|  |
| --- |
| Aarón Hernández Álvarez |

**HELPDESK-NOW**

aRQUITECTURA DE sOFTWARE

2022

Contenido

[Introducción 3](#_Toc98080718)

[Descripción del proyecto 4](#_Toc98080719)

[Primera Práctica 5](#_Toc98080720)

[Diagrama de Casos de Uso 5](#_Toc98080721)

[Funcionalidades de cada actor 6](#_Toc98080722)

[Usuario 6](#_Toc98080723)

[Cliente 6](#_Toc98080724)

[Agente 6](#_Toc98080725)

[Listado de CU 7](#_Toc98080726)

[Usuario 7](#_Toc98080727)

[Cliente 7](#_Toc98080728)

[Agente 7](#_Toc98080729)

[Especificaciones de casos de uso 8](#_Toc98080730)

[Diagrama de Clases 10](#_Toc98080731)

[Control 11](#_Toc98080732)

[Modelo 12](#_Toc98080733)

[ANEXO I Comentarios del alumno relativos a las prácticas 14](#_Toc98080734)

[Práctica 1 14](#_Toc98080735)

[Sobre la inclusión de Bases de Datos 14](#_Toc98080736)

[Sobre la implementación del patrón Two-Step View 14](#_Toc98080737)

[ANEXO II Gráficos empleados en la memoria 15](#_Toc98080738)

# Introducción

La gestión de incidencias dentro del devenir diario de una empresa puede llegar a ser crucial pasa el correcto funcionamiento los procesos productivos de ésta. Multitud de problemas que pueden abarcar desde los más simples, como una solicitud de instalación de software, hasta otros más complejos burocráticamente, como por ejemplo la reposición completa de un equipo, pueden acabar causando problemas en el desarrollo si no son tratados en tiempo y forma.

Tanto es así que, en el marco de la Tecnologías de la Información, el sector de soporte ha experimentado un aumento reseñable en las últimas décadas, con corporaciones cuya principal actividad productiva es la prestación de servicios de soporte a terceros, en lo que se conoce en la jerga técnica como *outsourcing*. Corporaciones como Salesforce, Microsoft o Wix ya disponen de secciones dedicadas específicamente a este ámbito para dar cobertura a los clientes ante posibles incidencias que puedan darse con el uso habitual de sus productos.

En este proyecto, se planteará el desarrollo de una herramienta de gestión de incidencias de forma escalada, aplicando los principios de diseño aprendidos en la asignatura y durante el transcurso de la carrera, pasando de una primera versión hasta la definitiva.

# Descripción del proyecto

El proyecto a desarrollar se trata de una aplicación de tipo *service-now*, esto es, una aplicación para clientes y agentes de soporte de una pequeña entidad. En su primera versión, tendrá como objetivo facilitar la gestión de incidencias, permitiendo la apertura, la actualización y el cierre de éstas, ampliándose estas funciones a lo largo del desarrollo del proyecto.

En esta primera versión destacaremos dos elementos primarios, que serán el usuario, quien interactúa con el elemento base y la incidencia o *ticket*, que es el propio elemento base. Como ya se ha comentado, ambos se irán ampliando en su composición a lo largo del desarrollo, permitiendo ampliar las capacidades y funcionalidades preestablecidas originalmente.

# Primera Práctica

Al tratarse de un primer prototipo, los usuarios e incidencias con los que se podrá interactuar serán generados de forma dinámica en memoria, pero se dejará establecida la infraestructura necesaria para realizar la conversión a un sistema convencional de BB.DD. en posteriores entregas. Siguiendo en la misma dinámica, la interfaz gráfica se revisará en cada iteración para cumplimentar los datos incluidos en la capa de negocio, así como facilitar y mejora la Experiencia de Usuario en general.

## Diagrama de Casos de Uso

Como se ha comentado, en esta primera versión se implementarán los modelos básicos de la aplicación, permitiendo realizar un uso primitivo de ésta. Por ello, como se puede observar en la figura 1, para esta entrega se han definido los casos de uso nucleares de la aplicación.

|  |
| --- |
| **Figura 1**  *Diagrama de casos de uso, primera versión.*    *Nota: En gris los casos implementados en esta versión. Este diagrama será sujeto de revisiones y ampliaciones en futuras entregas.* |

## Funcionalidades de cada actor

Se resumen a continuación las funcionalidades propias para cada actor para esta versión:

### Usuario

* Se trata del actor principal, del cual generalizarán los otros dos descritos a continuación. Sus acciones son:
  + Todo usuario de la aplicación puede crear una cuenta y gestionarla.
  + Todo usuario puede crear una incidencia y actualizarla añadiendo comentarios o modificando su estado dentro de unos parámetros preestablecidos. En su caso podrá cerrarla.
  + Todo usuario puede crear y leer comentarios dentro de las incidencias.

### Cliente

Se trata de los usuarios externos al entorno de resolución de incidencias. Disponen de acceso limitado a la herramienta. Sus acciones son:

* + Todo cliente puede ver sus incidencias abiertas y su estado. Por herencia de *Usuario*, podrá cerrarla si lo ve necesario.
  + Todo cliente puede visualizar un reporte o informa de las actualizaciones de los tickets.

### Agente

Son los encargados de actualizar el estado de las incidencias conforme se obtenga algún progreso dentro de éstas. Sus acciones son:

* + Todo agente puede asignarse una incidencia del pool de incidencias abiertas.
  + Todo agente puede modificar el estado de una incidencia, pudiendo ponerla en Sin asignar, Asignada o En progreso. Por herencia de *Usuario*, podrá cerrarla si lo ve necesario.
  + Todo agente podrá rechazar una incidencia si ésta no cumple con los parámetros preestablecidos por la empresa.

## Listado de CU

Se detalla a continuación el listado de Casos de Uso con sus correspondientes ID:

### Usuario

* + CU01: Manage Ticket
  + CU0101: Manage Comment
  + CU0102: Change Ticket Status
  + CU02: Manage Account
  + CU0201: Create Account

### Cliente

* + CU03: View My Tickets
  + CU04: Check Latest Ticket Updates

### Agente

* + CU05: Auto Assign Ticket
  + CU06: Set Ticket Status
  + CU07: Reject Ticket

## Especificaciones de casos de uso

Se muestran a continuación los tres CU más relevantes de esta versión.

|  |  |  |  |
| --- | --- | --- | --- |
| **CU03** | **Realizar compra** | | |
| **Descripción** | El cliente va a realizar una compra de algún objeto | | |
| **Precondiciones** | Existe el objeto y consta como disponible en el inventario | | |
| **Flujo normal** | | | |
| 1 | El Cliente declara que quiere realizar una compra de un objeto | | |
| 2 | El Agente toma los datos del cliente necesarios para iniciar el proceso de compra | | |
| 3 | El Agente verifica que los datos del Cliente son correctos y cumplen la norma especificada para la transacción | | |
| 4 | El Agente informa del precio del objeto al Cliente y el cliente acepta: Variante 4a.  El Agente informa del precio del objeto al Cliente y el cliente rechaza: Variante 4b. | | |
| 5 | INCLUDE CU10 (Establecer Contrato) | | |
| 6 | Finaliza el proceso de compra. Se guarda un copia del contrato establecido en 5 y se da otra al cliente | | |
| **Variantes** | | | |
| 4a | El Cliente acepta el precio ofrecido | | |
|  | 4a.1 | Ir al paso 5 del flujo normal | |
| 4b. | El Cliente rechaza el precio ofrecido | | |
|  | 4b.1 | Finaliza el proceso de compra. No se generan copias. | |
| **Extensiones** | | | |
| - | - | | - |
| **Postcondiciones** | La compra se realiza correctamente | | |
| **Excepciones** | | | |
| 1 | El objeto no se encuentra disponible, aunque consta en el inventario. | | |
|  | 3.1 | Se informa al cliente de la situación de inexistencia del objeto. | |
|  | 3.2 | Finaliza el proceso de compra | |
| 3 | La verificación de los datos del cliente no resulta exitosa | | |
|  | 3.1 | Se informa al cliente del error | |
|  | 3.2 | Finaliza el proceso de compra | |

|  |  |  |  |
| --- | --- | --- | --- |
| **CU03** | **Realizar compra** | | |
| **Descripción** | El cliente va a realizar una compra de algún objeto | | |
| **Precondiciones** | Existe el objeto y consta como disponible en el inventario | | |
| **Flujo normal** | | | |
| 1 | El Cliente declara que quiere realizar una compra de un objeto | | |
| 2 | El Agente toma los datos del cliente necesarios para iniciar el proceso de compra | | |
| 3 | El Agente verifica que los datos del Cliente son correctos y cumplen la norma especificada para la transacción | | |
| 4 | El Agente informa del precio del objeto al Cliente y el cliente acepta: Variante 4a.  El Agente informa del precio del objeto al Cliente y el cliente rechaza: Variante 4b. | | |
| 5 | INCLUDE CU10 (Establecer Contrato) | | |
| 6 | Finaliza el proceso de compra. Se guarda un copia del contrato establecido en 5 y se da otra al cliente | | |
| **Variantes** | | | |
| 4a | El Cliente acepta el precio ofrecido | | |
|  | 4a.1 | Ir al paso 5 del flujo normal | |
| 4b. | El Cliente rechaza el precio ofrecido | | |
|  | 4b.1 | Finaliza el proceso de compra. No se generan copias. | |
| **Extensiones** | | | |
| - | - | | - |
| **Postcondiciones** | La compra se realiza correctamente | | |
| **Excepciones** | | | |
| 1 | El objeto no se encuentra disponible, aunque consta en el inventario. | | |
|  | 3.1 | Se informa al cliente de la situación de inexistencia del objeto. | |
|  | 3.2 | Finaliza el proceso de compra | |
| 3 | La verificación de los datos del cliente no resulta exitosa | | |
|  | 3.1 | Se informa al cliente del error | |
|  | 3.2 | Finaliza el proceso de compra | |

|  |  |  |  |
| --- | --- | --- | --- |
| **CU03** | **Realizar compra** | | |
| **Descripción** | El cliente va a realizar una compra de algún objeto | | |
| **Precondiciones** | Existe el objeto y consta como disponible en el inventario | | |
| **Flujo normal** | | | |
| 1 | El Cliente declara que quiere realizar una compra de un objeto | | |
| 2 | El Agente toma los datos del cliente necesarios para iniciar el proceso de compra | | |
| 3 | El Agente verifica que los datos del Cliente son correctos y cumplen la norma especificada para la transacción | | |
| 4 | El Agente informa del precio del objeto al Cliente y el cliente acepta: Variante 4a.  El Agente informa del precio del objeto al Cliente y el cliente rechaza: Variante 4b. | | |
| 5 | INCLUDE CU10 (Establecer Contrato) | | |
| 6 | Finaliza el proceso de compra. Se guarda un copia del contrato establecido en 5 y se da otra al cliente | | |
| **Variantes** | | | |
| 4a | El Cliente acepta el precio ofrecido | | |
|  | 4a.1 | Ir al paso 5 del flujo normal | |
| 4b. | El Cliente rechaza el precio ofrecido | | |
|  | 4b.1 | Finaliza el proceso de compra. No se generan copias. | |
| **Extensiones** | | | |
| - | - | | - |
| **Postcondiciones** | La compra se realiza correctamente | | |
| **Excepciones** | | | |
| 1 | El objeto no se encuentra disponible, aunque consta en el inventario. | | |
|  | 3.1 | Se informa al cliente de la situación de inexistencia del objeto. | |
|  | 3.2 | Finaliza el proceso de compra | |
| 3 | La verificación de los datos del cliente no resulta exitosa | | |
|  | 3.1 | Se informa al cliente del error | |
|  | 3.2 | Finaliza el proceso de compra | |

## Diagrama de Clases

Para gestionar todo el entorno se ha decidido implementar el código de la aplicación partiendo del modelo MVC, con tres grandes paquetes donde contendremos a las clases: Modelo, Vista y Control. Puesto que las vistas quedan implementadas con JSP’s que reciben los atributos de las HttpRequest durante el tiempo de navegación, este apartado se centrará mayormente en los paquetes de Control, ubicado en el proyecto *war* y Modelo, localizado en el proyecto *ejb* asociado.

### Control

Se muestra la estructura del paquete de Control en la Figura 2, abarcando los paquetes internos implicados.

|  |
| --- |
| **Figura 2**  *Diagrama de clases, paquete de control, primera versión.*    *Nota: Este diagrama será sujeto de revisiones y ampliaciones en futuras entregas. Las clases pueden ser sujeto de renombrado.* |

Dentro de éste se destacan las siguientes clases y sub-paquetes:

* + **control.FrontController:** Es la clase que implementa el patrón FrontController. Su función principal consiste en recuperar el comando transmitido desde la request, localizarlo en el árbol de paquetes de la aplicación, instanciarlo y llamar a su ejecución antes de finalizar el proceso de llamada.
  + **command.FrontCommand:** Clase abstracta de la que extienden todos los comandos de la aplicación. El método abstracto *execute()* procesa la request de cada una de sus instancias hijas en función a cómo se haya programado a nivel concreto.
  + **command.UnknownCommand:** Extensión del FrontCommand que resuelve aquellos llamamientos a comando no existentes.
  + **twoStepTrsnformation.XsltTranformer:** Clase integrada en el paquete de transformación que se encarga de integrar el patrón Two Step View recibiendo dos ficheros de plantilla y el texto original, convirtiéndolo en HTML listo para ser mostrado.

### Modelo

Se muestra la estructura del paquete de Modelo en la Figura 2, abarcando los paquetes internos implicados.

|  |
| --- |
| **Figura 3**  *Diagrama de clases, paquete de modelo, primera versión.*    *Nota: Este diagrama será sujeto de revisiones y ampliaciones en futuras entregas. Las clases pueden ser sujeto de renombrado.* |

En este caso, se destacan las siguientes clases:

* + **singletons.UsersDB:** Singleton encargado de inicializar los usuarios de la aplicación.
  + **singletons.TicketsDB:** Singleton encargado de inicializar los tickets de la aplicación. Éstos son creados y asignados de forma aleatoria en cada inicio (Run) de la aplicación.
  + **helpers.DBHelper:** Es la clase encargada de suministrar los datos necesarios para las vistas de la aplicación.
  + **helpers.DataHelper:** Clase encargada de asistir al Singleton *TicketsDB* para gestionar correctamente la creación de los tickets.

Es importante destacar que las clases Singleton son una solución de prueba para esta primera versión y serán sustituidas por clases correspondientes que realicen consultas a un sistema de almacenamiento.

# ANEXO I Comentarios del alumno relativos a las prácticas

## Práctica 1

### Sobre la inclusión de Bases de Datos

De acuerdo a lo que se especifica en los puntos de entrega de la práctica, entiendo que la inclusión de la Base de Datos se realizará una vez lleguemos a la parte de lógica de negocio.

Por ello, para esta entrega incluí los singleton para poder utilizar la aplicación.

### Sobre la implementación del patrón Two-Step View

El patrón está implementado únicamente sobre la asignación de status al ticket. En principio el transformador solo funciona sobre esa función aunque la idea que tengo es ir convirtiendo los JSP a texto y que los comandos recurran al transformador.

Por el momento me he encontrado con el problema principal de definir una estructura que pueda ser aplicada para la mayor cantidad posible de casos sin tener que cambiar en exceso la primera fase. La idea que tengo es hacer ese cambio con las tablas de incidencias.

# ANEXO II Gráficos empleados en la memoria

[**Figura 1** 5](#_Toc98079807)

[**Figura 2** 8](#_Toc98079808)

[**Figura 3** 9](#_Toc98079809)