PRÁCTICA 8: QUIZÁS NO HOY NI MAÑANA, PERO SI ALGÚN DÍA Y ENTONCES SERÁ PARA SIEMPRE[1]

1. Introducción

De acuerdo con los principios del *Diseño Dirigido por el Dominio*, un modelo de dominio debe constituir una especie de lenguaje universal, compuesto por una serie de conceptos de alto nivel, que permita la comunicación entre expertos los expertos del dominio, las personas que financian el desarrollo del sistema, sus (futuros) usuarios, y los desarrolladores.

Dicho modelo de dominio suele especificarse normalmente mediante la creación de un diseño orientado a objetos que captura las entidades y eventos existentes dentro de un dominio, así como las relaciones y restricciones existentes entre ellos. Este modelo de dominio se entiende que ayuda a trabajar con la cada vez más creciente complejidad de los sistemas software mediante la creación de modelos de alto nivel, cercanos al dominio del usuario, que ocultan de cierto modo las complejidades de más bajo nivel propias de los sistemas software.

Una de dichas complejidades de bajo nivel es que, si queremos conservar los datos correspondientes a una entidad de un modelo de dominio, debemos almacenarla en un medio no volátil. Es decir, un objeto de un modelo de dominio no puede vivir eternamente en memoria RAM, en algún momento, si queremos que dicha entidad perdure en el tiempo, debe almacenarse en disco. Obviamente, si lo almacenamos, también deberemos tener en cuenta que con posterioridad podríamos necesitar recuperarlo, sobrescribirlo o borrarlo.

La manera más habitual de persistir un objeto en memoria es almacenarlo en una base de datos relacional, aunque en los últimos años van ganando cada vez más protagonismo otras alternativas, como la utilización de bases de datos NoSQL. La elección de bases de datos relacionales se debe a razones puramente pragmáticas: se trata de una tecnología bastante madura, sobre la cual se lleva trabajando durante, donde se han conseguido importantes avances, y para la que existen potentes sistemas gestores de bases de datos.

No obstante, no es posible persistir modelos orientados a objetos sobre bases de datos relacionales directamente ya que ambos paradigmas, aunque similares, no son directamente compatibles entre ellos. Por ejemplo, supongamos que, para una plataforma de compra *online*, queremos especificar que un usuario pueda tener asociada a su cuenta diferentes direcciones de envío. Utilizando modelos orientados a objetos, este requisito se podría especificar de manera natural colocando una asociación con multiplicidad infinita entre la clase *Usuario* y la clase *Dirección*.

En este caso, se modela de manera natural que objetos de una clase A pueden contener una colección de referencias a objetos de otra clase B, siendo esta colección además de tamaño indefinido y variable. No obstante, este concepto no existe tal cual en el mundo relacional, siendo necesario desarrollar trucos, como la creación de tablas intermedias o la inserción de claves foráneas, para representarlo.

[1] Frase extraída del diálogo final de la película *Casablanca* (Michael Curtiz, 1942)

Por tanto, para poder representar modelos orientados a objetos sobre bases de datos relacionales debemos *traducir* o *trasladar* los objetos de un modelo de dominio a una representación relacional equivalente. Para que dicho proceso de conversión se transparente a los desarrolladores que manipulan objetos se han desarrollado en los últimos años los puentes objeto relacionales, conocidos como *ORM* (del inglés, *Object-Relational Mapper*), los cuales se basan en un conjunto de patrones de diseño.

El objetivo de esta práctica es que el alumno interiorice el problema del desacoplamiento entre los paradigmas orientado a objetos y relacional. Para ello, el alumno deberá implementar una serie de patrones habitualmente utilizados por los puentes objeto-relacional. La siguiente sección describe cómo este objetivo general se plasma en una serie de objetivos concretos.

2. Objetivos

Los objetivos concretos que se pretenden alcanzar mediante el desarrollo de esta práctica son:

1. Aprender a desarrollar modelos de dominio que capturen las entidades y relaciones existentes dentro del ámbito de una aplicación.
2. Aprender a persistir modelos de dominio sobre bases de datos relacionales, utilizando para ello un puente objeto-relacional.
3. Aprender a implementar una estrategia de *Lazy Load* mediante la utilización del patrón *Proxy*.

3. *Polaflix*: Sistema de Visualización en Línea de Series

Paco y Lola son dos oriundos de Polaciones que viven algo aislados del mundo actual. Por ello piensan que el desarrollo de una plataforma *online* para la visualización de series de televisión bajo demanda es una idea innovadora y un proyecto de éxito seguro.

La idea básica del sistema es que cada usuario con cuenta en el sistema pueda buscar una serie, añadirla a su espacio personal y visualizarla. Por cada capítulo visualizado, se cobrará al usuario una pequeña cantidad. Los cargos se van acumulando en la cuenta del usuario y al final de cada mes se pasa el recibo por la cuenta bancaria de cada usuario.

Las series se clasifican en tres categorías: (1) *estándar*, para series de menor demanda o actualidad; (2) *silver*, para series de demanda o actualidad media; y (2) *gold*; para series de gran demanda y actualidad. La visualización de un capítulo de una serie *estándar* se cobra a 0.50€; el capítulo de una serie *silver* a 0.75€; y el de una serie *gold* a 1.50€.

Para los grandes consumidores de series existe la opción de pagar una cuota fija mensual de 20€ y poder visualizar todos los capítulos de todas las series que se desee.

Cada usuario, para poder acceder a la plataforma, tendrá que proporcionar un nombre de usuario, que será único para cada usuario dentro de la plataforma, una contraseña de acceso al sistema y una cuenta bancaria de acuerdo al formato ISBAN.

Mientras cuidan de su nutrida y hermosa cabaña bovina, Paco y Lola han diseñado una serie de *mock-ups* que ilustran cómo debería funcionar de su aplicación. Estos *mock-ups* se adjuntan en el Apéndice A y están también disponibles en la plataforma *Moodle*. El funcionamiento de Polaflix, el sistema ideado por Paco y Lola, de acuerdo con los *mock-ups* diseñados, sería tal como sigue.

Tras autenticarse en el sistema, cada usuario accede en primer lugar a su espacio personal (Apéndice A, Figura 1). Dicho espacio personal deberá mostrar, para cada usuario, la lista de series que actualmente está viendo (etiquetadas como *Empezadas*), la lista de series que desea ver pero que aún no ha empezado (etiquetadas como *Pendientes*), y la lista de series que ha terminado de ver (etiquetadas como *Terminadas*).

El nombre del usuario se muestra en la parte superior de la página. En el caso de la Figura 1, el usuario sería *John Nieve*. Desde esta página, cada usuario puede realizar las siguientes acciones: (1) seleccionar una serie para su visualización; (2) agregar una nueva serie a la lista de pendientes; (3) comprobar el estado de la factura actual o visualizar las facturas ya cobradas. Dichas acciones se describen a continuación.

Seleccionar una serie para su visualización

Si dentro de la página inicial (Figura 1) se selecciona una serie, la aplicación redirige a la interfaz *Ver Serie* (Figura 2). Esta página muestra todos los capítulos disponibles dentro de una serie temporada por temporada. Por cada capítulo se muestra el número de capítulo dentro de la temporada, su título, un enlace que permite comenzar la visualización del capítulo y un mensaje que indica si el capítulo ha sido visto o no. Un capítulo se considera visto desde el mismo momento en el que comienza su reproducción. Es decir, el sistema no verifica que el capítulo haya sido visualizado de manera completa para marcarlo como visto.

Cuando se abre la página *Ver Serie*, si la serie a mostrar es una serie empezada, la página se abriría inicialmente por la temporada correspondiente al último capítulo visualizado. En el caso de que la serie a visualizar esté *pendiente* o *terminada*, esta página se abriría siempre por la primera temporada.

Por último, en la esquina superior derecha, junto al nombre de la serie, se deberá mostrar el tipo de serie de la que se trata: (1) *estándar*; (1) *silver*; o (2) *gold*.

Por último, dentro de esta página, si se pincha con el ratón sobre el nombre de una serie, se deberá mostrar debajo de su título la descripción de correspondiente capítulo.

Agregar una nueva serie

Al seleccionar la opción agregar serie, se debe redirigir el usuario a la interfaz *Agregar Serie* (Figura 3). Esta página muestra el listado de todas las series disponibles en la plataforma. Para facilitar la visualización de dicha lista, sólo se deben mostrar las series que comparten la misma inicial por página. Para facilitar la navegación por las diferentes letras del alfabeto, la interfaz proporciona una serie de enlaces en su parte superior para poder navegar por las diferentes

letras del alfabeto. Además, cómo es natural, la lista de series correspondientes a una misma inicial se muestran ordenadas alfabéticamente.

Al lado del nombre de cada serie se muestra un enlace que permite agregar la serie a la página personal de cada usuario. Si la serie ya estuviese agregada, la acción no tiene efecto. Si la serie no estuviese previamente agregada, se incorporaría a la lista de *pendientes*.

Complementariamente, la interfaz deberá disponer de un cuadro de texto que permita buscar una serie directamente por su nombre. Si la realización de una búsqueda tiene éxito, la página deberá mostrar el listado de la inicial que corresponda con la serie encontrada destacada de algún modo dentro de dicho listado. Si una búsqueda concluyese sin éxito, la página mostrará simplemente un cuadro de diálogo reportando el error.

Dentro del listado de series, si se selecciona el nombre de una serie determinada, se deberá mostrar bajo su título una pequeña sipnosis de la serie, junto con los nombres de su creador y sus principales actores.

Comprobar facturación

Cuando se selecciona la opción *Ver Cargos* (Figura 4), la aplicación redirigirá al usuario a la página de facturación. Dicha página mostrará inicialmente la factura correspondiente al mes en curso.

La factura contendrá una entrada por cada capítulo visualizado. Por cada entrada, se muestra fecha de visualización del capítulo, nombre de la serie a la que pertenece el capítulo, temporada y número de capítulo, y, finalmente, el cargo correspondiente a dicho capítulo, en función de si pertenece a una serie *estándar*, *silver* o *gold*.

Al final de los cargos se mostrará el importe total de la factura. En el caso de los clientes que opten por la opción de una cuota mensual fija, dicho importe total será siempre dicha cuota, como es el caso de la Figura 4.

La interfaz deberá proporcionar además una serie de botones que permitan avanzar o retroceder el mes mostrado, de manea que sea posible la consulta y revisión de facturas correspondientes a meses anteriores al actual.

A la hora de desarrollar el sistema, debe tenerse en cuenta que la lista de todas las series disponibles en la plataforma consume menos de 1Mb.

4. Actividades

El alumno, para alcanzar el objetivo general perseguido, deberá completar las siguientes actividades, utilizando para ello el lenguaje Java cuando sea necesario:

1. Crear un modelo de dominio (diagrama de clases) para dar soporte al sistema descrito. Como resultado de esta actividad el alumno deberá entregar una imagen en formato *png*,

que ilustre dicho diagrama. Si el diagrama fuese muy grande, puede dividirse en varias imágenes.

2. Crear, aplicando los patrones para la transformación estructural objeto-relacional, un esquema relacional que permita persistir el modelo de dominio creado con anterioridad. Como resultado de esta actividad el alumno deberá entregar un documento *pdf* en el que describa paso por paso el proceso de transformación aplicado, indicando por cada paso el patrón de transformación aplicado.
3. Construir un generador de claves artificiales o surrogadas para cada elemento que precise ser persistido.
4. Crear un *DataMapper* que de soporte a las operaciones CRUD para la clase *Serie*. Para la carga de una serie determinada desde el almacén persistente deberá utilizarse una estrategia de *LazyLoad* mediante la aplicación del patrón *Proxy*. Para obtener las conexiones con la base de datos del almacén persistente, el *Data Mapper* deberá hacer uso del *pool* de conexiones creado en el punto anterior.

5. Criterios de Evaluación y Aclaraciones

La práctica se entregará a través de la plataforma *moodle* siguiendo las instrucciones en ella proporcionadas.

La práctica se calificará atendiendo a los siguientes criterios de evaluación:

(1) Corrección del modelo de dominio (2 puntos).
(2) Corrección de la transformación del modelo de dominio en un esquema de bases de datos relacional (2 puntos).
(3) Corrección de la implementación del generador de claves (1 punto).
(4) Corrección de la implementación del *DataMapper* (2 puntos)
(5) Corrección de la implementación de la estrategia de *Lazy Load* (2 puntos).
(6) Limpieza y claridad de los documentos entregados (0.5 puntos).
(7) Cumplimiento de la *Guía de Buenas Prácticas en Programación* (0.5 puntos).

Apéndice A. *Mock-ups* del Sistema *Polafix*

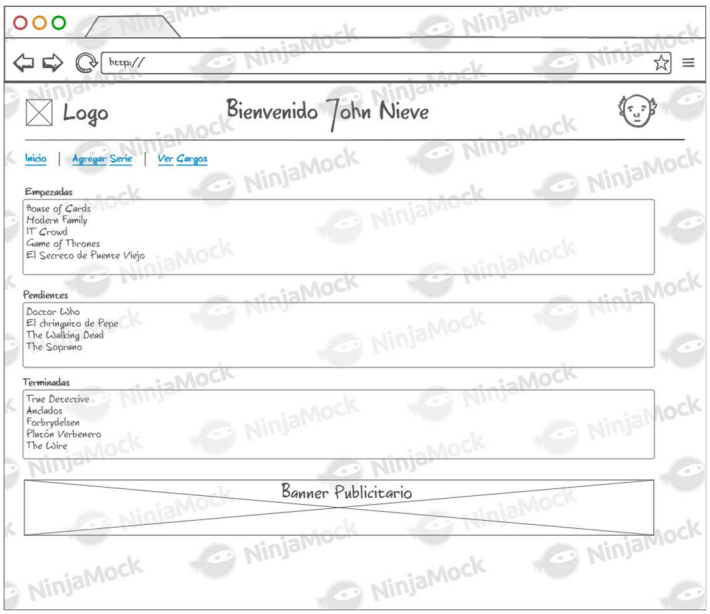

Figura 1. Página Personal de Inicio

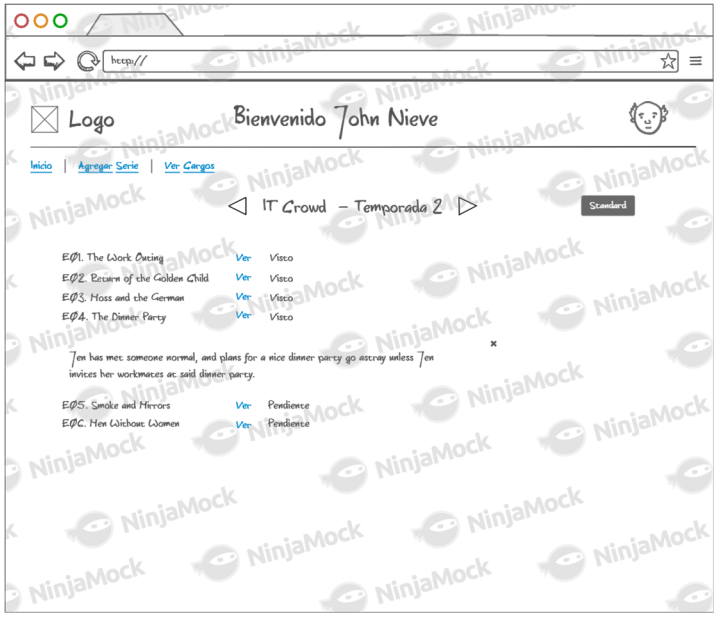

Figura 2. Interfaz Ver Serie

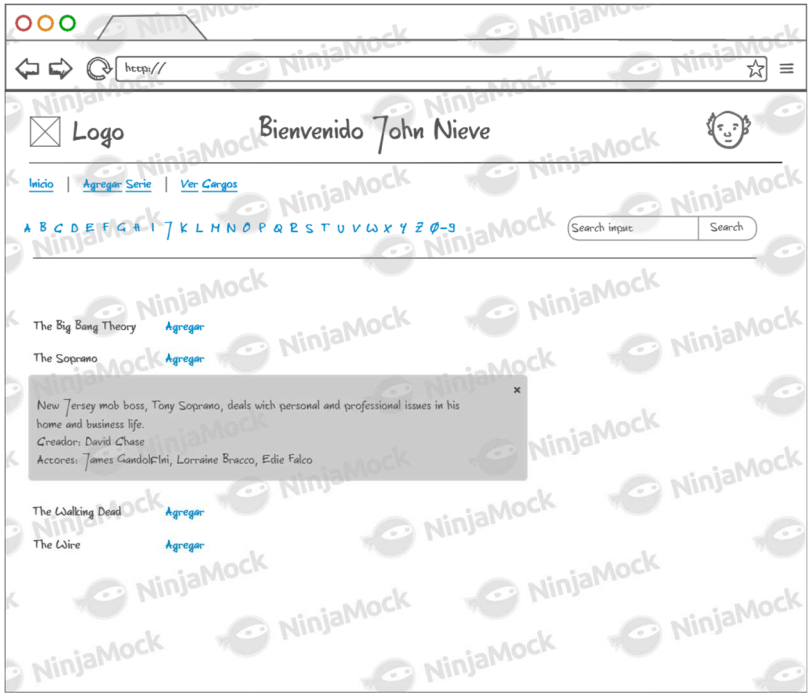

Figura 3. Interfaz Agregar Serie

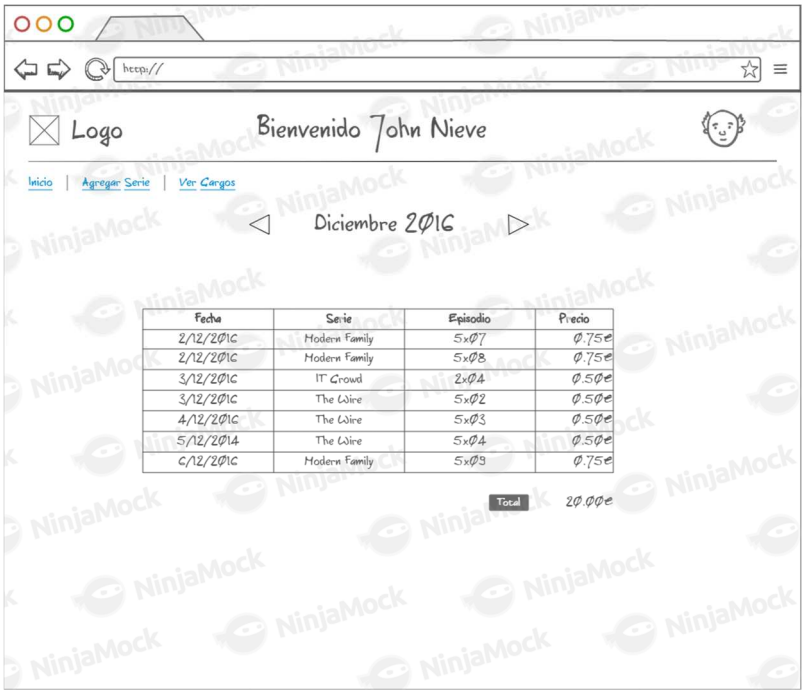

Figura 4. Interfaz Ver Cargos