Facultad de Informática

Universidad Nacional de La Plata

Refactorings para mejorar Procesos de Negocio en Aplicaciones Web

Tesis presentada para obtener el grado de Magister en Ingeniería de Software

Autora: Julia Camelier Carvajal

Directora: Dra. Alejandra Garrido

Co-Directora: Dra. Roxana Giandini

Septiembre 2013

Dedicatoria:

A mis padres que nunca me dejaron desistir y

A mi marido por siempre estar a mi lado

Agradecimientos:

A mi querida tía Catalina por corregir pacientemente este trabajo y a mis profesoras Alejandra y Roxana por hacer posible esta tesis y por poder superar la distancia.

ÍNDICE

INDICE DE FIGURAS

1.INTRODUCCIÓN

La técnica de *refactoring* es una técnica que viene siendo utilizada con suceso desde más de 20 años. Su principal objetivo es promover mejorías en determinado dominio sin comprometer o alterar el comportamiento inicial del mismo. La acción de "refactorizar" fue introducida inicialmente en el dominio del código fuente de software con el intuito de reestructurar la jerarquía de las clases de un diseño orientado a objeto [30]. Años más tarde, Fowler [12] desenvolvió un poco más la definición del término describiendo un catálogo de *refactoring*s para mejorar y/o eliminar lo que llamaba de "*bad smells*", siempre respectando la premisa de preservar el comportamiento del software. La técnica que empezó apenas focalizada en códigos orientados a objeto y tenía como objetivo mejorar calidades internas como mantenibilidad, extensibilidad y eficiencia, comenzó a ser extendida a distintos dominios debido a los buenos resultados obtenidos.

El dominio de las aplicaciones web se tornó un ambiente propicio para la aplicación de *refactorings* debido a su constante y rápida evolución. *Refactorings* en el dominio web se focaliza en proporcionar mejoras en los modelos de las aplicaciones web (modelos de navegación, presentación, contenido) trayendo beneficios tanto para la estructura interna de los modelos [43] como para mejorar calidad externas que son percibidas por el usuario, como por ejemplo la usabilidad [7][15].

La evolución de las aplicaciones web fue con el tiempo aumentando la complejidad de las mismas. Los sitios web en un principio eran páginas que se focalizaban puramente en exhibir datos e informaciones a través de internet, sin interacción del usuario. De a poco, el usuario fue adquiriendo más protagonismo y las aplicaciones fueron ampliando su campo de alcance más allá de páginas de contenido estático y empezando a ser fuente de operaciones más complejas e interactivas. La inclusión de los procesos de negocio (que se define como una secuencia de pasos que el usuario tiene que ejecutar en las aplicaciones web con el objetivo de completar una determinada tarea) en las aplicaciones web posibilitó que una serie de operaciones y transacciones pudiesen ser realizadas a través del ambiente web. Antiguamente no se podría imaginar que el usuario sería capaz de hacer las compras del mes sin tener que ir al supermercado o comprar un pasaje aéreo sin ir a la agencia de viajes, todo hecho a través de una computadora conectada a internet. Actualmente todo ese progreso es posible y viable debido a esta característica de las aplicaciones web que es soportar los procesos de negocio, trayendo con eso innumerables beneficios a los usuarios.

Más allá de todos los beneficios logrados, el hecho de manejar complejos procesos de negocio añadió nuevos desafíos y problemas a ser resueltos por los desarrolladores y diseñadores [2]. Fue demostrado que la introducción de los procesos de negocio llevó a las aplicaciones web a un nivel más alto y totalmente distinto de las páginas web convencionales. Tratar procesos de negocios como un conjunto de pasos navegacionales conlleva la introducción de problemas en el comportamiento de la página, que pueden afectar su usabilidad [5]. Por esa razón, los procesos de negocio deben ser definidos, diseñados y modelados por separado y antes de la etapa de implementación, asegurando de esta manera la calidad de la aplicación web [6].

Con la finalidad de definir adecuadamente estas nuevas aplicaciones web, algunas metodologías tuvieron que evolucionar con el intuito de modelar específicamente los procesos de negocio. Metodologías como OO-H [3], OOHDM [17][18], UWA [35][37] y UWE [20][21][23] ahora incluyen nuevos conceptos y modelos que son utilizados para definir en detalle el diseño de los procesos de negocio. UWE es la metodología que será

utilizada para modelar los diagramas en esta tesis. Se eligió esta metodología debido a la facilidad de modelado de los procesos de negocio donde son utilizados diagramas específicos para este fin aparte de la completa integración de los procesos en los otros modelos. UWE está basada en el lenguaje estándar UML, un lenguaje que tiene gran aceptación en el modelado de sistemas de diferentes dominios. Su enfoque de modelado de aplicaciones web se divide en varias etapas como: Análisis de Requerimiento, Modelo Contenido, Modelo de Proceso, Modelo de Navegación y Modelo de Presentación. UWE agrega un modelo específico, el modelo de proceso, para modelar los procesos de negocio de las aplicaciones web. Con excepción del modelo de contenido que es el único modelo estático, que no provee información relacionada con el proceso de negocio embebido, los otros fueron adaptados con el objetivo de representar los procesos de negocio.

Para ilustrar las aplicaciones web, fueron seleccionados dos tipos de aplicación que tiene vasto uso y gran popularidad actualmente, las páginas de *E-Commerce* y las páginas de *Home Banking*. Ambos tipos son ejemplos de aplicaciones web que soportan procesos de negocio en su estructura y por ser muy populares actualmente fueron elegidas como objeto de estudio en este trabajo, donde fueron identificados problemas de usabilidad y sugeridos *refactorings* para mejorar y/o eliminar estos problemas.

1.1. Motivación

Con la popularización de internet en los últimos años, las aplicaciones web fueron ganando cada vez más adeptos y usuarios. Tales aplicaciones fueron evolucionando y con ellas el rol manejado por los usuarios en este contexto. Si antes los usuarios eran pasivos y sólo utilizaban internet para tener acceso a informaciones y datos, hoy en día esta relación fue modificada. Con aplicaciones web cada vez más poderosas, los usuarios pueden interactuar más cada día y beneficiarse de las ventajas de poder realizar diversas actividades y transacciones a través de la computadora. Páginas de comercio electrónico (también conocidas como *E-Commerce*) donde se compran y venden los más variados productos, crecen y se multiplican velozmente debido al grande éxito con los usuarios, que adoptaron este nuevo modo de comprar sin salir de casa. Las páginas de *Home Banking* ya hicieron de la acción de realizar transacciones bancarias por medio de internet una rutina para gran parte de los usuarios que están conectados a la red y sin duda se puede afirmar que cambiaron radicalmente la forma de controlar y manejar las finanzas personales.

Aplicaciones como las páginas de comercio electrónico y las páginas de *Home Banking* son apenas un ejemplo de cómo las aplicaciones web se tornaron populares y ya son parte de la vida de muchos. La posibilidad de realizar diversas actividades y operaciones a través de internet llama la atención en un tema que muchas veces pasa desapercibido por muchos desarrolladores y diseñadores de aplicaciones web, la usabilidad. Para ser completados, los procesos de negocio exigen bastante interacción entre el usuario y aplicación. Por ejemplo, si el usuario desea comprar un pasaje en el sitio de una compañía aérea, éste tiene que ejecutar una serie de pasos hasta concluir su objetivo, como informar las fechas, el origen y el destino del viaje, informar datos del pasajero y las informaciones referentes al pago. Después de completadas exitosamente todas estas etapas, el proceso de negocio es finalizado con la compra del ticket. Tantos pasos y tantas etapas de los procesos deben ser exhibidos al usuario de manera amigable y deben ser intuitivos guiando el mismo y llevándolo a la conclusión de su objetivo en la aplicación. Sin embargo, muchas veces por subestimar la etapa de definición y diseño de los procesos de negocio, la usabilidad de la aplicación queda

seriamente comprometida haciendo que los usuarios encuentren problemas en ejecutar las operaciones ofrecidas en la página.

No es difícil encontrar diversas páginas con problemas de usabilidad hoy en día. Problemas en la navegación de la página, en los links y en la distribución del contenido son ejemplos recurrentes y que vienen siendo blanco de quejas por muchos usuarios. Más allá de los problemas comunes encontrados en los sitios web, existen también otros referentes a los procesos de negocio de las aplicaciones web. Cuando un proceso de negocio no está bien definido o bien integrado con el resto de la aplicación, su calidad se ve afectada impactando directamente al usuario que tiene que ejecutar el proceso y a la empresa que posee una aplicación web deficiente. Procesos muy complicados, poco intuitivos, de difícil acceso o con poca flexibilidad hacen que los usuarios tengan dificultades para ejecutar y por lo tanto concluir el proceso deseado en la página. Tales problemas hacen que aspectos como la usabilidad y eficiencia de la aplicación web sean comprometidas, generando frustración para aquellos usuarios que utilizan la página en cuestión.

1.2. Propuesta de la Tesis

El objetivo principal perseguido en esta tesis consiste en proponer un catálogo de *refactorings* para mejorar aspectos como usabilidad, eficiencia y eficacia de los procesos de negocio de las aplicaciones web. Mientras otras publicaciones [7][15][18] identifican posibilidades de mejorías en los modelos de las aplicaciones web, esta tesis se focaliza en identificar problemas relacionados exclusivamente con la ejecución de los procesos de negocio y sugerir cambios para optimizar la experiencia del usuario mientras este navega por los procesos embebidos en las aplicaciones.

Los procesos de negocio impactan directamente en los otros modelos de la aplicación web (con excepción del modelo de contenido por tratarse del único que es estático y que no modela procesos), por eso fueron identificadas posibilidades de mejoras en todos los modelos que están directamente relacionados con la ejecución de los mismos. El catálogo de *refactorings* sugerido demuestra que pequeños cambios pueden optimizar la forma como el usuario ejecuta los procesos de negocio en la aplicación sin alterar los requerimientos y las reglas de negocio de los mismos. El proceso de negocio será analizado en detalle a través del Modelo de Proceso con el objetivo de identificar problemas que pueden ser solucionados con pequeños cambios en la estructura del mismo. Reordenar, remover o agregar actividades pueden facilitar la forma que el usuario ejecuta el proceso de negocio tornándolo más intuitivo y fácil de ser concluido. La forma como el usuario accede o navega por los procesos de negocio será analizada a través del Modelo de Navegación donde serán sugeridos cambios para hacer que los procesos sean más accesibles y que la navegación sea más intuitiva y benéfica para el usuario. En el Modelo de Presentación serán analizados aspectos de cómo el proceso es exhibido al usuario y cómo modificaciones en la presentación de las páginas del proceso pueden facilitar la ejecución del mismo haciendo que sea más eficiente y amigable.

El objetivo de esta tesis focaliza en analizar aplicaciones web de amplia utilización como los sitios de *Home Banking* y las páginas de *E-Commerce* identificando problemas relacionados a la ejecución de los procesos de negocio que afectan la usabilidad y eficiencia de las aplicaciones web. A través de la metodología UWE, serán diagramados todos los modelos de la aplicación web que son afectados por los procesos de negocio (modelo de proceso, navegación y presentación). Serán representados los modelos antes y después de los cambios sugeridos, detallando la motivación para la aplicación de los *refactorings* sugeridos, el mecanismo

que debe ser seguido para aplicar los mismos y la intención de mejoras logradas después de tener los cambios aplicados. En el caso que los *refactorings* se relacionen entre sí, estos serán mencionados también como los *refactorings* relacionados.

1.3. Contribuciones de la Tesis

Como será mencionado en el capítulo 7, el *refactoring* fue inicialmente pensado para mejorar aspectos relacionados al código fuente de un programa de software introduciendo mejoras que optimizarían aspectos como mantenibilidad y otras características internas del código. El campo de estudio de los *refactorings* fue ampliándose y extendiéndose a distintas áreas como base de datos, UML, HTML.

Esta tesis se focaliza en aplicar *refactorings* para mejorar la ejecución de los procesos de negocio en las aplicaciones web contribuyendo con los aspectos externos de las mismas como usabilidad. A continuación se describen las principales contribuciones de esta tesis:

- Análisis de calidades externas, como usabilidad y eficiencia, de los procesos de negocio embebidos en las aplicaciones web reales.

- Identificación de oportunidades de mejoras en los procesos de negocio de las aplicaciones web, con el objetivo de mejorar la experiencia del usuario al ejecutar tales procesos.

- Catálogo de *refactorings* exclusivamente dedicado a optimizar características de uso de los procesos de negocio, clasificados de acuerdo con modelo impactado.

- Demostración del impacto de los *refactorings* sugeridos en los diferentes modelos de la aplicación web.

- Implementación de los *refactorings* propuestos en aplicaciones reales y de amplio uso entre los usuarios.

Parte de estas contribuciones fueron presentadas en el artículo titulado "*Business Processes Refactoring to Improve Usability in E-Commerce Applications*" y serán publicadas en el *Electronic Commerce Research Journal* [85] este año.

Por último es menester hacer la salvedad de que muchos de los beneficios listados son el resultado de aplicación en más de un *refactoring* y que las mejoras en común pueden verse con más detalles en los cuadros presentados en el Anexo B.

1.4. Estructura de la Tesis

Con el fin de alcanzar los objetivos planeados, esta tesis está estructurada básicamente en tres partes descriptas a continuación:

La primera parte está compuesta de la base teórica, donde son abordados los conceptos fundamentales de temas relevantes e importantes para el entendimiento del trabajo propuesto. En el capítulo 2, "Usabilidad en Aplicaciones Web y Procesos de Negocio" se detallará el crecimiento del número y del uso de las aplicaciones web y como la usabilidad fue tornándose un tema de extrema importancia en este contexto.

Serán identificados problemas comunes encontrados en las aplicaciones web y sugerencias para mejorar la experiencia del usuario al navegar por estas páginas. Además en este capítulo, se hace una comparación de los sitios web tradicionales con las aplicaciones web que soportan procesos de negocios, así como una descripción de las metodologías más utilizadas para el modelado de aplicaciones web. Por último, se describirá en más detalles la metodología UWE, con sus particularidades y como los *refactorings* afectan los diferentes modelo de la aplicación web.

En la segunda parte de esta tesis se describe la importancia que tienen los procesos de negocio en las aplicaciones web y todos los beneficios ganados por estas aplicaciones. En el Capitulo 3, "*Refactorings* para mejorar los Procesos de Negocio en aplicaciones web" será definido el concepto de *Refactoring* y como este concepto fue extendido para otros dominios como para mejorar aspectos externos de las aplicaciones web, como la usabilidad. Todavía en este capítulo, serán descriptos los dos tipos de aplicaciones web que fueran elegidas como casos de estudio de este trabajo, las páginas de *E-Commerce* y las páginas de *Home Banking*. Los *refactoring* sugeridos en este trabajo se focalizan en mejorar problemas de usabilidad encontrados en estos dos tipos de aplicaciones específicamente.

La tercera y última parte de esta tesis se focaliza en describir un catálogo de *refactorings* focalizados en mejorar la ejecución de los procesos de negocio en las aplicaciones web y son clasificados de acuerdo con el modelo que afectan: el Capitulo 4, "*Refactoring* sobre el Modelo de Proceso", el Capitulo 5, "*Refactoring* sobre el Modelo de Navegación" y el Capitulo 6, "*Refactoring* sobre el Modelo de Presentación". Los tres capítulos ofrecerán una definición detallada de cómo el *refactoring* impacta en cada diagrama, las características que deberán ser preservadas, así como la lista de *refactorings* sugeridos para cada modelo, donde serán descriptos la motivación, los pasos necesarios para la aplicación del *refactoring*, el ejemplo utilizado apelando a los diagramas de la metodología UWE y las mejoras intencionadas para cada uno de los cambios realizados.

Para finalizar, en el Capitulo 7, "Trabajos Relacionados" serán descriptas y mencionadas publicaciones que están relacionadas con la temática abordada en esta tesis y que fueron referenciadas a lo largo de este trabajo. En el Capitulo 8, "Conclusión" se hará un cierre puntualizando las conclusiones alcanzadas con el desarrollo de esta tesis así como sugerencias de trabajos futuros basados en los resultados alcanzados.

2. USABILIDAD EN APLICACIONES WEB Y PROCESOS DE NEGOCIO

2.1. Aplicaciones Web

El término "Aplicación Web" posee varias definiciones en la literatura. En su libro, Wilson de Padua [70] describe las aplicaciones web como "productos de software o sistemas informáticos que utilizan arquitectura distribuida con el protocolo HTTP, siendo así accesibles por un navegador (*browser*)." En una definición más sencilla, Pressman [71] afirma que una aplicación web puede ser desde una página web sencilla, hasta un complejo sitio web. Las definiciones anteriores son tomadas como punto de arranque ya que describen un concepto primario de las aplicaciones web, sin embargo no se consideran suficientes para este trabajo ya que describen un concepto muy general y también por considerar las aplicaciones web como sencillas y simples, cuando en realidad estas aplicaciones están cada día más complejas. Con un concepto más acorde con lo se aborda en este trabajo, Conallen [69] describe las aplicaciones web como un sistema web que implementa lógica de negocio y donde la interacción del usuario afecta el estado del negocio de la aplicación.

En un estudio realizado por el Ministerio de Ciencia y Tecnología del Gobierno Federal de Brasil en 2001 [84] fue constatado que el desarrollo de aplicaciones web crece progresivamente, siendo que, de 433 empresas que desarrollan software, 133 (31,6%) desarrollaban páginas para Web, 123 (28,4%) estaban desarrollando aplicaciones de *E-Business* y 111 (25,6%) estaban desarrollando aplicaciones de *E-Commerce* [68]. En más de diez años los números crecieron bastante lo que confirma la tendencia de aplicaciones basadas en el ambiente web, que, según Shah y Shoaib [74], tiene un crecimiento de 300% por año.

Desde su invención hasta hoy, la tecnología para la creación de los sitios web evolucionó mucho en cantidad y calidad, e hizo de la web una herramienta con alto grado de popularidad, a la que pueden acceder desde los usuarios más avanzados a personas con rudimentarios conocimientos en computación. Hoy en día están disponibles en la red aplicaciones web de alta complejidad que permiten que los usuarios puedan realizar incontables tareas online. La variada gama de actividades que se puede realizar a través de internet actualmente conlleva que el número y la diversidad de usuarios que la utilizan crezca progresivamente requiriendo asimismo que sea cada vez más eficiente en cuanto a prestaciones y de mayor sencillez y facilidad de manejo para los usuarios. En este contexto y con una audiencia tan grande, que crece a cada día, la calidad de las aplicaciones web, y en especial su usabilidad, se ha convertido en una variable esencial para la funcionalidad y el suceso del sistema.

Como una ventana que puede ser vista por miles de usuarios en todo el mundo, las empresas han visto y ven en internet una posibilidad de ampliar los negocios, razón por la cual crean páginas web para establecer nuevos canales con clientes potenciales que existen del otro lado de la pantalla. Bancos, compañías aéreas, librerías, por nombrar algunas actividades comerciales, tienen su página web disponible, ofreciendo informaciones y servicios para sus usuarios, sus clientes virtuales. Esto sin mencionar la enorme cantidad de empresas de comercio electrónico, donde el *web site* se resume a la propia empresa. Sin embargo, la popularidad de la web no siempre significa que los usuarios estén conforme con lo que se les muestra, especialmente cuando se refiere a usabilidad [19].

2.2. Usabilidad en Aplicaciones Web

Según Offutt [73], hablar de usabilidad es referirse a uno de los tres principios de calidad usados en el desarrollo de aplicaciones web. Juntamente con la seguridad y confiabilidad, la usabilidad es por lo tanto un importante factor que debe ser considerado en el momento de la construcción de este tipo de aplicación, con la finalidad de garantizar la satisfacción del usuario. Debido a su importancia, existen diversas líneas de investigación que desarrollaron técnicas y metodologías para medir la usabilidad en las aplicaciones web, como es el caso de [75] donde los autores Conte y Travassos desarrollaron una técnica basada en evidencias llamada WDP (*Web Design Perspectives-based Usability Evaluation*) usada para inspeccionar y evaluar la usabilidad con el objetivo de atender los requisitos de las aplicaciones web. Otro ejemplo que se puede mencionar es la herramienta TOWABE (*TOol for Web Application usaBility Evaluation*) desarrollada por Itakura y Vergilio en [76] que integra diferentes técnicas de evaluación de usabilidad generando reportes que evalúa distintas perspectivas.

La importancia de la usabilidad en internet evolucionó con el pasar de los años. Si antes el usuario evaluaba la usabilidad de un producto después de la compra (como por ejemplo la compra de un producto software), en la actualidad el usuario evalúa la usabilidad antes de comprometerse a usar la aplicación o antes mismo de gastar dinero en posibles compras [14]. Con el número de sitios web disponibles que aumenta día a día, la variedad de páginas disponibles para la elección del usuario también crece, así como la competencia entre las empresas que poseen su página web disponible en la red. La usabilidad, en este contexto, se establece como un factor diferencial entre las páginas online disponibles y tiene el objetivo de mejorar la satisfacción y la experiencia del usuario mientras navega por estas aplicaciones, aumentando al mismo tiempo la ganancia de las empresas. En medio de tanta variedad, es probable que los usuarios elijan aquellos sitios que consideran amigables y que pueden entender con facilidad, antes que aquellos que son complicados y difíciles de usar.

Como se dijo, las aplicaciones web evolucionan rápidamente debido a su característica dinámica. Diversos factores son responsables de esta dinámica, como por ejemplo requerimientos nuevos para ampliar las funcionalidades de la página, modificaciones solicitadas por los clientes o incluso variaciones devenidas del avance de la propia tecnología. Todos esos factores de cambio condicionan la usabilidad de las aplicaciones y por lo tanto pueden afectar la forma como el usuario interactúa con la página [18].

Por su lado Krug [10] define la usabilidad como "estar seguro de que algo funciona bien y que alguien con una habilidad y experiencia promedio, o mismo abajo del promedio, pueda usar algo de acuerdo con su objetivo y sin frustrarse – como por ejemplo una página web". Si un sitio web de una empresa es difícil de entender, posee un diseño pobre y/o es complicado de utilizar, los usuarios tienden a no volver más a este sitio ya que, esas deficiencias les han impedido encontrar la información que buscaban. Esa ineficiencia lesiona los intereses de la empresa en cuestión pues los usuarios pasan a tener una representación negativa de la misma, lo que puede generar una posible pierda de los clientes frente a mejores *web sites* de la competencia, que, como afirma Nielsen [14], está a un clic de distancia.

Es importante resaltar que usabilidad depende de variables como: el tipo de aplicación web, el público que va a utilizar la aplicación, el avance de las tecnologías, el cambio de requerimientos, el aumento del número de usuarios en la página, el objetivo del usuario en la página, etc. No existe una fórmula que determine la usabilidad de un sitio web ya que esta está relacionada con todos esos factores que pueden variar de aplicación para aplicación. Con independencia de estas variables, el punto central de la usabilidad es la

información que brinde, por lo tanto, cuanto más auxilio preste el sitio web al usuario en función de encontrar la información que busca, más usable será este sitio [19][61].

Como se mencionó, hoy en día las grandes aplicaciones web suelen servir simultáneamente a millones de usuarios distribuidos por todo el mundo, por eso, crear y mantener una aplicación de este porte no es una tarea fácil [69]. Entre todas las variables y restricciones, los desarrolladores y arquitectos tienen la usabilidad como una preocupación extra. Garantizar la satisfacción de los usuarios en estas páginas es uno de los principales desafíos enfrentados por la equipe responsable por la creación de la aplicación. Introducir el análisis de la usabilidad de las páginas web durante el proceso de desarrollo de la misma evitará problemas de usabilidad, valga la redundancia, o al menos lograra minimizar sus efectos adversos.

Tanto la teoría como la práctica permiten observar que los problemas en el diseño de las aplicaciones web son recurrentes y su origen se debe a diversas razones, como por ejemplo:

- Gestión del desarrollo de las aplicaciones web como proyectos tradicionales y no como proyectos direccionados al usuario.

- Organización del sitio web reflejando la estructura de la compañía a quien pertenece, cuando el sitio debe ser estructurado para reflejar las tareas del usuario.

- Diseño de páginas sin consideración previa de circunstancias reales que pueden incidir sobre su usabilidad, como ser: la velocidad de internet o el hecho de que los usuarios no sepan usar la página, entre otras.

Al respecto, Nielsen [14] afirma que la usabilidad es un atributo de calidad que evalúa la facilidad de uso de las interfaces por parte del usuario y que, al mismo tiempo el termino refiere al método utilizado para mejorar la facilidad de uso durante el proceso de desarrollo.

Existen diferentes mecanismos que posibilitan detectar problemas de usabilidad durante el proceso de desarrollo, uno de ellos es hacer un testeo de usabilidad con usuarios reales. Al respecto Krug [10] afirma que si la empresa quiere desarrollar una página web de calidad, con buena usabilidad, es fundamental testearla, para lo cual propone que personas externas al equipo de desarrollo utilicen la página y proporcionen información acerca de las contrariedades que observan en su uso. Esto permite tomar en cuenta los problemas que pueden habérsele pasado por alto al equipo de diseño; o a inconvenientes surgidos del hecho de que no todos pensamos de la misma manera, ni utilizamos las páginas de igual forma, o para satisfacer las mismas necesidades.

La usabilidad es definida por cinco factores de calidad:

- **Facilidad de aprendizaje**
  Refiere a la simplicidad que experimentan los usuarios para realizar una tarea básica cuando entran por primera vez en la página web. Desde esta perspectiva se dice que la misma posee buena usabilidad cuando el usuario entra en ella por primera vez y es capaz de comprender su estructura y su lógica de funcionamiento y sabe qué hacer para completar su objetivo.

- **Eficiencia**
  Siendo la eficiencia, según el diccionario de la RAE[1], la "capacidad de disponer de alguien o de algo para conseguir un efecto determinado", la velocidad con que los usuarios puedan realizan una tarea básica, una vez que ya conocen el diseño de la página se convierte en un buen instrumento para inferir en qué medida dicha página es eficiente.

- **Memorabilidad**
  Se relaciona con la facilidad en recordar cómo realizar una tarea en una página web a pesar de no haberse producido ingreso alguno al sitio por un periodo de tiempo indeterminado.

- **Errores**
  Los usuarios, cuando navegan en una página, indefectiblemente comenten errores. Por lo tanto, es síntoma de buena usabilidad la posibilidad de predecir la criticidad de los errores y facilitar el pronto retorno a la aplicación una vez cometido el error.

- **Satisfacción**
  Refiere al comportamiento resultante a partir de la puesta en tensión entre el diseño de la página web y que tan agradable resulta su navegación al usuario.

Otro atributo de calidad a tener en cuenta es la utilidad de la página web, o sea si el *web site* hace u ofrece lo que el usuario necesita. De lo expuesto hasta aquí se puede acotar que, utilidad y usabilidad son igualmente importantes ya que, de nada sirve que la página sea fácil de usar, si en realidad no tiene lo que el usuario necesita que tenga, o bien tiene lo que usuario necesita pero este no puede usarlo porque la interface es muy complicada [11].

A pesar de no contar con una fórmula capaz de determinar el mayor o menor grado de usabilidad de un sitio web, la experiencia permite afirmar que existen determinados factores que ayudan para que unas aplicaciones sean más usables que otras, como por ejemplo, cuando la aplicación tiene contenido objetivo claro; si es de fácil uso; si el usuario puede navegar ágilmente por los links sin perderse; si permite que el usuario rápidamente encuentre lo que está buscando, principalmente cuando el sitio comunica el mensaje y objetivo inmediatamente a quien lo lee. Si el usuario encuentra dificultad en cualquiera de los puntos mencionados, hay altas posibilidades de que abandone la página [11]. Las personas, al ingresar en una página web, deben saber en pocos segundos exactamente de que se trata, sin pensar, e intuitivamente deben entender cómo usarla y cómo encontrar lo que desean [10].

Son muchos los problemas que hacen que una página web deje de ser usable. Algunos son bastante recurrentes y pueden ser identificados en diversas aplicaciones web, como:

- **Navegabilidad**: El usuario tiene dificultad de encontrar la información que desea en la página o tiene problemas para retornar a una página visitada anteriormente, lo que indica que el usuario puede estar perdido en el sitio web y no puede entender la lógica con que se presentan las informaciones en la página. Si la navegación está bien definida, el usuario tiene una visión obvia y clara del sitio, pero si la página posee problemas de navegabilidad, es muy común que los usuarios abandonen el sitio y desistan de entrar en su consulta ya que no pueden encontrar lo que buscan.

[1] http://lema.rae.es/drae/?val=eficiencia

Esto puede pasar porque el usuario no posee el conocimiento necesario para navegar por la página o porque la estructura del sitio no está organizada como el usuario espera o como debería [61].

- **Mecanismos de búsqueda ineficientes**: Cuando la navegación falla y el usuario es incapaz de encontrar lo que desea, el sistema de búsqueda es el mecanismo más utilizado por los usuarios como última forma de lograr su objetivo. Cuando este mecanismo no funciona como debería, sea porque el usuario no supo hacer la búsqueda (lo que indica que el mecanismo es confuso) o porque el algoritmo no está bien definido o porque los resultados de la búsqueda no son entendibles, los problemas de usabilidad de la página aumentan más todavía.

- **Textos extensos**: Escribir para web no es lo mismo que escribir para un diario o escribir un texto común. El contenido de la web deber ser objetivo, claro y conciso. Textos extensos hacen que los usuarios se aburran o que no los lean directamente. Es muy raro que los usuarios lean textos en la web, a menos que eso sea lo que estaba buscando. De no ser así, ellos apenas escanean la página buscando algún indicio que se asimile con la información que necesitan [10][61].

- **Links no identificables**: Muchas veces los links en las páginas web no son formateados como deberían y el usuario no percibe que se trata de un link. Otro problema es que los links visitados no cambian de color lo que provoca que el usuario no esté seguro sobre si ya visitó el link y pueda volver a visitarlo por repetidas veces [14].

Para minimizar estas dificultades es necesario hacer un testeo de la página con "usuarios reales", como mencionado anteriormente. Cuanto antes se ejecute estos tipos de *testings*, más rápido se identificarán los problemas de usabilidad y más rápido podrán ser eliminados [10]. Para realizar un testeo de usuario es necesario considerar tres componentes: (1) usuarios que representan potenciales clientes; (2) proponer que estos usuarios realicen determinada tarea en la página y (3) controlar u observar como la realizan, prestando atención a los acontecimientos o lugares donde se les presentan dificultades para completar la tarea requerida [11].

Es importante focalizar en lo siguiente:

- **Conclusión de la tarea**
  Verificar si el usuario pudo completar la tarea pedida. En el caso que el usuario no pueda completar la tarea o pueda completarla apenas parcialmente, es una fuerte indicación de que la página tiene problemas de usabilidad.

- **El tiempo que el usuario realiza la tarea**
  En el caso que el usuario finalice la tarea con éxito, hay que analizar cuanto tiempo le llevó concluirla. Si el usuario tarda mucho para realizar una tarea básica, eso puede indicar que es más complicado de lo que debería ser.

- **Errores encontrados**
  Hay que estar atento a todos los pasos equivocados y todos los errores cometidos por el usuario durante la ejecución de la tarea. Eso puede indicar que el camino para realizar la tarea no está claro o que el sistema no ofrece mensajes de errores significativos para el usuario.

Para resolver los principales problemas, existen algunas recomendaciones básicas a tener en cuenta en el momento de desarrollar la página web que, si son consideradas oportunamente pueden evitar varios de esos problemas:

- **Contenido**
  Todo el texto de la página web debe ser escrito de forma objetiva y clara. Los usuarios no leen las páginas web, ellos pasan la vista por arriba escaneando su contenido, tratando de encontrar lo que están buscando. Esto se debe a que generalmente los usuarios tienen prisa y que saben que no hace falta leer todo el contenido de la página para lograr un objetivo. Ellos tienden a buscar por frases o palabras que puedan llevarles a completar lo que tratan de hacer en la página, por eso eliminar de la página todo el texto innecesario hace que su apariencia sea más limpia, más concisa y le confiere mayor objetividad, lo que redunda en que el contenido que realmente importa quede más visible. Páginas de introducción o con instrucciones de cómo utilizar la página, por ejemplo, no son recomendables ya que la mayoría de los usuarios los las leen antes de efectuar cualquier tarea [10][14].

- **Navegabilidad**
  Este punto se refiere a como las informaciones están distribuidas y como son presentadas en la página. Una página con buena navegabilidad hace que el usuario encuentre fácilmente lo que necesita y pueda entender sin dilación la dinámica de la página. Para lograr una buena navegabilidad es necesario definir la topología del sitio. Una de las más usadas es la topología jerárquica, donde el contenido más importante de la página queda destacado y es más visible al usuario. Otra técnica es agrupar los contenidos similares, presentándolos de una misma manera haciendo uso de secciones y sub-secciones para indicar que algo está categorizado adentro de una sección [10]. Todo eso hace que el usuario pueda entender la lógica de la página y sea capaz de encontrar otras informaciones adentro del sitio.

- **Links**
  Este punto puede ser clasificado como unos de los ítems fundamentales para la navegación. Los links son el canal que lleva el usuario a encontrar lo que está buscando en la página de inicio (*Homepage*) de una página web. Nielsen [11] define el link como el elemento de interacción número uno de la web. Si los links no están bien identificados (en general los links son subrayados) o no usan el vocabulario correcto, es muy probable que el usuario no los entienda o no los reconozca y no haga clique en ellos. El texto de los links, por ejemplo, debe ser lo más obvio y claro posible, de esta manera será prontamente identificado por el usuario, quién lo usará sin dudar. Agregar descripción a los links también puede ser deseable, así el usuario puede tener una idea previa de lo que va a encontrar en el caso de hacer clic en él [10][11].
  Otra consideración importante a la hora de diseñar un link, es su cambio de color una vez utilizados, hecho éste que le indica al usuario que ya lo recorrió anteriormente. Cambiando el color el usuario puede saber donde estuvo evitando cliquear varias veces en un mismo link, con la consabida pérdida de tiempo que ello significa. El cambio de color hace que el usuario se oriente en relación a las páginas que ya accedió y evita pérdida de tiempo [10][11].

- **Búsqueda**
  El campo de búsqueda debe estar siempre presente en una página web como un salvavidas en el momento que no puede encontrar lo que desea y no sabe más como buscar. En el caso que el usuario esté en un sitio de comercio electrónico y quiera encontrar un producto que no sabe cómo fue categorizado, probablemente va a recorrer al campo de búsqueda. Ese es un buen motivo para que el mismo este siempre visible en la página, en general se lo ubica como un en el topo de la página con un campo en blanco y un botón al lado que dice: "Buscar" o "*Search*". Cambiar esta convención puede confundir el usuario.
  Otro punto relevante es el mecanismo de búsqueda que debe ser eficiente. En el caso que el usuario tenga que determinar cómo quiere hacer la búsqueda, es importante lograr que las opciones sean significativas y fáciles, sino el usuario accederá a resultados que no corresponden con lo que él estaba buscando [10]. En general los mecanismos de búsquedas fallan frente a errores de ortografía, palabras en plural o otras variantes en la palabra-clave. El algoritmo de búsqueda debe ser eficaz en la detección de la importancia de los términos buscados y capaz de retornar los ítems de mayor relevancia para el usuario. Si la búsqueda anda mal es probable que el usuario quede frustrado con los resultados obtenidos. La búsqueda es un elemento fundamental adentro de la página y está directamente conectada con la experiencia del usuario [11].

Estos son algunos de los puntos que ayudarían al usuario y le garantizarían una mejor experiencia al navegar en una página web. Un texto legible, contenido objetivo y que responda las preguntas del usuario, navegación fácil que le permita encontrar lo que busca, formularios sencillos, simples y cortos, texto coherente y links funcionales. Son características sencillas y básicas pero que son de extrema importancia para la usabilidad de una aplicación web [11].

2.3. Modelado de Aplicaciones Web

En el proceso de construcción de una página web común, se utilizan generalmente 3 modelos: modelo de aplicación, modelo de navegación y modelo de presentación.

Modelo de Aplicación, también llamado de **Modelo de Contenido**: describe la estructura de los datos de la aplicación o sea, define el contenido que el usuario va a ver en la página y como este se comporta. *Refactorings* que se aplican en este modelo tienen el objetivo de cambiar factores internos de la aplicación.

Modelo de Navegación: define los nodos de la página (llamados también de unidad de información), la navegación entre esos nodos a través de links y las operaciones que se habilitan con estos links. A través de los nodos de navegación el usuario puede acceder a la información disponible en la página. *Refactorings* en este modelo proponen cambios en la topología de la navegación del sitio web pero sin alterar la accesibilidad de la información y de las operaciones existentes en los nodos, como por ejemplo, crear un link, crear un nuevo nodo o remover nodos no accesibles [43].

Modelo de Presentación: define la interface del usuario y se especifica la apariencia que va a tener la página, como van a distribuirse los links y cómo la interface es modificada de acuerdo con la acciones del usuario [18]. Los *refactorings* en este modelo sugieren cambios en la apariencia de la página pero sin modificar las operaciones que están disponibles en cada página ni su semántica, como por ejemplo reorganizar los elementos que componen la página o separar o unir páginas [15][18].

En las aplicaciones web, que son páginas que manejan y soportan procesos de negocio, aparte de los modelos mencionados anteriormente, se agrega otro, el **Modelo de Proceso** que define los procesos de negocios presentes en una aplicación web, detallando los pasos que el usuario tiene que recorrer para completar el proceso en cuestión. Los refactorings aplicados en este modelo sugieren cambios en la estructura interna del proceso con el fin de mejorar la experiencia del usuario facilitándole cada vez más la tarea de completar las actividades en la aplicación web, como por ejemplo, la compra de un pasaje aéreo. Es importante mencionar que estos pueden afectar también los modelos de presentación y de navegación, eso porque, cuando se altera alguna característica del proceso, la navegación y la presentación de la aplicación web durante la ejecución del proceso también pueden verse alteradas afectando a sus respectivos modelos.

Los modelos de navegación, presentación y proceso afectan a cómo el usuario percibe en contenido en la página y cómo puede interactuar con este contenido [15]. Todos los modelos pueden ser representados por diagramas UML [7], lenguaje popular de modelado de sistemas. UML ofrece un estándar para describir sistemas que incluye aspectos tanto concretos como diagramas de clase o conceptuales como procesos de negocio.

Así como los *refactorings* tradicionales, los aplicados en aplicaciones web, más específicamente en los modelos mencionados anteriormente, hacen mejoras internas preservando la funcionalidad de la aplicación. Un punto de diferencia con los *refactorings* tradicionales es que, al hacer mejoras en los modelos, generalmente lo que el usuario percibe es impactado y haciendo que, aparte de las calidades internas que fueran mejoradas, sean mejoradas también calidades externas de la aplicación, como la usabilidad.

2.4. Proceso de Negocio en Aplicaciones Web

Hoy en día, las aplicaciones web concentran mucho más funciones que exhibir una gran cantidad de información en páginas de internet. Las páginas web modernas evolucionaron, dejaron de ser estáticas y aumentaron su complejidad en virtud de que ahora son capaces de soportar y ejecutar procesos de negocio [1][2].

Las aplicaciones web difieren de los sitios web por su nivel de complejidad. Los sitios web están focalizados en exhibir información para su navegación, mientras que las aplicaciones web requieren una interacción más profunda en tanto y en cuanto comprometen cada vez más las decisiones del usuario [72], llegando incluso a afectar la forma como interactúa con la página, pasando a ejecutar diferentes acciones, como comprar y vender distintos productos o controlar y realizar transacciones bancarias, entre otras.

Las páginas web pueden ser clasificadas de acuerdo con su nivel de complejidad: sitios estáticos y menos complejos que proveen pocas funcionalidades y ninguna interacción con el usuario son clasificados como "Clase 1". Páginas que poseen interacción con el cliente debido al uso de tecnologías multimedia como HTML Dinámico, que permiten interacción con el usuario son denominadas "Clase 2". Sitios "Clase 3" engloban "Clase 1" y "Clase2" y agregan la funcionalidad de contenido dinámico que se hace posible con el uso de base de datos de servidores [29]. Las páginas más modernas que se construyen hoy en día son clasificadas como "Clase 4". Este tipo de página posee una infraestructura de sistemas distribuidos que utilizan Internet como medio de comunicación y la Web como una interface para acceder a una variedad de servicios e informaciones online [6][9].

Ciertos estudios como el de Brambilla et al. [47] especifican el ciclo de vida de una sitio web centralizado solamente en datos (Clase 1, 2 y 3 según la definición descripta por [29]) y el ciclo de vida de una aplicación web que modelan procesos de negocios (Clase 4). La Figura 1 ilustra, a la izquierda el proceso de desarrollo de un sitio web "convencional" que apenas soporta informaciones y datos y, a la derecha, el proceso de desarrollo de una aplicación web que soporta procesos de negocios. El diagrama de la derecha supone una extensión del modelo de una página web común a una aplicación web más compleja, incorporando modelado de procesos a su diseño.

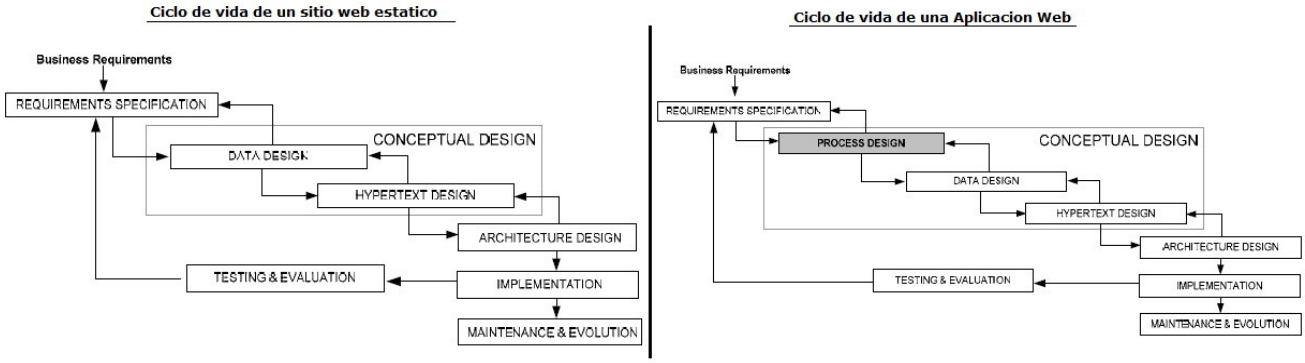


Figura 1: Ciclo de vida de un sitio web estático (izquierda) y de una aplicación web que maneja procesos (derecha) [47]

La fase de definición de los procesos es agregada al inicio del ciclo de vida de la aplicación, antes de la fase de implementación o definición de datos. Es importante que esta etapa sea modelada en una fase por separado y temprana en el proceso de desarrollo de software, de esta manera se puede garantizar que los procesos serán priorizados y modelados adecuadamente en la aplicación web en cuestión.

Por su parte Baresi et al. en [57] detalla la diferencia entre el diseño de los sitios web y las aplicaciones web como muestra la Figura 2. Como los sitios web "tradicionales" focalizan en la organización de la estructura de los datos y la navegación, se representan con un diagrama 2D, donde los únicos factores modelados son la información y la navegación. Cuando se agregan los procesos de negocio, eses deben ser adicionados al diagrama modelando las operaciones del proceso, así como la información y la navegación, que ahora adquieren un carácter dinámico.

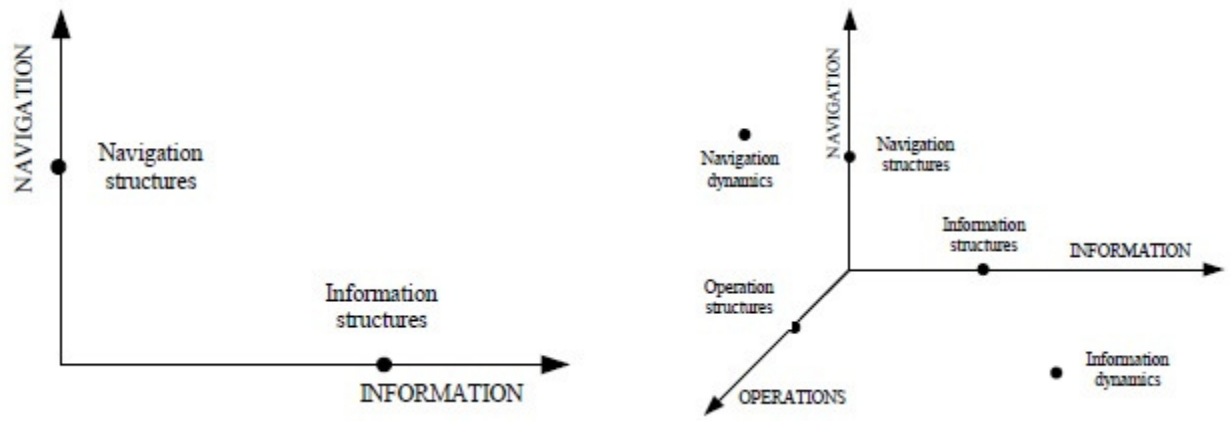


Figura 2: Comparación entre el diseño de los sitios web (izquierda) y las aplicaciones web (derecha)[57]

La incorporación de los procesos de negocio en las aplicaciones web incrementó la realización de un sinnúmero de actividades online. Por definición, los procesos de negocio constituyen un conjunto de acciones o actividades que deben ser completadas correctamente para viabilizar la realización de una tarea o de un objetivo [2][4] e involucran que el usuario siga una secuencia de pasos para poder completar exitosamente su objetivo [1][2]. Por ejemplo, un usuario, para poder realizar una transferencia bancaria debe informar el número de la cuenta de origen, el número de la cuenta de destino, informar el valor que desea transferir y confirmar la transferencia. Después de seguir y completar esos pasos, finaliza exitosamente la tarea.

Este comportamiento de las aplicaciones web tiene impacto en todo el proceso de desarrollo y por eso debe ser analizado y planeado detalladamente antes de la construcción de una página web. La inclusión de reglas de negocio complejas, así como el orden en que son ejecutadas las transacciones durante el proceso y la interacción entre las tareas y la navegación del sitio impactan directamente en la forma como el usuario interactúa con la aplicación. Eso se torna un desafío para los desarrolladores y diseñadores ya que tienen considerar no sólo las estrechas reglas de negocio sino la manera en que estas afectarán al proceso, evitando al mismo tiempo que eso afecte negativamente en la usabilidad de la aplicación web [1][2].

Para introducir los procesos de negocio en el proceso de creación de una aplicación web, se incorporó en la Figura 1 (derecha) una actividad de "Diseño de Proceso", mediante la cual se define, en alto nivel, el esquema de los procesos que van a estar presentes en la aplicación. Esta nueva actividad impacta en las otras etapas del ciclo de vida ya que la navegación, por ejemplo, se ve afectada por los requerimientos del proceso.

El hecho de que los procesos de negocios afectan otras fases del proceso de desarrollo es suficiente para determinar que ellos sean modelados separadamente; pero a pesar de todo el impacto que representan en la aplicación web, es común que su fase de diseño y análisis no se priorice y que sea considerada como una parte del proceso de desarrollo y no como una etapa individual y anterior a este.

El hecho de no considerar adecuadamente esta etapa o de que no esté bien planeada y definida, ocasiona que la aplicación web tenga comportamientos inadecuados que resultan en mala o pobre usabilidad [6]. Otro problema que puede presentarse, según lo apuntan Rossi y Schmid, es que muchas veces los procesos son tratadas como una secuencia de pasos navegacionales, lo que redunda en problemas de usabilidad y de diseños, aparte de generar inconsistencias en las reglas de negocio [5][34]; esto sin mencionar que muchas metodologías existentes no soportan el modelado de proceso. Esto se debe a que muchas de estas metodologías fueron construidas para modelar sistemas de información (IS) que manejan básicamente datos [23][50] y no aplicaciones web más complejas que soportan los procesos de negocio.

Por causa de esa complejidad, la incorporación de procesos de negocios en las aplicaciones web demanda del uso de metodologías específicas, que fueron extendidas con el objetivo de diseñar y guiar el desarrollo para integrar los procesos en las aplicaciones web. Metodologías como OOHDM *(Object-Oriented Hypermedia Design Method*) [5][34][53] y WebML [36][38] por ejemplo, tuvieron sus módulos (contenido, navegación y presentación) extendidos para soportar los procesos de negocio. Otras metodologías como OO-H (*Object-Oriented Hypermedia Method*) [3][49] y UWE (*UML Based Web Engineering*) [3][17][21][22][23][26] crearon módulos adicionales con el fin de modelar los procesos por separado. En el desarrollo y planeamiento de esta tesis se utilizará la metodología UWE para modelar los diagramas en los que se basará el estudio.

Muchas metodologías utilizan el lenguaje UML como notación para representar los procesos de negocio, como por ejemplo OO-H, OOHDM, UWAT+ [35] y UWE. Sin embargo otras notaciones fueron propuestas para el modelado de procesos. Wisdom [52] es una notación que propone el uso de clases estereotipadas pero no muy intuitivas. Ya Markopoulos en [51] sugiere una extensión de los casos de uso y otra basada en gráficos de estado y diagramas de actividad. Esta última opción es hoy en día usada por muchas metodologías (como UWE) para modelar los procesos de negocio por tratarse de un mecanismo práctico para modelar los flujos del proceso y por su amplia utilización.

Las diferencias entre páginas navegacionales y aplicaciones web se pueden analizar también desde lo semántico. Una página navegacional posee una semántica más simple como:

- Tiene una estructura más linear donde el usuario es el que decide cual será el próximo link a ser visitado.
- El estado de los links visitados no interfiere en el estado del nodo actual. Eso indica que el diseñador no puede predecir cuál será la elección del usuario y que nodo va a visitar a seguir.

En tanto una página con proceso de negocio, con una estructura más compleja, plantea:

- El usuario es guiado por una secuencia de pasos ya definidos que debe ejecutar para finalizar la tarea con éxito.
- El proceso mantiene su estado internamente, describiendo las acciones que se ejecutaron (historial) y las que todavía faltan ser ejecutadas.
- El proceso puede modificar su estado dependiendo de la acción de usuario. Presionar los botones del *browser* no influyen en el estado del proceso.
- Procesos poseen ciclos de vida y estados temporarios.

Todo eso influye en las decisiones del desarrollador quien para desarrollar la página deberá considerar la semántica de las actividades y los estados que soportarán los procesos de negocios en las aplicaciones web [5][8].

Es posible simular un proceso de negocio por medio de una página navegacional, si se trata de un proceso bastante simple y lineal, que incorporan botones para guiar el usuario a través de los pasos a seguir, no obstante esa representación puede ocasionar problemas. El primero de ellos ligado a la imposibilidad de definir o predecir los pasos del usuario. Caso este quiera navegar por alguna página afuera del proceso, es posible que resulte en su desorientación, incidiendo en que el mismo no sepa cómo volver para proseguir con el proceso interrumpido. Otro problema refiere al estado del proceso caso el usuario decida salir sin completarlo. En este caso cabe preguntarse ¿El proceso debe abortarse o debe seguir cuando el usuario vuelva? Por lo tanto, el estado de inconsistencia del proceso una vez que el usuario visita otras páginas, también es un inconveniente en este tipo de propuesta [5][8].

Un proceso de negocio contiene varias sub-actividades que, internamente que genera un flujo de control. Solamente cuando esas actividades fueron completadas, siguiendo la secuencia de pasos prevista, el proceso ha concluido. Un proceso que posee muchas sub-actividades no puede ser simulado por una secuencia de pasos navegacionales, ya que adentro de un proceso de negocio, el orden en que las actividades son ejecutadas depende de la interacción de usuario y de las acciones y decisiones del mismo en el transcurso del proceso.

Para posibilitar la integración entre la navegación de la página y la ejecución de los procesos de negocio se debe coordinar la transición entre uno y otro, de manera de permitir:

- **Empezar y completar con un proceso de negocio desde la navegación**: El usuario mientras navega por la página puede empezar un proceso de negocio haciendo clic en un link o un botón que dispara e inicia el proceso. Eso hace que cambie el nodo de navegación por el nodo de proceso, interrumpiendo la navegación en sí y iniciando el proceso. Para completar el proceso, el usuario puede finalizarlo o cancelarlo (si posible). En el caso de la última opción, el proceso es finalizado retrocediendo todos los cambios de estado que ocurrieron durante su ejecución. En ambos casos, el proceso finaliza volviendo al nodo de navegación [5][8].

- **Suspender un proceso para navegar por páginas afuera del proceso y reanudándolo después**: En el momento de diseñar un proceso de negocio, el desarrollador debe decidir si permite que el usuario pueda suspender el proceso o no. Esa opción significa que el usuario puede salir del proceso antes de finalizarlo, navegar por páginas afuera del proceso y después volver para continuar con el mismo desde el punto en que lo dejó. Para permitir tal suspensión del proceso, el estado del mismo debe ser guardado en el momento en que se interrumpió, de manera que, cuando el usuario regrese el proceso identifique el estado en que se encontraba y pueda de esta manera proseguir [5][8].

No obstante las diferencias de semántica y comportamiento entre los dos tipos de aplicaciones, gran parte de las herramientas no han evolucionado a fin de modelarlas y diseñarlas de manera diferente, y se continúa tratando los procesos de negocio como una vertiente de las antiguas páginas navegacionales ocasionando problemas de usabilidad [8].

Para tratar los procesos de negocio como una etapa separada del proceso de desarrollo y como páginas que exigen más complejidad que las simplemente navegacionales, es necesario considerarlos antes de la fase de desarrollo como ilustrado en la Figura 1, modelando su estructura para ver el impacto que tiene en las aplicaciones web. Para modelar e integrar los procesos de negocio en las aplicaciones web de forma adecuada, hay que considerar que el mismo tiene impacto específicamente en tres modelos diferentes de las aplicaciones web:

- **Modelo de Procesos**: En ese modelo refleja el detalle del proceso de negocio, sea en su forma estática (Modelo de Estructura de Proceso) o en su forma dinámica (Modelo de Flujo de Proceso). Especifica cómo es el flujo del proceso de negocio, cuales son los roles del usuario y del sistema en el desarrollo del proceso, qué dependencias existen y el conjunto de reglas de negocio que guían este proceso.

- **Modelo de Navegación**: En este modelo se muestra en que etapa de la navegación se encuentra la llamada para el proceso de negocio, representando cual el nodo de navegación que dispara el proceso y qué nodo de navegación es llamado cuando el proceso se finaliza. También se describe cómo la navegación puede interferir en el andamiento de un proceso de negocio.

- **Modelo de Presentación**: Representa los elementos de la interface del usuario que disparan un proceso de negocio en la aplicación. Especifica también cuáles son los *widgets* usados y presentados al usuario cuando este está ejecutando un proceso de negocio.

El impacto que la integración de los procesos de negocio tienen en cada modelo y cómo la aplicación de los *refactorings* en esos modelos puede mejorar la experiencia del usuario a la hora de ejecutar dichos procesos es el foco principal de esta tesis y va a ser tratado más en detalle en los capítulos siguientes.

2.5. Metodología UWE

UML-based Web Engineering (UWE*)* es una metodología basada en el lenguaje UML (*Unified Modeling Language*) utilizada para modelar aplicaciones web, permitiendo representar todas las etapas del proceso de desarrollo de software. Se basa en el enfoque centrado en modelos de desarrollo de aplicaciones web con foco en el diseño sistemático, personalización y generación semiautomática. Este enfoque es utilizado por las principales metodologías de diseño y desarrollo de aplicaciones web como OOHDM [17][18][23][53], UWA [35][37], WSDM [38], WebML [36][38] y OOWS [40][48]. La mayoría de estas metodologías utilizan modelos separados para representar las vistas de la aplicación, como presentación, navegación, procesos, entre otros.

UWE utiliza diagramas UML (con la adición de algunos estereotipos) para modelar diversos aspectos de la aplicación web, incluido el modelado de procesos de negocio. UML es el lenguaje utilizado para el modelado de procesos de negocio (para dominios distintos de la Web) según Markopoulos [51] y Nunes y Cunha [52] por tratarse de un lenguaje ampliamente difundido, con lo cual el modelado se torna más fácil ya que, los diagramas y la metodología no son sólo bien soportados por diversas herramientas CASE, sino son ampliamente conocidos.

Asimismo, el lenguaje UML posee modelos y diagramas suficientes para representar la mayoría de los requerimientos y especificaciones de todos los tipos de software sin necesitar de anexos o extensiones. No obstante, existen algunos aspectos específicos del diseño de una aplicación web que necesitan ser representados con más detalles y que requieren vistas especiales. En tal sentido UWE, extensión más simple del estándar UML, provee esas vistas a través de la inclusión de estereotipos, utilizando exclusivamente las técnicas, la notación y los mecanismos de extensión provistos por este lenguaje.

Los estereotipos son nuevos elementos utilizados en los modelos, que extienden la semántica permitiendo crear fácilmente modelos más expresivos y precisos. Se tratan de extensiones para personalizar los modelos ofrecidos por UML con el fin de adaptarlos a determinado dominio de la aplicación. Su uso tiene ventajas y riesgos. La ventaja es que, con los estereotipos los modelos quedan mejor representados y más descriptivos. El riesgo aparece con su uso excesivo, pues la saturación de estereotipos puede producir el efecto contrario, en lugar de facilitar el entendimiento de los modelos, los puede tornar difíciles de comprender y manejar [23].

El lenguaje UWE posee definiciones que representan características específicas y necesarias para el diseño de modelos en el dominio Web y el hecho de ser una ramificación del lenguaje UML le provee de la flexibilidad necesaria para la definición en este dominio. Como el lenguaje UML es un lenguaje de amplio uso en la mayoría de las herramientas CASE y en la ingeniería de software en general, la aplicación de UWE es de fácil entendimiento y de simple utilización. Aunque esté basada en UML, la metodología UWE también utiliza otros estándares OMG (*Object Management Group*) por ejemplo XMI (*XML Metadata Interchange*) como modelo de intercambio de formato y XML [17].

Las principales características de la metodología UWE refieren a que:

- Provee un lenguaje completamente basado en UML.
- Se basa en patrones.
- Se basa en los sistemas personalizados y en la sistematización.
- Es una metodología que añade modelos utilizados para representar aplicaciones web.
- Proporciona una herramienta para generar aplicaciones web semiautomáticamente.
- Proporciona herramientas CASE para el diseño de los modelos, como por ejemplo ArgoUWE y MagicUWE, herramientas que soportan la notación UWE.

MagicDraw y MagicUWE

Las principales actividades de modelado son el análisis de requerimientos de la aplicación, modelado de su contenido, modelado de la navegación y presentación de las páginas, y el modelado de los procesos de negocio. Como UWE es una extensión del lenguaje UML, el uso de las herramientas CASE que crean los diagramas UML también puede ser extendidos para crear modelos UWE. Con el objetivo de ofrecer las herramientas necesarias para el modelado en UWE, se creó un *plugin* específico para la herramienta CASE MagicDraw, llamado MagicUWE que se basa en el metamodelo UML2.0.

MagicDraw es una herramienta CASE desarrollada por la empresa *No Magic Inc*., que soporta UML 2.3, ingeniería para múltiples lenguajes de programación, como Java y C++, así como modelado de datos. Como fue dicho, MagicUWE, es el *plugin* para esta herramienta y ofrece los artefactos necesarios para la implementación de los modelos de UWE; soporta el diseño sistemático de aplicaciones web utilizando la notación descripta por UWE así como su proceso de desarrollo proveyendo barra de herramientas especificas con los estereotipos y diagramas necesarios para crear las diferentes vistas de la aplicación web, a saber: contenido, navegación, presentación y proceso [22]. Con este *plugin* también es posible realizar transformaciones automáticas desde un modelo a otro, como crear un modelo básico de navegación basado en el modelo de contenido creado.

2.5.1.Modelos de la Metodología UWE

UWE utiliza notación UML así como sus diagramas para hacer el análisis y el diseño de aplicaciones web. Para los componentes que son usados específicamente para el dominio Web (links y nodos por ejemplo) UWE posee estereotipos y limitaciones en el modelado de dichos elementos. La notación utilizada en UWE cubre los aspectos de contenido, navegación, presentación y proceso del dominio web como se ejemplifica en la Figura 3.

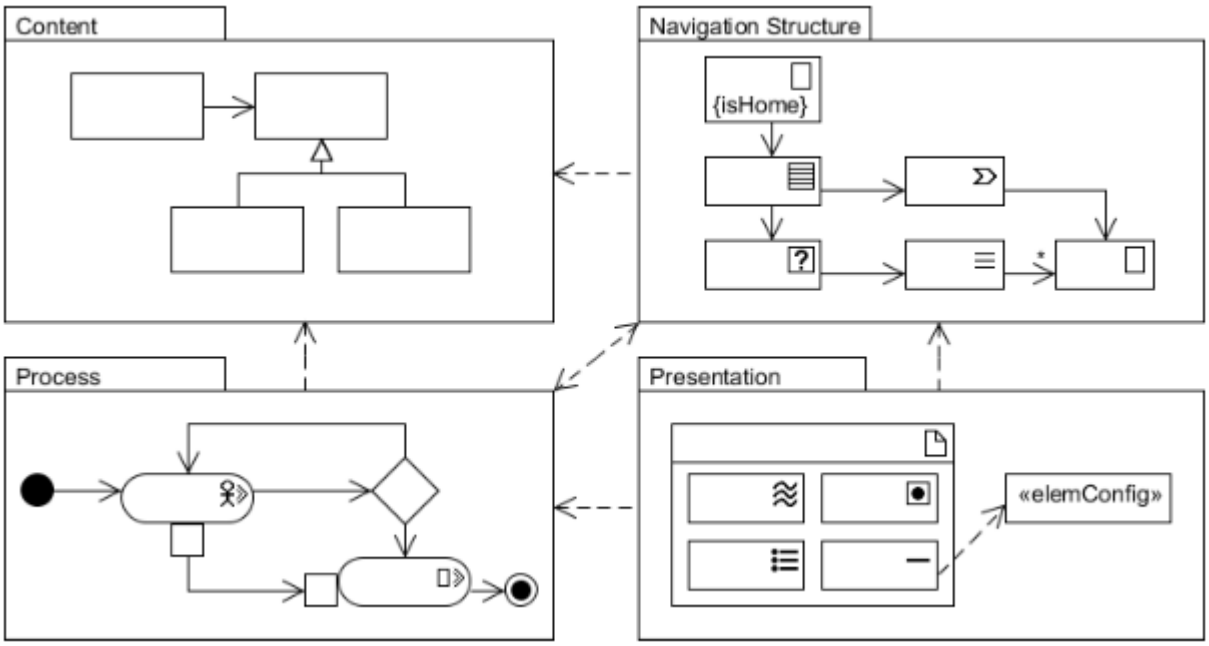


Figura 3: Vista general de los principales modelos UWE.

Cada modelo utiliza determinado tipo de diagrama UML así como estereotipos específicos para representar cada vista de la aplicación. En los próximos apartados se profundizarán las características principales de cada modelo, aunque se podrá acceder a los detalles de cada metamodelo en el Anexo A.

2.5.1.1. Modelo de Contenido

El modelo de contenido muestra la representación de la información del dominio, relevante para ser desarrollada en la aplicación web; en síntesis, describe como están distribuidos los datos, como se relacionan y como están estructurados en la aplicación, ignorando los aspectos de los modelos de navegación, presentación y proceso. El objetivo de este modelo es proporcionar una especificación visual de la información relevante de aplicación web, describiendo la estructura de los datos en la página y las operaciones entre esos contenidos.

El modelo precisado en UWE para definir aplicaciones web, no se diferencia de un modelo que define otro dominio; por tal motivo el modelo de contenido se especifica a través del diagrama de clase UML conteniendo clases, atributos, asociaciones y otros elementos UML, todo eso sin necesitar de ninguna extensión o estereotipo en su construcción [21].

El modelo de contenido no modela procesos y se utiliza para describir un conjunto de elementos estáticos de la aplicación, por eso que se lo considera un modelo estático. En él se representan tanto los objetos que están involucrados en las actividades que los usuarios van a realizar en la página web, como los que sirven de entrada o son resultado de una actividad.

Por tratarse del único modelo que no modela procesos, dentro del universo de los modelos UWE, este modelo no se incluirá en el desarrollo de esta tesis.

La Figura 4 a seguir muestra como ejemplo, un Modelo de Contenido representado a través de un diagrama de clase para ejemplificar los datos de un agenda telefónica.

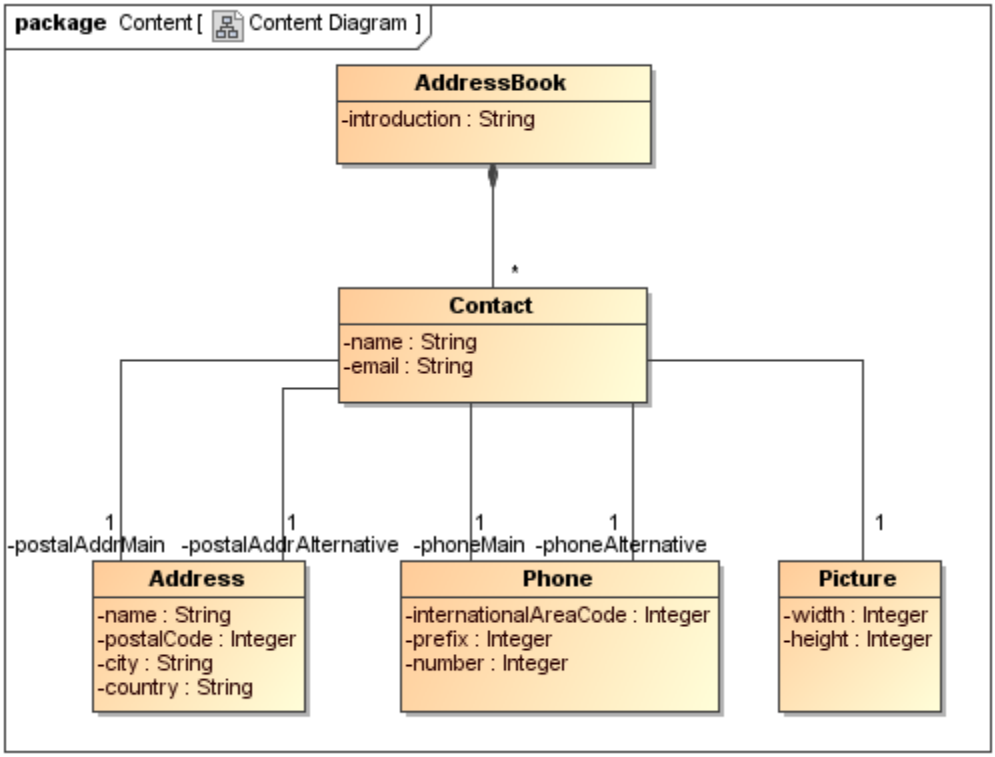


Figura 4: Modelo de contenido de un sitio de agenda de contactos.

2.5.1.2. Modelo de Navegación

Este modelo está basado en el modelado de los requerimientos y contenido. Las clases del modelo de contenido que son consideradas relevantes para la navegación se incluyen en el modelo de navegación, así como sus asociaciones, representando los *navigationClass* y *navigationLinks*, respectivamente [20][23].

El modelo de navegación provee una visión de las páginas que componen una aplicación web y como estas se conectan internamente de forma que es posible entender la estructura de la misma. Tiene como objetivo representar los nodos y links de la estructura de hipertexto y diseñar el camino de navegación de la página, mostrando cuales son los objetos a los que se puede acceder a través de la navegación y cómo el usuario puede acceder a ellos [26]. En este modelo es posible ver los nodos (<*navigationClasses*>) – unidades de navegación – y los links (<*navigationLinks*>) existentes que conectan esos nodos. Los nodos contienen los datos perceptibles al usuario y las operaciones sobre esos datos. Los links hacen las conexiones entre los nodos y reflejan las asociaciones del modelo de aplicación [18]. Al definir la estructura de navegación de la aplicación web se puede evitar que existan desconexiones entre los links o identificar la facilidad del acceso a los nodos. En las aplicaciones web que manejan procesos de negocio, los procesos están integrados en el modelo de navegación. Para eso se utilizan dos estereotipos adicionales: <*processClass*> y <*processLink*>, el primero es usado para representar el proceso de negocio que está embebido en el modelo de navegación y el segundo, que supone la asociación entre un <*navigationClass*> y un <*processClass*>, indica cual es el nodo

de navegación que dispara el proceso y cómo la navegación sigue una vez que el proceso ha finalizado (más detalles pueden observarse en el Anexo A).

El modelo de navegación es representado por diagrama de clases UML, pero como se mencionó y al contrario del modelo de contenido, necesita elementos específicos para modelar aplicaciones web: los estereotipos.

Las bases del modelo de navegación son justamente las metaclases *NavigationNode* y Link y sus asociaciones, como se muestra en la Figura 5. Un conjunto de subclases del *NavigationNode* y Links determina metaclases especificas para construir el modelo de navegación: *NavigationClass* y *ProcessClass* con sus *NavigationLinks* y *ProcessLinks*, así como el *Menu* y los accesos primitivos como *Index*, *GuideTour* y *Query* [21].

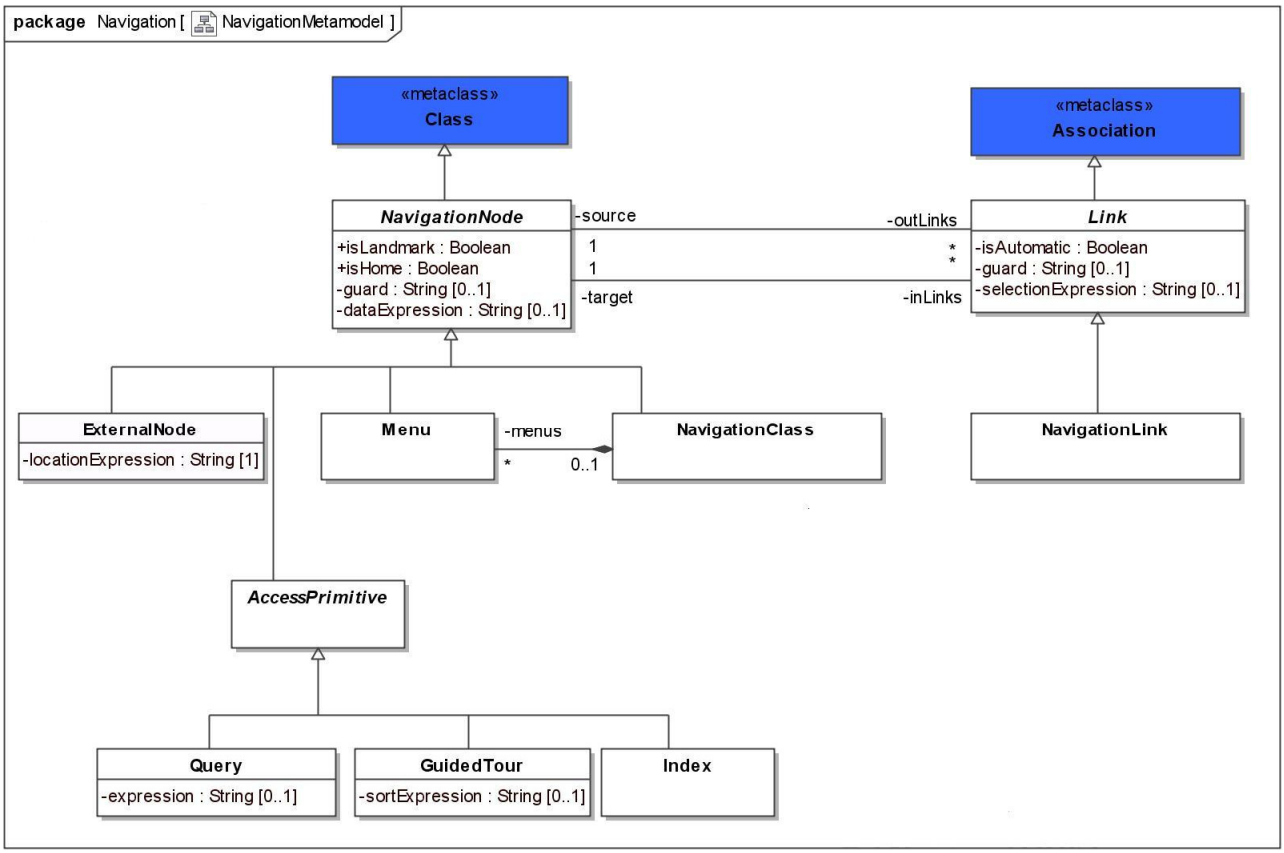


Figura 5: Vista parcial del metamodelo del diagrama de navegación

La Figura 6 muestra un ejemplo que utiliza los elementos e estereotipos mencionados anteriormente para representar el modelo de navegación de una agenda de contactos.

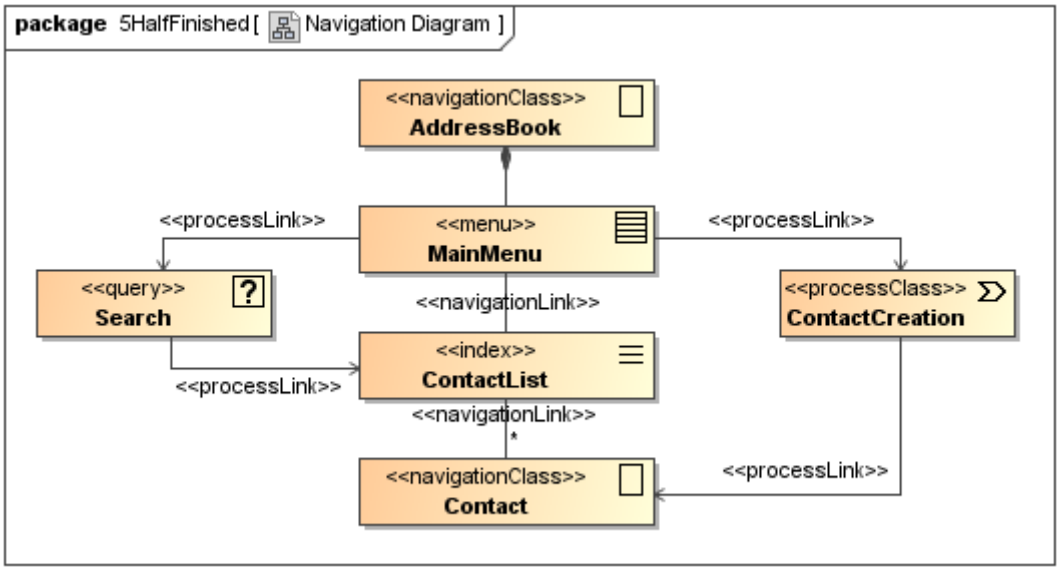


Figura 6 : Modelo de navegación de un sitio de agenda de contactos

Los estereotipos <*query*>, <*menu*>, <*index*> y <*processClass*> son usados en el diagrama anterior para representar elementos específicos de la navegación de la página representada. Más detalles respecto a los estereotipos mencionados pueden ser encontrados en el Anexo A.

2.5.1.3. Modelo de Presentación

El modelo de presentación provee una vista abstracta y describe la estructura básica de la interface del usuario (UI) de una aplicación web [21]. Es un modelo basado en el modelo de navegación y los elementos representados son usados para presentar los nodos de navegación – cada atributo del <*navigationClass*> está representado por un elemento de la UI. Cada atributo del <*navigationClass*> es representado en el modelo de presentación por el elemento de la UI correspondiente, como por ejemplo: un elemento de "*text*" es usado para representar el atributo "titulo" del <*navigationClass*> y un elemento "*image*" es usado para representar el atributo "foto". Generalmente el contenido de distintos <*navigationNodes*> es presentado en una página web, las <p*resentationPage*> en UWE. Para facilitar la identificación de que <*navigationNode*> es representado por el <*presentationClass*>, esas, por defecto, llevan el mismo nombre del <*navigationNode*> [20].

Este modelo describe la estructura básica de la UI definiendo cuales elementos de la interface (ancla, texto, imagen, formularios, etc.) van ser usados para representar los <*navigationNodes*>. Sin embargo, el modelo de presentación no considera los aspectos concretos de la interface del usuario, tales como colores, fuentes, y donde estarán localizados los elementos en la UI, por eso se trata de una presentación abstracta de la interface, representando algunos aspectos concretos [20][26]. Además, los elementos de la UI no representan ninguna tecnología pero si describen qué funcionalidad es necesaria en un punto particular de la interface. Eso puede significar que un texto o una imagen deban ser mostrados o que el usuario sea capaz de disparar una transición en el modelo de navegación [21].

Para representar este modelo se utiliza el diagrama de clases con notación de composición UML para clases como agrupación grafica de los símbolos [23]. La ventaja del modelo de presentación es que, independiente de la técnica de implementación que va a ser utilizada para construir la UI, los *stakeholders* pueden analizar y discutir la presentación de la página antes que esta sea implementada [20].

Los elementos básicos del modelo de presentación son: el <*presentationClass*> y los <*presentationGroup*>. Las <*presentationClass*> *se* basan directamente en los nodos del modelo de navegación (<*navigationClass*>, <*menus*>, <*processClass*>, etc.) y *r*epresentan páginas web o parte de páginas web, mostrando una composición de elementos de la interface del usuario, como <*text*> (texto), <*anchor*> (ancla), <*form*> (formularos), <*image*> (imagen), <*button*> (botón), etc., que están detallados en el Anexo A [20][26]. Los <*presentationGroup*> son un conjunto de <*presentationClass*> que se muestran de manera alternada, dependiendo de la navegación.

En el modelo de presentación no existe un elemento específico que modele los procesos, como lo es el <*processClass*> en el modelo de navegación. Pueden ser representados por links (<*anchors*>) que disparan el inicio de un proceso, o por elementos de la UI (*widgets*) que facilitan la ejecución de los mismos mejorando la experiencia del usuario cuando están navegando por un *business process*.

La Figura 7 muestra el metamodelo del diagrama de presentación y en el Anexo A se presentan los principales elementos del mismo.

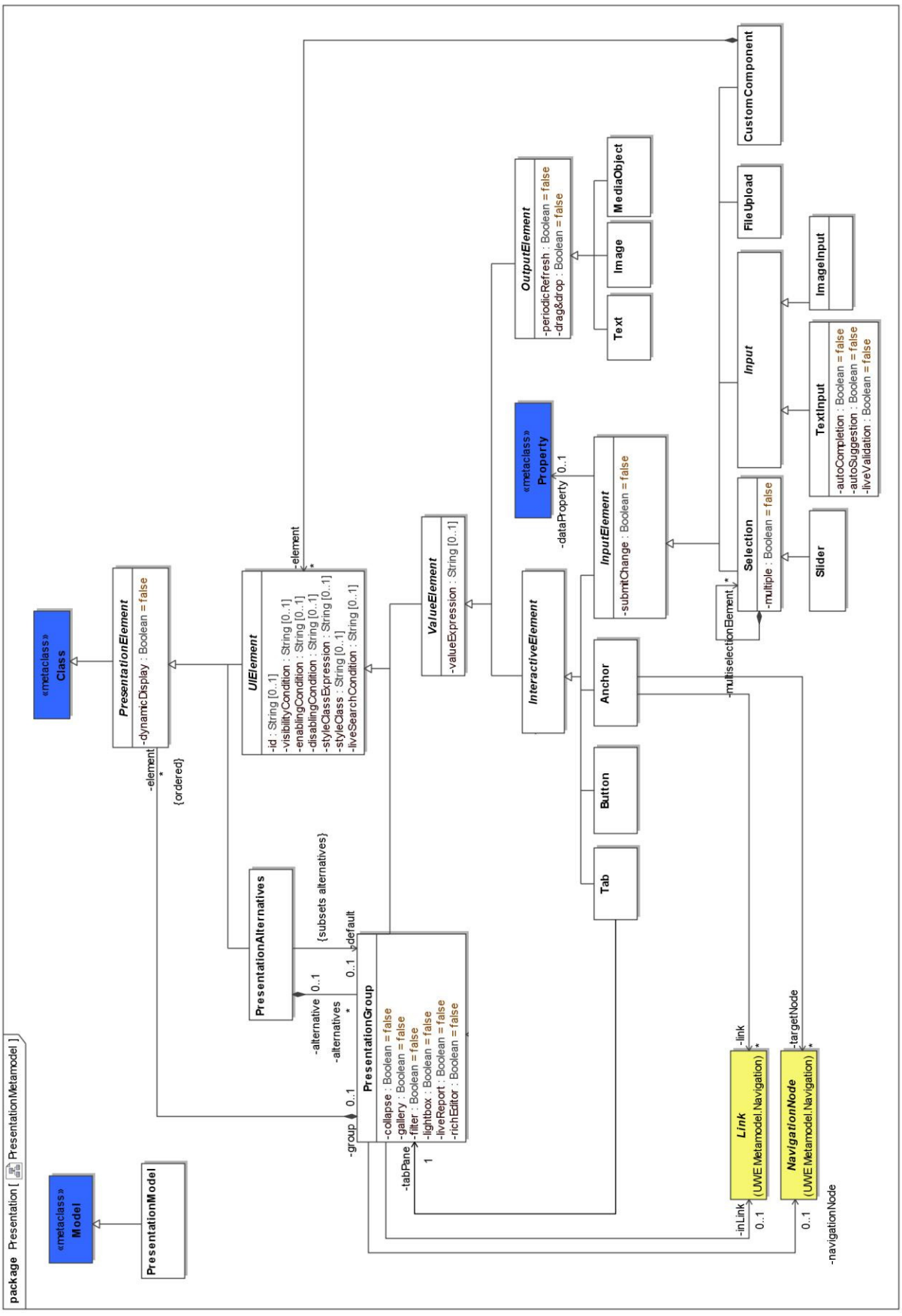


Figura 7: Metamodelo del diagrama de presentación.

Existen dos vistas distintas del modelo de presentación: Vista Estructural que muestra la estructura del espacio de presentación y Vista de *User Interface* (UI) que describe los detalles de los elementos de UI en la página.

- **Vista Estructural**:
  Su objetivo es modelar cómo está dividido el espacio de presentación, mostrando cuales son los elementos exhibidos en el mismo espacio (pero no al mismo tiempo) y cómo pueden agruparse los elementos de presentación.

  La Figura 8 muestra el modelo de presentación de una página de c*heckout* representado según la vista estructural.

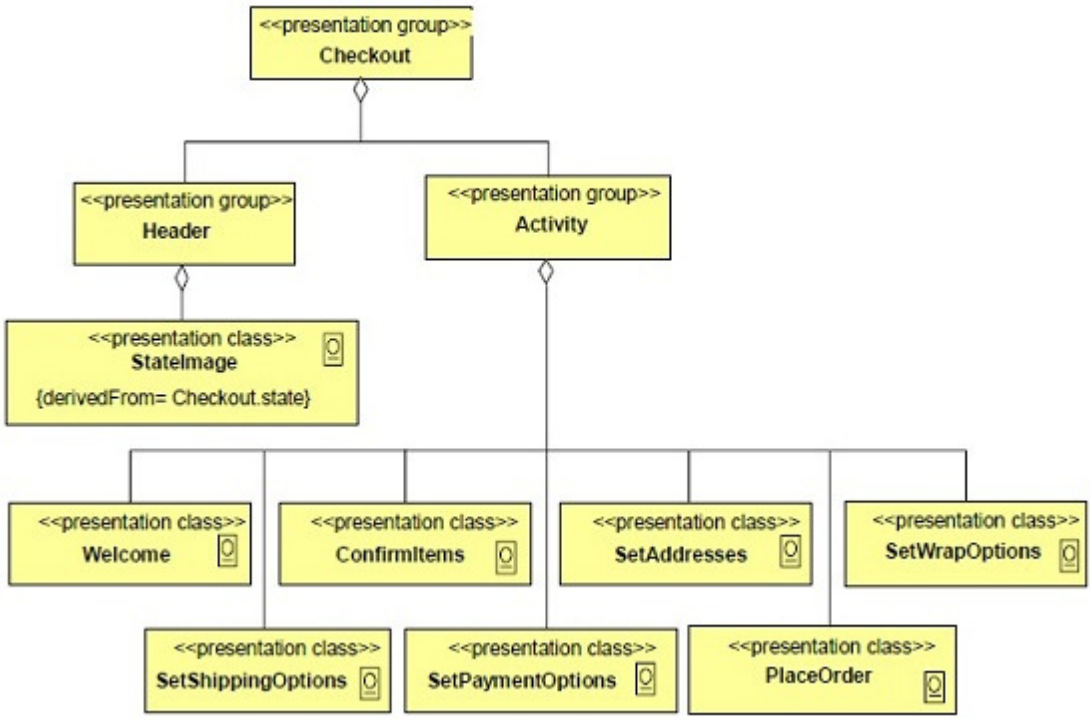


Figura 8: Vista estructural del una página de C*heckout*

- **Vista de *User Interface*:**
  Tiene el objetivo de mostrar los elementos de la UI adentro de la <*presentationClass*> permitiendo una vista más intuitiva de página. Cada elemento de la UI tiene un estereotipo asociado y están conectados con los atributos y operaciones representados en el diagrama de navegación y proceso. El tipo de elemento utilizado para representar estos atributos depende de la intención de uso, como por ejemplo un botón (<*button*>) puede usarse para cambiar la navegación de la página y/o para disparar un proceso de negocio [49].

  La Figura 9 muestra un ejemplo de agenda de contactos representado a través del diagrama de presentación con la vista de *User Interface*.

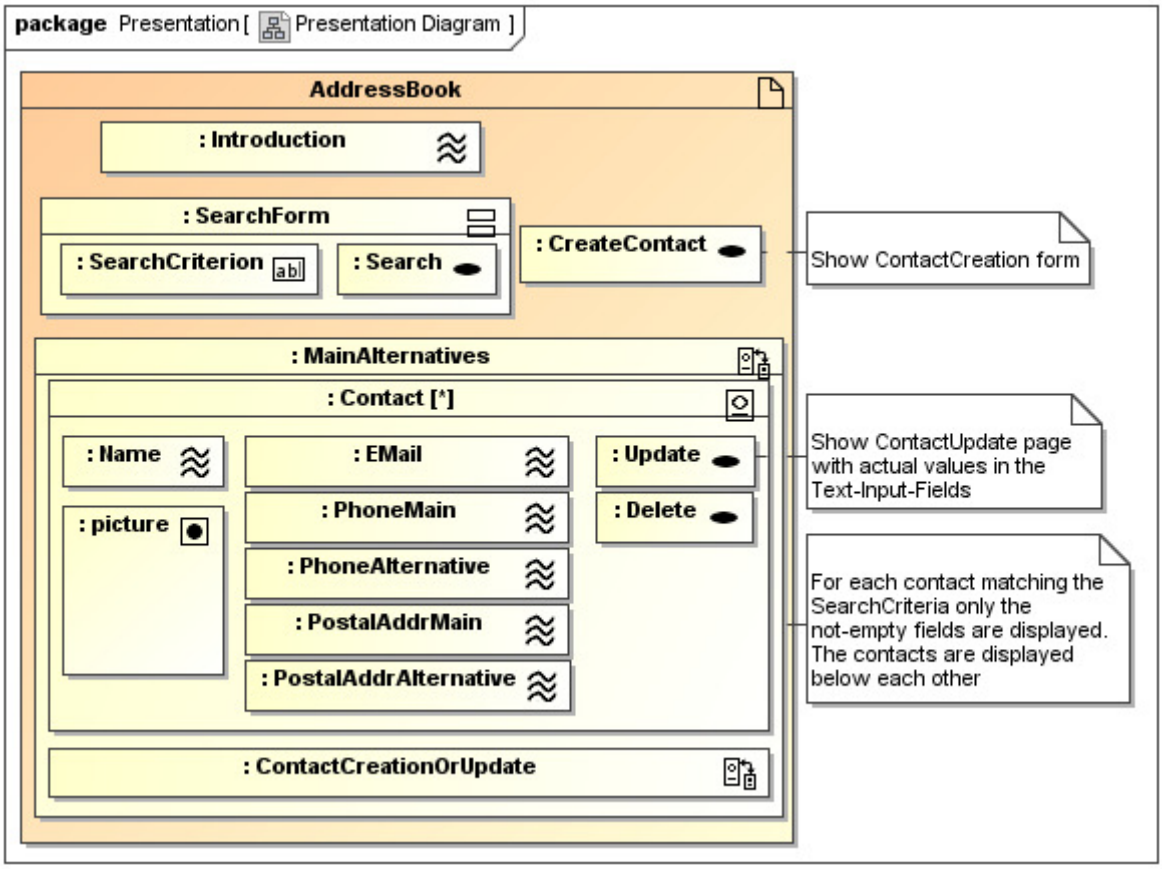


Figura 9: Modelo de presentación de un sitio de agenda de contactos.

2.5.1.4. Modelo de Proceso

Este modelo provee los elementos necesarios para representar procesos de negocio en un modelo UWE [21]. Al contrario del modelo de navegación que representa la estructura estática de la aplicación web, el modelo de proceso representa la parte dinámica de la misma; en sí mismo supone el itinerario que hará el usuario guiado por un serie de pasos, con la finalidad de completar una tarea, representando las acciones de las *processClass* y especificando la funcionalidad de una transacción y complejos flujos de actividades adentro de la página web [26].

El paquete de procesos posee 3 finalidades básicas:

- **Integrar los procesos de negocio en el modelo de navegación:** Las metaclases *processClass* y *processLink* extienden *Node* y *Link* definiendo como un proceso puede ser alcanzado a través de la navegación y como la navegación continua después del proceso. Cada *processClass* especificadas en el modelo de navegación es expandida en el modelo de proceso, que consiste en un modelo de flujo de proceso o en un modelo de estructura de proceso, situación que se detallan en el Anexo A [20].

- **Definir la interface del usuario para que soporte procesos**: Cada *processClass* puede tener la interface definida usando el modelo de presentación UWE, puesto que la mayoría de las veces los

procesos requieren de una interface con el usuario para que este pueda proveer los datos necesarios para llevar a cabo el proceso.

- **Definir el comportamiento del proceso:** El comportamiento del proceso se define a través de un diagrama de actividad UML y representa cuando el sistema tiene la acción en determinado punto del proceso (*systemAction*) o cuando se le solicita al usuario ingresar datos e informaciones (*userAction*)[21].

En la Figura 10 está detallado el metamodelo del diagrama de proceso. Las clases descriptas en este metamodelo están detalladas en el Anexo A.

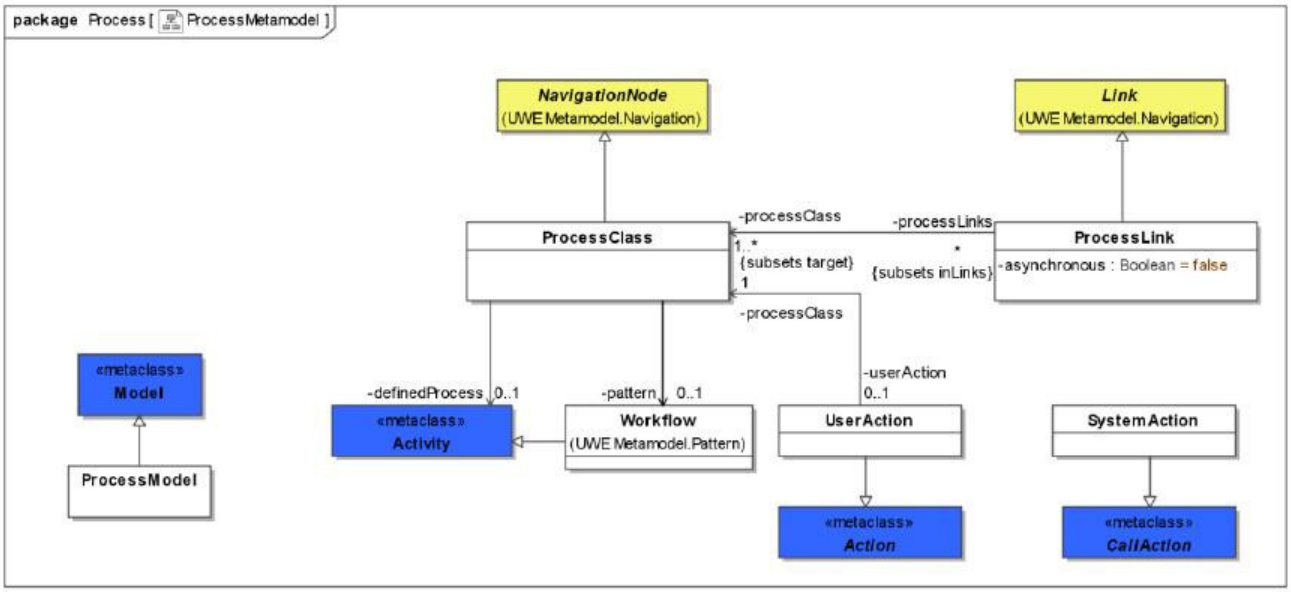


Figura 10: Metamodelo del diagrama de proceso.

La metodología UWE, al contrario de otras metodologías, se caracteriza por introducir un nuevo diagrama específico y exclusivo para el modelado de los procesos de negocio. En ella, los procesos son definidos por separado, no como una extensión de los otros modelos existentes y representados en el modelo de navegación a través de los <*processClass*> pero, los detalles y el flujo interno de los mismos son desmenuzados en el modelo de proceso disponible en esta metodología.

Como fue mencionado anteriormente, el modelo de proceso puede ser representado básicamente en dos formas:

- **Modelo de Estructura de Proceso**

  Este tipo de modelo deriva del modelo de contenido y describe la relación entre las diferentes clases de proceso usadas para soportar los procesos de negocio [20][49]. Para representar este tipo de modelo se utiliza el diagrama de clase UML con el adicionado del estereotipo <*processClass*>; el desarrollador es el responsable de definir cuáles son las clases relevantes para el proceso. El objetivo de esta vista es capturar las informaciones relacionadas con el proceso de negocio, para eso, cada

atributo de las clases expresan los datos necesarios en función del proceso, incluyendo los datos que deben ser provistos por los usuarios (los *user inputs*).

Este tipo de modelo muestra cómo interactúan las clases, pero no detalla las actividades del proceso, las mismas serán definidas en el modelo de flujo de proceso. En realidad este modelo representa la relación entre las clases, los cambios en jerarquía y semántica de los procesos, como se puede ver en el ejemplo ilustrado en la Figura 11.

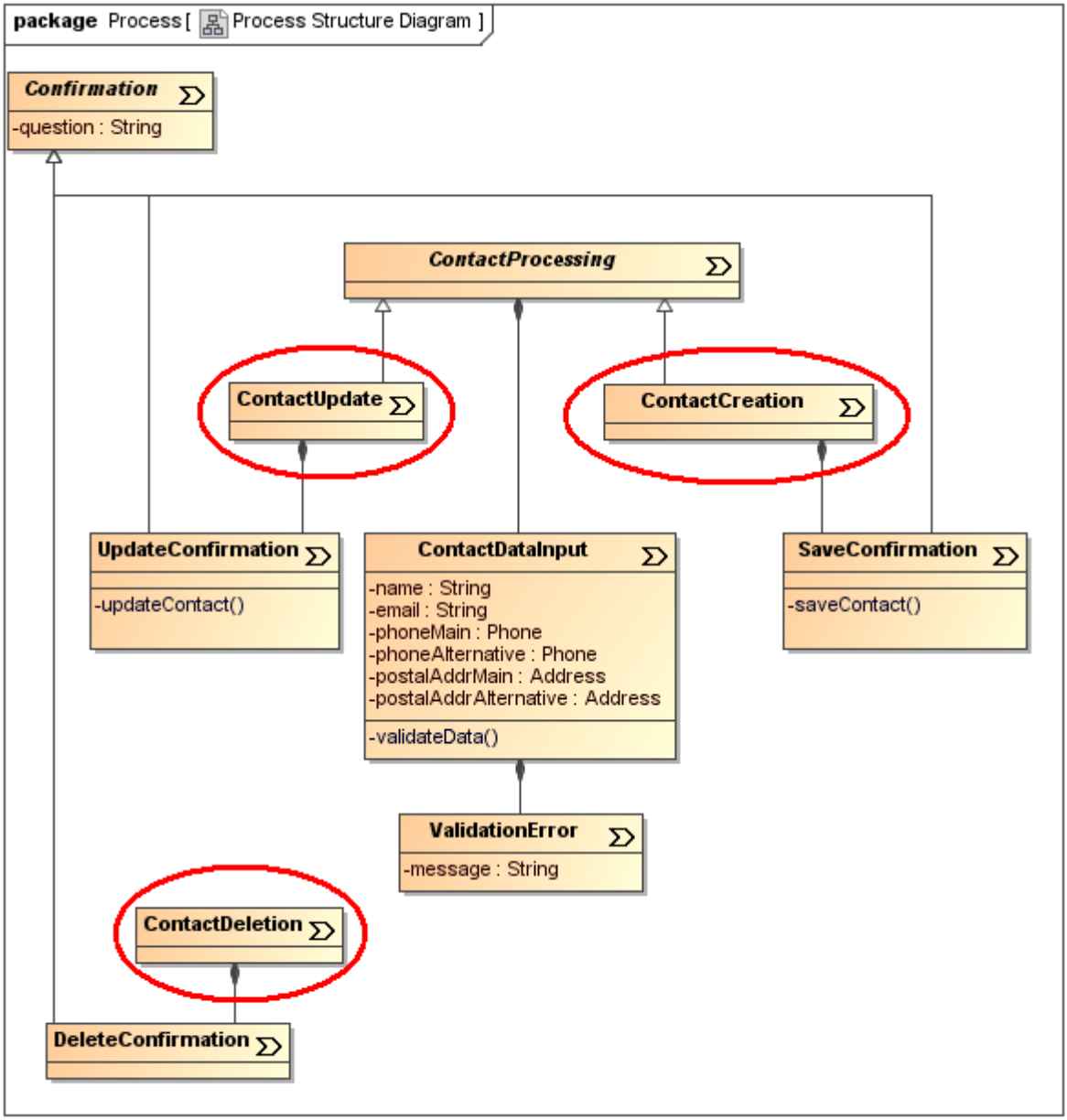


Figura 11: Modelo de estructura de proceso de un sitio de agenda de contactos

- **Modelo de Flujo de Proceso**

Este tipo de modelo define el comportamiento y los detalles de un proceso de negocio. El *process flow*, como también se lo llama, describe minuciosamente los pasos dentro de un proceso, en el caso que el usuario navegue por éste. Está representado por diagramas de actividad UML, sin la necesidad de estereotipos y describe las acciones del usuario y del sistema, vistas anteriormente, durante el proceso.

De todas las técnicas de modelado de UML, el diagrama de flujo o diagrama de actividad es el que mejor describe los procesos de negocio. Cada actividad representa un paso del proceso y las transiciones capturan el flujo del mismo, incluyendo las junciones que expresan las actividades que pueden ser ejecutadas en un orden arbitrario.

La Figura 12 muestra un ejemplo de modelo de flujo de proceso para describir la creación de un nuevo contacto.

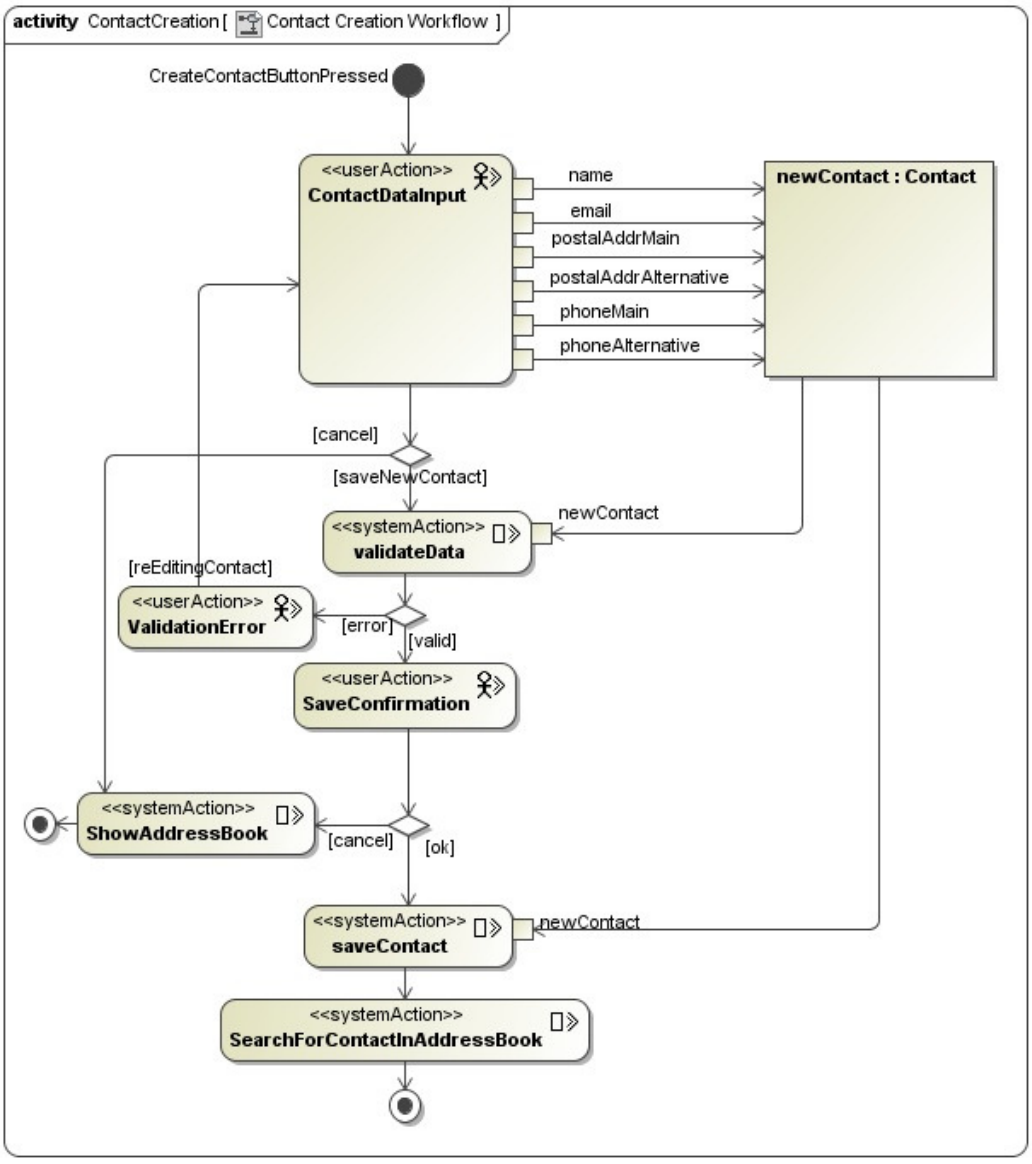


Figura 12: Modelo de flujo de proceso para la creación de un contacto.

3.*REFACTORINGS* PARA MEJORAR PROCESOS DE NEGOCIO EN APLICACIONES WEB

Con el avance de las aplicaciones web que ponen a disposición diversos procesos de negocio, los usuarios ejecutan cada día más tareas a través de Internet. El número de operaciones que se puede realizar en estos tipos de páginas es muy extenso y, hoy en día, ya son parte de la vida cotidiana de muchos usuarios.

Este nuevo medio de operar ha cambiado la forma que las personas realizan determinadas actividades y, si antes era necesario ir al banco o a una agencia de viajes para realizar una transferencia bancaria o comprar un pasaje aéreo, hoy se puede lograr lo mismo a través de una computadora conectada a la red, sin salir de la casa o de la oficina.

Este nuevo hábito de utilizar aplicaciones web para ejecutar procesos de negocios centró la atención en una variable indispensable en este ambiente: la usabilidad. Las empresas que poseen su versión online, ven la necesidad de mejorar la usabilidad de sus sitios a fin de captar nuevos usuarios y posibles consumidores. Garantizar en la página web una experiencia agradable al usuario, es garantizar que él pueda realizar una determinada operación o compra sin mayores complicaciones o dudas. Dificultades en el uso pueden ser el motivo por que el usuario abandone esa página y frecuente la de otra empresa de la competencia.

La manera en que las actividades de un proceso de negocio son ejecutadas en una aplicación web, depende o es consecuencia del diseño de los procesos y de las restricciones impuestas por las reglas de negocio que están por detrás del modelo. De esta manera, la calidad del diseño de los procesos de negocio influye directamente en la calidad de las experiencias del usuario [56]. Lograr mejoras en la usabilidad de aplicaciones web, con el fin de que la ejecución de los procesos de negocio sea más fácil e intuitiva para el usuario, es el foco principal de este trabajo de tesis.

Con tal motivo se propondrá un catálogo de *refactorings* para optimizar la ejecución de los procesos de negocio específicos para cada modelo, focalizando en mejorar los criterios de: usabilidad, eficiencia, accesibilidad, credibilidad y productividad, en las aplicaciones web.

Explicitar pasos de un proceso de negocio, crear accesos directo para facilitar que el usuario tenga un acceso rápido a una proceso muy utilizado, posponer pasos de un proceso para que el usuario tenga acceso al máximo de información posible o agregar la posibilidad de cancelar un proceso en la mitad de su ejecución, son algunos de los *refactorings* considerados como mejoras en el uso de procesos de negocio en las aplicaciones web, que van a ser explorados en más detalles en este trabajo.

Como esta tesis está focalizada en *refactorings* para mejorar los procesos de negocio en aplicaciones web, se considera que el comportamiento de la aplicación no fue alterado si los requerimientos y las reglas de negocio propias del proceso de negocio para el cual se plantea la intervención, no se ven alterados y el usuario puede obtener el mismo resultado al ejecutar el proceso, antes y después de la aplicación del *refactoring*.

Para su tratamiento el tema se dividirá en sub-secciones organizadas de la siguiente manera: Sección 3.1 describe los campos de aplicación de los *refactorings* empezando por los *refactorings* internos aplicados para mejorar características externas y *refactorings* en aplicaciones web que mejoran características externas como la usabilidad; Sección 3.2 describe el nuevo campo de aplicación de los *refactorings* abordado en esta tesis, *refactoring* para mejorar procesos de negocio y como los *refactorings* propuestos impactan en los distintos modelos (modelo de proceso, navegación y presentación); Sección 3.3 ejemplifica los casos de estudio que serán abordados en la tesis, las páginas de *E-Commerce* y las páginas de *Home Banking*.

3.1. Aplicación existente de *Refactoring*

3.1.1.*Refactoring* Interno

Refactoring se define como el proceso de cambiar un sistema de software sin cambiar su comportamiento externo y mejorando su estructura interna [12][45]. La técnica de *refactoring* se usa para mejorar la estructura del diseño del sistema ya que es común que el código vaya creciendo debido a sucesivas iteraciones. A medida que surgen nuevos requerimientos, mejoras y/o modificaciones en el sistema, el código fuente originalmente sencillo se complejiza alejándose cada vez más de su diseño inicial, con una consecuente disminución de su calidad en tanto producto de software [13]. Así, el término "*refactoring*" se utiliza para describir la reestructuración en el contexto de la programación orientada a objeto, tomando como entrada un código modificado en varias iteraciones que no obstante resulta obsoleto y del cual se devuelve un código con buen diseño, extensible y mantenible. En virtud de lo expresado, Fowler define *refactoring* como una manera disciplinada de limpiar el código minimizando las chances de introducir errores [12].

Inicialmente esta técnica era aplicada básicamente por desarrolladores de software y consiste en aplicar pequeñas mejoras en el código fuente reorganizando el programa sin cambiar su comportamiento, con el objetivo de mejorar su estructura interna. La idea es redistribuir clases, variables y métodos por la jerarquía con la finalidad de facilitar adaptaciones y extensiones futuras [13].

De las diversas razones para aplicar *refactoring* en el código, se destacan las siguientes:

- **Mejora el diseño del sistema**
  Las repetidas modificaciones hechas en el código de un sistema para cumplir con objetivos a corto plazo o cambios sin analizar el impacto en el diseño de la aplicación hacen que el diseño del programa empeore ya que el código va perdiendo su estructura [12].

- **Reduce la complejidad de software y mejora legibilidad del código**
  Refactoring produce mejoras en la estructura interna del sistema, haciendo una limpieza general en el código a través de pequeños pasos, como remover duplicaciones, mover un campo de una clase a otra, mover código por la jerarquía. Esas pequeñas modificaciones permiten que el código quede más ordenado, menos complicado y por eso más fácil de ser entendido por cualquier desarrollador que lo vea.

- **Reduce costos con mantenimiento de software y facilita cambios futuros**
  Un código que pasa por *refactoring* deviene en un código más entendible y más fácil de mantener. Gran parte del costo total de un proyecto de desarrollo de software se gasta en mantenimiento y mejoras debido a que el código es o se ha vuelto confuso, extenso y desordenado [13]. Eso demanda de más tiempo para entenderlo y eventualmente modificarlo. Teniendo un código limpio y lógico, es más fácil para el desarrollador descifrar de qué se trata y como está estructurado, costándole menos el cambio y mantenimiento del mismo.

- **Ayuda a encontrar errores en el código**
  Un código limpio, ordenado y mejor estructurado permite clarificar su comprensión detectando problemas antes no identificables debido a la complejidad y desorden del código.

- **Aumenta la velocidad de desarrollo**
  Esta es una de las principales ventajas de tener un buen diseño, poder desarrollar más rápido. Teniendo una lógica más comprensible y mejor diseño, los desarrolladores tienen mejor visión del código y pueden desarrollarlo más rápido, evitando la pérdida de tiempo que supone de descifrar el código, lidiar con duplicidades o depurar errores.

- **Preserva el comportamiento del software**
  Una de las principales premisas del *refactorings* es que mismo después de la aplicación de los cambios en el código, el software debe mantener el mismo comportamiento que tenía antes de la aplicación del *refactoring*.

Más allá de las ventajas mencionadas, una de las inquietudes más frecuentes entre los desarrollados refiere a ¿cuándo aplicar *refactoring* en el código? Fowler [12] sugiere aplicarlo mediante lo que él llama de "Regla de tres", lo cual supone que:

- Se agrega una nueva funcionalidad al sistema.

- Se arregla un error o se hace *debug* del código

- Se realiza una revisión del código del programa.

Otra forma muy común de identificar cuando "refactorizar" es analizando si un código posee "*bad smells*" o, en otras palabras, cuando el código huele mal. Esta expresión refiere detectar determinadas estructuras en el código que sugieren la posibilidad de aplicar *refactoring* ya que afectan negativamente algunas propiedades del sistema como mantenibilidad, testibilidad, extensibilidad y reusabilidad.

Se han detectado indicios de estructuras que se caracterizan como "*bad smell*" y que ya poseen una aceptación de que se tratan de casos que necesitan la aplicación de *refactoring*, a saber:

- **Código duplicado**
  Partes de código que están duplicadas en más de un lugar en el código, como tener la misma expresión en 2 métodos en la misma clase.

- **Método muy extenso**
  Cuanto más extensa es el método, más difícil de entender que lo hace.

- **Clase muy extensa**
  Clases que con mucho código y que tratan de ejecutar muchas funciones son una fuente de redundancias y duplicación de código.

- **Larga lista de parámetros**
  Extensas listas de parámetros también son difíciles de comprender porque se convierten inconsistentes y difíciles de usar ya que los parámetros son constantemente modificados siempre que se necesita de más datos.

- **Clase Perezosa**
  Clases que no aportan mucho valor y por eso deben ser eliminadas.

Aplicación del *Refactoring*

Para aplicar *refactoring* es necesario tener en cuenta y ejecutar las siguientes actividades:

1. Identificar las partes del software que deberían ser "refactorizadas".
2. Definir que *refactorings* deben ser aplicados a las partes identificadas.
3. Garantizar que el *refactoring* preserve el comportamiento del software.
4. Aplicar el *refactoring.*
5. Evaluar el efecto del *refactoring* en las características de calidad del sistema (como complexidad, legibilidad, mantenibilidad) o del proceso (costo, esfuerzo, productividad).

Después de refactorizado el código, es importante mantener la consistencia con los otros elementos del sistema como documentación, documentos de diseño, requerimientos, especificaciones, *tests*, etc. [13].

Como mencionado previamente, otra importante característica del *refactoring* es que, por definición, después de su aplicación, el comportamiento externo del software no debe verse alterado, o sea el comportamiento debe ser preservado. Aunque una definición precisa de esta premisa de "comportamiento preservado" no existe y generalmente es muy difícil y caro probar que el comportamiento de un software no fue alterado [13][46]. Opdyke [44] afirma que para preservar el comportamiento del software se debe garantizar que, para los mismos valores ingresados, los mismos resultados deben ser retornados, antes y después de la aplicación del *refactoring*. Más allá de lo sugerido por Opdyke, se considera que su definición de "comportamiento preservado" no es suficiente cuando se amplía el uso del *refactoring* a otros dominios diferentes del código, como es el caso de los modelos de aplicaciones web abordados en esta tesis.

Ampliando las definiciones anteriores, Mens [46] propone que, para garantizar que el comportamiento del software no se vea alterado, se debe asegurar que el usuario pueda ejecutar el mismo conjunto de métodos antes y después de la aplicación del *refactoring*.

3.1.2.*Refactoring* en Aplicaciones Web

Como se mencionó, *refactoring* constituye el proceso de transformaciones que preservan el comportamiento del programa y fue pensado inicialmente para mejorar la mantenibilidad y extensibilidad

del código fuente de los programas. Estas transformaciones en el código mejoran su calidad interna haciendo que sea más fácil de entender y mantener, sin alterar su funcionalidad [12].

El *refactoring* fue pensado originalmente como una técnica utilizada para reestructurar la jerarquía de la clase de un diseño orientado a objeto [30]. Sin embargo el concepto del *refactoring* se viene extendiendo a otros elementos del proceso de desarrollo de software y su aplicación visa mejorar no solamente aspectos internos del código fuente. Actualmente existen diversas investigaciones que aplican *refactoring* a otros contextos como base de datos [32], HTML [33] y UML [31]. Muchos de eso *refactorings* no influyen en la calidad interna del software pero si en factores externos como performance, accesibilidad, usabilidad de aplicaciones web, entre otros [7][15][16].

Las aplicaciones web son bastante propicias para la aplicación de *refactorings* debido a su constante evolución y cambios en los requerimientos posibilitando su aplicación tanto en el área de la implementación como en los modelos. Los *refactorings* en el nivel de implementación son muy similares a los tradicionales que trabajan con el código fuente de una aplicación, pero en este caso la diferencia es que el código fuente estará escrito en lenguajes como HTML, Javascript, XML y no en lenguajes orientadas a objeto como originalmente fue pensado. Reestructuraciones en el código de las aplicaciones web fueron descriptos en [77] donde los autores definieron distintas categorías de reestructuración, como por ejemplo, actualizaciones de sintaxis y mejoramiento interno de las páginas. Los cambios se aplican sobre el código escrito en PHP, Javascrip y HTML que, como se mencionó, son los lenguajes usados en el desarrollo de aplicaciones web.

3.1.3. *Refactorings* para mejorar la Usabilidad en Aplicaciones Web

Como se mencionó, los cambios y mejoras aplicados en los modelos de las páginas web, permiten que el usuario pueda acceder a la misma información y las mismas funcionalidades sin alterar o comprometer el comportamiento de la aplicación. Los *refactoring* en aplicaciones web son clasificados no solo por el modelo en que son empleados pero también por el factor de usabilidad que quieren mejorar.

Existen una serie de factores que están directamente ligados a la usabilidad de una aplicación web y pueden contribuir con la misma. Estos son:

- **Accesibilidad**: Capacidad de la página de ser accesible a la mayoría de las personas, incluso a aquellas con alguna discapacidad.

- **Navegabilidad**: Calidad de estructura de navegación del *web site*, relacionada con cómo están distribuidos los links y la facilidad de la misma en función de la obtención de un determinado contenido de la página.

- **Efectividad**: Capacidad de la página de ofrecer un rápido entendimiento para usuarios, más o menos experimentados y hasta usuarios nuevos.

- **Credibilidad**: Relación de confianza que la página transmite a los usuarios.

- **Comprensibilidad**: Factor relacionado con la transmisión clara de los propósitos y objetivos de la página y el fácil acceso a las funcionalidades que se ofrecen a los usuarios.

- **Personalización**: Habilidad de incorporar recomendaciones y personalizaciones de acuerdo con experiencias pasadas del usuario.

- **Facilidad**: Facilidad de aprender a usar la página web y con el debido soporte.

Los factores citados anteriormente son pasibles de ser mejorados mediante la aplicación de *refactorings* [7], ayudando a generar mejoras en el modo como el usuario interactúa con la aplicación web.

Para saber si una aplicación web necesita de algún tipo de *refactoring*, es necesario evaluar e identificar los "*bad smells*" de la aplicación o sea, problemas o indicios de que algo no anda bien en la página y podría ser mejorado. Para detectar estos "*bad smells*" existen varios métodos como por ejemplo: *testing* con usuarios, análisis de la página por expertos en usabilidad o evaluación heurística; entre otros. El objetivo es testear la página verificando los principios de usabilidad. Una vez detectados los problemas (*bad smells*), se aplican las mejoras (*refactorings*).

Los problemas identificados son clasificados según los modelos (de navegación, de presentación o de proceso) y conectados al factor de usabilidad al que afectan. El impacto del problema identificado debe ser analizado tomando en consideración el tipo de aplicación al que está afectando. La identificación de los *bad smells* y la decisión de que *refactoring* qué debe ser aplicado en cada caso depende de la capacidad del desarrollador para analizar los problemas encontrados y elegir en el catálogo cual es el *refactoring* que mejor soluciona el problema [7].

Después de haber aplicado los cambios necesarios a fin de eliminar los "*bad smells*" de la aplicación, es necesario evaluar si los *refactorings* aplicados produjeron una mejora en la aplicación y si la experiencia del usuario fue beneficiada con estos cambios. Para evaluar esto, se puede utilizar, nuevamente, al usuario final quien avalará si las modificaciones introducidas en la página mejoraran o no su experiencia en la aplicación. Ejecutar *testings* de usabilidad después de la aplicación de los *refactorings* es un termómetro para evaluar de la calidad lograda con los mismos. Cuando los usuarios realizan las mismas tareas ejercidas antes de la introducción de los cambios, pueden decir si observaron mejoras en su ejecución, en el tiempo de realización, o en la disminución de los errores anteriormente encontrados.

3.2. Nueva Aplicación de *Refactoring*

3.2.1. *Refactoring* para mejorar Procesos de Negocio

El uso de aplicaciones web que manejan procesos de negocio crece a cada día y tiene millones de usuarios en todo el mundo. En todos los países, en cualquier momento, siempre hay alguna persona comprando algún libro u otros productos en una tienda virtual, o haciendo una transferencia bancaria, o comprando entradas para el cine por Internet, o averiguando los vuelos en la página de alguna compañía aérea. Si hoy en día, el usuario conectado a la red accede a muchas funcionalidades mediante las cuales puede realizar las más variadas tareas, se debe al hecho de que hoy las aplicaciones web soportan los procesos de negocio. En

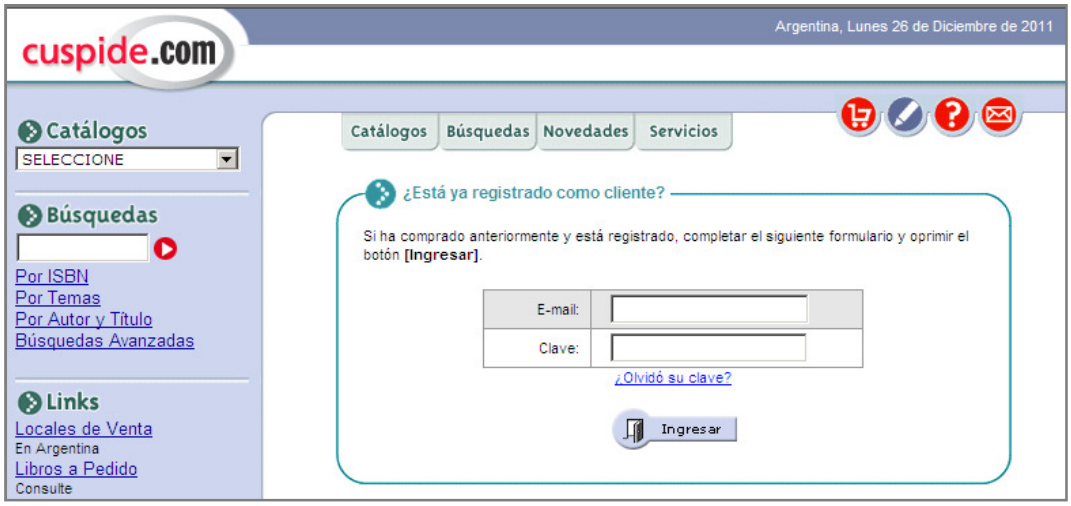
esas aplicaciones los usuarios están constantemente ejecutando procesos a fin de realizar alguna tarea en particular, sea una compra, una búsqueda, una consulta o un pago online.

Todos los procesos disponibles realizados diariamente, más allá de su apariencia sencilla o simple, suponen una ejecución sumamente compleja, porque subyacen en ella una intrincada red lógica y diversas reglas de negocio a cumplir. El usuario tiene plena participación en su conclusión, pues la continuidad del proceso depende de las informaciones que él vaya proveyendo, según los pasos determinados previamente por el desarrollador, así deberá: llenar formularios, seleccionar opciones, aguardar verificaciones del sistema, etc. Todas estas acciones impactan en la experiencia del usuario incidiendo en su grado de conformidad con la aplicación web, por eso los procesos deben estar adecuadamente modelados y diseñados, para no comprometer la usabilidad de la página.

Sin embargo, obedecer todas las reglas de negocio por detrás de los proceso, sin comprometer la usabilidad es un desafío grande para los desarrolladores, quienes no siempre pueden lograr un equilibrio afectando muchas veces la usabilidad que queda como un punto a mejorar a futuro. No es difícil encontrar páginas en la web donde el usuario no entiende el proceso que está llevando a cabo, tiene que repetir una y otra vez sus pasos, o se siente confundido sin saber cuánto le falta para finalizar una tarea. Los *refactorings* propuestos en esta tesis pretenden introducir mejoras y aspiran a hacer más amigable y eficiente la experiencia del usuario al ejecutar eses procesos de negocio en aplicaciones web.

Uno de los muchos problemas enfrentados por los usuarios durante la ejecución de procesos de negocio es no saber cuántos pasos componen el proceso que está ejecutando. En la parte superior de la Figura 13 se muestra el proceso de *checkout* en la tienda virtual de Cuspide (www.cuspide.com.ar). Este sitio no posee ninguna indicación respecto al progreso del proceso y el usuario no sabe cuántos pasos le faltan para finalizar la compra. La Figura 13 en su parte inferior, muestra asimismo el *checkout* en la tienda virtual de Amazon (www.amazon.com), en la cual se observa una barra de progreso que informa al usuario los pasos que ya ejecutó y los que aún le faltan ejecutar.

Ese cambio, que puede parecer menor a simple vista, se entiende que mejora notablemente la usabilidad de la página y es un ejemplo de lo que consideramos un *refactoring* en esta tesis.



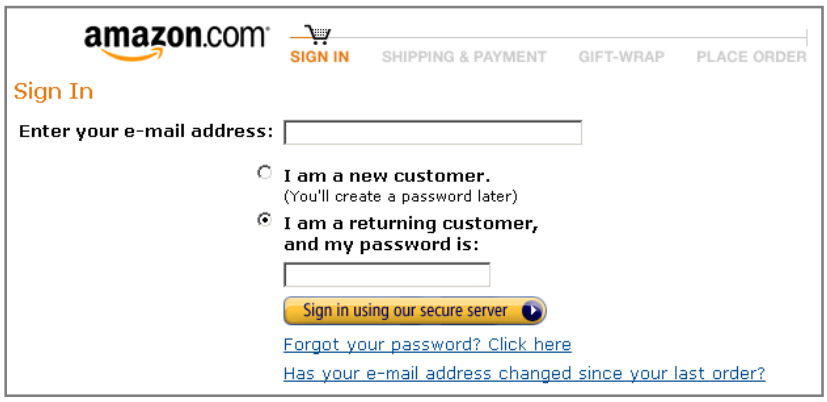


Figura 13: Página de *checkout* en Cuspide (arriba) y Amazon (abajo). La última con la barra de progreso.

Webber [24] define una serie de *refactorings* para modelos de proceso que tiene como objetivo mejorar calidades internas de los mismos, atendiendo por tanto su mantenibilidad, comprensibilidad y claridad, con lo cual logra facilitar el trabajo de mantenimiento de los desarrolladores y reducir costos de cambios en el código.

Por su parte esta tesis también sugiere un catálogo de *refactorings* en los procesos de negocio, pero al contrario de lo planteado por Webber, en este caso se pensó en optimizar las características externas de los mismos, como usabilidad, accesibilidad, navegabilidad, eficiencia, confiabilidad, entre otras. Estas mejoras impactarán directamente sobre la percepción que los usuarios tengan de la página, mejorando sus experiencias de ejecución frente a los procesos de la aplicación web, hecho que se estima incidirá directamente en el consumo que haga el usuario en esas páginas.

Los *refactorings* propuestos en esta tesis son definidos como:

- Cambios para mejorar la ejecución de los procesos de negocio en las aplicaciones web.
- Mejoras en la forma que el usuario interactúa con los procesos en las aplicaciones web, sugiriendo cambios tanto en el proceso en si como en la forma de acceder al mismo en la página o en la forma como son presentados.

Los *refactorings* deben preservar los requerimientos y casos de uso del proceso de negocio sin cambiar sus reglas de negocio, por eso una vez aplicados los *refactorings* en la página, *tests* de aceptación deben ser ejecutados con el fin de verificar que la funcionalidad del proceso no fue alterada y que sigue cumpliendo con las reglas definidas previamente.

Como se mencionó anteriormente los t*ests* de aceptación con usuarios son de gran importancia tanto en la etapa inicial, para identificar *bad smells* [10], como en la fase final para corroborar que la funcionalidad principal de la aplicación web se mantuvo después o a pesar de la aplicación de los *refactorings* y que mejoras fueron logradas.

Con base en las definiciones y restricciones descriptas anteriormente, los *refactorings* en los procesos de negocio pueden realizar, entre otras, las siguientes operaciones:

- Cambiar la estructura del proceso, cambiando el orden de las actividades dentro del flujo;

- Agrupar, remover o agregar actividades en el flujo;
- Cambiar, agregar o remover elementos de la UI de la aplicación web, a fin de mejorar la presentación y la eficiencia de los procesos de negocio;
- Mejorar la forma de navegación dentro de los proceso de negocio, acrecentando o eliminando elementos que disparan el proceso;
- Agregar o remover links para mejorar la navegación en el proceso;
- Cambiar la agrupación o el aspecto de elementos de la UI con el objetivo de ser más intuitivo para el usuario entender el proceso.

En los próximos capítulos, se definirán los *refactorings* que se emplearán en tres modelos de aplicaciones web: modelo de proceso, modelo de presentación y modelo de navegación. En estos capítulos se describirán *refactorings* específicos para cada uno de esos modelos, siguiendo la estructura abajo detallada:

- **Motivación**: En esa sección se describen las razones por las cuales se motiva la aplicación del *refactoring* sugerido, identificando los problemas enfrentados por los usuarios al utilizar la página y los "*bad smells"* que deben ser atacados.
- **Mecanismo**: Se describe la secuencia de pasos que deben ser seguidos a fin de aplicar el *refactoring* a la aplicación web.
- **Ejemplo**: En esta sección se ilustra ejemplo reales de aplicaciones web modeladas a través de diagramas UWE, presentando el antes y después del *refactoring* sugerido.
- **Mejorar Intencionadas**: Se listan las intenciones de mejora en las aplicaciones web después de la aplicación de los *refactoring*.
- **Refactorings Relacionados**: Si se aplica, son listados los *refactoring* pertenecientes a este catálogo que se relaciona con el *refactoring* descripto.

3.2.2. Clasificación de *Refactorings* por modelo

Como mencionado anteriormente, los *refactorings* aplicados a modelos de proceso, de navegación y de presentación tienen como objetivo mejorar características externas y la experiencia del usuario enriqueciendo su forma de interactuar con los procesos en la aplicación web.

Estos tres modelos fueron elegidos por que son de incidencia directa en el manejo de los procesos de negocio en las aplicaciones web, y lo hacen mostrando su comportamiento, describiendo la navegación hacia y durante el proceso y como éste se presenta. El modelo de contenido a su vez no modela procesos y es un modelo que describe solamente los elementos estáticos de la aplicación web.

Aunque todos los *refactorings* propuestos se ajustan, en teoría, a cualquier tipo de aplicación web independiente de la tecnología con la que estas fueron desarrolladas o de su metodología de diseño, cabe a los desarrolladores y/o diseñadores evaluar si la aplicación de los *refactorings* no altera el comportamiento o alguna regla del negocio de los proceso de negocio.

Con el objetivo de guiar a los desarrolladores en su elección y optimizar sus decisiones sobre los *refactorings* más apropiados para aplicar en la aplicación web, éstos se clasificaron según el modelo en que se aplican, como será detallado en los siguientes capítulos.

3.3. Casos de Estudio

En función de profundizar este trabajo de tesis se han elegido dos tipos de aplicaciones web que contienen una gran cantidad de procesos de negocio embebidos y al mismo tiempo, son ampliamente utilizadas en la actualidad por usuarios de todo el mundo para realizar variadas tareas y procesos online. Se trata de: Páginas de *Home Banking* y de Comercio Electrónico (también conocidas como *E-Commerce*). Tal selección responde al deseo de ilustrar a partir de ejemplos concretos, la lección y aplicación de los *refactorings* para mejorar los procesos de negocio involucrados en estas aplicaciones.

Los procesos de negocio hasta hace poco tiempo atrás no eran soportados en las aplicaciones web y, a partir del momento que fueron integrados a las mismas, una nueva gama de actividades pudieron pasar del ámbito presencial al ámbito online. Con ese nuevo mecanismo, los usuarios pueden ejecutar diversas tareas siguiendo un determinado número de pasos indicados, hasta concluir el proceso en la página web, muchas veces sin salir de la comodidad de su casa.

Con el foco en la comodidad y la practicidad, este tipo de aplicaciones creció rápidamente, tornándose cada día más populares. Hoy en día hacer una transferencia bancaria, consultar el saldo o hacer una compra por internet ya es una rutina para muchos y por eso es importante garantizar la usabilidad ya que el número de usuario que utilizan esos servicios aumenta exponencialmente a cada día. A continuación se describirán en más detalles esos dos tipos de aplicación web, mencionando tanto las características generales como las particularidades de cada una de ellas.

3.3.1. *E-Commerce*

El término *E-Commerce* significa *Electronic Commerce* o Comercio Electrónico y es frecuentemente utilizado para referirse a la compra y venta a través de Internet. Es muy común que la mayoría de las personas asocie el termino solamente a sitios de venta por Internet como Amazon, pero *E-Commerce* involucra más que eso.

Muchos se refieren al *E-Commerce* como todas las transacciones electrónicas entre una organización y un tercera parte. Loshin y Vacca [80] definieron *E-Commerce* como el uso de poder de la información digital para entender las necesidades y preferencias de cada cliente, permitiendo entregar productos y servicios lo más rápido posible. Awad [82], describe *E-Commerce* como la acción de vender mercancías y servicios vía Internet. En tanto Kalakota y Whinston [79], en su libro, proveen una definición más amplia del término cuando dicen que *E-Commerce* tratase de:

- **Una perspectiva de Comunicación**: Entrega de informaciones, productos, servicios o pagos por medios electrónicos.
- **Una perspectiva de Proceso de Negocio**: Aplicación de tecnología para la automatización de transacciones de negocio y *workflows*.

- **Una perspectiva de Servicio**: Aumento de la velocidad y calidad del servicio al mismo tiempo que se reduce los costos de la empresa.

- **Una perspectiva online**: Compra y venta de productos y informaciones online.

La industria de *E-Commerce* surgió y maduró en menos de una década en los Estados Unidos y tubo su periodo de explosión entre los años 1995 y 2000. La posibilidad de hacer negocios electrónica y virtualmente desde cualquier lugar donde se halle una computadora conectada a Internet trajo nuevas alternativas a las empresas y un gran cambio en el comportamiento de los consumidores en general, cambiando la manera de hacer negocios y creando un nuevo mecanismo de compra y ventas. Para calibrar la incidencia de este tipo de aplicación en el comercio internacional es importante tener en cuenta que, solo en 2004, la industria de comercio electrónico generó más de US$ 100 billones de dólares en negocios de ventas por menor [82].

Debido al popularización de Internet y con usuarios que pasan cada día más tiempo conectados a la red, sea por la computadora o mediante un teléfono inteligente (los también llamados *smart phones*), las empresas identificaron una oportunidad de ampliar sus negocios captando a los usuarios que navegan diariamente transformándolos en consumidores en potencial, creando una nueva forma de comprar sin salir de casa. Por esa razón el fenómeno del *E-Commerce* sigue creciendo a grandes proporciones, captando nuevos consumidores alrededor del mundo, todos los días.

Actualmente se puede decir que la mayoría de las empresas posee negocios en el ambiente web para aprovechar de los beneficios ofrecidos por esta nueva forma de negocio. Según Chaffey [78] y Awad [82], las empresas interesadas en la creación de su propia página de *E-Commerce*, concretaron ese emprendimiento a partir de reconocerlos como principales los siguientes beneficios:

- **Potencial crecimiento de las ventas**: Aumento de ventas en las empresas debido al mayor alcance de clientes alrededor del mundo, lo que no era posible antes. Al contrario de una tienda física que solo alcanza clientes locales, el *E-Commerce* genera posibilidades de venta a nivel global, alcanzado consumidores en todo el mundo.

- **Aumento de las ganancias**: Realizar transacciones electrónicamente en lugar de manualmente permite que las empresas puedan ahorrar en costos operativos, obteniendo más ganancias en las ventas. Según Awad [82], el costo de procesar un ticket convencional de una compañía aérea en 2007 era de US$ 8. El mismo ticket, procesado electrónicamente (llamado de e-ticket) costaba, en el mismo año, US$ 1. Como se puede ver a partir de los valores suministrados, con la versión electrónica la rentabilidad en favor de la empresa crece notablemente, además logran tener más control y flexibilidad, economizando tiempo y dinero.

- **Reducción de costos operacionales**: Con el *E-Commerce* se reducen problemas de logística, y de costos, con la entrega de productos electrónicamente. Para las empresas vender por internet es ventajoso además porque les permite ahorrar en gastos de personal, alquiler del espacio, transportes, entre otros.

- **Mejor Servicio al Consumidor**: Existe una mejora en la comunicación con los clientes ya que los mismos tienen acceso a todas las informaciones relacionadas a su cuenta online, promoviendo que el cliente sea auto-suficiente y que cada vez dependa menos de ayuda. No obstante, en el caso que el usuario necesite consultar, existen diversas formas de contactarse con las empresas, como por ejemplo a través de *chat* online, que permite clarificar las dudas en el momento.

- **Aumento de la velocidad en el acceso a la información**: Publicar informaciones en las páginas web permite que los usuarios busquen por las informaciones que desean y las respuestas para las cuestiones que puedan tener. La empresa IBM pudo ahorrar más de US$ 800 millones de dólares solo en publicar respuestas a problemas técnicos de sus productos en su página web, permitiendo así que los usuarios pudiesen desenvolverse solos, con autonomía. Con esas publicaciones redujeron la necesidad de personal especializado para atenderlos.

Más allá de los beneficios mencionados con relación a los E-Commerce, cabe asimismo decir que, desarrollar, desplegar y mantener esos sistemas no es una tarea fácil debido a la complejidad que existe por detrás de ellos. Debido a esta complejidad, muchas veces las empresas necesitan reestructurar los procesos de negocio existentes con el fin de maximizar los beneficios de alcanzados con estas aplicaciones [80].

De entre todas las áreas de *E-Commerce* existentes, se evidencian dos áreas prioritarias en que las compañías hacen negocios online: Servicio Financieros y Compra y venta a través de marketing directo [80].

- Servicios Financieros: Este tipo de aplicaciones ayudan a los clientes, empresas e instituciones financieras, al distribuir en la web gran cantidad de información financiera, permitiendo fácil acceso y amigable operatividad. Las aplicaciones de *Home Banking* es un ejemplo de este tipo de aplicación que viene creciendo bastante en los últimos años permitiendo que los usuarios realicen una gran variedad de operaciones online. Más detalles sobre este tipo de aplicación se podrá leer en la sección 3.3.2 de este capítulo.

- **Compra y Venta a través de Marketing Directo**: La venta directa de productos y servicios a través de internet fue el primero tipo de comercio electrónico y probó ser un espacio de triunfo para muchas compañías. Historias de éxito como es el caso de Amazon, Dell, y la introducción de la venta de e-tickets por las compañías aéreas demuestran el gran crecimiento de esta área.

Las aplicaciones de *E-Commerce* permiten que los usuarios realicen diversas tareas, como por ejemplo: Buscar por productos de un catálogo a través de una serie de filtros; Conocer e examinar revisiones de los productos hechas por otros usuarios; Comprar productos usando tarjeta de crédito u otros medios de pago similares; Rastrear el progreso de un pedido, entre otras. Para posibilitar que el usuario realice estas tareas, las páginas de *E-Commerce* deben soportar una cantidad de funciones, entre ellas se pueden mencionar las siguientes [63]:

- **Gestión de Stock**: Las aplicaciones deben ser capaces de mantener un registro del estoque real de los productos para ser consistente con lo que se muestra en la aplicación.
- **Gestión de Pago**: Refiere al pago de los proveedores, manteniendo registro del pago de los clientes y de las compañías de tarjeta de crédito.
- **Entrega**: Gestionar el proceso de entrega de los productos pedidos en el sitio.
- **Análisis de mercado**: Análisis de los productos vendidos para tener una estadística de cuáles son los productos más/menos vendidos.

Actualmente, en Internet existe un incontable número de tiendas virtuales que ofrecen la posibilidad de realizar todas esas tareas online. Hoy en día se puede encontrar las más variadas opciones de productos disponibles en aplicaciones de *E-Commerce* como ropas, zapatos, piezas electrónicas, muebles, computadoras, pasajes aéreos, paquetes turísticos, comida y hasta autos. El rubro de compras online fue

creciendo y es muy popular entre los usuarios, resultando tres de cada cinco los consumidores que realizan compras online regularmente, según Reynolds [62] y Ince [63].

Tanto éxito evidencia que los usuarios encuentran diversas ventajas con esta forma de transacciones Los consumidores perciben así que, realizar compras online redunda en su:

- **Conveniencia**: La facilidad de comprar en cualquier lugar (con acceso a Internet) y en cualquier hora del día o de la noche, de acuerdo con la conveniencia del cliente y sin las restricciones de horarios que existen en las tiendas físicas. No hace falta movilizarse hasta la tienda o esperar que alguien lo atienda, basta tener una computadora o un teléfono con conexión a Internet (*smart phone*). Para manejar el "retraso" en la posesión del producto comprado (ya que el cliente no recibe el producto en el mismo momento), las aplicaciones ofrecen entregas rápidas para suavizar la espera. Otro beneficio es que el usuario puede cotejar el precio del producto de su interés en diversos sitios de *E-Commerce* para certificarse que está comprando el más conveniente.

- **Economía de tiempo**: Comprar desde casa o desde trabajo permite al usuario manejar su propio tiempo sin estar atado a variables como el tránsito, los horarios de funcionamiento de las tiendas o el tiempo que supone transportarse hasta el local. En cambio las compras online se pueden realizar durante las 24 horas del día, los 7 días de la semana y puede realizarse rápidamente siempre que el usuario sepa lo que desea comprar, sin enfrentar colas o tiempo de espera en ser atendido. Todo en el confort de su casa o oficina.

- **Personalización**: Una de las características principales de los sitios de *E-Commerce* es la posibilidad de personalización que ofrecen. El usuario puede elegir lo que quiere ver y las aplicaciones se amoldan a sus necesidades mostrándoles ofertas de acuerdo con su historial de compras. En la sección 3.3.1.1 se define con más detalles la personalización en esas aplicaciones.

- **Gran variedad de productos disponibles**: Con el inmenso número de aplicaciones de *E-Commerce* disponibles en la web, el usuario tiene acceso a una enorme variedad de productos, de distintas marcas, de distintos precios, distintos países a tan solo a un clic de distancia. Aparte del ahorro de tiempo que eso permite, también es una ventaja poder comparar o buscar el mismo producto en diferentes lugares, todo a través de la computadora. Además cuando el usuario no encuentra un producto en un sitio de *E-Commerce*, puede cambiar a otro rápidamente para continuar sus averiguaciones.

- **Disponibilidad de Informaciones**: En la web es posible tener acceso a una gran cantidad de información antes de decidir la compra online. Se puede tener acceso a una pesquisa técnicas con respecto al producto que desea comprar, al vendedor y la empresa que vende el producto; acceder a aclaraciones comerciales referidas a las formas de pago, de envío, garantías en caso de desperfectos, etc. Se pueden leer revisiones, especificaciones técnicas, fórums, ver fotografías, comparar con productos similares y hacer preguntas online a un representante, todo antes de decidir comprar o no el producto deseado. Todo ese acervo de información digital ayuda al usuario a tomar decisiones más conscientemente respecto del producto que compra.

- **Precios competitivos**: No es raro ver ofertas disponibles apenas para compras realizadas en las páginas web de determinada empresa. Las empresas suelen ofrecen descuentos y promociones

en su página web para estimular la compra online; así es posible acceder a importantes descuentos, entregas sin costo o promociones relámpago. Estas promociones se deben a que el costo de los productos se reduce como resultado del bajo costo operativo que supone mantener una página de E-Commerce. Los precios competitivos aliados a la calidad de los productos ofrecidos son importantes factores para construir la lealtad del cliente, esencial en los días de hoy ya que la competencia es tan feroz [81].

Con tantos sitios de comercio electrónico disponibles en la red, la gran competencia crea un desafío a los desarrolladores de las páginas de *E-Commerce*: mejorar la experiencia del usuario creando un entorno agradable y fácil de usar. Como dijo Nielsen [14], la usabilidad guía la web y, en el caso de los sitios de *E-Commerce*, define si el usuario compra o no un producto.

Si el usuario no puede encontrar en la página web el producto de su interés, la compra no se realizará, por lo tanto, la página habrá fallado en su objetivo principal que es vender. Para lograr más ventas, este tipo de aplicación debe mantener el usuario el mayor tiempo posible en la página, satisfecho y comprando cada vez más. Por tal razón, la facilidad con que el usuario encuentra el producto que desea, tanto la facilidad con que realice el proceso de compra es crucial y es lo que determina si el usuario permanece en la página o si la abandona y va para la página de la competencia. Según Nielsen [64], una página de comercio electrónico puede aumentar significativamente sus ventas, no raras veces duplicándolas, apenas mejorando su usabilidad.

Como dicho anteriormente, construir aplicaciones de *E-Commerce* no es una tarea sencilla. Frecuentemente aparte de las reglas de negocio, este tipo de aplicación debe también soportar distintos perfiles de usuario que acceden a grandes volúmenes de datos simultáneamente. Otro factor a considerar es que la navegación y la presentación de la página deben estar bien definidas para garantizar que la aplicación sea usable y el usuario la perciba amigable [42].

Estudios realizados en páginas de *E-Commerce* [64], identificaron una lista con los principales problemas y razones por las cuales los usuarios abandonan una página web. El gráfico mostrado de en la Figura 14 es una reproducción adaptada del gráfico mostrado en [64], página 26, donde se muestra el resultado de *testings* de usabilidad en aplicaciones de *E-Commerce.* A pesar de reflejar datos adquiridos en un *testing* de usabilidad, los porcentajes exhibidos en los gráficos pueden significar más daños de lo que está ilustrado. Como ejemplo se puede citar el porcentaje del precio: 5% no es un número alto, pero debe considerarse que los usuarios que realizaron el *test* no estaban gastando su propia plata, y si la situación saliera del ámbito de prueba para una situación real, es muy probable que los usuarios que se quejaron del tema de precio, no hubiesen proseguido con la compra, al contrario de lo que ocurrió en el *test* donde la compra fue concluida a pesar de este problema.

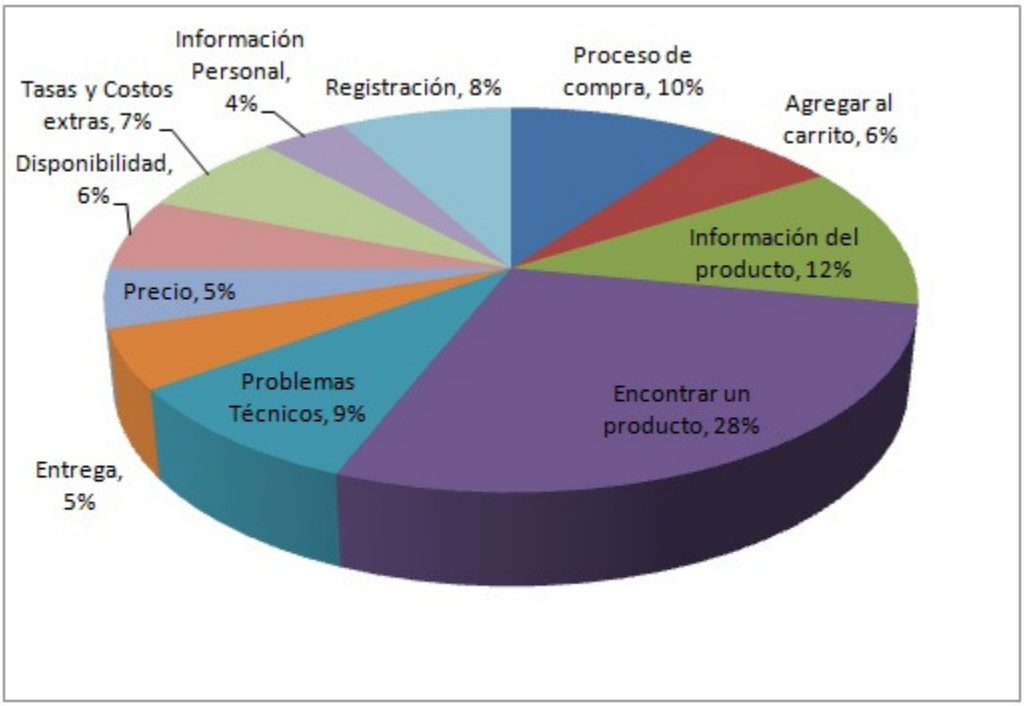


Figura 14: Problemas comunes en páginas de *E-Commerce*

Respecto al gráfico en la Figura 14, se describen a continuación algunos de los problemas:

- Problemas para encontrar el producto deseado: En muchas páginas de *E-Commerce*, muchos usuarios no pueden siquiera encontrar lo que estaban buscando en la aplicación y terminan desistiendo de seguir con la búsqueda.

- **Proceso de compra muy extenso y complicado**: Una de las razones más comunes que impiden que el usuario concluya la compra es porque encuentran dificultades en proseguir con el proceso de *checkout* o no hay informaciones importantes referentes al proceso, por ejemplo, conocer de cuantos pasos está compuesto;

- Proceso de Registración: Cuando el proceso de registración requiere muchas informaciones o es requerido antes de que el usuario sepa de los costos totales o informaciones relevantes respecto al producto, los usuarios tienden a no proseguir, abandonando la página;

- Precio: Precios elevados o precios que no son exhibidos inmediatamente al usuario son razones que hacen que el usuario desista o no vuelva a página.

- Tasas y Costos Extras: No saber el costo final de la compra en etapas tempranas del proceso es una de las razones que pueden sorprender negativamente al usuario y hacerle desistir de comprar;

- Informaciones del Producto: La falta de informaciones detalladas sobre los productos en la página puede hacer el usuario dudar si se trata de lo que necesita o no y resultando que el mismo busque por más informaciones en otras páginas, abandonando la actual;

- **Entrega**: La preocupación con la fecha de la entrega de los productos es responsable por 5% de los problemas en las páginas de *E-Commerce*. Usuarios que buscan por un producto para regalar a alguien por

un cumpleaños, por ejemplo, estaría muy interesado en saber la probable fecha de entrega y podría desistir de la compra caso esta no cumpla con sus expectativas.

Como se mencionó, las páginas de comercio electrónico poseen una extensa cantidad de procesos de negocios embebidos y esos procesos deben ser bien definidos para garantizar una buena experiencia a los consumidores que entran en la página para comprar. Por ejemplo, el proceso de compra (conocido en muchos sitios como *Checkout*) es uno de los principales procesos disponibles en una aplicación de *E-Commerce* y requiere atención especial de los desarrolladores. Como se trata de un conjunto de operaciones que involucra también dinero, el usuario debe sentirse seguro y entender los pasos que tiene que seguir para completar el pago del producto elegido. Cuanto más extenso, confuso y complejo es esta secuencia, la tendencia es que los usuarios desistan y migren a páginas similares. Los procesos de negocios deben guiar el usuario, ser auto-explicativos e intuitivos, ofreciendo al mismo la máxima información posible, conquistando su confianza.

3.3.1.1. Personalización de Aplicaciones Web

Debido al grande número de sitios de *E-Commerce* y la competencia que existe en el mundo virtual, los desarrolladores tienen el desafío de lograr páginas cada vez más atractivas para los usuarios. Uno de los recursos más utilizados últimamente es la personalización de las aplicaciones web. Una aplicación web personalizada es aquella que cambia su vista de acuerdo con el perfil o el interés del usuario, amoldándose a lo que este necesite. Las páginas de *E-Commerce* ven en la personalización una forma de adelantarse a lo que el usuario desea basándose en su historial de compras y de búsquedas, o en las categorías que están clasificados internamente en el sistema. Así, basadas en el conocimiento de las prioridades de los usuarios, las aplicaciones ofrecen productos similares a los que compró poniendo a su disposición links que le sean muy relevantes, es decir, que puedan motivarlo para una nueva compra [41].

Por ejemplo, si el usuario compró o buscó por una lapicera en una página de *E-Commerce*, la próxima vez que entre en la página, esta tendrá disponibles links con otras opciones de lapiceras y/o artículos de oficina, como cuadernos, anotadores, kit de oficinas, etc. estarán entre las sugerencias de la aplicación.

Rossi et al. en su paper [28], identificó algunos patrones de los sistemas de recomendación, también discutido por Schafer et al. en [27], que personalizan la navegación de las aplicaciones web. Con este mecanismo de personalización, las páginas pueden ofrecer un diferencial en relación a las tiendas físicas: tratar de predecir lo que al usuario le gustaría de acuerdo con la necesidad y preferencias de cada uno. Este tipo de aplicaciones son de realización muy complejas y tienen el objetivo de manejar y satisfacer las necesidades del usuario ayudándolo a encontrar lo que desea y facilitando el proceso de compra. No obstante, la identificación de esos patrones ayuda a los desarrolladores a organizar la información y diseñar interfaces más usables.

Para personalizar una aplicación web existen dos posibilidades: o bien se propone a través de links que se hacen disponibles en la página o a través del contenido que es exhibido ante el usuario.

El primero enfoque cambia los links mostrados habitualmente en la página por links que son relevantes o similares a aquellos en los que el usuario puede estar interesado. Este mecanismo cambia la navegación de la aplicación de acuerdo con el usuario que accede la página, mostrando links que llevan a nodos distintos. El

segundo enfoque se focaliza en la información que exhibe al usuario en la página, centralizando en el contenido de la página más allá de los links [41].

3.3.2. *Home Banking*

Como fue descripto en la sección anterior, las páginas de *Home Banking* son un tipo de aplicación de *E-Commerce* utilizadas para gestionar informaciones financieras a través de Internet. Esta nueva forma de acceso a la información bancaria surgió en 1995 juntamente con el nacimiento de las páginas de comercio electrónico. Su aparición revolucionó los hábitos de las personas en relación a la administran sus finanzas. Las páginas de *Home Banking* se multiplicaron rápidamente y fueron adoptabas por millones de usuarios en todo el mundo por constituir una forma rápida y sencilla de realizar las más variadas transacciones bancarias sin salir de casa, como el propio nombre sugiere.

A través de ellas los usuarios pueden realizar una variedad de transacciones que antes podían ser hechas únicamente de manera presencial, en la sede de la agencia bancaria correspondiente. Chequear el saldo de las cuentas, realizar transferencias, realizar pagos de cuentas, etc., están entre las actividades más frecuentes realizadas por los usuarios online.

Los clientes de las instituciones bancarias adquirieron, con el surgimiento de los *Home Bankings,* autonomía para la realización del movimiento de sus cuentas, un mejor acceso a la información de sus finanzas y mayor claridad con relación a cómo hacer una inversión o, a las condiciones para conseguir un préstamo, por ejemplo.

Así como las páginas de comercio electrónico percibieron una oportunidad de disminuir los costos, el sector financiero también identificó en Internet una posibilidad de acotar gastos. Lograr que trámites que únicamente podían ser realizados en los locales bancarios estén disponibles online constituyó una ventaja no sólo para los usuarios, sino también para los mismos bancos, pues con ello bajaron costos operativos en personal y en el funcionamiento mismo de las agencias. Otro beneficio en común con las páginas de *E-Commerce* es la ganancia que se consigue al realizar las transacciones electrónicamente en lugar del método convencional. Estudios [83] muestran que una operación bancaria realizada en una agencia tiene un valor medio de entre US$ 1 y US$ 4, mientras la misma operación realizada online cuenta US$ 0,05 centavos, siendo, por lo tanto, extremamente conveniente para esas instituciones proveer un servicio de calidad en el ambiente web, capaz de atraer la mayor cantidad de usuarios posible. Los beneficios son bastante atractivos, tanto que incluso existen bancos que existen solo en la versión online [83].

Actualmente, las páginas de *Home Banking* es un servicio ofrecido por la mayoría de los bancos y es una herramienta ampliamente utilizada por gran parte de los clientes que tienen acceso a Internet. Este tipo de aplicación web tiene una tendencia de crecimiento considerable y significativa ya que brinda practicidad y comodidad al usuario al ahorrarle tiempo de espera en las colas de banco y brindarle la posibilidad de hacer desde su casa, transacciones que, hace pocos años atrás eran imposibles.

Hoy en día, el usuario corriente puede, clave de acceso por medio, ingresar al sistema de *Home Banking* y allí realizar una diversidad de operaciones como por ejemplo: consulta de saldo, consulta de movimientos pendientes, consulta de consumo de la tarjeta de crédito, obtener créditos o prestamos, realizar

transferencias, hacer inversiones, comprar dólares, pagar facturas, etc., todo online y sin salir de la comodidad de su casa u oficina.

Un estudio realizado por *Online Banking Report* muestra el progreso del uso de la web por las instituciones bancarias entre los años de 1995 y 2002. Esto puede observarse en la Tabla 1 que es una reproducción de la tabla publicada en [78], en ella se expone la comparación del crecimiento de las páginas de *Home Banking* en este periodo. Si bien estos datos son de hace 10 años atrás, los mismos resultan significativos para mostrar comparativamente a explosión de las páginas de *Home Banking* y la alta aceptación que estas tuvieron entre los usuarios.

Tabla 1: Siete años y medio de Web Banking [78].
Fuente: Online Banking Report Estimates, +- 25%, 11/02, Online Banking Report, Número 89, 10 de Diciembre de 2002.

Métrica	**Mayo 1995**	**Diciembre 2002**
Instituciones Financieras con web *banking*	1	6.000
Instituciones Financieras con páginas web	50	14.000
Total de online *banking*	5 millones	100 millones
Total de online *banking* en EEUU	300.000	28 millones
Tráfico mensual de transacciones bancarias y con tarjetas de crédito online en E.E.U.U.	100.000	50 millones
Cantidad mensual de aplicaciones de crédito presentadas online en E.E.U.U.	0	1.5 millón

Actualmente ya es parte de la rutina de muchas personas acceder a sus cuentas personales y realizar transacciones a través de Internet. La aceptación ha sido tan grande y los beneficios tan significativos que los bancos ya están ampliando las aplicaciones de *Home Banking* para un nuevo dispositivo, los teléfonos.

La gran aceptación de este tipo de aplicación por parte de los usuarios se debe a diversas razones, entre las cuales se citan:

- Flexibilidad de horarios independiente del horario de funcionamiento de las agencias bancarias.
- Reducción de la dependencia entre el usuario y la agencia ya que el usuario puede hacer trámites y transacciones solo, sin tener que dirigirse a la agencia.
- Economía de tiempo ya que no necesita movilizarse o enfrentar colas en las agencias.
- Seguridad y privacidad en la realización de las transacciones.
- Rapidez en el acceso a las informaciones y en la realización las operaciones.
- Libertad para operar y manejar su dinero de la forma que deseen.

Las aplicaciones de *Home Banking* pueden ser consideradas una de las aplicaciones más estables y de mayor éxito hoy en día. Toda la practicidad ofrecida por este servicio es posible debido al avance de las aplicaciones web.

Como ya se dijo en otras oportunidades, actualmente las aplicaciones web soportan procesos de negocio y por consecuencia tienen que respetar un sin número de estrictas reglas de negocio que garanticen la consistencia de los datos, la seguridad de la información y la ejecución de las operaciones en el ambiente web. Las páginas de *Home Bankings* no escapan a estas exigencias, sin embargo son particularmente complejas pues también deben considerar la seguridad de las operaciones realizadas online. La seguridad es un punto de extrema importancia en este tipo de página, en las que el objetivo primordial es el manejo del dinero de los clientes, quienes exigen que se les garantice la integridad de su patrimonio. En consecuencia, este tipo de páginas debe además incluir mecanismos que atiendan a la confidencialidad y a la confiabilidad de los datos de manera de captar la tranquilidad de los usuarios.

La mayoría de las transacciones disponibles en las páginas de *Home Banking* son complejos procesos de negocio que necesitan ser completados hasta el final para concluir con una operación. Transferencias, pagos de cuentas, aplicación de fondos de inversión, todos son ejemplos de procesos de negocios que están embebidos en las páginas de los bancos. La manera como se realizan es esencial para el éxito de la operación ejecutada y por eso es importante que, en este tipo de aplicaciones, los mecanismos de actuación estén claramente definidos a fin de promover una experiencia sencilla, usable y amigable al usuario. Además por manejar un asunto tan delicado como las finanzas de los usuarios, estos procesos necesitan ser intuitivos y garantizar transparencia para quien lo está ejecutando, haciendo que este pueda realizarlo sin dudas, más allá de trasmitir la seguridad que siempre le ofreció el banco.

Las páginas de *Home Banking*, así como las páginas de *E-Commerce* descriptas anteriormente, como se dijo son aplicaciones web de marcada popularidad en todo el mundo y poseen millones de usuario que utilizan sus servicios. Muchas veces, por tratarse de aplicaciones que manejan un nivel alto de complejidad, la usabilidad es un factor subestimado y no tenido en cuenta.

En virtud de eso, en los próximos capítulos serán propuestos *refactorings*, aplicados a distintos modelos, que plantean cambios a fin de mejorar la usabilidad y la experiencia del usuario al ejecutar todos los procesos de negocio embebidos en este tipo de aplicación.

4. *REFACTORINGS* SOBRE EL MODELO DE PROCESO

4.1. Definición

Los procesos de negocios son definidos como una secuencia de actividades que permiten que el usuario pueda alcanzar un objetivo o ejecutar alguna tarea en particular [9][25]. El diagrama de proceso muestra todas las actividades de los procesos que están embebidos en la aplicación web detallando su comportamiento y cómo funcionan internamente. Los *refactorings* definidos sobre este modelo proponen cambios en los pasos de los procesos a fin de mejorar la experiencia del usuario mientras ejecutan los mismos.

Esos *refactorings* permiten cambios en la estructura de los procesos pero deben preservar las siguientes características:

- El resultado final del flujo normal del proceso;
- El resultado final de los flujos alternativos del proceso;

En base a lo mencionado anteriormente, las siguientes operaciones pueden aplicarse a los modelos de proceso sin alterar las características descriptas:

- Reemplazar *userAction* por un *systemAction;*
- Cambiar el orden de las actividades de proceso;
- Agrupar, agregar o remover actividades del proceso;

Al contrario de los diagramas diseñados usando la notación BPMN, que permiten especificar un proceso como una transacción [48], actualmente los diagramas de actividad UML no soportan el concepto de transacciones y, por esa razón, las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) no pueden ser aplicadas en este caso. En el caso de que pudieran ser soportadas, las propiedades ACID deberían ser preservadas [25].

Es importante mencionar que, en general, los *refactorings* aplicados en el modelo de proceso tienen impacto también en los modelo de navegación y de presentación.

4.2. Catálogo

4.2.1. Agrupar Validaciones

Motivación

En los procesos cortos, la gran cantidad de validaciones de datos consume tiempo del usuario y complica sin necesidad el proceso, que debería ser simple. Por esa razón, reducir el número de validaciones durante la ejecución de un proceso hace que este sea más rápido, dinámico y objetivo, como debe ser un proceso pequeño.

El propósito de este *refactoring* es simplificar el proceso de validación evitando que se haga una por cada campo informado por el usuario, lo que se traduce en la reducción de la interacción entre el servidor y el cliente, y consecuentemente el tiempo de respuesta y tiempo de espera por parte del usuario. Otra ventaja es que al agrupar las validaciones se reduce también la cantidad de pasos necesarios para completar el proceso, ya que en este caso las validaciones no son transparentes al usuario (o sea, las validaciones no son realizadas internamente por el sistema) y exigen que el mismo interactúe en el proceso presionando los botones para completar las mismas.

El número excesivo de validaciones confunde al usuario, hace que la ejecución del proceso deje de ser fluida y prolonga el tiempo de espera mucho más allá de lo necesario. Validaciones realizadas precipitadamente en transcurso de un proceso simple generan ruido innecesario, por eso es importante que sean evitadas en un etapa temprana del mismo. El *refactoring* "Agrupar Validaciones" propone que estas validaciones se agrupen y se retrasen en el final de la carga de datos, validándolos todos juntos, resultando que el proceso sea más sencillo para el usuario. Este *refactoring* es indicado para ser aplicado a procesos pequeños, no muy extensos, donde la cantidad de campos a validar no es numerosa, como por ejemplo, el proceso de Login a un sistema.

Este *refactoring* representa un cambio en la estructura del proceso, donde se hace una agrupación de actividades para reducir los pasos y hacer que sea más rápido para el usuario.

El "*bad smell*" que motiva este *refactoring* sería el tiempo excesivo que se pierde en validar cada campo individualmente y navegar al siguiente campo a ser completado.

Mecanismo

Este *refactoring* propuesto puede ser realizado en cinco pasos:

1. Identificar todos los <*userActions*> del proceso que se validan individualmente.

2. Agrupar los <*userActions*> identificados en el paso 1 creando un nuevo <*userAction*>. Insertar el nuevo <*userAction*> en la posición del primer <*userAction*> del grupo. Colocar en el nuevo <*userAction*> todos las salidas (*output pins*) que estaban en cada uno de los <*userActions*> individuales.

3. Identificar cada una de las validaciones del sistema (<*systemActions*>)

4. Crear un nuevo <*systemAction*> que representa el conjunto de los <*systemActions*> identificados en el paso 3 y ubicarlo en la posición donde se encontraba el último de los <*systemActions*> individuales que se agruparon.

5. Crear un nuevo diagrama de actividad mostrando cómo se compone el nuevo <*systemAction*> que contiene todas las validaciones agrupadas.

Ejemplo

El ejemplo para este *refactoring* se refiere a un proceso de Login de la página de *Home Banking* del Banco Nación Argentina (www.bna.com.ar), como muestra la Figura 15. En esta página el usuario tiene que ingresar su nombre de usuario y presionar el botón "Ingresar" para que se efectúe la validación del usuario ingresado. Después de la validación, el usuario tiene que ingresar la clave y nuevamente presionar el botón "Ingresar" para que se valide la clave de seguridad correspondiente a esta persona.

Las dos validaciones son realizadas en páginas distintas, lo que hace que el usuario tenga que esperar que después de la primera validación se cargue la segunda página, para recién llenar y validar los próximos datos.

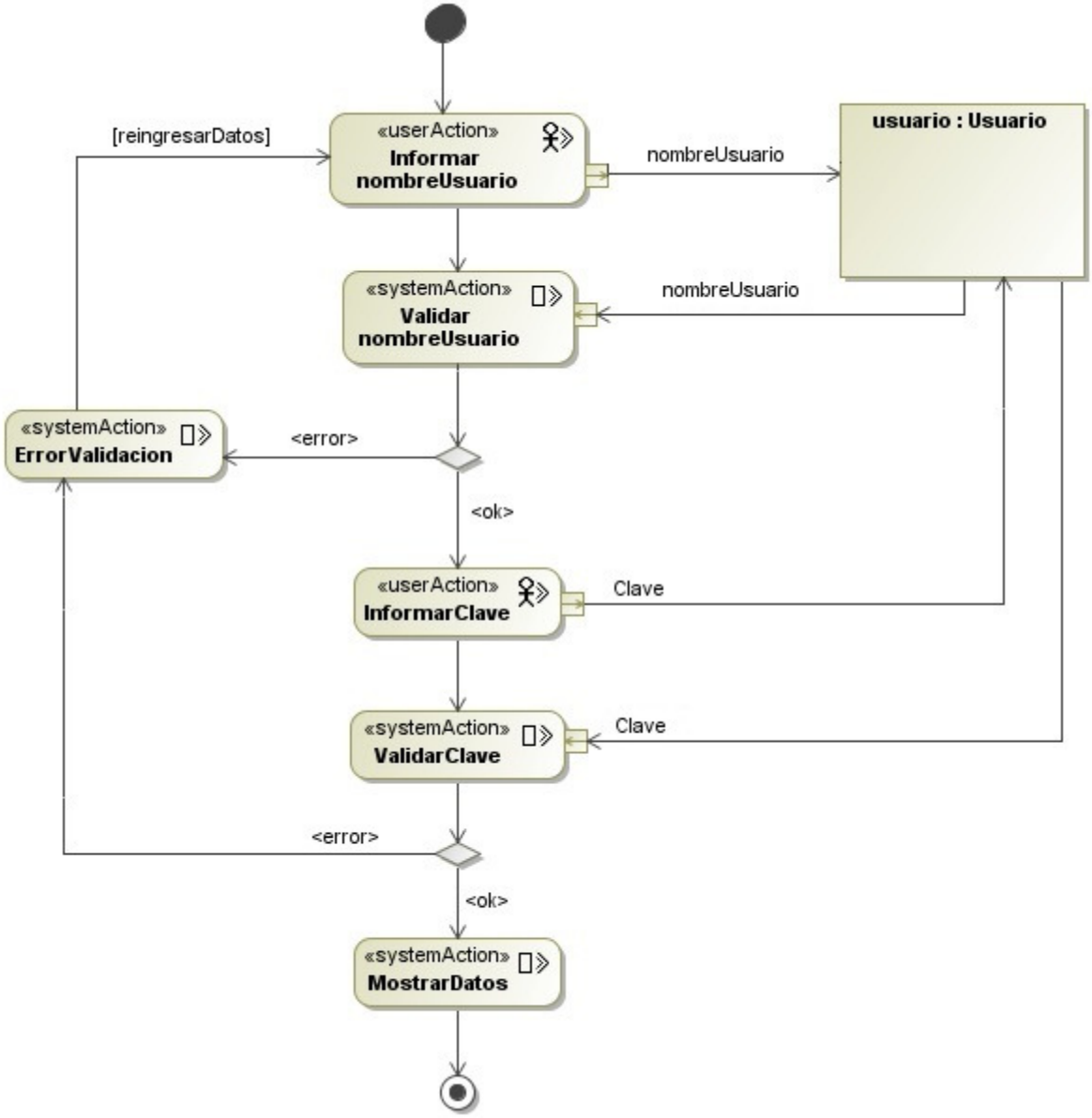


Figura 15: Proceso de login de la página de *Home Banking* del Banco Nación Argentina antes del *refactoring*

Se identificaron dos validaciones individuales en la página: la de usuario y la de contraseña. Luego, para evitar que el usuario tenga que ingresar datos y esperar por dos validaciones en dos momentos distintos, las dos validaciones fueron agrupadas. Con eso el usuario tiene que entrar los datos al mismo tiempo y en la misma página y presionar el botón "Ingresar" una sola vez. Una vez hecho eso, el sistema se encarga de hacer interna y centralizadamente las dos validaciones, o sea: verificar si el usuario existe y si la contraseña ingresada corresponde al usuario aludido.

La Figura 16 muestra la refactorización de este proceso, donde ambas validaciones fueron centralizadas en un solo paso.

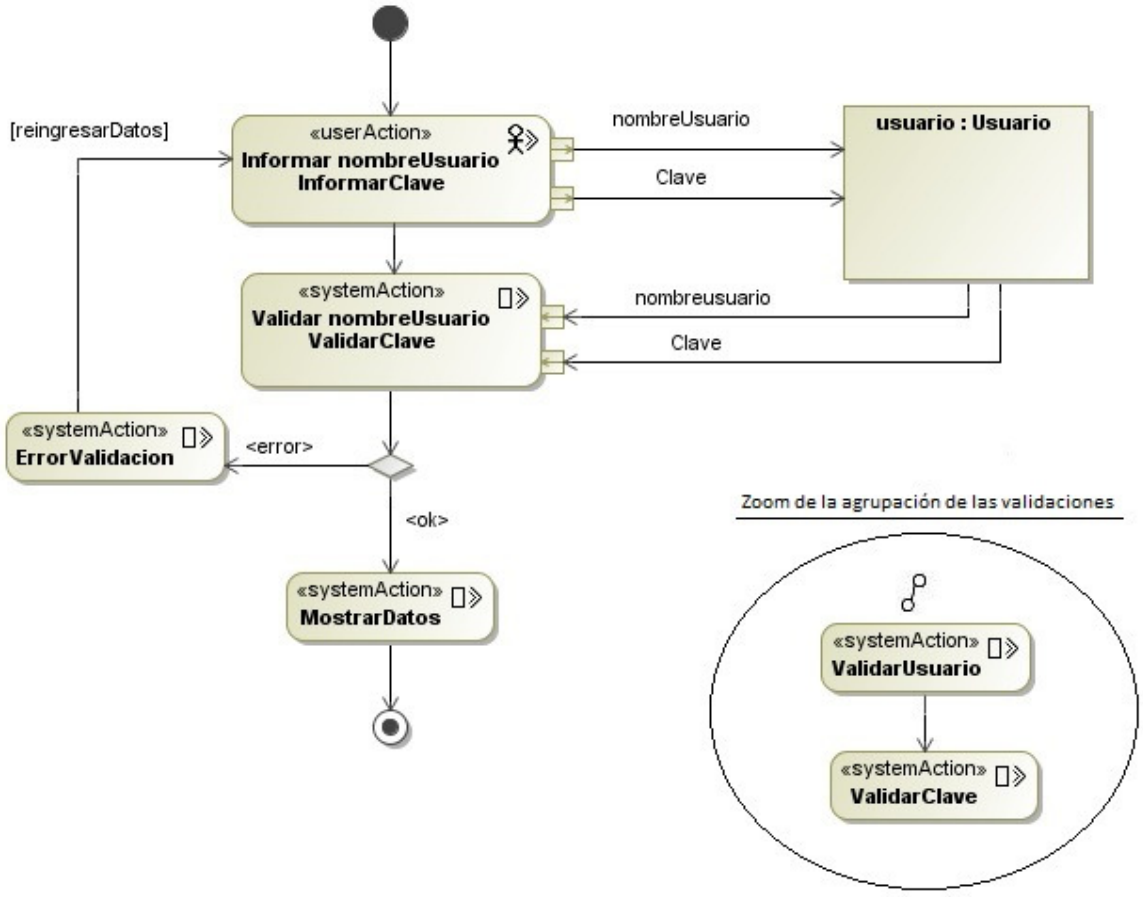


Figura 16: Proceso de login de la página del Banco Nación Argentina después del *refactoring* "Agrupar Validaciones"

El mismo problema puede ser visto en otras páginas de *Home Banking* como el *web site* del banco *Nation Wide* (www.nationwide.co.uk) y del banco Santander del Reino Unido (www.santander.co.uk).

Mejoras Intencionadas

- Reducción del número de pasos del proceso ya que algunos fueron agrupados;
- Reducción del tiempo de espera del usuario para ejecutar el proceso ya que se eliminó el tiempo de carga de la segunda página;
- Centralización de los pasos y de las validaciones del formulario en una sola página, como se verá en más detalles en el capítulo 6.2.1;

Refactorings Relacionados

Este *refactoring* impacta no sólo en el diagrama de proceso como también en el de presentación de la página. El cambio estructural que se realiza con este *refactoring* se refleja también en el modelo de presentación, como está descripto en el capítulo 6.2.1: "Agrupar validaciones en la misma página".

Otro *refactoring* relacionado está descripto en la sección 4.2.5 de este capítulo: "Anticipar Validaciones", que sugiere el opuesto de este *refactoring.*

4.2.2.Posponer Registración

Motivación

Estudios e investigaciones muestran que gran parte de los usuarios tienden a abandonar un sitio web cuando este les solicita una registración anticipada. Este problema es recurrente en sitios de comercio electrónico en los que el usuario es obligado a registrarse antes de empezar el proceso de *checkout* o de saber detalles relevantes de su compra. Hacer que el usuario se registre en el sitio antes de iniciar el proceso de compra es citado por Nielsen [64] como una de las quejas frecuentes de los usuarios en las aplicaciones web y es clasificado como una de las principales razones de abandono de compras en sitios de *E-Commerce*.

El *refactoring* "Posponer Registración" propone aplazar la registración para el final del proceso cuando el usuario ya posee todas las informaciones necesarias para confirmar y proseguir con la compra. Posponer la registración posibilita que el usuario pueda evaluar factores importantes para la compra como el precio del flete, el valor de impuestos o la fecha de entrega, antes de optar por comprar. Hacer una registración en el inicio del proceso puede ser frustrante, más todavía si el usuario no está registrado y tiene que llenar largos formularios con informaciones personales antes de poder proseguir.

El "*bad smell*" que se ataca con este *refactoring* es el abandono prematuro del sitio web.

Mecanismo

Este *refactoring* puede ser realizado en tres pasos:

1. Identificar dentro del diagrama de proceso la secuencia de pasos que corresponde al proceso de registración en el sitio web.

2. Identificar dentro del diagrama de proceso todas las actividades que pueden ocurrir sin necesitar de la registración del usuario.

3. Desplazar todo el proceso de registración identificado en el paso 1, para después de la etapa identificada en el paso 2.

Ejemplo

El ejemplo para este *refactoring* describe un proceso de compra en el sitio de la librería online Tematika (www.tematika.com). Como mostrado en la Figura 17, el usuario es obligado a registrarse en el sitio antes de

saber detalles de la compra como los costos de envío y beneficios ofrecidos por la página. Un usuario nuevo tiene que llenar un extenso formulario con datos personales y adicionales antes de ingresar las informaciones relacionadas con la compra y antes mismo de saber cuánto es el costo final de la misma.

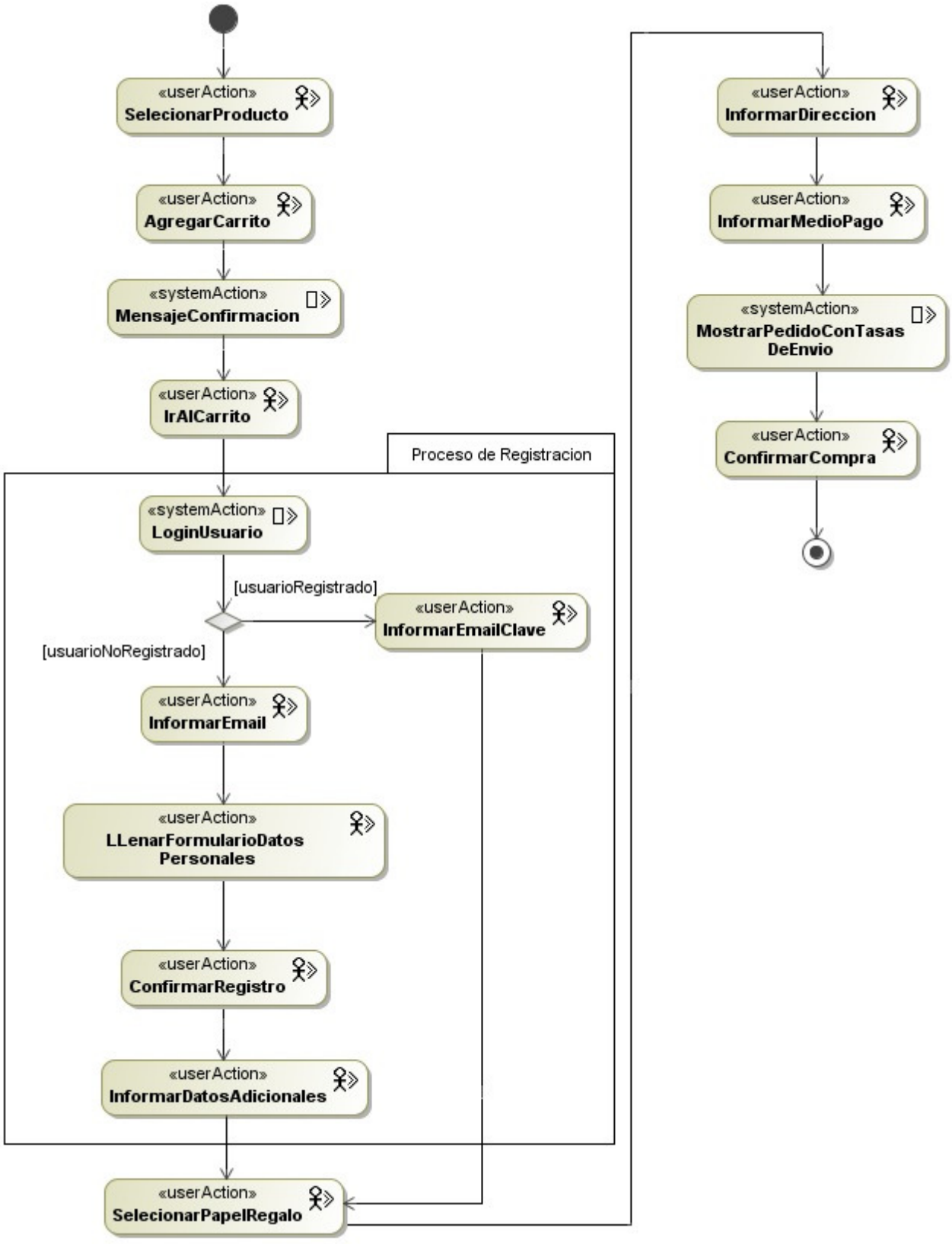


Figura 17: Proceso de compra en Tematika antes del *refactoring*

El *refactoring* “Posponer Registración” propone demorar el proceso de registración permitiendo que el usuario tenga acceso a informaciones relevantes del producto que pretende comprar antes de tener que completar extensos formularios de registración o antes de hacer login en la aplicación. De esta manera el usuario solamente tiene que hacer login o registrarse una vez que esté seguro de la compra.

La Figura 18 muestra el proceso de registración pospuesto para el final del proceso de compra, como resultado de la aplicación del *refactoring* "Posponer Registración".

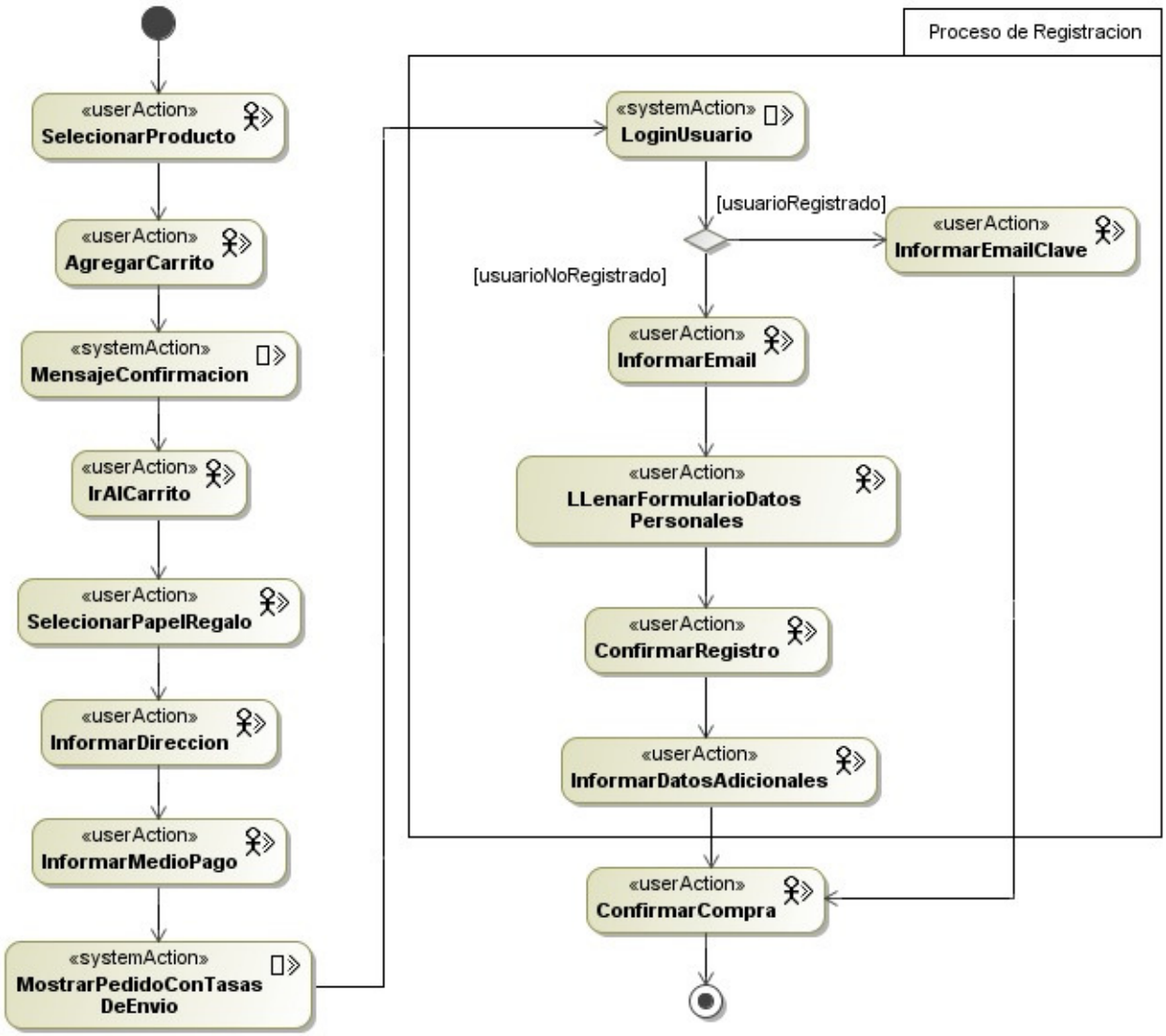


Figura 18: Proceso de compra en Tematika después del *refactoring* “Posponer Registración”

Este *refactoring* tiene mayor aplicación en sitios de comercio electrónico o de ventas de tickets, donde el proceso de registración es necesario para concluir con una compra de un producto. Podemos identificar este comportamiento en páginas como Amazon (www.amazon.com) u otros similares y también en otro tipo de páginas como es el caso del área de clasificados en el diario online www.atarde.com.br.

Mejoras Intencionadas

- Aumentar la confianza del usuario en la página ya que al momento de hacer login ya tiene acceso a toda la información relevante de la compra;
- Evitar la incomodidad que supone llenar largos formularios al principio del proceso al usuario no registrado;
- Evitar que el usuario tenga que pasar innecesariamente por el proceso de registración o login en caso que no prosiga con la compra;
- Disminuye la posibilidad de abandono en la página, ya que el usuario no tiene pasar por el proceso de registración sin necesidad.

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* del modelo de presentación "Exhibir costo extra del producto" descrito en el capítulo 6.2.3 que destaca la importancia de mostrar al usuario todas las informaciones relevantes a la compra en etapas tempranas del proceso.

4.2.3.Reemplazar *<userAction>* por *<systemAction>*

Motivación

Un proceso de negocio está compuesto de <*systemActions*> y <*userActions*>. Los <*systemActions*> corresponden a todas las actividades procesadas internamente por el sistema y que son transparentes para el usuario, como: validaciones de datos o búsqueda de datos en el sistema. Por otra parte, un proceso depende del usuario, que es quien lo alimenta, proveyendo los datos necesarios para que el proceso de negocio continúe. Esas acciones son llamadas de <*userActions*>, y son ejemplo de ellas, la entrada de datos o el accionar de los botones.

En los procesos de negocio embebidos en las aplicaciones web, cuanto menos el usuario tenga que interactuar con el sistema (cuanto menos <*userActions*>) mejor, porque cuanto menos se solicite al usuario entrar datos, ejecutar acciones o responder a preguntas, más intuitivo y rápido será el proceso y menos laboriosa su ejecución. Por estas razones, este *refactoring* propone hacer una análisis de todos los <*userActions*> del proceso con el objetivo reemplazarlos por <*systemActions*>, siempre que sea posible. En otras palabras, este *refactoring* busca hacer que, si es posible, el sistema ejecute algunas acciones que hasta entonces eran realizadas por el usuario, haciendo que estas se resuelvan internamente y sin la intervención del mismo.

El *bad smell* que vulnera este *refactoring* es la cantidad excesiva de datos que los usuarios tienen que completar cuando hay posibilidad de obtener esos datos automáticamente.

Mecanismo

Este *refactoring* puede ser realizado en cuatro pasos:

1. Identificar dentro del diagrama de proceso las actividades ejecutadas por los usuarios, los <*userActions*>.

2. Identificar el o los <*userActions*> identificados en el paso 1 que puedan ser reemplazados por un <*systemAction*>.

3. Crear un nuevo <*systemAction*> para cada <*userAction*> identificado en el paso 2 asociándolos a los datos que se necesite acceder.

4. Eliminar los <*userActions*> que fueron identificados en el paso 2 e insertar en su lugar el/los nuevo(s) <*systemAction(s)*>, conectándolo(s) a través de nuevos *processLinks* según corresponda.

Ejemplo

El ejemplo de este *refactoring* describe un proceso de registración de grandes sitios de comercio electrónico de Brasil, Submarino (www.submarino.com.br) y Saraiva (www.livrariasaraiva.com.br). Estos sitios, así como la mayoría de las páginas de *E-Commerce*, requieren que un usuario nuevo se registre proveyendo información personal, como nombre, e-mail, dirección, datos de tarjeta de crédito, etc. El proceso de registración es generalmente largo y requiere mucho del usuario haciendo que este ingrese gran cantidad de datos en los formularios mostrados en las páginas.

En el proceso de registración de una de las páginas seleccionadas, se identificó una sección que demanda del usuario llenar muchos campos, la sección "Dirección"; en ella se pide al usuario que ingrese uno por uno los siguientes datos: Código Postal, País, Calle, Número, Barrio, Provincia y Ciudad como se observa en la Figura 19.

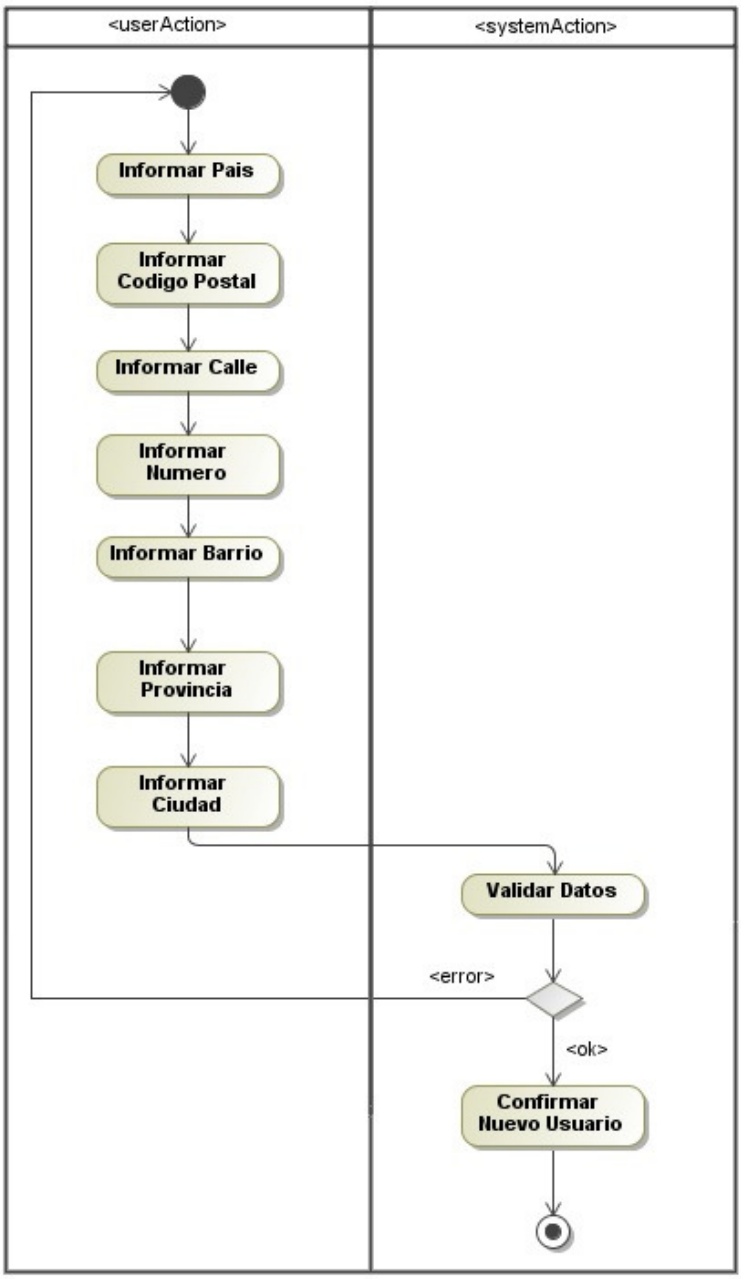


Figura 19: Vista parcial del proceso de registración de la página web de Saraiva antes del *refactoring*

Como se observa en el diagrama, el usuario tiene que informar siete campos distintos relacionados con su dirección. Sin embargo, los códigos postales de Brasil identifican detalles de la dirección del usuario y por esa razón bastaría con que se ingrese el código postal para que el sistema identifique automáticamente la dirección del usuario, evitando que el mismo tenga que ingresar manualmente todos esos datos. Con este *refactoring* se cambia la acción del usuario de entrar múltiples datos a ingresar únicamente el código postal de su domicilio, a partir del cual, automáticamente el sistema hace la verificación internamente de la dirección postal del cliente.

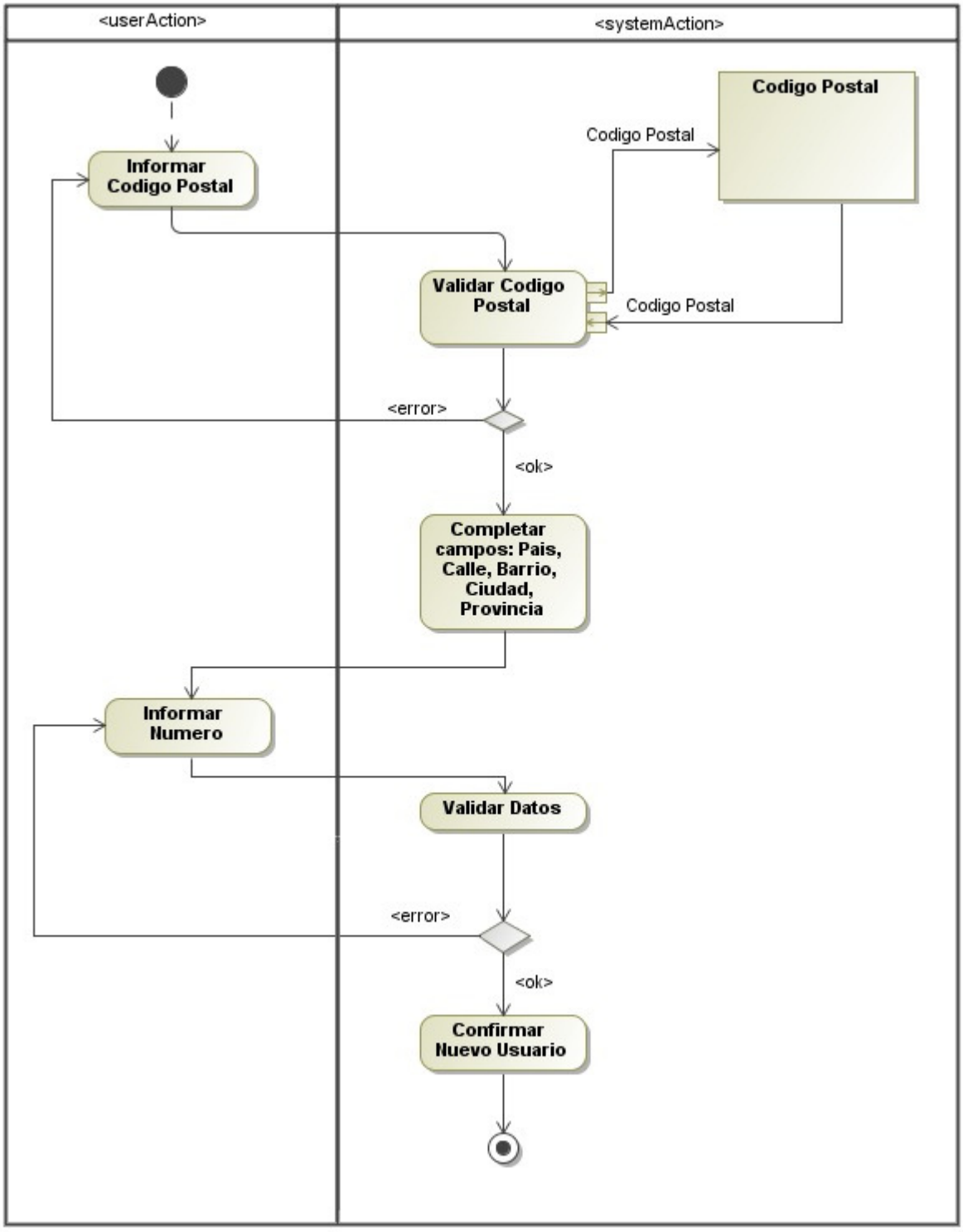


Figura 20: Vista parcial del proceso de registración de la página web de Saraiva después del *refactoring* "Reemplazar <*userAction*> por <*systemAction*>

En la Figura 20 se advierte que parte de las acciones que antes necesitaban del usuario ahora son realizadas internamente por el sistema. El sistema, a través del código postal informado al principio, procesa la mayoría de las informaciones de la dirección del usuario, completándolas automáticamente en el formulario. Con eso

el usuario solo necesita ingresar dos campos (código postal y número) en lugar de los siete que tenía que ingresar antes de la aplicación del *refactoring.*

En el sitio de comercio electrónico Submarino ya existe esta opción y el usuario solamente necesita ingresar su código postal para que la mayoría de las informaciones de su dirección sean cargadas automáticamente por la aplicación. El usuario tiene apenas que cargar informaciones personales como el número y teléfono, como se observa en la Figura 21. Este mismo mecanismo se utiliza en otras páginas de *E-Commerce* como el del supermercado Tesco (www.tesco.com).

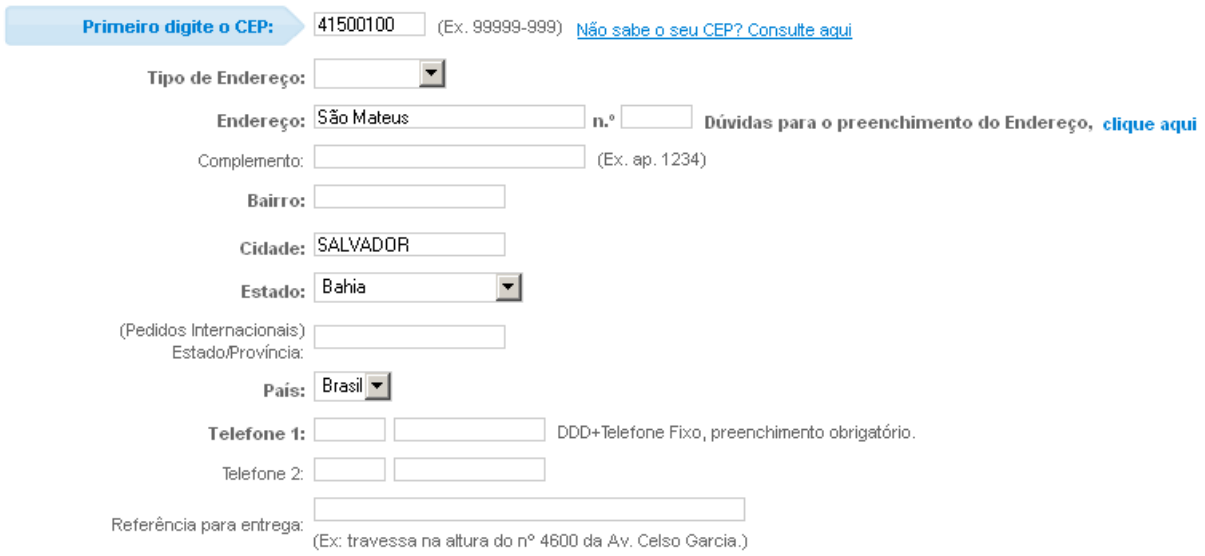


Figura 21: Ejemplo de la página de Submarino donde la dirección es cargada automáticamente por la aplicación después del usuario ingresar su código postal

Mejoras Intencionadas

- Reducción del número de acciones ejecutadas por el usuario haciendo que sea menos costoso para este ejecutar el proceso;
- Reducción del tiempo necesario para finalizar con el proceso;
- Aumento de la eficiencia de la aplicación web donde el sistema procesa las informaciones internamente sin depender del usuario;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* descripto a seguir en el capitulo 4.2.4: "Remover pasos duplicados" y también con el *refactoring* detallado en el capitulo 4.2.7: "Agregar funcionalidad de Autocompletar en formularios" que serán analizados en más detalle en las próximas secciones.

4.2.4.Remover pasos duplicados

Motivación

El propósito de un buen proceso de negocio es hacer que todo el proceso sea lo más intuitivo posible para el usuario, resultándole naturalmente fluido y rápido. Para eso es importante optimizar todas las actividades pertenecientes al proceso, manteniendo vigente las mínimas requeridas para su conclusión, evitando redundancias o repeticiones que lo alarguen innecesariamente.

Hacer que el usuario ingrese la misma información varias veces durante el proceso genera esa redundancia que debe ser evitada. Para ello el sistema debe almacenar las informaciones ingresadas por los usuarios en distintas etapas del proceso, a fin de utilizarlas en otras etapas que requieran los mismos datos, evitando de esta manera la repetición de operaciones.

La motivación principal para este *refactoring* es eliminar pasos repetidos en el flujo del proceso evitando que se solicite al usuario reiteradamente la misma información.

El *bad smell* abordado por este *refactoring* es la duplicación de pasos durante la ejecución del proceso.

Mecanismo

El *refactoring* puede ser ejecutado siguiendo los pasos descriptos abajo:

1. Identificar los pasos que están repetidos en el flujo del proceso.
2. Guardar en el sistema los datos ingresados por el usuario por primera vez.
3. Identificar el paso del proceso que necesita de los datos almacenados en el paso 2.
4. Crear *Input Pins* que transfieren directamente la información almacenada en el paso 2 hacia el paso identificado en 3.
5. Remover en el proceso los pasos duplicados identificados en el paso 1 y rehacer los *processLinks* que sean necesarios.

Ejemplo

Como ejemplo de este *refactoring* se puede mencionar el proceso de compra de la página de Saraiva (www.livrariasaraiva.com.br). En este sitio de comercio electrónico, cuando el cliente decide finalizar con la compra de un producto, la página requiere que el usuario, ya *logueado* en el sistema, complete nuevamente el login llenando los campos de e-mail y contraseña, ingresados anteriormente. Se constata por lo tanto que hay una repetición en el proceso de compra para un usuario ya *logueado* en la página.

En la Figura 22 se puede observar que el proceso de Login aparece dos veces en distintas etapas del proceso.

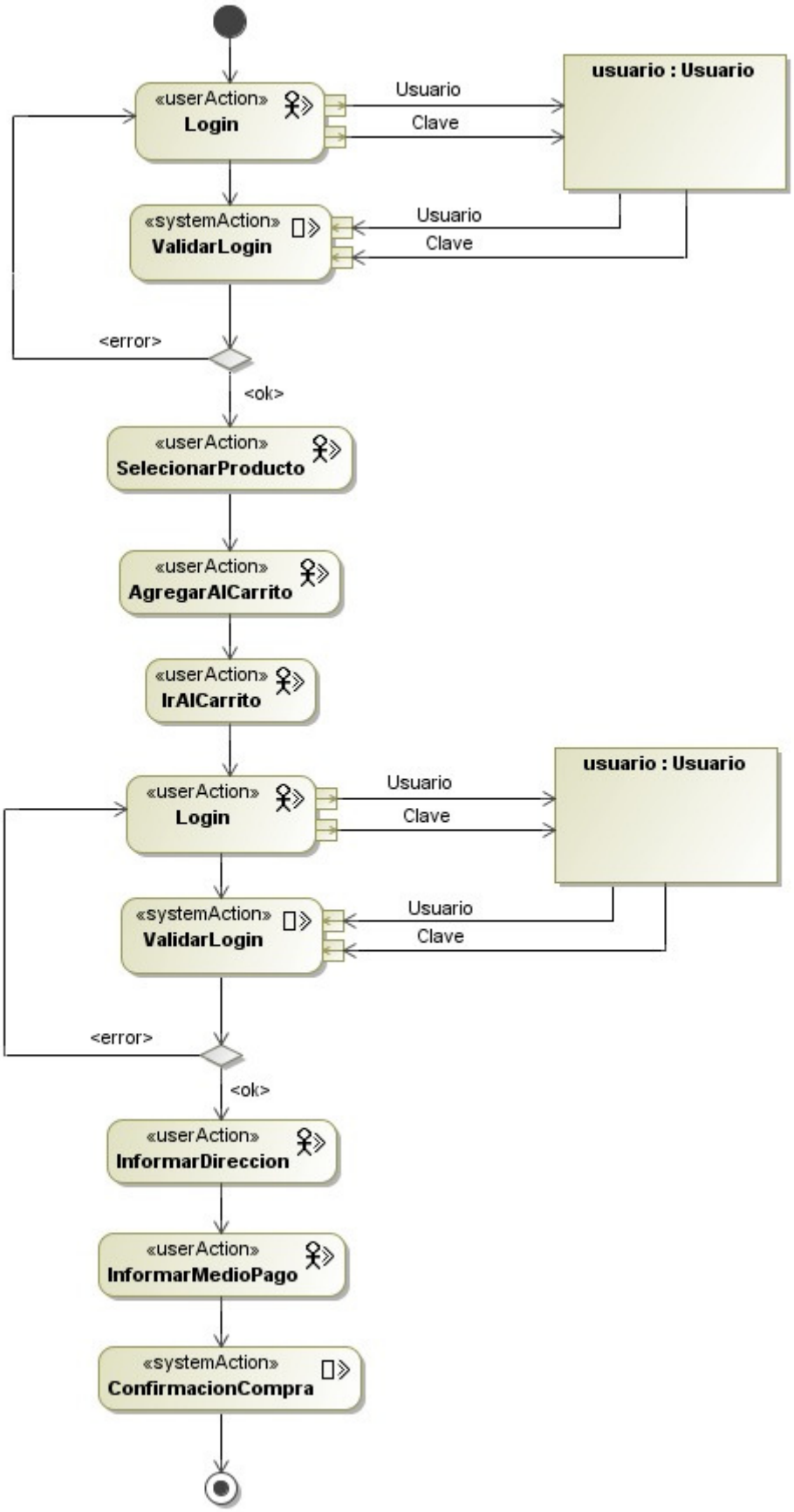


Figura 22: Proceso de compra para usuario *logueado* en la página web de Saraiva antes del *refactoring*

Para evitar la duplicación del Login, el sistema debe internamente almacenar los datos ingresados por el usuario en la primera vez que lo hace, logrando que estén disponibles cuando se los requiera en etapas posteriores del proceso. Para alcanzar este objetivo se agregaron *input Pins* que llevan la información que se guardó inicialmente en el sistema, directamente al paso que la requiere, situación que, como se dijo libera al usuario de la necesidad de cargar nuevamente los mismo datos.

El mismo comportamiento repetitivo puede ser observado en otros sitios de comercio electrónico como Amazon (www.amazon.com) y Submarino (www.submarino.com.br).

La Figura 23 muestra el diagrama de proceso de la página web de Saraiva después de la aplicación del *refactoring*. Los datos del login del usuario fueron guardados internamente por en el sistema y utilizados en una etapa posterior del mismo proceso.

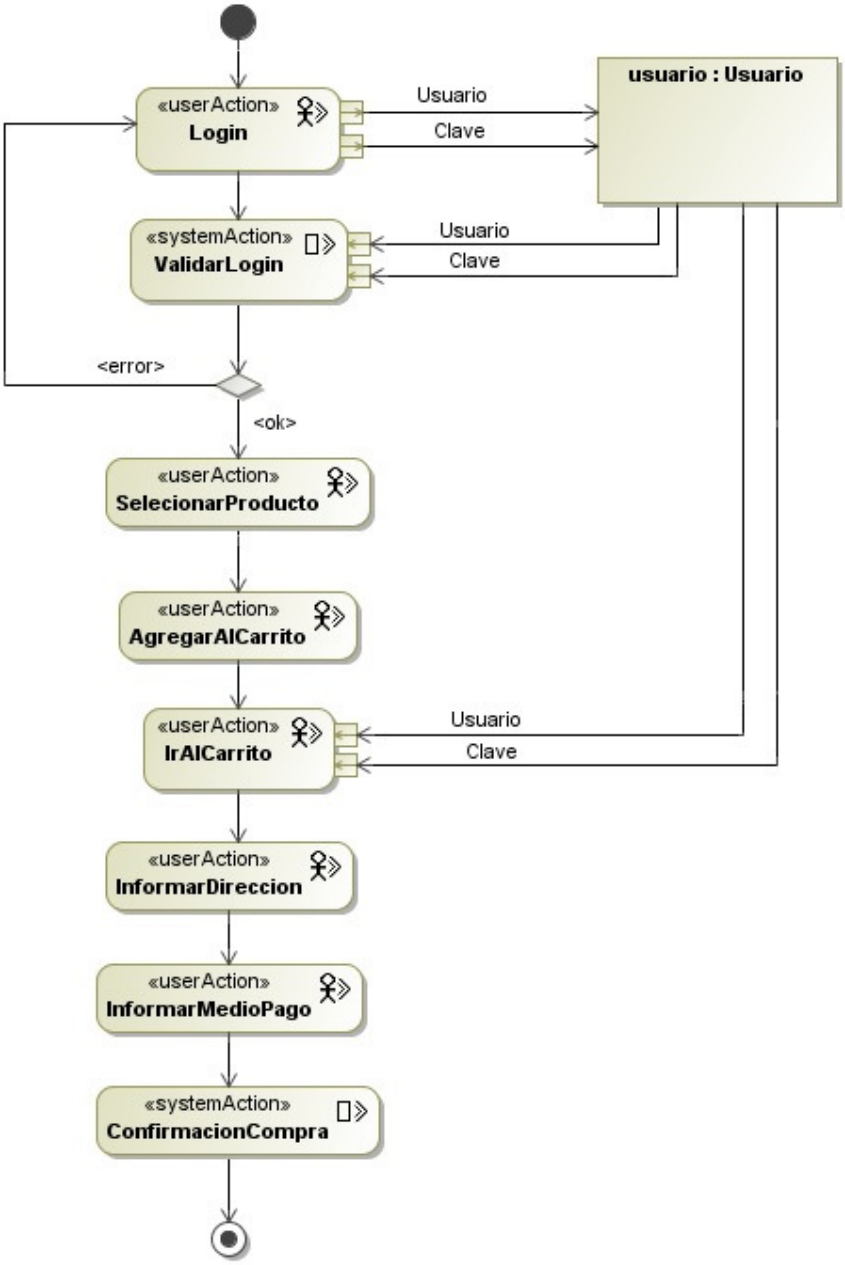


Figura 23: Proceso de compra para un usuario *logueado* en la página de Saraiva después del *refactoring* "Remover pasos duplicados"

Mejoras Intencionadas

- Reducción de tamaño del proceso, ya que pasos duplicados fueran eliminados;
- Menos <*userActions*> en el proceso, generando así menos esfuerzo para el usuario;
- Logro de un proceso más consistente y limpio, libre de redundancias;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* descripto anteriormente en el capitulo 4.2.3: "Reemplazar <*userAction*> por <*systemAction*>" ya que ambos reducen el número de <*userActions*> existentes en el proceso reemplazando por acciones realizadas internamente por el sistema.

Otro *refactoring* relacionado es el descripto sobre el diagrama de presentación en el capitulo 6.2.5: "Remover <*processLinks*> duplicados" donde también son eliminadas duplicaciones presentes en el proceso de negocio.

4.2.5. Anticipar validaciones

Motivación

Como fue mencionado previamente, en los procesos de registración de las aplicaciones web generalmente el usuario tiene que llenar largos formularios y completar diversos tipos de datos. Muchas veces la validación de los datos solo se da al final del proceso y en caso de que haya algún error en las validaciones realizadas, el usuario tiene que volver al inicio del formulario para hacer las consiguientes correcciones.

A diferencia del proceso de login detallado en el capítulo 4.2.1 "Agrupar Validaciones", el de registración supone completar formularios largos y con muchos campos que el usuario tiene que llenar. Hacer que un individuo vuelva al principio del formulario para corregir errores no es eficiente, ya que recién va a saber que entró un dato incorrecto después de haber conformado todo el formulario. Para evitar esto el sistema debe verificar los campos "*on the fly*", en otras palabras, en el momento mismo que el usuario digita los datos en los campos del formulario. Esta funcionalidad se implementa mediante el uso de la tecnología AJAX (Asynchronous JavaScript And XML).

Anticipar las validaciones en este tipo de formulario, haciendo que se hagan en paralelo y en el momento de entrada de los datos, permite que el usuario sepa anticipadamente si el dato que acaba de ingresar cumple con los requisitos del campo y si está o no correcto. Con este nuevo mecanismo, no es necesario esperar hasta el final del formulario para descubrir cuáles son los campos con problemas de validación permitiendo que los usuarios puedan corregirlos en el momento sin tener que repetir toda la secuencia.

El *bad smell* que se diluye con este *refactoring* es el re trabajo de reingresar datos en los formularios.

Mecanismo

Este *refactoring* puede ser realizado a través de los siguientes pasos:

1. Identificar en el formulario los campos que necesitan tener formato validado.
2. Por cada campo identificado en el paso 1, crear un nuevo <*systemAction*> que valide el campo en paralelo con el <*userAction*> de completar el campo. Conectar el <*systemAction*> a través de un *processLink* que se active al cambiar el foco del campo asociado.
3. Por último eliminar el <*systemAction*> que agrupaba todas las validaciones al final.

Ejemplo

El ejemplo del formulario de registración del sitio de *E-Commerce* Submarino (www.submarino.com.br) muestra que, una persona para registrarse tiene que llenar todos los campos del formulario. El sistema no acompaña el ingreso de datos con la validación correspondiente, por lo tanto el usuario no puede , sino hasta completado el proceso, verificar si lo que ingresó es correcto o no. Recién cuando al final presiona el botón “Continuar” es que el sistema realiza las validaciones y, en el caso de que algo falle, lista todos los errores en los que incurrió, teniendo el usuario que regresar al inicio del formulario para hacer las correcciones requeridas y volver a someter el formulario a una nueva validación. De persistir uno o varios errores, el mismo proceso tiene que repetirse hasta que los errores sean resueltos.

Hacer que las validaciones se centralicen en el final del proceso (como fue sugerido en el *refactoring* “Agrupar Validaciones” descripto en el capítulo 4.2.1) no es aconsejable para formularios largos y con muchos datos. Si hay errores, el usuario se verá obligado a recorrer todos los campos nuevamente revisando y corrigiendo los que sean incorrectos y luego volver a presionar en “Continuar” para que el sistema vuelva a validar los nuevos datos introducidos. Por esa razón, en los formularios largos conviene que las validaciones sean realizadas antes, en paralelo e internamente por el sistema.

La Figura 24 describe el proceso de registración en la página de Submarino, mostrando que la validaciones de los campos están centralizadas en el final del proceso, lo que hace que el usuario tenga que retroceder hasta el principio en caso en que el sistema identifique algún problema en la validación.

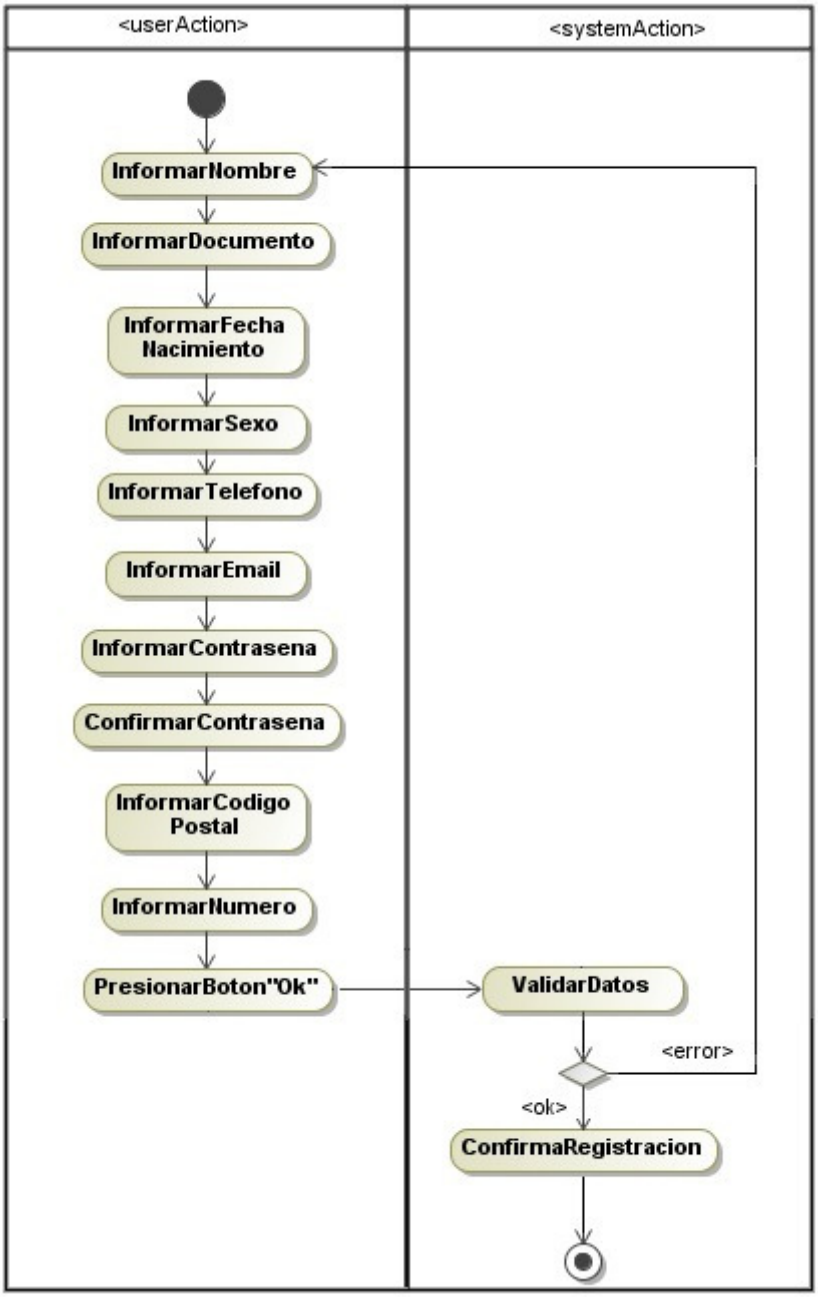


Figura 24: Proceso de registración en el sitio Submarino antes del *refactoring*

Para evitar la frustración que le supone al usuario tener que volver al principio para corregir errores, se considera oportuno alertarlo paulatinamente, en el mismo momento que ingresa los datos incorrectos, situación que le permitirá superar los inconvenientes a medida que aparecen, evitando así la demora de completar todo el formulario para darse cuenta de las anomalías; por lo tanto el sistema debe validar los datos paralelamente a su ingreso, detectando si los mismo son correctos o no.

La Figura 25 muestra el ejemplo de la página de registración de Submarino después del *refactoring*, donde las validaciones fueron anticipadas y son hechas en paralelo por el sistema.

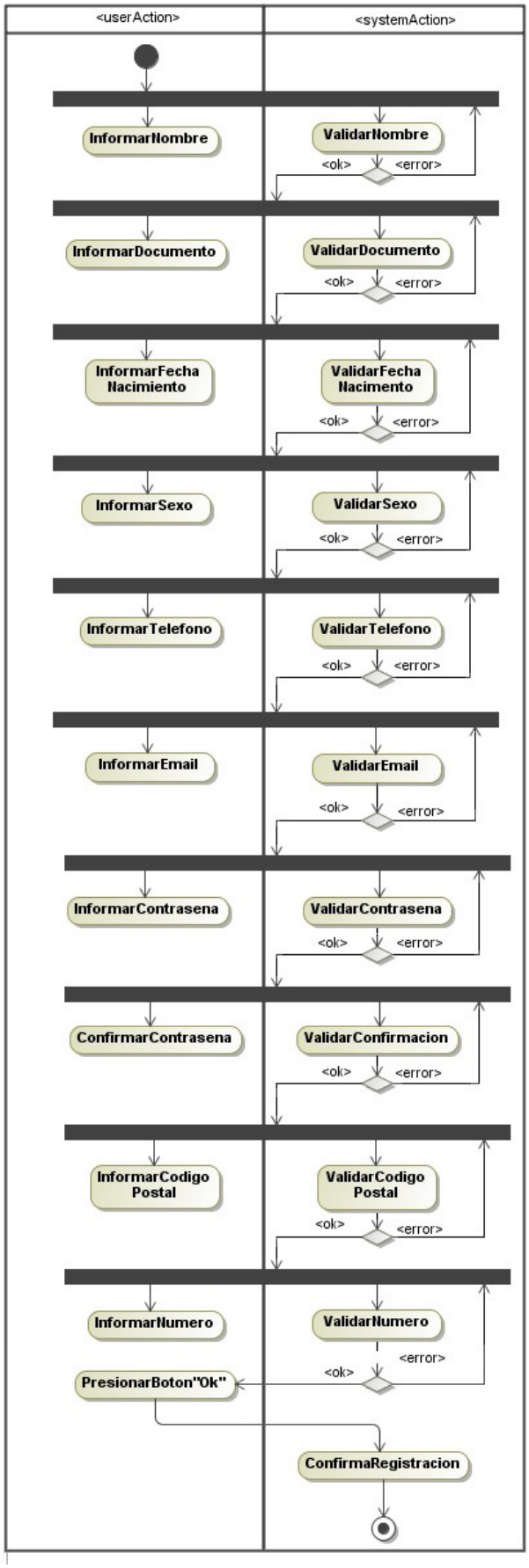


Figura 25 Proceso de registración en el sitio Submarino después del *refactoring* "Anticipar Validaciones"

Como este *refactoring* es indicado para facilitar la experiencia del usuario en los formularios largos, se sugiere también la aplicación del *refactoring* descripto en el capitulo 4.2.7: "Agregar funcionalidad de Autocompletar en formularios" que también tiene el objetivo de ayudar el usuario y de agilizar la tarea de llenar formularios largos en las aplicaciones web.

Mejoras Intencionadas

- Mayor eficiencia de la aplicación web, que será capaz de validar la información suministrada detectando los errores incurridos, en el mismo momento que se cargan los datos;
- Aumento de la satisfacción del usuario respecto de la página web, ya que puede reaccionar y superar los errores instantáneamente;
- Evitar repeticiones innecesarias que supone el completado del formulario total debido a errores en el ingreso de determinados campos;
- Aumento de la velocidad de registración en la aplicación;

Refactorings Relacionados

El *refactoring* descripto en la sección 4.2.1: "Agrupar Validaciones" tratase del opuesto de este *refactoring* ya que sugiere posponer las validaciones para el final del proceso y no anticiparlas. La diferencia entre ellos es que el primero es más útil para formularios cortos, en los que no conviene tener la validación particionada ya que se agregaría complicaciones innecesarias al proceso. Ya el *refactoring* presentado en esta sección es recomendable para formularios largos en los que el usuario tiene que llenar muchos datos, ocasión en la que es más práctico contar con validación en el momento, ahorrando tiempo del usuario.

Otro *refactoring* relacionado es el *refactoring* descripto en el capitulo 4.2.7: "Agregar funcionalidad de Autocompletar en formularios" que será detallado en las próximas secciones.

4.2.6.Agregar actividad Resumen

Motivación

En la mayoría de las páginas de *E-Commerce*, los usuarios tienen que pasar por muchos pasos antes de efectivamente confirmar el pago de la compra. Son muchos detalles igualmente relevantes y importantes, por eso es de gran utilidad tener una actividad, preferentemente anterior al paso de confirmación de la compra, que contenga un resumen con todas las informaciones referidas a la compra, como serían: la cantidad y descripción de cada producto; el precio individual de los mismos; los costos extras devenidos de envío, entrega y tasas impositivas; el costo final equivalente al precio producto más costos extras; la dirección y forma de entrega del producto; la forma de pago elegido; la fecha de entrega estimativa, etc.

Agregar una actividad de Resumen introduce confiabilidad a la página web e inspira seguridad a los usuarios, ya que al conferir todas las informaciones necesarias previne errores y transmite certidumbre antes de proceder con la compra.

El *bad smell* que se ataca con este *refactoring* es la posibilidad de errores en el proceso de compra.

Mecanismo

Para aplicar este *refactoring*, es necesario llevar adelante los siguientes pasos:

1. Identificar el paso final del proceso donde el usuario tiene que confirmar la compra.
2. Agregar un nuevo <*systemAction*> antes de la actividad identificada en el paso 1 llamado "Resumen" en el que se exhibirán todos los detalles de la compra a ser efectuada.
3. Alterar tantos <*processLinks*> como sean necesarios para conectar el nuevo <*systemAction*> creado en el paso 2 (dos) con las otras actividades.

Ejemplo

La página web del supermercado Extra (www.extra.com.br) se utiliza como ejemplo para la aplicación de este tipo de *refactoring* puesto que, en el proceso de compra no tiene una actividad que exhibe todos los detalles de la misma. El proceso de *checkout,* como se ve en la Figura 26, cuenta con tres pasos: Login, Dirección y Forma de Pago/Confirmación de Compra. En el último paso, antes que el usuario seleccione la forma de pago, se muestra apenas el valor del producto y del envío con el subtotal, pero no informa al usuario sobre detalles importantes de la compra como las cantidades adquiridas o el plazo de entrega de los productos comprados, por ejemplo.

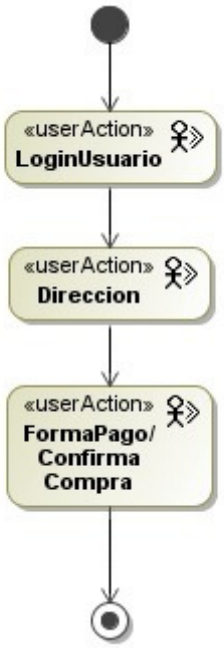


Figura 26: Proceso de *checkout* del sitio del supermercado Extra antes del *refactoring*

El *refactoring* propuesto sugiere agregar al proceso de compra una nueva actividad llamada "Resumen", mediante la cual el usuario puede tomar conocimiento de los detalles de la operación que está cerrando, antes de seleccionar la forma de pago y de confirmar la compra. Esta nueva actividad debe contener detalles como la cantidad de los productos, precios individuales, costo de envío, costo total de la compra, la dirección elegida para la entrega, el tipo de encomienda, entre otras. Todas esas informaciones son de utilidad para que el usuario quien sabrá con exactitud lo que está comprando y podrá controlar que está todo bien, antes de confirmar la operación.

La Figura 27 muestra el proceso de *checkout* del sitio de Extra con la aplicación del *refactoring* que agrega un <*systemAction*> con el que se muestra todos los detalles relevantes a la compra que se está realizando en el momento. Este nuevo <*systemAction*> se incluyó antes del último paso del proceso, donde el usuario tiene que informar la forma de pago y confirmar la compra. De esta manera el usuario hace una compra consciente y puede confirmar sin temor de estar cometiendo algún error.

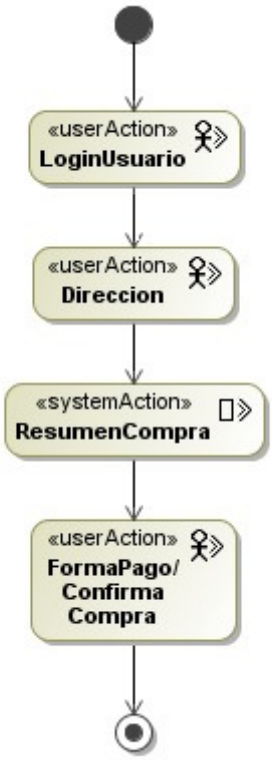


Figura 27: Proceso de *checkout* del sitio del supermercado Extra después del *refactoring* "Agregar actividad resumen"

En el *web site* del también supermercado Tesco (www.tesco.co.uk), antes de proceder con el pago, la aplicación exhibe una página donde el usuario puede acceder al carrito de compras y ver todos los productos comprados con cantidades y precios, verificar los detalles de la entrega, y instrucciones relacionadas con la compra. Eso permite que el usuario pueda visualizar en una página separada todos los detalles de lo que está comprando, antes de pagar.

Mejoras Intencionadas

- Reducción del riesgo de errores en el proceso de compra;

- Aumento de la confianza de los usuarios en la página web;
- Aumento de la satisfacción del usuario que puede conferir los datos ingresados en etapas anteriores del proceso;
- Aumento de las informaciones referentes al proceso;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* descripto en el capitulo 6.2.4: "Explicitar pasos del proceso" descripto en el modelo de presentación.

4.2.7.Agregar funcionalidad de Autocompletar en formularios

Motivación

La tarea de completar formularios en una aplicación web es una actividad recurrente que el usuario tiene que repetir con frecuencia. Todas las veces que hace una búsqueda o que utiliza un filtro para encontrar determinada información, el usuario tiene que llenar los campos disponibles en la página, especialmente en las páginas de comercio electrónico como compañías aéreas o de empresas que ofrecen servicios online.

Si los campos del formulario quedan abiertos al usuario, este puede ingresar cualquier tipo de información y muchas veces, por no saber bien que poner o por error de digitación, escribe mal los datos. Los datos ingresados incorrectamente provocan un error en la aplicación web informando que no se puedo encontrar lo que se especificó en el filtro. Este tipo de error es muy común y por esa razón toda asistencia que el usuario reciba de la aplicación web con el objetivo de facilitar la conclusión del formulario, evita que este tipo de problema suceda.

La funcionalidad de autocompletar es una ayuda ofrecida por muchas aplicaciones web para restringir y guiar al usuario en el momento que está digitando datos en algún campo de un formulario. Cuando el mismo empieza a digitar, el sistema busca internamente cuales son los valores posibles de acuerdo con lo que está digitado y los muestra en el mismo momento, a través de una lista. El usuario puede elegir en esa lista la opción deseada. La disponibilidad de la lista cargada por el sistema facilita la visualización de las opciones disponibles en la aplicación web y guía el usuario a medida que va digitando, indicando que opciones tiene según el filtro informado por esa persona. Otra ventaja es que esa funcionalidad previne posibles errores de digitación y reduce por lo tanto los errores de sistema o búsquedas vacías debido a filtros incorrectos.

El *bad smell* que se trata de reducir con este *refactoring* es la repetición de los formularios con el objetivo de corregir errores en el filtro y la frustración del usuario en tener resultados de búsqueda vacios.

Mecanismo

Para aplicar este *refactoring* es necesario seguir los siguientes pasos:

1. Identificar los campos del formulario que pueden tener la funcionalidad de autocompletar agregada;
2. Agregar un <*systemAction*> paralelo a cada campo identificado en el paso 1 (uno) para autocompletar lo que fue digitado en el campo.

Ejemplo

Hacer que el sistema sugiera posibles opciones de acuerdo con lo que está siendo digitado por el usuario trae practicidad al proceso y evita problemas de digitación. La mayoría de las páginas de comercio electrónico tienen esa funcionalidad implementada, así como muchos de los sitios de venta de pasaje online. Es una funcionalidad bastante utilizada por ofrecer una forma simple y practica de guiar el usuario en su proceso de búsqueda por algún producto.

El sitio de alquiler de autos Enterprise (www.enterprise.com) no cuenta con el sistema de autocompletar y caso el usuario digite el nombre de la localidad con errores, la aplicación indica, sin más, que la localidad informada no pudo ser encontrada, exigiendo al usuario que vuelva a ingresar los datos en el formulario y intentar nuevamente.

La Figura 28 muestra el diagrama de actividad del proceso de búsqueda de un auto en la página de Enterprise. En caso de que el usuario informe una localidad no válida o bien una localidad que no forme parte de la lista de ciudades en las que trabaja la empresa (situación que en principio el usuario desconoce), el sistema retorna un error, volviendo al formulario y solicitando que el usuario modifique el filtro de búsqueda.

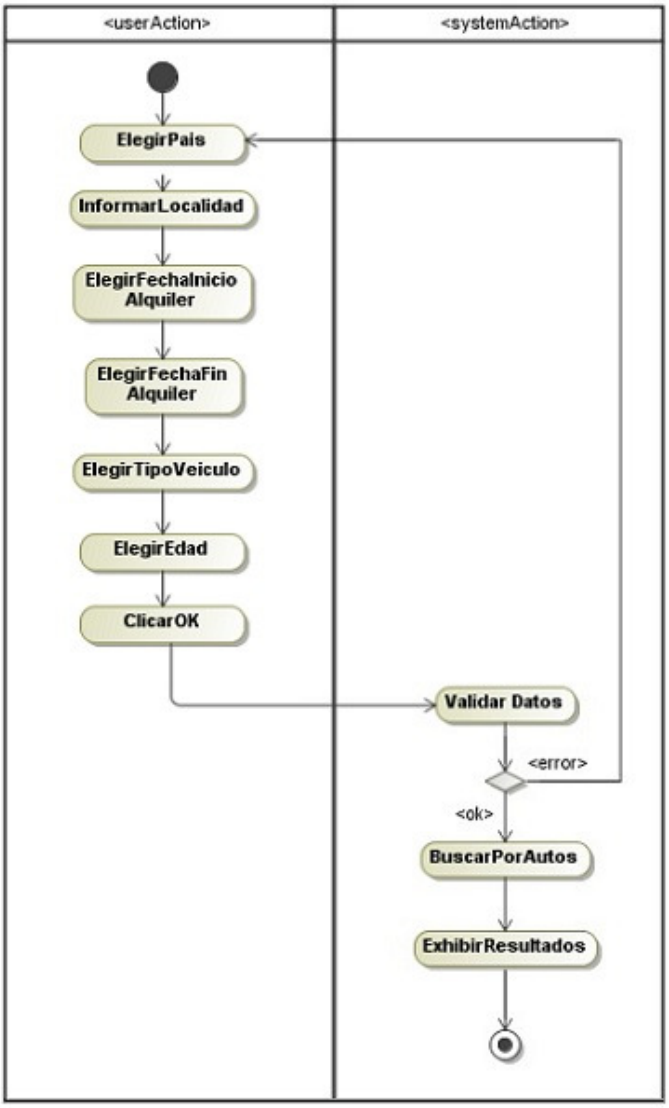


Figura 28: Página de alquiler de autos de la empresa Enterprise antes del *refactoring*.

Con la implementación de la funcionalidad de autocompletar, el usuario puede saber si la localidad que está buscando está en la lista de localidades disponibles y en el caso que no aparezca en ella, puede corregir y digitar otra localidad antes de proseguir con el formulario. La Figura 29 muestra el diagrama de proceso después de la aplicación de este *refactoring*. La funcionalidad de autocompletar se activa cuando el usuario está digitando la localidad. Como este campo es el único que está abierto para el ingreso libre de información en la página analizada, será el único campo que tiene la funcionalidad habilitada.

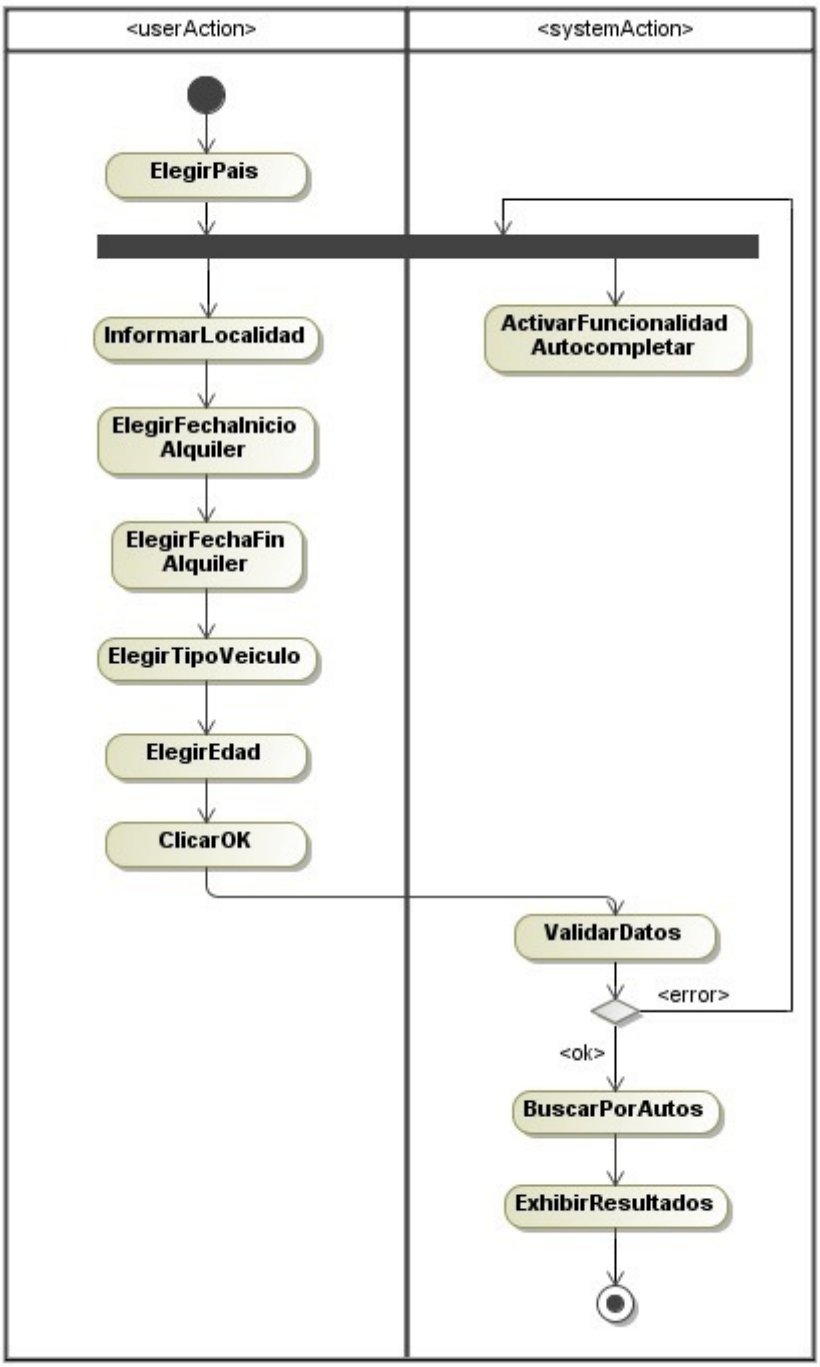


Figura 29: Página de alquiler de autos de la empresa Enterprise después del *refactoring* "Agregar funcionalidad de Autocompletar en formularios"

Mejoras Intencionadas

- Instruir el usuario informando cuales son las opciones disponibles en el campo de acuerdo con lo que este digita;
- Acelerar el proceso de completar un formulario ya que si el usuario digita una letra, puede seleccionar la opción correcta en la lista, sin tener que terminar de digitar la palabra;

- Reducción de las posibilidades de errores en los formularios evitando resultados vacíos;
- Aumentar la eficiencia de la página y la satisfacción del usuario mediante la practicidad de los formularios;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* descripto en el capitulo 4.2.3: "Reemplazar <*userAction*> por <*systemAction*>". Mismo que el *refactoring* descripto en esta sección no reemplace totalmente la acción del usuario de completar el filtro como fue realizado en el *refactoring* detallado en 4.2.3, fue agregada una acción de sistema (<*systemAction*>) con el objetivo de auxiliar el usuario a completar el campo, consecuentemente ayudándolo a concluir el formulario con más agilidad.

Otro *refactoring* también relacionado es el *refactoring* "Anticipar validaciones" descripto en el capítulo 4.2.5 de esta sección ya que ambos, en distintas maneras, validan el campo que está siendo informado por el usuario en el momento en que este está informándolo.

4.2.8.Agregar posibilidad de Cancelar un proceso

Motivación

Una vez iniciado un proceso en una aplicación web, el usuario tiene que seguir el itinerario de pasos previstos hasta llegar al final y concluirlo. Muchas veces los sitios web no incluyen la alternativa de cancelar o interrumpir una aplicación, el caso de que el usuario lo desee. Esa falta de practicidad del sistema se trasunta como rigidez en el proceso de negocio, al acotar al cliente, negándole la flexibilidad de sus decisiones frente al proceso iniciado.

En el caso de no tener mecanismos para cancelar un proceso antes de completarlo, el usuario que desee interrumpirlo tiene que cerrar la página en que está abierto el proceso, o usar los botones de navegación disponibles en el *browser*, o bien digitar una nueva URL para dirigirse a una página distinta. Todas estas opciones no son deseables pues pueden generar un estado de inconsistencia en el proceso, aparte de ser frustrantes para el usuario.

Si la lógica de la aplicación no ofrece un método que posibilite cancelar el proceso, el usuario se beneficia de los elementos que son inherentes a la aplicación como las operaciones disponibles en el *browser*, como los botones de Volver y Avanzar. Como afirmó Baresi et al. en [6], hacer uso de esos mecanismos generan problemas de inconsistencia en la aplicación y pueden resultar confuso para el usuario ya que el estado de la aplicación muchas veces no coincide con la realidad.

Agregar la posibilidad de cancelar un proceso antes de su conclusión muestra que la aplicación web es flexible y que considera al usuario, al incorporar la posibilidad de este no querer (o no poder) concluir el proceso en el momento o de poder volver atrás sobre una decisión (como por ejemplo, clicar sin querer en link que dispara el proceso). Al agregar la posibilidad de cancelar el proceso, el usuario sería removido de los

nodos del proceso y re direccionado a la *Homepage* de la página en cuestión volviendo a los nodos de navegación, donde puede elegir que hacer o como seguir.

El *bad smell* tratado por este *refactoring* es la falta de flexibilidad de los procesos de negocio y el estado de inconsistencia generado por el hecho de que el usuario navegue por nodos afuera del proceso.

Mecanismo

Los siguientes pasos deben ser ejecutados para aplicar ese *refactoring*:

1. Identificar los pasos que componen el proceso de negocio;

2. Agregar un punto de decisión (representado en UML por el elemento ) entre cada paso del proceso agregando la opción de cancelar.

Ejemplo

El ejemplo usado para este *refactoring* describe el proceso de *checkout* del sitio de Amazon (www.amazon.com). Cuando el usuario inicia el proceso de compra en la página, solo están disponibles botones para proseguir y avanzar con el proceso, no existiendo ninguna opción disponible para que el mismo pueda ser cancelado. Si el usuario desea salir del proceso que inició tiene que salir de la página (cerrando la ventana o digitando una nueva URL en el *browser*) o utilizar los botones de volver en el *browser* para retornar la navegación hacia las páginas anteriores al inicio del proceso. Estos mecanismos son paliativos encontradas por los usuarios debido a falta de opción ofrecidas por la aplicación.

Ofrecer posibilidad de corregir errores o de volver atrás en una acción realizada aumenta la confiabilidad y la satisfacción que el usuario tendrá respecto a la aplicación web pues significa que la página asume que el usuario puede equivocarse, puede querer cambiar de idea y por consiguiente desistir de ejecutar el proceso, no dejándole "preso" en este.

La Figura 30 muestra el proceso de *checkout* del sitio de Amazon, donde se puede ver que la única manera de salir del proceso es con su conclusión, en la etapa final de confirmación. En caso de que el usuario no pueda completar el proceso, tiene que recurrir a las alternativas mencionadas anteriormente pues la página no incluye otra posibilidad de salida.

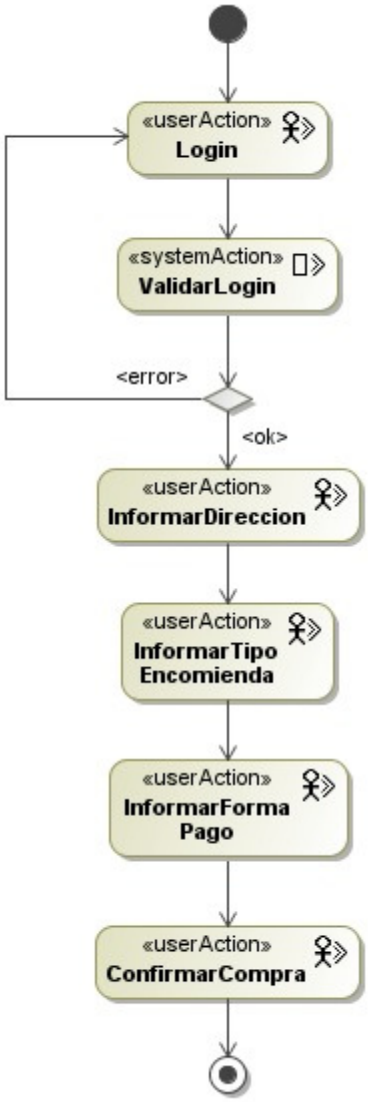


Figura 30: Proceso de *checkout* del sitio de Amazon antes del *refactoring*

Con la implementación de este *refactoring* se agrega en cada paso del proceso un punto de decisión, la posibilidad de cancelar o de proseguir con el mismo. Si el usuario decide proseguir, el flujo de actividades se ejecutará linealmente hasta su conclusión. Por el contrario, si desea abandonarlo o interrumpirlo antes del final, tiene la posibilidad de cancelar y salir del proceso, sin tener que salir de la página o cambiar la URL del *browser*.

La Figura 31 muestra el proceso de *checkout* con la opción de "Cancelar" implementada entre cada uno de los pasos del proceso; si esta es la opción elegida, el proceso se interrumpirá automáticamente y el usuario será re direccionado a los nodos de navegación de la aplicación.

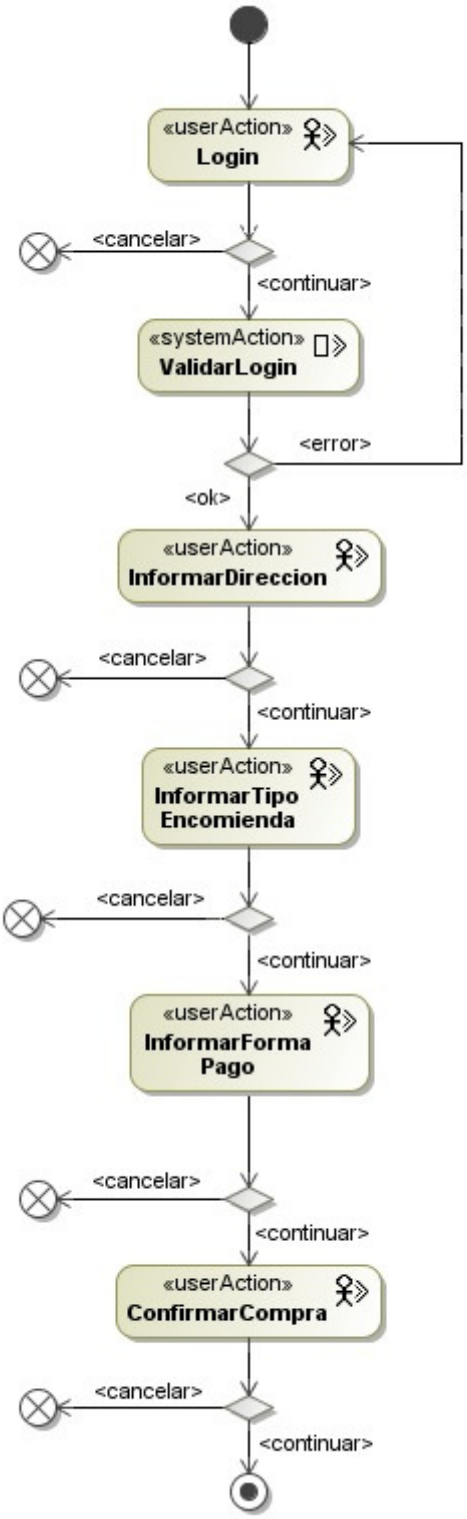


Figura 31: Proceso de *checkout* del sitio Amazon con la opción de Cancelar el proceso, después del *refactoring* "Agregar posibilidad de cancelar un proceso"

Este *refactoring* también afecta el diagrama de navegación pues al incluir la posibilidad de cancelar un proceso durante su ejecución, necesariamente altera la navegación en el mismo.

Mejoras Intencionadas

- Ampliar la experiencia del usuario al flexibilizar la ejecución del proceso ofrecido;
- Aumentar la confianza del usuario respecto de la página, quien podrá decidir libremente si continúa o no con el proceso iniciado;
- Evitar las inconsistencias devenidas por el uso de paliativos con el fin de salir del proceso antes que el mismo se concluya;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* del modelo de navegación descripto en el capitulo 5.2.2: "Agregar posibilidad de retroceder en el proceso".

5. *REFACTORINGS* SOBRE EL MODELO DE NAVIGACION

5.1. Definición

El modelo de navegación es uno de los modelos más importantes de las aplicaciones web pues si no se lo define correctamente puede lograr que, aún pequeños, sitios web tengan una navegación complexa y complicada, afectando directamente su calidad y la usabilidad [43]. El modelo de navegación describe los nodos y links de una aplicación web definiendo como el contenido de la página es accesible al usuario. Este modelo representa como la aplicación web está estructurada, cuales son los nodos que la componen, cómo se puede acceder a ellos y cómo están conectados entre sí.

Cabot y Gomez [43], propusieron un catálogo de *refactorings* aplicados en el modelo de navegación de una aplicación web. Dichos *refactoring* se focalizan en los links, páginas y caminos de navegación de las páginas de la aplicación utilizando la metodología WebML [39], no envolviendo en tal mecanismo los procesos de negocio.

Por su parte, los *refactorings* propuestos para el modelo de navegación desde esta tesis, focalizan en mejorar la navegación y el acceso a los *processsClass (*clases usadas para integrar los procesos de negocio en el diagrama de navegación*)* de la aplicación web focalizando así en los procesos de negocio embebidos en las páginas.

Los *refactorings* en el modelo de navegación deben preservar:

- El conjunto de operaciones accesibles desde cada nodo.
- El acceso de cada operación a través de los nodos desde la *Homepage.*

Con base en las premisas anteriores, los *refactorings* que se aplican en el modelo de navegación pueden:

1. Agregar links hacia un *processClass.*
2. Cambiar el camino de navegación hacia un *processClass.*
3. Agregar una entrada en el menú para acceder al *processClass.*
4. Agregar links para mejorar la navegación en el *processClass.*

5.2. Catálogo

5.2.1. Crear acceso directo

Motivación

Al acceder a una aplicación web el usuario está enfocado en completar una tarea o un objetivo específico, y cuanto más fácil sea para él encontrar lo que busca, mejor. Con la gran cantidad de información, imágenes y

links que usualmente están disponibles en las páginas web, es de gran valor que la información o los procesos más utilizados resulten clara y fácilmente accesibles, permitiendo su pronta identificación y utilización, evitando la navegación innecesaria por muchas páginas antes de localizarlos.

La intención de este *refactoring* es reducir, para determinados procesos, el tiempo de navegación de los usuarios en la aplicación mediante a la generación de acceso directo que permita completar la tarea deseada de manera más directa y rápida. En un proceso popular y de gran uso por parte de los usuarios, la creación de un acceso directo permite que el mismo esté más accesible permitiendo que el usuario pueda acceder a la información deseada con más rapidez y en menos pasos. Acortar los pasos necesarios para acceder un proceso de uso, muchas veces diario, otorga a la página una navegación más fluida y más objetiva. Eso también evita que el usuario recorra obligadamente a un camino innecesario siempre que quiera disparar el proceso, permitiéndole que pueda concretar con mayor celeridad su objetivo.

Krug afirma que la información deseada no debe estar a muchos cliques del usuario ya que, cuando este tiene que navegar mucho para acceder a determinada información se genera en él un nivel de frustración y cansancio propio de cuando a uno no le resulta práctico completar el objetivo deseado en la aplicación web [10].

El "*bad smell*" para este *refactoring* es el exceso de pasos necesarios para acceder a un proceso de mucho uso en una aplicación web.

Mecanismo

Es necesario identificar, en el diagrama de navegación, el proceso para el cual se quiere crear el acceso directo, para ello se ejecutaran los siguientes pasos:

1. Identificar desde la *Homepage* el camino de nodos y links que permiten acceder al proceso para al cual se quiere crear el acceso directo.
2. Agregar un <*processLink*>, es decir una relación de asociación, desde la *Homepage* creando un acceso directo al proceso deseado.
3. Determinar si los nodos y links que existían previamente hacia el <*processClass*> siguen siendo necesarios. De no ser así, eliminar esos nodos y links.

Ejemplo

El ejemplo seleccionado se aplica a la página de *Home Banking* del Banco Nación Argentina (BNA) donde el usuario tiene que hacer clic en 2 links que están ubicados en 2 páginas distintas para finalmente poder disparar el proceso de Login en el que ingresará su nombre de usuario y contraseña, datos necesarios para acceder a la información de sus cuentas (una de las tareas más utilizadas en los sitios de *Home Banking* actualmente), como se puede ver en la Figura 32.

En la página actual del banco el usuario tiene que navegar por la *Homepage* buscando el link que lo llevará a una página de *Home Banking*, en esta hay otro link que lleva al inicio del proceso de Login de Cuentas, donde el usuario debe ingresar su nombre de usuario y contraseña y al presionar el botón "Ingresar", el usuario finalmente puede acceder a la información de sus cuentas.

Todas las veces que el usuario desea ingresar para chequear sus cuentas, tiene que recorrer el mismo y largo camino, una y otra vez. Navegar por diversas páginas para alcanzar a un proceso que es muy utilizado puede ser frustrante y poco intuitivo, esto sin mencionar la falta de practicidad que supone repetir lo mismo varias veces. Por esa razón el acceso directo, como el proprio nombre insinúa, acerca el proceso al usuario, haciéndolo más accesible.

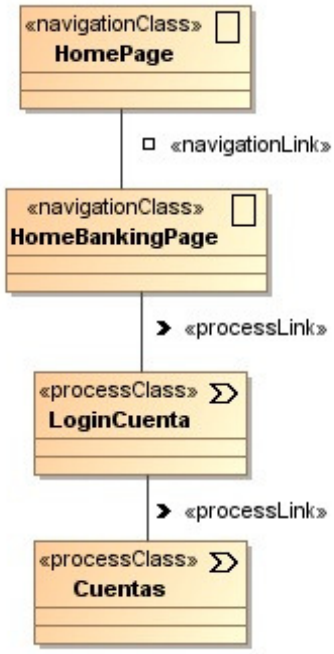


Figura 32 : Vista parcial del diagrama de navegación de la página de un *Home Banking* antes del *refactoring*

El *refactoring* propuesto crea un acceso directo desde la *Homepage* del BNA permitiendo que el usuario pueda ingresar al proceso de Login ingresando directamente su nombre de usuario y clave en el página principal y pueda acceder a sus cuentas con apenas un clic.

Con la creación de este acceso directo, la navegación se convierte en más rápida y objetiva al reducir el número de links que el usuario tiene que recorrer para acceder a un proceso que repite con frecuencia. Asimismo, se evita que el usuario tenga que navegar por varias páginas para disparar una de las funcionalidades más utilizadas en las páginas de bancos y hace que el proceso quede a un clic de distancia del usuario.

La Figura 33 muestra el <*processLink*> creado desde la *Homepage* y la reducción de nodos y links para acceder al proceso de Login del usuario. Con la aplicación del *refactoring*, el usuario tiene disponible en la *Homepage* los campos de usuario y contraseña y puede ingresar a la página de sus cuentas mucho más rápido.

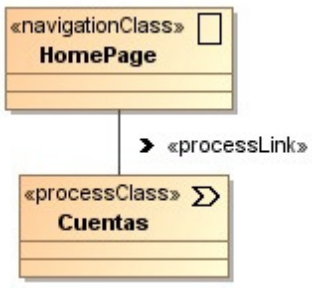


Figura 33: Vista parcial del diagrama de navegación después del r*efactoring* "Crear acceso directo"

Mejoras Intencionadas

- Reducción del número de nodos necesarios para acceder al proceso;
- Mejoramiento del acceso al proceso;
- Reducción del tiempo necesario para acceder al proceso;

Refactoring Relacionados

Este *refactoring* afecta también el modelo de presentación y por eso está directamente relacionado con el *Refactoring* "Agregar Acceso Directo a la *Homepage*" descripto en el capítulo 6.2.2 en el diagrama de presentación.

5.2.2.Agregar posibilidad de retroceder en el proceso

Motivación

Muchos procesos de negocio en las aplicaciones web tienden a ser largos, con muchos pasos y requieren que el usuario ingrese mucha información. La gran cantidad de pasos y las distintas informaciones solicitadas pueden confundir el usuario a respecto de los datos ingresados en pasos anteriores al que navega, por tanto resulta útil que se pueda retroceder en el proceso a fin de confirmar o de cambiar la información ingresada previamente.

Agregar la posibilidad de retroceder en el proceso proporciona al usuario la tranquilidad de que puede retornar, en el caso de que se haya equivocado en algo o en caso quiera hacer alguna modificación de lo que hizo con anterioridad. Teniendo la opción de volver, el usuario puede simplemente corregir o alterar lo deseado, en lugar de tener que empezar con el proceso desde el principio y repetir todos los pasos nuevamente. Esa posibilidad agrega flexibilidad al proceso y aumenta la confianza del usuario en la página ya que disminuye la posibilidad de errores.

No tener una posibilidad de retroceder disponible en la página puede hacer que el usuario utilice otros mecanismos como los botones de navegación del *browser*. Hacer uso de esos botones, como ya se dijo,

puede traer inconsistencia en el estado del proceso ya que retrocede los nodos de navegación y no el estado interno del proceso.

En este *refactoring* el *bad smell* que se ataca es la falta de flexibilidad del proceso de negocio y muchas repeticiones del mismo, en caso datos sean ingresados erróneamente.

Mecanismo

Este *refactoring* se ejecuta por medio de los siguientes pasos:

1. Identificar el nodo de inicio y el nodo final del proceso.
2. Con excepción del nodo de inicio, crear un *outgoing* link en los nodos definiendo como *target* el nodo anterior del proceso.
3. Repetir paso 2 (dos) hasta el nodo final.

Ejemplo

En el proceso de compra de un pasaje aéreo en el sitio de GOL (www.voegol.com.br), una vez que fue iniciado el proceso de compra de un pasaje luego que el usuario elige el vuelo deseado, no es posible retroceder en el proceso para alterar o confirmar los datos ingresados anteriormente, por esa misma razón, cada vez que el usuario desee cambiar la fecha de los vuelos o cambiar alguno de los datos personales del pasajero, debe volver a la *Homepage* de GOL para reiniciar todo el proceso de nuevo.

La Figura 34 muestra el ejemplo de la página de GOL donde sólo existen links para avanzar en el proceso, no ofreciendo la posibilidad de volver a los pasos ejecutados anteriormente.

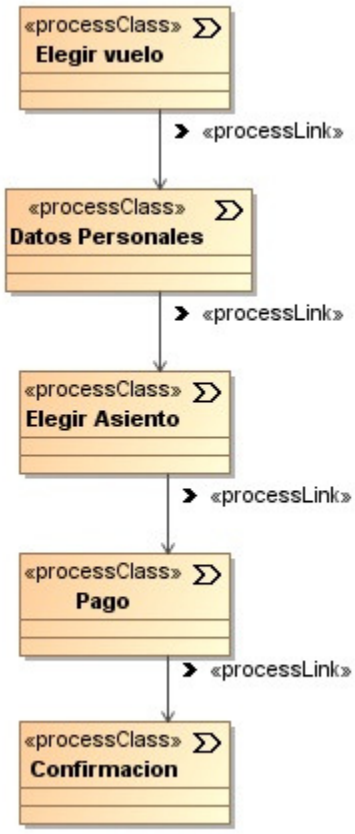


Figura 34: Proceso de compra de pasajes aéreas en el sitio de GOL antes del *refactoring*

La búsqueda de vuelos es un proceso que suele repetirse con frecuencia por la mayoría de los usuarios quienes tratan de buscar el mejor precio probando con distintas fechas o diferentes destinos. Si el proceso no permite una forma práctica y rápida de cambiar los datos, el usuario se ve obligado a repetir todas las veces, desde el principio, el mismo proceso hasta lograr su objetivo.

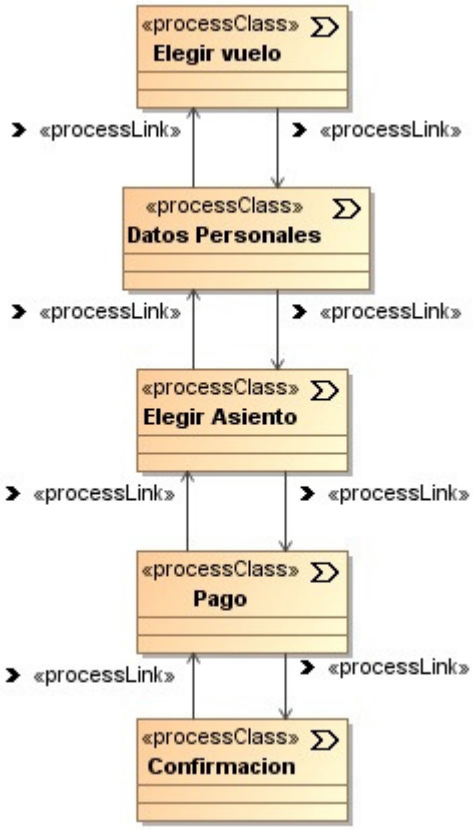


Figura 35: Sitio de la empresa aérea GOL después del *refactoring* "Agregar la posibilidad de retroceder en el proceso"

En la Figura 35 podemos ver el modelo de navegación de la página de la compañía aérea GOL modificado para ilustrar este *refactoring*. Con este propósito se agregaron *outgoing* links en todas las etapas, con excepción de la página del inicio del proceso, definiendo como target el nodo anterior del proceso, posibilitando de esta manera que el usuario pueda retornar a los pasos ya ejecutados para editar o chequear los datos ingresados antes de confirmar el proceso de compra. Con la aplicación de este *refactoring* ya no es más necesario repetir todo el proceso en caso de que el usuario desee cambiar algún dato, pues con los links agregados se le permite navegar por todos los pasos, cambiar o chequear lo que haga falta y continuar con la operación desde el paso en el que se encuentra hasta el final del proceso. Al retroceder en el proceso, se retrocede también en su estado interno, lo que evita la ocurrencia de inconsistencias, al contrario de la utilización de los botones de navegación de *browser*, que retroceden en las páginas pero no en el estado del proceso, ocasionando problemas muchas veces desconocidos por los usuarios [28].

El sitio de la compañía aérea Pluna (www.flypluna.com), como se muestra en la Figura 36, es un buen ejemplo de cómo se puede implementar este *refactoring*: todos los pasos del proceso de compra de un pasaje aéreo (excepto el primero) poseen un botón “Volver” que tiene como *target* al nodo anterior, permitiendo al usuario retroceder, alterar o confirmar los datos ingresados.

Figura 36: Página de la compañía aérea Pluna donde se muestra los botones “Volver” que permite al usuario retroceder en el proceso.

Para aumentar la flexibilidad del proceso, se sugiere también aplicar el *refactoring* descripto en el capitulo 4.2.8: “Agregar posibilidad de Cancelar un proceso", de esta manera el usuario no solo tiene la posibilidad de retroceder en los pasos del proceso, pero cancelarlo caso no pueda proseguir.

Mejoras Intencionadas

- Evitar la repetición del proceso en el caso de error o caso el usuario desee hacer algún cambio;
- Aumentar la confianza del usuario en la aplicación ya que ofrece la posibilidad de volver y corregir posibles errores;
- Aumentar la flexibilidad del proceso, permitiendo al usuario navegar por los pasos que ya fueron ejecutados;
- Disminuir la posibilidad de ingresar datos incorrectos en el proceso;
- Evita el uso de los botones de navegación del *browser* evitando el estado de inconsistencia del proceso al utilizar este mecanismo;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* de modelo de proceso definido en el capitulo 4.2.8: "Agregar posibilidad de Cancelar un proceso". Ambos sugieren cambios para mejorar la flexibilidad del proceso, ofreciendo opciones básicas que facilitan su ejecución por los usuarios.

Otro *refactoring* relacionado es el *refactoring* descripto en el modelo de presentación en el capitulo 6.2.4: "Explicitar pasos del proceso". Con estos dos *refactorings* el usuario puede visualizar los pasos existentes del proceso permitiendo navegar por el mismo de forma más consciente.

5.2.3.Agregar la posibilidad de extender filtros del proceso

Motivación

Es habitual que el precio de un pasaje varíe de acuerdo con el día elegido, por eso es común que los usuarios repitan el mismo filtro cambiando solamente la fecha del viaje a fin de comparar las tarifas ofrecidas en distintos días. Debido a esto, en los sitios de venta de pasaje online, una de las acciones principalmente repetidas por los usuarios es el cambio de fechas quienes lo hacen, como se mencionó, en función de averiguar la diferencia de precios en un rango de días.

La búsqueda del mejor precio hace que el interesado llegue a repetir la misma acción innumerable cantidad de veces para alcanzar a visualizar en cual período le es más económico viajar. Para evitar que el interesado tenga que cambiar el filtro a cada vez que quiera cambiar la fecha del viaje, es de gran utilidad tener un link que extienda los resultados de la búsqueda mostrando un rango de días más amplio de lo seleccionado con el filtro, mostrando al mismo tiempo los precios para cada día. Esto le proporcionará una mejor visualización y le permitirá comparar de una sola vez los precios disponibles para los distintos días, o en un rango amplio de fechas.

El *bad smell* tratado por este *refactoring* es la repetición continua de determinados pasos del proceso por el usuario.

Mecanismo

Para aplicar este *refactoring* se deben seguir los siguientes pasos:

1. Identificar la página del proceso en la que se muestran los precios de los pasajes.
2. Agregar un nuevo <*processClass*> que exhibirá los resultados de la búsqueda, extendido el parámetro a un rango más amplio de días (a la elección del desarrollador).
3. Agregar un <*processLink*> en la página identificada en el paso 1 (uno) y definir como *target* el nuevo <*processClass*> definido en 2 (dos).
4. Crear las asociaciones que sean necesarias para conectar la nueva <*processClass*> definida en el paso 2 (dos) con los el resto de los pasos del proceso.

Ejemplo

El ejemplo utilizado para demostrar este *refactoring* es de la página de venta de pasajes de trenes online de la empresa Virgin (www.virgintrains.co.uk). En esta página el usuario determina en el filtro el origen y

destino del viaje y los días en que pretende viajar. Como resultado el sistema muestra una página en la que se exhiben los precios de los pasajes para los días seleccionados solamente, mostrando la variación de precio de acuerdo con la hora del viaje, pero no de acuerdo con el día. En caso de que el usuario desee seleccionar otro día para comparar las tarifas ofrecidas por la empresa, necesita clicar en el link "*Edit Journey*" (Editar Jornada) para alterar las fechas y volver a ejecutar el filtro para obtener los nuevos resultados.

Si el usuario desea comparar precios en un rango más amplio de días, como una semana por ejemplo, tiene que repetir el cambio de fechas al menos seis veces, además, como el usuario no puede visualizar de una sola vez los diferentes precios, la posibilidad de compararlos resulta poco práctico, hecho que incide en la usabilidad de la página.

En la Figura 37 se puede ver que los resultados de la búsqueda son mostrados en una <*processClass*> llamada "*ResultadosFiltro*".

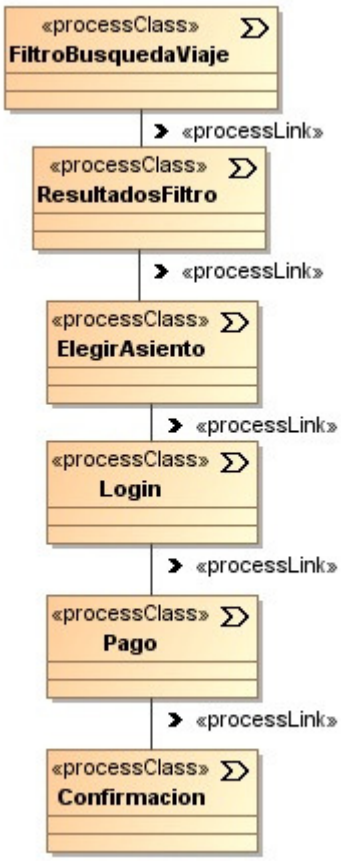


Figura 37: Diagrama de navegación del proceso de compra de tickets en el sitio de Virgin antes del *refactoring*

Para ofrecer esta nueva visualización al usuario, es necesario agregar un nuevo <*processLink*> a la página de resultados, que llevara a otras páginas donde se exhibirán los resultados extendidos. Se crea por lo tanto otro <*ProcessClass*> donde se exhiben los resultados de los precios de los tickets para días próximos a los seleccionados en el filtro, como se ve en el diagrama representado en la Figura 38.

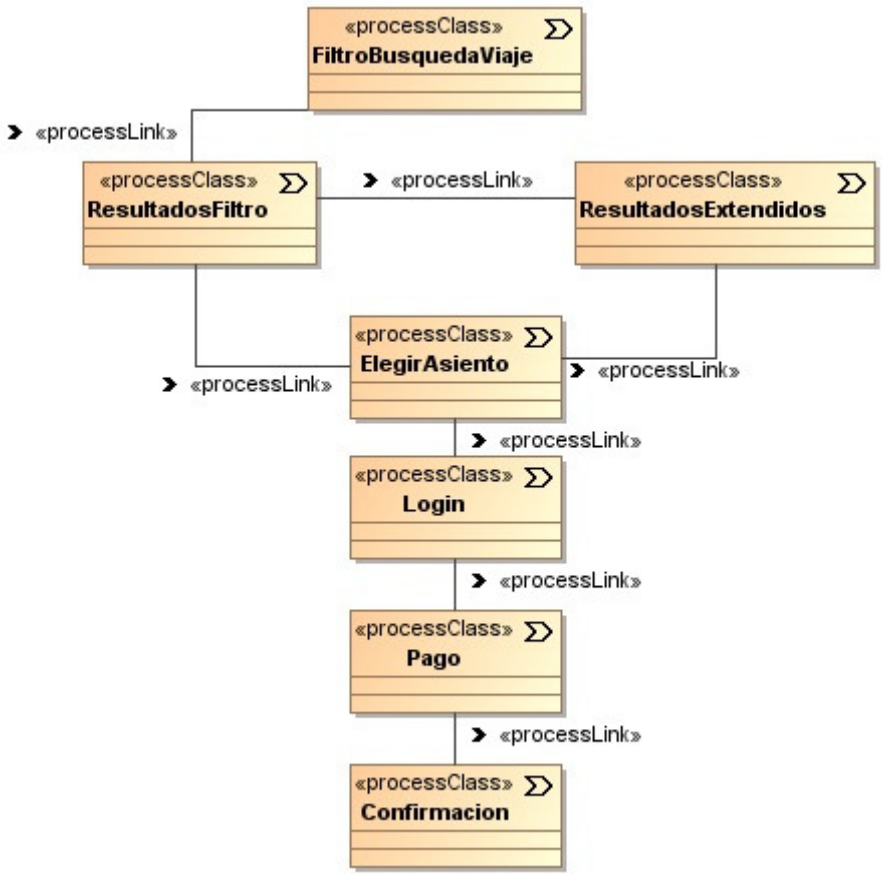


Figura 38: Diagrama de navegación del proceso de compras de tickets en el sitio de Virgin después del *refactoring* "Agregar la posibilidad de extender filtros del proceso"

Con la aplicación del *refactoring* se agregó un nuevo link en la página de resultados y un nuevo nodo de navegación en el proceso. En caso de que el usuario no tenga el interés de visualizar otras fechas distintas de las definidas por él mediante el filtro, puede seguir con el proceso de compra sin pasar por el nuevo nodo agregado, ya que no afecta la navegación principal, caso no sea seleccionado. El nodo agregado sirve como una ayuda a aquellos usuarios que están interesados en tener una visualización más amplia y que tienen flexibilidad en las fechas del viaje. Una vez que esté en el nuevo nodo y pueda seleccionar los días que más le conviene, puede seguir con el flujo de la navegación del proceso y completar el proceso de compras, sin la necesidad de repetir pasos.

En la Figura 39 se puede visualizar un ejemplo de la aplicación de este *refactoring* en la página de venta de tickets online de la empresa "*The Train Line*" (www.thetrainline.com). En la página en que se exhiben los resultados fue agregado un link que dice "*5 day view*" (vista de 5 días), como se puede ver marcado en rojo en la Figura 39. Al clicar en este link el usuario puede visualizar las diferentes opciones de precio para un rango de cinco días, incluyendo la fecha seleccionada por el usuario. Se estima que esta modificación convierte a la página en práctica pues el usuario pueda comparar, de un solo vistazo, las distintas tarifas para los diferentes días y con eso tomar la decisión que más le convenga con relación a sus necesidades y posibilidades económicas.

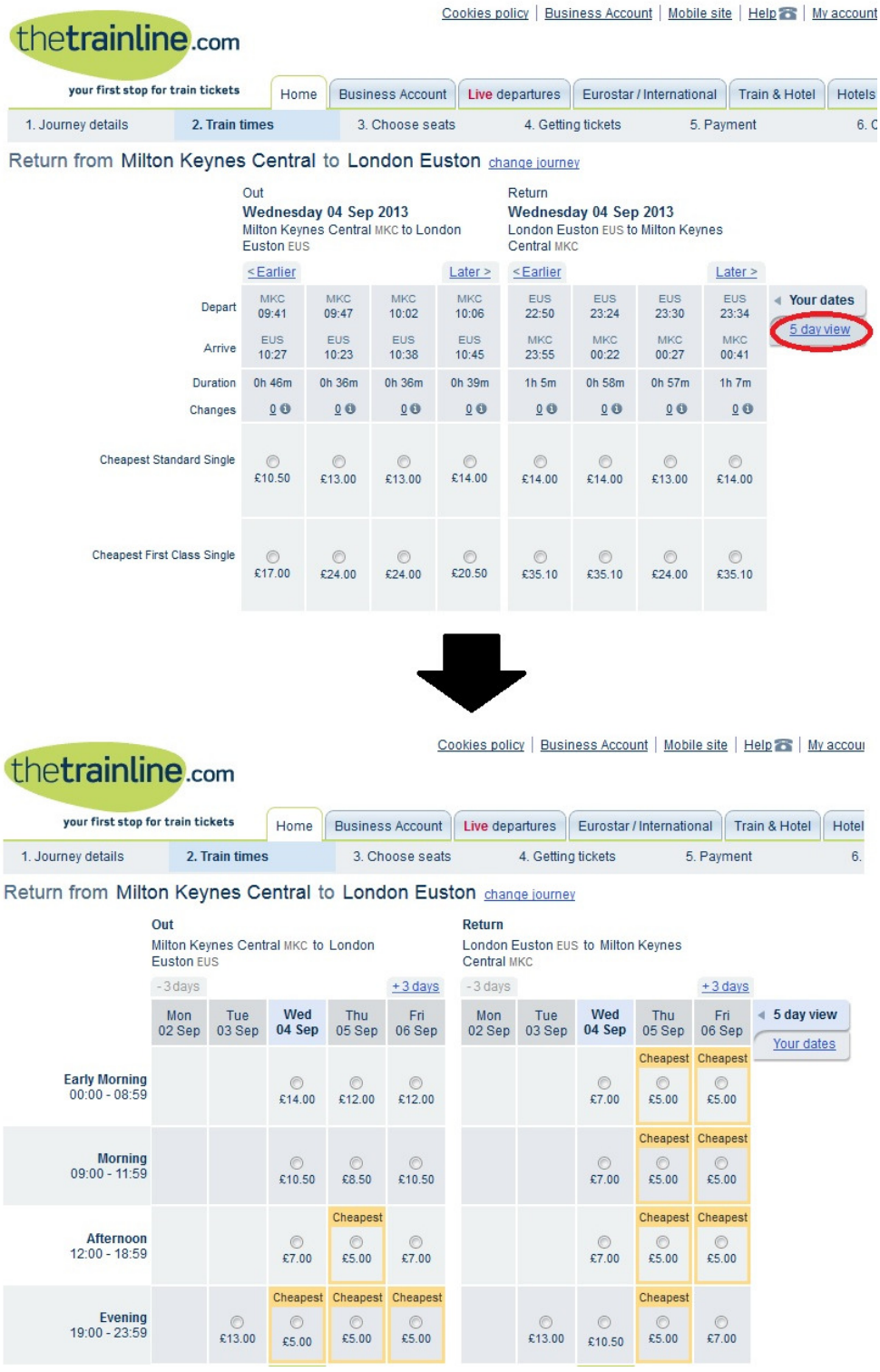


Figura 39: Ejemplo del sitio de compras de tickets online www.thetrainline.com, mostrando el link para extender el filtro.

Mejoras Intencionadas

- Reducción de la repetición de pasos específicos del proceso;
- Incremento de la flexibilidad del proceso, que puede ofrecer más opciones al usuario;

- Mayor eficiencia en el proceso de búsqueda ya que visualizando más opciones el usuario puede hacer su elección más rápido y con más consciencia;
- Reducción del tiempo para completar el proceso;
- Aumento de la satisfacción del usuario que es brindado con opciones que facilitan la ejecución del proceso de negocio en la aplicación;

5.2.4.Eliminar <*navigationLinks*> innecesarios

Motivación

Los <*navigationLinks*> son links que posibilitan que el usuario pueda navegar por los nodos de la aplicación, a diferencia a los <*processLinks*> que son links que inician la ejecución de un proceso de negocio. Cuando un proceso de negocio es disparado, la intención es que el usuario siga una secuencia de pasos hasta concluirlo. Para mantener el usuario focalizado en completar esos pasos, los nodos del proceso deben tener la navegación limitada y contener solamente los links relevantes y necesarios respecto del proceso. En caso de que existan <*navigationLinks*> disponibles en las páginas del proceso, el usuario tiene la opción de navegar por esos links que apuntan a páginas afuera del proceso y interrumpiéndolo antes de su finalización.

Los <*navigationLinks*> que no están relacionados o que no son necesarios para la ejecución del proceso de negocio deben ser eliminados de las páginas del proceso por significar una distracción para el usuario, además de ser una fuente de confusión y/o desvío del foco u objetivo principal, que es concluir con el proceso iniciado. Por eso, para evitar posibles inconsistencias generadas por clicar en links que no pertenecen al proceso, este *refactoring* sugiere eliminar los <*navigationLinks*> que están disponibles en las páginas del proceso y que no sean necesarios para su conclusión. De esta manera la navegación del proceso queda restringida solamente a los pasos estrictamente necesarios para concluir el mismo.

El *bad smell* que se ataca con este *refactoring* es la confusión del usuario al navegar por páginas afuera del proceso así como la inconsistencia generada en caso de que el proceso se interrumpa antes de su conclusión.

Mecanismo

Para aplicar este *refactoring* es necesario seguir los siguientes pasos:

1. Identificar los <*navigationLinks*> presentes en las páginas del proceso.
2. Identificar los <*navigationLinks*> que no son necesarios para la ejecución del proceso de negocio a realizar.
3. Eliminar los <*navigationLinks*> identificados en el paso 2 (dos).

Ejemplo

El ejemplo que se analiza para este *refactoring* es la página de la librería online de la Editora UFV (www.editoraufv.com.br). El usuario cuando inicia el proceso de *checkout* para la compra de un producto puede ver en la primera página del proceso la descripción de los próximos pasos y los botones para avanzar con el proceso. Sin embargo, mientras navega por el proceso, el usuario tiene libre acceso a otros links que llevan a nodos de navegación que no pertenecen al mismo e incluso a links que apuntan a nodos de navegación externos a la aplicación web (*external links*), como links a Facebook, Twiter, etc.

La disponibilidad de links que no guardan relación con la ejecución del proceso, puede resultar confuso y da al usuario la posibilidad de abandonar el proceso sin haberlo concluido. Como se mencionó, ese tipo de acción no es deseable ya que genera inconsistencia del proceso que se interrumpe y confusión del usuario que tiene que volver a iniciarlo desde principio, por haber navegado por páginas externas al mismo.

En la Figura 40 se puede observar que, para completar el proceso el usuario tiene que navegar apenas por cuatro <*processClass*>: loginUsuario, Dirección, ConfirmacionDatos y FormaPago. Sin embargo, la navegación se puede extender si se toma en consideración todos los <*navegationLinks*> que se encuentran disponibles en cada uno de los nodos del proceso y que habilitan la navegación del usuario por páginas propias de la aplicación o externas a ella. Mientras navega por los cuatro pasos del proceso, el usuario puede tener acceso a links que los llevan a las páginas pertenecientes a la librería como las páginas de Promociones, Lanzamientos, Productos, entre otros. Están también disponibles links a aplicaciones externas al sitio de la librería, como Facebook, Blogger y Twitter.

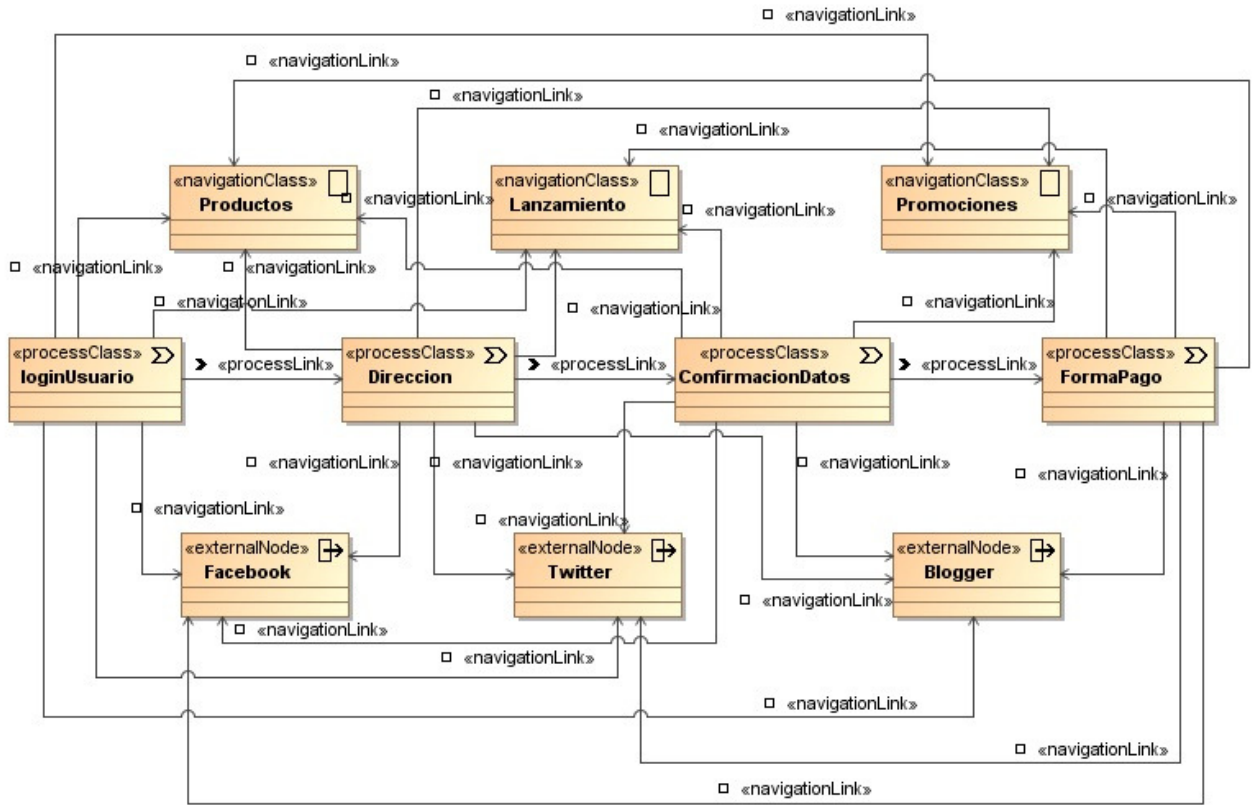


Figura 40: Diagrama de navegación de la librería online UFV antes del *refactoring*

Todos eses <*navigationLinks*> que no son parte de proceso y que representan una distracción al usuario deben ser eliminados para evitar que el mismo se vea tentado a abandonar el proceso sin concluirlo. Al eliminar esos <*navigationLinks*> innecesarios, el usuario es guiado a través de los pasos del proceso y no tiene la posibilidad de navegar por páginas externas al mismo. De esta manera, el usuario puede ejecutar el proceso focalizando en su conclusión sin distracciones y sin confusión.

En la Figura 41 se muestra el diagrama de navegación del proceso de compra del sitio de la librería online después de la aplicación del *refactoring*. Fueron eliminados todos los <*navigationLinks*> que no estaban directamente relacionados con el proceso. Sin los variados links, la navegación del proceso queda más clara y limpia y el usuario puede identificar fácilmente los links que lo llevan a avanzar en el proceso.

Figura 41: Diagrama de navegación del proceso de *checkout* en la página de la librería online UFV después del *refactoring* "Eliminar <navigationLinks> innecesarios

En las principales páginas de comercio electrónico, cuando un <*processClass*> se dispara, la navegación de la página se altera para limitar el usuario a visualizar únicamente los links relevantes al proceso. En la Figura 42 se puede visualizar la página de Amazon (www.amazon.com) antes de disparar el proceso de *checkout* (arriba) y después de iniciado el mismo (abajo). En la primera, se puede ver que antes de iniciar el proceso el usuario tiene acceso a varios links de la aplicación y puede navegar por todos ellos. Pero, una vez que el botón de "*Proceed to Checkout*" (marcado en rojo) es presionado, la navegación de la página es alterada y sólo contiene los links necesarios para la ejecución del proceso de compra, no ofreciendo ningún link que lleve el usuario por afuera del proceso, tampoco que le permitan acceso a la *Homepage*.

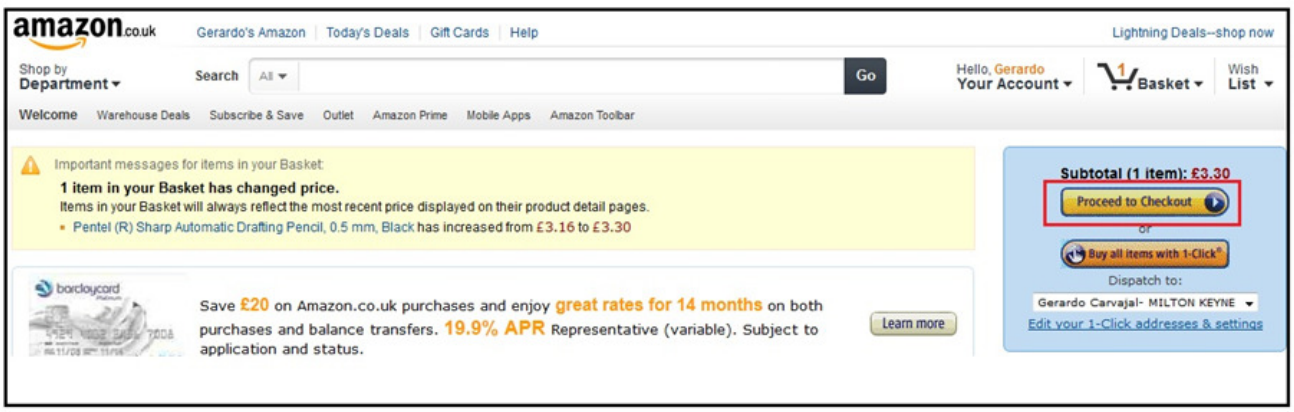


Figura 42: Página de Amazon antes de iniciar el proceso de *checkout* (arriba) y después de disparar el proceso de *checkout* (abajo)

Este *refactoring* tiene el objetivo de limitar la navegación para que el usuario pueda focalizar en la realización del proceso, no obstante es importante también ofrecerle una forma de salir del proceso antes de su conclusión, por eso se recomienda fuertemente la aplicación del *refactoring* "Agregar posibilidad de Cancelar un proceso" descripto en el capítulo 4.2.8. el cual oferta una salida, en caso en que el usuario no quiera o no pueda finalizar el proceso. En esta situación se podrá cancelar el proceso y volver a navegar por las los nodos de navegación normales de la aplicación sin dificultad.

Mejoras Intencionadas

- Mantener el usuario sin distracciones, focalizado en la ejecución del proceso;
- Evitar el estado de inconsistencia del proceso iniciado caso este sea interrumpido antes de su finalización;
- Evitar la desorientación del usuario quien puede llegar a no saber cómo retornar al paso dejado de lado en el proceso;
- Reducción de las repeticiones del proceso , puesto que la mayoría de las veces el usuario tiene que volver a iniciar el proceso interrumpido;
- Reducción del riego de errores ya que previne que el usuario salga del proceso antes de su conclusión;

6. *REFACTORINGS* SOBRE EL MODELO DE PRESENTACIÓN

6.1. Definición

Los *refactorings* aplicados sobre el modelo de presentación persiguen como objetivo: optimizar la interface de manera que los elementos que puedan disparar un <*processClass*> sean más accesibles, visibles y fáciles de localizar por los usuarios en la aplicación web y mejorar la exposición en la interface mientras el usuario ejecuta un proceso de negocio. Por consiguiente estos *refactorings* deben preservar la:

- Disponibilidad de cada elemento existente en el modelo de navegación.
- Disponibilidad de elementos que disparen los <*processClass*>

Basadas en los condicionantes anteriores, los *refactorings* en el modelo de presentación pueden realizar las siguientes operaciones:

- Agregar informaciones relevantes en los elementos que disparan un *processClass;*
- Describir pasos de un *processClass;*
- Eliminar duplicaciones de elementos que disparan un mismo *processLink*;
- Alterar la presentación de un *processClass;*

6.2. Catálogo

6.2.1. Agrupar validaciones en la misma página

Motivación

En el *refactoring* descripto en el capítulo 4.2.1 "Agrupar Validaciones" aplicado sobre el diagrama de proceso, se sugirió agrupar las validaciones para disminuir el tiempo de espera y el esfuerzo del usuario. El *refactoring* fue aplicado en el proceso de negocio pero la presentación de este proceso también se ve afectada por los cambios realizados. El *refactoring* descripto en este capítulo expande el impacto de lo que fue realizado en el modelo de proceso para el modelo de presentación.

En los formularios cortos que necesitan de la interacción del usuario, es importante que la información relevante esté centralizada en la misma página, así la persona puede visualizar exactamente lo que se requiere de ella y los datos que debe tener en cuenta para completar el formulario. Cuando la información y las validaciones están separadas en distintas páginas de la aplicación, se dificulta la visualización y deja de ser práctico para el usuario finalizar la tarea ya que este tiene que esperar que se carguen las otras páginas para poder proseguir con el proceso.

Según uno de los patrones de usabilidad publicado por Perzel y Kane et al. en [72], "*All they see is all they get*", es deseable que el usuario visualice todo el formulario que tiene que llenar de una sola vez, sin tener que bajar o cambiar de página para ver todo el contenido, de esta manera este puede percibir inmediatamente lo que tiene que hacer, sin acciones extras o distracciones.

El *bad smell* tratado por este *refactoring* es la falta de fluidez en el proceso.

Mecanismo

Para ejecutar este *refactoring* los siguientes pasos deben ser realizados:

1. Identificar todas las validaciones de un formulario que son realizadas en distintas páginas.
2. Mover, según su orden, los campos de las últimas páginas hacia la primera página del formulario.
3. Agrupar en el primero botón de validación todas las validaciones de las páginas posteriores.
4. Eliminar las páginas inmediatamente posteriores a la primera, en el caso de no haber más ningún otro objeto o campo por validar que justifique la existencia de dichas páginas.

Ejemplo

Para este *refactoring* se utiliza el mismo ejemplo del capítulo 4.2.1 "Agrupar Validaciones", el proceso de Login de la página de *Home Banking* del Banco Nación Argentina (www.bna.com.ar).

El proceso de Login posee un pequeño formulario que se divide en dos páginas distintas. En la primera página el usuario tiene que digitar su nombre de usuario y presionar el botón "Ingresar" para que el nombre digitado sea validado. Si es válido, el sistema carga una segunda página donde el usuario tiene que informar su clave y otra vez presionar el botón "Ingresar" para que el sistema ahora valide su contraseña. En resumen, si bien el login a una cuenta supone un formulario sencillo y con pocos campos a validar, los campos y las validaciones están innecesariamente divididos en dos páginas distintas, como se ilustra en la Figura 43.

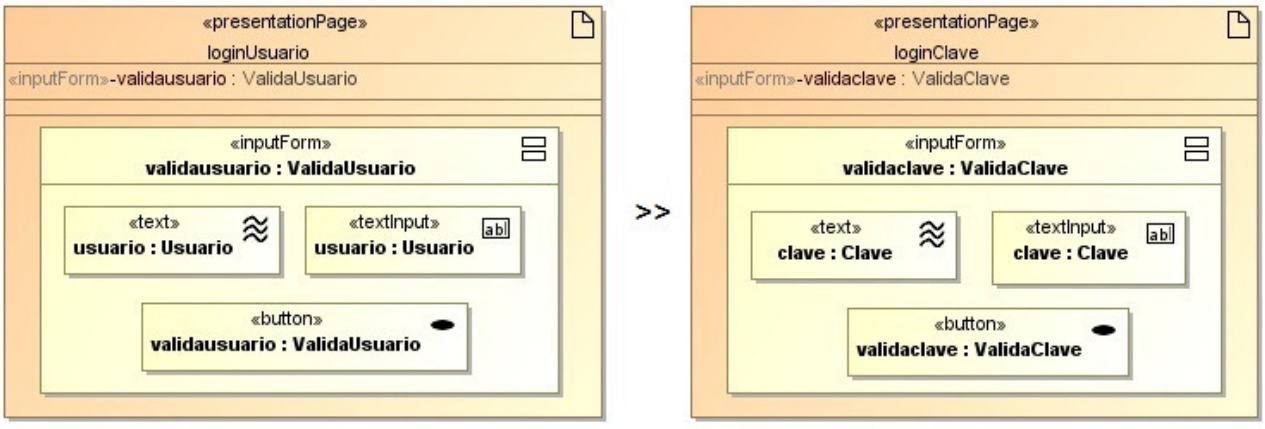


Figura 43: Modelo de presentación de las páginas de *Home Banking* de Banco Nación Argentina antes del *refactoring*

Este *refactoring* propone centralizar todos los campos del formulario en la primer página y agrupar las validaciones que antes se realizaban por separado. En la Figura 44 se puede percibir que los campos Usuario y Clave ahora están en la misma página y que, el botón que antes validaba solamente el Usuario, ahora valida también la Clave, operación que inicialmente se realizaba en la segunda página. De esta manera el usuario tiene todos los campos a la vista y puede completar el formulario de forma más fluida y rápida.

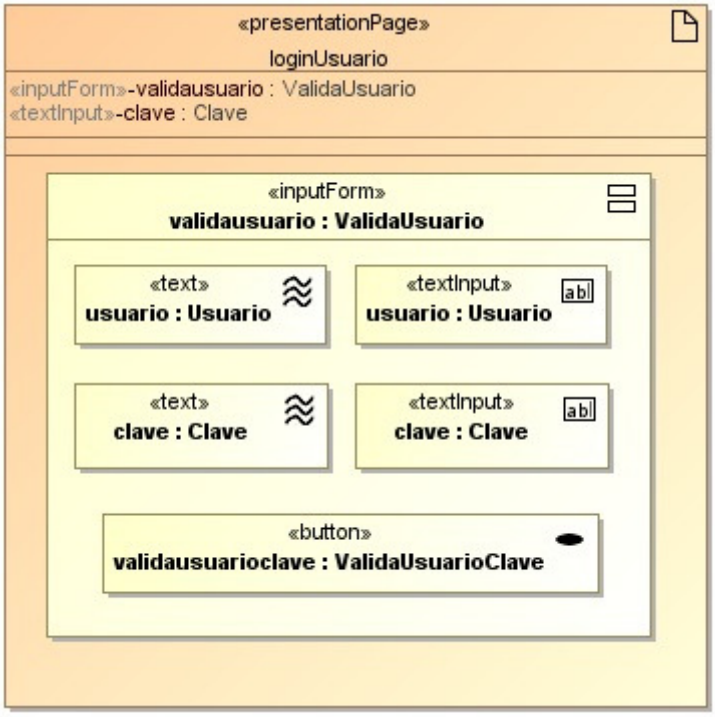


Figura 44: Modelo de presentación de la página de *Home Banking* de Banco Nación Argentina después del *refactoring* "Agrupar Validaciones en la misma página"

Mejoras Intencionadas

- Reducción del tiempo empleado para completar el proceso de login;
- Mejor visualización del proceso a ejecutar que se encuentra centralizado en una sola página;
- Reducción de la interacción cliente-servidor;

Refactorings Relacionados

Como mencionado al principio de este capítulo, este *refactoring* está relacionado con el *refactoring* descripto en el capítulo 4.2.1: "Agrupar Validaciones" que fue aplicado sobre el modelo de procesos. El *refactoring* descripto en esta sección se trata del impacto en el modelo de presentación debido a los cambios realizados en la estructura del proceso descripto en el capitulo 4.2.1.

6.2.2.Agregar Acceso Directo a la *Homepage*

Motivación

El *refactoring* descripto en el capitulo 5.2.1 “Crear Acceso Directo” aplicado sobre el diagrama de navegación sugirió cambios en la navegación de la página a fin de mejorar el acceso a un proceso de negocio. El *refactoring* descripto en este capítulo demuestra el impacto de la creación del acceso directo en el modelo de presentación de la página web y como este modelo se altera.

Krug [10] y Nielsen [11] hacen una importante observación respecto de cómo los usuarios interactúan en los *web sites*. Ellos afirman que estos no leen las páginas web sino las escanean buscando la información que más se asimila con lo que tienen en mente y con lo que están buscando en ella. Sostienen que mayoritariamente los interesados buscan completar el objetivo que los llevó a esa aplicación, por eso no les interesa leer o perder mucho tiempo en la página en pos de la información, por el contrario, cuanto más rápido puedan localizar lo que buscan, mejor y más satisfechos van a estar.

Para evitar que el usuario tenga que vagar por la página en búsqueda de la información deseada, es siempre deseable que los procesos más utilizados estén visualmente accesibles. En una página web, un acceso directo es un atajo a una funcionalidad bastante utilizada por los usuarios con el objetivo de facilitar su acceso. La creación del acceso directo hace que sea más rápido y más simple al usuario llegar a la funcionalidad que desea, reduciendo el número de pasos necesarios para eso, como se demostró en el *refactoring* “Crear Acceso Directo” aplicado sobre el diagrama de navegación. Desde el punto de vista de la presentación de la página, el acceso directo hace que dichas funcionalidades queden más visibles y destacadas en la *Homepage,* más accesibles al usuario.

El *bad smell* que se aborda es la dificultad de encontrar y acceder a funcionalidades básicas en la aplicación.

Mecanismo

Para aplicar el *refactoring* en el diagrama de presentación, se debe:

1. Identificar el <*processClass*> que necesita un acceso directo.
2. Agregar en la *Homepage* los *widgets* necesarios para acceder al <*processClass*> identificado en el paso 1 (uno).

Ejemplo

El ejemplo se basa en la página de un banco donde la funcionalidad de acceso a la cuenta no sigue el patrón de la mayoría de las páginas de *Home Banking:* tener un acceso directo rápido para que el usuario pueda ingresar su Usuario y Clave y acceder a sus cuentas rápidamente. En la mayoría de los bancos esa funcionalidad se encuentra en el costado superior de la página, resultando visible y fácilmente identificable dados sus campos de: Usuario y Clave.

Los patrones son buenos indicadores de que lo que se implementó tuvo una cierta aceptación. En el caso de las aplicaciones web es positivo hacer uso de los patrones existentes puesto que los usuarios ya están familiarizados con ellos y suponen mayor facilidad de uso [10]. Uno de los patrones utilizados en las páginas

de *Home Banking* consiste en presentar al cliente, de manera visible, los campos de Usuario y Clave para el ingreso a las cuentas. Un acceso distinto a este patrón acarrea al usuario un grado de dificultad adicional y a prima face innecesario, como lo es encontrar el nuevo mecanismo de Login a las cuentas.

En la página del Banco Nación Argentina, en vez de tener el acceso directo, la página muestra una imagen que también es un link a otra página. Un problema es que, por tratarse de una imagen y no de un texto sub-rayado como convencionalmente se usa, resulta difícil para el usuario asociar la imagen a un link, visto que está habituado al otro modelo. Otra contrariedad es que la mayoría de los usuarios buscan en la página principal de un banco, automáticamente, por los campos de Usuario y Clave y no por una imagen. El hecho de no tener esos elementos demanda tiempo adicional para que el usuario busque en la página por links que puedan llevarle a realizar esta tarea. Ese tiempo adicional es vivido como pérdida de tiempo.

La Figura 45 muestra la vista parcial del diagrama de presentación que describe como es el acceso a la funcionalidad Login de Cuentas en la secuencia de páginas del banco estudiado. La *Homepage* posee una imagen con un link que lleva a una segunda página llamada *Home Banking*, esta a su vez posee también una imagen con un link a una tercer página llamada *Login Usuario*; recién en esta se encuentran los campos "Usuario" y "Clave" y el botón "Ingresar" que permite que el usuario finalmente entre mediante sus datos y acceda a las informaciones de su cuenta.

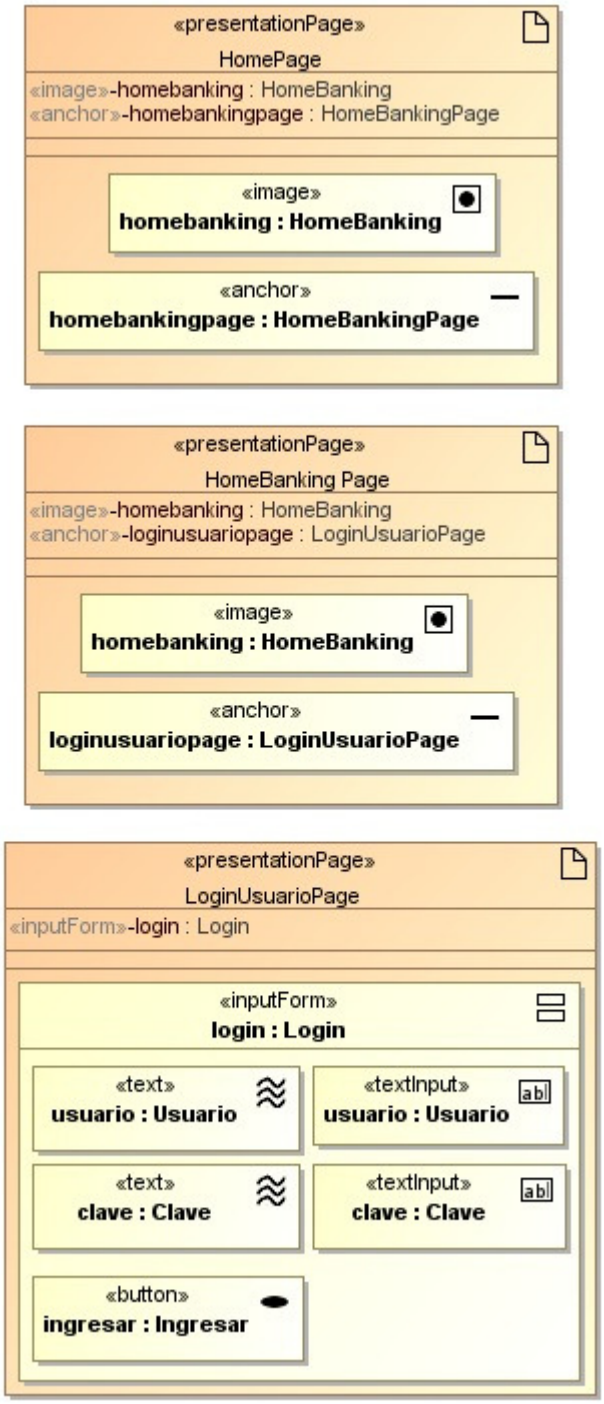


Figura 45: Vista parcial del diagrama de presentación antes del *refactoring*

La Figura 46 exhibe como el *refactoring* “Agregar Acceso Directo a la *Homepage*” reemplaza en la *Homepage* la imagen/link que llevaba a página de *Home Banking* por los campos de "Usuario" y "Clave" y el botón “Ingresar”, los que antes se encontraban en la página de Login de Usuario. Dado que el diagrama de presentación en UWE no define aspectos concretos de la página, como es la posición, ésta no será definida.

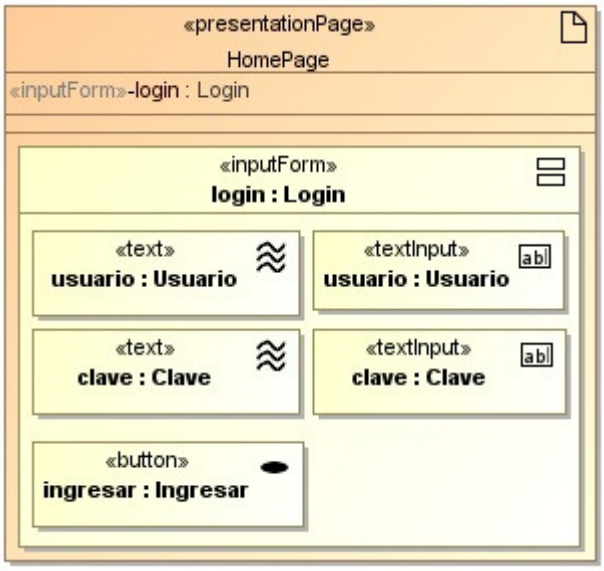


Figura 46: Vista parcial del diagrama de presentación después del *refactoring* "Agregar Acceso Directo a la Homepage".

Creando este atajo el usuario sabe, al instante que entra en la página, donde debe ir para acceder a sus cuentas y lo hará sin perder tiempo rastreando en la página por links o botones que lo lleve a esta funcionalidad. Otra ventaja es que el usuario se ve re-direccionado a otra página, con lo cual se acortan los pasos necesarios para la concretización de su objetivo, como se describió en el *Refactoring* "Crear Acceso Directo" sobre el diagrama de navegación.

Mejoras Intencionadas

- Optimizar la presentación del <*processClass*> en la *Homepage;*
- Reducción del tiempo necesario para disparar el <*processClass*>;
- Aumento de la satisfacción del usuario que puede acceder a un proceso usado frecuentemente más fácilmente;
- Mejorar la presentación de la aplicación web que se aliña con los patrones utilizados por las páginas de *Home Banking;*

Refactorings Relacionados

Como se dijo previamente, este *refactoring* está relacionado con el *refactoring* "Crear acceso directo" descripto en la capítulo 5.2.1 sobre el diagrama de Navegación ya que se trata del impacto en el diagrama de presentación debido a los cambios realizados anteriormente en el proceso.

6.2.3.Exhibir costos extras del producto

Motivación

En sitios de comercio electrónico o de venta de pasajes aéreos el valor que el usuario tiene que pagar por la compra es muchas veces superior al valor del producto que está comprando. Para el caso, un pasaje o cualquier otro producto, servirían de ejemplo. En este tipo de sitio las compras poseen costos extras que generalmente no se dan a conocer al usuario hasta el último momento, como por ejemplo gastos de envío de la mercadería en el caso de sitios de comercio electrónico o costos de impuestos y tasas de aeropuertos en el caso de pasajes aéreos.

Estos costos extras son variables y adicionan un valor importante a la compra, influyendo en muchos casos en que el cliente concrete o no la operación. Debido a esto se considera de importancia informar al usuario, lo antes posible, acerca de todos los adicionales que va a tener con la compra, para que el mismo pueda hacer una evaluación más precisa acerca del costo ya que este es unos de los principales factores que define si el usuario prosigue o desiste de la compra. Además se estima que, exhibir desde el principio todos los gastos devenidos de la compra, generan como redundancia un efecto de credibilidad en la página a la cual el usuario percibe como sincera, hecho que también repercutirá en la frecuencia de uso de la misma.

Caso idénticos se vive en muchas de las páginas de venta de pasajes, en las que el cliente no tienen acceso a los costos extras de la compra que están realizando hasta las los pasos finales del proceso. Esto supone que quien esté interesado tenga que recorrer todo el proceso de compra para saber finalmente cuanto tiene que abonar. Como en el caso anterior, se estima que, cuanto antes tenga esta información disponible, antes estará en condiciones de tomar la decisión sobre su compra.

El *refactoring* “Exhibir costos extras del producto” propone que los costos extras relacionados a tarifas de envío o tasas sean exhibidos al usuario en la misma página donde se muestran el precio de los ítems, de esta manera el usuario ya sabe de antemano el costo final del producto antes de decidir comprarlo o no.

El *bad smell* que motiva este *refactoring* es la ausencia de información relevante en etapas tempranas del proceso y el consecuente abandono de la aplicación web.

Mecanismo

En el diagrama de presentación aplicar los siguientes pasos:

1. Identificar la página donde el precio del producto está exhibido.
2. Agregar al lado del precio del producto la información con los costos extras

Ejemplo

El ejemplo utilizado para este *refactoring* se extrajo del sitio Amazon (www.amazon.com). Al hacer una búsqueda de determinado producto en la página de Amazon, el sistema muestra un listado de productos con imagen, descripción, precio y otras informaciones como la popularidad del vendedor. Sin embargo, los valores de los costos de envío del producto no se exhiben en la página en un primer momento, como se muestra en la Figura 47. Para visualizar los costos extras el usuario tiene que seleccionar el producto

deseado y esperar abrir una segunda página donde se muestran más detalles. Si el usuario desea saber el precio final de cada producto debe repetir el proceso tantas veces como productos desee comprar.

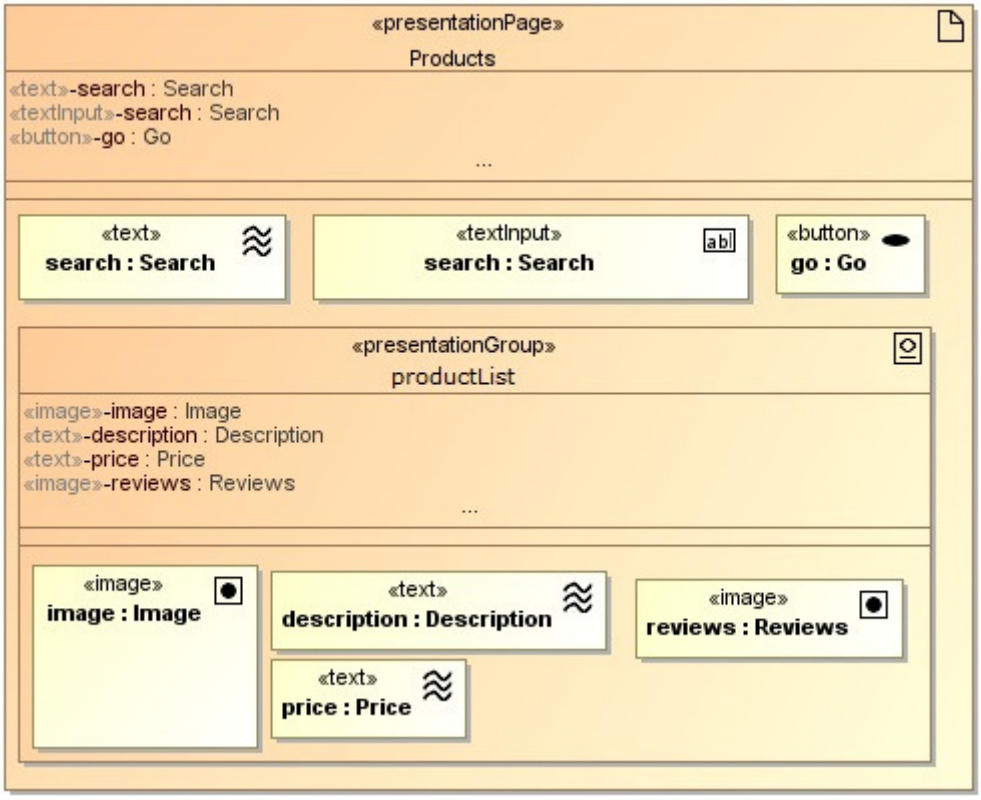


Figura 47: Vista parcial de la página de Amazon antes del *refactoring*

Conviene recordar que no solo la confiabilidad incide en la elección de las páginas de *E-Commerce*, sino también la eficiencia que el usuario pueda percibir en ella, en tal sentido se piensa que agregar el valor de los costos extras al lado del precio del producto, al ahorrar tiempo del usuario que puede visualizar en la misma página el valor final de todos los productos exhibidos en la lista, repercute favorablemente en el imaginario del cliente respecto de la página en cuestión. Se puede observar este *refactoring* en la Figura 48.

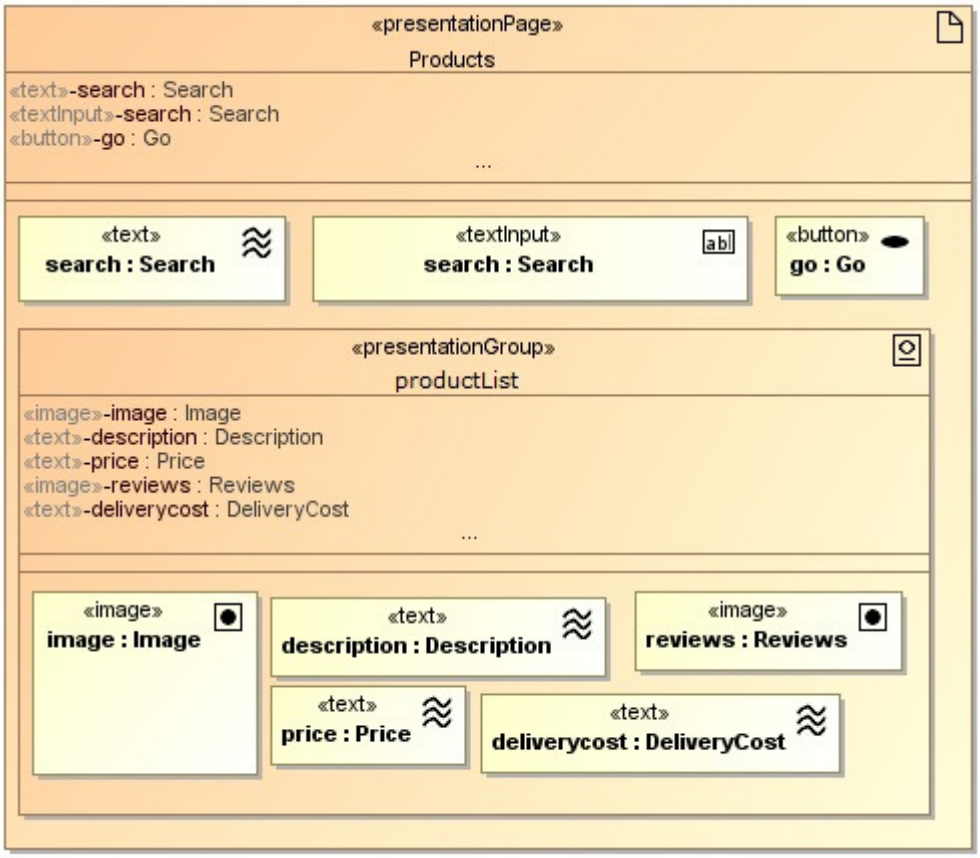


Figura 48: Vista parcial de la página de Amazon después del *refactoring* "Exhibir costos extras del producto"

En otras páginas similares, como EBay (www.ebay.com), el valor de los costos se muestra conjuntamente con el listado de producto, abajo del precio, como se puede apreciar marcado en rojo en la Figura 49. En esta página el usuario cuenta con todas las informaciones del producto y puede tomar una decisión más rápida y segura respecto a la compra.

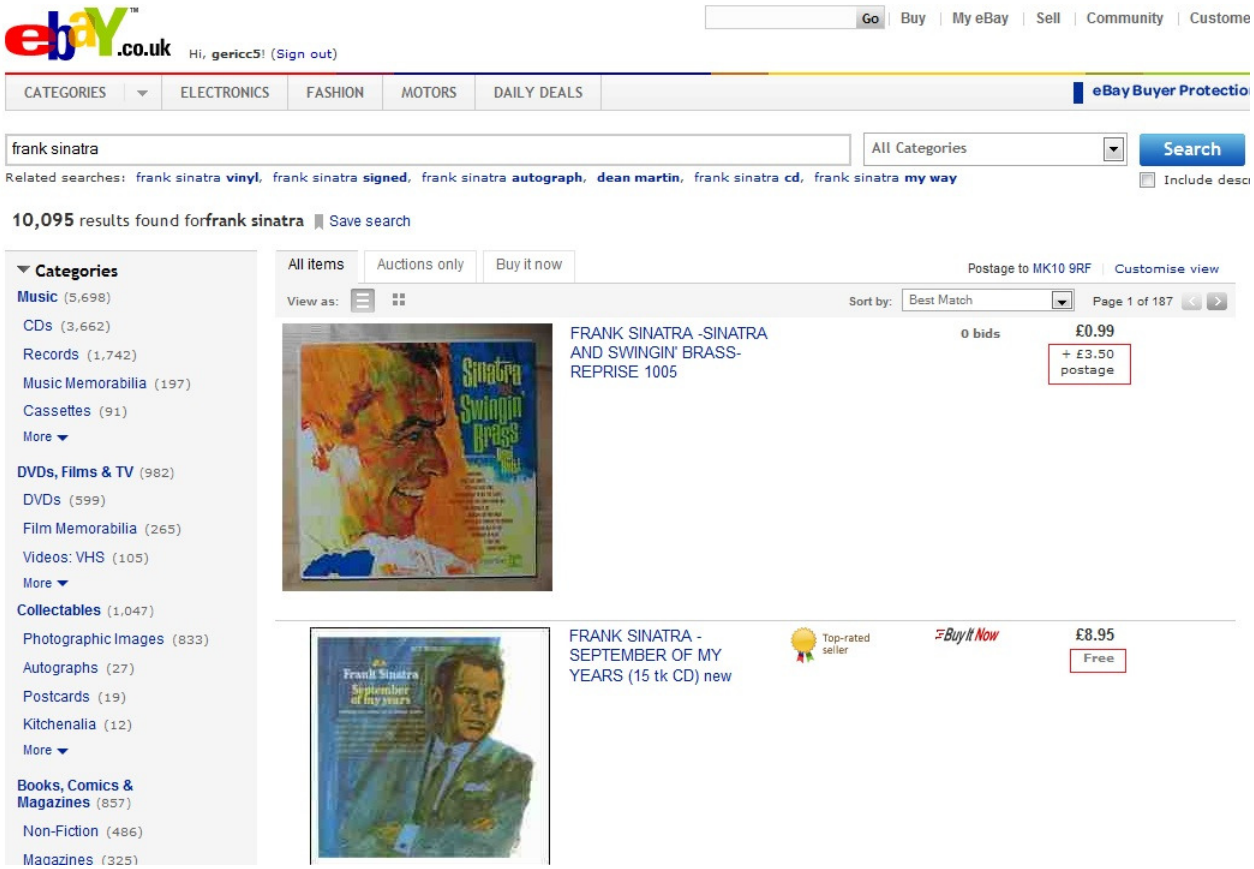


Figura 49: Ejemplo de la página de EBay donde los costos de envío son exhibidos al lado del precio

Mejoras Intencionadas

- Mayor practicidad y facilidad en la ejecución del proceso ya que el usuario puede visualizar el precio de los costos del producto más rápidamente;
- Reducción de las repeticiones en el proceso ya que la información se exhibe en la misma página del precio del producto;
- Incremento de agilidad y rapidez en el proceso de compra;
- Disminución del abandono del sitio web en el proceso de compras debido al costo extra que era exhibido al final;

Refactorings Relacionados

Este *refactoring* se relaciona con el *refactoring* descripto en el capitulo 4.2.2 "Posponer Registración" ya que ambos sugieren cambios para que el usuario tenga el máximo de información disponible en etapas tempranas del proceso.

6.2.4.Explicitar pasos del proceso

Motivación

Los procesos embebidos en las aplicaciones web tienen distintos propósitos por lo que poseen varios pasos. Un proceso de Login generalmente es corto y puede tener dos pasos mientras que un proceso de compra puede tener más de diez pasos dependiendo de la página. Procesos complejos pueden ocasionar en confusión y desorientación de los usuarios mientras navegan por el mismo trayendo dudas respecto a la totalidad del mismo, como afirma también Rossi y Schwabe en [28].

Explicitar los pasos del proceso en la página ayuda al usuario a saber cuantos pasos tiene que recorrer para concluir el proceso, en que paso se encuentra actualmente, sabiendo así cuantos pasos faltan para completar el proceso y presuponer qué puede esperar de esos pasos (por ejemplo, si el próximo paso se llama "Forma de Pago" el usuario puede intuir que puede necesitar informaciones de su tarjeta de crédito). Si por el contrario, no se exhiben dichos pasos el usuario no puede conocer cuantos le falta para completar el proceso o qué tipo de informaciones puede necesitar, situaciones que en ocasiones incide en el abandono prematuro del sitio o en el uso de métodos no deseables como los botones de navegación del *browser*, lo que, como dicho, puede traer inconsistencias al estado del proceso. Por eso se estima que, cuanto más información se dé al usuario respecto del proceso que está ejecutando, más seguridad y confianza va a sentir acerca de la aplicación.

El *bad smell* que se ataca con este *refactoring* es la falta de información respecto al proceso y el abandono prematuro del sitio durante la ejecución del proceso.

Mecanismo

El *refactoring* puede ser aplicado a través de los siguientes pasos:

1. Identificar las páginas que componen el proceso.
2. Agregar *widgets* en las páginas identificadas en 1 (uno) describiendo: numeración del paso actual, descripción de los pasos del proceso, cantidad total de pasos, etc.

Ejemplo

Para ilustrar este *refactoring* se utiliza la página de la librería Cuspide (www.cuspide.com). Al iniciar el proceso de compra de un ítem, la página no provee información respecto de cuántos y cuáles son los pasos que deben seguirse para finalizar con la compra, ni acerca de en qué paso está ejecutando el usuario en el momento. La falta de información produce desorientación en el usuario que no sabe si va a poder concluir el proceso, ni qué pasos e información le falta completar.

En la Figura 50 se ilustra la primera página del proceso de *checkout* del sitio de Cuspide. El proceso se inicia con el Login del usuario al sistema y luego se continúa mediante otros pasos, como los datos de facturación, forma de pago etc. No obstante se puede visualizar que, desde la primera página, tanto como en las otras que componen el proceso, no existe indicación de los pasos a seguir en el proceso.

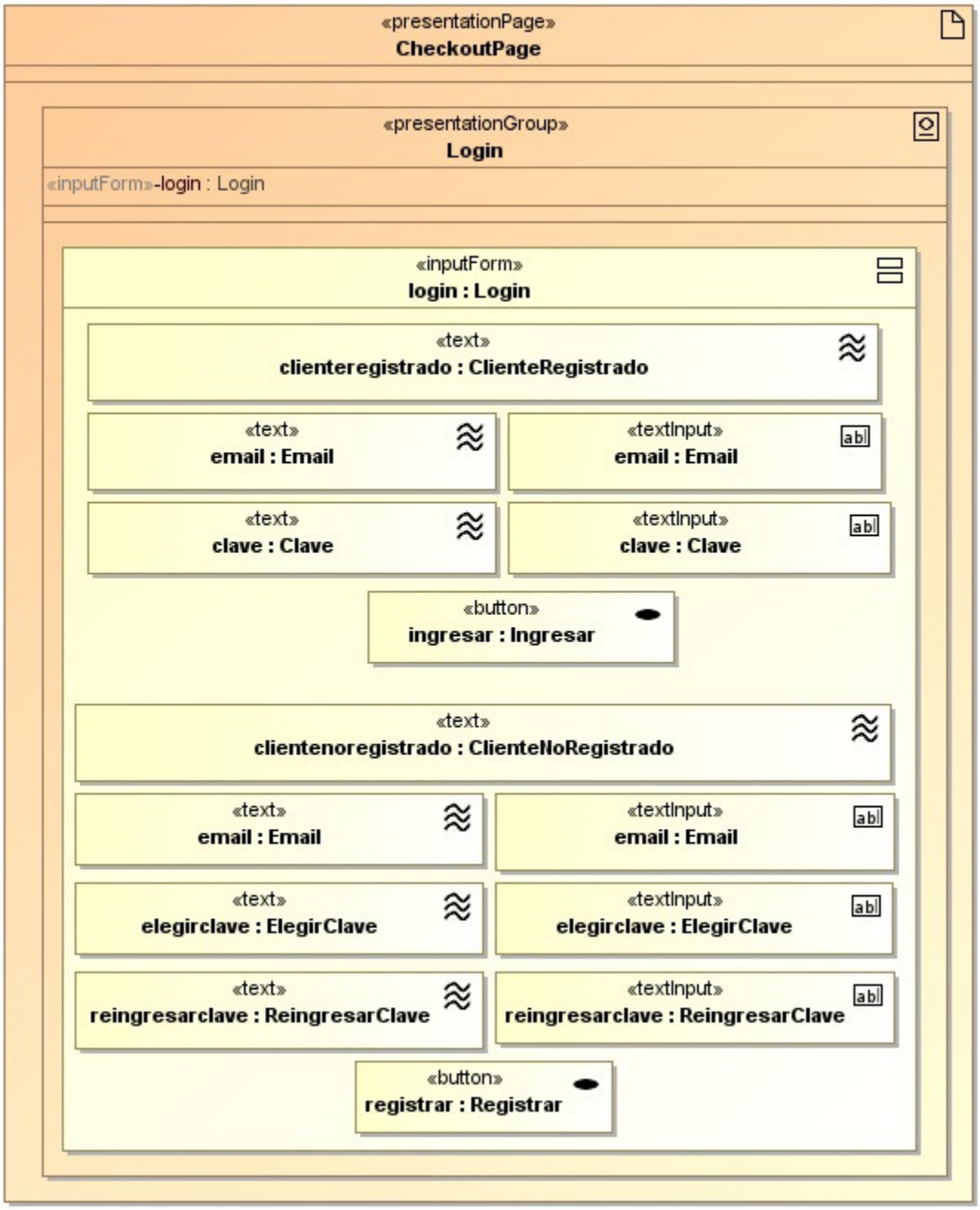


Figura 50: Primera página del proceso de *Checkout* en el sitio de la librería Cuspide antes del *refactoring*

El *refactoring* "Explicitar los pasos del proceso" sugiere exhibir en cada página del proceso todos los pasos con una breve descripción y marcando al mismo tiempo cual es el paso que está siendo ejecutado en el momento. De esta manera el usuario sabe exactamente en qué punto del proceso se encuentra y puede visualizar fácilmente los pasos que ya ejecutó y los que aún le restan ejecutar. La implementación de este *refactoring* es sencilla y puede ser alcanzada con una simple implementación de HTML o combinación de HTML con gráficos [28].

La Figura 51 muestra el diagrama de presentación de la primera página del proceso de *Checkout* de página de Cuspide, al que se agregó un texto que describe los pasos que componen el proceso. Un texto similar

tendría que agregarse en todas las páginas que componen el proceso, de manera tal que el usuario pueda registrar el tamaño del proceso, calcular el tiempo que le demandará su ejecución, cuantos pasos le faltan y en que paso se encuentra.

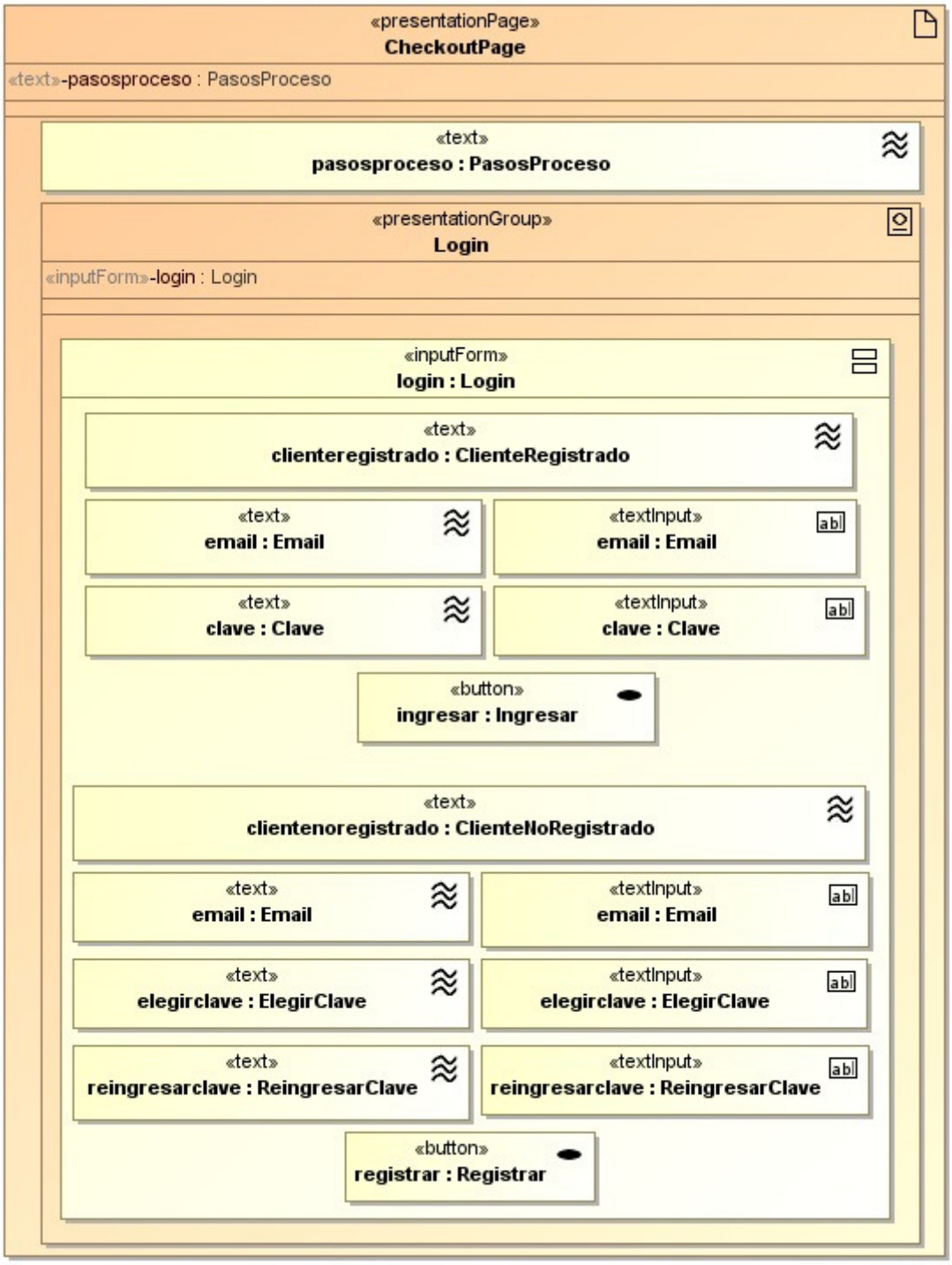


Figura 51: Primera página del proceso de *Checkout* en el sitio de la librería Cuspide después del *refactoring* "Explicitar pasos del proceso"

En el caso que la aplicación no tenga implementado, se sugiere que se aplique el *refactoring* descripto en el capitulo 5.2.2 :"Agregar posibilidad de retroceder en el proceso", de esta manera el usuario puede visualizar

las etapas del proceso y navegar por ellas caso sea necesario corregir o verificar los datos ingresados anteriormente.

Mejoras Intencionadas

- Mejorar el nivel de información ofrecida al usuario respecto del proceso;
- Incorporación de un mecanismo de localización que le permita al usuario conocer en qué punto del proceso se encuentra y cuanto le falta para concluirlo;
- Ofrecer al usuario la posibilidad de calcular el tiempo que le demandará concluir el proceso y que informaciones puede necesitar;
- Evitar el abandono de la aplicación debido a falta de información respecto al proceso;
- Evita la inconsistencia del estado del proceso caso el usuario utilice los botones de navegación del *browser*;

Refactorings Relacionados

Este *refactoring* está relacionados con el *refactoring* descripto en el capitulo 4.2.6: "Agregar actividad Resumen" ya que ambos proponen mejoras en el nivel de información ofrecido al usuario mientras este navega por el proceso de negocio.

Otro *refactoring* relacionado es el detallado en el capitulo 5.2.2: "Agregar posibilidad de retroceder en el proceso" ya que este ofrece una opción segura en el caso de que necesite verificar pasos anteriores en el proceso, previniendo el usuario de utilizar los botones de navegación ofrecidos por el *browser*.

6.2.5.Remover <*processLinks*> duplicados

Motivación

Los <*processLinks*> conducen el usuario a un determinado proceso y están distribuidos en varios lugares de la aplicación web. Como en general existen muchos <*processLinks*> que disparan distintos procesos en la página, tener <*processLinks*> duplicados produce redundancia y confusión al usuario que puede entender que los mismos lo llevan a procesos diferentes.

Se consideran duplicaciones los <*processLinks*> que poseen el mismo nombre y tiene como *target* el mismo proceso de negocio. Por esta razón, con el objetivo de mejorar consistencia y la usabilidad de la aplicación web es recomendable eliminar aquellos que se encuentran duplicados y que tengan la misma función en la página.

El *bad smell* que se ataca con este *refactoring* es la redundancia de información en las aplicaciones web.

Mecanismo

Para ejecutar este *refactoring* es necesario seguir los siguientes pasos:

1. Verificar en la página la existencia de <*processLinks*> con el mismo nombre y que disparan el mismo proceso.

2. Eliminar el/los <*processLink(s)*> excedente(s) dejando sólo uno en la página, para cada función.

Ejemplo

Como ejemplo vamos analizar la página de *Home Banking* del Banco Nación Argentina (www.bna.com.ar). Como se puede percibir en la Figura 52 , al entrar en la página de transferencia de dicho banco, se visualizan varios botones que disparan distintos procesos, como "Imprimir" y "Descargar", marcados en rojo en la figura. Se puede notar que, un poco más abajo en la página, existen otros dos botones, exactamente iguales, con el mismo nombre y que disparan los mismos procesos. La existencia de dos botones iguales y que llevan al usuario al mismo proceso pueden generar confusión en el cliente quien puede pensar que se refieren a distintos procesos o son relacionados con distintas entidades de la aplicación.

Figura 52: Página de Banco Nación Argentina ilustrando la repetición de <*processLinks*> antes del *refactoring*

Los dos botones de "Imprimir" y los dos botones de "Descargar" están relacionados con la misma entidad de la página: la transferencia y por lo tanto imprimen y descargan exactamente la misma información. En cambio en la Figura 53, se puede observar que, con la aplicación de este *refactoring*, se eliminaron los <*processLinks*> "Imprimir" y "Descargar" repetidos, dejando sólo un botón para cada uno de ellos. Con eso se otorga además un aspecto más limpio a la página y se eliminan redundancias innecesarias.

Figura 53: Página de Banco Nación Argentina después del *refactoring*"Remover <processLinks" duplicados

Mejoras Intencionadas

- Incrementar la claridad en la aplicación web;
- Eliminar redundancias innecesarias, que produzcan confusiones en el usuario;
- Mejorar la consistencia de la página que resulta más fácil de entender;

Refactorings Relacionados

Este *refactoring* está relacionado con el *refactoring* descripto en el capitulo 4.2.4: "Remover pasos duplicados" definido sobre el modelo de proceso ya que ambos sugieren cambios para mejorar la consistencia y eliminar redundancias mientras el usuario ejecuta un proceso de negocio.

7. TRABAJOS RELACIONADOS

El campo de *refactoring* fue y sigue siendo constantemente estudiado, desde su primera mención planteada por Opdyke [65] en 1992 en su tesis de doctorado donde definió un catálogo con *refactorings* básicos y mostró, además, que es posible la migración de atributos entre las clases.

Otra contribución a este tema fue el concepto de precondiciones como forma de validar el comportamiento y verificar que no fue alterado debido a la aplicación de los *refactorings.* El concepto de precondiciones también fue mencionado por Roberts [66] quien introduce además el concepto de poscondiciones para poder encadenar *refactorings*. Este autor asimismo planteó el desarrollo de una herramienta usando el lenguaje *SmallTalk* mediante la cual se realizaba automáticamente *refactorings* en los programas, preservando su comportamiento, tal lo expresado en su publicación hecha en conjunto con Brant y Johnson [67].

Más tarde Fowler [12] popularizó el uso de los *refactorings* al definirlos como una forma de limpiar el código fuente de un programa disminuyendo la posibilidad de introducir errores en él. Definió practicas y principios como cuando un *refactoring* debe ser aplicado en el código fuente además de presentar una serie de situaciones que podrían indicar la necesidad de aplicar un *refactoring*, como cuando nuevas funciones son agregadas al programa o cuando se detecta los que define como "*bad smells*". En su publicación este autor definió una diversidad de *refactorings* explicitando el mecanismo a ser ejecutado para aplicar el *refactoring* y la motivación del mismo.

Como se dijo anteriormente el concepto de *refactoring* fue extendiéndose y se aplicó a distintas áreas como en base de datos en "Refactoring Databases: Evolutionary Database Design" de Damabler y Sadalage [32], HTML en "Refactoring HTML: Improving the Design of Existing Web Applications" de Harold [33] y UML en “Refactoring Browser for UML" de Boger, Sturmt y Fragemann [31].

Así como se produjo una ampliación en las áreas de aplicación, el objetivo de los *refactoring* también fue ampliándose pasando de proponer mejoras únicamente en los aspectos internos de las aplicaciones como mantenibilidad, también apunta a proporcionar mejoras en aspectos externos como usabilidad, tal el propósito de esta tesis. La publicación de Garrido, Rossi y Distante [7] es un ejemplo de ello. En la misma los autores presentan un catálogo de *refactorings* destinados a mejorar la usabilidad en aplicaciones web, a través de los cuales proponen cambios en la estructura y definen "*bad smells*" relacionado con la usabilidad. En este trabajo los *refactorings* son clasificados de acuerdo con el factor de usabilidad mejorado (navegabilidad o comprensibilidad, por ejemplo) y el "*bad smell*" vulnerado mediante ellos.

De manera similar estos autores describieron en [15] *refactorings* referidos a los modelos de las aplicaciones web y profundizaron la confección del catalogo anterior con transformaciones aplicados en los modelos de navegación y presentación. Ellos mismos, en sucesivas publicaciones continuaron ensanchando este campo de conocimiento como se observa en [18], donde los autores clasifican los refactorings como WDI - *Web Design Improvements*. En esta publicación también fue analizada la relación entre los *refactorings* de navegación y presentación.

Aun con el objetivo de mejorar la usabilidad, los autores Conte y Travassos [75] desarrollaron una técnica basada en evidencias llamada WDP (*Web Design Perspectives-based Usability Evaluation*) usada para

inspeccionar y evaluar la usabilidad con el objetivo de atender los requisitos de las aplicaciones web. Otro ejemplo que se puede mencionar es la herramienta TOWABE (*TOol for Web Application usaBility Evaluation*) desarrollada por Itakura y Vergilio en [76] que integra diferentes técnicas de evaluación de usabilidad generando reportes que evalúa distintas perspectivas.

En la misma línea de *refactorings* aplicados a modelos de aplicaciones web, Cabot y Gomez [43] expusieron en su trabajo ejemplos de *refactorings* que se aplican al modelo de navegación representado utilizando la metodología WebML. En dicho trabajo los autores hacen una definición de "preservación de comportamiento" aplicable a los modelos de navegación, que se utilizada para garantizar que el comportamiento de la aplicación fue preservado después de la aplicación de los *refactorings* sugeridos. El catálogo de estos *refactoring* apuntan principalmente a mejorar calidades internas del modelo que no son visibles al usuario, como por ejemplo el *refactoring* "*Head-merge navigation path*" que sugiere unir partes similares del camino de navegación mejorándolo internamente, hecho que no es perceptible al usuario final.

Asimismo, en el contexto de los procesos de negocios, existen diversas publicaciones relacionadas con el tema en cuestión, en ellas se considera oportuno rescatar la de Webber [24] quien propone una serie de *refactorings* para ser aplicados en repositorios de procesos, permitiendo que se pueda manejar la complejidad de los mismo haciendo que los modelos de proceso sean más fáciles de mantener y entender. Esos *refactorings* se focalizan únicamente en mejorar aspectos internos del modelo de proceso.

Por su parte Küster, Koehler y Ryndina [59] presentan un enfoque de *process merge* mediante el cual establecen la correspondencia entre dos procesos de negocio, permitiendo identificar la semejanzas entre ellos, situación que facilita la aplicación de *refactorings*. Dijkman, Gfeller, Küster y Völzer [58], por su vez, plantean una técnica utilizada para detectar las oportunidades de *refactorings* de forma automática. La misma se basa en el uso de distintas métricas para determinar el tipo de problema y consecuentemente el tipo de *refactoring* que es necesario aplicar en esas circunstancias. El trabajo muestra la aplicación con resultado de la técnica en dos largos repositorios de procesos. En suma se puede decir que todas estas publicaciones, en general, apuntan a mejorar características internas como comprensibilidad, mantenibilidad y legibilidad de procesos de negocios largos y complejos.

Vanhatalo, Völzer, Leymann y Moser en su publicación [54] refieren a técnicas que posibilitan la realización automática de *refactorings* en los gráficos de *workflow*, generando modelos más bien estructurados y completos, utilizando para ello la notación BPMN. Esta técnica permite que se concluyan gráficos incompletos y descompone de forma modular los gráficos en partes menores.

El modelado de procesos de negocio en las aplicaciones web también es un asunto ampliamente explorado, que puede ser estudiado en diversas metodologías, Distante, Rossi, Canfora y Tilley [2] presentan un modelo de diseño basado en UWAT+, utilizado para integrar los procesos de negocio en la aplicaciones web. En el marco de este trabajo describen también los problemas de diseños enfrentados por los desarrolladores y enumeran requerimientos necesarios para integrar los procesos de negocio en las aplicaciones web, basados en las semejanzas entre las principales metodologías utilizadas para modelar procesos de negocio. En cambio Koch, Kraus, Cachero y Melià [3] utilizan dos metodologías con distintas propuestas, OO-H y UWE, y con ellas hacen una comparación entre el modelado de procesos de negocio con el objetivo de definir un enfoque común en ambas. De esta manera se muestra como los procesos de negocio impactan y están integrados en los modelos de la aplicación web.

Por otro lado Brambilla, Ceri, Fraternali y Manolescu [47] proponen en su publicación un nuevo método para la especificación de aplicaciones web que manejan procesos de negocio utilizándose de la notación WebML. Con este estudio es mostrado como extender WebML usando conceptos básicos de modelado de proceso (BPMN) permitiendo que los desarrolladores puedan especificar tanto los requerimientos de proceso como interacciones en la Web.

La temática de la calidad de los procesos de negocio fue discutida por Griori, Casati, Dayal y Shan [60] quienes propusieron una herramienta para analizar, predecir y prevenir la ocurrencia de excepciones, que son desvíos del comportamiento deseable que el proceso debe tener. La herramienta está focalizada en ayudar las grandes compañías a proveer servicios de alta calidad, consistente y predecible. Para ello sugieren la implementación de técnicas como *datamining* y *data warehousing* para analizar los registros de ejecución de los procesos de negocio siendo posible extraer datos importantes de cuando se produce una excepción; esto permite explicar la causa de tales excepciones e inclusive predecir otras a futuros antes de que ocurran.

Por su parte Weber [55] sugiere un conjunto de dieciocho patrones para mejorar la flexibilidad en los sistemas de información basados en procesos (PAISs). Los PAISs generalmente soportan el diseño de los procesos de negocio a través de gráfico-orientado BPMLs en conjunto con especificaciones textuales. Distante, Tilley y Huang [1] presentan una técnica para la evolución de sitios web a través de reingeniería de transacciones, que son los procesos de negocio. La técnica analiza el modelo de la transacción y determina opciones de reestructuraciones, rediseñando el modelo basado en la análisis realizada.

En el campo de modelado de procesos de negocio en aplicaciones de *E-Commerce* específicamente, Schmid y Rossi [8] presentan un nuevo enfoque que permite tratar los procesos de negocio y la navegación como entidades de primera clase marcando la importancia del modelado de los procesos en aplicaciones como los *E-Commerce*. En su trabajo profundizan la metodología OOHDM introduciendo actividades del proceso en el espacio conceptual y nodos de actividad y proceso en el modelo de navegación. En una nueva publicación [5] estos autores amplían y mejoran el enfoque presentado en anteriormente modelando los procesos de negocio juntamente con los elementos de la navegación. Asimismo, estos autores, en otro escrito que presentan conjuntamente con Lyardet [34] analizan diversos problemas relacionados con la introducción de procesos de negocio en el ciclo de vida de las aplicaciones de *E-Commerce*; en ese estudio enfocan los problemas de personalización de procesos de negocios en función de los distintos perfiles de usuario y analizan estrategias delimitar esos problemas.

Con independencia del modelado de proceso antes mencionado, en otro trabajo Rossi, Schwade y Lyardet identifican en [28] cinco patrones relacionados con las páginas de comercio electrónico, describiendo las características de los cambios sugeridos y los beneficios logrados en las páginas, pero sin modelar los cambios en los modelos de las aplicaciones.

En función de los antecedentes mencionados in supra y atentos a que esos estudios descuidaron el tratamiento de la calidad externa de la aplicaciones web con relación a la ejecución de los procesos de negocio, la propuesta de la tesis fue atender a esta área de vacancia y por tanto se centró el estudio optimizar la manera en que los usuarios de estas aplicaciones se desenvuelven en los procesos de negocio, sea a partir de sus interacciones en la presentación, navegación o en la semántica del proceso de negocio en sí, para lo cual se elaboró y ejemplificó un catálogo de *refactorings* destinado a introducir mejoras tanto en los modelos de presentación, navegación y proceso, objetivo al que espero haber dado cumplimiento con este escrito y en el articulo "*Business Processes Refactoring to Improve Usability in E-Commerce Applications*" [85] asociado que será publicado en el *Electronic Commerce Research Journal este año.*

8. CONCLUSIONES

Debido a la rápida y amplia aplicación en distintos áreas, los *refactorings* resultaron ser una técnica de suceso para combatir la decadencia de los sistemas de software en general y no fue diferente en el ambiente de las aplicaciones web. Como se mencionó, la técnica inicialmente se aplicó con éxito en la mejora el código fuente de *web sites* y también en la reestructuración de los modelos de las aplicaciones web con el objetivo de optimizar características internas y externas de las aplicaciones, como mantenibilidad, eficiencia y usabilidad.

Por su parte esta tesis también se focalizó en la aplicación de *refactorings* en las aplicaciones web, pero a diferencia de otras publicaciones, se concentra especialmente en mejorar características externas relacionadas exclusivamente con la ejecución de los procesos de negocio en las mismas, como usabilidad, eficiencia y la satisfacción del usuario.

Este enfoque se eligió debido al crecimiento vertiginoso del número de aplicaciones que manejan procesos de negocios en el ambiente web y la popularización que tales servicios adquirieron entre los usuarios. Como cada vez es más común ejecutar operaciones y actividades online, se identificó la necesidad de mejorar la calidad de uso de esos procesos a fin de garantizar una mejor experiencia a los usuarios a la hora de ejecutar los mismos.

Debido a esta vasta diseminación entre los usuarios hoy en día, se estudiaron específicamente dos tipos de aplicaciones web, las páginas de *E-Commerce* y las páginas de *Home Banking* y de estos dos tipos de aplicaciones se analizaron ejemplos reales de diferentes aplicaciones web, en los que se identificaron los problemas relacionados puntualmente con la ejecución de los procesos de negocio embebidos en ellas. Detectados los problemas se sugirió un conjunto de *refactorings* que, cuando aplicados, promueven mejorías en los procesos de negocio de las aplicaciones estudiadas, optimizando así características como la usabilidad, sin modificar la esencia de los procesos modificados, ni los requerimientos de la página.

Los procesos de negocio fueron modelados utilizando la metodología UWE que, por basarse en el lenguaje UML, es de fácil comprensión y entendimiento, sin mencionar la amplia aceptación entre las herramientas CASE. La utilización de UWE y de sus estereotipos resultó beneficioso para este trabajo por ser una metodología sencilla que al mismo tiempo traduce fácilmente los diferentes modelos de la aplicación web, en particular el impacto de los procesos de negocio en esos modelos, ya que, como pocas metodologías, UWE ofrece la ventaja de un modelo exclusivo para el modelado de los proceso de negocio simplificando la etapa de diseño y definición de los mismos.

En el transcurso de esta tesis se propuso un total de 17 *refactorings* con en el objetivo de optimizar la calidad de los proceso de negocio en determinadas aplicaciones web, tornándolos más usables y eficientes. Mediante ellos se demostró, a través de ejemplos reales, que la introducción de pequeños cambios en la estructura, en la navegación y presentación de los procesos pueden traer mejoras significativas en la experiencia del usuario, haciendo que les sea más fácil e intuitivo la ejecución los procesos de negocio disponibles en las páginas.

Por otra parte, por una cuestión organizativa de la presentación del trabajo, los *refactorings* se clasificaron de acuerdo con el modelo que impactaban, aunque se demostró que los cambios realizados en un modelo

pueden estar también estar relacionados con los otros modelos de la aplicación web y de hecho afectarlos. Por ese motivo se identificó, asimismo, la relación existente entre los *refactorings* propuestos e incluso se incorporaron sugerencias de combinación entre ellos con el propósito de optimizar los resultados.

En el análisis de las aplicaciones web realizado, se identificaron problemas, los llamados *bad smells,* que motivaron la aplicación de los *refactorings*. Los *refactorings* propuestos tienen el objetivo de eliminar o minimizar los problemas identificados, no obstante, al emplear los cambios sugeridos se pudo observar que la aplicación de cada *refactoring* trajo un conjunto de mejoras que están relacionadas a diversos otros problemas de la aplicación y no solamente al *bad smell* identificado inicialmente. Al finalizar la aplicación de los mismos y con el análisis de los resultados obtenidos, se pudo percibir un conjunto de mejoras comunes logradas en las distintas aplicaciones analizadas, las que se pueden sintetizar de la siguiente manera:

- **Reducción del número de pasos del proceso analizado:** Cuanto más largo es un proceso de negocio, más costoso para el usuario es ejecutarlo. En función de este principio se evaluaron cuidadosamente todos los pasos existentes en un proceso con el objetivo de mantener apenas los necesarios para la conclusión del proceso. El análisis de la estructura del proceso reveló oportunidades para agrupar actividades o identificar repeticiones, por ejemplo, lo que dio lugar a una reducción en la cantidad de pasos, modificación que se estimó de gran valor porque incide directamente en la merma del tiempo requerido para la ejecución de los procesos.

- **Reducción del tiempo de ejecución del proceso:** Si un proceso de negocio puede cumplir su objetivo en menos tiempo, será más eficiente y más beneficioso para el usuario. Partiendo de esa premisa y con el objetivo de evitar frustraciones innecesarias en los usuarios, se estudiaron oportunidades de cambios que aceleran la ejecución del proceso permitiendo que el usuario llegara más rápidamente a su objetivo. En tal sentido resultó de mucha utilidad eliminar tareas innecesarias, pasos duplicados, e implementar cambios que mejoraran la percepción del usuario respecto a la sencillez del proceso y que al mismo tiempo acrecentaran la eficiencia en la conclusión del mismo.

- Incremento de la confianza en la aplicación: La confianza es una de las premisas de mayor importancia en el ambiente de las aplicaciones web en general y en las páginas de *E-Commerce* y *Home Banking* en particular. Si el usuario no confía en la aplicación usada, difícilmente realizará cualquier operación en la misma. En función de aumentar la confiabilidad de las páginas estudiadas se proveyó mecanismos que permitan reducir la posibilidades de error durante el transcurso del proceso, o bien la posibilidad de comprobar los datos ingresados antes de presionar "Confirmar". Aunque que pequeños, estos cambios hacen la diferencia respecto de la confiabilidad de la página al introducir en el proceso un rango de seguridad extra.

- Aumento de la eficiencia de la aplicación: Se estima que, en los procesos de negocio, cuanto más eficiente es la aplicación, más trabajo será realizado por el sistema y menos trabajo a ser realizado por los usuarios. Asimismo se consideró que una aplicación eficiente es aquella que ayuda al usuario a concluir el proceso de negocio en la página, realizando actividades internamente o implementando ayudas que aceleraran su ejecución. Además de resultar más eficiente, se estima también que la página se torna más amigable ya que hay poca demanda de esfuerzo del usuario.

- Aumento de la satisfacción del usuario: La satisfacción con el servicio y con el sitio es un requisito indispensable en el ambiente web para que el usuario permanezca en página y decida usarla. Su desatención es sinónimo de abandono de la aplicación. Con este objetivo se pensaron en función de las páginas

estudiadas, estrategias que facilitaron encontrar, navegar y/o ejecutar procesos y se incluyeron mecanismos de ayuda que facilitaron su ejecución o mantuvieran al usuario informado respecto de las acciones realizadas durante el mismo. Estos mecanismos no solo inciden en la satisfacción del usuario sino que aumentan la usabilidad de la página.

- Reducciones de repeticiones en el proceso: La eliminación de repeticiones por deficiencia del diseño de las aplicaciones trae aparejado un incremento en la eficiencia de la página, incide en el buen uso del tiempo y reduce la frustración al que se ve expuesto el usuario después de repetir diversas veces el mismo paso. Ejemplo de estas repeticiones pueden constituir la búsqueda de pasajes *online* o la introducción de un error durante la ejecución de un proceso, en una página que ofrece como única alternativa la opción de reiniciar el proceso desde cero. Como se dijo en el ítem "Reducción del número de pasos del proceso analizado", estas repeticiones pueden conducir al desinterés y posterior abandono del proceso de negocio. En función este tipo de situaciones se incluyeron filtros o la posibilidad de retroceder en determinadas etapas del proceso. La aplicación de tales medidas redunda no sólo en la realización más rápida de los objetivos, sino que incrementa la amigabilidad de la página.

- Eliminación/Reducción de inconsistencias en el proceso: La reducción de inconsistencias al interior de los procesos de negocio contribuye a la satisfacción inmediata del cliente y la maximización de los negocios con altos resultados para las empresas. No obstante esto, en ocasiones los procesos de negocio resultan rígidos y no permiten que los usuarios realicen operaciones básicas como cancelar o retroceder en pasos ya ejecutados. Esta falta de opciones ofrecidas por la aplicación, hacen que la persona recurra a opciones no deseables, como la utilización de los botones de navegación del browser, para poder retroceder en las páginas a fin de cancelar o corregir ciertos datos en alguna etapa previa. Muchos usuarios desconocen que el uso de este mecanismo puede generar inconsistencias en el estado interno del proceso pudiendo ocasionar resultados no esperados o errores en la aplicación. La introducción de vías alternativas de retroceso aumenta la conformidad frente a la ejecución y eleva el grado de satisfacción de los clientes.

- Menos esfuerzo para el usuario ejecutar el proceso: Implementar determinados cambios en los procesos de negocio facilitan la vida de los usuarios en el momento de ejecutar el conjunto de tareas necesarios para su conclusión. Una aplicación eficiente ofrece mecanismos para agilizar la conclusión del proceso de negocio haciendo que el usuario tenga que realizar el mínimo de tareas necesarias, siendo intuitivo y lo más fácil posible.

- Eliminación de Redundancias: Pasos y *processLinks* repetidos en el transcurso del proceso generan una redundancia que puede y debe ser eliminada. Como dicho anteriormente, tiempo de ejecución, eficiencia de la aplicación y esfuerzo del usuario son características importantes en el contexto de los procesos de negocio y la duplicación no aporta beneficios en estos rubros.

- Reducción del riesgo de errores en la aplicación: La ejecución de proceso de negocio muchas veces puede resultar en una tarea demandante y que consume mucho tiempo. Debido a la complejidad de algunos procesos de negocio, es común que los usuarios se queden confusos en relación a los pasos ejecutados o no sepan como concluir una tarea. Para evitar este problema, la aplicación debe ofrecer el máximo de ayuda en relación a las actividades que deben ser ejecutadas y ofrecer la posibilidad de retroceder en el proceso con el intuito de conferir o cambiar lo que está por ser concluido, de esta forma el usuario puede confiar en la aplicación y chequear que esta todo correcto antes de concluir con el proceso.

- Aumento de la flexibilidad en el proceso: La inflexibilidad en los procesos es una característica que va en contra la usabilidad, ya que condiciona al usuario impidiendo el cambio de idea o la no realización de algunas operaciones básicas. Los procesos deben mantener el usuario focalizado en la ejecución de las tareas y al mismo tiempo deben considerar que la persona puede cambiar de idea o puede cometer errores durante la realización del las mismas y por consiguiente querer corregirlos, por lo tanto no es raro que el usuario quiera desistir de ejecutar un proceso o quiera volver a pasos anteriores para cambiar algunos de los datos ingresados. En función de esto las páginas deben ser flexibles a punto de permitir determinadas tareas básicas y evitar que los usuarios recurran a medios no deseables como sería recurrir al botón de navegación del browser, como se mencionó anteriormente. Tener presente el requerimiento de flexibilidad es trabajar en función de la amigabilidad y usabilidad de la aplicación.

- Mejora en el acceso al proceso: Los proceso de negocio son muy comunes y están distribuidos por todas las aplicaciones web, sin embargo algunos son normalmente utilizados con mayor frecuencia por los usuarios y por lo tanto merecen ser más destacados en la página. Con el objetivo de mejorar el acceso a los procesos más frecuentes, es necesario analizar la navegación de la aplicación a fin de detectar la manera de facilitar que ese proceso sea más accesible al cliente. Por ejemplo, la compra de pasajes online sería mucho más difícil si el proceso de búsqueda de billetes no estuviera bastante visible en la *Homepage* de las compañías aéreas, haciendo que los usuarios puedan disparar y ejecutar el proceso de búsqueda fácil y rápidamente. Este tipo de estrategias enriquece la amigabilidad y por lo tanto genera más usabilidad.

- Aumento de informaciones referentes al proceso: Debido a la complejidad existente en muchos procesos de negocio, es importante que las aplicaciones web mantengan al usuario informado respecto de lo que están ejecutando, evitando de esta manera errores y posibles confusiones. Con informaciones disponibles, los usuarios no se sienten perdidos en el proceso; además conocer lo que está ejecutando y lo que aún le falta por realizar aumenta la seguridad del individuo. Cuanto más conocimiento tengan las personas respecto de lo que están ejecutando, más confianza sentirán hacia la aplicación web y más seguridad sobre su propia actuación en ella, al observar que no está cometiendo errores.

- Reducción del abandono de la aplicación: Repetición de pasos, procesos de registraciones anticipados, procesos muy largos y complicados, confusión en el proceso, son muchos los factores que llevan un usuario a abandonar una aplicación web. Muchos pueden ser evitados cuando los procesos son optimizados trayendo una mejor experiencia al usuario.

Muchos de los beneficios listados fueron logrados en más de un *refactoring.* Las mejoras en común pueden ser vistas en más detalles en cuadro presentado en el Anexo B.

8.1. Trabajos Futuros

Los beneficios logrados después de la aplicación de los *refactorings* sugeridos se identificaron a través de un análisis de las aplicaciones web, sin embargo sería ventajosa la creación de un método automático para evaluar el nivel de mejoras logrado en los procesos de negocio una vez aplicados los *refactorings*. En tal sentido un área de vacancia para investigaciones posteriores referiría al estudio de las necesidades que darían sustento al mencionado método, el diseño del mismo, su automatización y posterior puesta a prueba.

Otra área de estudio refiere al hecho de extender el catálogo de *refactorings* para mejorar los procesos de negocio de las aplicaciones web identificando nuevas oportunidades de mejoras en aplicaciones web similares a las analizadas en esta tesis. Actualmente el catalogo incluye 17 *refactorings.*

Asimismo otra posibilidad de investigación se centraría en extender el alcance de las aplicaciones web analizadas, incluyendo diferentes tipos de las páginas de comercio electrónico y de páginas de *Home Banking* que fueron el foco de este trabajo, o dicho de otra manera se trataría de ampliar el espectro del estudio actual diversificando las páginas sobres las cuales se basó esta tesis.

Por último otra gran área de investigación y experimentación referiría a la creación de herramientas para automatizar y acelerar los procesos de *refactoring* en las aplicaciones web seleccionadas, efectuando los procesos de modelación, contrastación y rediseño de los mismos si hubiese lugar, a fin de garantizar un producto bien estandarizado.

9.BIBLIOGRAFIA

[1] DISTANTE, D.; TILLEY, S. and HUANG, S. “Web Site Evolution via Transaction Reengineering”. In Proceedings of the 6th IEEE International Workshop on Web Site Evolution (WSE 2004: September 11, 2004; Chicago, IL). Los Alamitos, CA: IEEE Computer Society Press (2004).

[2] DISTANTE, D.; ROSSI, G.; CANFORA, G and TILLEY, S. “A Comprehensive Design Model for Integrating Business Processes in Web Applications”. International Journal of Web Engineering and Technology (IJWET). Interscience Publishers, Vol. 3, Nº. 1 (2007).

[3] KOCH, N.; KRAUS, A.; CACHERO, C. and MELIÀ S. “Modeling Web Business Processes with OO-H and UWE”. 3rd International Workshop on Web-Oriented Software Technology (IWWOST 2003, Oviedo, Spain) (2003).

[4] ROSSI, G.; SCHMID, H. and LYARDET, F. “Engineering Business Processes in Web Applications: Modeling and Navigation issues”. 3rd International Workshop on Web Oriented Software Technology (IWWOST 2003: Oviedo, Spain) (2003).

[5] ROSSI, G. and SCHMID, H. “Modeling and Designing Processes in E-Commerce Applications”. IEEE Internet Computing, January/February (2004).

[6] BARESI, L.; DENARO, G.; MAINETTI, L. and PAOLINI, P. "Assertions to better specify the Amazon Bug". Proceedings of the 14th International Conference of Software Engineering and Knowledge Engineering (SEKE 2002), Ischia, Italy, 15–19 July (2002).

[7] GARRIDO, A.; ROSSI, G. and DISTANTE, D. “Refactoring for Usability in Web Applications”. IEEE Software, May/June 2011, pp. 31-38, IEEE Computer Society (2011).

[8] SCHMID, H. and ROSSI, G. “Designing Business Processes in E-commerce Applications”. 3rd International Conference on Electronic Commerce and Web Technologies (ECWEB03: Aix en Province, France) (2002).

[9] DISTANTE, D.; PARVEEN, T. and TILLEY, S. "Towards a technique for reverse engineering web transactions from a user’s perspective". Proceedings of the 12th IEEE International Workshop on Program Comprehension (IWPC 2004), Bari, Italy, Los Alamitos, CA: IEEE CS Press, 24–26 June (2004).

[10] KRUG, S. “Don’t Make me Think – A Common Sense Approach to Web Usability”. Indianapolis, IN: New Riders (2000).

[11] NIELSEN, J. "useit.com: Jakob Nielsen's Website". Disponible en: http://www.useit.com

[12] FOWLER, M. “Refactoring: Improving the Design of Existing Code”. Addison-Wesley (1999).

[13] MENS, T. and TOURWÉ, T. "A Survey of Software Refactoring". IEEE Transactions on Software Engineering, vol. 30, n° 2, February (2004).

[14] NIELSEN, J. “Designing Web Usability”. New Riders Publishing, California, USA (2000).

[15] GARRIDO, A.; ROSSI, G. and DISTANTE, D. “Model Refactoring in Web Applications”. In Proceedings of the 9th IEEE International Symposium on Web Site Evolution (WSE'07), Paris, France, October (2007).

[16] OLSINA, L.; ROSSI, G.; GARRIDO, A.; CANFORA, G and DISTANTE, D. “Web Application Evaluation and Refactoring: A Quality-Oriented Improvement Approach”. Journal on Web Engineering, Vol.7, pp. 258-280 (2008).

[17] KOCH, N. - LMU - Ludwig-Maximilians-Universität München – “UWE – UML-based Web Engineering”. Disponible en: http://uwe.pst.ifi.lmu.de/index.html

[18] GARRIDO, A.; ROSSI,G. and DISTANTE,D. “Systematic Improvement of Web Applications Design”. Journal of Web Engineering, Special Issue on Web Application Evolutions, Vol. 8, Nº4, 371-404. Rinton Press (2009).

[19] WINCKLER, M. and SOARES PIMENTA, M. “Avaliação de Usabilidade de Sites Web” . In X Escola de Informática da SBC-Sul. ERI 2002 (2002).

[20] KOCH, N.; KNAPP, A.; ZHANG, G. and BAUMEISTER, H. “UML-Based Web Engineering: An Approach based on Standards” (book chapter). In “Web Engineering: Modeling and Implementing Web Applications”. ROSSI,G.; PASTOR, O.; SCHWABE, D. and OLSINA,L. (Eds.), chapter 7, 157-191, ©Springer, HCI (2008).

[21] KROIß, C. and KOCH, N. “UWE Metamodel and Profile: User Guide and Reference”. LMU Technical Report (2008).

[22] KOCH,N. and BUSCH, M. “MagicUWE - A CASE Tool Plugin for Modeling Web Applications”. En Proceedings *9th Int. Conf. Web Engineering (ICWE'09)*, LNCS, volume 5648, pages 505-508. ©Springer, Berlin (2009).

[23] KOCH, N. and KRAUS, A. “The Expressive Power of UML-based Web Engineering”. At Second International Workshop on Web-oriented Software Technology (IWWOST02) at ECOOP02, Malaga, Spain (2002).

[24] WEBBER, B. and REICHERT, M. “Refactoring Process Model in Large Process Repositories” In: CAiSE’08. LNCS 5074, Montpellier 124–139 (2008)

[25] GALLINA, B.; GUELFI, N. and MAMMAR, A. "Structuring business nested processes using UML 2.0 activity diagrams and translating into XPDL". In XML4BPN XML Integration and Transformation for Business Process Management, pages 281–296. GITO-Verlag (2006).

[26] KOCH, N. “UWE – UML-based Web Engineering Presentation”. Almeria, June (2010).

[27] SCHAFER, J. B.; KONSTAN, J. and RIEDL, J. “E-Commerce Recommendation Applications” Journal of Data Mining and Knowledge Discovery 5: 115–152 (2000)

[28] ROSSI, G.; SCHWABE, D. and LYARDET, F. “Patterns for E-commerce Applications”, Proceedings of EuroPLoP00, Germany, July (2000).

[29] TILLEY, S. and HUANG, S. “Evaluating the Reverse Engineering Capabilities of Web Tools for Understanding Site Content and Structure: A Case Study.” Proceedings of the 23rd International Conference

on Software Engineering (ICSE 2001: May 12-19, 2001; Toronto, Canada), pp. 514-523. Los Alamitos, CA: IEEE Computer Society Press (2001).

[30] OPDYKE, W., JOHNSON R. "Creating Abstract Superclasses by Refactoring." In Proceedings of the 1993 ACM Conference on Computer Science (CSC 93), ACM Press, pp. 66–73 (1993).

[31] BOGER, M., STURM T., and FRAGEMANN P. "Refactoring Browser for UML." Objects, Components, Architectures, Services, and Applications for a Networked World, Lecture Notes in Computer Science 2591, pp. 366–377, Springer-Verlag Berlin Heidelberg (2003).

[32] AMBLER, S.W., SADALAGE, P.J. "Refactoring Databases: Evolutionary Database Design." Addison-Wesley (2006).

[33] HAROLD, E.R. "Refactoring HTML: Improving the Design of Existing Web Applications." Addison-Wesley (2008).

[34] ROSSI, G.H., SCHMID, H.A., and LYARDET, F. "Customizing Business Processes in Web Applications." In 4th International Conference on E-Commerce and Web Technologies (2003).

[35] DISTANTE, D., TILLEY, S. "Conceptual Modeling of Web Application Transactions: Towards a Revised and Extended Version of the UWA Transaction Design Model." In Proceedings of the 11th International Multi-Media Modelling Conference (MMM2005: January 12-14, 2005; Melbourne, Australia). Los Alamitos, CA: IEEE Computer Society Press (2005).

[36] CERI, S., FRATERNALI, P., and BONGIO, A. "Web modeling language (WebML): a modeling language for designing web sites." In Proc WWW9 Conference.(also in Computer Networks, 33 (2000), Amsterdam, NL (2000).

[37] UWA Consortium. "Ubiquitous web applications." In Proceedings of the eBusiness and eWork Conference e2002, Prague, Czech Republic (2002).

[38] DE TROYER, O. and LEUNE, C." WSDM: A user-centered design method for web sites." Published in Computer Networks and ISDN systems. Proceedings of the 7th International World Wide Web Conference, Elsevier, pp. 85 - 94, (1998)

[39] BONGIO, A., CERI, S., FRATERNALI, P., and MAURINO, A. "Modeling data entry and operations in webml." In WebDB (Informal proceedings) 2000, pages 87–92 (2000).

[40] PASTOR, O., ABRAHO, S.M., and FONS, J. "An object-oriented approach to automate web applications development." In Proceedings of EC-Web, Munich, Germany (2001).

[41] ROSSI, G., SCHWABE, D. and GUIMARÃES, R. M. "Designing Personalized Web Applications." In World Wide Web. Pages 275–284 (2001).

[42] SCHWABE, D., ROSSI, G., ESMERALDO, L. and LYARDET, F. "Engineering Web Applications for Reuse." IEEE Multimedia. Special Issue on Web Engineering, 01-03, 20–31 (2001).

[43] CABOT, J. and GOMEZ, C. “A Catalogue of Refactorings for Navigation Models”, Proc. 8th International Conference on Web Engineering (ICWE 08), Yorktown Heights, New York, (2008).

[44] OPDYKE, W. "Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameworks" University of Illinois (1992).

[45] JOHNSON, R. and OPDYKE, W. "Refactoring and aggregation." In Object Technologies for Advanced Software, First JSSST International Symposium, volume 742, pages 264-278. Springer-Verlag (1993).

[46] MENS, T., DEMEYER, S. and JANSSENS, D. ""Formalizing Behaviour Preserving Program Transformation" presented at 1st International Conference on Graph Transformation (2002).

[47] BRAMBILLA, M., CERI, S., FRATERNALI, P. and MANOLESCU, I. "Process Modeling in Web Applications." ACM Transactions on Software Engineering and Methodology (TOSEM) (2006).

[48] TORRES, V., PELECHANO, V. "Building Business Process Driven Web Applications." In Proceedings of Business Process Management 2006, pp. 322-337, Springer Berlin / Heidelberg (2006).

[49] KOCH, N., KRAUS, A., CACHERO C., MELIÁ, S." Integration of Business Processes in Web Applications." Journal of Web Engineering, Vol. 3, Nº1, 022-049. Rinton Press (2004).

[50] BARESI, L., GARZOTTO, F., PAOLINI, P. "Extending UML for Modeling Web Applications" In 34th Hawaii International Conference on Systems Sciences (2001).

[51] MARKOPOULOS, P. "Supporting Interaction Design with UML." In the TUPIS Workshop at the UML'2000. (2000).

[52] NUNES, N., CUNHA, J. "Towards a UML Profile for Interaction Design: The Wisdom approach." In the 3rd International Conference on the Unified Modeling Language (UML´2000), A. Evans and S. Kent (Eds.). LNCS 1939, Springer Verlag, 100-116. (2000).

[53] SCHWABE, D. and ROSSI, G. “An Object-Oriented Approach to Web-Based Application Design”. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October (1998).

[54] VANHATALO, J., VÖLZER, H., LEYMANN, F. and MOSER, S. "Automatic workflow graph refactoring and completion." In Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC '08), Lecture Notes in Computer Science, 5364 LNCS, pp. 100-115, Springer, Heidelberg (2008).

[55] WEBER, B., REICHERT, M. and RINDERLE-MA, S. "Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems." Data & Knowledge Engineering 66 (2008) pp. 438–466, Elsevier (2008).

[56] TILLEY, S., DISTANTE, D. and HUANG, S. “Design Recovery of Web Application Transactions.” Submitted to The 11th IEEE Working Conference on Reverse Engineering (WCRE 2004: Nov. 9-12, Delft, The Netherlands). June. (2004).

[57] BARESI, L., GARZOTTO, F. and PAOLINI, P. "From Web Sites to Web Applications: New issues for Conceptual Modeling." In Proceedings of the International Workshop on The World Wide Web and Conceptual Modeling, co-located with the 19th International Conference on Conceptual Modeling. (2000).

[58] DIJKMAN, R., GFELLER, B., KÜSTER J. and VÖLZER, H. "Identifying refactoring opportunities in process model repositories." Information and Software Technology 53 pp. 937–948, Elsevier (2011).

[59] KÜSTER, J. M., KOEHLER J. and RYNDINA, K. "Improving Business Process Models with Reference Models in Business-Driven Development." Proceedings of the International Business Process Management Workshops 2006, Lecture Notes in Computer Science, Volume 4103/2006, pp. 35-44, Springer-Verlag Berlin Heidelberg (2006).

[60] GRIGORI, D., CASATI, F., DAYAL, U. and SHAN, M.-C. "Improving Business Process Quality through Exception Understanding, Prediction, and Prevention." In Proceedings of the 27th Very Large Data Base Conference (VLDB '01), Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (2001).

[61] SPOOL, J.M., SCANLON, T., SCHROEDER, W., SNYDER, C. and DEANGELO, T. "Web Site Usability: A Designer's Guide" User Interface Engineering (1997)

[62] REYNOLDS, J. "The Complete E-Commerce Book - Design, Build & Maintain a Successful Web-based Business" CMP Books - 2nd Edition (2004).

[63] INCE, D. "Developing Distributed and E-Commerce Applications" Pearson Education Limited - 2nd Edition (2004).

[64] NIELSEN,J.; MOLICH, R.; SNYDER, C. and FARRELL, S. " E-Commerce User Experience" Nielsen Norman Group. USA (2001).

[65] OPDYKE, W. "Refactoring Object-Oriented Frameworks." PhD Thesis. University of Illinois at Urbana-Champaign (1992).

[66] ROBERTS, D. "Practical Analysis for Refactoring." Tesis doctoral. Universidad de Illinois (1999).

[67] ROBERTS, D., BRANT, J. y JOHNSON, R. "A Refactoring Tool for Smalltalk, Theory and Practice of Object Systems." Vol. 3, Issue 4. Editor: John Wiley & Sons, Inc. Páginas: 253-263 (1997).

[68] GONÇALVES, R. F., GAVA, V. L., PESSÔA, M. S. P. and SPINOLA, M. M. "Uma Proposta de Processo de Produção de Aplicações Web." Articulo publicado en la Revista Produção, v. 15, n. 3, Set./Dez. (2005).

[69] CONALLEN, J. "Modeling Web Application Architetures with UML" Communications of ACM, v. 42, n. 10, October (1999).

[70] PAULA FILHO, W.P. "Engenharia de Software". Rio de Janeiro: LTC (2003).

[71] PRESSMAN, R. "Software Engineering: A Practitioner's Approach". 3rd Edition. McGraw-Hill, USA. ISBN 0-07-050814-3 (1992).

[72] PERZEL, K. and KANE, D. "Usability Patterns for Applications on the World Wide Web", in Proceedings of PLoP'1999 (1999).

[73] OFFUTT, J. "Quality Attributes of Web Software Applications." IEEE Software, v.19, n. 2, pp. 25-32 (2002).

[74] SHAH, A. and SHOAIB, M. "Sources of Irrelevancy in Information Retrieval Systems." International Conference on Software Engineering Research & Practice, Las Vegas, USA, Pp 877-883 (2005).

[75] CONTE, T. and TRAVASSOS, G.H. " Técnica de Inspeção de Usabilidade Baseada em Perspectivas de

Projeto WEB Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil. En el VIII Simposio Brasileiro de Qualidade de Software (2009).

[76] ITAKURA, F. T. and VERGILIO, S.R. " TOWABE – Uma Ferramenta para Avaliação de Usabilidade em Aplicações para Web." En el XVI Simposio Brasileiro de Engenharia de Software (SBES 2002), Gramado, RS, Brasil, Outubro (2002).

[77] RICCA, F. and TONELLA, P."Program Transformations for Web Application Restructuring." In Web Engineering: Principles and Techniques. Woojong Suh (eds.), chapter XI, pages 242-260. (2005).

[78] CHAFFEY, D. "E-Business and E-Commerce Management." Person Education Limited. Second Edition (2004).

[79] KALAKOTA, R. and WHINSTON, A.B. "Electronic Commerce: A Manager´s Guide." Boston: Addison-Wesley. (1997).

[80] LOSHIN, P. and VACCA, J. "Electronic Commerce." Charles River Media, Inc. Fourth Edition (2004).

[81] JAISWAL, M.P and KUMAR, V.G. "E-Business Models: Success Strategies." Excel Books. First Edition. New Delhi (2002).

[82] AWAD, E. M. "Electronic Commerce: From Vision to Fulfillment." Pearson Education Inc. Third Edition. (2207).

[83] Keynote.com. Online banking critical to bank selection and brand perception. Press release January, 6th (2005).

[84] MANNAN, M. and VAN OORSCHOT, P.C. "Security and usability: the gap in real-world online banking." In NSPW ’07: Proceedings of the 2007 Workshop on New Security Paradigms, pages 1–14, New York, NY, USA, ACM (2008).

[85] DISTANTE, D., GARRIDO, A., CAMELIER-CARVAJAL, J., GIANDINI, R. and ROSSI, G. "Business Processes Refactoring to Improve Usability in E-Commerce Applications" in Electronic Commerce Research Journal (2013)

ANEXO A: METAMODELO UWE

CLASES DEL METAMODELO DE NAVEGACIÓN

- ***NavigationNode***

  Un *NavigationNode* puede ser cualquier nodo del gráfico de navegación. Cuando un nodo es alcanzado durante la navegación, el usuario visualiza determinada información que puede llevarle a realizar otras acciones en la aplicación. Un nodo de navegación no significa ser necesariamente una página de la aplicación web, pero muchas veces lo sea. Lo que es mostrado en la página es definido en el modelo de presentación.

  Los nodos poseen asociaciones como los *inGoing* links, que son el conjunto de links que llevan al nodo, y los *outGoing* links, que son el conjunto de links que son originados desde el nodo. Aparte las asociaciones, los *NavigationNodes* también traen atributos como:

  - {*isLandmark*}: Indica que el nodo es alcanzable por todos los otros nodos del gráfico de navegación. Todos los otros nodos poseen un link para el nodo marcado como *landmark*;
  - {*IsHome*}: Identifica si el nodo es el origen del gráfico de navegación, siendo el nodo inicial de la aplicación web (nodo que no posee *inGoing* links). El nodo identificado como {*IsHome*} debe tener su valor como "*true*". Apenas un único nodo de navegación puede tener este atributo marcado como "*true*".

- **Link**

  Un link es una conexión del gráfico de navegación usado para conectar dos nodos. En el modelo de presentación es definido si los dos nodos que son conectados por un link son mostrados al mismo tiempo o si el usuario tiene que clicar en una ancora para navegar de un link a otro (Ver sesión 3.3.3).

  El link de origen es llamado de "*source"* y el link de destino de "*target"*.

- ***NavigationClass***

  Representa un nodo navegable y establece la conexión entre el modelo de contenido y el modelo de navegación. Especifica el nodo visitado por el usuario o por el sistema a través de la navegación. Una *navigationClass* que está asociada con una clase del modelo de contenido (*contentClass*) representa el contenido de una instancia de esa clase [21]. La *navigationClass* posee el mismo nombre de la *contentClass* que representa.

  El estereotipo utilizado por el *NaviagationClass* es: 

Un ejemplo de una *navigationClass* y de cómo es representada puede ser visto en la Figura 54 en la continuación.

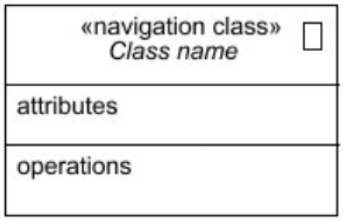


Figura 54:Ejemplo de *navigationClass*

- ***NavigationLink***

  Un *navigationLink* es un link que conecta cualquier tipo de nodo con excepción de los <*ProcessClass*>. Especifica un hipervínculo usado para acceder el nodo definido como "*target*" a través del nodo definido como "*source*". Representan posibles pasos a ser seguidos por el usuario [20].

  La Figura 55 ilustra que el menú posee un *navigationLink* que lleva a la *navigationClass* Contactos. En el caso que el nodo de origen o el nodo de destino sea un *processClass*, no se podría usar un *navigationLink* y si un *ProcessLink*.

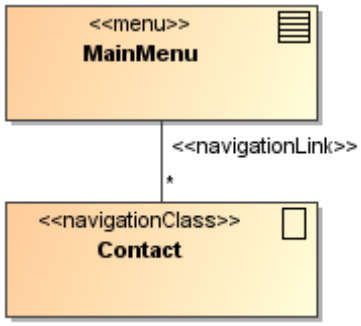


Figura 55: Ejemplo de *navigationLink*.

- **Menu**

  Los menus son usados para estructurar los *outGoing* links desde un nodo, para proveer un camino de navegación y manejar alternativas de navegación. Los menus son adicionados a cada *navigationClass* que tiene más de un *outGoing* link [20]. Consiste en un conjunto de links de elementos heterogéneos como los *guideTour*, *indexes*, *queries* e incluso otros menus [26]. El menu es representado en UWE por el siguiente estereotipo: menu y puede su representacion puede ser vista en la Figura 56 abajo.

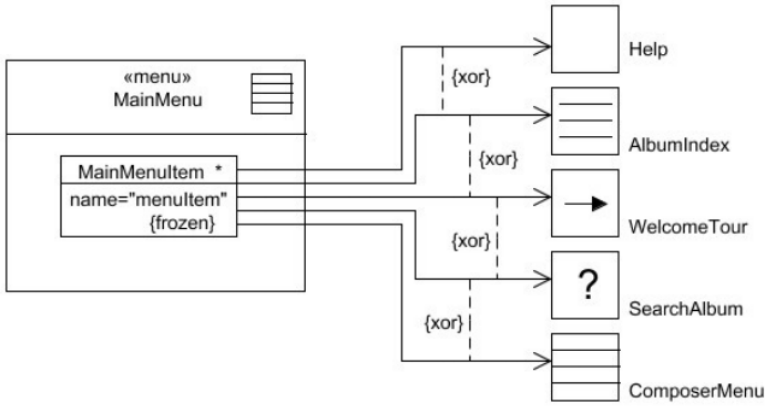


Figura 56: Ejemplo de menu

- ***ExternalNode***

  Los *externalNode* son nodos que no pertenecen a la estructura de la aplicación actual. Son nodos afuera de la aplicación web. Poseen una *locationExpression* que especifica la URL del nodo externo [21].

  El estereotipo usado para representar un *ExternalNode* es: externalNode

- ***Index***

  Representa el acceso directo para todas las instancias a través de una lista que contiene un atajo a todos los elementos. A través de esta lista el usuario puede seleccionar cualquier atajo para seguir navegando por la página. Permite seleccionar una clase de contenido de un conjunto de instancias que fueron compiladas durante una navegación previa. Cada ítem del *index* es un objeto que posee un nombre y un link para una instancia de la *NavigationClass*. El ítem elegido se torna el *navigationClass* que sucede el *index* [21].

  El estereotipo usado para representar el *Index* es: index y un ejemplo puede ser visualizado en la Figura 57.

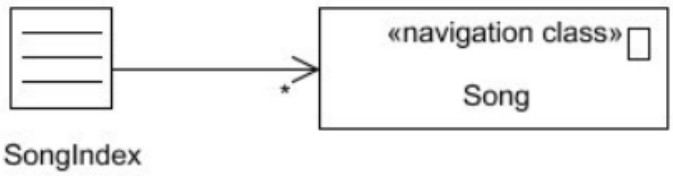


Figura 57: Ejemplo de *Index*

Query

- Una *query* es usada para recuperar datos de una base de datos. En diferencia al *index*, una *query* no trae sus resultados a través de un predecesor en el *navigation path* y si de una base de datos [21].

  Una *query* puede requerir que el usuario ingrese parámetros para filtrar la búsqueda como por ejemplo, filtrar músicas que poseen la palabra “Amor” en el nombre, como mostrado en la Figura 58. La opción de filtro es definida en el *filterExpression*. En el caso de ofrecer opciones para filtrar, el modelo de presentación debe proveer una interface que permita que el usuario pueda ingresar el parámetro a ser utilizado en el filtro. (Ver más detalles en la sección 3.3.3).

Si la *query* no requiere la entrada de parámetros, es ejecutada automáticamente cuando alcanzada en el gráfico de navegación. Un ejemplo sería una *query* que muestra las 10 películas más vistas en el año.

El estereotipo usado para representar una *Query* en UWE es: 

Figura 58: Ejemplo de *Query*

- ***GuidedTour***

  Provee un acceso secuencial para instancias del *navigationClass*. Partiendo de un conjunto de instancias que poseen *outGoing* links para un *navigationClass*. El usuario puede navegar para atrás y adelante seleccionando una instancia por vez. El orden en que las instancias son visitadas es especificada usando un filtro, el *sortExpression*[21].

  El estereotipo usado para representar un *GuidedTour* en UWE es: 

y la Figura 59 ilustra como los *GuidedTours* son representados.

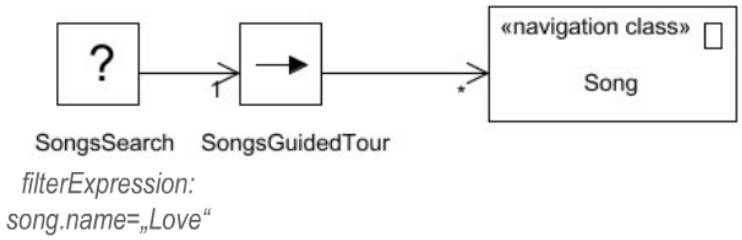
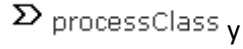


Figura 59: Ejemplo de guidedTour

- ***ProcessClass***

  Cuando la aplicación web contiene procesos de negocio, eses procesos deben ser representados en el modelo de navegación a través de los *processClass*. Los *processClass* representan un proceso no definido que guía el usuario a través de pasos en la aplicación web. Los links que llevan a un *processClass* son llamados de *processLinks*. Los detalles referentes a los pasos internos del proceso no son mostrado en el modelo de navegación y serán detallados en el Modelo de Proceso (Ver sección 3.2.1.4).

  El estereotipo usado para representar un *processClass* es: 

y un ejemplo puede ser visto en la Figura 60 ilustrada abajo:

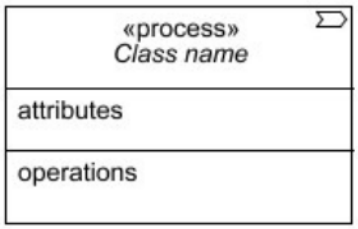


Figura 60: Ejemplo *processClass*

CLASES DEL METAMODELO DE PRESENTACIÓN

- ***PresentationModel***
  El *presentationModel* refina el modelo de navegación de una aplicación web representando el *layout* de la página.

- ***PresentationClass***
  Representa el fragmento lógico de la página y es compuesta de los elementos de la interface del usuario que son exhibidos en la aplicación. Cada <*presentationClass*> está asociada directamente con el <*navigationClass*> correspondiente en el diagrama de navegación o a un <*processClass*> definiendo por lo tanto la presentación de este elemento en particular.

- ***PresentationPage***
  Un *presentationPage* tiene la misma semántica que un *presentationGroup* con la diferencia de que esta puede no estar incluida dentro de un *presentationGroup*. A diferencia del *presentationGroup*, una página no tiene que estar asociada con un *navigationNode* si incluye por lo menos un *presentationGroup* que posee referencia para el modelo de navegación. La restricción es que una *presentationPage* no puede estar adentro de un *presentationGroup*.
  Son definidas como una especialización del *presentationGroup* y son usadas para modelar la menor unidad de presentación que puede ser exhibida al usuario.
  El siguiente estereotipo es usado para representar una *presentationPage*: presentationPage

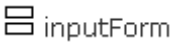
- ***PresentationAlternatives***
  PresentationAlternatives son utilizados para modelar un conjunto de *presentationGroup* que son presentados adentro del mismo grupo, pero solamente un *presentationGroup* es visible al usuario.
  El estereotipo usado para representar un *PresentationAlternative* es: presentationAlternatives

- ***PresentationGroup***
  Un *presentationGroup* es utilizado para definir un conjunto de elementos de presentación que son mostrados alternativamente en una misma área de la página, dependiendo de la navegación. Cuando un *navigationNode* es alcanzado, el contenido del *presentationGroup* reemplaza el contenido del *presentationGroup* mostrado en el momento. Se puede determinar cuál es el *presentationGroup* visible por defecto y cuál es el *presentationGroup* mostrado cuando ninguno de

los *navigationNodes* asociados fueron accedidos.

El estereotipo usado para los *presentationGroup* es: 

- ***InputForm***
  Se trata del formulario donde varios elementos de la UI son agrupados con la finalidad de capturar datos para un proceso.

  El estereotipo usado para los *Input Form* es: 

- ***Anchor***
  Un *anchor* (ancora) permite que el usuario pueda disparar un cambio en el modelo de navegación a través de un link. En UWE no se especifica como una ancora es representada. En HTML puede ser utilizado una ancora (<a>) o un botón. Se especifica el link que seguido cuando se clica en la ancora y el nodo que es alcanzado cuando la ancora es accedida.
  El estereotipo usado para los *Anchor* es:

- ***Button***
  Un botón es un elemento que posibilita que el usuario pueda iniciar una acción en la aplicación web. El modo más común es cuando está asociado a elementos de input donde el usuario puede entrar datos y se puede procesar informaciones o ejecutar un *query*. Así como la ancora, UWE tampoco describe como un botón es representado. En HTML se usa <*input*> con el tipo "*button*" o también se puede usar una imagen.

  El estereotipo usado para representar *Button* es: 

- ***Text***
  El texto es utilizado en el diagrama de presentación para mostrar información estática de la página.

  El estereotipo usado para representar *Text* es: 

- ***Image***
  Es usada para mostrar una imagen estática en la página.

  El estereotipo usado para representar una *Image* es: 

- ***TextInput***
  Permite que el usuario pueda ingresar texto que puede ser usado para una *query* o datos de un formulario.

  El estereotipo usado para representar un *text Input* es: 

- ***FileUpload***
  Tiene como objetivo permitir al usuario la posibilidad de subir archivos a la aplicación web.

  El estereotipo usado es: 

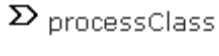
CLASES DEL METAMODELO DE PROCESO

- ***ProcessModel***

  El modelo de proceso es utilizado para modelar procesos de negocio, también llamados de *bussiness processes*, de una aplicación web. Consiste en 2 modelos: Modelo de estructura de proceso y modelo de flujo de proceso, que fueron detallados en el capítulo 4.

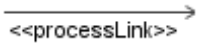
- ***ProcessClass***

  Las *processClass* son usadas para integrar los procesos de negocio en el modelo de navegación y también para definir los datos que son intercambiados con los usuarios durante el proceso. En el modelo de navegación, los *processClass* pueden estar conectados con otros nodos de navegación a través de los *processLinks* definiendo como el proceso puede ser iniciado en la navegación.

  El siguiente estereotipo es usado para representar un *processClass*: 

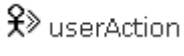
- ***ProcessLink***

  ProcessLink son utilizados para conectar *processClass* a otros nodos de navegación. Son usados para modelar la asociación entre un *navigationClass* y un *processClass*. Indica los puntos de entrada y salida de los procesos adentro de la estructura de navegación [26].
  Debe haber por lo menos una asociación uni-direccional <*processLink*> entre un <*processClass*> y un <*navigationClass*> indicando cual es el nodo de navegación que dispara el proceso. Otro uni-direccional <*processLink*> debe existir para indicar como la navegación continua después del término del proceso. En el caso de que el mismo nodo de navegación sea el que dispara y el nodo que continua después del fin del proceso, un link bi-direccional debe ser usado. (sin flechas).

  Son representados en el modelo de navegación por el siguiente estereotipo: 

- ***UserAction***

  El *userAction* indica el punto en el proceso en que el usuario es requerido para proveer datos que van a alimentar el proceso. Indica la interacción del usuario con la aplicación web cuando este tiene que iniciar algún proceso o responder a algún requerimiento de la página. Cuando el proceso requiere el *userAction*, la interface tiene que mostrar el UI *Elements* correspondiente para que el usuario pueda ingresar los datos [21]. Después de que los datos son ingresados, el flujo del proceso sigue.

  El estereotipo usado para representar el *UserAction* es: 

- ***SystemAction***

  Un *systemAction* corresponde a los pasos de proceso en donde el sistema tiene que procesar datos. El sistema ejecuta las acciones internamente, no dependiendo del usuario.

  El estereotipo usado para representar el *SystemAction* es:

ANEXO B: MEJORAS LOGRADAS

Mejorías	Modelo de Proceso								Modelo de Navegacion				Modelo de Presentacion				
	4.2.1	4.2.2	4.2.3	4.2.4	4.2.5	4.2.6	4.2.7	4.2.8	5.2.1	5.2.2	5.2.3	5.2.4	6.2.1	6.2.2	6.2.3	6.2.4	6.2.5
Reducción de numero de pasos del proceso	X			X													
Reducción del tiempo de ejecución del proceso	X		X		X		X				X		X		X		
Aumento de la confianza en la aplicación		X				X		X		X						X	
Aumento de la eficiencia de la aplicación			X		X		X						X				
Aumento de la satisfacción del usuario					X	X	X				X			X	X		
Reducción de repeticiones en el proceso					X					X	X	X			X		
Eliminación/Reducción de inconsistencias en el proceso								X		X		X				X	
Menos esfuerzo del usuario para ejecutar el proceso			X	X							X				X		
Eliminación de redundancias				X													X
Reducción de riesgo de errores en la aplicación						X	X			X		X					
Aumento de flexibilidad en el proceso								X		X	X						
Mejora en el acceso al proceso									X					X			
Aumento de informaciones referentes al proceso						X										X	
Reducción del abandono de la aplicación		X													X	X	