Trabajo Final MOO

OBJETIVO GENERAL
Desarrollar el modelamiento de una aplicación para solucionar un problema que requiera sistematizar la información en una empresa del sector industrial o de servicios

OBJETIVOS ESPECÍFICOS
 Realizar ingeniería de requerimientos para el desarrollo de una aplicación, bajo la metodología de la licitación de requisitos de software.
 Modelar una aplicación utilizando leguaje de modelado unificado UML.
 Diseñar interfaces que simulen el comportamiento de la solución

Modelamiento de un Sistema de compras de boletos de avión para una aerolínea utilizando el proceso unificado para el desarrollo de software

Nombre de los Estudiantes
XXXXX
XXXXX
XXXXX

Nombre del Profesor
Luis Enrique Peña Mendoza

Fecha de entrega
XXXXX

UPC - Universidad Peruana de Ciencias Aplicadas

Tabla de contenido

Ejemplo

1. Introducción

La aplicación de metodologías y prácticas ágiles dentro de empresas dedicadas al desarrollo de software, es en la actualidad una realidad que se viene acrecentando en los últimos años alrededor del mundo.

Son muchas las personas y empresas dedicadas al desarrollo de software que se enfrentan hoy por hoy con el dilema de ser o no ser ágiles. Bien sea porque sus actuales metodologías de desarrollo no dan los resultados esperados o bien porque desean incursionar en prácticas que están de moda a lo largo y ancho de las comunidades de desarrollo en todo el mundo, lo cierto es que el auge de las metodologías ágiles es un tema que apasiona y hace reflexionar a todo aquel que se encuentre inmerso dentro del proceso de desarrollo de software.

Este trabajo presenta un enfoque práctico para la elección y adecuación de metodologías ágiles de desarrollo de software a un proyecto real: El desarrollo de un sistema de reservas para una agencia de viajes usando tecnología J2EE.

1.1. Justificación del proyecto

El sector de las Agencias de Viaje ha vivido una crisis desde la aparición de Internet.

Según el Foro Internacional de Turismo, las compras de viajes por Internet han crecido de forma significativa hasta el punto de llevar a la crisis a las agencias de viaje tradicionales. La tendencia se presenta como consolidada aunque ello no tiene por qué suponer la desaparición de estas agencias sino un cambio en su concepción original de manera que se puedan adaptar a los nuevos tiempos.

Las compañías aéreas y las centrales de reservas de hoteles fueron algunas de las primeras entidades en usar grandes sistemas en red para gestionar la reserva y venta de sus productos. Estos sistemas conectaban los ordenadores centrales con los terminales de las Agencias de Viajes. Éstas por su parte vendían luego el producto al público final llevándose una comisión por la gestión.

La incorporación de las agencias a la red ha supuesto la liberalización masiva del sector, y cualquier competencia en la oferta es beneficiosa para el cliente ya que la competitividad conlleva una bajada de precios y una mejora del producto.

Con la aparición de Internet las compañías aéreas y las centrales de reservas de hoteles han extendido sus redes de computadores que antes únicamente servían a agencias de viajes. En muchos casos hoy en día sale mas barato comprar un billete por Internet, que en una Agencia. Y las compañías aéreas recortan cada vez más sus comisiones a las Agencias.

Las compañías de viajes (ferroviarias, aéreas, de alquiler de coches…), los hoteles, guías y otros muchos servicios que antes dependían de una agencia u organizador de viajes, ahora pueden ofertarse directamente en la red, sin intermediarios y sin monopolios que limiten su actividad comercial, de manera que se ponen en manos de una buena gestión para ofrecer la mejor opción al cliente y del marketing.

Internet es donde se compra, vende y se contratan la inmensa mayoría de los viajes, cualquiera con conexión puede hacerlo con cuatro clicks.

Las Agencias de viajes están pues en crisis, necesitan reinventarse, ya no pueden ser intermediarios en la venta de billetes de proveedores. Necesitan poder competir en este nuevo mercado de venta por Internet.

El producto diferenciado que ofrecen las agencias de viajes es el paquete turístico, que para poder ser un producto alternativo a las ofertas de productos individuales de transporte y alojamiento, debe contar con un sistema de reservas por Internet que supere al de las compañías aéreas y centrales hoteleras, un sistema de reservas que ofrezca al público los productos que las agencias elaboran.

El sistema de reservas que elaboraremos en este proyecto contará con una ventaja competitiva frente a los sistemas de las grandes mayoristas tradicionales. Nuestra tecnología estará basada en la potencia de Java y J2EE y utilizará metodologías ágiles apoyadas en tests automatizados con las que podremos ser más dinámicos frente a los cambios que puedan surgir en el futuro.

1.2. Objetivos del trabajo

Con este proyecto pretendemos explorar tanto objetivos técnicos como metodológicos.

1.2.1. Objetivos técnicos

Para poder competir en este nuevo mercado de venta de viajes por Internet, necesitamos por una parte que la aplicación sea extremadamente **rápida** en la gestión de reservas de paquetes túrisiticos. Para ello debemos limitar al máximo la contención en las transacciones y asegurar al mismo tiempo su corrección, o lo que es lo mismo, impedir interferencias con otras transacciones en curso por ausencia de bloqueos.

Por otro lado, también será deseable que la aplicación sea **testable** con facilidad, para ello construiremos una arquitectura en la que los objetos de negocio se puedan ejecutar tanto fuera como dentro del contenedor J2EE, para así poderlo someter a todas las pruebas necesarias, sin la complicación, y lentitud añadida de arrancar un servidor J2EE.

Otra característica necesaria y derivada de la anterior, es construir una arquitectura basada en el paradigma de la orientacion a objetos, conociddo como **Open / Closed principle**. Este principio postula que el código este abierto para la extension pero cerrado para la modificación ya que cada vez que modifiquemos

código corremos el riesgo de estropearlo. Por ello debemos utilizar las tecnicas de herencia y composición para añadir funcionalidad a nuestros programas.

2. Especificación y requerimientos

2.1. Información inicial

Queremos crear un sistema reservas on-line para una agencia de viajes. El negocio de una agencia de viajes es la venta de paquetes turísticos que consisten en una combinación de diferentes servicios (transporte, alojamiento, excursiones, guías, etc.…) en un solo producto.

El paquete turístico consta de un código interno, un nombre, un precio por persona, una descripción, una presentación más o menos elaborada y un número de plazas ofertadas.

El sistema de reservas debe ofrecer un catálogo de paquetes turísticos por el que se pueda buscar por diferentes criterios, como fechas de salida, duración del viaje, precio, etc.

Cuando un cliente hace una reserva, el sistema debe actualizar el número de plazas disponibles, restando las que el cliente ha reservado, evitando que se puedan solicitar más plazas de las existentes.

El sistema debe guardar con la reserva, el nombre del cliente, el paquete solicitado y el número de plazas que quiere.

El sistema debe permitir la inclusión futura con de una interfaz con medios de pago electrónico.

El sistema tiene que permitir la introducción de paquetes turísticos en el catálogo, su modificación, consulta y borrado. La modificación y el borrado solo se podrán hacer si no se ha vendido ninguna reserva.

El sistema tiene que permitir el registro de los clientes donde introducirán información sobre sus datos personales, tarjetas de crédito, etc. También la modificación, consulta y borrado.

El sistema tiene que manejar de manera óptima las transacciones, minimizando bloqueos y asegurando la consistencia.

2.2. Glosario

Oferta: Producto en venta por la agencia de viajes

Cliente: Usuario de la aplicación registrado

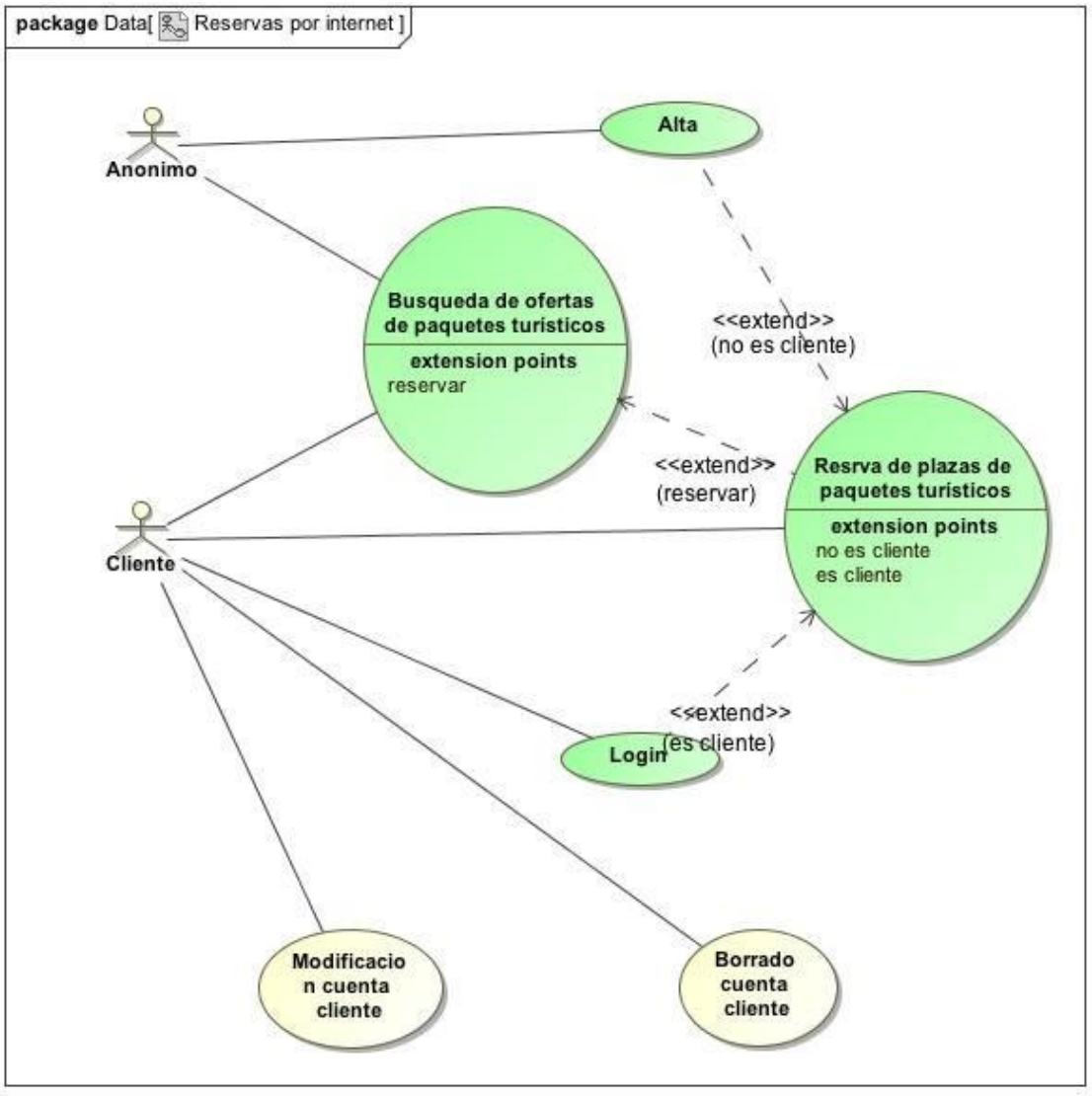
Reserva: Solicitud de venta de una oferta

Paquete Turístico: Agregado de servicios turísticos que vende una agencia de viajes

2.3. Modelo del dominio

- Oferta:
    - Nombre
    - Descripción
    - Fecha de Inicio
    - Fecha de Fin
    - Duración
    - Plazas Disponibles
    - Plazas Totales
    - Fecha límite de compra
- Reserva
    - Numero de plazas reservadas
    - Oferta
    - Cliente
- Cliente
    - Nombre
    - Apellidos
    - Login
    - Password
    - Fecha de Alta

2.4. Diagrama de casos de uso



2.5. Documentación textual de los casos de uso

Explicación del modelo seguido para documentar los casos de uso

Se ha seguido un modelo de Alistair Cockburn, gran especialista en metodologías de desarrollo.

La única pequeña dificultad de la estructura de estos casos de uso es el seguimiento del flujo. Como sabemos, una interacción con un sistema puede tener ramificaciones, y estas suelen ser difíciles de explicar. Trataremos de mostrar como con un ejemplo como las expresa este modelo

Caso de uso:
Sistema de alta seguridad de acceso a control de lanzamiento de misiles
Flujo principal:
1 El usuario se sienta a los mandos y pulsa el botón de acceso

2 El sistema pregunta su nombre de usuario y contraseña

3 El usuario introduce su nombre y contraseña

4 El sistema comprueba sus credenciales y le permite el acceso al control de lanzamiento de misiles

Flujo alternativo:

*a El usuario pulsa el botón 'ABORT'

1 El sistema vuelve a la posición standby

4a El sistema comprueba que sus credenciales son erróneas

1 El sistema devuelve al usuario al punto 2 informándole del error

4b El sistema detecta que el usuario ha introducido sus credenciales erróneamente tres veces

1 El sistema bloquea la cuenta de usuario

2 El sistema cierra las compuertas blindadas del edificio

3 El sistema abre las válvulas de gas paralizante

4 El sistema dispara la alarma de INTRUSO EN INSTALACIONES

Explicación:

El flujo principal se numeran los pasos

En el flujo alternativo se explican las ramificaciones. El primer número es el paso del flujo principal, la letra es la referencia de la ramificación de ese paso. Si un paso 1 se puede hacer de 3 formas diferentes, en el flujo alternativo tendrá tres ramas 1a , 1b, 1 c.

Dentro de la rama del flujo alternativo los pasos se numeran empezando por el uno.

2.5.1. Búsqueda de ofertas de paquetes turísticos

Actores:

Usuario

Flujo principal:

1) El usuario elije la opción de búsqueda de oferta.

2) El sistema muestra una pantalla con los criterios de selección de ofertas:

- nombre (criterio: contiene palabra)

- descripción (criterio: contiene palabra)
- fechaInicio (criterio: mayor o igual que)
- fechaFin (criterio: menor o igual que)
- duración (criterio: igual a con dos días por arriba y dos días por abajo)
- plazasDisponibles (criterio: mayor que)

3) El usuario elije uno o varios de los criterios de búsqueda y selecciona el filtro.

4) El sistema muestra una pantalla con los resultados de la búsqueda. En ningún caso mostrará aquellas ofertas para las que no existan plazas disponibles, ni cuya fecha límite de compra sea mayor o igual a la actual.

5) El usuario elige una de las ofertas.

6) El sistema muestra el detalle de la oferta.

Flujo alternativo:

*a) El usuario cancela en cualquier momento.

1) el sistema devuelve a la pantalla principal

4a) El sistema detecta que no se ha seleccionado ningún criterio de búsqueda

1) El sistema informa del error y devuelve al punto 2

4b) El sistema detecta que alguno de los campos no contienen datos válidos para la búsqueda. Ej. las fechas no son fechas válidas, el campo duración o el campo plazasDisponibles no son números

1) El sistema informa del error y devuelve al punto 2

Puntos de extensión:

7) El usuario elije reservar la oferta mostrada en detalle.

1) El sistema enlaza con el caso de uso de reserva de plaza.

Requisitos no funcionales:

El sistema debe ser rápido en devolver los resultados. Esto implica que la transacción debe permitir hacer lecturas sucias, ya que es mas importante la velocidad de respuesta que la exactitud del contenido.

Temas pendientes:

Comprobar riesgo de *sql injection* al construir una cláusula *where* dinámica.

2.5.2. Reserva de plazas de paquetes turísticos

Actores:

Usuario

Disparador:

El usuario ha seleccionado una de las ofertas existentes que tienen disponibilidad de plaza en ese momento.

Precondición:

El usuario está registrado.

Flujo principal:

1) El sistema presenta una pantalla con los datos de la oferta:

- código interno (no se mostrará) (no editable)
- nombre (no editable)
- descripción (no editable)
- fechaInicio (no editable)
- fechaFin (no editable)
- duración (no editable)
- número de plazas disponibles (no editable)
- número de plazas deseadas (editable)

2) El usuario introduce el número de plazas deseado

3) El sistema registra la reserva y notifica al usuario el éxito de la operación.

Flujo alternativo:

1-2 a) El usuario cancela la operación

1) El sistema devuelve a la pantalla principal

2a) El número de plazas que solicita el usuario es 0 o nulo

1) El sistema informa del error y vuelve al punto 1 (con datos actualizados).

2b)El número de plazas que solicita el usuario es superior al número de plazas disponibles

1) El sistema informa del error y vuelve al punto 1 (con datos actualizados).

Poscondición:

El usuario tiene N reservas de la oferta O

Requisitos no funcionales:

Las transacciones deben ser rápidas y correctas., por lo que deben bloquear la lectura de plazas para luego poder actualizarla, restando las plazas, pero el mínimo tiempo posible para permitir que otras transacciones operen sobre el mismo registro.

La transacción tiene que ser segura, no puede ser falsificable alterando parámetros de la petición.

2.5.3. Login

Disparadores:

El usuario elige hacer una reserva y no esta registrado en el sistema.

El usuario elige la opción de registrarse en el sistema.

Flujo principal:

1) El sistema muestra una pantalla con los campos de:

- nombre de usuario
- password

2) El usuario escribe su nombre y password.

3) El sistema comprueba que el usuario existe y su contraseña es correcta, por lo que devuelve al usuario al punto donde se originó el evento, es decir pantalla de inicio o a la reserva que estaba apunto de reservar.

Flujo alternativo:

1-2 a) El usuario cancela la operación.

1) El sistema devuelve al usuario al menú principal

3a) El sistema detecta que el usuario no existe o que la contraseña es incorrecta.

1) El sistema notifica el error y devuelve al usuario al punto 1 con los datos introducidos por el usuario.

Requisitos no funcionales:

En ningún caso debe intercambiarse entre el navegador y el servidor el nombre de usuario y/o la contraseña una vez pasada la validación, para evitar suplantación de identidad o robo de contraseña.

Temas pendientes:

Hacer que el proceso de autenticación vaya encriptado con SSL (estimar el coste)

Hacer que el password de usuario vaya encriptado dentro de la base de datos (estimar coste)

2.5.4. Mantenimiento cuenta de usuario

2.5.4.1. Alta de usuario

Disparadores:

Desde pantalla de login el usuario elige la opción de darse de alta en el sistema.

Desde la pantalla de mantenimiento de cuenta de usuario, el usuario elige la opción de darse alta.

Flujo principal:

1) El sistema muestra una pantalla con los campos de (todos obligatorios):

- nombre (campo obligatorio)
- apellidos (campo obligatorio)
- login (campo obligatorio)
- password (campo obligatorio)

2) El usuario rellena los datos.

3) El sistema registra los datos en la base de datos, rellenando una fecha de alta y devuelve al usuario a la pantalla de login comunicando el éxito de la operación.

Flujo alternativo:

1-2 a) El usuario cancela la operación.

1) El sistema devuelve al usuario a la pantalla de inicio.

3a) El sistema detecta que no se han rellenado todos los campos

1) El sistema indica que campos son los que no se han rellenado y devuelve al punto 1 con los datos que ha llenado el usuario.

3c) El sistema detecta que ya existe un usuario con el mismo login

1) El sistema comunica el error e insta al usuario a cambiar el login

2) El sistema devuelve al usuario al punto 1

Requisitos no funcionales:

En ningún caso debe ser visible el nombre de usuario y/o la contraseña una vez pasada la validación.

Temas pendientes:

Hacer que el proceso de alta vaya encriptado con SSL (estimar el coste)

Hacer que la contraseña de usuario vaya encriptada dentro de la base de datos (estimar coste)

Requerir una contraseña con un nivel de seguridad aceptable es decir con un número de caracteres suficientes, que contenga números, etc. (estimar coste)

2.5.4.2. Consulta de usuario

Precondición:

El usuario se ha logado en el sistema previamente con lo que tiene visible la opción de mantenimiento de cuenta de usuario.

Flujo principal:

1) El usuario elige la opción de mantenimiento de usuario

2) El sistema recupera los datos del usuario y muestra una pantalla con los campos de:

- nombre
- apellidos
- login
- password

Flujo alternativo:

*a) El usuario cancela la operación

1) El sistema devuelve a la pantalla de inicio

Puntos de extensión:

Borrar (desactivar) usuario.

Modificar usuario.

2.5.4.3. Modificación de usuario

Disparadores:

Desde la pantalla de mantenimiento de cuenta de usuario, el usuario elige la opción de modificar su cuenta.

Precondición:

El usuario se ha logado en el sistema previamente con lo que tiene accesible la opción de mantenimiento de cuenta de usuario.

Flujo principal:

1) El sistema recupera los datos del usuario y muestra una pantalla con los campos de:

- nombre
- apellidos
- login
- password

2) El usuario rellena los datos.

3) El sistema registra los datos en la base de datos y devuelve al usuario a la pantalla inicial.

Flujo alternativo:

1-2 a) El usuario cancela la operación

1 El sistema devuelve a la pantalla anterior

3a) El sistema detecta que no se han rellenado todos los campos

1) El sistema indica que campos son los que no se han rellenado y devuelve al punto 1

3c) El sistema detecta que ya existe un usuario con el mismo login

1) El sistema comunica el error e insta al usuario a cambiar el login

2) El sistema devuelve al usuario al punto 1

Temas pendientes:

Hacer que el proceso de modificación vaya encriptado con SSL (estimar el coste).

Hacer que la contraseña de usuario vaya encriptada dentro de la base de datos (estimar coste).

Requerir una contraseña con un nivel de seguridad aceptable es decir con un número de caracteres suficientes, que contenga números, etc. (estimar coste).

2.5.4.4. Borrado de usuario

Precondición:

El usuario se ha logado en el sistema previamente con lo que tiene accesible la opción de mantenimiento de cuenta de usuario.

Flujo principal:

1) El usuario elige borrar o darse de baja

2)El sistema pide confirmación

3) El usuario acepta

4) El sistema escribe en *fecha de baja* el día de hoy, con lo que queda registrada la baja. (baja lógica) y devuelve al usuario a la pantalla de inicial de la aplicación.

Flujo alternativo:

1-3) El usuario cancela la operación

1 El sistema vuelve a la pantalla de detalle de usuario.

2.5.5. Mantenimiento de ofertas

Sistema opcional que estará situado dentro de la intranet del cliente, por lo que no será en principio necesario ninguna consideración de seguridad especial.

2.5.5.1. Alta de oferta

Flujo principal:

1) El usuario elige la opción de crear una nueva oferta.

2) El sistema muestra una pantalla con los campos:

- Nombre (campo obligatorio)
- Descripción (campo obligatorio)
- fechaInicio (campo obligatorio)
- fechaFin (campo obligatorio)
- plazasTotales (campo obligatorio)
- fechaLimiteDeCompra (campo obligatorio)

3) El usuario rellena los datos.

4) El sistema registra los datos en la base de datos, inicializa plazasDisponibles = plazasTotales, y devuelve al usuario a la pantalla de login comunicando el éxito de la operación.

Flujo alternativo:

1-3 a) El usuario cancela la operación

1) El sistema devuelve al usuario a la pantalla de inicio.

4a) El sistema detecta que no se han rellenado todos los campos

1) El sistema indica que campos son los que no se han rellenado y devuelve al punto 2 con los datos que rellenó el usuario.

4b) El sistema detecta inconsistencias en los datos. Ej. fechaInicio mayor o igual a fechaFin. fechaLimite de compra menor que hoy().

1) El sistema informa de los errores y devuelve al punto 2 con los datos que rellenó el usuario.

2.5.5.2. Consulta de oferta

Flujo principal:

1) El usuario elije la opción de búsqueda de ofertas

2) El sistema muestra una pantalla con los criterios de selección de ofertas:

- nombre (criterio: contiene palabra)
- descripción (criterio: contiene palabra)
- fechaInicio (criterio: mayor o igual que)
- fechaFin (criterio: menor o igual que)

3) El usuario elije uno o varios de los criterios de búsqueda y selecciona el filtro.

4) El sistema muestra una pantalla con los resultados de la búsqueda.

5) El usuario elige una de las ofertas.

6) El sistema muestra el detalle de la oferta.

Flujo alternativo:

*a) El usuario cancela en cualquier momento.

1) El sistema devuelve a la pantalla principal.

4a) El sistema detecta que no se ha seleccionado ningún criterio de búsqueda

1) El sistema informa del error y devuelve al punto 2

4b) El sistema detecta que alguno de los campos no contienen datos validos para la búsqueda. Es decir las fechas no son fechas válidas o el rango no es correcto.

1) El sistema informa del error y devuelve al punto 2.

Requisitos no funcionales:

En el listado deben salir todas las ofertas pero deberíamos evitar las lecturas sucias, ver el nivel de aislamiento optimo de transacciones.

2.5.5.3. Modificación y baja de oferta

Queda fuera del ámbito del proyecto por ser demasiado complejo. El sistema necesitaría detectar si se ha hecho alguna reserva, si es así dependiendo del dato que se quiera modificar, deberá permitir o no la modificación. También debería avisar a los usuarios y permitir que cancelasen la reserva, etc.

En cuanto a las transacciones, cuando menos, tendría que bloquear durante la modificación, la inserción de nuevos registros.

2.6. Requisitos de la interfaz de usuario

La interfaz de usuario será HTML, ya que es una aplicación Web. Prescindiremos de javascript, porque el usuario lo puede desactivar. Evitaremos los diseños complicados para que el tratamiento de pruebas con httpUnit sea posible.

Haremos hincapié en que la navegación esté muy controlada. La navegación ha de permitir buscar los viajes y solo cuando se seleccione uno, nos pedirá logarnos, o darnos de alta en el sistema.

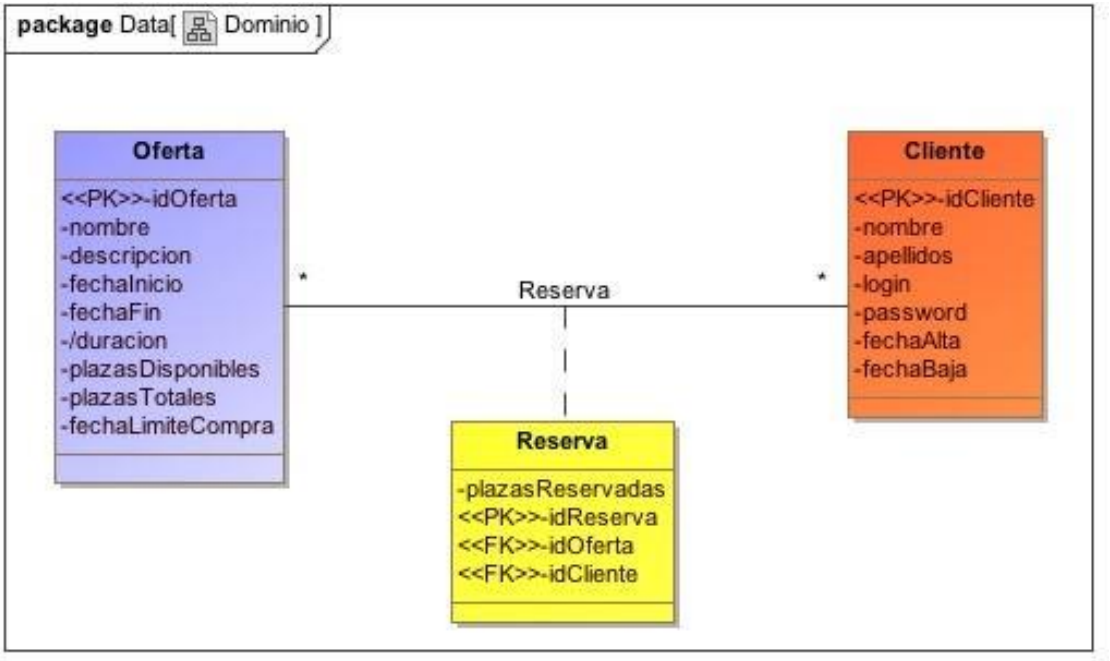
El sistema ha de ser amigable, en particular a la hora de mostrar los errores de validación de los formularios. Estos tienen que presentarse al lado del mismo campo que originó el error.

Usaremos css para separar el diseño de los datos.

Cuidaremos al máximo la seguridad evitando el paso de parámetros sensibles al navegador, como por ejemplo datos de la cuenta de usuario.

3. Análisis

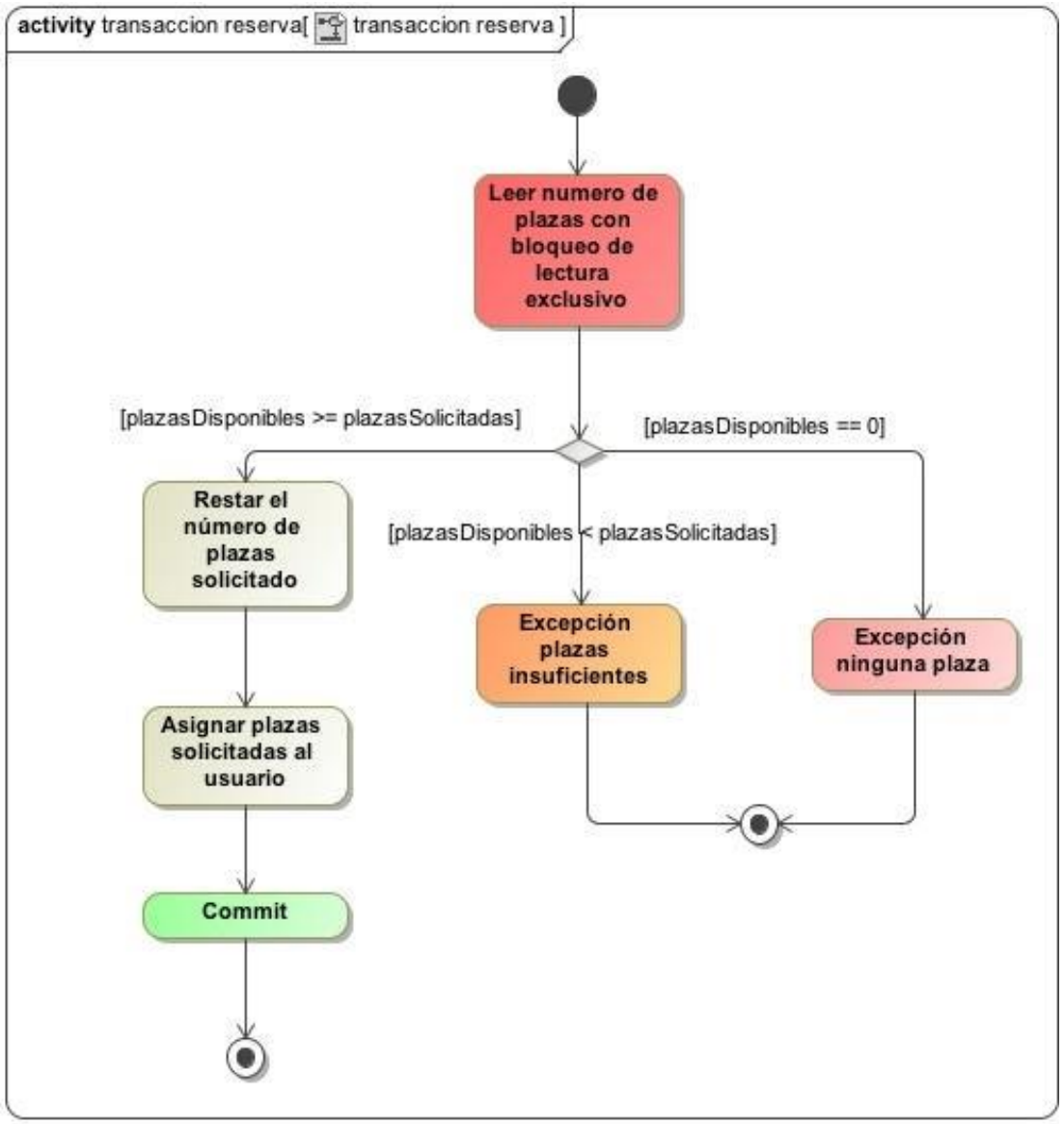
3.1. Identificación de las clases de entidades y sus atributos



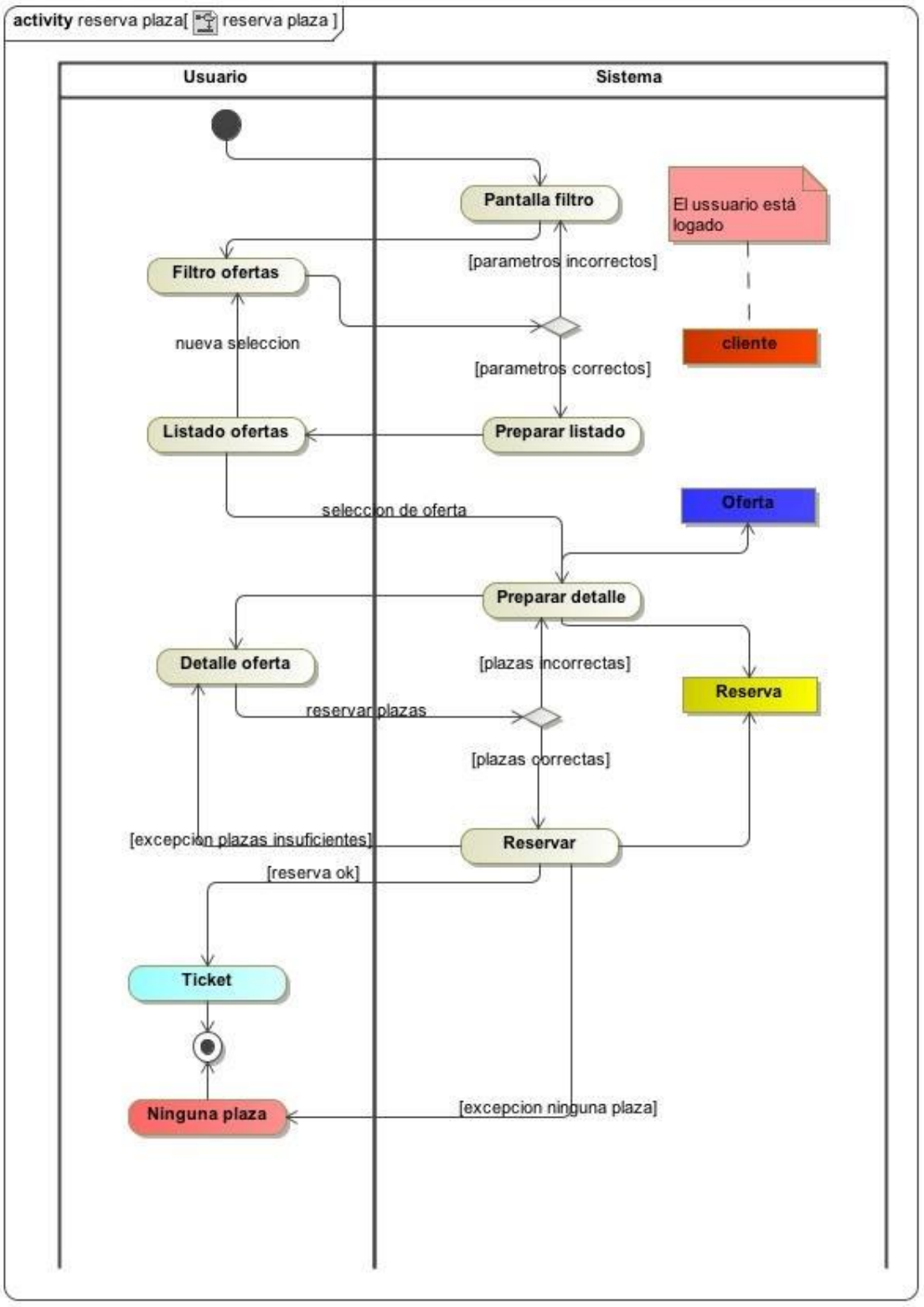
3.2. Diagramas de estados

3.3. Diagramas de actividades

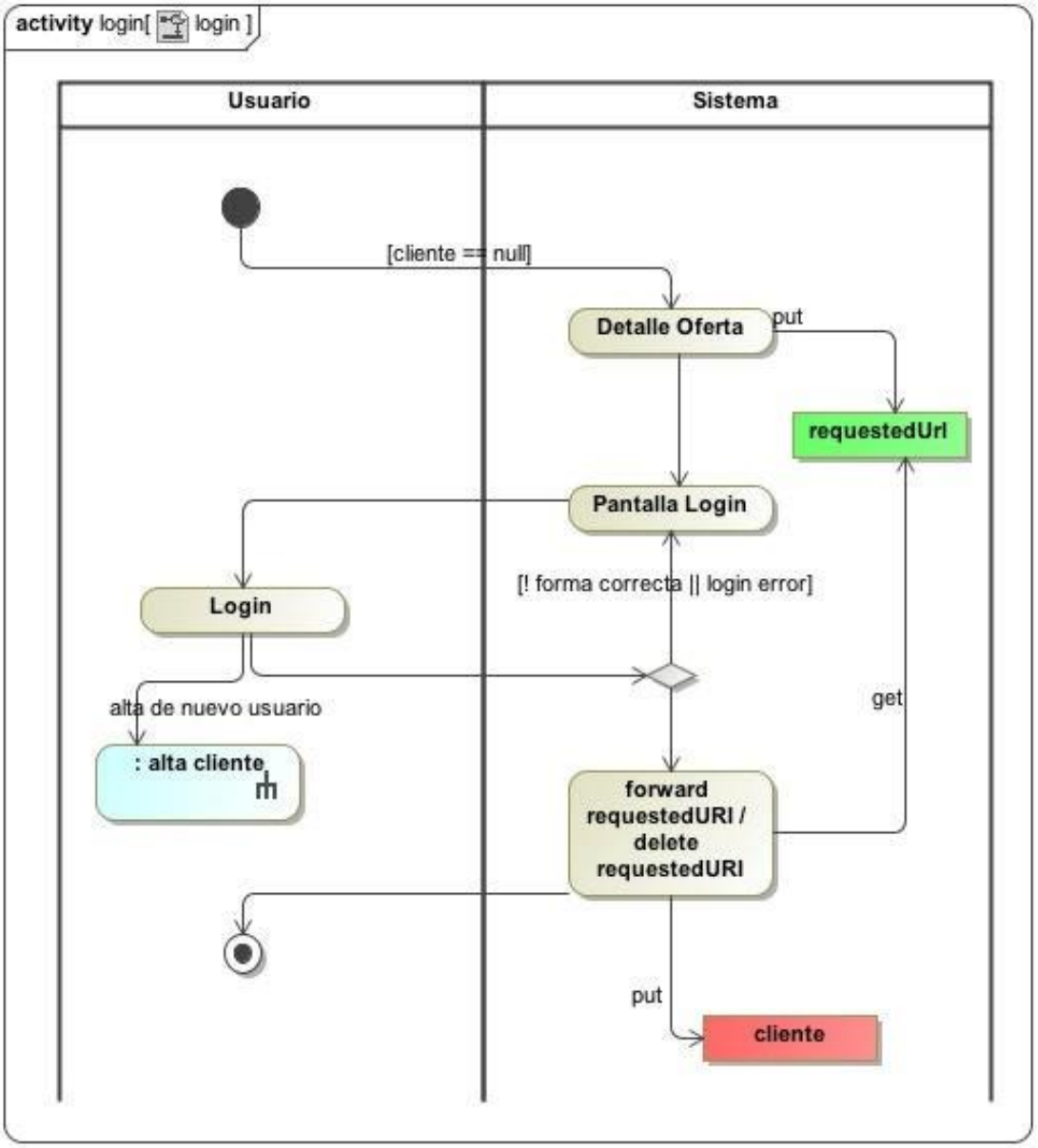
3.3.1. Transacción de reserva de plaza



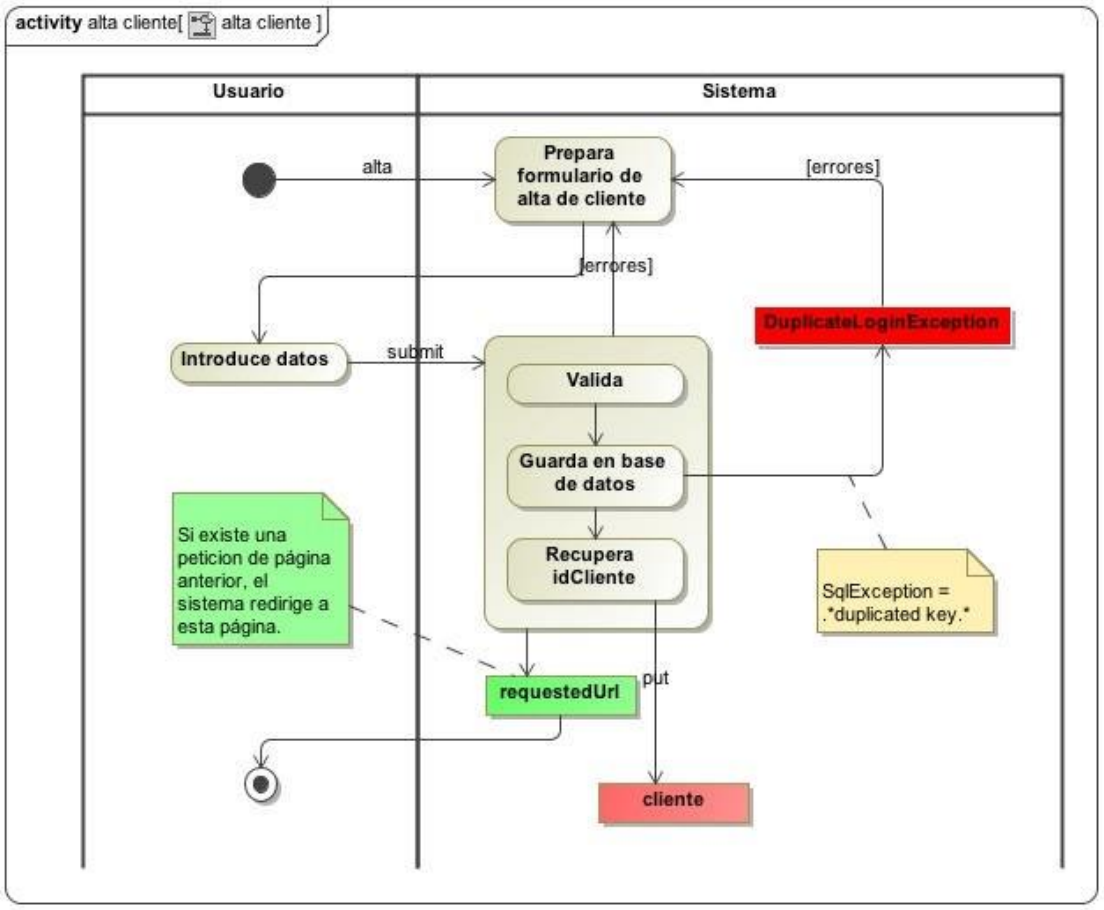
3.3.2. Reserva de plaza (búsqueda + reserva)



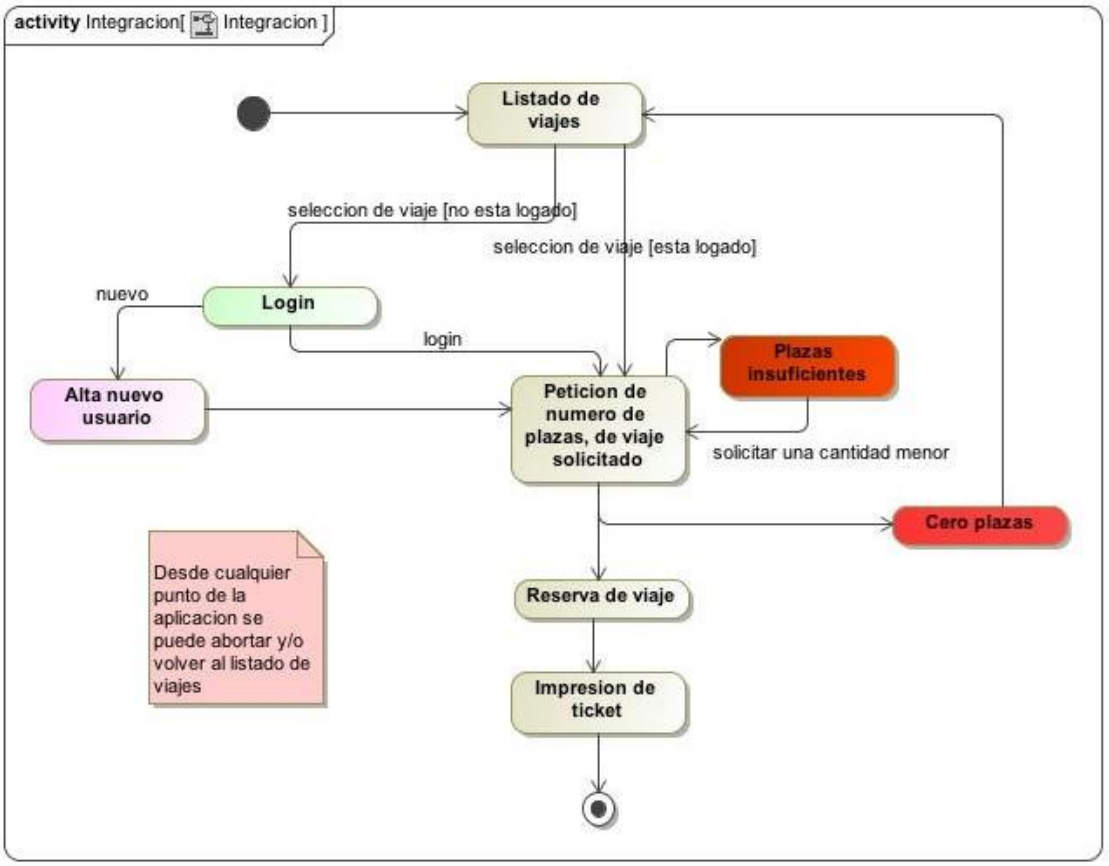
3.3.3. Login


activity login[login]
Usuario
Sistema
[cliente == null]
Detalle Oferta
put
requestedUrl
Pantalla Login
[! forma correcta || login error]
Login
alta de nuevo usuario
: alta cliente
get
forward requestedURI / delete requestedURI
put
cliente

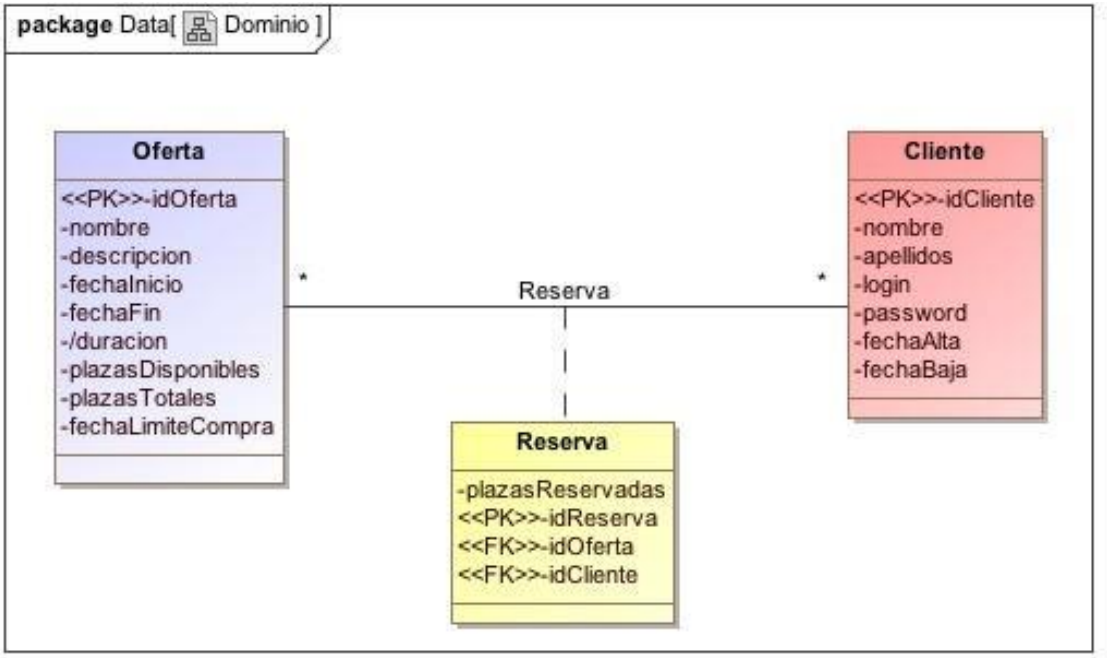
3.3.4. Alta de usuario



3.3.5. Integración (búsqueda, Login, alta, reserva)



3.4. Modelo de datos



4. Conclusiones

Uso de metodología ágil

No hemos podido aplicar una metodología de desarrollo ágil completamente, porque nos faltaba la parte fundamental que es tener al cliente dentro del equipo de desarrollo, para verificar el producto y proponer cambios. Pero hemos usado otras prácticas que si podemos evaluar, como los ciclos de desarrollo cortos y las pruebas automatizadas.

Ciclos de desarrollo cortos

Hemos afrontado el proyecto como una sucesión de pequeños subprogramas completos.

Como primera consecuencia, el centrarnos en una pequeña parte nos ha permitido definir los casos de uso perfectamente, sin flecos, en especial la interacción con el sistema de forma excelente.

La notación de casos de uso de Alistair Cockburn y los diagramas de actividad han demostrado ser unas excelentes herramientas para esta labor.

Pruebas de unidad automatizadas

Han sido mucho mas laboriosas de hacer de lo que pensaba, creo que tardábamos hasta cuatro o cinco veces mas en hacer las pruebas que en escribir el código. Y escribir código que se pueda probar ya es de por si mas laborioso.

Sin embargo creemos que han sido muy útiles, apenas hemos pasado tiempo depurando código. Y no hemos perdido nada de tiempo en la integración.

Creemos que en el punto 1.3.2.1. queda demostrado la necesidad de hacer pruebas de unidad, si se quiere probar el código. Solo con las pruebas de integración es imposible.

Hoy en día todavía se siguen dejando las pruebas para el final en los desarrollos, entonces solo podrán ser pruebas de integración. Y como hemos visto la

complejidad de una prueba completa de integración crece exponencialmente en función del número de componentes sobre los que opere, mientras que el coste de las pruebas de unidad es lineal

En definitiva, creemos que las pruebas de unidad deben ser un paso necesario en todo desarrollo de software.

Metodología de tests y herramientas.

Creemos que hemos logrado un buen sistema para hacer pruebas con Eclipse, la forma de separar las pruebas en un proyecto aparte es muy útil.

5. Bibliografía

JUnit Recipes , JBRainsberr Editorial Manning

Head First Design Patterns, Eric Freeman &Elisabeth Freeman Editorial O'Reilly

Piensa en Java, Bruce Eckel Editorial Prentice Hall

Core Java Volume II Advanced Features, Cay S. Horstmann Editorial PH PTR

Struts KICK START, James Turner and Kevin Bedell