JUAN CASADO BALLESTEROS
MIGUEL ÁNGEL LOSADA FERNÁNDEZ
LAURA PÉREZ MEDEIRO
SERGIO SANZ SACRISTÁN

6 Plan de pruebas

Nuestro objetivo con el **plan de** pruebas es establecer unos pasos y condiciones para obtener un sistema sólido y robusto y que cumpla todos los requisitos establecidos por el cliente. El plan de pruebas se compone de 4 tipos de pruebas distintas, pruebas unitarias, de integración, de sistema y de estrés. Los objetivos de dichas pruebas son:

- **Pruebas unitarias**: garantizar el cumplimiento y funcionamiento de los componentes del sistema, en concreto se centrarán en la fiabilidad de llamadas a métodos.
- **Pruebas de integración**: comprobar el correcto funcionamiento de las llamadas entre métodos.
- **Pruebas de sistema**: probar exhaustivamente tanto los casos de uso como los requisitos no funcionales.
- **Pruebas de estrés**: probar la robustez del sistema ante posibles sobrecargas, fallos del sistema, número de usuarios, etc.

Tanto las pruebas unitarias, como las de integración se realizan en el mismo entorno de programación que utilizamos. Mientras tanto, las pruebas de sistema y estrés se realizan ya en el entorno del cliente, es decir, tal y como el cliente va a ver la aplicación. La base de datos se aloja en único sistema, además la base de datos será **Access**. En cuanto al hardware general de la empresa, se dispone de un ordenador por cada coordinador, ayudante de coordinador, responsable de almacén y ayudante de responsable de almacén. Respecto a los técnicos, cada uno tendrá un dispositivo móvil conectado al sistema. Todos los ordenadores de la empresa funcionarán mediante **Windows** y utilizarán **Mozilla Firefox** como navegador predeterminado. Los móviles de los técnicos dispondrán de **Android** (opción de **iOS** para la versión + del sistema).

Los programas utilizados para la realización de pruebas son:

- **Junit** → para llevar a cabo las pruebas unitarias.
- **PMD** → analizador estático de código (principalmente java).
- **JMeter** → permite realizar pruebas funcionales y de rendimiento para aplicaciones web.
- **Bugzilla** → seguimiento de errores.

Al finalizar el desarrollo del sistema, los desarrolladores realizan las pruebas unitarias y de integración. También durante todo el desarrollo del proyecto se llevan a cabo más pruebas unitarias y de integración, para ir comprobando en todo momento el correcto funcionamiento. Para las pruebas de sistema y de estrés se forman equipos de desarrolladores para realizar dichas pruebas. Estas se llevan a cabo para comprobar que se cumplen los requisitos funcionales, no funcionales y los casos de uso. Estas pruebas se realizan a partir de que haya software suficiente para implementar un caso de uso, y así según se van implementando los nuevos casos de uso. La prueba de estrés se realiza al finalizar el proyecto.

Identificador Prueba	**Elemento de prueba**	**Nº del caso / total**	**Objetivo**	**Entrada**	**Salida esperada**
1	Identificación	1/103	Comprobar que no puede acceder un usuario no registrado	Usuario y contraseña incorrectos	Error al identificar usuario, no existe dicho usuario en la base de datos
2	Identificación	2/103	Comprobar que no puede acceder un usuario con contraseña incorrecta	Usuario correcto y contraseña errónea	Error al identificar usuario, contraseña incorrecta

3	Identificación	3/103	Comprobar que usuario y contraseña correctos	Usuario y contraseña correctos	Acceso al menú principal del sistema
4	Identificación	4/103	Comprobar que se accede desde un dispositivo de la empresa	Acceso desde dispositivo de la empres	Acceso al menú principal del sistema
5	Identificación	5/103	Comprobar que no se accede desde un dispositivo externo	Acceso desde un dispositivo no propio de la empres	Acceso denegado, no se puede acceder desde un dispositivo no reconocido
6	Dar de alta parte de trabajo	6/103	Comprobar que el caso de uso de dar de alta parte de trabajo funciona correctamente bajo los valores de entrada esperados	La entrada esperada, la información correcta y no hay ningún parte de trabajo iniciado.	El sistema creará satisfactoriamente el parte de trabajo y mostrará un mensaje por pantalla
7	Dar de alta parte de trabajo	7/103	Comprobar que no hay campos incompletos	La entrada no tiene todos los campos necesarios	El sistema no permitirá crear el parte hasta que se rellenen todos los campos
8	Dar de alta parte de trabajo	8/103	Comprobar que hay campos no válidos (tipo de datos)	Se introduce un campo no válido en la entrada	El sistema no creará el parte de trabajo y mostrará un mensaje por pantalla
9	Dar de alta parte de trabajo	9/103	Comprobar que no hay ningún parte de trabajo iniciado.	Se introducen todos los datos para crear un parte de trabajo, pero hay un parte de trabajo iniciado.	El sistema no creará el parte de trabajo y mostrará un mensaje por pantalla
10	Dar de alta parte de trabajo	10/103	Comprobar que no es un parte de trabajo duplicado	Se introducen unos campos válidos que ya coinciden un parte de trabajo existente	El sistema no creará el parte de trabajo y mostrará un mensaje por pantalla
11	Dar de alta parte de trabajo	11/103	Comprobar que la petición asignada existe	Prueba a crear un parte de trabajo sobre una petición no existente	El sistema no permitirá crear un parte de trabajo sobre una petición no existente
12	Dar de alta parte de trabajo	12/103	Comprobar valor de la fecha de fin es superior a la fecha de inicio	Prueba que el sistema ha introducido las fechas correctamente	El sistema asigna los valores de las fechas a los partes de trabajo cuando estos son creados y finalizados

13	Dar de alta parte de trabajo	13/103	Comprobar valor del campo estado de la petición de trabajo	Prueba a actualizar estados de la petición de trabajo	El sistema actualiza automáticamente el valor del estado de la petición de trabajo asignada a dicho parte.
14	Crear factura	14/103	Comprobar valor de la factura	El sistema fuerza la entrada de un valor negativo o 0.	El sistema no permitirá crear una factura cuyo valor sea menor o igual a 0
15	Crear factura	15/103	Comprobar que el número de factura es único	Se intentan crear facturas con un número de factura repetido.	El sistema no permitirá crear una factura cuyo número de factura sea repetido.
16	Crear factura	16/103	Comprobar el parte de trabajo de la factura	Se vincula la factura a un parte de trabajo no existente	El sistema no permitirá crear una factura hasta que el campo del parte de trabajo corresponda con un parte de trabajo de la base de datos y mostrará un mensaje de error.
17	Crear factura	17/103	Comprobar que todos los campos son correctos	Todos los campos son correctos, el número y fecha de factura son correctos y se vincula a un parte de trabajo existente en la base de datos	El sistema permitirá la creación de la factura y mostrará un mensaje
18	Actualizar fechas	18/103	Comprobar fecha valida	Todos los valores de los campos de las fechas son correctos	El sistema introduce el valor de la fecha automáticamente.
19	Realizar pedido de piezas especiales	19/103	Comprobar que todas las piezas existen	Se introducen piezas inexistentes	El sistema no permitirá crear un pedido sobre piezas inexistentes y mostrará un mensaje de error
20	Realizar pedido de piezas especiales	20/103	Comprobar campos de la cabecera de pedido	Se introducen campos erróneos para los valores de la cabecera del pedido, del estado	El sistema no permitirá la creación de un con tipos de datos incorrectos,

				del pedido y de las líneas del pedido	mostrará un mensaje de error
21	Realizar pedido de piezas especiales	21/103	Comprobar valor del campo fecha	Se introduce un valor para el campo fecha incorrecto	El sistema no permitirá crear un pedido con unos valores de fecha erróneos y mostrará un mensaje de error
22	Realizar pedido de piezas especiales	22/103	Comprobar valor del campo proveedor	Se introduce un proveedor inexistente en al base de datos	El sistema no permitirá crear un pedido a un proveedor inexistente y mostrará un mensaje de error
23	Realizar pedido de piezas especiales	23/103	Comprobar valor del tipo de pedido	Se introduce un pedido con piezas especiales y tipo de pedido de piezas básicas	El sistema no permitirá crear un pedido de piezas básicas sobre piezas especiales.
24	Realizar pedido de piezas especiales	24/103	Comprobar que los pedidos son de piezas especiales o piezas básicas	Se introducen piezas especiales y piezas básicas en un mismo pedido.	El sistema no permitirá crear un pedido con tipos de piezas diferentes.
25	Realizar pedido de piezas espaciales	25/103	Comprobar estado del pedido	Probamos que el campo de estado enviado, **recibido** o **pagado** sea correcto.	El sistema actualiza el estado del pedido de piezas especiales automáticamente.
26	Realizar pedido de piezas especiales	26/103	Comprobar precio del articulo	Se introduce un precio para el artículo menor o igual que 0	El sistema no permitirá crear un pedido donde el valor de los artículos es menor o igual que 0 y mostrará un mensaje de error
27	Realizar pedido de piezas especiales	27/103	Comprobar cantidad de piezas	Se introduce una cantidad de piezas menor o igual que 0	El sistema no permitirá crear un pedido donde el número de piezas **no sean** mayor o igual que 1 y mostrará un mensaje de error
28	Realizar pedido de piezas especiales	28/103	Comprobar todos los campos del pedido	Se introducen todos los campos con valores dentro del rango esperado y correctos	El sistema permitirá crear el pedido de piezas especiales y mostrará un mensaje

					con la información del pedido
29	Asignar petición de trabajo	29/103	Comprobar petición existente	Se introduce una petición inexistente	El sistema no permitirá asignar una petición inexistente a un trabajador y mostrará un mensaje de error
30	Asignar petición de trabajo	30/103	Comprobar trabajador existente	Se introduce un trabajador inexistente	El sistema no permitirá asignar una petición a un trabajador inexistente y mostrará un mensaje de error
31	Asignar petición de trabajo	31/103	Comprobar estado de trabajador	Se introduce un trabajador el cual está de baja o de vacaciones	El sistema no permitirá asignar una petición a un trabajador que esté de naciones o de baja y mostrará un mensaje
32	Asignar petición de trabajo	32/103	Comprobar estado de petición	Probamos que el estado cancelada, terminada o pendiente de presupuesto sea correcto.	El sistema solo permitirá asignar peticiones cuyo estado **pendiente** y mostrará un mensaje. Siendo el sistema quien actualiza los estados.
33	Asignar petición de trabajo	33/103	Comprobar todos los valores son correctos	Se introduce una petición correcta y un trabajador correcto	El sistema asignará la petición de trabajo al trabajador indicado y mostrará un mensaje con información de la petición
34	Control de stock	34/103	Comprobar que los valores en stock son superiores a los mínimos.	Se introducen en la lista de las piezas básicas el número de piezas que hay. Siendo todas superiores al valor mínimo de cada pieza.	El sistema no detectará ningún déficit de piezas en el stock y por lo tanto no hace nada al respecto.
35	Control de stock	35/103	Comprobar que el sistema detecta valores de stock por debajo del stock mínimo	Uno de los valores de la lista de piezas es inferior a su mínimo.	El sistema detecta el déficit y por lo tanto realiza un pedido al proveedor correspondiente a dicha pieza.

36	Control de stock	36/103	Comprobar el buen registro de adiciones a stock de piezas básicas.	Se produce una entrada a stock de una pieza básica, es decir, se recibe un pedido.	En el listado del sistema se aumenta el número de piezas que hay en stock correspondiente al pedido recibido.
37	Control de stock	37/103	Comprobar el buen registro de salidas de stock de piezas básicas.	Se produce una salida de stock de una pieza básica, es decir, se emite una factura tras finalizar una petición de trabajo.	En el listado del sistema se disminuye el número de piezas que hay en stock correspondiente a la petición de trabajo de la factura emitida.
38	Gestión piezas	38/103	Comprobar que el listado de piezas este completo.	Se produce algún cambio de stock en el almacén.	El sistema comprueba que todas las piezas que estén en stock se encuentren en el listado de piezas.
39	Gestión piezas	39/103	Comprobar que todas las piezas tengan un proveedor asignado.	Se produce algún cambio de proveedor o se introduce una nueva pieza.	El sistema debe actualizar el listado comprobando que el proveedor sea el correspondiente a cada pieza.
40	Gestión piezas	40/103	Comprobar que las listas estén bien creadas.	Se produce alguna entrada en el stock del almacén.	El sistema comprueba si las nuevas piezas coinciden con las básicas, si es así, lo añade a la lista de piezas básicas. Y si no, lo añade a la lista de piezas especiales.
41	Gestión piezas	41/103	Comprobar que hay piezas especiales suficientes.	El responsable de almacén recibe las piezas que se van a necesitar para un trabajo.	Si las piezas especiales que se necesitan no están en stock, el responsable de almacén debe hacer un pedido a los proveedores correspondientes.
42	Gestión de peticiones	42/103	Comprobar el correcto listado de peticiones de trabajo	Se introduce una petición de trabajo.	El sistema debe comprobar que todos los datos correspondientes a la petición de trabajo estén rellenados con datos correctos.

43	Gestión de peticiones	43/103	Comprobar modificación de peticiones de trabajo.	Se selecciona una petición de trabajo del listado y se modifica su estado (o cualquier dato de la petición).	El sistema debe actualizar el estado de la petición seleccionada, comprobando que este estado sea correcto. Si es incorrecto mostrara un mensaje de error y no dejara realizar la modificación.
44	Gestión de peticiones	44/103	Comprobar la correcta creación de presupuestos.	El cliente pide al coordinador un presupuesto de su petición de trabajo.	El coordinador crea un presupuesto de la petición de trabajo realizada por el cliente.
45	Gestión de peticiones	45/103	Comprobar la correcta creación de facturas	El trabajo ha finalizado.	El coordinador debe emitir una factura a partir del parte de trabajo y se la entrega al cliente solicitante de ese trabajo.
46	Gestión de peticiones	46/103	Comprobar el correcto funcionamiento de aceptar un presupuesto	El cliente acepta un presupuesto realizado anteriormente por el coordinador.	El coordinador debe cambiar el estado de la petición de trabajo a la de presupuesto aceptado.
47	Gestión de peticiones	47/103	Comprobar el correcto funcionamiento de declinar un presupuesto en la primera versión.	El cliente no acepta un presupuesto realizado anteriormente por el coordinador.	El coordinador debe cambiar el estado de la petición de trabajo a la de presupuesto declinado.
48	Gestión de peticiones	48/103	Comprobar el correcto funcionamiento de declinar un presupuesto en la segunda y tercera versión.	El cliente no acepta un presupuesto realizado anteriormente por el coordinador.	El sistema debe actualizar el estado de la petición de trabajo a presupuesto declinado.
49	Gestión de peticiones	49/103	Comprobar el correcto funcionamiento de solicitar una petición de trabajo en la primera versión.	El cliente acepta la realización de una petición de trabajo.	El coordinador debe cambiar el estado de la petición de trabajo a la de aceptada y solicitar la petición de trabajo.

	Gestión de peticiones		Comprobar el correcto funcionamiento de solicitar una petición de trabajo en la segunda y tercera versión.	El cliente acepta la realización de una petición de trabajo.	El sistema debe actualizar el estado de la petición de trabajo automáticamente a aceptada y solicitar la petición de trabajo.
50	Gestión de peticiones	50/103	Comprobar el correcto funcionamiento de cancelar una petición de trabajo.	El cliente rechaza o cancela una petición de trabajo.	El coordinador debe cambiar el estado de la petición de trabajo a la de cancelada.
51	Gestión trabajadores	51/103	Comprobar que el listado de trabajadores este completo.	Se produce algún cambio de asignación de trabajo.	El sistema comprueba que todas las modificaciones quedan reflejadas en el listado.
52	Gestión trabajadores	52/103	Comprobar que se registran todos los trabajos realizados.	Se asigna un trabajo a un trabajador.	El sistema lo registra en el historial de trabajo del trabajador que le corresponde.
53	Gestión trabajadores	53/103	Comprobar las modificaciones realizadas.	Se cambia algún dato respecto a un trabajador.	El sistema debe registrar las modificaciones realizadas en los datos de los trabajadores del listado.
54	Gestión trabajadores	54/103	Comprobar la correcta asignación de peticiones de trabajo.	Se asigna una petición de trabajo a un trabajador.	El sistema debe registrar en el historial de trabajo que esta petición de trabajo ha sido asignada al trabajador correspondiente.
55	Gestión trabajadores	55/103	Comprobar la correcta eliminación de un trabajador.	Se despide a un trabajador de la plantilla de la empresa.	El sistema elimina del listado de trabajadores al trabajador que ha sido despedido.
56	Gestión trabajadores	56/103	Comprobar la correcta adición de un trabajador.	Se contrata a un trabajador para la empresa.	El sistema añade en el listado de trabajadores al trabajador que ha sido contratado.
57	Mostrar peticiones de	57/103	Comprobar los trabajos pendientes.	El técnico comprueba que trabajos tiene	El sistema muestra los trabajos que están con el estado

	trabajo pendientes			pendientes y tiene varios.	pendiente y están asignados al técnico que mira esta información.
58	Mostrar peticiones de trabajo pendientes	58/103	Comprobar los trabajos pendientes.	El técnico comprueba que trabajos tiene pendientes y no tiene ninguno	El sistema muestra un mensaje que indica que el técnico no tiene ningún trabajo pendiente.
59	Gestión de informes	59/103	Comprobar la correcta creación de resúmenes de trabajo	Se realiza un parte de trabajo.	El sistema registra la realización del parte de trabajo para posteriormente realizar las estadísticas que muestra el resumen de trabajo
60	Gestión de informes	60/103	Comprobar la correcta creación de informes de beneficios.	Entran beneficios a la empresa.	El sistema registra dichos beneficios para posteriormente realizar un informe de beneficios.
61	Gestión de informes	61/103	Comprobar la correcta creación de informes de trabajo.	Se asigna una petición de trabajo.	El sistema registra los cambios en los trabajadores para posteriormente realizar un informe de trabajos.
62	Gestión de informes	62/103	Comprobar la correcta creación de resumen de trabajo	Se emite un resumen de trabajo	El sistema genera a partir de los datos relacionados con los partes de trabajo un resumen de trabajo.
63	Gestión de informes	63/103	Comprobar la correcta creación de informes de beneficios.	Se emite un informe de beneficios.	El sistema genera a partir de los datos relacionados con los gastos y beneficios un informe de beneficios.
64	Gestión de informes	64/103	Comprobar la correcta creación de informes de trabajo.	Se emite un informe de trabajo.	El sistema genera a partir de los datos relacionados con los trabajadores y partes de trabajo un informe de trabajo.

65	RNF1	65/103	Comprobar la buena instalación de la base de datos	Se instala la nueva base de datos.	Se instala en el mismo ordenador donde estaba la anterior base de datos.
66	RNF2	66/103	Comprobar el buen acceso a los equipos	Inicia sesión correctamente o un responsable técnico, o un ayudante de almacén, o un coordinador técnico, o un ayudante de coordinador.	El equipo permite el acceso a la información del sistema.
67	RNF2	67/103	Comprobar el buen acceso a los equipos	Inicia sesión de forma errónea alguien.	El sistema emite un mensaje de error y recuerda que solo pueden acceder al sistema los responsables de almacén, ayudantes de almacén, coordinadores técnicos y ayudantes del coordinador.
68	RNF3	68/103	Comprobar que los técnicos pueden utilizar sus móviles.	Un técnico informático inicia sesión desde su dispositivo móvil	El sistema reconoce a este trabajador y le da los permisos necesarios para que éste pueda realizar su trabajo correctamente, aunque sea desde su dispositivo móvil.
69	RNF4	69/103	Comprobar el auto apagado del servidor	Finaliza la jornada laboral del día.	El sistema y los dispositivos utilizados para trabajar quedan suspendidos, ya que, el horario de trabajo ya ha finalizado.
70	RNF4	70/103	Comprobar el auto encendido del servidor	Comienza la jornada laboral del día.	El sistema y los dispositivos utilizados para trabajar quedan habilitados para la realización de la jornada laboral.
71	RNF5	71/103	Comprobar la disponibilidad (2 y 3)	-	La disponibilidad en las versiones que tienen Apache es máxima, ya que, no

					necesita apagarse para realizar cambios en él.
72	RNF6	72/103	Comprobar el sistema de actualizaciones (1)	Se va a realizar una actualización de la JMV y de la aplicación java.	El sistema realiza un aviso con dos días de antelación para advertir a los trabajadores.
73	RNF7	73/103	Comprobar el correcto funcionamiento de la base de datos	El tiempo de respuesta para las aplicaciones de escritorio es mayor a 3s. y de las aplicaciones móviles con todos los clientes activos mayor a 5s.	Por lo tanto, el sistema se ve obligado a cambiar la base de datos en el ordenador que corresponda.
74	RNF8	74/103	Comprobar que las aplicaciones se adaptan a los ajustes por defecto.	-	Éstas se deben adaptar al tamaño de fuente, fondo, volumen y estilo que este por defecto en el equipo que utilizan.
75	RNF9	75/103	Comprobar que se muestra bien la aplicación.	-	La aplicación creada debe mostrar bien el contenido adaptándose al hw del equipo en cuestión.
76	RNF10	76/103	Comprobar la correcta distribución de la interfaz.	-	La aplicación se basa en una interfaz de botones y listas manteniendo el diseño entre las aplicaciones móviles y las de escritorio.
77	RNF11	77/103	Comprobar que los usuarios se adaptan a la aplicación	-	Hemos creado un plan de adaptación al uso para que los usuarios puedan ser aconsejados por otros usuarios para la correcta utilización de la aplicación.
78	RNF12	78/103	Comprobar que la aplicación se adapta a cada plataforma móvil	-	Dependiendo de la plataforma móvil que se disponga, la aplicación integra a Siri o Google Assistant

					como característica adicional.
79	RNF13	79/103	Comprobar la protección de datos	-	Todos los datos están guardados en una base de datos, para así protegerlos ante posibles fallos del sistema.
80	RNF14	80/103	Comprobar el servicio de mantenimiento	-	Si ocurre un fallo, hay personal de guardia para subsanar dicho fallo.
81	RNF15	81/103	Comprobar la estabilidad del sistema (1)	-	Todo el software será probado exhaustivamente para reducir el número de fallos.
82	RNF16	82/103	Comprobar la estabilidad del sistema (2 y 3)	-	Los servidores Apache proporcionan fiabilidad y estabilidad.
83	RNF17	83/103	Comprobar la estabilidad del sistema (3)	-	Las aplicaciones nativas evitan los posibles errores y el elevado número de dependencias.
84	RNF18	84/103	Comprobar la portabilidad del sistema en aplicación de escritorio (1)	-	Las aplicaciones basadas en java tienen muy buena portabilidad, pero las actualizaciones del sistema pueden afectar a esto.
85	RNF19	85/103	Comprobar la portabilidad del sistema en aplicación de escritorio (2 y 3)	-	Éstas corren sobre el navegador, reduciendo costes de mantenimiento y los problemas de actualizaciones.
86	RNF20	86/103	Comprobar la portabilidad del sistema en aplicación móvil.	-	En la versión barata hay una gran portabilidad, pero en la más cara al tener dos aplicaciones móviles nativas hay

					mayores ventajas de portabilidad.
87	RNF21	87/103	Comprobar la portabilidad de la base de datos.	-	Portabilidad inmejorable siempre que se pueda crear una conexión TCP con ella.
88	RNF22	88/103	Comprobar el costo del sistema (1)	-	Económica, pero hay menos funcionalidades, sobre todo para los clientes.
89	RNF23	89/103	Comprobar el costo del sistema (2)	-	Mas cara, pero se centra en los clientes, con una página web para ellos.
90	RNF24	90/103	Comprobar el costo del sistema (3)	-	Igual de cara que la anterior, pero está más centrada en facilitar el trabajo de los técnicos informáticos.
91	RNF25	91/103	Comprobar el costo del sistema (2 y 3)	-	Tiene un coste añadido, ya que, mejora el HW del ordenador en el que este el servidor con Apache.
92	RNF26	92/103	Comprobar la interoperabilidad del sistema (1)	-	El cliente java se comunica directamente con la base de datos mediante una conexión TCP.
93	RNF27	93/103	Comprobar la interoperabilidad del sistema (2 y 3)	-	Los clientes web se comunican con el servidor Apache, y éste con la base de datos mediante PHP.
94	RNF28	94/103	Comprobar la interoperabilidad de las aplicaciones móviles.	-	Se comunican directamente con la base de datos mediante TCP.
95	RNF29	95/103	Comprobar la escalabilidad del sistema.	-	El código es creado mediante patrones de diseño.

96	RNF30	96/103	Comprobar la escalabilidad del sistema.	-	Se aplicarán técnicas de POO para mayor estabilidad y mantenimiento.
97	RNF31	97/103	Comprobar número de usuarios simultáneos	-	Habrá tantos como trabajadores haya en la empresa, con un margen por si se produce un pico de contrataciones.
98	RNF32	98/103	Comprobar la mantenibilidad del sistema.	-	Se ofrecen distintos planes de mantenimiento.
99	RNF33	99/103	Comprobar la seguridad de acceso.	-	Cada usuario tiene una contraseña y usuario únicos.
100	RNF34	100/103	Comprobar la privacidad.	-	Cada modificación en la base de datos queda registrada quien la hizo.
101	RNF35	101/103	Comprobar las limitaciones de permisos	-	Cada usuario solo puede acceder a lo que tiene permisos, así como mostrar ciertos datos.
102	RNF36	102/103	Comprobar que se realiza una copia de seguridad por semana	-	Una vez a la semana el sistema hace una copia de seguridad de la base de datos, que se guarda 30 días.
103	RNF37	103/103	Comprobar que no se puede acceder al sistema desde un equipo ajeno a la empresa.	-	Solo se puede acceder al sistema a través de un equipo perteneciente a la empresa, aunque el usuario sea válido.

1.1 PRUEBAS UNITARIAS

Las pruebas unitarias consisten en comprobar el correcto funcionamiento de una unidad de código. Probamos una clase del sistema, que funcione correctamente individualmente. Así comprobamos que esta clase funcione correctamente y de una forma eficiente por separado al resto del programa. Así también podemos comprobar la correcta utilización de los nombres del sistema, los tipos de parámetros, el tipo que se devuelve y si el estado inicial y final son válidos. Para que estas pruebas sean de calidad debemos tener en cuenta que deben ser automatizables, es decir, que no debe requerir una intervención manual. Completas, que deben cubrir la mayor cantidad de código posible. Repetibles o reutilizables. Independientes para que la ejecución de una prueba no afecte a la ejecución de otra y deben estar documentadas y bien realizadas.

JUAN CASADO BALLESTEROS
MIGUEL ÁNGEL LOSADA FERNÁNDEZ
LAURA PÉREZ MEDEIRO
SERGIO SANZ SACRISTÁN

Nuestro sistema está realizado en varios lenguajes de programación. En la primera solución de nuestro sistema en la aplicación de escritorio utilizamos el lenguaje de programación Java. Por lo tanto, en este caso para realizar las pruebas unitarias de esta parte del código lo realizaremos con el framework **JUnit** el cual está integrado en NetBeans.

Para la realización de las aplicaciones móviles utilizamos el framework Ionic 2, el cual está desarrollado por el lenguaje de programación JavaScript, al igual que en la aplicación de escritorio de las soluciones 2 y 3. Las pruebas unitarias de la parte de JavaScript las realizaremos con el framework **QUnit**, este framework está integrado en el entorno de desarrollo JetBrains WebStorm.

En la comunicación entre el servidor Apache y la base de datos de la 2º solución se realiza a través del lenguaje PHP. Estas pruebas unitarias se realizarán a través del framework **PHPUnit**, este framework está integrado en el entorno de desarrollo PHPStorm.

En la aplicación móvil de la solución 3 para realizar la funcionalidad para iOS utilizamos el lenguaje de programación Swift. Para realizar las pruebas unitarias de estos métodos utilizaremos el framework **XCode**.

Pruebas unitarias con JUnit

Aquí realizaremos pruebas unitarias de las clases desarrolladas con Java. Estas pruebas unitarias las realizaremos a través del entorno de desarrollo NetBeans.

El funcionamiento de JUnit se basa en asserts. Esto es una clase del paquete junit.framework la cual nos proporciona un conjunto de métodos los cuales son lo que realmente hacen las pruebas si un método especifico de nuestra clase hace las cosas como debería. Estos métodos no son complicados, si la condición da algún tipo de error entonces la prueba no cumple con lo solicitado y se da la prueba como no superada. Es decir, estos métodos los utilizamos para verificar resultados que ya conocemos de antemano.

Para realizar las pruebas unitarias debemos tener el código de la clase, con sus métodos correspondientes para hacer la prueba en NetBeans. Para realizar la prueba con JUnit solo debemos hacer click derecho en nuestra clase y seleccionamos “Crea pruebas JUnit” y seleccionamos la versión de JUnit. Ahora en NetBeans tenemos creada una nueva clase de prueba, que nos servirá para probar directamente los métodos de nuestra clase.

Entonces ahora es donde tenemos que ir cambiando los valores de nuestras variables y de los atributos a introducir a los métodos correspondientes. Aquí también tenemos una instancia a nuestra clase previa con la que obtenemos el valor de salida que conocemos de antemano. Entonces ya solo queda ejecutar la prueba sí coinciden los valores de salida entonces la prueba será superada satisfactoriamente. Si no coinciden dará un error y la prueba no será superada.

Pruebas unitarias QUnit

Las pruebas unitarias QUnit las desarrollaremos en el entorno de desarrollo JetBrains WebStorm. Estas pruebas aparte de QUnit son conocidas en el entorno WebStorm como JsTestDriver.

Para comenzar debemos iniciar el servidor JsTestDriver localmente. Una vez iniciado la barra de estado se vuelve amarilla para avisarle que el servidor se está ejecutando, pero no tiene navegadores esclavos.

Luego, se copia la URL de captura y se pega en un navegador. La barra de estado se vuelve verde y el icono del navegador correspondiente se ilumina. Entonces ahora seria cuando se pueden realizar las pruebas correspondientes.

Las pruebas se ejecutarán en el servidor JsTestDriver local. En caso de que la prueba sea fallida puede navegar fácilmente desde el seguimiento de la pila hasta el código fuente que causa problemas. También puede navegar desde el árbol de resultados de prueba a la declaración de función de caso / prueba de prueba usando la tecla F4. Así conseguimos localizar fácilmente donde se encuentra el error dado.

Pruebas unitarias PHPUnit

Las pruebas unitarias con el framework PHPUnit las desarrollamos en el entorno de desarrollo PHPStorm.

Una vez que tenemos el código de la clase a analizar hacemos click derecho sobre ella y seleccionamos Prueba de PHP->PHPUnit Test. Así entonces generamos una prueba para la clase PHP definida.

Una vez creada la nueva clase de pruebas PHPUnit, tan solo queda ejecutarla. Si los datos de salida de las clases corresponden entonces se da la prueba como superada. Si no, dará un error y la prueba no habrá sido superada por la clase.

Pruebas unitarias XCode

Para las pruebas unitarias en iOS utilizamos el framework XCode. Aquí nos encontramos con dos tipos de pruebas unitarias, las lógicas, que verifican el correcto funcionamiento de un fragmento de código de forma independiente y las de aplicación que verifican el correcto funcionamiento de fragmentos de código dentro del contexto de nuestra app. A nosotros actualmente nos interesa el primer caso.

Para la creación de métodos de prueba se utiliza el prefijo test seguido del nombre del caso de prueba. Dado que estos métodos no tienen ni parámetros de entrada ni de retorno, XCode nos proporciona una serie de Macros, que podemos utilizar para la comprobación de las diferentes validaciones que queramos verificar en nuestros casos de prueba.

Entonces, una vez que tenemos creados todos los métodos de prueba introducimos distintos tipos de valores de entrada para ver como funcionarán los métodos de la clase a probar y si coincide con los resultados que sabíamos de antemano entonces la clase habrá superado la prueba.

1.1.1 EJEMPLO PRUEBA UNITARIA

Vamos a explicar cómo realizar una prueba unitaria tomando como ejemplo el método cantidadPedir() de la clase Piezas Básicas, que hereda de Piezas. Dicho método calcula de una pieza la cantidad que se debe pedir. Para ello, utiliza los atributos de la pieza básica retornando 0, si no hace falta pedir piezas de ese tipo. O un número mayor que 0 indicando la cantidad de piezas que es necesario pedir.

1.1.1.1 ESQUEMA PSEUDOCODIGO DE PIEZAS BÁSICAS

```
Func CantidadPedir(void) -> int{
        If self.CantidadAlmacen < self.StockMinimo{
                Return self.CantidadPedido
        }
        Return 0
}
```

1.1.1.2 SET DE DATOS

Para realizar las pruebas crearemos un set de dato en el que incluimos todas las posibilidades del código que queremos probar. Trataremos al método desde una perspectiva de caja negra a la que proporcionaremos una entrada y de la que esperamos una salida derivada de la entrada proporcionada.

En este caso concreto crearemos un total de 10 piezas de las que 5 no deberán retornar un número mayor que 0 y 5 sí. En las que deban retornar 0 el stock mínimo será superior a la cantidad de almacén en las que no deban retornar 0 el stock será inferior.

Cuando el numero retornado sea distinto de 0 este debe coincidir con la cantidad pedido.

1.1.1.3 REALIZACION DE LA PRUEBA

Para crear el código de la prueba tendremos en cuenta las siguientes consideraciones, este deberá ser reutilizable en múltiples escenarios, diciéndonos en todo caso si el método a probar funciona correctamente o no. La prueba unitaria que creemos será compatible con las pruebas de regresión.

Nuestro código tomara un set de Piezas Básicas y del resultado esperado al ejecutar el método que queremos probar. Sobre el set de Piezas seguirá ejecutando el código y el resultado proporcionado por este se comparará con el esperado. En el caso de que el resultado esperado no coincidiera con el esperado se notificara que al menos una Pieza

ha fallado al realizarse la prueba. En el caso de que hubiera habido fallos al realizarse la prueba obtendremos el listado de piezas en las que se hayan producido fallos junto con el resultado esperado en cada caso.

1.2 PRUEBAS DE INTEGRACION

Estas pruebas se realizan posteriormente a las pruebas unitarias de modo que los fallos en métodos aislados hayan sido previamente descartados en los set de datos de prueba proporcionados. Lo que pretendemos probar es que las llamadas entre métodos funcionan adecuadamente siendo compatibles los parámetros de salida de unos con los de entrada de otros, así como que el sistema se coordina adecuadamente para hacer las acciones que deseamos.

1.2.1 EJEMPLO PRUEBA DE INTEGRACION

Para realizar la prueba de integración hemos seleccionado como método iniciante de la prueba Agrupar Pedidos. Este método seria llamado por la clase Timer en su método acción cuando la clase sistema se subscribiera a él. Nosotros realizaremos ahora la llamada de callback que dicho método realizaría.

Las acciones que realiza este método y de las que comprobaremos su integración son las siguientes:

- Actualizar los pedidos almacenados en la base de datos: para ello la clase sistema llamara a la fachada de la base de datos proporcionándole la fecha en la que actualizo los pedidos por última vez obteniendo un HashMap con los nuevos pedidos que se hayan creado en la bbdd.
- Iteración sobre el HashMap: se leerá secuencialmente el HashMap agrupando los pedidos que haya por proveedor. Para esto se creará un nuevo HashMap en el que la clave de búsqueda será el identificador del proveedor y el contenido a almacenar será un ArrayList de Pedidos.
- Notificación al proveedor: por cada proveedor para el que haya pedidos crearemos un email mediante la clase GeneradorDeEmails que contendrá el listado de piezas que le debamos solicitar junto con la cantidad de estas.

1.2.1.1 SET DE DATOS

La prueba se iniciará de modo que seremos nosotros y no el timer el que llame al método agrupar pedidos. El enfoque que aplicaremos sobre la prueba será el de caja negra. Para realizar la prueba introduciremos un set de datos falsos sobre la bbdd y un host falso al que enviar los emails.

Lo que comprobaremos será que ante el set de datos proporcionados se crearan tantos emails como sean necesarios, uno por proveedor al que haya que solicitar piezas y en cada email se deberá indicar todas las piezas junto con su cantidad.

El set de datos incluirá la mayor variedad posible de escenarios. Al menos uno en que no se haya que crear ningún email, ya que, no habrá pedidos nuevos en la bbdd, al menos otro en el que todos los pedidos tengan la misma cantidad de piezas, otro más, en el que las cantidades de pieza para cada proveedor sean variadas, y otro más, en el que a todos los proveedores se le tenga que pedir al menos una de todas las piezas que nos puedan proporcionar. Dichos escenarios se proporcionarán como tablas con datos falsos en la bbdd

1.2.1.2 REALIZACION DE LA PRUEBA

Proporcionado el set de datos con las tablas rellenas de datos falsos de la bbdd y su correspondiente resultado esperado se hará al sistema tomar como entrada los datos de dichas tablas. Comprobaremos interceptando los mensajes que reciba el host simulado si los emails se crearon correctamente tanto en cantidad como en contenido. En caso de haberse producido algún fallo se nos indicara el set de datos y el resultado esperado de dicho set en el que el fallo ocurrió.