TICKETING

Índice

Descripción de los grupos de trabajo

Grupo compuesto por Andrés Villalba, Cristian Peter, Cristina Tomás y Javier González.

Las tareas se asignarán de manera equitativa en función de las aptitudes.

Plantilla de documentación

Documentación de pruebas:

Documentación de entrada

ID	Caso de Prueba	Descripción	Fecha	Función probada	Procedimientos

Documentación de salida

ID	Caso de Prueba	Descripción	Fecha	Función probada	Salida esperada	Resultado obtenido

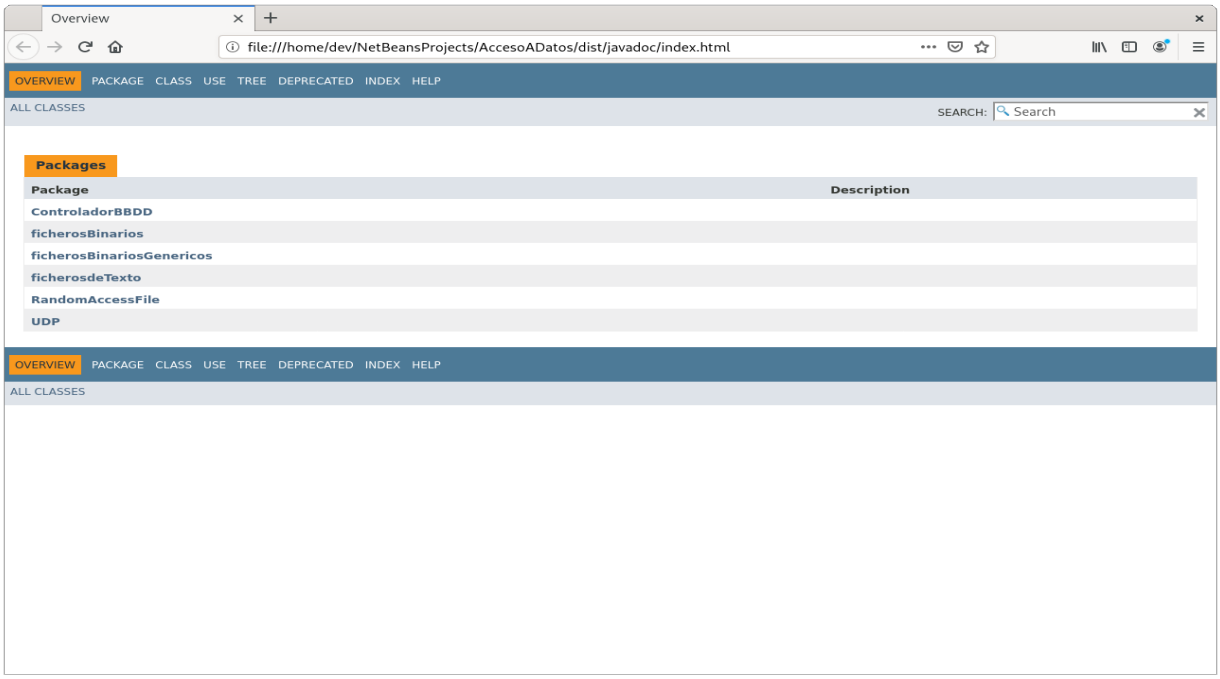
Documentación técnica

Documentación Interna

Se incluirán comentarios en el código para mantenerlo ordenado y claro, ayudando a entender el código con facilidad evitando escribir comentarios que indiquen cosas obvias.
Se incluirán comentarios al comienzo del modulo, se comentarán las variables, constantes,procedimientos y funciones.

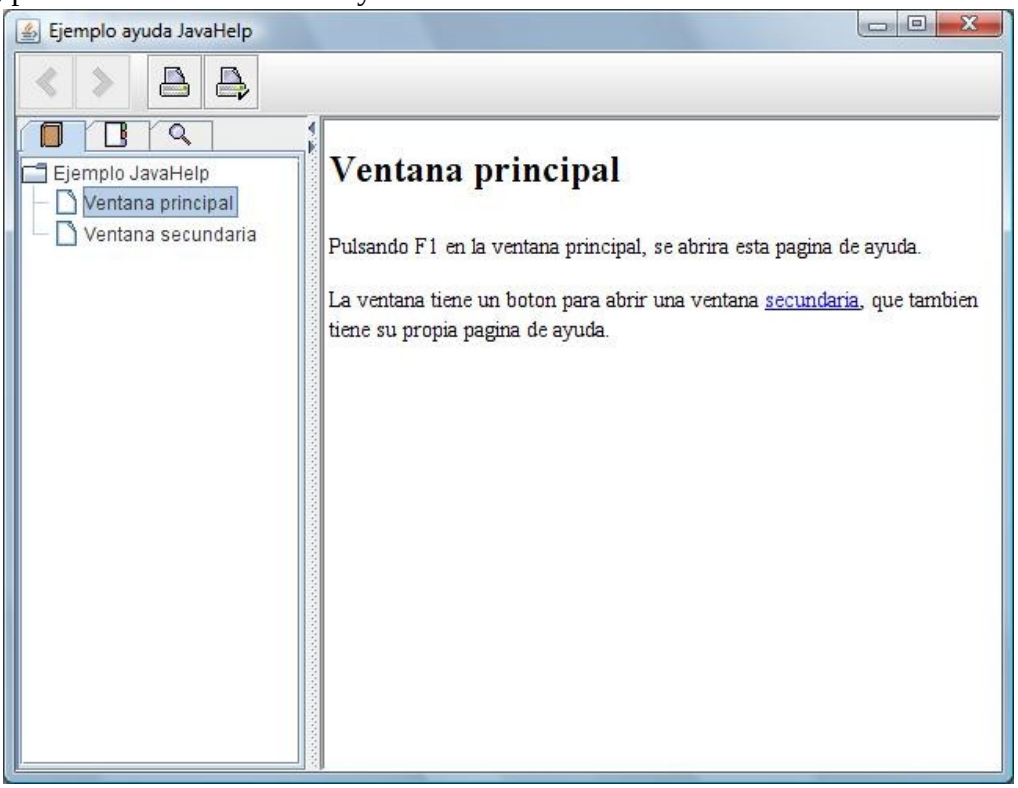
Documentación Externa

Utilizaremos la función de javadoc para crear un manual de nuestra aplicación gracias a los comentarios que añadamos en el código

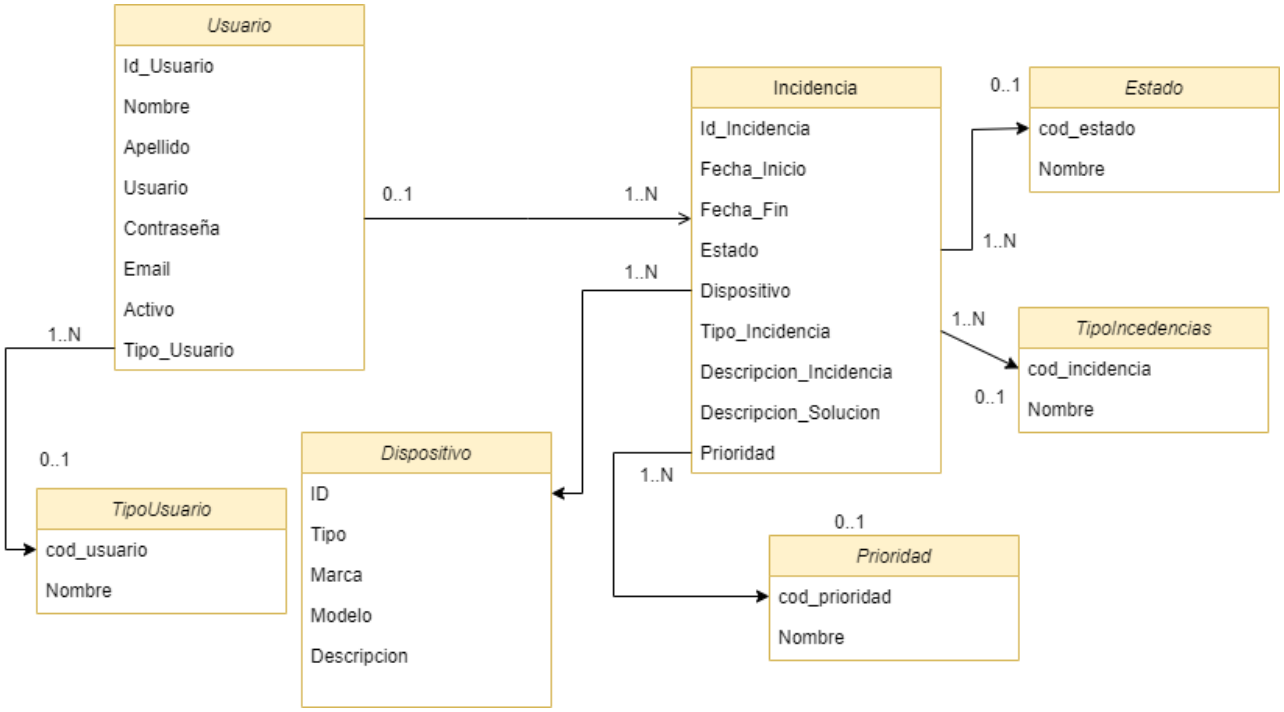


Manuales, guías y ficheros de ayuda

Realizaremos todo el sistema de manual, guía y ficheros de ayuda mediante el sistema JavaHelp, pues es sencillo de utilizar y fácilmente escalable.



Modelo de datos



Definiciones y especificación de requisitos

Definición general

Realización de una aplicación para gestionar los contactos de los trabajadores de una empresa con el departamento de informática mediante la apertura y seguimiento de tickets de petición de trabajo.

El programa dispondrá de diferentes tipos de usuario (básico, gestor, técnico y administrador) y tendrán acceso a diferentes funciones derivadas de esto.

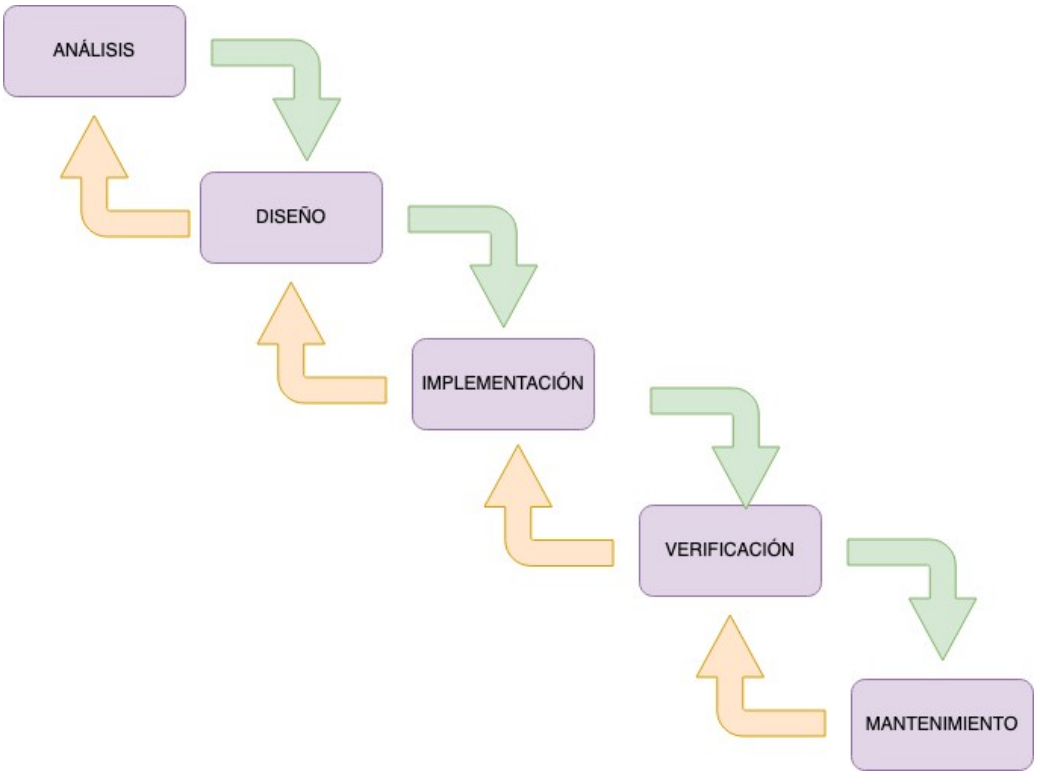
Especificaciones de requisitos

El objetivo principal del sistema de ticketing es servir como medio de comunicación entre trabajadores de la empresa y el departamento de IT. En los tickets deben recogerse las necesidades, problemas o solicitudes de los usuarios y permitir el seguimiento de estas desde el mismo sistema.

Los tickets abiertos deben tener carácter contractual y el departamento de IT debe comprometerse a buscar soluciones a los problemas recibidos.

Procedimientos de desarrollo

El modelo de desarrollo será en cascada con retroalimentación



Procedimientos de instalación y prueba

Arquitectura del sistema

Descripción jerárquica

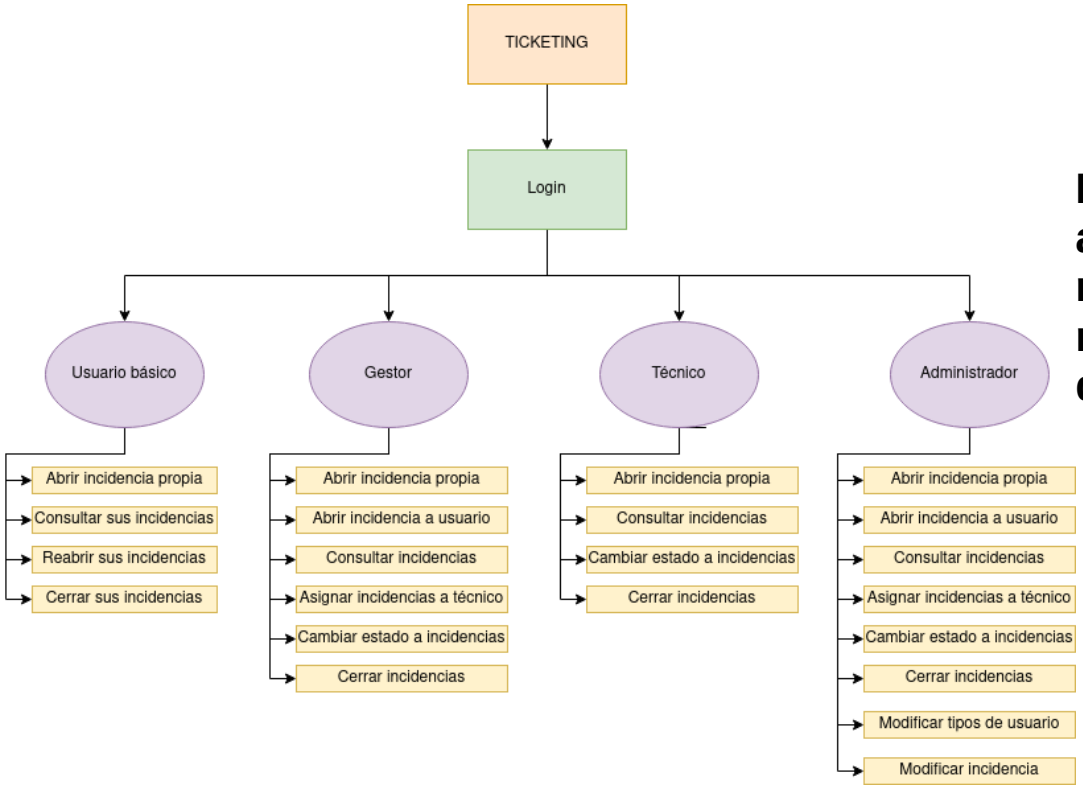
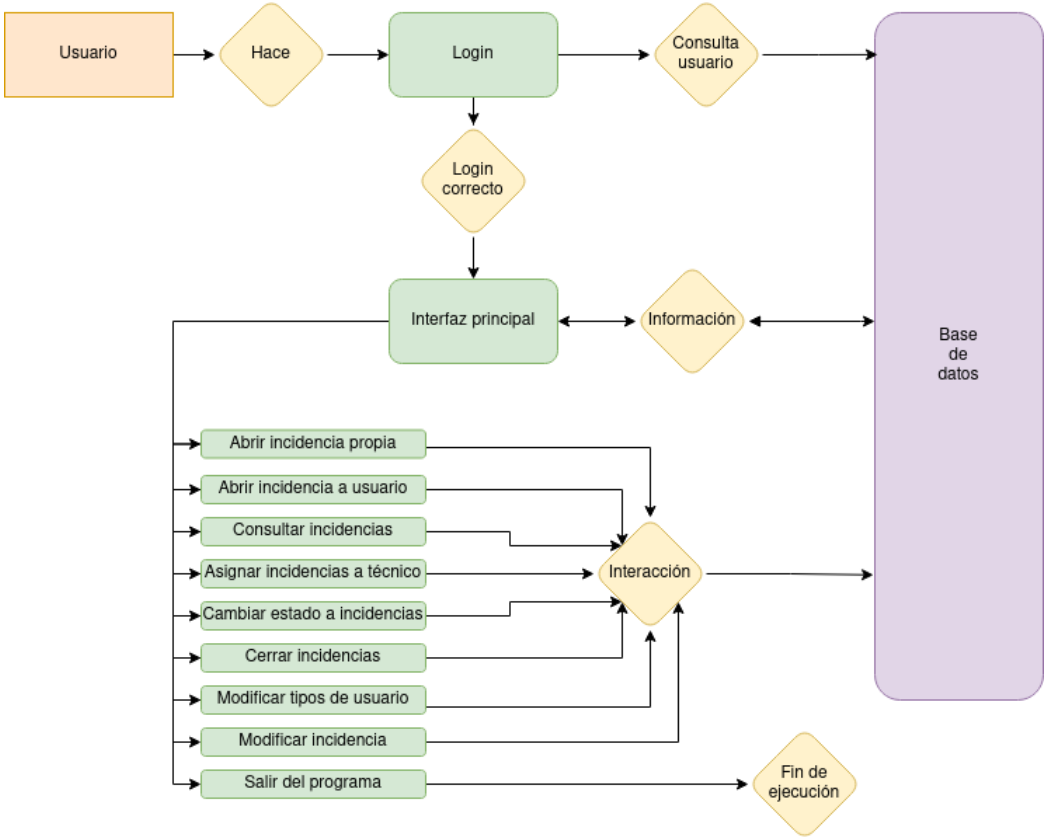


Diagrama de módulos



Descripción individual de los módulos

Abrir incidencia: Permite realizar la apertura de una incidencia en el sistema.

Consultar sus incidencias: Permite ver un listado de las incidencias propias.

Consultar incidencias: Permite a un gestor, técnico o administrador, ver todas las incidencias.

Cerrar incidencia: Permite cerrar incidencias, por parte del usuario o los técnicos.

Asignar incidencia: Permite al gestor, asignar las incidencias a un técnico en concreto.

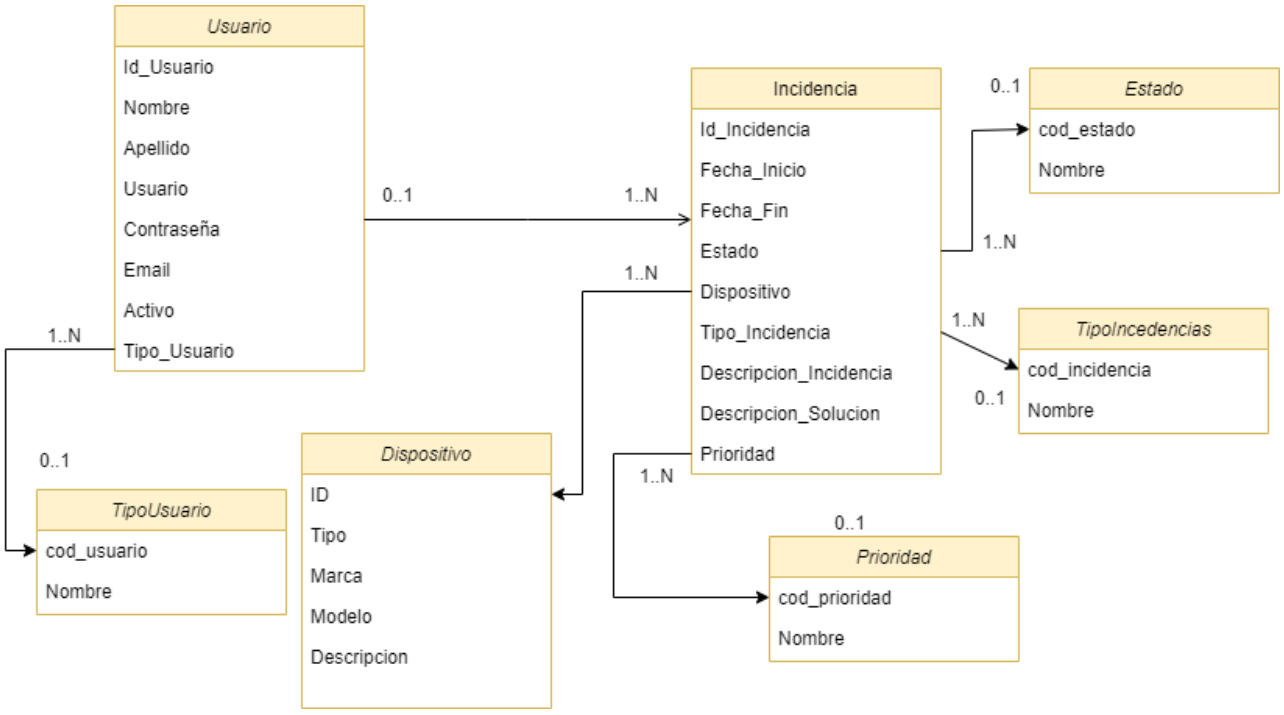
Cambiar estado a incidencia: Permite modificar el estado de una incidencia.

Modificar tipo de usuario: permite al administrador otorgar permisos de gestor o técnico a los usuarios básicos.

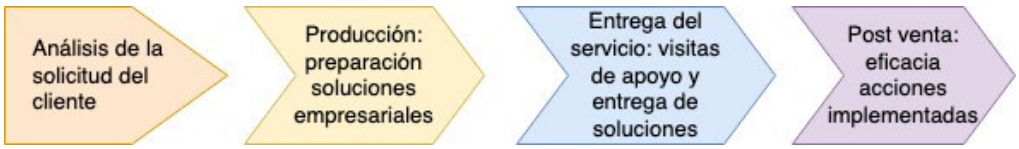
Dependencias externas

En este caso, las dependencias externas se compondrán básicamente en la distribuidora de internet que de servicio para que la aplicación tenga un funcionamiento adecuado y constante, también habrá que tener en cuenta que el funcionamiento de los equipos de trabajo sea el correcto para la ejecución software.

Diseño del modelo de datos



Descripción de procesos y servicios



Plan de realización de pruebas

Hay dos tipos de modificaciones o ampliaciones dentro de un plan de pruebas, que han de tomarse en cuenta para que se pueda cubrir toda la funcionalidad existente:

- **Funcionalidades de usuario final:** cuando se agregan nuevos módulos, pantallas o flujos en las funcionalidades que son utilizadas por el cliente o se ven visualmente.
- **Funcionalidades internas:** son cambios internos que mantienen todos los elementos visuales de la misma manera, por lo tanto, el cliente final no observa ninguna modificación, pero al modificar componentes o flujos internos (como accesos a base de datos o a capas de lógica de negocio), deben de cubrirse con casos de pruebas dentro del plan de pruebas y cubrir toda la funcionalidad de este tipo.

- Realizaremos los siguientes casos de prueba:

Pruebas Unitarias

Pruebas de Integración

Pruebas del Sistema

Pruebas de Implantación

Pruebas de Aceptación

Pruebas de Regresión

Detalle de las pruebas que se van a hacer y tipos

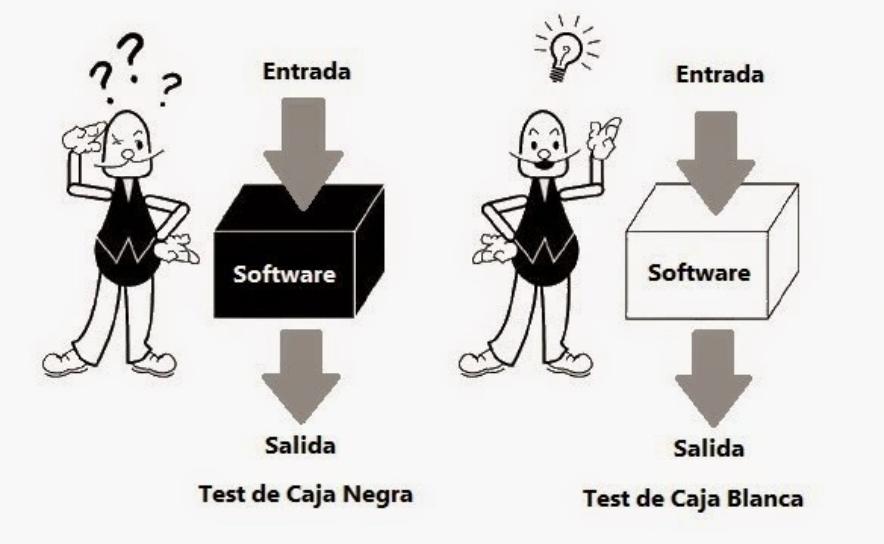
Pruebas unitarias

→ **Caja blanca (interno):** Se verifica la estructura interna del componente con independencia de la funcionalidad establecida para el mismo. Por tanto, no se comprueba la corrección de los resultados si éstos se producen. Ejemplos de este tipo de pruebas pueden ser ejecutar todas las instrucciones del programa, localizar código no usado, comprobar los caminos lógicos del programa, etc.

→ **Caja negra (externo):** Se comprueba el correcto funcionamiento de los componentes del sistema de información, analizando las entradas y salidas y verificando que el resultado es el esperado. Se consideran exclusivamente las entradas y salidas del sistema sin preocuparse por la estructura interna del mismo.

Para un correcto funcionamiento de las pruebas unitarias habría que ejecutar todos los casos de prueba posibles, registrando ese resultado, contemplando siempre las condiciones válidas e inválidas. Tanto como corregir los errores o defectos encontrados y repetir las pruebas que los detectaron. Si se considera necesario, debido a su implicación o importancia, se repetirán otros casos de prueba ya realizados con anterioridad.

Haremos las dos pruebas



Pruebas de integración

→ **Integración incremental:** Se combina el siguiente componente que se debe probar con el conjunto de componentes que ya están probados y se va incrementando progresivamente el número de componentes a probar. Con el tipo de prueba incremental lo más probable es que los problemas que surjan al incorporar un nuevo componente o un grupo de componentes previamente probado, sean debidos a este último o a las interfaces entre éste y los otros componentes.

→ **Integración no incremental:** Se prueba cada componente por separado y posteriormente se integran todos de una vez realizando las pruebas pertinentes. Este tipo de integración se denomina también *Big-Bang* (gran explosión).

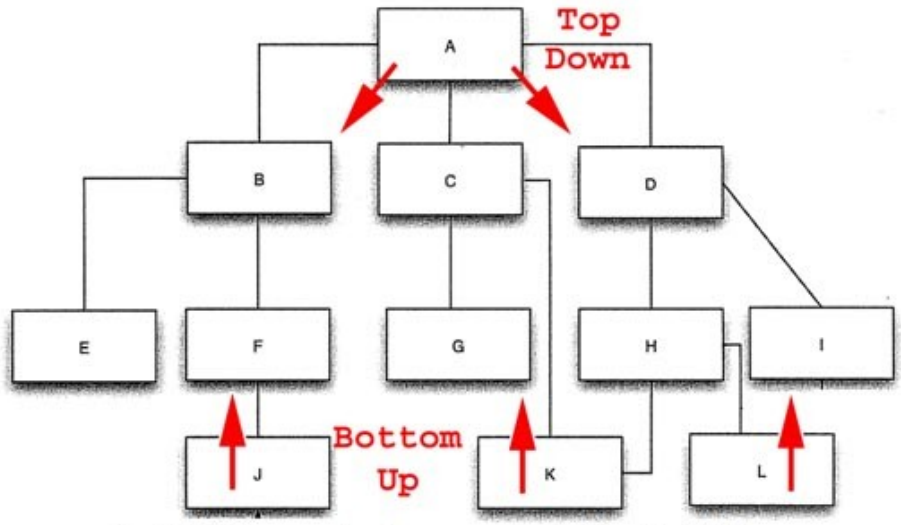


Fig 5.8 *"The art of software testing"*. Glenford J.Myers.

En la integración incremental de componentes (ver figura anterior) se distinguen las siguientes estrategias de integración:

→ De arriba abajo (*top-down*). El primer componente que se desarrolla y prueba es el primero de la jerarquía (A). Los componentes de nivel más bajo se sustituyen por componentes auxiliares para simular a los componentes invocados. En este caso no son necesarios componentes conductores. Una de las ventajas de aplicar esta estrategia es que las interfaces entre los distintos componentes se prueban en una fase temprana y con frecuencia.

→ De abajo arriba (*bottom-up*). En este caso se crean primero los componentes de más bajo nivel (E, F) y se crean componentes conductores para simular a los componentes que los llaman. A continuación se desarrollan los componentes de más alto nivel (B, C, D) y se prueban. Por último dichos componentes se combinan con el que los llama (A). Los componentes auxiliares son necesarios en raras ocasiones.

Este tipo de enfoque permite un desarrollo más en paralelo que el enfoque de arriba abajo, pero presenta mayores dificultades a la hora de planificar y de gestionar.

→ **Estrategias combinadas**. A menudo es útil aplicar las estrategias anteriores conjuntamente. De este modo, se desarrollan partes del sistema con un enfoque "*top-down*", mientras que los componentes más críticos en el nivel más bajo se desarrollan siguiendo un enfoque *"bottom-up*". En este caso es necesaria una planificación cuidadosa y coordinada de modo que los componentes individuales se "encuentren" en el centro.

En muestro proyecto elegiremos las pruebas de integración incremental – de arriba abajo (top-down)

Pruebas del sistema

Las pruebas del sistema son pruebas de integración del sistema de información completo, y permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen. Dan una visión muy similar a su comportamiento en el entorno de producción.

Varios tipos :

→ **Pruebas funcionales:** Dirigidas a asegurar que el sistema de información realiza correctamente todas las funciones que se han detallado en las especificaciones dadas por el usuario del sistema.

→ **Pruebas de comunicaciones:** Determinan que las interfaces entre los componentes del sistema funcionan adecuadamente, tanto a través de dispositivos remotos, como locales. Asimismo, se han de probar las interfaces hombre/máquina.
→ **Pruebas de rendimiento:** Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema.
→ **Pruebas de volumen:** Consisten en examinar el funcionamiento del sistema cuando está trabajando con grandes volúmenes de datos, simulando las cargas de trabajo esperadas.
→ **Pruebas de sobrecarga:** Consisten en comprobar el funcionamiento del sistema en el umbral límite de los recursos, sometiéndole a cargas masivas. El objetivo es establecer los puntos extremos en los cuales el sistema empieza a operar por debajo de los requisitos establecidos.
→ **Pruebas de disponibilidad de datos:** Consisten en demostrar que el sistema puede recuperarse ante fallos, tanto de equipo físico como lógico, sin comprometer la integridad de los datos.
→ **Pruebas de facilidad de uso:** Consisten en comprobar la adaptabilidad del sistema a las necesidades de los usuarios, tanto para asegurar que se acomoda a su modo habitual de trabajo, como para determinar las facilidades que aporta al introducir datos en el sistema y obtener los resultados.

→ **Pruebas de operación:** Consisten en comprobar la correcta implementación de los procedimientos de operación, incluyendo la planificación y control de trabajos, arranque y re-arranque del sistema, etc.
→ **Pruebas de entorno:** Consisten en verificar las interacciones del sistema con otros sistemas dentro del mismo entorno.
→ **Pruebas de seguridad:** Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.

Pruebas de implantación

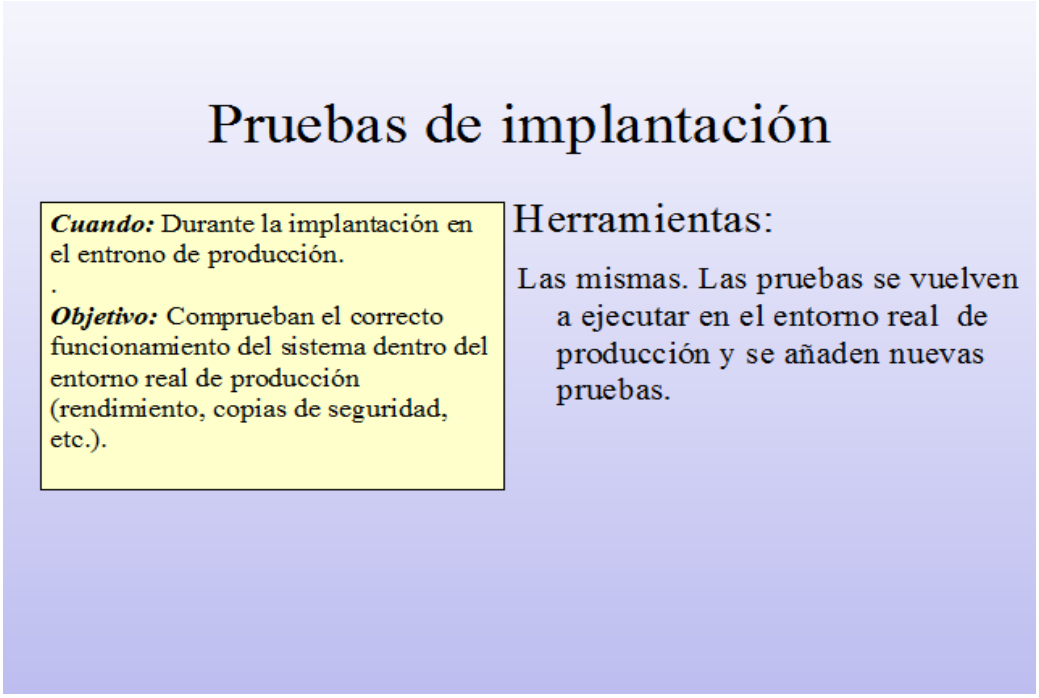
Una vez realizadas las pruebas de sistemas, se llevan a cabo las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación. Debe comprobarse que responde satisfactoriamente a los requisitos de rendimiento, seguridad, operación y coexistencia con el resto de los sistemas de la instalación para conseguir la aceptación del usuario de operación.

Seguridad → van dirigidas a verificar que los mecanismos de protección incorporados al sistema cumplen su objetivo.

Rendimiento → dirigidas a asegurar que el sistema responde satisfactoriamente en los márgenes establecidos en cuanto tiempos de respuesta, de ejecución y de utilización de recursos, así como los volúmenes de espacio en disco y capacidad.

Operación → se comprueba que la planificación y control de trabajos del sistema se realiza de acuerdo a los procedimientos establecidos, considerando la gestión y control de las comunicaciones y asegurando la disponibilidad de los distintos recursos.

Gestión de copias de seguridad y recuperación → su objetivo es que el sistema no vea comprometido su funcionamiento al existir un control y seguimiento de los procedimientos de salvaguarda y de recuperación de la información, en caso de caídas en los servicios o en algunos de sus componentes.

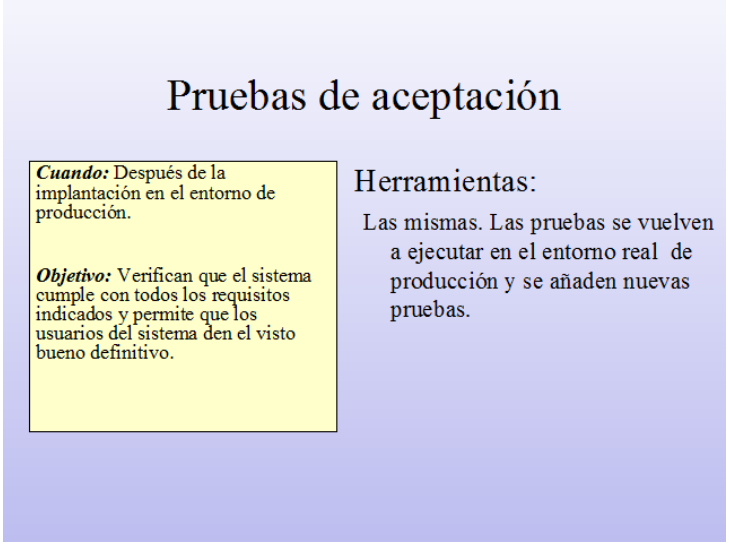


Las verificaciones de las pruebas de implantación y las pruebas del sistema tienen muchos puntos en común al compartir algunas de las fuentes para su diseño como pueden ser los casos para probar el rendimiento (pruebas de sobrecarga o de *stress).*

Pruebas de aceptación

Las pruebas de aceptación → son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario.

La validación del sistema se consigue mediante la realización de pruebas de caja negra que demuestran la conformidad con los requisitos y que se recogen en el plan de pruebas, el cual define las verificaciones a realizar y los casos de prueba asociados. Dicho plan está diseñado para asegurar que se satisfacen todos los requisitos funcionales especificados por el usuario teniendo en cuenta también los requisitos no funcionales relacionados con el rendimiento, seguridad de acceso al sistema, a los datos y procesos, así como a los distintos recursos del sistema.



Pruebas de regresión

Se pueden considerar como el subconjunto de pruebas planificadas que se seleccionan para ser ejecutadas , generalmente de forma automática y periódicamente en cada nueva liberación del producto/software , teniendo como objetivo la verificación de que el producto no haya sufrido regresiones.

Este tipo de cambio puede ser debido a prácticas no adecuadas de control de versiones, falta de consideración acerca del ámbito o contexto de producción final y extensibilidad del error que fue corregido (fragilidad de la corrección), o simplemente una consecuencia del rediseño de la aplicación.