desafíos, ya que el equipo de la empresa no trabaja específicamente con esta tecnología. Aunque no pudieron brindarme ayuda específica, sí me orientaron.

Inicialmente, me proporcionaron algunos tutoriales, pero tuve que dejarlos debido a que estaban desactualizados. Finalmente, encontré uno que funcionó, pero surgieron aspectos importantes para el desarrollo que no se mencionan en él, por lo que tuve que buscar soluciones y resolver errores por mi cuenta. Cada vez que encontraba un error, tenía que buscar información para solucionarlo.

Debido a estos inconvenientes, la fase de identificación de lo que podía hacer me llevó varias semanas. Además, al utilizar la versión gratuita de SAP BTP, experimento fallos en algunos momentos, lo que dificulta y ralentiza el proceso de creación del proyecto. Sin embargo, una vez que comprendí cómo crear un servicio oData, pude establecer las fases del desarrollo.

Durante el transcurso del proyecto, me percaté de que había asumido un desafío considerablemente ambicioso, el cual requería un mayor conocimiento de oData. Encontré dificultades significativas al intentar modificar e insertar datos en el servicio oData, lo cual se convirtió en un obstáculo para el avance del proyecto. Realicé extensas investigaciones para intentar solucionar esta problemática.

Uno de los retos asociados a SAP es su naturaleza de tecnología de nicho, lo cual limita considerablemente la disponibilidad de recursos informativos para llevar a cabo

investigaciones efectivas. Además, dado que mi equipo no se especializaba en esta tecnología, no conté con su apoyo y orientación. Como resultado, no pude completar la configuración necesaria para lograr las acciones de modificación e inserción en el servicio oData al momento de la entrega del proyecto.

Inicialmente, tenía la expectativa de encontrar una solución a este problema durante el desarrollo del proyecto. Sin embargo, lamentablemente, no fue posible alcanzarla en ese período.

No obstante, en el desarrollo de la documentación y el código del proyecto, se contempla detalladamente el funcionamiento de estas acciones, de manera que cuando se logre resolver el inconveniente, la aplicación pueda estar completamente funcional al 100%.

El desarrollo del proyecto consta de 5 semanas:

Semana 1:

- Durante la primera semana he planeado una fase de planificación y análisis del proyecto. En este análisis he definido las funcionalidades y características para las aplicaciones. He pedido consejo y opiniones al grupo de desarrollo para planificar el proyecto

Semana 2:

- Durante la primera mitad de la semana he iniciado el desarrollo del servicio OData. He desarrollado la lógica necesaria para exponer y consumir los datos que tendrá la aplicación.
- En la segunda mitad de la semana, he comenzado el desarrollo de la aplicación “Recibir envíos”. Creando parte de la página main, junto con sus respectivas vistas y controladores. Estableciendo las interacciones y flujos de datos necesarios para la funcionalidad básica de esta página.

Semana 3:

- Esta semana consiste en terminar de desarrollar las paginas para recibir los envíos. Terminar la página main, shipping_details y visualice_forniture.

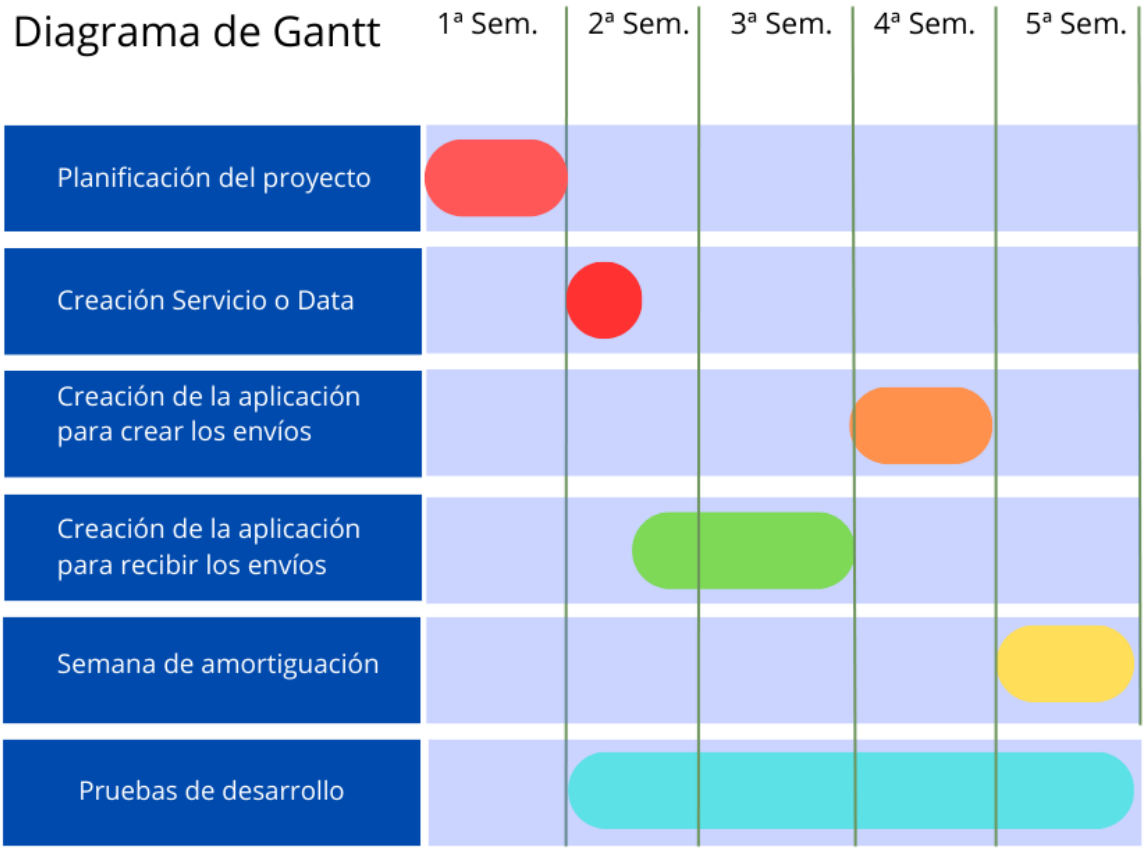
Semana 4:

- Toda la semana la he dedicado al desarrollo de la aplicación “crear envíos”. He creado las páginas y funcionalidades requeridas mencionadas anteriormente en la especificación de la aplicación, siguiendo las mismas prácticas y estándares usados en la aplicación anterior.

Semana 5:

- Durante esta semana, me he enfrentado a los inconvenientes que han ido surgiendo a lo largo del desarrollo. Como por ejemplo a la hora de actualizar el servicio o Data desde Fiori, tanto en la aplicación para crear envíos como en la aplicación para recibirlos.

Quiero destacar que a medida que he ido desarrollando el proyecto he comprobado el funcionamiento de las partes realizadas del mismo.



3. Requisitos mínimos y recomendados

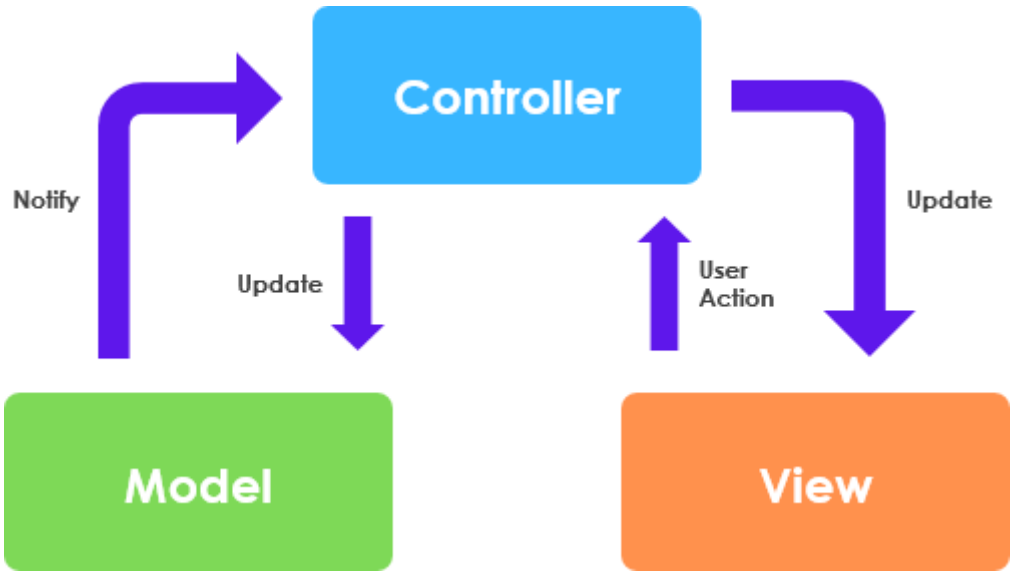
Requisitos Hardware		
Componente	Requisitos Mínimos	Requisitos Recomendados
Procesador	Single core de 3 GHz con arquitectura x86-64	Dual core de 2GHz compatible con arquitectura x86-64
Memoria RAM	2 GB	4 GB
Espacio en disco	2GB	2GB

Requisitos Software		
Componente	Requisitos Mínimos	Requisitos Recomendados

Sistema Operativo	Windows ● Windows 10 ● Windows Server 2012 o versiones posteriores Linux: ● Red Hat Enterprise Linux 7.0 o versiones posteriores ● SUSE Linux Enterprise Server 12.0 o versiones posteriores macOS: ● macOS 10.12 Sierra o versiones posteriores	Windows ● Windows 10 ● Windows Server 2012 o versiones posteriores Linux: ● Red Hat Enterprise Linux 7.0 o versiones posteriores ● SUSE Linux Enterprise Server 12.0 o versiones posteriores macOS: ● macOS 10.12 Sierra o versiones posteriores
Java Development Kit (JDK)	JDK8	Se recomienda utilizar la versión proporcionada por SAP, que se puede descargar desde SAP Development Tools for Eclipse
SAP Development Tools for Eclipse	Descargar el archivo de instalación del Cloud Connector desde SAP Development Tools for Eclipse	Descargar el archivo de instalación del Cloud Connector desde SAP Development Tools for Eclipse

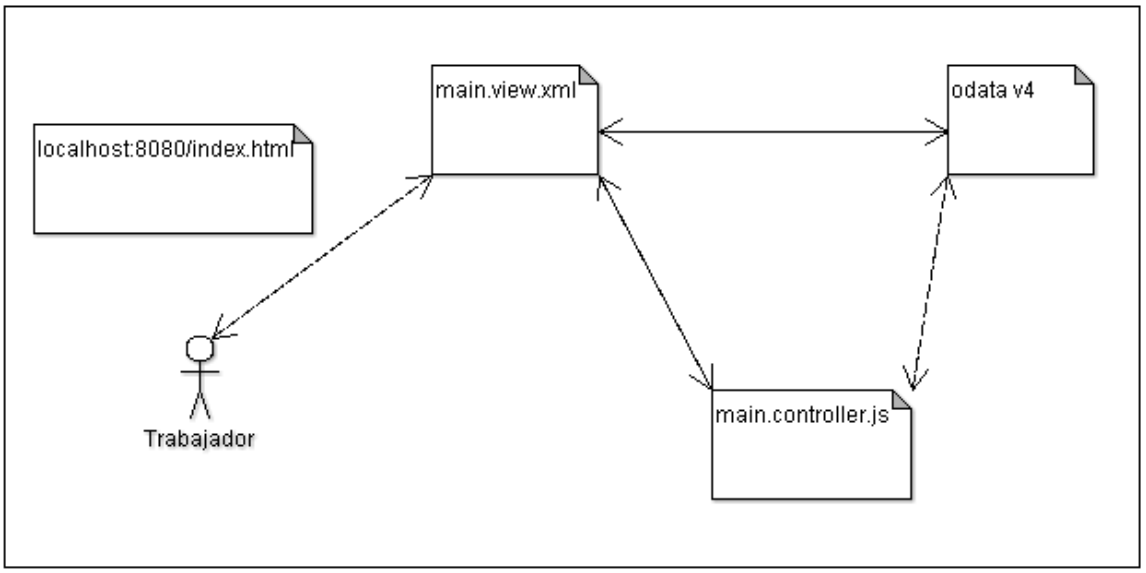
4.1. Arquitectura de Sistemas

Es necesario mencionar la arquitectura MVC para entender mejor cómo funciona SAP



- Modelo (Model): Representa los datos y la lógica de negocio de la aplicación. El modelo es responsable de gestionar la manipulación, el acceso y la persistencia de los datos. En el contexto de SAP, el modelo puede corresponder a las estructuras de datos, la capa de acceso a datos y la lógica empresarial asociada.
- Vista (View): Es la interfaz de usuario que muestra los datos al usuario final. La vista es responsable de la presentación y visualización de los datos. En SAP, las vistas pueden ser páginas web, formularios, informes u otras interfaces a través de las cuales los usuarios interactúan con el sistema.
- Controlador (Controller): Se encarga de manejar las interacciones del usuario y coordinar las acciones entre el modelo y la vista. El controlador recibe las entradas del usuario, realiza las operaciones requeridas en el modelo y actualiza la vista correspondiente con los datos actualizados. En SAP, el controlador puede ser el código que gestiona los eventos y las acciones del usuario en una aplicación o el código que controla la lógica de flujo en una transacción.

Arquitectura de sistemas de la aplicación para crear los envios



En esta aplicación, el único actor humano involucrado es el trabajador encargado de crear el envío. Los otros actores implicados son los componentes de software y servicios utilizados en la arquitectura.

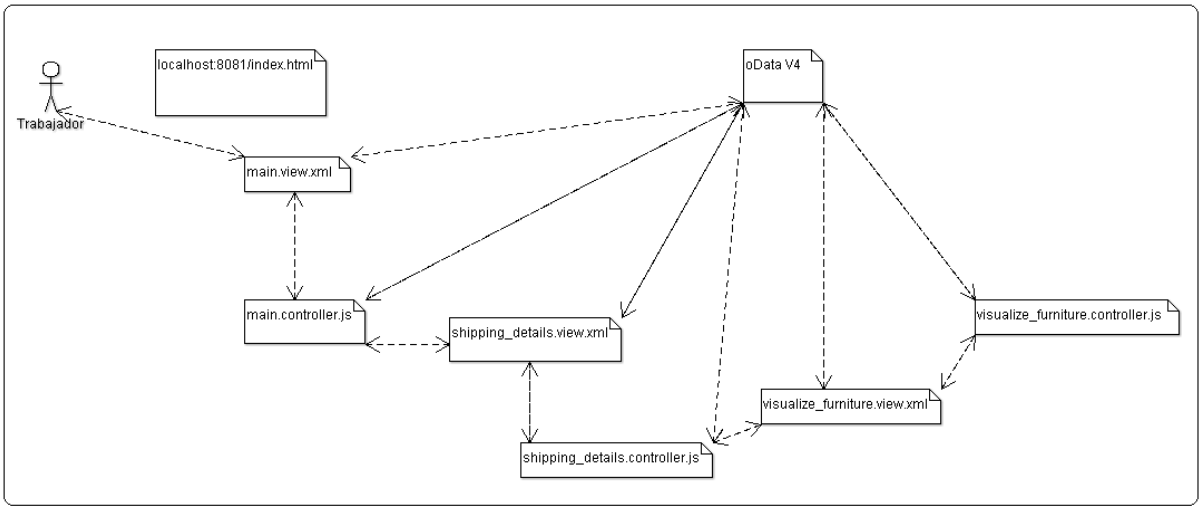
En el lado del cliente (frontend), la aplicación muestra una página principal (main view) que permite al usuario interactuar con la aplicación. Esta vista actúa como una interfaz de usuario donde se visualizan todas las interacciones disponibles. Si una interacción es simple, como seleccionar conductores o indicar lugares de origen y destino, el frontend puede interactuar directamente con un servicio de datos (odata service) para obtener o enviar la información necesaria. Sin embargo, si una interacción es más compleja, como crear un envío, el frontend llama a una función específica en el controlador (backend) para realizar la acción.

En el lado del servidor (backend), se encuentra el controlador, que actúa como una capa intermedia entre el frontend y los servicios de datos. Cuando se requiere una interacción compleja, el frontend llama a la función correspondiente en el controlador. Esta función en el controlador se encarga de procesar la solicitud, recopilar los datos necesarios y comunicarse con el servicio de datos (odata service) para crear el envío.

La comunicación entre los elementos de alto nivel se produce a través de solicitudes y respuestas. El frontend realiza solicitudes al controlador cuando se requieren acciones complejas, y el controlador procesa esas solicitudes, obtiene los datos necesarios y devuelve una respuesta al frontend. Además, cuando se requieren datos simples, el frontend puede comunicarse directamente con el servicio de datos para obtener la información requerida.

Arquitectura de sistemas para recibir envíos.

En esta aplicación, el trabajador sigue siendo el único actor humano que utiliza la



aplicación, mientras que los demás actores son los componentes de software y servicios involucrados.

En el lado del cliente (frontend), las vistas son la interfaz de usuario con la que el trabajador interactúa en la aplicación. Estas vistas representan diferentes páginas y permiten al usuario realizar diversas acciones. Si se trata de operaciones simples que no requieren cambios de página, como ingresar información en un formulario, la vista puede manejar directamente la operación. Sin embargo, si se requiere una operación más compleja que involucre cambios de página, se llama a una función en el controlador correspondiente.

La navegación entre diferentes páginas se realiza a través de llamadas a funciones en el controlador. Cuando el usuario realiza una acción en la vista que requiere un cambio de página, se llama a una función en el controlador. El controlador procesa la solicitud y redirige al usuario a la página indicada.

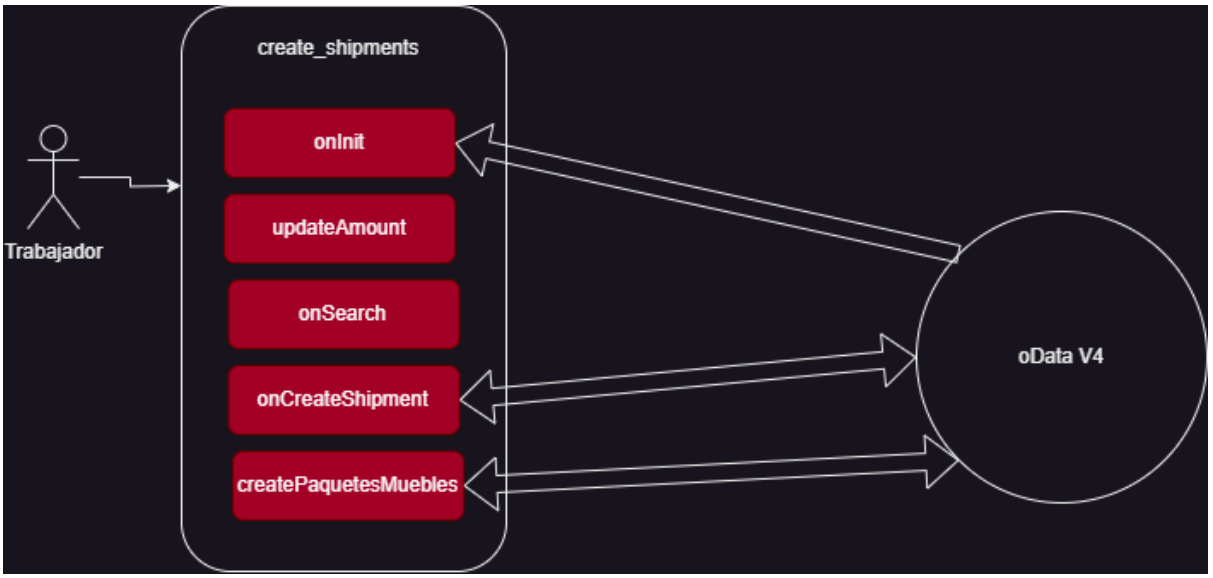
Es importante destacar que el proceso de navegación no implica la participación de OData en ningún momento. El controlador se encarga de gestionar la navegación y enviar al usuario a la página adecuada sin la necesidad de interactuar con servicios de datos.

En esta aplicación, la navegación es lineal, lo que significa que el usuario solo puede ir hacia adelante o hacia atrás en la secuencia de páginas establecida. Esto implica que la navegación se realiza de manera secuencial y no permite saltar directamente entre diferentes páginas.

4.2. Arquitectura de Software

Aplicación para crear envios

main



onInit: Esta función se ejecuta al iniciar la página. Obtiene todos los muebles del modelo y crea un objeto para cada uno de ellos, que incluye el ID y el nombre del mueble, junto con un campo de cantidad inicializado como "null". Los muebles se almacenan en un array llamado furnitureArray. A continuación, se crea un modelo local utilizando JSONModel y se le asigna el array de muebles. Por último, se establece este modelo local en la vista con el nombre "localModel". Los datos de la furnitureArray se visualizan en la tabla correspondiente en la vista.

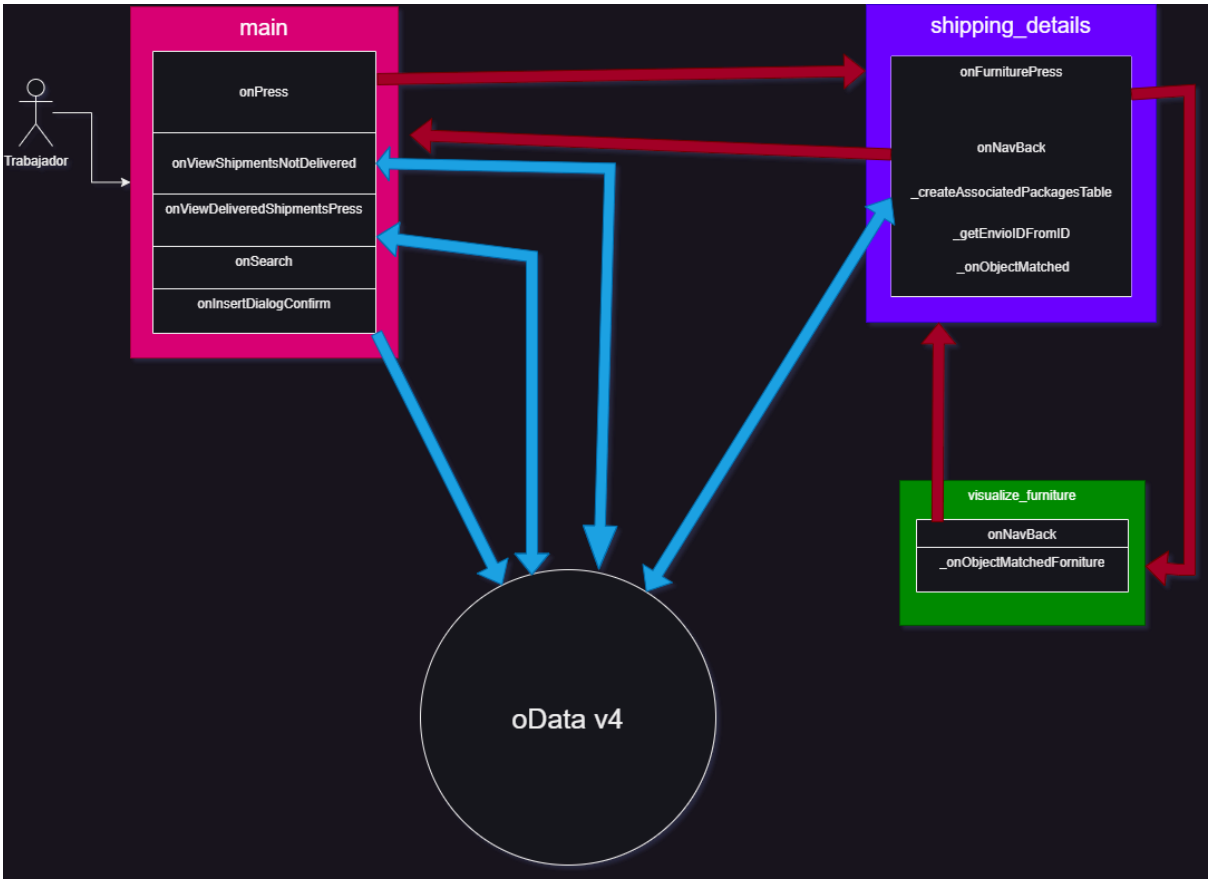
onSearch: Permite buscar por nombre los muebles en la tabla “furnitureTable”.

updateAmount: Esta función se encarga de actualizar la cantidad de un mueble en el array furnitureArray cuando se introduce un número en el campo de entrada correspondiente. Cuando se produce este evento, el valor ingresado se obtiene del campo de cantidad. Luego, se recupera el contexto de enlace y la ruta del elemento actual para acceder al modelo y los datos asociados. A continuación, se crea un nuevo objeto que copia los datos existentes y actualiza el campo de cantidad con el nuevo valor.

onCreateShipment:Esta función permite crear envíos. Primero, obtiene los datos del formulario y formatea las fechas en el formato especificado (YYYY/MM/DD). Luego, verifica que la fecha de llegada no sea anterior a la fecha de salida. Una vez completadas estas validaciones, se genera el ID del envío. Se obtiene el último ID de la entidad /Envios, se extraen los últimos cuatro dígitos, se convierten a entero, se incrementa en 1 y se vuelven a convertir a una cadena, a la cual se le agrega el prefijo "EN". Después de todas estas operaciones, se crea el envío con todos los datos el formulario y se llama a la función createPaquetesMuebles pasándole el ID del envío como parámetro, para así crear los paquetes asociados al envío.

createPaquetesMuebles: Permite crear los paquetes asociados a un envío. Se realiza un recorrido por todos los muebles mediante un bucle for. Si un mueble tiene una cantidad mayor a 0, se procede a crear el paquete correspondiente.
En primer lugar, se genera el ID del paquete. Se obtiene el último ID de la colección /PaquetesMuebles, se extraen los últimos 4 dígitos y se les suma 1 después de convertirlos a un número entero. Luego, se convierte nuevamente el resultado en una cadena de caracteres y se agrega el prefijo "PM" al comienzo del ID generado.
A continuación, se busca el ID del mueble asociado a la cantidad en la colección /Muebles. Se extraen los datos relevantes de ese mueble y se insertan en la nueva colección /PaquetesMuebles.
Por último, se añade el ID del envío, que se pasa como argumento a la función, al paquete creado.
Con esta función, se logra crear los paquetes correspondientes a los muebles seleccionados, estableciendo los ID de manera adecuada y vinculados correctamente con el envío.

Aplicación para recibir envíos



Main

onPress: Esta función se activa cuando en la vista podemos ver los envíos y en el momento que hacemos clic en uno, nos llevará a la página "shipping_details" para ver los detalles del envío y los paquetes asociados al envío.

onViewShipmentsNotDelivered: Esta función se activa cuando pulsamos el botón con el ID "shipmentsNotDeliveredButton". Permite ver los envíos no entregados en la tabla "shippingTable".

onViewDeliveredShipmentsPress: Esta función se activa cuando pulsamos el botón con el ID "shipmentsDeliveredButton". Permite ver los envíos entregados en la tabla "shippingTable".

onSearch: Esta función sirve para buscar un envío por su ID, dependiendo del filtrado de la tabla. Es decir, si la tabla muestra los envíos entregados y yo busco un envío con un ID no entregado, mostrará un mensaje diciendo que no se encuentra el envío. En el caso de ingresar un ID que no tiene asociado un envío, se mostrará el mismo mensaje de error.

onInsertDialogConfirm: Esta función permite recibir el envío, es decir, establece el campo "Entregado" a "true" del envío cuyo ID se ha introducido. Para el conductor y el vehículo asociados al envío, se modifica sus campos "Ocupados" a "false".

Shipping_details

onFurniturePress: Esta función permite acceder a la página "visualize_furniture" y transmitir el ID del mueble asociado al paquete. Al hacer clic en un mueble, se navegará a la página de visualización de muebles y se transferirá el ID del mueble correspondiente.

onNavBack: Esta función permite regresar a la página anterior. Al activarse, se realiza la navegación hacia la página previamente visitada.

_createAssociatedPackagesTable: Desde la vista en la tabla "packagesTable", se muestran todos los envíos y se llama a esta función para filtrar la tabla y mostrar los paquetes asociados al envío seleccionado en la página anterior. Esta función se encarga de crear y presentar una tabla con los paquetes asociados al envío seleccionado.

_getEnvioIDFromID: Cuando en la página principal se proporciona el ID del envío, se pasa en el formato "Envios('EN0001')". Esta función se encarga de extraer el contenido entre paréntesis, que representa el ID real del envío. Luego, este ID se pasa como parámetro a otras funciones, como es el caso de "_createAssociatedPackagesTable", donde se utiliza para realizar operaciones relacionadas con ese envío en particular.

onObjectMatched: Esta función se encarga de obtener el ID del envío a partir de los parámetros de coincidencia de patrón recibidos en el evento. Luego, vincula la vista actual al elemento del modelo correspondiente al ID del envío, lo que permite mostrar los detalles de ese envío en la vista. Además, crea una tabla de paquetes asociados al envío para mostrar los paquetes relacionados con dicho envío. En resumen, esta función se encarga de cargar y mostrar los detalles y la información relacionada con un envío específico en la vista.

Visualize_forniture

onNavBack: Esta función permite regresar a la página anterior. Al activarse, se realiza la navegación hacia la página previamente visitada.

_onObjectMatchedFurniture: Obtiene el identificador del mueble de los parámetros de la ruta, permite mostrar el mueble asociado al id filtrando la tabla por el id.

5 Descripción de datos

Descripción de funcionamiento de la función applyFilters

```
applyFilters: function () {
  //Obtiene el valor seleccionado en el campo de origen
  var selectedOrigin = this.byId("originComboBox").getSelectedKey();
  //Obtiene el valor seleccionado en el campo de destino
  var selectedDestination = this.byId("destinationComboBox").getSelectedKey();
  //Obtiene una referencia a la tabla de envíos
  var oTable = this.byId("shippingTable");
  //Obtiene el enlace de datos de la tabla mediante
  var oBinding = oTable.getBinding("items");
  //Creamos un array para almacenar los filtros
  var oFilters = [];
  if (selectedOrigin) {
    // Crea un filtro para la propiedad "Origen" con el valor seleccionado
    var originFilter = new sap.ui.model.Filter("Origen", sap.ui.model.FilterOperator.EQ, selectedOrigin);
    oFilters.push(originFilter);
  }
  if (selectedDestination) {      You, a few seconds ago • Uncommitted changes
    // Crea un filtro para la propiedad "Destino" con el valor seleccionado

    var destinationFilter = new sap.ui.model.Filter("Destino", sap.ui.model.FilterOperator.EQ, selectedDestination);
    oFilters.push(destinationFilter);
  }
  // Agregar filtro para envíos entregados o no entregados según el valor de isDelivered
  var deliveryStatusFilter = new sap.ui.model.Filter("Entregado", sap.ui.model.FilterOperator.EQ, this.isDelivered);
  oFilters.push(deliveryStatusFilter);
  //Aplica los filtros a la tabla
  oBinding.filter(oFilters);
},
```

En este fragmento de código se encarga de aplicar filtros a una tabla de envíos en función de las opciones seleccionadas en los campos de origen y destino, así como en el estado de entrega. A continuación, se describen los elementos involucrados:

Atributos/Variables:

- selectedOrigin (String): Variable que almacena la clave del origen seleccionado en un combo box.
- selectedDestination (String): Variable que almacena la clave del destino seleccionado en un combo box.
- oTable (Object): Variable que almacena la referencia a la tabla de envíos.
- oBinding (Object): Variable que almacena el enlace de datos de la tabla.
- oFilters (Array): Array que almacena los filtros que se aplicarán a la tabla.

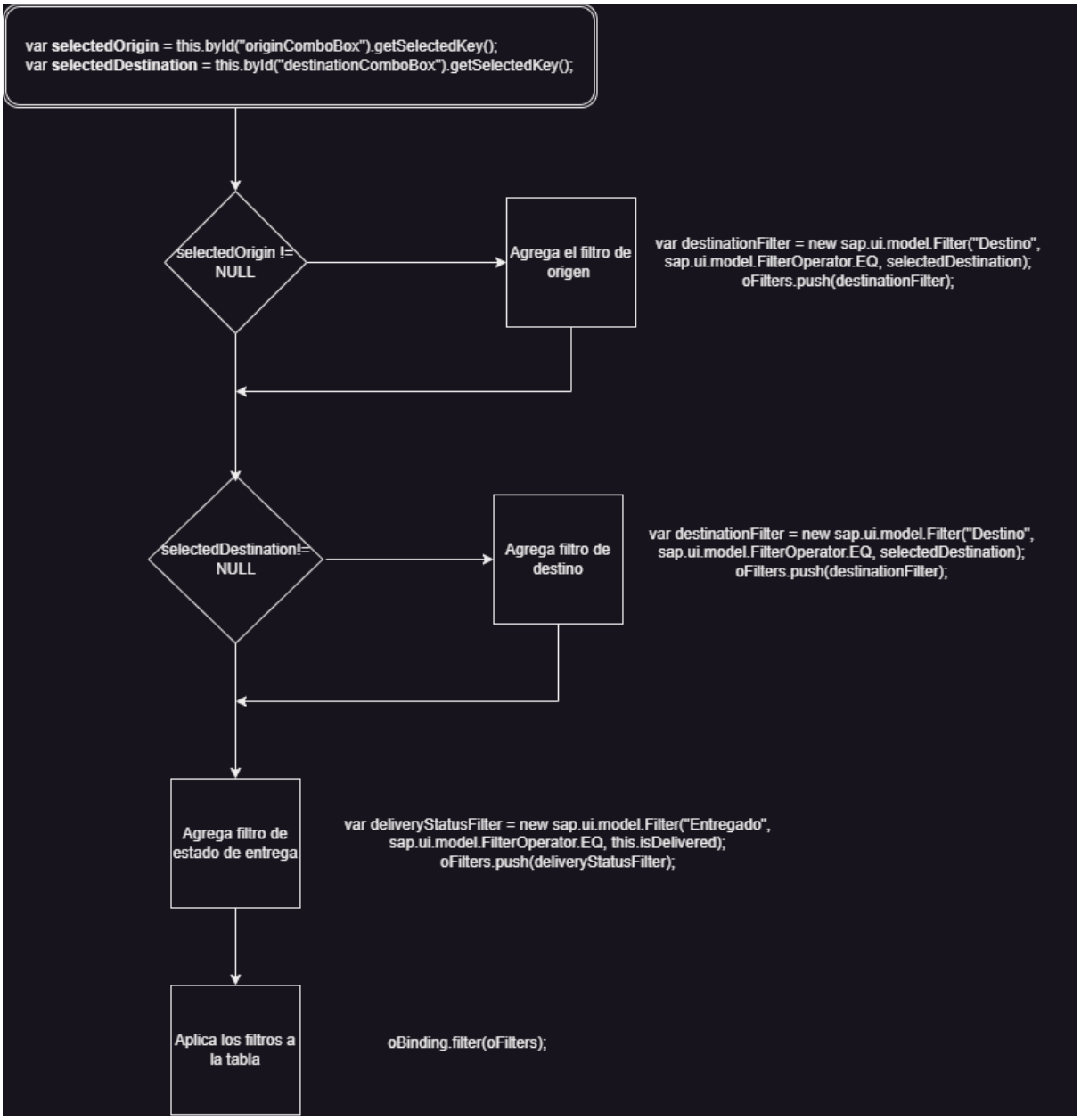
Tipo de datos:

- selectedOrigin y selectedDestination son de tipo String.
- oTable y oBinding son de tipo Object.
- oFilters es un Array.

Funcionamiento:

- Se obtienen las claves seleccionadas en los combo boxes de origen y destino y se almacenan en las variables selectedOrigin y selectedDestination, respectivamente.
- Se obtiene la referencia a la tabla de envíos y se guarda en la variable oTable.
- Se obtiene el enlace de datos de la tabla y se guarda en la variable oBinding.
- Se crea un array vacío oFilters que almacenará los filtros a aplicar.
- Si se ha seleccionado un origen, se crea un filtro basado en la propiedad "Origen" y se agrega al array oFilters.
- Si se ha seleccionado un destino, se crea un filtro basado en la propiedad "Destino" y se agrega al array oFilters.
- Se crea un filtro basado en la propiedad "Entregado" y se agrega al array oFilters utilizando el valor de la variable this.isDelivered.
- Finalmente, se aplica el conjunto de filtros oFilters al enlace de datos de la tabla utilizando el método filter.

Datos de salida:
La función no produce datos de salida directamente, pero al aplicar los filtros a la tabla de envíos, se filtran y muestran únicamente los registros que cumplen con las condiciones establecidas. Esto modifica la visualización de la tabla en la interfaz de usuario.



Descripción de funcionamiento de la función onInserDialogConfirm

verifica y realiza modificaciones de un envío al pasarle el ID. Verifica si el envío ha sido entregado y si el ID del envío existe en el modelo de datos. Luego, modifica el campo "Entregado" del envío a true y guarda los cambios en el modelo. Además, actualiza las propiedades "Ocupado" del conductor y del vehículo relacionados con el envío. La función produce mensajes de éxito o error según el resultado de las operaciones.

```
onInsertDialogConfirm: function () {      You, 3 days ago • Actualizacion recibir envios
  // Obtener el ID del envío desde la propiedad interna guardada
  var sId = this._inputValue;
  // Construir la ruta del envío utilizando el ID
  var sPath = "/Envios('" + sId + "')";
  // Obtener el modelo de datos de la aplicación
  var oModel = sap.ui.getCore().getModel();
  // Comprueba que están definidos los siguientes valores
  if (oModel && sId) {
    // Comprobar si el envío tiene el campo Entregado igual a false
    var oContext = oModel.createBindingContext(sPath);
    var bEntregado = oModel.getProperty(oContext + "/Entregado");
    if (bEntregado == false) {
      // El envío ya ha sido entregado, mostrar mensaje de error y salir de la función
      var sMessage = this.getView().getModel("i18n").getResourceBundle().getText
      ("shipmentAlreadyDelivered");
      sap.m.MessageToast.show(sMessage);
      return;
    }
    // Comprobar si existe el ID del envío
    var bExists = oModel.hasContext(oContext);
    if (!bExists) {
      // El ID del envío no existe, mostrar mensaje de error y salir de la función
      var sMessage = this.getView().getModel("i18n").getResourceBundle().getText
      ("shipmentIdDoesNotExist");
      sap.m.MessageToast.show(sMessage);
      return;
    }
    // Realizar las modificaciones
    oModel.setProperty(oContext + "/Entregado", true);
    oModel.submitChanges({
      success: function () {
        // Los cambios se han guardado correctamente en el modelo de datos
        var sMessageSuccess = this.getView().getModel("i18n").getResourceBundle().getText
        ("shipmentMarkedDelivered");
        sap.m.MessageToast.show(sMessageSuccess);
        // Actualizar la propiedad "Ocupado" del conductor relacionado con el envío
        var sIdConductor = oEnvio.idConductor;
        if (sIdConductor) {
          var sConductorPath = "/Conductor('" + sIdConductor + "')";
          var oConductorContext = oModel.createBindingContext(sConductorPath);
          oModel.setProperty("Ocupado", false, oConductorContext);
```

```
                oModel.setProperty("Ocupado", false, oConductorContext);        You, 3 days ago • Actu
                }
                // Actualizar la propiedad "Ocupado" del vehículo relacionado con el envío
                var sMatricula = oEnvio.Matricula;
                if (sMatricula) {
                  var sVehiculoPath = "/Vehiculos('" + sMatricula + "')";
                  var oVehiculoContext = oModel.createBindingContext(sVehiculoPath);
                  oModel.setProperty("Ocupado", false, oVehiculoContext);
                }
                var sMessageFinal = this.getView().getModel("i18n").getResourceBundle().getText
                ("updateFinished");
                sap.m.MessageToast.show(sMessageFinal);
              },
              error: function () {
                // Error al guardar los cambios en el modelo de datos
                var sMessageErrorShipment = this.getView().getModel("i18n").getResourceBundle().
                getText("errorUpdatingShipment");
                sap.m.MessageToast.show(sMessageErrorShipment);
              }.bind(this)
            });
          } else {
            var sErrorMessage = this
              // Datos inválidos, mostrar mensaje de error
              .getView().getModel("i18n").getResourceBundle().getText("invalidDataError");
            sap.m.MessageToast.show(sErrorMessage);
          }
        }


    });
});
```

Atributos/Variables:

- sId (String): Almacena el ID del envío obtenido desde una propiedad interna guardada.
- sPath (String): Ruta del envío construida utilizando el ID.
- oModel (Object): Modelo de datos de la aplicación obtenido mediante sap.ui.getCore().getModel().

Tipo de datos:

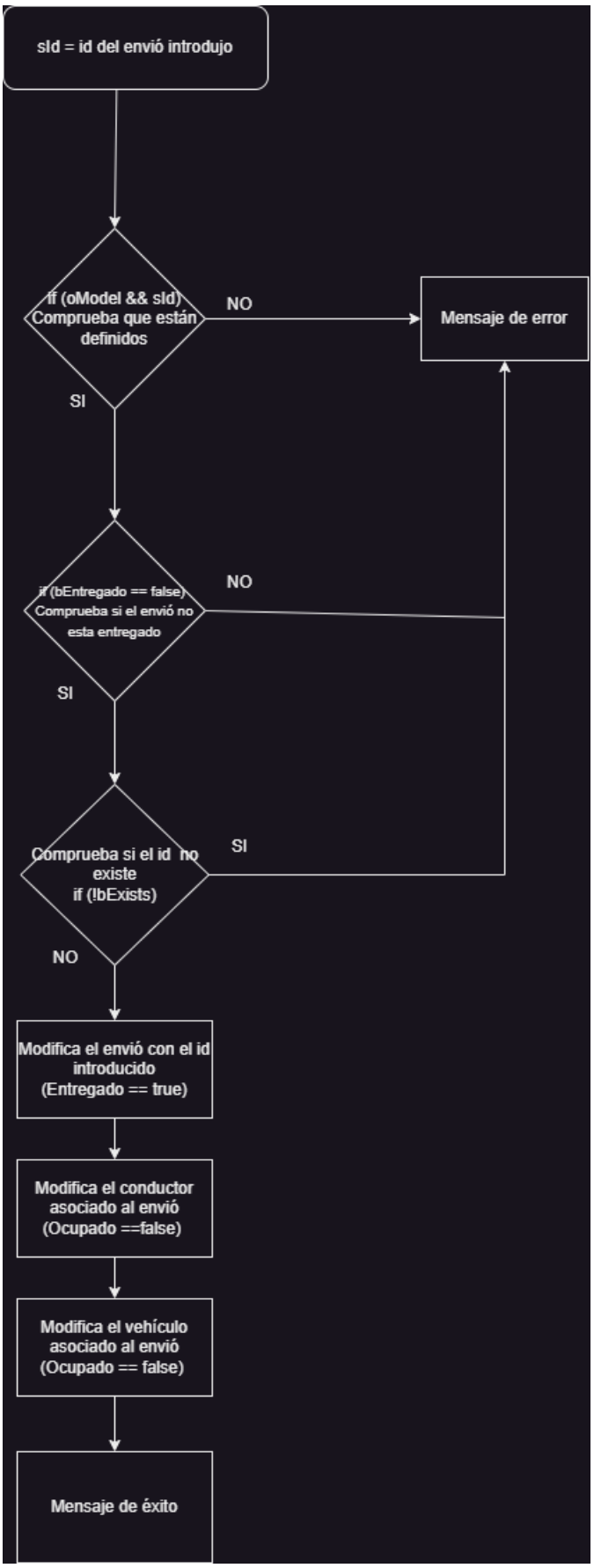
- sId y sPath son de tipo String.
- oModel es de tipo Object.

Funcionamiento:

- Se obtiene el ID del envío desde una propiedad interna guardada y se almacena en la variable sId.

- Se construye la ruta del envío utilizando el ID almacenado en sId y se guarda en la variable sPath.
- Se obtiene el modelo de datos de la aplicación y se guarda en la variable oModel.
- Se comprueba que tanto oModel como sId estén definidos antes de continuar con la ejecución.
- Se crea un contexto de enlace utilizando oModel.createBindingContext(sPath) y se guarda en la variable oContext.
- Se obtiene el valor del campo Entregado del envío mediante oModel.getProperty(oContext + "/Entregado") y se guarda en la variable bEntregado.
- Si el valor de bEntregado es false, se muestra un mensaje de error indicando que el envío ya ha sido entregado y se sale de la función.
- Se comprueba si el ID del envío existe en el modelo de datos mediante oModel.hasContext(oContext).
- Si el ID del envío no existe, se muestra un mensaje de error indicando que el ID del envío no existe y se sale de la función.
- Se realizan las modificaciones en el modelo de datos:
- Se establece el campo Entregado del envío como true utilizando oModel.setProperty(oContext + "/Entregado", true).
- Se guardan los cambios en el modelo de datos utilizando oModel.submitChanges().
- Si los cambios se guardaron correctamente en el modelo de datos, se muestra un mensaje de éxito, se actualiza la propiedad "Ocupado" del conductor relacionado con el envío y se actualiza la propiedad "Ocupado" del vehículo relacionado con el envío.
- Si ocurrió un error al guardar los cambios en el modelo de datos, se muestra un mensaje de error.
- Si tanto oModel como sId no están definidos, se muestra un mensaje de error indicando datos inválidos.

Datos de salida:
La función produce datos de salida en forma de mensajes mostrados mediante sap.m.MessageToast.show(). Los mensajes pueden indicar si el envío ya ha sido entregado, si el ID del envío no existe, si los cambios se guardaron correctamente o si ocurrió un error al guardar los cambios en el modelo de datos


sId = id del envió introdujo
if (oModel && sId)
Comprueba que están definidos
NO
Mensaje de error
SI
if (bEntregado == false)
Comprueba si el envió no esta entregado
NO
SI
Comprueba si el id no existe
if (!bExists)
SI
NO
Modifica el envió con el id introducido
(Entregado == true)
Modifica el conductor asociado al envió
(Ocupado ==false)
Modifica el vehículo asociado al envió
(Ocupado == false)
Mensaje de éxito

Descripción de funcionamiento de la función createPaquetesMuebles

```
createPaquetesMuebles: function(sLastEnvioId) {
  var oView = this.getView();
  var oModel = oView.getModel("localModel");
  var furnitureArray = oModel.getProperty("/furniture");

  var sLastEnvioId = this.getView().getModel("i18n").getResourceBundle().getText("lastEnvioId");

  for (var i = 0; i < furnitureArray.length; i++) {
    var furnitureItem = furnitureArray[i];
    var cantidad = furnitureItem.Cantidad;

    if (cantidad > 0) {
      var idMueble = furnitureItem.id;
      var sMueblesPath = "/Muebles('" + idMueble + "')";

      oModel.bindContext(sMueblesPath, null, {
        success: function(oData) {
          var muebleData = oData.getModel().getProperty(oData.getPath());

          var aPaquetes = oModel.getProperty("/PaquetesMuebles");
          var iNumber = 0;

          if (aPaquetes.length > 0) {
            var lastId = aPaquetes[0].idPaquete;
            iNumber = parseInt(lastId.substr(2)) + 1;
          }

          var sNewNumberPart = iNumber.toString().padStart(4, "0");
          var sNewId = "PM" + sNewNumberPart;

          var oNewPaqueteMueble = {
            idPaquete: sNewId,
            idEnvio: sLastEnvioId,
            idMueble: muebleData.id,
            Nombre: muebleData.Nombre,
            Tipo: muebleData.Tipo,
            Material: muebleData.Material,      You, 19 minutes ago • Uncommitted changes
            Dimensiones: muebleData.Dimensiones,
            Peso: muebleData.Peso,
            Cantidad: cantidad,
            idEnvio:sLastEnvioId
          };
```

```
            oModel.create("/PaquetesMuebles", oNewPaqueteMueble, {
              success: function() {
                MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("shipmentCreated"));
              },
              error: function() {
                MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("errorCreatingShipment"));
              }
            });
          },
          error: function() {
            MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("errorFetchingFurniture"));
          }
        });
      }
    }
  },
```

La función **createPaquetesMuebles** se encarga de crear paquetes de muebles a partir de los datos almacenados en el array furnitureArray. A continuación se describe su funcionamiento:

1. Se obtiene la referencia a la vista actual y al modelo asociado a dicha vista.
2. Se obtiene el valor del último ID de envío desde algún lugar, como un modelo o una propiedad del controlador, y se almacena en la variable sLastEnvioId.
3. Se inicia un bucle for para iterar sobre los elementos del array furnitureArray.
4. En cada iteración, se obtiene un elemento del array y se almacena en la variable furnitureItem. También se obtiene la cantidad de muebles del elemento y se almacena en la variable cantidad.
5. Se verifica si la cantidad de muebles es mayor que cero.
6. Si la cantidad es mayor que cero, se procede a realizar una serie de acciones:
    - Se obtiene el ID del mueble y se crea una ruta de contexto basada en dicho ID.
    - Se realiza un enlace de contexto al modelo utilizando la ruta anteriormente creada.
    - En caso de éxito en la vinculación de datos, se obtienen los datos del mueble a través del contexto.
    - Se obtiene el array de paquetes de muebles y se inicializa la variable iNumber en cero.
    - Si el array de paquetes no está vacío, se obtiene el ID del primer paquete, se extrae una parte del ID y se incrementa en uno.
    - Se genera una nueva parte del ID con ceros a la izquierda y se concatena con el prefijo "PM" para formar un nuevo ID de paquete.
    - Se crea un nuevo objeto oNewPaqueteMueble con los datos del paquete de mueble a crear.
    - Se utiliza el método create del modelo para crear el nuevo paquete de mueble en el servicio de datos.

    - En caso de éxito, se muestra un mensaje de éxito en la vista.
    - En caso de error, se muestra un mensaje de error en la vista.
7. Si ocurre un error en la vinculación de datos, se muestra un mensaje de error en la vista.

En cuanto a los atributos y variables utilizados:

- **oView** y **oModel** son variables de tipo Object que hacen referencia a la vista y al modelo asociado.
- **furnitureArray** es una variable que almacena un array de objetos de muebles.
- **sLastEnvioId** es una variable de tipo String que almacena el último ID de envío obtenido.
- **furnitureItem** es una variable que almacena un objeto de mueble en cada iteración del bucle.
- **cantidad** es una variable de tipo numérico que almacena la cantidad de muebles en cada iteración.
- **sMueblesPath** es una variable de tipo String que almacena la ruta de contexto del mueble.
- **muebleData** es una variable que almacena los datos del mueble obtenidos a través del contexto.
- **aPaquetes** es una variable que almacena el array de paquetes de muebles.
- **iNumber** es una variable de tipo numérico que almacena un número utilizado para generar el nuevo ID del paquete.
- **sNewNumberPart** y sNewId son variables de tipo String que almacenan partes del nuevo ID del paquete.
- **oNewPaqueteMueble** es un objeto que almacena los datos del nuevo paquete de mueble a crear.

En cuanto a los datos de salida, la función no produce datos de salida directamente, pero al crear los paquetes de muebles en el servicio de datos, se puede obtener una respuesta de éxito o error, la cual se refleja en mensajes que se muestran en la vista.

Descripción de funcionamiento de la función onCreateShipment

```
// Función para crear un envío
onCreateShipment: function () {
  var oView = this.getView();// Obtener la vista

  // Obtener los valores de los campos del formulario
  var oForm = oView.byId("shippingData");
  var oFormElements = oForm.getFormContainers()[0].getFormElements();
  var oFormData = {};

  //Recorre los elementos del formulario
  oFormElements.forEach(function (oFormElement) {
    var sLabel = oFormElement.getFields()[0].getId();
    var oField = oFormElement.getFields()[0];
    var sValue;

    // Validar si el campo es un DatePicker y formatear la fecha
    if (oField instanceof sap.m.DatePicker) {
      var oDate = oField.getDateValue();
      sValue = oDate ? oDate.toISOString().split("T")[0] : "";
    } else {
      sValue = oField.getValue();
    }

    oFormData[sLabel] = sValue;
  });

  // Validar la fecha de llegada
  var dStartDate = new Date(oFormData["shippingData--FechaSalida"]);
  var dEndDate = new Date(oFormData["shippingData--FechaLlegada"]);

  if (dEndDate < dStartDate) {
    MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("errorDate"));
    return; // Detener la ejecución de la función si la fecha de llegada es inferior a la de salida
  }

  // Obtener el último ID de /Envios y generar el nuevo ID
  var oModel = this.getOwnerComponent().getModel();
  var sLastEnvioId = "";

  oModel.bindList("/Envios", null, null, null, {
    success: function (oData) {
```

```
    oModel.bindList("/Envios", null, null, null, {
      success: function (oData) {
        var aEnvios = oData.getModel().getProperty("/");
        if (aEnvios.length > 0) {
          var sLastId = aEnvios[aEnvios.length - 1].id;
          var sNumberPart = sLastId.substr(2); // Obtener los últimos cuatro dígitos
          var iNumber = parseInt(sNumberPart); // Convertir a número entero
          iNumber++; // Incrementar en 1
          var sNewNumberPart = iNumber.toString().padStart(4, "0"); // Convertir nuevamente a cadena y relle
          sLastEnvioId = "EN" + sNewNumberPart; // Generar el nuevo ID de envío
        }

        // Crear el nuevo envío
        var oNewEnvio = {
          id: sLastEnvioId,
          fechaSalida: oFormData["shippingData--FechaSalida"],
          fechaLlegada: oFormData["shippingData--FechaLlegada"],
          origen: oFormData["shippingData--Origen"],
          destino: oFormData["shippingData--Destino"],
          idConductor: oFormData["shippingData--idConductorSelect"],
          matricula: oFormData["shippingData--idVehiculoSelect"]
        };

        oModel.create("/Envios", oNewEnvio, {
          success: function () {
            MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("shipmentCreated"));
            this.createPaquetesMuebles(sLastEnvioId);

          },
          error: function () {
            MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("errorCreatingShipment"));
          }
        });
      },
      error: function (oError) {
        MessageToast.show(oView.getModel("i18n").getResourceBundle().getText("errorFetchingData"));
      }
    });
  }      You, 10 days ago • Actualizacion primera aplicacion
```

La función onCreateShipment se encarga de crear un nuevo envío y llamar a la función createPaquetesMuebles para crear los paquetes de muebles asociados a dicho envío. A continuación se describe su funcionamiento:

1. Se obtiene la referencia a la vista actual.
2. Se obtienen los valores de los campos del formulario de envío.
3. Se crea un objeto oFormData para almacenar los datos del formulario.
4. Se recorren los elementos del formulario y se obtiene el valor de cada campo. Si el campo es un DatePicker, se formatea la fecha como una cadena.
5. Se valida la fecha de llegada para asegurarse de que sea posterior a la fecha de salida. En caso contrario, se muestra un mensaje de error y se detiene la ejecución de la función.
6. Se obtiene la referencia al modelo de datos.
7. Se realiza un enlace a la lista de envíos para obtener el último ID de envío.
8. En caso de éxito en la vinculación de datos, se obtiene el array de envíos y se extrae el último ID.
9. Se obtienen los últimos cuatro dígitos del último ID y se incrementa en uno.

10. Se genera una nueva parte del ID con ceros a la izquierda y se concatena con el prefijo "EN" para formar el nuevo ID de envío.
11. Se crea un nuevo objeto oNewEnvio con los datos del nuevo envío a crear.
12. Se utiliza el método create del modelo para crear el nuevo envío en el servicio de datos.
13. En caso de éxito, se muestra un mensaje de éxito en la vista y se llama a la función createPaquetesMuebles pasando el ID del envío creado.
14. En caso de error, se muestra un mensaje de error en la vista.

En cuanto a los atributos y variables utilizados:

- oView es una variable que hace referencia a la vista actual.
- oForm es una variable que almacena el formulario de envío.
- oFormElements es una variable que almacena los elementos del formulario.
- oFormData es un objeto que almacena los datos del formulario.
- dStartDate y dEndDate son variables que almacenan las fechas de salida y llegada obtenidas del formulario.
- oModel es una variable que almacena el modelo de datos.
- sLastEnvioId es una variable de tipo String que almacena el último ID de envío obtenido.
- aEnvios es una variable que almacena el array de envíos obtenido del modelo.
- sLastId, sNumberPart y iNumber son variables utilizadas para generar el nuevo ID de envío.
- sNewNumberPart es una variable que almacena la parte del nuevo ID generada con ceros a la izquierda.
- oNewEnvio es un objeto que almacena los datos del nuevo envío a crear.

En cuanto a los datos de salida, la función no produce datos de salida directamente, pero al crear el nuevo envío y los paquetes de muebles asociados, se puede obtener una respuesta de éxito o error, la cual se refleja en mensajes que se muestran en la vista.