PROYECTO CINE+

Informe Individual

Descripción breve

Describir la participación del autor en el desarrollo e implementación del proyecto de desarrollo de software Cine+.

Ariel Alfonso Triana Pérez
ariel.triana@estudiantes.matcom.uh.cu

1. Introducción

En el proyecto Cine+ se desarrollaron 3 Sprints para desarrollar el software propuesto. En estos Sprints se ejecutaron tareas como la concepción del software, levantamiento de requerimientos, especificación de los mismos, implementación de la arquitectura de software, y de los patrones de visualización y acceso a datos, refactorización de código.

2. Participación en los distintos Sprints

Estos sprints tuvieron una duración de 15 días, a continuación, se describe la participación del autor en cada uno de ellos.

2.1. Sprint 1:

- Definición de la arquitectura:

El autor propuso para el sistema el empleo de la arquitectura de cebolla, conocida por su nombre en inglés, Onion Architecture. Esta arquitectura es una variante de la clásica arquitectura de n capas; la arquitectura de cebolla basa su concepto en el principio SOLID de Inversión de dependencia entre los Datos del sistema y la capa de Lógica del Negocio.

Se definieron las siguientes capas para la arquitectura: como capa central se presenta la capa DomainLayer, la cual representa los datos del sistema o el dominio del problema. Luego, la próxima capa es RepositoryLayer, en la cual se implementan los patrones de acceso a datos propuestos, como Repository o Eager Loading. Luego, se tiene la capa de ServiceLayer en la que se implementa la lógica del negocio del sistema que se pretende implementar. Posteriormente, se tienen dos capas que interactúan con la anterior: los Testers y la capa CineWeb que es la interfaz de usuarios del sitio web desarrollado.

- Definición de los patrones de visualización y de acceso a datos:

Como patrón de visualización se definió la utilización del patrón Modelo-Vista-Controlador (MVC, por sus siglas). En este patrón las capas interiores de la arquitectura, entiéndase DomainLayer, RepositoryLayer, y ServiceLayer, representan el modelo del patrón en cuestión. Mientras que el controlador y las vistas representan la interfaz de usuario (UI).

Para el acceso a los datos se sugirió el empleo del patrón Repository para almacenar las instancias de los datos en la base de datos. Así como el empleo de Eager Loading para cargar los datos que se conozca se van a utilizar en el momento. Además, se propuso el empleo de la tecnología Object-Relational Mapping (ORM), utilizando el ORM de Microsoft.EntityFrameworkCore.

- Implementación de a pasarela de pagos:

La pasarela de pagos en un nombre de espacio dentro de la capa de ServiceLayer que tiene como nombre PaymentGateway. Este namespace contiene dos clases: Bernoulli Variable y Bank Total. La primera de estas representa una variable aleatoria discreta cuyo valor de probabilidad de tener un éxito es de 0.9. Mientras, que la segunda clase representa el cajero del banco que realiza las transacciones pedidas por los usuarios. O

sea, esta clase es la encargada de crear una instancia de BernoulliVariable para reportar si un pago fue correcto o no.

2.2. Sprint 2:

- Se implementó la autorización de usuario:

Para toda lo relacionado con la autorización de usuarios se utilizó el framework Microsoft.AspNetCore.Identity. Para ello se crea un tipo denominado AppUser, en el cual se almacena la dirección de lo usuarios. Además, se utilizan clases propias del framework como UserManager, RoleManager o SignInManager. Se definieron tres roles de usuarios según los cuales se definirían los permisos de los mismos, los tres roles son: Member representa el socio del Club Cine+, BoxOfficer representa el taquillero del cine y Manager, que representa el gerente del Cine.

Se tienen dos interfaces IAutorizeUser e IUserStore, la primera interface define las entidades encargadas de autorizar usuarios en el sistema, y la otra interface define el gestor de usuarios. Se implementaron dos clases que cumplen con estas interfaces: CinemaAuthorization y CinemaUserStore.

Para el acceso a la capa ServiceLayer.Identity desde capas superiores se utilizó el patrón de diseño Facade, creando una clase denominada CinemaUserFacade que almacena una instancia de IAutorizeUser y de IUserStore. Además, se utiliza el principio de Inversión de Dependencias pues la clase Facade no se encarga de decidir que implementación de las interfaces va a utilizar si no que las recibe desde el constructor. Las implementaciones a utilizar se definen en la capa CineWeb utilizando inyección de dependencia apoyados en el framework Microsoft.AspNetCore.DependencyInjection.

Como parte de la autorización de usuarios se implementaron las vistas y los controladores relacionados con la creación de usuarios y autorización de los mismos.

2.3. Sprint 3

- Implementación de las vistas

El autor se encargó de la implementación de las vistas del proyecto utilizando jQuery 3.6 y Bootstrap 4.5.3, además para la generación de gráficos se utilizó Charts.js.

- Refactorización del código del módulo de compra y el módulo de gerencia

Se revisó el código del módulo de compra de entradas y el modelo de dominio del proyecto, refactorizando el código tanto de los controladores, como de la capa ServiceLayer.