OSCA
SISTEMAS

Aplicación Web Armarios

Documento técnico

Hoja de control

Título	Arquitectura aplicación web gestión armarios		
Autor	José Ángel Caudevilla Casaús		
Versión/Edición	V.1.5	**Fecha Versión**	02/03/2018
Revisado por	José Ángel Caudevilla Casaús	**Fecha Revisión**	02/03/2018
Aprobado por	José Ángel Caudevilla Casaús	**Fecha de Aprobación**	02/03/2018
		Nº Total Páginas	

CONTROL DE CAMBIOS

Versión	Causa del Cambio	Responsable	Fecha
V 1.0	Versión inicial	José Ángel Caudevilla Casaus	02/03/2018

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos	Cargo	Área / Empresa
José Ángel Caudevilla Casaús	Ingeniero informático	Osca sistemas

Tabla de contenido

1. Introducción

1.1. Objetivo

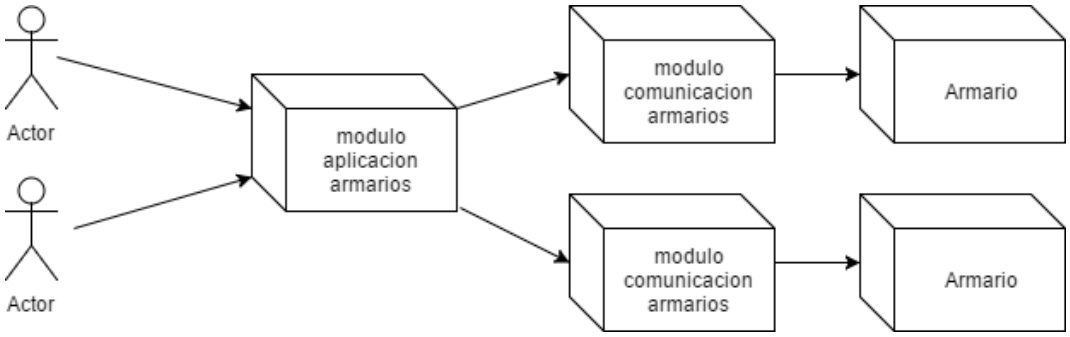
El objetivo fundamental del documento que se presenta a continuación es la realización de un diseño técnico pormenorizado de los procesos que desarrollan los casos de usos recogidos en el análisis funcional realizado con anterioridad, necesarios para la implementación de la **aplicación web de gestión y control de armarios Hänel Lean-Lift.**

Así, este documento tiene por objetivo la recopilación del diseño técnico y trabajo realizado sobre la aplicación para guiar al desarrollador durante la fase de implementación.

2. Arquitectura de la Aplicación

2.1. Arquitectura general

El siguiente diagrama muestra la arquitectura general del sistema. Por un lado, están los usuarios que se conectan al **módulo aplicación armarios** y por otro esta los **módulos de comunicación armarios** encargados de recibir las peticiones de los usuarios y realizar operaciones sobre los armarios automáticos. La iteración entre los actores y servidores se realiza a través del servidor de la aplicación.



Arquitectura general

Los siguientes apartados abordan la arquitectura interna de los módulos **aplicación armarios** (aplicaciones y servicios) y su implementación y el módulo **comunicación armarios** (aplicaciones y servicios) y su implementación.

2.2. Contexto tecnológico

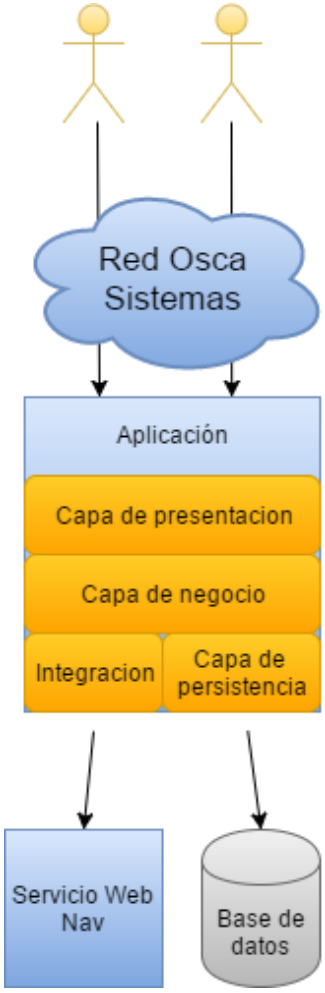
En el diseño de las aplicaciones distribuidas, se ha convertido en un principio ampliamente aceptado la división de cada aplicación en componentes que ofrezcan los servicios de presentación, lógica de negocio y persistencia (acceso a los datos). Los componentes que realizan los mismos tipos de funciones se pueden agrupar en capas que están organizadas en una jerarquía o pila. Así, los componentes que están en una misma capa interactúan entre ellos directamente y con los componentes de las capas inferiores-pero nunca invocan a los componentes de las capas superiores.

Es importante tener en cuenta que las capas son simplemente agrupaciones lógicas de los componentes de software que conforman la aplicación o servicio. Las capas ayudan a diferenciar entre los tipos de tareas que realizan los componentes y facilitan la comunicación del diseño y la reutilización en la solución.

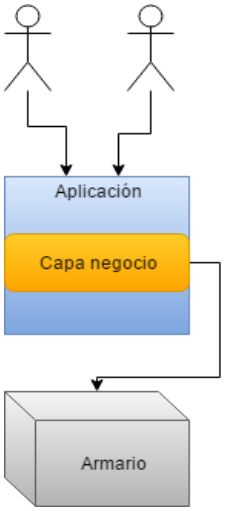
Basándome en los principios generales de diseño de arquitecturas de software, he establecido unas pautas para construir la arquitectura de los módulos **aplicación armarios** y **comunicación armarios** separando en capas funcionales cada uno de los "componentes" o "servicios" que darán la funcionalidad requerida de cada módulo. Esto nos reportara grandes beneficios en términos de reutilización, escalabilidad, mantenimiento y la optimización del funcionamiento a través de las últimas tecnologías existentes.

2.3. Arquitectura interna

Teniendo en cuenta lo anterior, a continuación, se muestra el diseño detallado de la aplicación. Los siguientes apartados, explican la función de cada una de las capas y los componentes que se usan para su implementación, el módulo aplicación armarios posee la siguiente arquitectura.



El módulo armarios no va a tener ninguna capa de persistencia de datos ni ninguna capa de presentación al cliente, solo va a estar formado por la capa de negocio, es siguiente diagrama muestra la arquitectura del módulo armarios.



La siguiente tabla resume las características de cada una de las capas. Los apartados que vienen a continuación las describen con más detalle.

Capa	Descripción	Componentes
Presentación	Implementación de las páginas de la interface gráfica del usuario. Es la cara visible de la aplicación.	Páginas JSP, HTML, Bootstrap, JQuery, Ajax, CSS.
Negocio	Implementación de la lógica del funcionamiento de la aplicación, gestión de transacciones, conversiones de tipos de datos. Es el corazón de la aplicación.	Objetos de negocio y servicios implementados con clases java y que se integran con Spring.
Persistencia	Permite el acceso a la base de datos relacional	Entities de persistencia(Hibernate), tablas de la base de datos.
Integración	Gestiona el acceso a todos los sistemas externos: Servicio Web, servidor comunicación armarios.	Servicio con stubs de acceso a servicios Web(SOAP), peticiones AJAX.

2.4. Capa de presentación

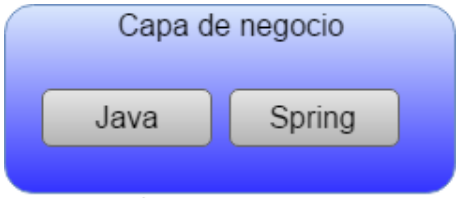
En el caso de las aplicaciones, la capa de presentación se encarga de implementar la interacción con los usuarios finales. Se implementa con paginas JSP, código HTML y Javascript. Para facilitar el desarrollo de esta capa se usan los Frameworks Bootstrap y JQuery.

Componentes y librerías utilizados para implementar la capa de presentación

2.5. Capa de Negocio

La capa de negocio está compuesta de objetos de negocio (beans de java) y un conjunto de servicios. Los objetos de negocio son las entidades identificadas en la aplicación, como, por ejemplo: las operaciones sobre una caja, el registro de una línea de picking... Los servicios se encargan de gestionar las transacciones y coordinan el flujo de las operaciones que se inician con las acciones de los usuarios desde la capa de presentación.

Esta capa delega los accesos a la base de datos en la capa de persistencia y las invocaciones de servicios remotos a la capa de integración. Finalmente, aquí también se generan los reportes (con POI).



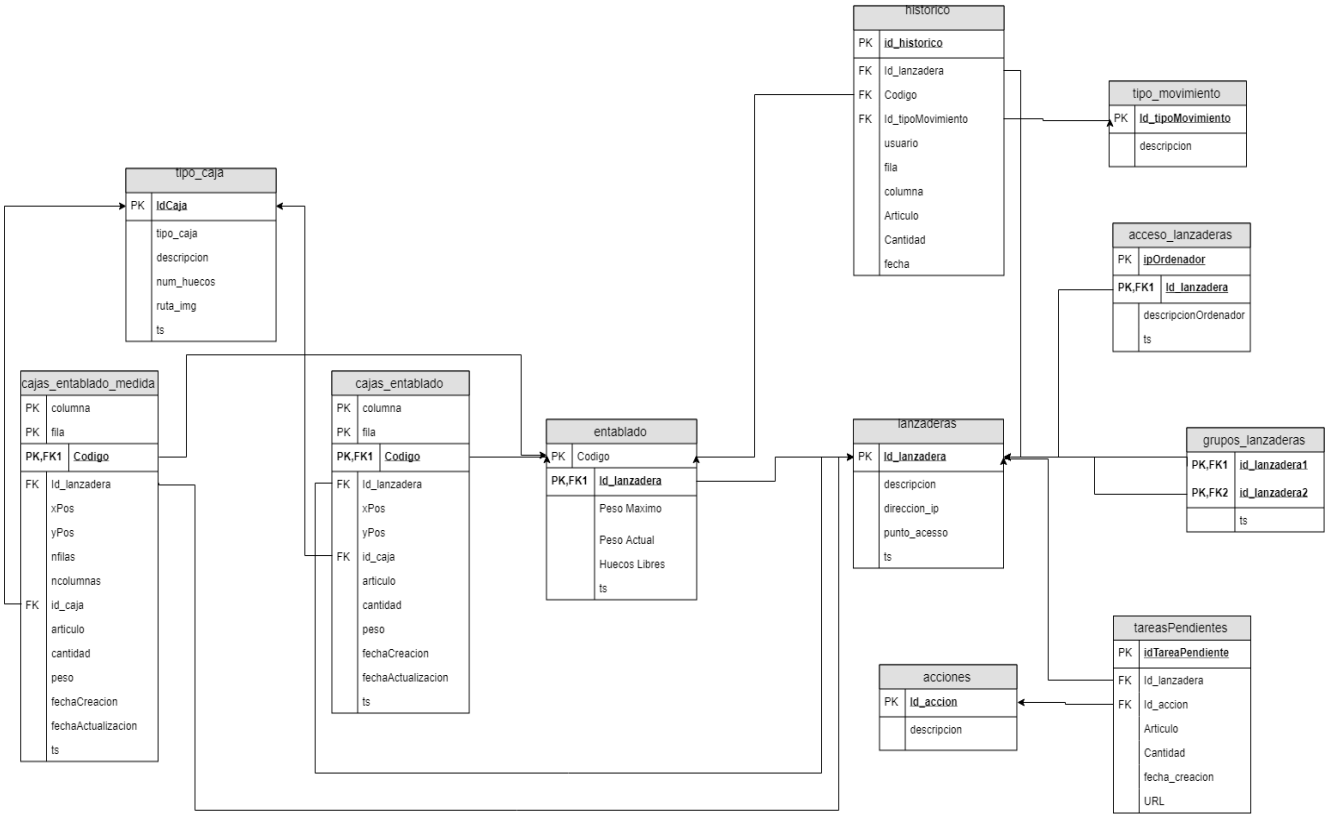
Componentes y librerías utilizados para implementar la capa de servicio

2.6. Capa de Persistencia

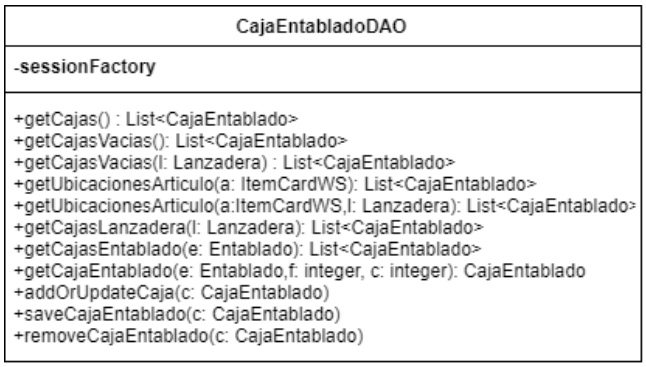
Esta capa, implementada con Hibernate, centraliza todo el acceso a la base de datos relacional (MySQL de Oracle) para hacer consultas, inserciones, actualización y borrado de datos en las tablas. Los datos mantenidos en la base de datos son representados con objetos denominados *entities* que definen el mapeo con las columnas de las tablas utilizando anotaciones especiales en las clases.

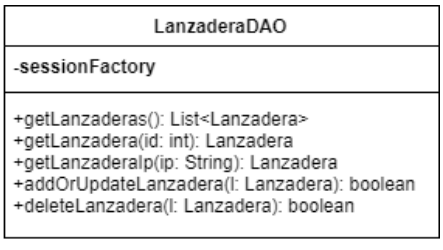
Componentes y librerías utilizados para implementar la capa de persistencia

El siguiente diagrama de clases muestra una parte de la jerarquía de *entities* de la capa de presentación de la aplicación:



Hibernate también proporciona un lenguaje de consulta para recuperar información de la base de datos. En nuestro caso, todas las operaciones de acceso a la base de datos están implementadas en unas clases Java que se denominan DAO. A continuación, se muestran algunas de esas clases:





2.7. Capa de Integración

Esta capa gestiona las invocaciones a los servicios remotos (llamada al servidor de comunicación, Servicios Web). Las llamadas SOAP a los servicios web se invocan mediante **stubs** generados con JAVA-WS. Las llamadas al servidor de comunicación se invocan mediante peticiones AJAX desde el lado del cliente.

Componentes y librerías utilizados para implementar la capa de integración

2.8. Versiones de las Librerías

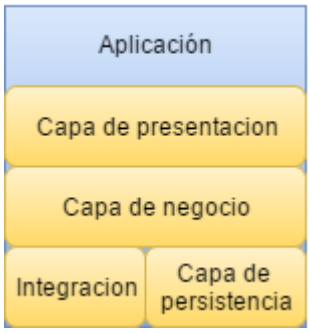
La siguiente tabla resume las versiones de las principales librerías utilizadas en la implementación de los módulos.

Nótese que la tabla no incluye todas las dependencias requeridas ni las librerías auxiliares.

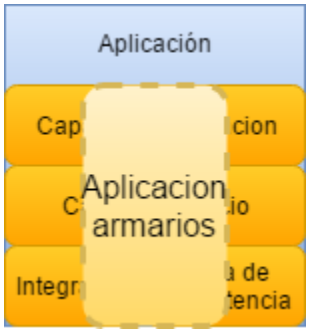
Librería	Versión
Spring	4.2.1
Gson	2.8.1
Hibernate	4.3.5
Java-WS	2.2.6
Mysql	5.1.6
JQuery	1.8.3

3. Organización Interna de los Módulos

En el apartado anterior se presentaba la siguiente jerarquía de capas "lógicas": Capas de presentación, Capa de negocio y capa de persistencia:

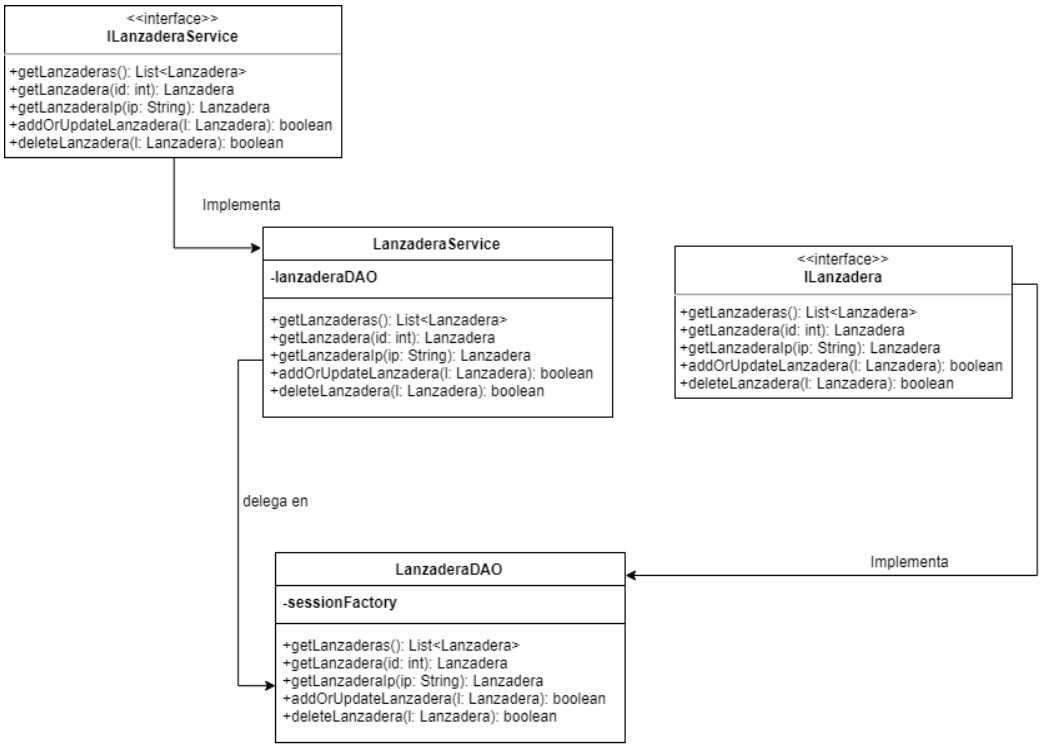


Las capas implementan diferentes aspectos del módulo de la aplicación y por lo tanto pueden verse que son transversales a estos.

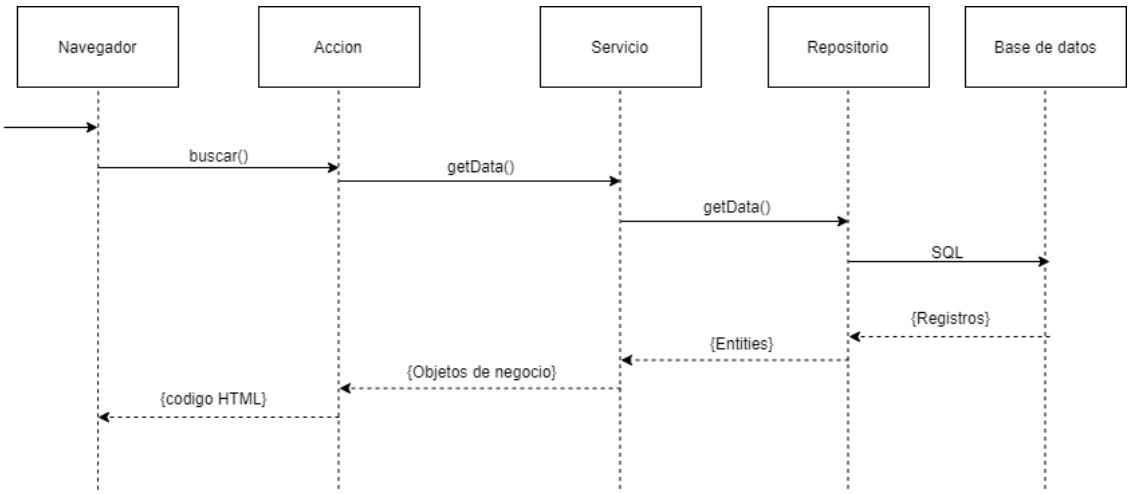


Las capas se integran entre si mediante el mecanismo de inyección de dependencias de Spring (anotación **@autowired**).

El siguiente diagrama muestra la cadena de delegación de responsabilidades desde un componente de la capa de presentación(**ITablaService**) en un servicio de la capa de negocio (**TablaService**) y este a su vez delega en un repositorio de la capa de persistencia (**TablaDAO**) que implementa una interfaz (**ITabla**):



El siguiente diagrama de secuencia muestra el flujo completo de invocaciones desde la capa de presentación (paginas HTML) hasta la capa de persistencia:



Los siguientes apartados describen los detalles de la implementación de este flujo en cada una de las capas.

4. Módulo de entrada de ubicación

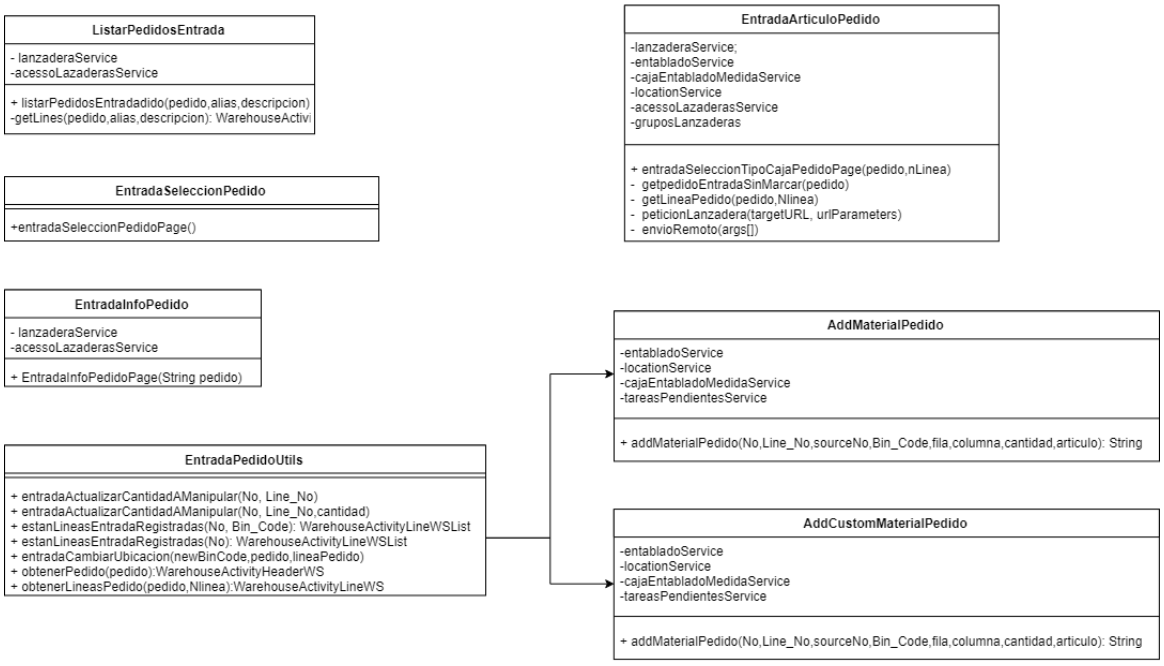
4.1. Funcionalidad

El módulo de entrada se encarga de:

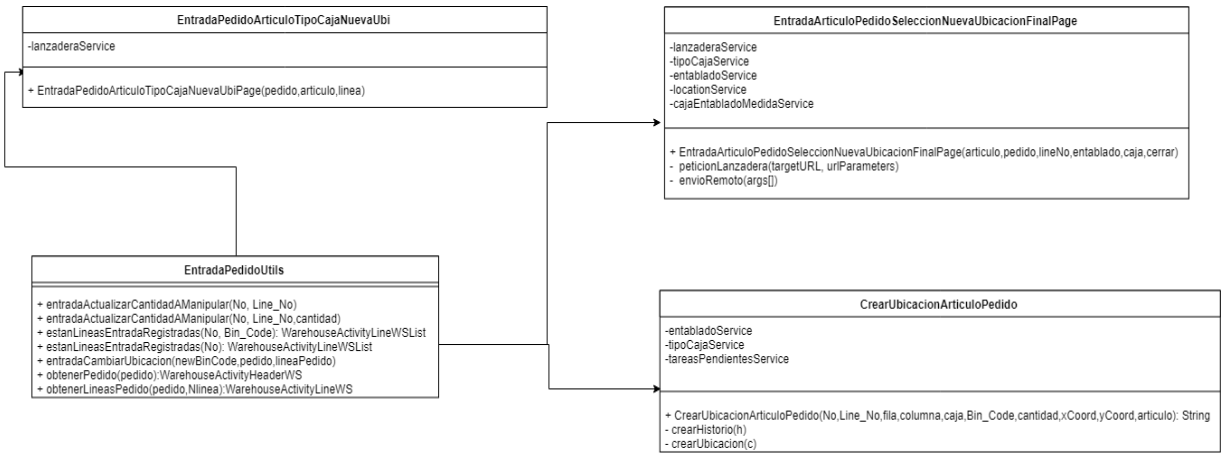
- Devolver paginas HTML relacionadas con entradas de ubicaciones.
- Traer los entablados en donde se encuentre el articulo
- Realizar búsquedas de ubicaciones.
- Procesar las entradas de las ubicaciones.
- Crear una nueva caja desde una entrada de ubicación.
- Reubicar una caja desde una entrada de ubicación.
- Añadir varias líneas de ubicación que contengan el mismo artículo en el mismo entablado.

4.2. Diseño

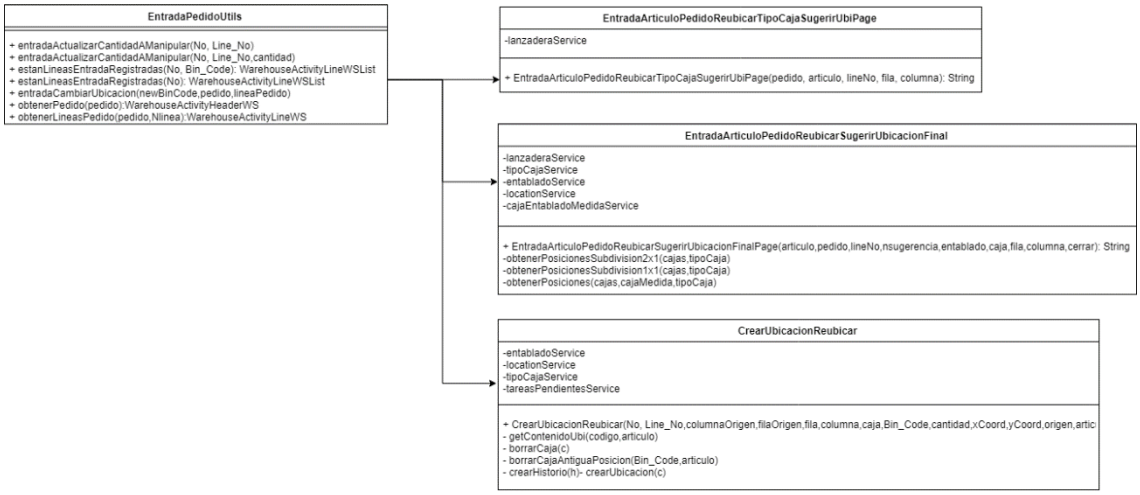
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la entrada de una ubicación:



El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la creación de una nueva ubicación desde una entrada de ubicación:



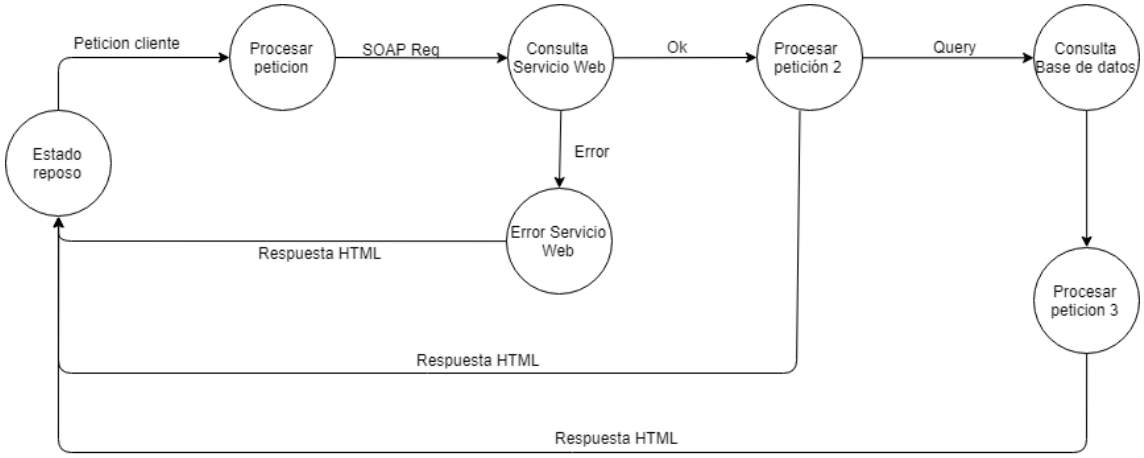
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la reubicación desde una entrada de ubicación:



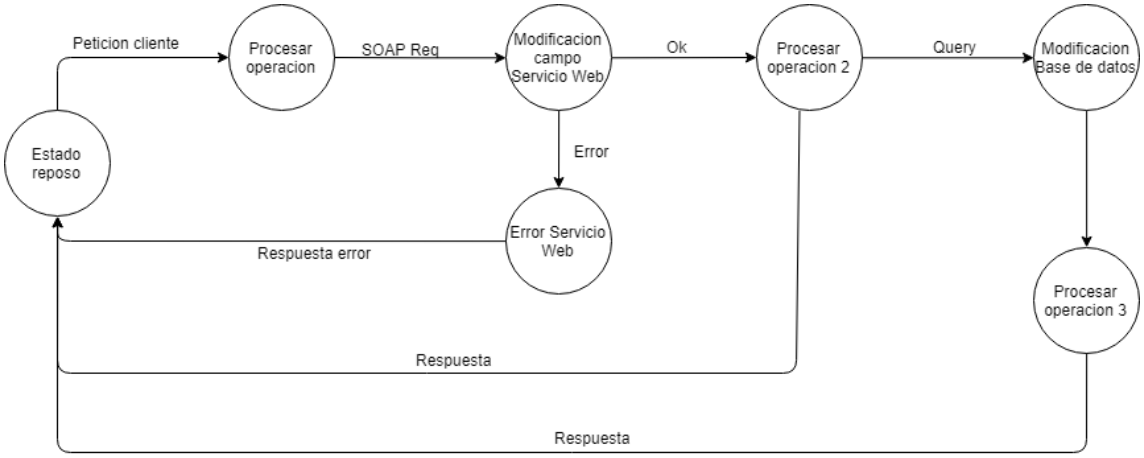
4.3. Transiciones entre Estados

Durante su ciclo de vida, los cambios de estado son iniciados por el usuario utiliza la aplicación e invoca acciones sobre el servidor (mostrar una página, registrar una línea de pedido…) algunas transiciones provocan consultas al servicio web, estas a su vez pueden producir un error, otras sobre la base de datos MySQL.

El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una petición de página:

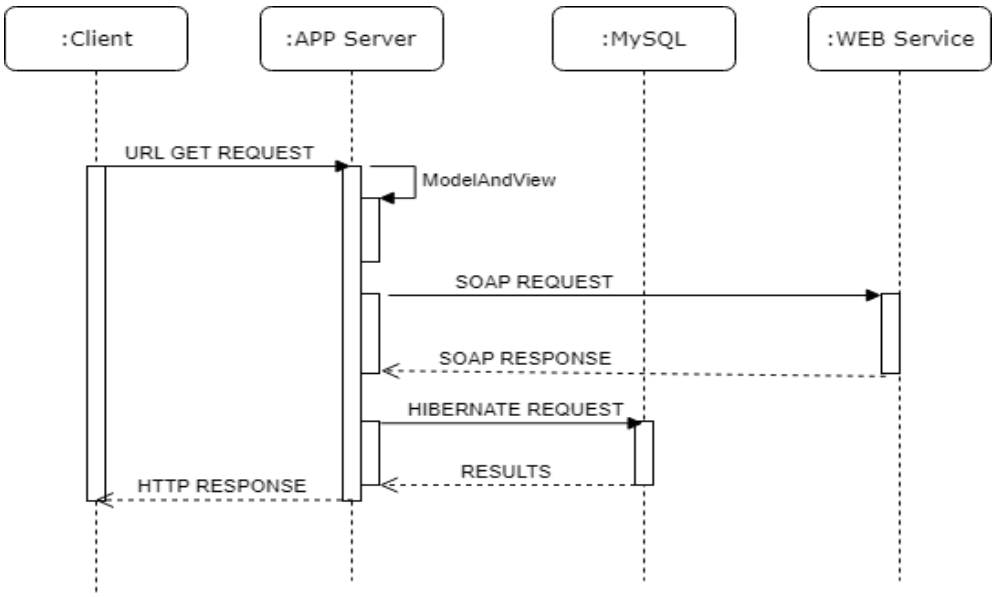


El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una función (registrar línea pedido, actualizar cantidad, etc.):

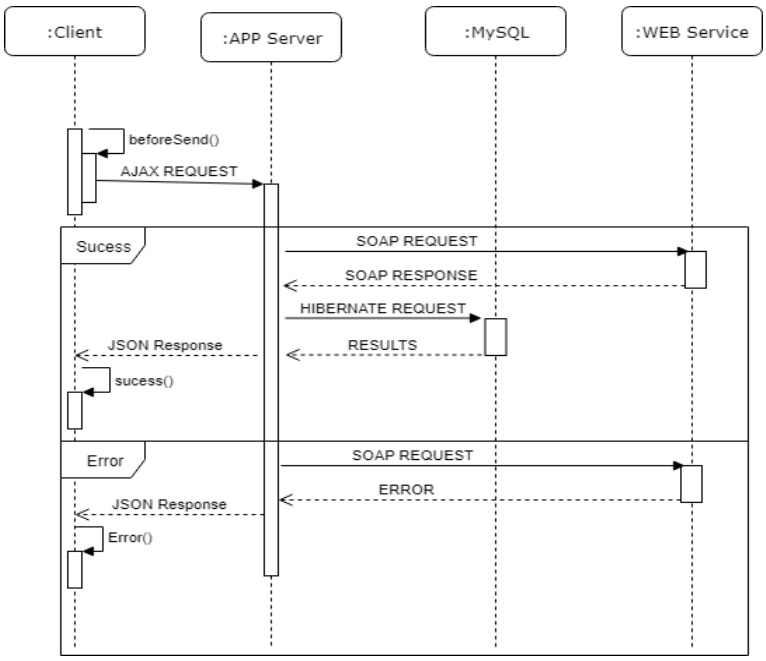


4.4. Diagrama de secuencia

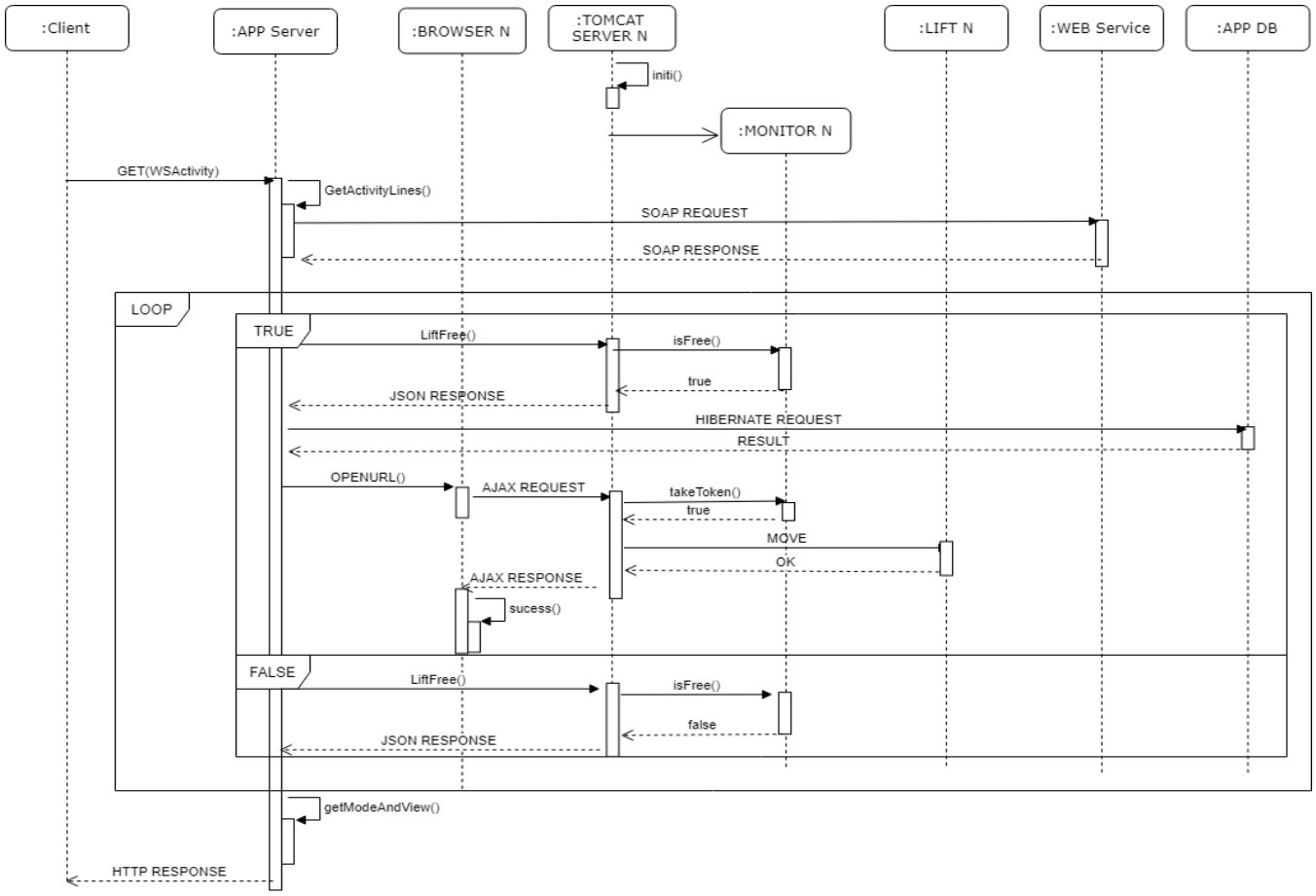
El siguiente diagrama muestra el envío de mensajes entre los componentes del sistema cuando se recibe la petición de un cliente para procesar una página web:



El siguiente diagrama muestra el envió de mensajes entre los componentes del sistema cuando el cliente quiere llamar a una función del servidor:



El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere procesar una línea de ubicación en cada una de las lanzaderas implicadas:



5. Módulo de entrada manual

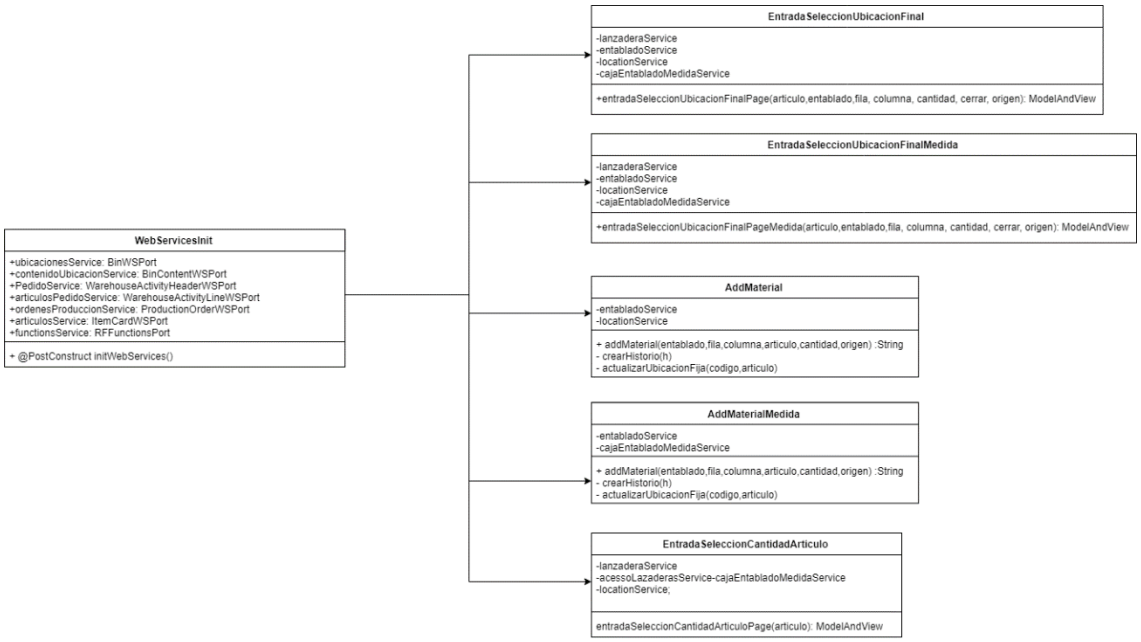
5.1. Funcionalidad

El módulo de entrada manual se encarga de:

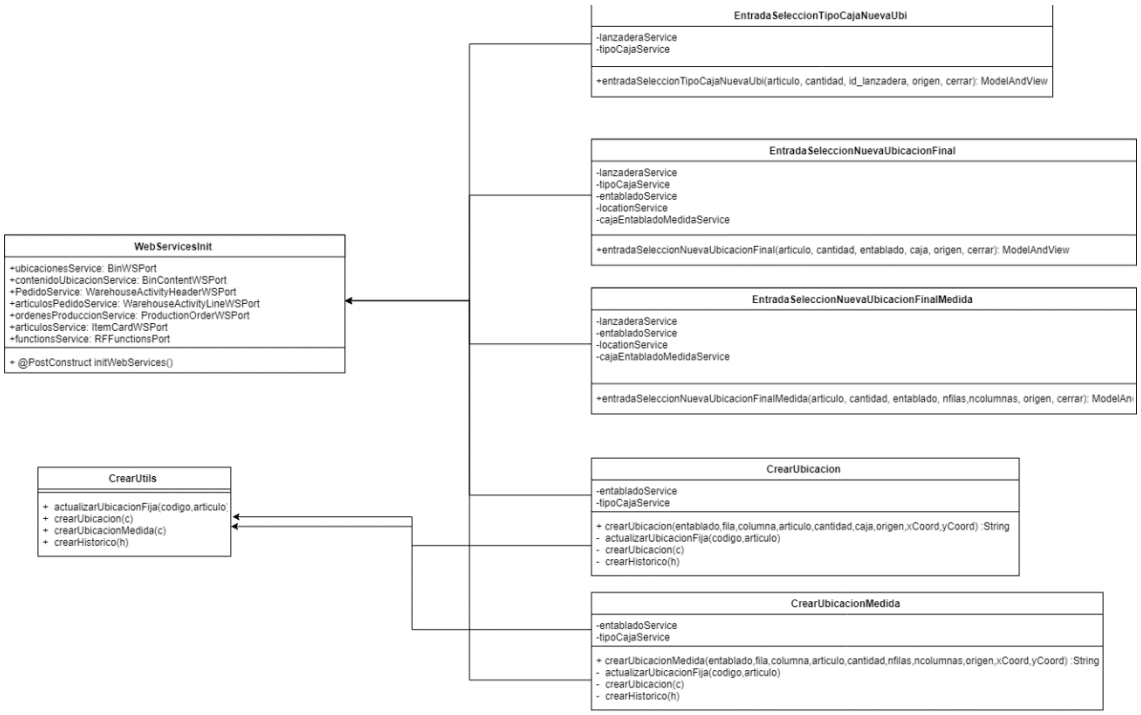
- Devolver paginas HTML relacionadas con entradas manuales
- Realizar búsquedas de artículos
- Añadir material en ubicaciones
- Crear una nueva ubicación dentro de una lanzadera
- Sugerir una nueva ubicación dentro de una lanzadera

5.2. Diseño

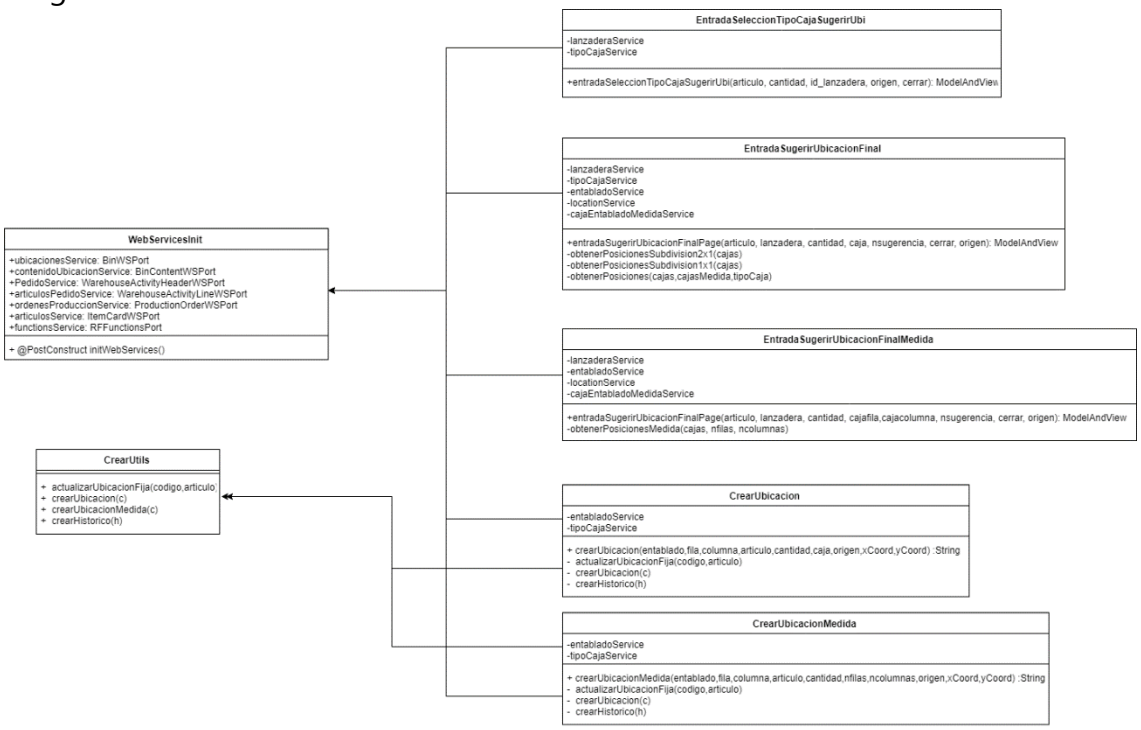
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en el proceso de añadir material en una determinada ubicación:



El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la creación de una nueva ubicación:



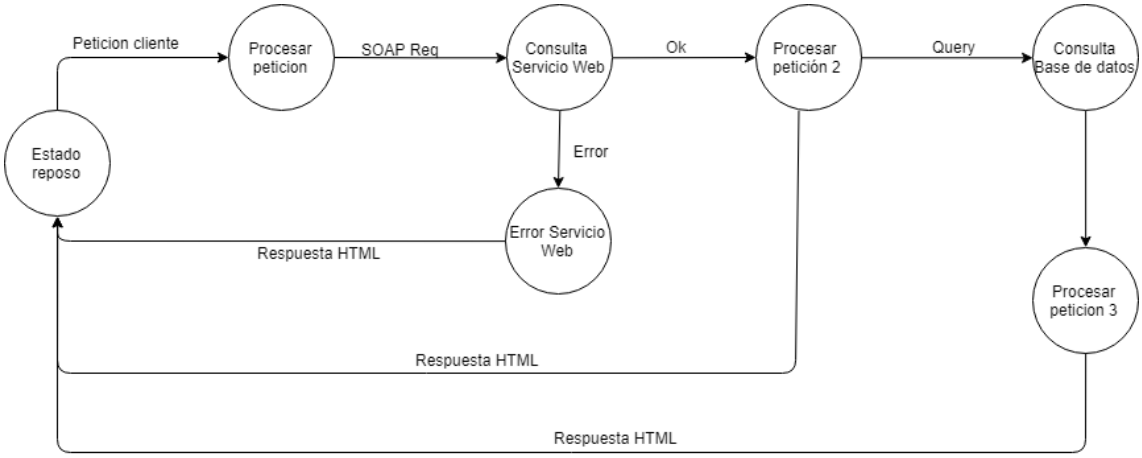
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la creación de una nueva ubicación mediante sugerencia de ubicación:



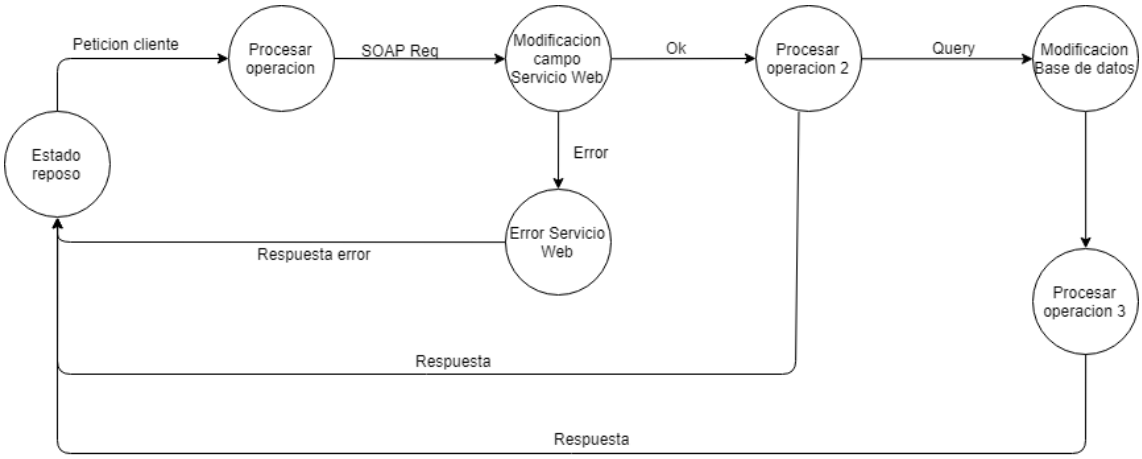
5.3. Transición entre estados

Durante su ciclo de vida, los cambios de estado son iniciados por el usuario utiliza la aplicación e invoca acciones sobre el servidor (mostrar una página, crear una nueva ubicación…) algunas transiciones provocan consultas al servicio web, estas a su vez pueden producir un error, otras sobre la base de datos MySQL.

El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una petición de página:

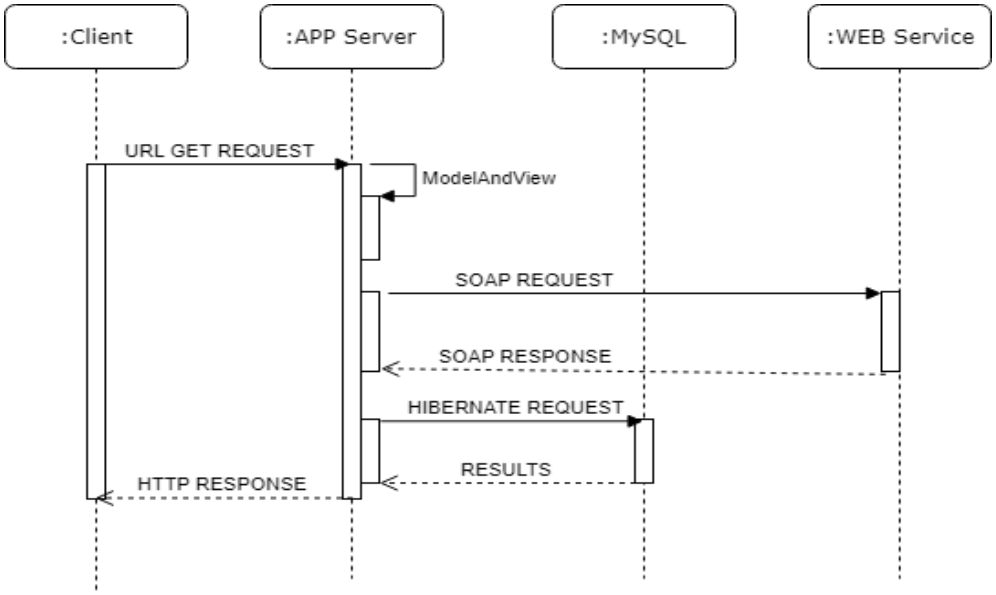


El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una función (crear una nueva ubicación, añadir material, etc.):

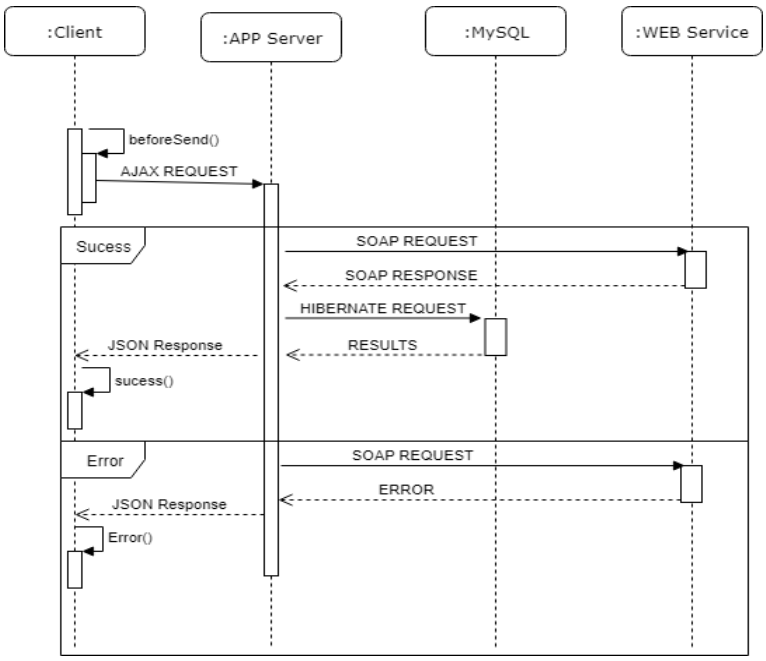


5.4. Diagrama de secuencia

El siguiente diagrama muestra el envío de mensajes entre los componentes del sistema cuando se recibe la petición de un cliente para procesar una página web:



El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere llamar a una función del servidor (añadir cantidad material, crear ubicación, etc.):



6. Módulo de salida de Picking

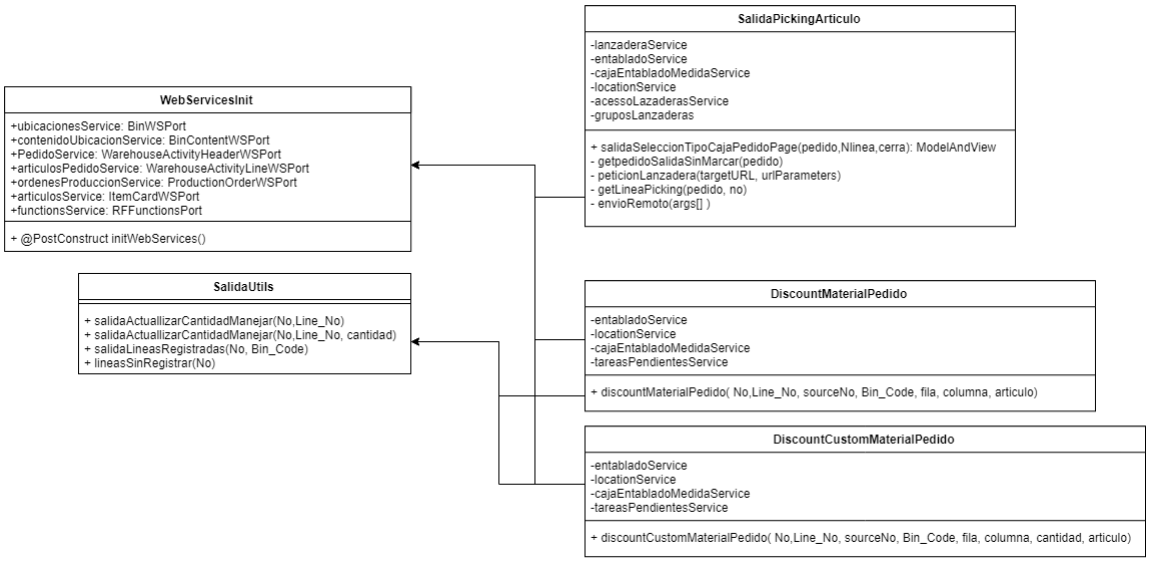
6.1. Funcionalidad

El módulo de entrada manual se encarga de:

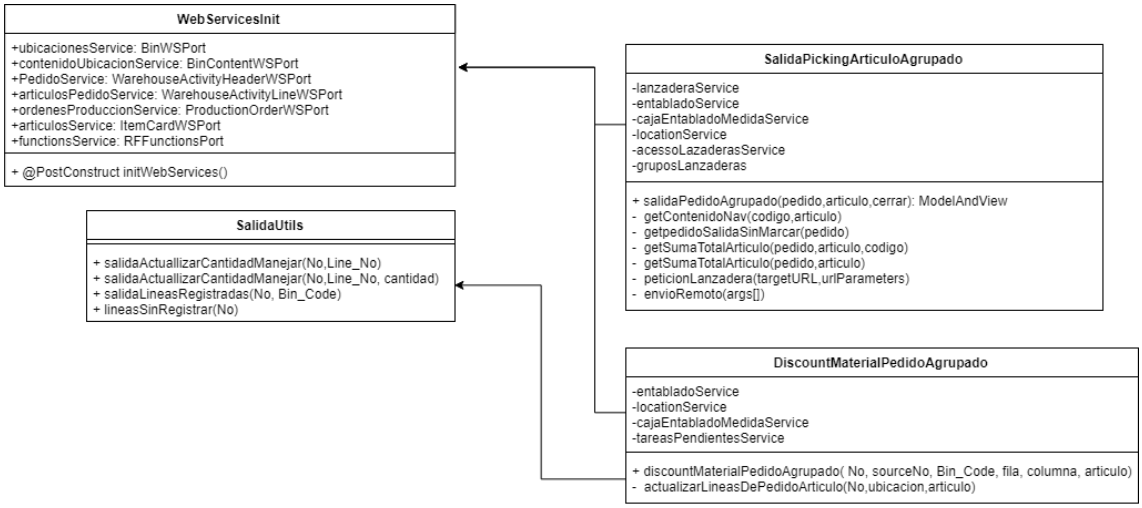
- Devolver paginas HTML relacionadas con picking
- Realizar búsquedas de picking
- Realizar líneas de picking
- Descontar varias líneas de picking que contengan el mismo artículo en el mismo entablado
- Descontar material de las ubicaciones
- Opción de borrar cajas si la ubicación se queda vacía

6.2. Diseño

El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la salida de un picking:



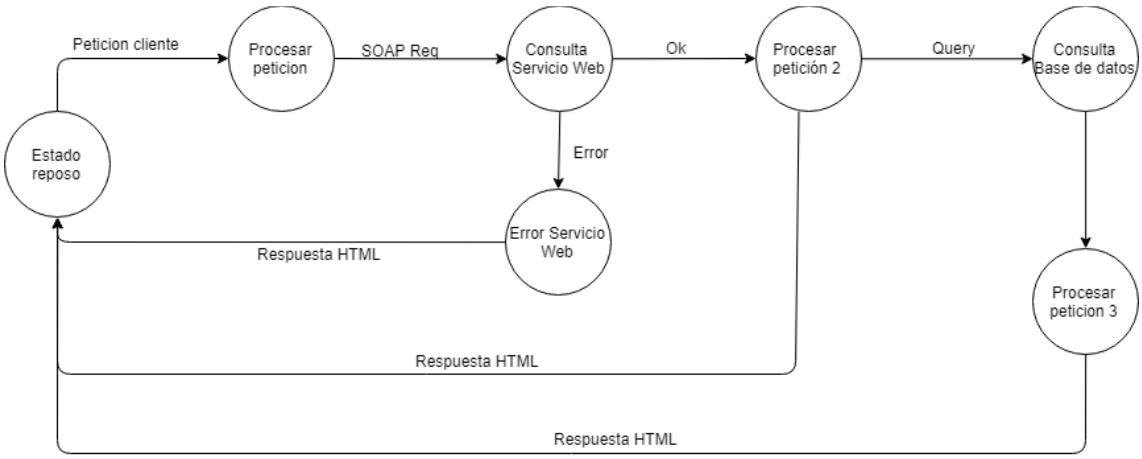
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la salida de un picking en el cual se suman las cantidades de aquellas líneas que contienen en mismo artículo en el mismo entablado:



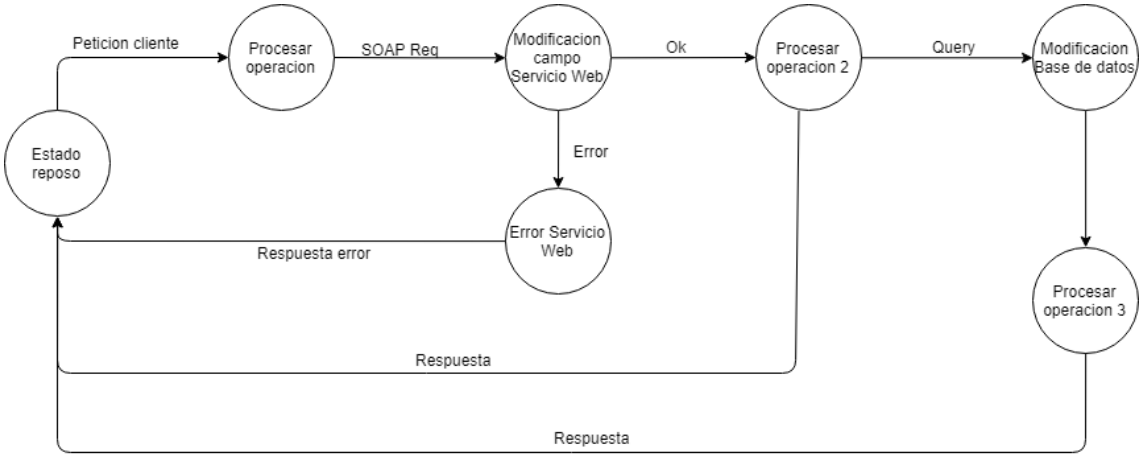
6.3. Transición entre estados

Durante su ciclo de vida, los cambios de estado son iniciados por el usuario utiliza la aplicación e invoca acciones sobre el servidor (mostrar una página, registrar una línea de pedido…) algunas transiciones provocan consultas al servicio web, estas a su vez pueden producir un error, otras sobre la base de datos MySQL.

El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una petición de página:

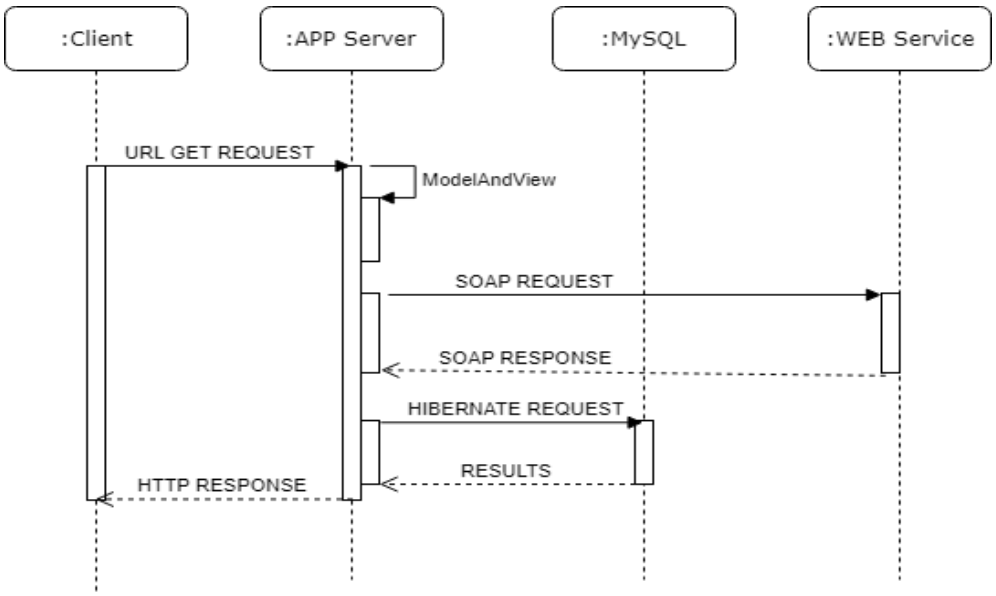


El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una función (registrar línea pedido, actualizar cantidad, etc.):

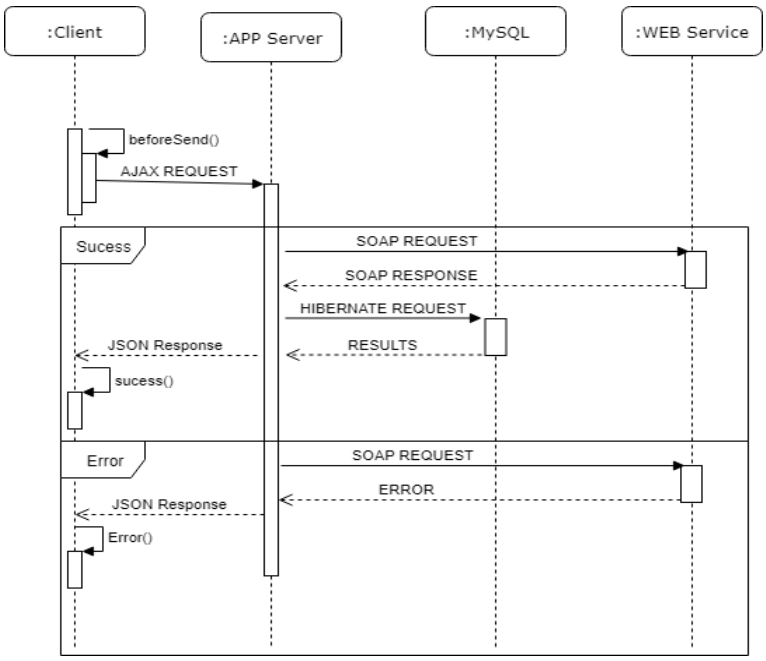


6.4. Diagrama de secuencia

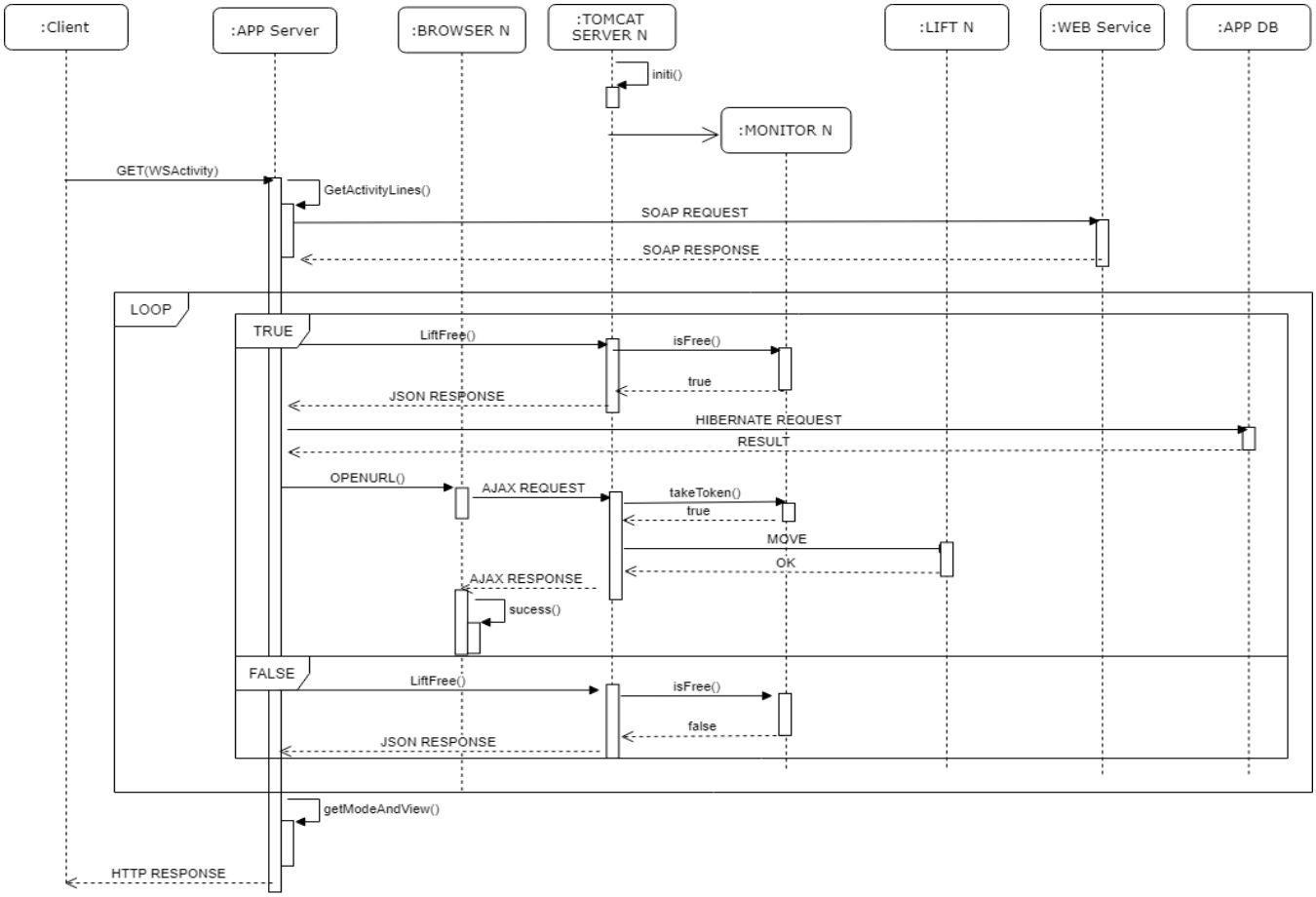
El siguiente diagrama muestra el envío de mensajes entre los componentes del sistema cuando se recibe la petición de un cliente para procesar una página web:



El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere llamar a una función del servidor:



El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere procesar una línea de picking en cada una de las lanzaderas implicadas:


:Client
:APP Server
:BROWSER N
:TOMCAT SERVER N
:LIFT N
:WEB Service
:APP DB
initi()
:MONITOR N
GET(WSActivity)
GetActivityLines()
SOAP REQUEST
SOAP RESPONSE
LOOP
TRUE
LiftFree()
isFree()
true
JSON RESPONSE
HIBERNATE REQUEST
RESULT
OPENURL()
AJAX REQUEST
takeToken()
true
MOVE
OK
AJAX RESPONSE
sucess()
FALSE
LiftFree()
isFree()
false
JSON RESPONSE
getModeAndView()
HTTP RESPONSE

7. Módulo de salida de manual

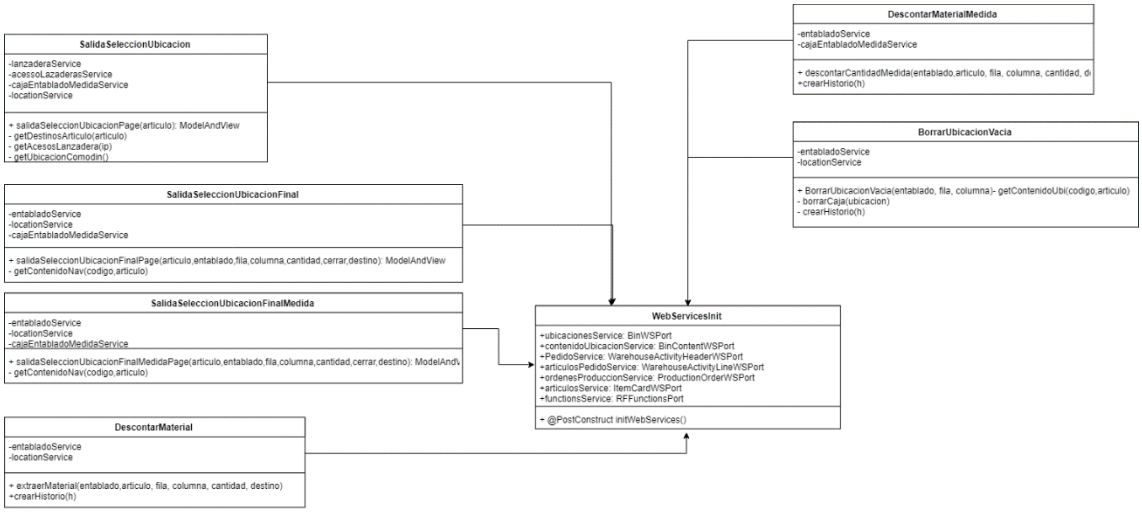
7.1. Funcionalidad

El módulo de entrada manual se encarga de:

- Devolver paginas HTML relacionadas con salidas manuales
- Realizar búsquedas de artículos dentro de una lanzadera
- Descontar material de las ubicaciones
- Opción de borrar cajas si la ubicación se queda vacía

7.2. Diseño

El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en la salida de material de manera manual:

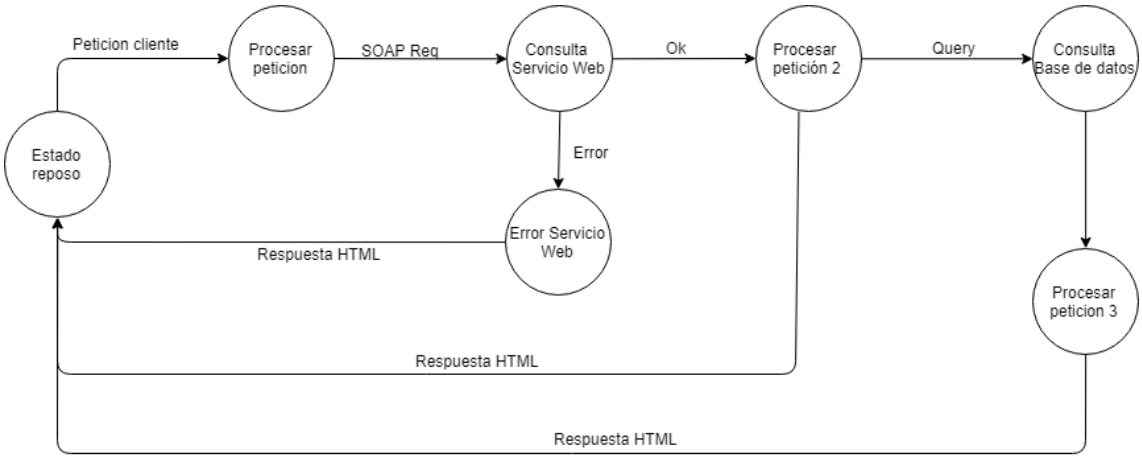


7.3. Transición entre estados

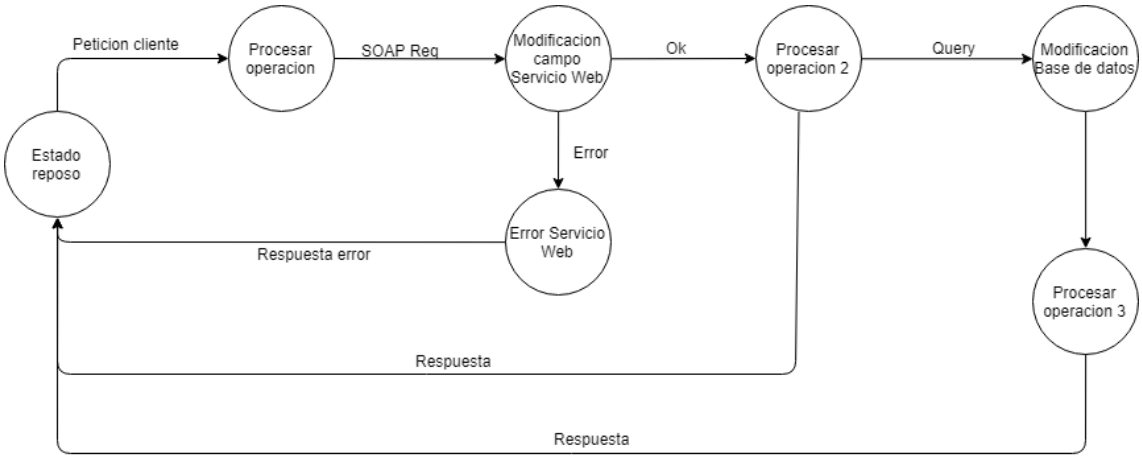
Durante su ciclo de vida, los cambios de estado son iniciados por el usuario utiliza la aplicación e invoca acciones sobre el servidor (mostrar una página, registrar una línea de pedido…) algunas transiciones provocan consultas al

servicio web, estas a su vez pueden producir un error, otras sobre la base de datos MySQL.

El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una petición de página:

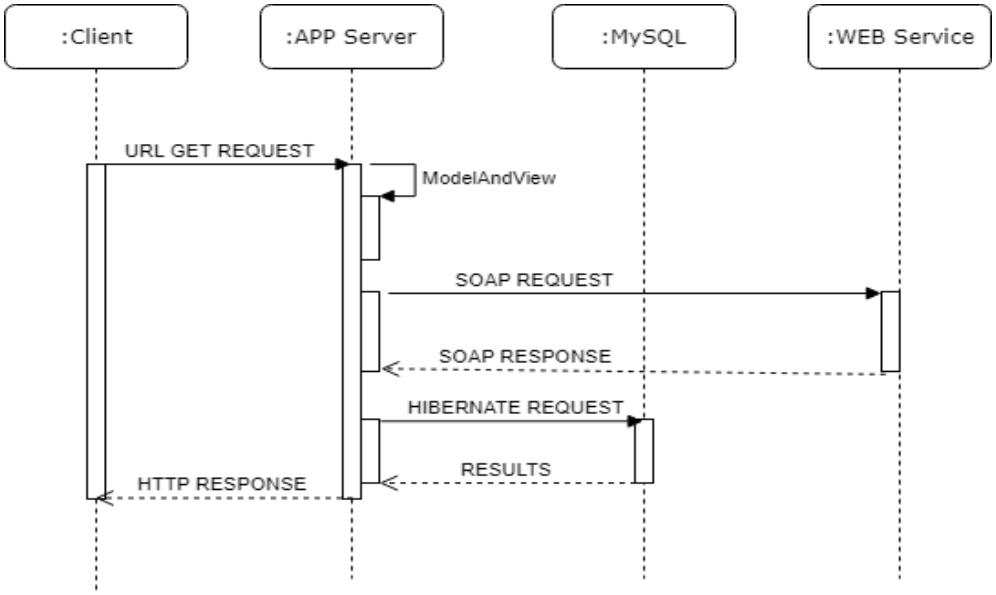


El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una función (registrar línea pedido, actualizar cantidad, etc.):

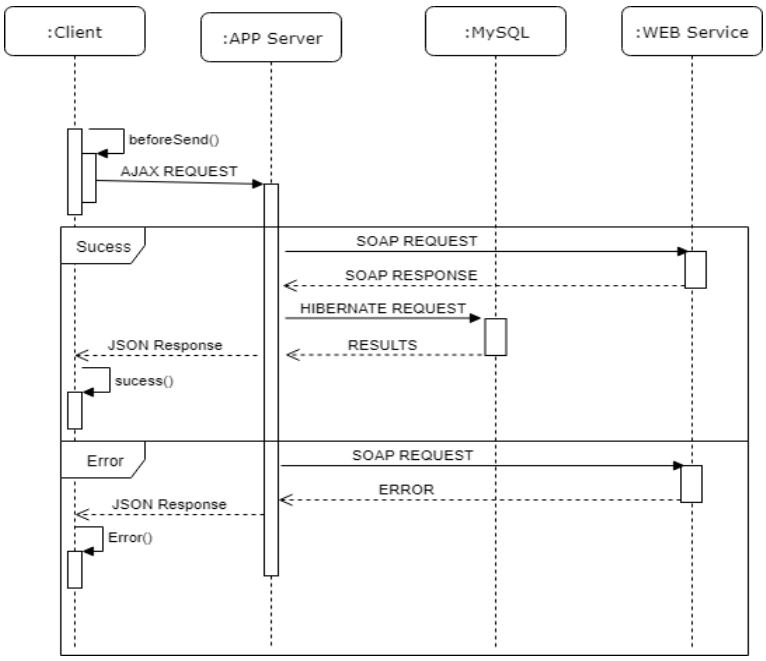


7.4. Diagrama de secuencia

El siguiente diagrama muestra el envío de mensajes entre los componentes del sistema cuando se recibe la petición de un cliente para procesar una página web:



El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere llamar a una función del servidor:



8. Módulo de consulta

8.1. Funcionalidad

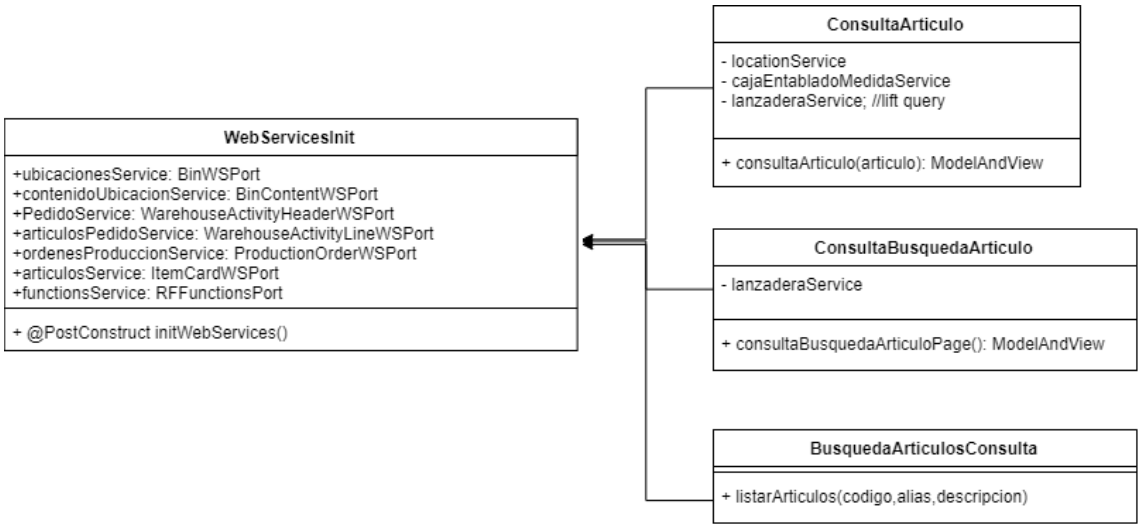
El módulo de entrada manual se encarga de:

- Devolver paginas HTML relacionadas con consultas
- Realizar búsquedas de artículos
- Obtener información de las ubicaciones de los artículos
- Consultar la ubicación de un articulo
- Consultar la ubicación de un entablado
- Buscar los entablados de la lanzadera
- Consulta de los contenidos de los entablados
- Consulta del historial de un entablado
- Borrar cajas vacías
- Traer entablado desde consulta entablado
- Mostrar cajas vacías

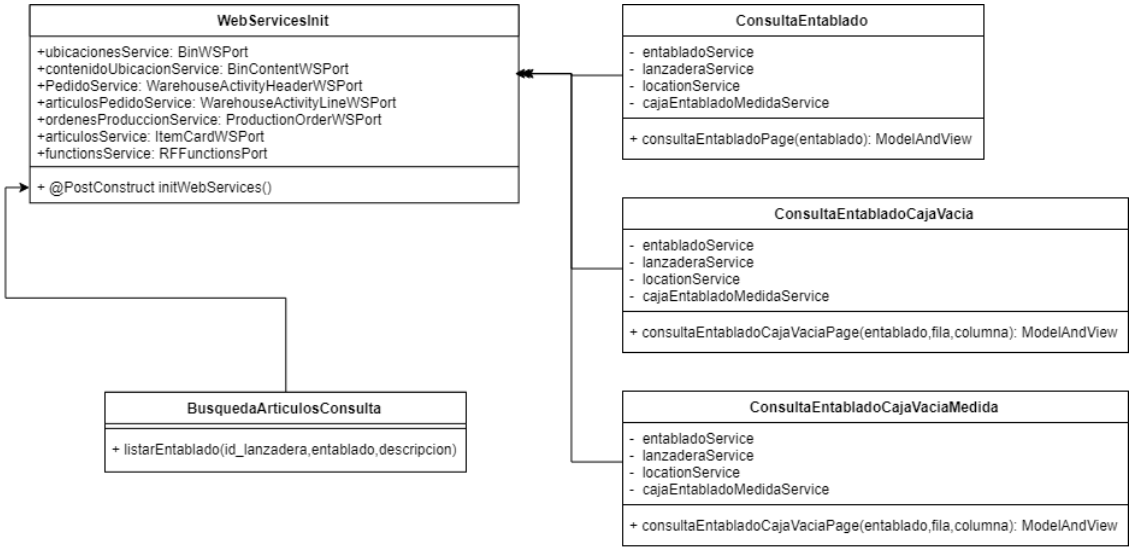
- Mostrar lanzaderas disponibles
- Mostrar contenidos lanzaderas

8.2. Diseño

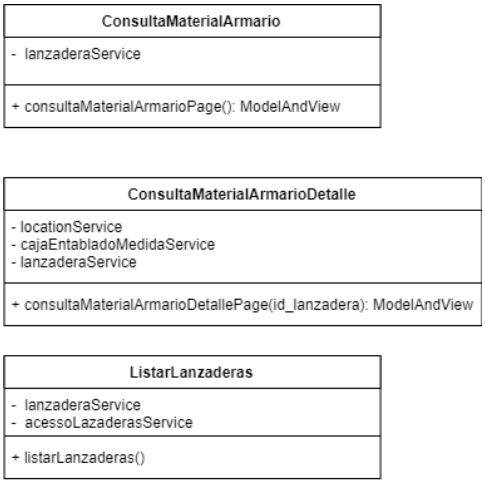
El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en consulta de un artículo:



El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en consulta del contenido de un entablado:



El siguiente diagrama de clases muestra las entidades de la capa de negocio que participan en consulta de material en las lanzaderas:

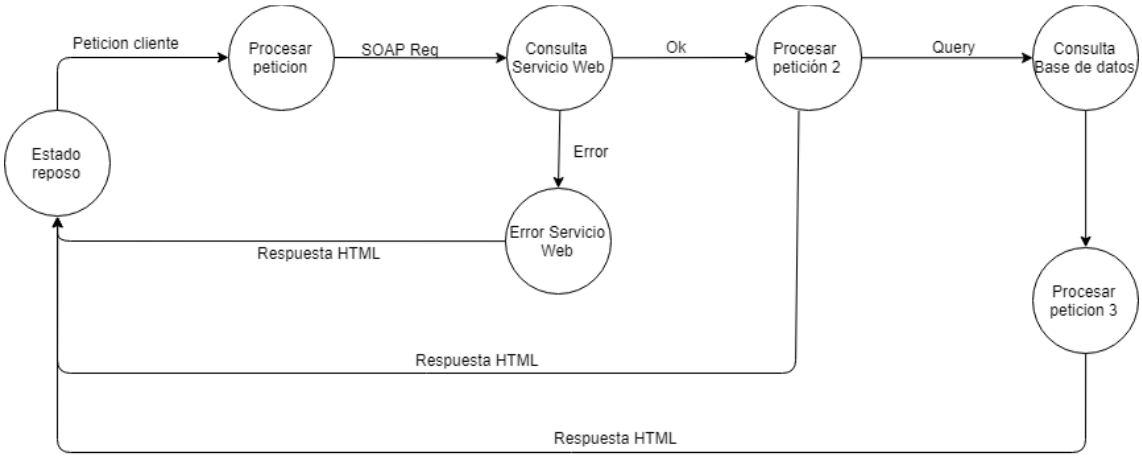


8.3. Transición entre estados

Durante su ciclo de vida, los cambios de estado son iniciados por el usuario utiliza la aplicación e invoca acciones sobre el servidor (mostrar una página,

registrar una línea de pedido…) algunas transiciones provocan consultas al servicio web, estas a su vez pueden producir un error, otras sobre la base de datos MySQL.

El siguiente diagrama de estados muestra los estados por los que pasa el sistema cuando un usuario realiza una petición de página:



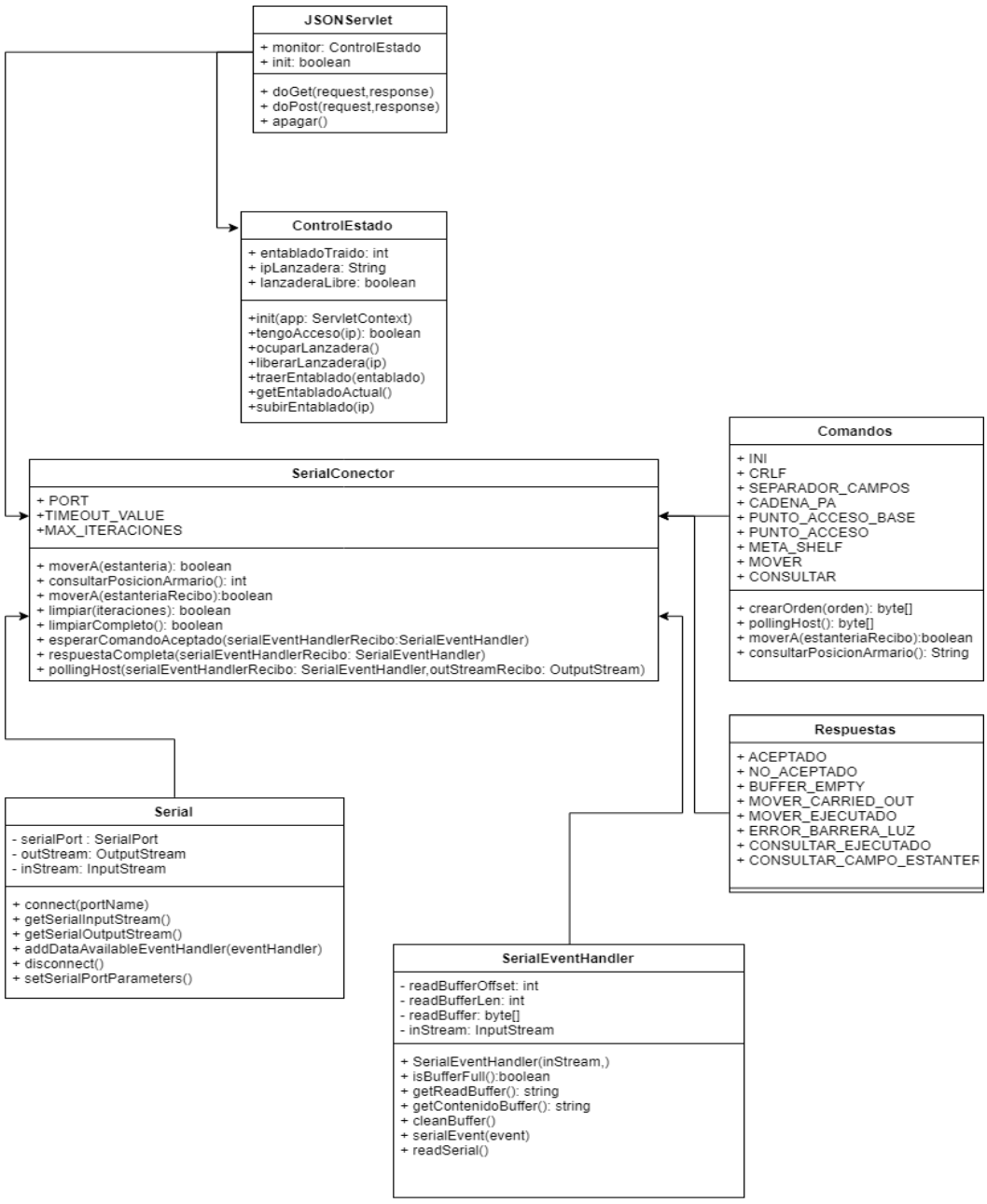
9. Módulo de comunicación con los armarios

9.1. Funcionalidad

El módulo de comunicación con los armarios se encarga de establecer la comunicación entre los usuarios y el armario físico.

9.2. Diseño

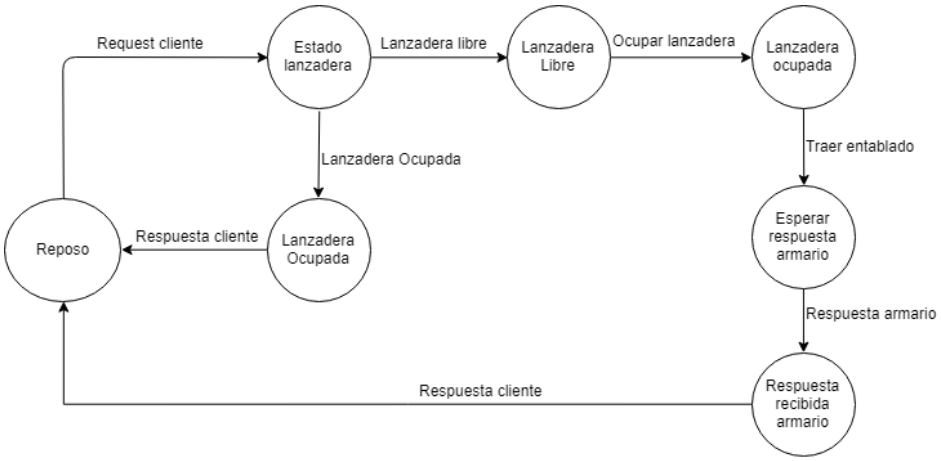
El siguiente diagrama de clases muestra las que participan en la gestión de la comunicación del armario físico:



9.3. Transición entre estados

Durante su ciclo de vida, los cambios de estado son iniciados por los usuarios que utilizan la aplicación e invocan acciones sobres los servidores de armarios (consultar ubicación, traer entablado…) estas provocan cambios de estado sobre el servidor de armarios, cada armario posee un token que simboliza si un armario está siendo ocupado o no, solo el propio dispositivo de la lanzadera se salta esta regla mediante su dirección IP.

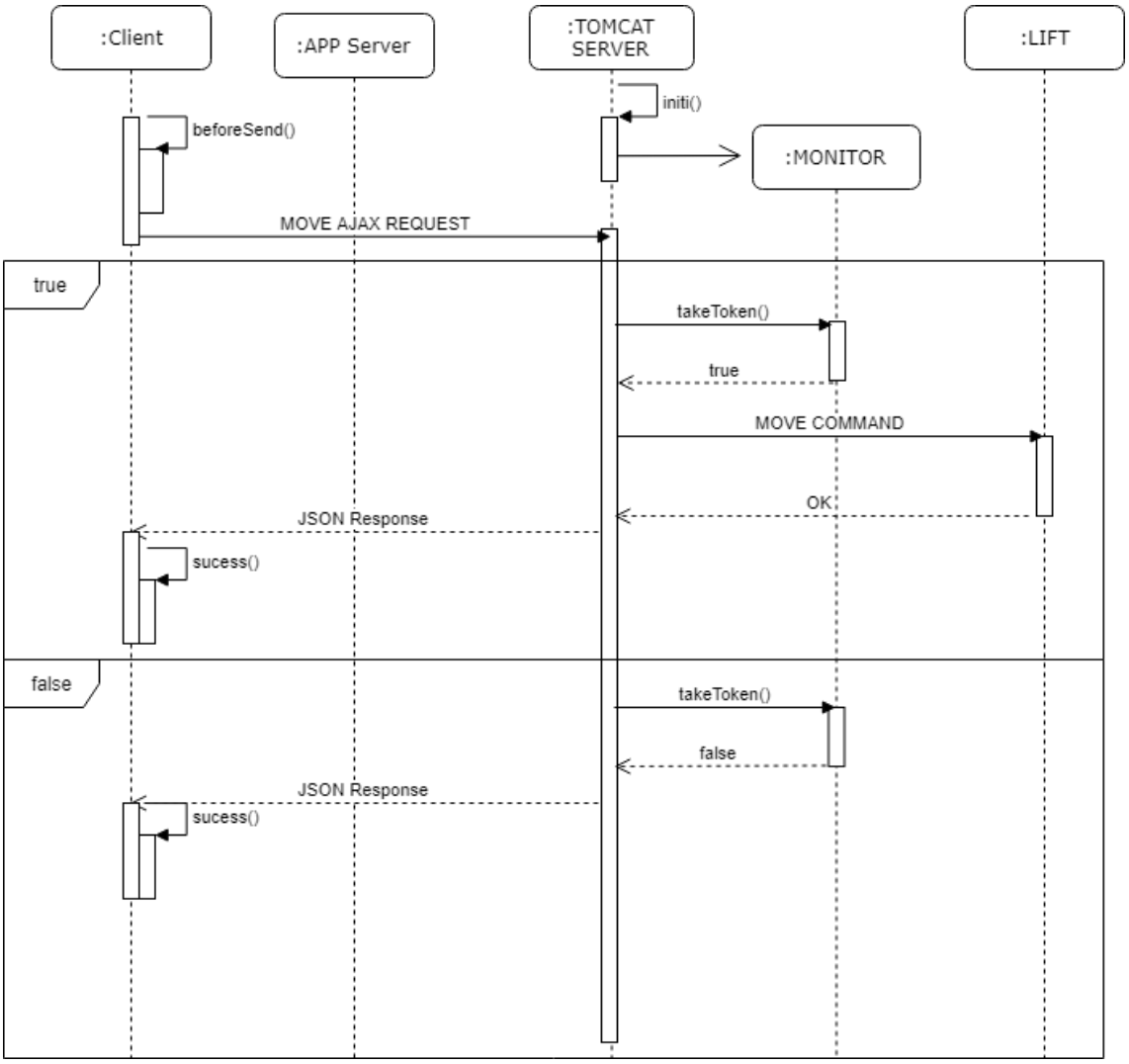
El siguiente diagrama de estados muestra los estados por los que pasa el servidor de la lanzadera cuando recibe una petición de movimiento por parte de un cliente:



El siguiente diagrama de estados muestra los estados por los que pasa el servidor de la lanzadera cuando recibe una petición de liberación por parte del cliente que está usando el dispositivo Raspberry asociado a la lanzadera:

9.4. Diagrama de secuencia

El siguiente diagrama muestra él envió de mensajes entre los componentes del sistema cuando el cliente quiere realizar un movimiento sobre un armario físico:



10. Ficheros de Configuración de la Aplicación

La siguiente tabla muestra todos los ficheros de configuración de la aplicación. Todos ellos van empaquetados en el fichero de despliegue (**.war**).

Nombre del fichero	Descripción
Pom.xml	Este fichero es utilizado por Maven. Contiene la declaración de todas las dependencias (librerías), scripts para generar stubs de los servicios Web, etc.
***.wsdl**	Es la definición de un servicio web consumido por la aplicación (Ej: obtener una lista de articulo). Estos ficheros son utilizados por Maven para generar las clases de los stubs en Java.
Web.xml	Proporciona información de configuración y de despliegue para los componentes web que conforman una aplicación web. En él se dirá que se utilize el framework spring.
spring-servlet.xml	Define los beans(conexión base de datos, Entidades Hibernate…) para la integración con spring.

10.1. Ficheros WSDL

La siguiente tabla muestra todos los ficheros de definición de servicios web usados por la aplicación para la conexión con el servicio web de NAV:

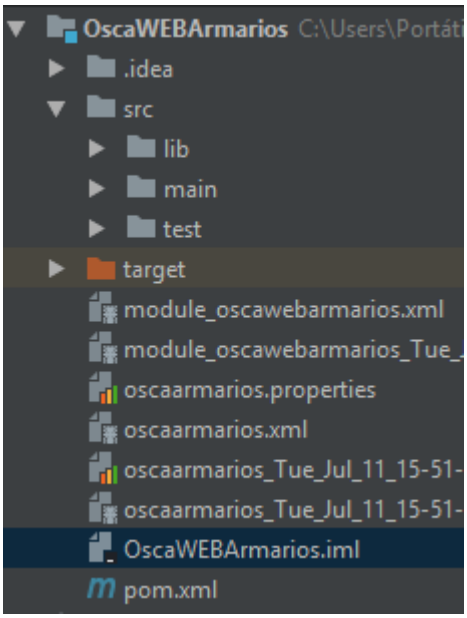
Nombre del fichero	Descripción
Bin_Content_WS.xml	Definición del servicio web usado para obtener manipular información acerca del contenido de los entablados
Bin_WS.xml	Definición del servicio web usado para manipular información acerca de los entablados
Employee_Login.xml	Definición del servicio web usado para obtener información acerca del login de los empleados de almacén.
Employee_WS.xml	Definición del servicio web usado para obtener información acerca de los empleados del almacén
Item_Card_WS.xml	Definición del servicio web usado para obtener información acerca del artículo de la empresa
RF_Functions.xml	Definición del servicio web usado para realizar funciones sobre el ERP Dynamics Nav, ofrece funciones para el registro de picking, descripción artículo, etc.

Warehouse_Activity_Header_WS.xml	Definición del servicio web usado para realizar consultas sobre las cabeceras de la actividad del almacén, obtención de ubicaciones, picking.
Warehouse_Activity_Line_WS.xml	Definición del servicio web usado para realizar consultas sobre las líneas de la actividad del almacén, obtención de líneas de ubicaciones, líneas de picking.

11. Convenciones del Desarrollo

11.1. Estructura del Proyecto

La aplicación se diseña siguiendo la decisión del estándar de J2EE. La estructura del proyecto en IntelliJ IDEA esta creada con el arquetipo de aplicaciones web de Maven, que se muestra a continuación:

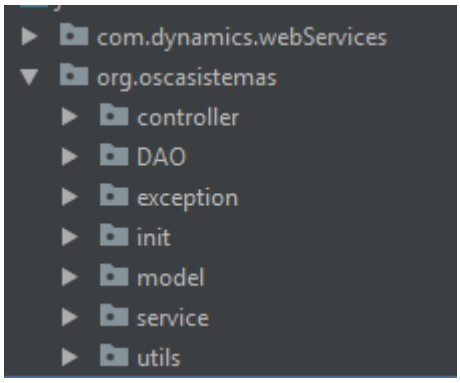


11.2. Definición de los paquetes de Aplicación Web

El proyecto (IntelliJ IDEA) tiene, por un lado, las clases y recursos que corresponden a la implementación de la aplicación (**src/main**), por otro las clases y recursos de las pruebas unitarias (**src/test**) y por otro aquellas librerías que no están disponibles en Maven (**src/lib**). El fichero pom.xml

contiene la definición de las dependencias, targets y demás parámetros utilizados por Maven.

Los paquetes del proyecto están organizados de la siguiente manera:



La ruta **src/main/java** contiene el codigo de la capa de servicios, persistencia e integración de la aplicación, dentro de esta ruta se encuentra el paquete **webServices** representa la capa de integración con los servicios web, contiene todas las clases generadas de manera automática mediante JAVA-WS. Por otra parte, el paquete **Oscasistemas** contiene las clases que representan la parte de la capa de negocio y de persistencia, dentro de este paquete se encuentra.

La carpeta **controller** que contiene todas las clases de la capa de servicios, la carpeta **DAO** contiene las clases e interfaces para realizar consultas CRUD sobre la base de datos, la carpeta **exception** se encarga de capturas las excepciones de la aplicación y mostrárselas al usuario por pantalla, por otro lado la carpeta **init** contiene las clases para inicializar el servidor, la carpeta **model** representa la capa de persistencia de la aplicación, dentro de ella se definen todos los *entities* (objetos que van a ser mapeados de la base de datos).La carpeta service contiene aquellas clases e interfaces para comunicar la capa de persistencia con la capa de presentación. La carpeta **utils** contiene funciones auxiliares estáticas.

11.3. Nombre de Clases, Metodos y Variables

En general, se adoptan todos los estándares y convenciones definidos para el lenguaje Java. Los nombres de las clases, métodos y variables deben ser siempre castellanos o castellanizados, evitando caracteres especiales como ñ, tildes, etc. Se admite el uso de prefijos o sufijos en ingles a efectos de homogeneizar la nomenclatura de clases, propiedades y métodos para adecuarlos a la tecnología especifica que se ha decidido utilizar. Por ejemplo, usar "get", "set" o "is" para los beans.

Los nombres de las clases corresponden a sustantivos y son descriptivos de la función dentro del modelo de objetos de la aplicación. En algunos casos también se añade un sufijo que especifica su papel papel dentro de la jerarquía de capa. Por ejemplo: **CajasEntabladoService**, **LanzaderasService**, **etc**.

Los nombres de los métodos deben ser siembre verbos. Si se trata de nombres compuestos por variar palabras, la primera letra de cada palabra es una mayúscula. Por ejemplo: **setCaja(...), updateCaja(...), etc**.

Los nombres de las variables de instancia y variables locales no pueden comenzar con "_" o "$". Deben ser breves y comunicar el significado de forma evidente. Se evitaran las variables de un carácter, con la sola excepción de las variables de las cláusulas **catch()** y los contadores temporales de los bucles (**e, i ,j, k, m**). Los nombres de las variables no tendrán ningún prefijo que indique el tipo de objeto u otro clasificador. Los nombres de constantes van en mayúsculas y con un "_" para separar las palabras (por ejemplo: **CANTIDAD_MAXIMA**).

11.4. Gestión de Excepciones

Las excepciones son errores que ocurren durante la ejecución de la aplicación y afectan el flujo normal de funcionamiento. Java ofrece una infraestructura robusta y basada en objetos para gestionar los diferentes escenarios de excepciones.

En general, cuando se produce una excepción, la aplicación debe tratarla, propagarla o generar una traza en el fichero log. Si el bloque de código donde ocurre la excepción puede tratarla, entonces deberá hacerlo. En caso contrario, por norma general, las excepciones serán propagadas y tratadas en los niveles superiores.

Nunca deben usarse clausulas try/catch vacías. Las clases de excepciones específicas de la aplicación seguirán las convenciones definidas previamente para el nombrado de clases y además deben terminar con el sufijo "Exception". Por ejemplo "GenericException".

Finalmente, debe tenerse en cuenta que las excepciones son caras de gestionar y que deben usarse razonablemente. Esto significa que debe analizarse si en algunos casos es más simple retornar un *flag* booleano para indicar que una operación se realizó con éxito o no- en lugar de lanzar y propagar una excepción.

11.5. Plantillas y Ensamblado de las paginas

Las páginas HTML que representan la funcionalidad y navegación de la aplicación para los usuarios finales serán generadas a partir de una o varias plantillas. Así, cada página únicamente cambia una porción—normalmente el cuerpo central—del contenido mostrado al usuario.

Esta forma de trabajo permite identificar y definir fragmentos que pueden ser ensamblados dinámicamente en tiempo de ejecución para crear las páginas HTML. El uso de fragmentos permite reducir la duplicación de elementos comunes a todas las páginas y trabajar con plantillas que pueden reutilizarse.

Más concretamente, las páginas de la aplicación incluirán los siguientes elementos: una cabecera de común, un pie, un menú de opciones y un cuerpo.

11.6. Convenciones para los Objetos de la Base de datos

A continuación, se mencionan las principales convenciones que se adoptan en la creación de los objetos de la base de datos.

En general, todos los nombres constarán de una sola palabra, pero se admite el símbolo "_" para unir las palabras en nombres compuestos. Todos los nombres se escribirán en castellano o serán castellanizados y utilizarán caracteres alfabéticos o dígitos (0-9). No se utilizarán vocales acentuadas ni caracteres propios de un idioma (como 'ñ', apóstrofes...) ni artículos, conjunciones, ni partículas de unión.

Para las tablas y vistas deben utilizarse nombres en singular. En el caso de tablas o vistas que procedan de una relación **M:N** entre dos tablas padre, el nombre será compuesto de la denominación de las tablas padre, intentando dar preferencia a la entidad padre más relevante o con más peso en la relación. En el caso de una vista que proceda de una tabla-base, se nombrará a partir de la tabla de origen, precedida del prefijo "**V_**". Se utilizarán numerales en caso necesario. Por ejemplo: "**V_Persona1**", "**V_Persona2**".

Si una vista se ha creado para su uso específico desde un esquema concreto, se colocará como sufijo el identificador de dicho esquema. Ejemplo: si se crea una vista a una tabla '**Persona**' para su uso desde un esquema llamado CTLXXADM, se nombraría como "**V_PERSONA_CTL**"

En el caso de entidades auxiliares que se encuentren en otro esquema (como tablas de propósito general), se procede de la siguiente manera:

- Si se utilizan la mayor parte de los atributos y tuplas de la entidad auxiliar, se referenciará ésta mediante un sinónimo, el cual mantendrá el nombre de la entidad original (aunque se encuentre en plural)
- Si se utiliza un subconjunto reducido de los atributos y tuplas de la entidad auxiliar, se accederá a ella mediante una vista. En ese caso, el nombre seguirá los mismos convenios que para tablas.

Los nombres de las secuencias comienzan siempre por "**S**_". Si las utiliza una sola tabla, llevarán a continuación el nombre de ésta. Si las utilizan varios objetos o tablas, tendrán un nombre compuesto "_".

Los nombres de los procedimientos almacenados comienzan por P_. La siguiente palabra debe ser un verbo infinitivo indicativo de su propósito principal (Ejemplo: "**P_ACTUALIZAR**...", "**P_CALCULAR...**"). Los nombres de las funciones comienzan por "**F**_". Como antes, la siguiente palabra será un verbo infinitivo indicativo de su propósito principal.

Los paquetes tienen un formato libre, pero sus procedimientos y funciones componentes respetarán las normas anteriores.

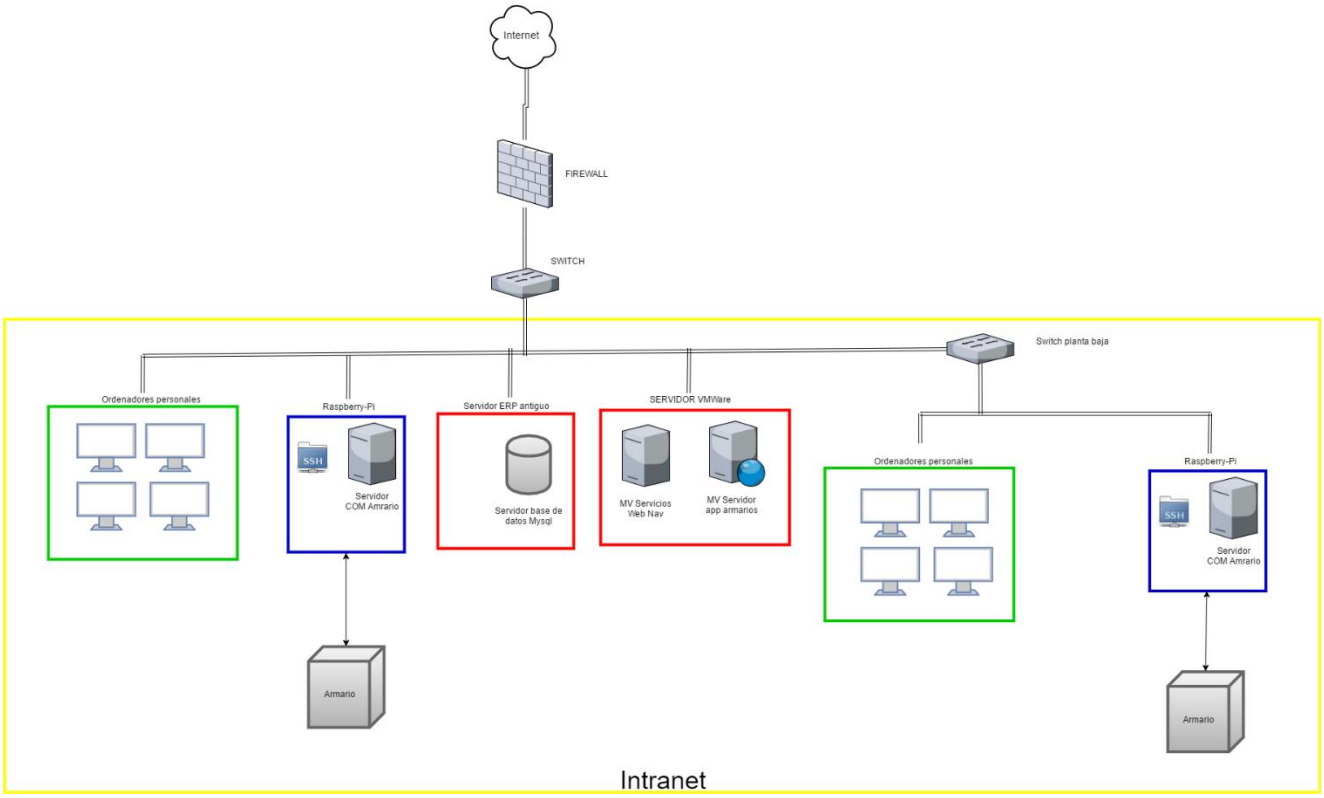
Las claves primarias deben nombrarse como la tabla y añadiendo el sufijo "**_PK**". Los índices únicos también se nombran como la tabla y añadiendo el sufijo "**_U99**", donde '**99**' son dos posiciones para numeral correlativo. Los índices no únicos se nombran como la tabla y añadiendo el sufijo "**_I99**", donde '**99**' son dos posiciones para numeral correlativo.

Las claves ajenas se nombran como la tabla en la que se define la restricción de clave ajena y se añade el sufijo "**_R99**", donde '**99**' son dos posiciones para numeral correlativo.

12. Arquitectura física y despliegue

12.1. Topología de la plataforma

El siguiente diagrama muestra la configuración del entorno para el despliegue de la aplicación:



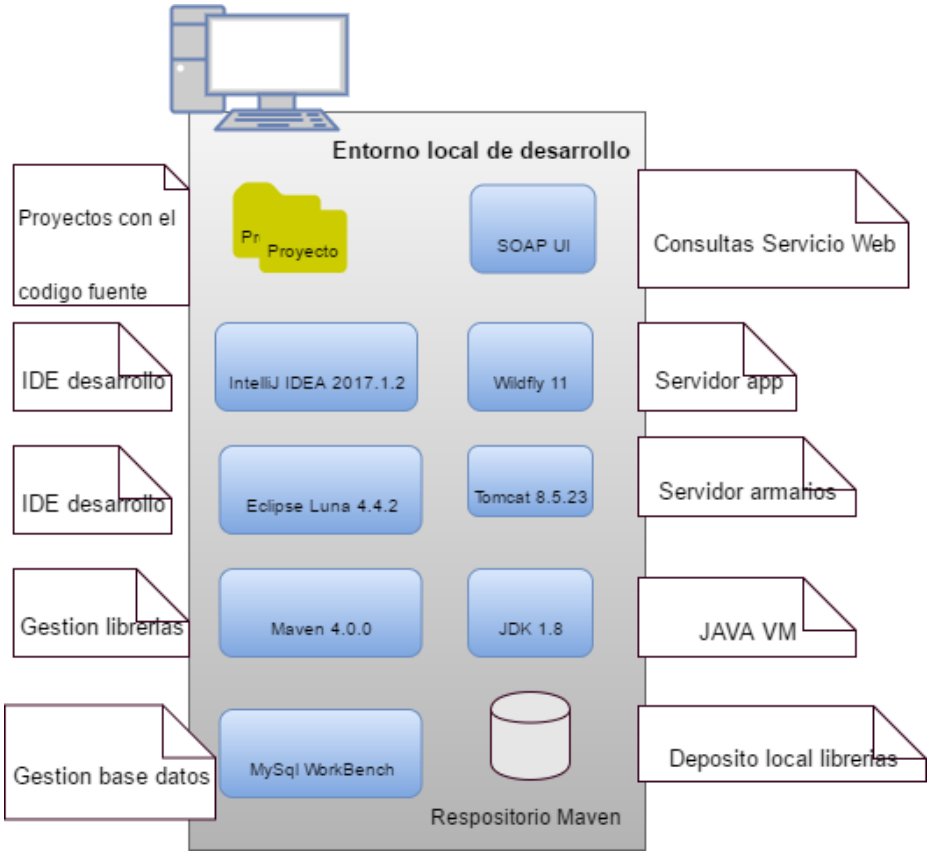
Hay que destacar los entornos de ejecución sobre los se basa el desarrollo e implementación del sistema:

- Sistema operativo: Ubuntu Mate, Debian para el servidor de la app
- Servidor de aplicaciones para los despliegues de las aplicaciones y servicios: Wildfly 11.X.X, Tomcat 8.X.X.
- Base de datos Oracle MySQL
- Servicio Web Nav
- Armario Hänel Lean-Lift

13. Entorno de desarrollo

13.1. Herramientas

El entorno de desarrollo tiene una cara: el ordenador individual con las herramientas utilizadas por el desarrollador.



El puesto de desarrollo cuenta con una instalación completa de las siguientes herramientas: JDK, intelliJ IDEA, Eclipse, Wildfly, Tomcat, Maven, MySql Workbench, SOAP UI y Notedpad++.

A continuación, se describen brevemente las principales herramientas mencionadas en el apartado anterior:

Maven es una herramienta open-source que simplifica los procesos de construcción y gestión de los ficheros de instalación de las aplicaciones y servicios (**.war**), a partir del código fuente. Entre las principales características de Maven, destacan las siguientes:

- Estandarización de la estructura interna del proyecto (carpetas de código fuente, ubicación de ficheros)
- Simplificación de la gestión de las librerías comunes (esto se denomina "*gestión de dependencias*"). Las librerías se descargan desde *repositorios públicos* accesibles por Internet.
- Automatización de tareas rutinarias, como la compilación y generación de ficheros de despliegue (**.war**)
- Cada proyecto tiene un fichero *pom.xml* con la configuración utilizada por Maven.
- Se invoca por línea de comandos o desde el menú de IntelliJ IDEA y Eclipse.

13.2. Versiones de las herramientas

Herramienta	Versión	Descripción
Java Development Kit (JDK)	**1.8.0_131**	El la máquina virtual de Java y el compilador. Distribución libre y gratuita
IntelliJ IDEA	**2017.1.2**	IDE de desarrollo usado para la aplicación WEB. Uso gratuito con licencia de estudiante.
Eclipse	**Luna Service Release 2 (4.4.2)**	IDE de desarrollo usado para el servidor de armarios. Distribución libre y gratuita.
MySQL Workbench	**6.3**	Programa para la gestión y

		mantenimiento de bases de datos MySQL
Apache Maven	**4.0.0**	Automatizar la gestión de las librerías utilizadas por las aplicaciones (dependencias). Distribución libre y gratuita.
Notepad++	**7.2.2**	Editor de textos (XML, HTML, ficheros configuración, etc.). Distribución libre y gratuita
Google chorme	**64.0.3282.186**	Navegador de internet. Distribución libre y gratuita
Microsoft word	**365**	Editor de textos para la documentación
Wildfly VM	**11.0.0-3**	Máquina virtual que contiene el servidor de aplicaciones java wildfly preconfigurado. Distribución libre y gratuita. Descargar
SOAP UI	**5.4**	Aplicación consultas servicio web mediante SOAP. Distribución libre y gratuita.
Apache Tomcat	**8.5.23**	Servidor de aplicaciones ligero para java. Distribución libre y gratuita.
Ubuntu Mate	**1.16.1**	Sistema operativo para Raspberry basado en Ubuntu. Distribución libre y gratuita.