Proyecto de desarrollo: Cine+

Equipo de desarrollo:
Carlos Toledo Silva C-311
Aylín Álvarez Santos C-312
Rocio Ortíz Gancedo C-311
Ariel Alfonso Triana Pérez C-311

2021

Índice general

Capítulo 1

Introducción

1.1. Alcance del producto

N/A

1.2. Descripción general

1.2.1. Perspectiva del producto

El producto busca controlar la venta de entradas de un cine denominado Cine+. Para esto el producto debe dar soporte tanto a la venta de entradas por taquillas como a la venta de entradas por internet, garantizándose la coordinación de las mismas.

1.2.2. Funciones del Producto

El producto es una aplicación web multiplataforma que permite la compra de entradas a través de su página web por parte de los clientes y calcula el costo de la entradas atendiendo a diferentes parámetros. Además da soporte a la inscripción de usuarios como socios del club de Cine+, así como los beneficios de los que estos pueden disfrutar. También el producto permite la anulación de las compras de entradas por parte de los usuarios y hace las actualizaciones pertinentes dada la anulación. El producto permite además que los gerentes del cine puedan actualizar el listado de películas y horarios disponibles y que estos sean mostrados en la web. Además estos pueden consultar las estadísticas de venta de entradas por diferentes parámetros. También se posibilita la visualización en una página web de las 10 películas sugeridas por uno de los gerentes del cine.

1.2.3. Características de los usuarios

Con el producto interactuarán tres tipos de usuarios: clientes, taquilleros y gerentes. Los gerentes dominan toda la información relacionada con la puesta en escena de las peliculas: horarios, precios, películas que se mostrarán, sugerencias, etc. Los taquilleros dominan como controlar el sistema de tal forma que la venta de entrada puedea mantenerse coordinada entre la venta física en la taquilla y la venta web. Los 3 tipos de usuarios están identificados tanto con dispositivos móviles y tabletas como con computadoras.

Los gerentes y taquilleros tienen el conocimiento necesario para operar este tipo de aplicación web realizando sus respectivas funciones. Los clientes conforman un grupo muy heterogéneo, algunos no tienen conocimiento con productos similares, ahí la necesidad de que la interfaz con la que interactúan sea cómoda.

1.2.4. Restricciones Generales

El cliente solicita que el sistema en cuestión sea una aplicación web, con las funcionalidades anteriormente mencionadas. Además informa que ya reservaron el hosting y dominio (www.cine+.com) de la página y la base de datos. El hosting tiene 4000 MB de almacenamiento.

El cliente pide que las páginas carguen en el tiempo recomendado por los expertos en posicionamiento SEO, para el posicionamiento en buscadores como Google o Bing. Además que el sitio tenga un flujo de navegación sencillo, y que la navegación no sobrepase el tercer nivel.

1.3. Resumen del resto del documento

En el Capítulo 2 nos referiremos a los requerimientos específicos del producto, por lo que abordaremos sus requerimientos funcionales, no funcionales y de entorno. En el Capítulo 3 abundaremos en las diferentes funcionalidades de la aplicación y se explicará como ocurre la interacción entre los diferentes tipos de usuarios y el producto.

El el Capítulo 4 comentaremos sobre la metodología seleccionada y daremos argumentos del porqué de su elección. Además nos referiremos a los principios que seguimos para el desarrollo de la aplicación. En el Capítulo 5 mencionaremos la arquitectura utilizada y expondremos los motivos de la elección de dicha arquitectura por encima de otras que también fueron analizadas.

En el capítulo 6 abordaremos tanto los patrones de visualización de datos como los patrones de acceso a datos empleados en el desarrollo del producto. Finalmente en el capítulo 7 estaremos viendo la modelación de la base de datos.

Capítulo 2

Requerimientos Específicos

2.1. Requerimientos funcionales

El sitio web debe permitir que cualquier usuario pueda comprar entradas. Para ello debe buscar la película deseada y mostrar los horarios y salas en que estará en pantalla dicha película. Luego de que el usuario escoja la sala y el horario de su preferencia, el sistema le pregunta al usuario el número de entradas y asigna unas butacas autómaticamente, pero da opción a que el usuario las modifique a su gusto, las cuales pasarán a estar reservadas de forma provisional. Si pasado algún un tiempo (por defecto 10 min) el usuario no ha efectuado la compra o éste la cancela las butacas vuelven a estar disponibles.

Para el cálculo del precio de la entrada, se deben tener en cuenta los diferentes descuentos que se ofrecen.

Los usuarios que los deseen puden darse de alta como socios del club Cine+, cumpliendo con las pasos pertinentes. A cada socio, cada vez que compre una entrada, se le sumarán 5 puntos, los cuales podrá cambiar en el futuro por entradas. Para hacer esto el socio deberá contar con la suficiente cantidad de puntos para poder pagar todas las entradas de su compra. El precio de la entrada es de 20 puntos por defecto, aunque este se podrá configurar.

La compra por web se realiza por medio de tarjeta de crédito, utilizándose una pasarela de pago segura. En taquilla se admite sólo pago en efectivo. Además se debe poder imprimir un comprobante de venta de las entradas.

Una compra realizada a través de la web puede ser anulada hasta 2 horas antes del comienzo de la sesión, restableciéndole al cliente el costo de la compra y dejando sus butacas disponibles. Si el pago fue hecho por un socio utilizando sus puntos, estos serán devueltos a su cuenta.

Los gerentes podrán actualizar el listado de películas y los horarios, los cuales serán mostrados en el sittio. Además estos podrán consultar estadísticas sobre las ventas de entradas.

Se exige mostrar una vista de 10 películas sugeridas para ver, la cual será actualizada periódicamente. Estará lista seguirá un criterio escogido por alguno de los gerentes.

2.2. Requerimientos no funcionales

Dado que esta es una aplicación web para la venta de entradas para un cine, se le debe dar bastante información sobre el tema a la aplicación para que un navegador pueda encontrala más rápidamente.

Los datos referentes a las películas (salas, horarios, etc) serán guardados en una base de datos SQLite. En esta se guardará además la información referente a los socios, taquilleros y a los gerentes del cine.

El acceso de los gerentes, los taquilleros y los socios de Cine+ al sistema es a través del nombre de usuario y contraseña. Para el acceso al perfil de usuario es necesario comunicaciones seguras, así como para navegar en la administración para el caso de los gerantes y gestionar las ventas para el caso de los taquilleros. Las mismas también son necesarias para la realización de los pagos mediante la pasarela. Por lo anterior es necesario contratar un certificado SSL para el sitio. En caso de que se acceda a través de HTTP será imposible ingresar el nombre y la contraseña, así como recuperar contraseñas.

La interfaz del usuario deberá ser tan familiar como sea posible a los usuarios, lo cual dependerá de la experiencia de los mismos en el uso de otras aplicaciones web. Se agregará además una documentación online para los clientes, para los taquilleros y para los gerentes; donde además la dedicada a los clientes contará con información referente sobre cómo convertirse en un socio de Cine+ y los beneficios que trae serlo.

2.3. Requerimientos de Entorno

Aunque los gerentes y taquilleros del cine pueden poseer una variada gama de dispositivos electrónicos se conece que la administración de Cine+ les proverá de recursos necesarios para la realización de sus funciones. Dicho esto se tiene la seguridad de que cada gerente o taquillero tiene asignado uno de estos dos dispositivos para el acceso:

1. Laptop ASUS con procesador Intel Core i7 de 4ta generación con navegador Mozilla Firefox 86.0

2. Laptop HP con procesador Intel Core i5 de 8va generación con navegador Google Chrome 88.0.4324.104

Además en caso de que el gerente no posea un dispositivo móvil de al menos gama media con el que pueda realizar sus funciones, la administración le proverá de un Samsung Galaxy A10 con conexión a Internet y navegadores como Google Chrome 89.0.4389.72 y Safari 14.0.2.

Los clientes como se conoce son una masa de usuarios heterogénea, como también lo es la masa de dispositivos que ellos tienen disponibles: laptops, móviles, tabletas todos con distintos tipos de sistemas operativos (Windows, Linux, iOS y Android en distintas versiones). Además, en este grupo de usuarios existen distintos tipos de navegadores como Mozilla Firefox, Google Chrome, Opera, Microsoft Edge, Brave, en distintas versiones de los mismos.

En cuanto al hosting, el cliente tiene disponible uno con Windows Server, con 4000 MB de almacenamiento, 1000 MB de base de datos en SQLite, 1 cuenta de acceso para administración, 1000 conexiones concurrentes, 3 cuentas FTP, 1024 Kbps de velocidad de transferencia, soporte para Javascript, para ASP.NET. La aplicación web se desarrollará sobre .NET 5, y C#.

Para el servidor se tiene un Intel(R) Core(TM) i7-8250U CPU 2.60 GHz, 2.60 GHz, con 16GB RAM. El servidor con arquitectura física de 64 bit.

Capítulo 3

Funcionalidades del producto

Se presentan las funcionalidades del producto de software Cine+ a través de diagramas Use Case de UML, máquinas de estados UML y otros.

3.1. Funcionalidades relacionadas con el gerente

Como se observa en la Fig 3.1, existe un actor denominado Gerente el cual se relaciona con varios casos de uso, los cuales representan las funcionalidades del producto relacionadas con el gerente.

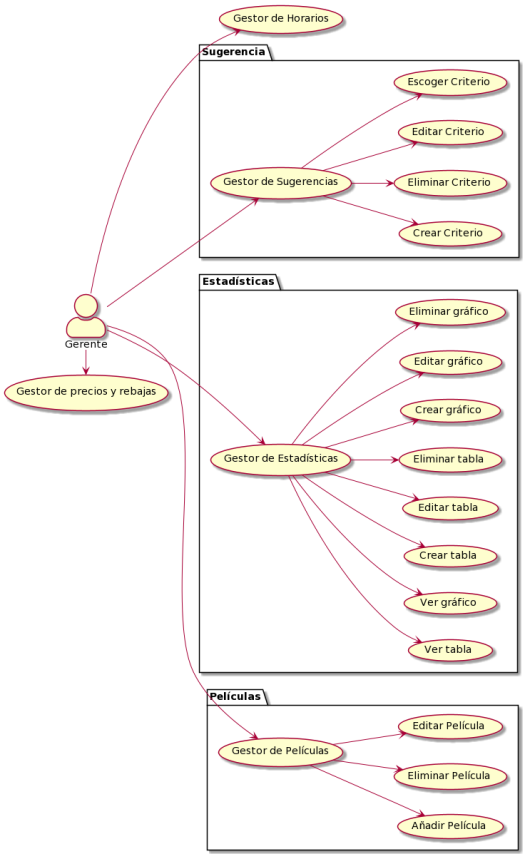

Figura 3.1: Use Case de las funcionalidades del gerente

A continuación se explican cada una de las funcionalidades:

- **Gestor de Películas:** éste pertenece al paquete de casos de uso Películas. Una vez el gerente acceda al gestor, podrá añadir o eliminar películas a proyectar en el cine, así como modificar sus campos mencionados en el Capítulo 7. Así como habilitar la proyección en un horario en específico del cine.
- **Gestor de Horarios:** a través de este caso de uso, se pueden añadir, modificar, o eliminar horarios para las proyecciones.
- **Gestor de Estadísticas:** éste pertence al paquete Estadísticas. Una vez el gerente acceda al gestor, podrá añadir tablas a las estadísticas (que no serán más que consultas a las tablas de la base de datos), eliminarlas, editarlas, o visualizarlas. Además puede añadir, eliminar, editar o visualizar los gráfico utilizando el script Charts.js.
- **Gestor de Sugerencias:** donde el gerente puede seleccionar un criterio para seleccionar las películas, así como crear un nuevo criterio, eliminarlo, o modificarlo.
- **Gestor de Precios y Rebajas:** el gerente puede definir los porcientos de rebajas y las razones de rebajas, así como eliminar o editar los mismos. Así como definir los precios de las entradas.

3.2. Funcionalidades relacionadas con el cliente y el taquillero

Como se observa en la Fig 3.2, existe un actor denominado Cliente y otro Taquillero que representan estos usuarios del sistema, los cuales se relacionan con distintos casos de uso.

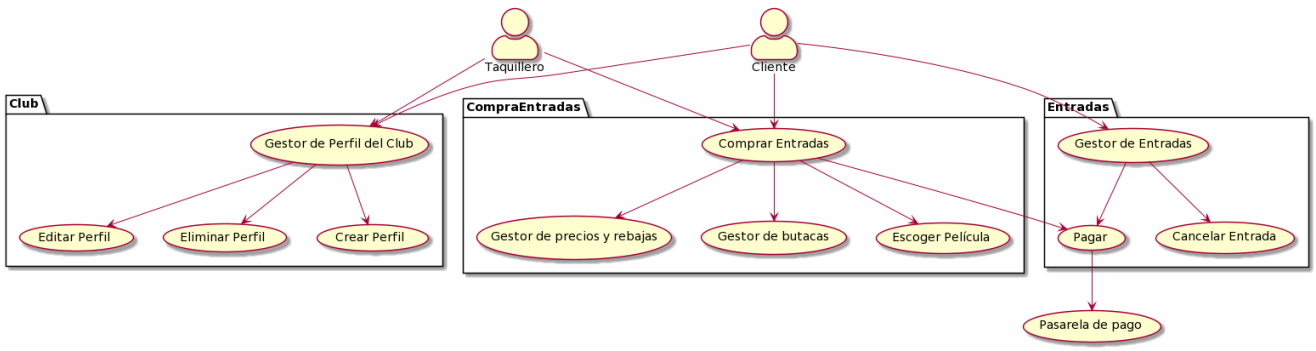

Figura 3.2: Use Case de las funcionalidades del cliente y el taquillero

3.2.1. Relacionadas con el cliente

A continuación se explican cada una de las funcionalidades relacionadas con el cliente:

- **Comprar entradas** este caso de uso permite al usuario seleccionar una película, escoger butacas, declarar circunstacias de rebajas, elegir el pago con puntos del club, cantidad de entradas a comprar, y pagar haciendo uso de la pasarela de pago que se explica en la sección 3.3.

- **Gestor de entradas** sección desde donde el usuario puede visualizar todas las entradas que ha reservado para pagarlas, consultar información o cancelarlas.
- **Gestor de Perfil del Club** sección desde donde el cliente puede darse de alta en el club, así como modificar su información personal, eliminarse del club, o visualizar su información.

3.2.2. Relacionadas con el taquillero

A continuación se explican cada una de las funcionalidades relacionadas con el taquillero:

- **Comprar entradas** este caso de uso permite al taquillero seleccionar una película, escoger butacas, declarar circunstacias de rebajas, elegir el pago con puntos del cliente que pertenzca al club, cantidad de entradas a comprar.
- **Gestor de Perfil del Club** sección desde donde el taquillero puede darle de alta a un cliente en el club, así como modificar su información personal, o eliminarle del club, o visualizar su información.

3.3. Pasarela de pago

No se cuenta con una pasarela de pago real con la cual deba comunicarse el sistema, de ahí la necesidad de emular un servicio que sea la pasarela de pagos. La cual cumple la siguiente máquina de estados UML:

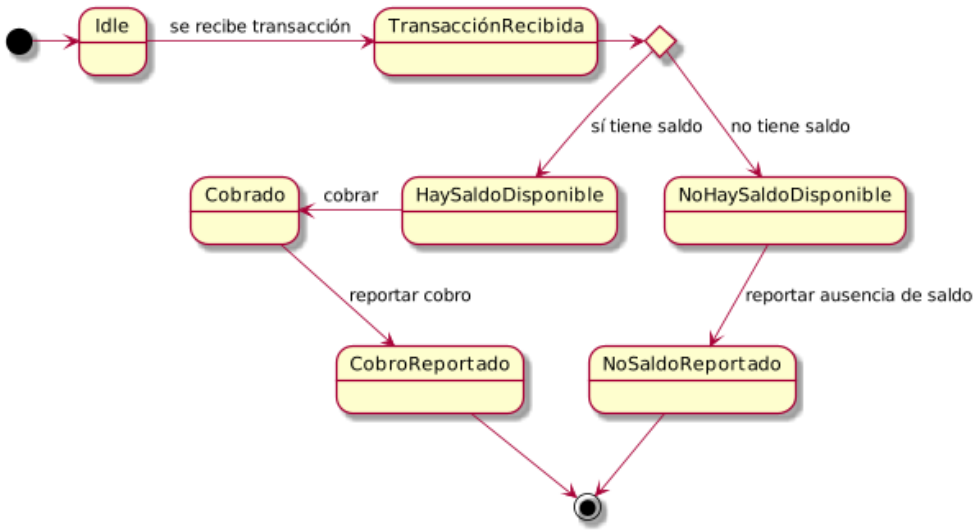

Figura 3.3: Máquina de estados UML para la pasarela de pagos

La decisión de si queda saldo o no se realiza de forma aleatoria, con una probabilidad $p = 0{,}9$ de que quede saldo en la tarjeta y $1 - p$ que no quede. En cualquier caso se devuelve un estado del pago: efecutado, o erróneo. Si fue aceptado entonces se procede a guardar toda la información restante en la base de datos y actualizar el estado de la compra de la entrada.

Capítulo 4

Enfoque Metodológico

Para el desarrollo de este proyecto se aplicará la metodología ágil Extreme Programming (XP).

Se seleccionó una metodología ágil debido a los beneficios de estas y la adaptabilidad a las características de nuestro problema. Una de sus ventajas es que facilita la planificación debido a la división de proyectos en sprints, siendo mas sencillo para el desarrollador abordar el proyecto en pequeñas fases con una duración y alcance determinados. El aumento de la implicación y de la motivación del equipo constituye otra ventaja, puesto que todos los miembros del equipo están informados del estado del proyecto. El cliente desde las primeras etapas del proyecto se le entrega un producto mínimo viable, por lo que este no debe esperar mucho tiempo para empezar a recuperar la inversión realizada. Las entregas continuas garantizan que tras la finalización de cada sprint el software se actualiza con nuevas modificaciones, mejora la calidad del producto debido a los testeos que se realizan después de la finalización de cada fase de desarrollo. La flexibilidad antes los cambios es otra de las características mas importantes ya que permite adaptación del proyecto a las nuevas necesidades o imprevistos que puedan surgir.

Entre el grupo de metodologías ágiles se seleccionó la metodología XP de acuerdo a las características y ventajas de esta sobre nuestro problema.

Una de estas características que nos llevó a escoger esta metodología y no Scrum es debido a los requisitos del cliente. Se planean reuniones cada 15 días donde se entrega los avances del producto y un informe adjunto sobre el trabajo realizado en esos días, no se planean reuniones diarias, aunque contamos con la disponibilidad del cliente para todos los encuentros en los que sea necesario debatir y valorar las sugerencias que brinde nuestro equipo y el cliente para lograr simplicidad, lo cual propicia la retroalimentación frecuente entre ambas partes, lo cual es una práctica de la metodología escogida (Cliente in-situ).

El trabajo se realizará en parejas lo cual constituye una principal característica la metodología XP, teniendo como gran ventaja la detección de errores conforme son introducidos en el código (inspecciones de código continuas), por consiguiente, la tasa de errores del producto final es más baja, los diseños son mejores y el tamaño del código menor (continua discusión de ideas de los programadores), varias personas entienden las diferentes partes sistema.

El listado de los requisitos del proyecto en cuestión se encuentra bien detallado, pero debido a la inexperiencia del personal, el desarrollo de este puede sufrir cambios por lo que el empleo de esta metodología facilitará la asimilación de estos.

El personal tiene como principio hacer un software de calidad, en el que se apliquen los principios SOLID, DRY, KISS, YAGNI, una arquitectura que permita el desacoplamiento y extensibilidad, además que constituye una exigencia del cliente. De esta manera pondríamos en práctica la refactorización y el diseño simple.

Como característica del personal se permite que cualquier programador puede cambiar cualquier parte del código en cualquier momento motivando a la aparición de nuevas ideas por parte del personal.

Cada uno de los motivos anteriores expuestos conllevó a la utilización de esta metodología ante el resto de metodologías ágiles en el proyecto asignado.

Capítulo 5

Arquitectura

En el presente capítulo se describe la arquitectura seleccionada para la implementación del proyecto, y además se justifican las razones de su selección, comparando con otras alternativas.

Se analizaron varios tipos de arquitecturas, entre las que se encuentran:

- Arquitectura de n capas (n-Layers)
- Arquitectura de cebolla (Onion Layers)
- Arquitectura orientada a Servicios (SOA)
- Arquitectura orientada a microservicios.

Las dos últimas alternativas a pesar de ser las que mejores rendimientos en escalabilidad, desacoplamiento y extensibilidad presentan, complicarían la implementación del proyecto influyendo en el atraso de los Sprint. Lo que influye directamente en el plazo de entrega final del proyecto.

En un primer análisis cualquiera de las dos primeras alternativas parece ser idónea para este proyecto, pero se decide utilizar Onion Layers.

5.1. Arquitectura seleccionada

La arquitectura Onion Layers no es más que una arquitectura n-Layers que cumple con el Principio de Inversión de Dependencias del conjunto de principios SOLID. Este principio plantea que:

1. Los módulos de alto nivel no deberían depender de los módulos de bajo nivel. Ambos deberían depender de abstracciones (p.ej., interfaces).
2. Las abstracciones no deberían depender de los detalles. Los detalles (implementaciones concretas) deben depender de abstracciones.

En la arquitectura n-Layers el Principio de Inversión de Dependencias se viola pues la capa de presentación depende indirectamente de la capa de datos (la capa de presentación no funciona si no está presente la de negocio, y esta no funciona si no está presente la capa de datos), cualquier cambio que ocurra en base de datos tendrá un impacto sobre las capas superiores. Además, la capa de negocio (abstracción) no debe depender de la de datos (detalle). Los detalles deben depender de las abstracciones, como plantea el Principio de Inversión de Dependencias.

La regla principal de Onion Layers es que todo el código debe depender de las capas centrales y no de las externas.

5.1.1. Diagrama de la arquitectura

Se presenta en la Fig 5.1.1 el diagrama de la arquitectura. Al observar los círculos concéntricos en el diagrama es que se entiende el por qué del nombre de arquitectura de cebolla.

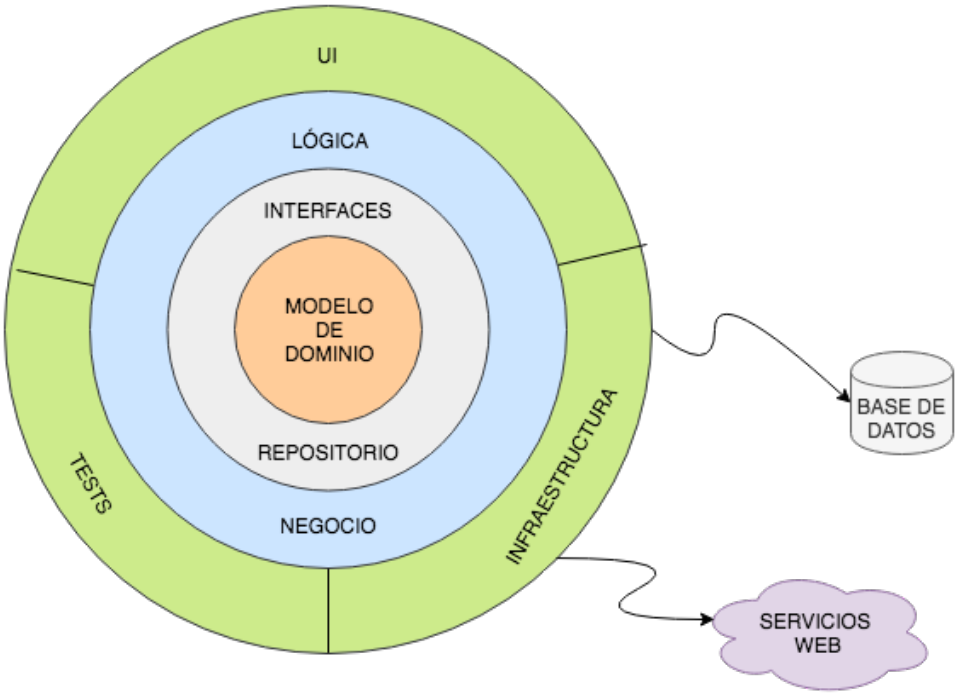

Figura 5.1: Diagrama de la Arquitectura Onion Layers

En este caso notan tres capas externas: UI, Test, Infraestructura. La UI no es más que la interfaz de usuario con la cual se comunican con la aplicación. Los test representan las pruebas que se realizan para el aseguramiento de la calidad del software en desarrollo. La infraestructura no es más que aquellos datos que son comunes a todo el sitio, por ejemplo los datos de autenticación, la base de datos y otros servicios web. Estas capas representan los detalles del software por tanto, dependen del Modelo de Dominio y el resto de capas interiores.

La finalidad de este estilo de arquitectura es poder construir aplicaciones que sean fáciles de mantener, probar y sobre todo que se encuentren desacopladas de elementos de infraestructura tales como base de datos o servicios.

- **Testeable:** Las reglas de negocio pueden ser testeadas sin necesidad de conocer la interfaz de usuario, la base de datos o algún servicio externo.
- **Independiente de la UI**: La interfaz de usuario puede cambiar fácilmente sin tener que afectar al resto del sistema. Un tipo de interfaz de usuario puede ser reemplazada por otra sin cambiar las reglas de negocio.
- **Independiente de la base de datos:** Se puede cambiar de motor de base de datos sin ningún problema. Las reglas de negocio no deben estar amarradas a esta.
- **Independiente de agentes externos:** Las reglas de negocio no deben saber nada que este fuera de su contexto.

Capítulo 6

Patrones de visualización y de datos

En el presente capítulo se detallan los patrones de visualización de datos en el software a desarrollar, y se especifican los patrones de acceso a datos.

6.1. Patrones de visualización de datos

Para la capa de Interfaz de Usuario se propone utilizar el patrón *Modelo-Vista-Controlador* (MVC), el cual propone separar la lógica del negocio de la forma en que se visualizan los datos. Se tienen 3 capas que nada tienen que ver con la arquitectura del software seleccionada en el Capítulo 5.1:

1. **Modelo:** Incluye todas las capas internas mencionadas en la arquitectura seleccionada en 5.1, o sea, constituye el Modelo de Dominio, la interfaz de repositorio y la capa de Lógica del Negocio.

2. **Vista:** Maneja la forma en que se visualiza la información disponible en el Modelo. Hace al modelo independiente de la forma en que se muestra.

3. **Controlador:** Observa las acciones solicitadas por el usuario y decide qué hacer con ellas.

6.2. Patrones de acceso a datos

Para el acceso a datos se propone el uso de la tecnología *Object-Relational-Mapping* (ORM). Como se menciona en el Capítulo 2, la base de datos ha emplear es SQLite, por tanto se debe emplear el marco de trabajo EntityFramework que da soporte a esta base de datos.

Además, se propone el uso del patrón *Repository* para desacoplar la aplicación de la fuente de los datos. También para facilitar cambios en las tecnologías de forma transparente para las capas superiores.

Para la carga de datos de la base de datos se proponen dos patrones:

- **Lazy Loading** consiste en retrasar la carga o inicialización de un objeto hasta el mismo momento de su requerimiento. Esto contribuye a la eficiencia de los programas, evitando la precarga de datos que podrían no llegar a utilizarse. Generalmente se utiliza en los repositorios o colecciones.

- **Eager Loading** consiste en la asociación de modelos relacionados para un determinado conjunto de resultados con solo la ejecución de una consulta, en lugar de tener que ejecutar n consultas, donde n es el número de elementos en el conjunto inicial. Este patrón debe utilizarse cuando las relaciones no son demasiadas, ya que es una buena práctica utilizar Eager Loading para reducir las consultas en el servidor. Cuando se está convencido de que el dato se utiliza en varios lugares junto a la entidad principal.

Capítulo 7

Modelo de datos

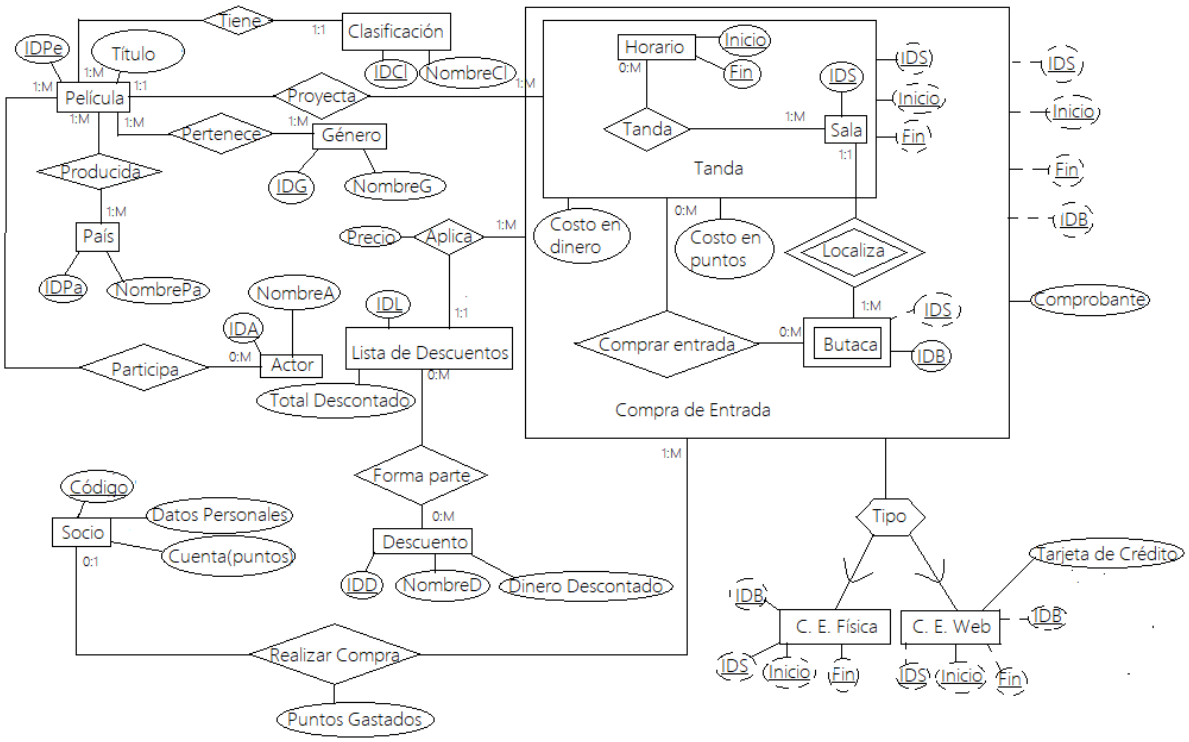

Figura 7.1: MERX de la modelación conceptual de la base de datos.

A partir de esta modelación se obtiene la siguiente descomposición que cumple la Propiedad de Preservación de las Dependencias Funcionales (PPDF) y la Propiedad del Join sin Pérdida de información y cada uno de sus esquemas relacionales se encuentra en 3ra Forma Normal.

$U_{Pelicula} = \{IDPe, Titulo\}$
$F_{Pelicula} = \{IDPe \rightarrow Titulo\}$

$U_{Pais} = \{IDPa, NombrePa\}$
$F_{Pais} = \{IDPa \rightarrow NombrePa\}$

$U_{Género} = \{IDG, NombreG\}$
$F_{Género} = \{IDG \rightarrow NombreG\}$

$U_{Actor} = \{IDA, NombreA\}$
$F_{Actor} = \{IDA \rightarrow NombreA\}$

$U_{Horario} = \{Inicio, Fin\}$
$F_{Horario} = \emptyset$

$U_{Sala} = \{IDS\}$
$F_{Sala} = \emptyset$

$U_{Butaca} = \{IDS, IDB\}$
$F_{Butaca} = \emptyset$

$U_{Tanda} = \{IDS, Inicio, Fin, Costo\ en\ dinero, Costo\ en\ puntos, IDPe\}$
$F_{Proyecta} = \{IDS, Inicio, Fin \rightarrow Costo\ en\ dinero, Costo\ en\ puntos, IDPe\}$

$U_{Descuento} = \{IDD, NombreD, DineroDescontado\}$
$F_{Descuento} = \{IDD \rightarrow NombreD, Dinero\ Descontado\}$

$U_{Lista\ de\ Descuentos} = \{IDL, Total\ Descontado\}$
$F_{Lista\ de\ Descuentos} = \{IDL \rightarrow Total\ Descontado\}$

$U_{Compra\ de\ Entrada} = \{IDS, Inicio, Fin, IDB, IDL, Precio, Comprobante\}$
$F_{Compra\ de\ Entrada} = \{IDS, Inicio, Fin, IDB \rightarrow IDL, Precio, Comprobante\}$

$U_{Socio} = \{Código, Datos\ Personales, Cuenta(puntos)\}$
$F_{Socio} = \{Código \rightarrow Datos\ Personales, Cuenta(puntos)\}$

$U_{Realizar\ Compra} = \{IDS, Inicio, Fin, IDB, Código, Puntos\ Gastados\}$
$F_{Realizar\ Compra} = \{IDS, Inicio, Fin, IDB \rightarrow Código, Puntos\ Gastados\}$

$U_{C.E\ Fisica} = \{IDS, Inicio, Fin, IDB\}$
$F_{C.E.\ Fisica} = \emptyset$

$U_{C.E\ Web} = \{IDS, Inicio, Fin, IDB, Tarjeta\ de\ Crédito\}$
$F_{C.E\ Web} = \{IDS, Inicio, Fin, IDB \rightarrow Tarjeta\ de\ Crédito\}$

$U_{Producida} = \{IDPe, IDPa\}$
$F_{Producida} = \emptyset$

$U_{Participa} = \{IDPe, IDA\}$
$F_{Participa} = \emptyset$

$U_{Forma\ Parte} = \{IDL, IDD\}$

$F_{Forma\ Parte} = \emptyset$

$U_{Producida} = \{IDPe, IDPa\}$
$F_{Producida} = \emptyset$

$U_{Pertence} = \{IDPe, IDPa\}$
$F_{Pertenece} = \emptyset$

$\rho = (Pelicula, Pais, Género, Actor, Horario, Sala, Tanda, Butaca, Proyecto, Descuento,$
$Lista\ de\ Descuentos, Compra\ de\ Entrada, Socio, Realizar\ Compra, C.E.Fisica, C.E.Web, Producida,$
$Participa, Forma\ Parte, Pertenece)$

Apéndice A

Manual de Usuario