Informe Técnico

Universidad de la Habana.

Facultad de Matemática y Computación, carrera: Ciencia de la Computación.

Evaluación del proyecto de curso para Ingeniería de Software. Curso: 2021.

I. Presentación

Nombre del proyecto: `Cine++`

Integrantes: Equipo #3:

- Carlos Luis Águila Fajardo
- Eduardo García Maleta
- Sergio Pérez Pantoja
- Ricardo Piloto Marin

II. Introducción

Alcance del Producto

Nuestro producto resuelve la necesidad del cliente de mantener y administrar toda la información y el funcionamiento de un cine, de forma organizada. Se atiende además a necesidades secundarias como la manipulación de la información, el mantenimiento del producto y la reusabilidad del mismo en futuros proyectos. La plataforma es basta y está correctamente diseñada para un uso sencillo y completo en la administración del negocio.

Descripción general del producto

La plataforma es capaz de mostrar, organizar, añadir, eliminar y modificar los datos de los clientes y socios del cine, al igual de los materiales que se proyectan, ya sean películas comerciales, cortos, documentales, filmes en general, estas funciones de cara al administrador del negocio. Todo esto con una solución sencilla y sin un alto grado de adistramiento necesario. De cara al cliente del negocio, el interesado puede comprar sus entradas para sus filmes favoritos, así como interactuar con la aplicación web, pudiendo conocer nuevas opciones como los filmes con mejores calificaciónes, los más vistos, etc. También puede ser partícipe de este proceso, pudiendo calificar los filmes que ya haya disfrutado.

La aplicación también se encarga de llevar pagos y demás cálculos del sistema, como hacer descuentos a determinados clientes con requisitos especiales o la

compra de entradas por puntos acumulados para los socios del cine. Los administradores del cine podrán adquirir un alto grado de familiaridad y experiencia con el producto por su facilidad y versatilidad, ya que se han respetado los requisitos requeridos en esta línea de trabajo de negocios que quieren ser administrados con software.

De acuerdo a las restricciones, se hace necesario un moderado nivel de seguridad. Dado que la información manejada es sensible, se desea un sistema de base de datos facilmente accesible pero solo para administradores. Ha sido garatizada una interfaz simple y minimalista para todo el que tenga acceso a visualizar los datos. Adicionalmente, se desarolló de forma tal que las medidas de seguridad sean implementadas de manera lo menos molesta posible, es decir, no más que un requerimiento de usuario/contraseña.

La plataforma facilita la integración entre los distintos sistemas operativos posibles: Windows, MacOS, o alguna distribución de GNU/Linux para sus administradores; puesto que el sistema no discrimina por este tipo de preferecnias. Para el correcto funcionamiento de la plataforma, es necesario cumplir con dependecias de software, de las cuales se detallará todo a lo largo del informe.

Resumen del resto del documento

El proyecto desarrolla una aplicación web para la total administración de un Cine, donde los propietarios de este podrán tener una productiva forma de interactuar con la plataforma. Detallamos las diferentes funciones de nuestro proyecto punto por punto, así de como ha sido gestionado su desarrollo.

El cliente hizo la petición de una plataforma amigable, con funciones clásicas de este tipo de establacimiento. Todos los requisitos fueron transformados a requerimientos más concretos y fueron desarrollados en una base de datos junto a la aplicación web como interfaz del usuario, donde pueda interactuar tanto el administrador del cine como su clientela. En el actual informe se detalla cada punto del desarrollo.

III. Requerimientos Específicos

Requerimientos Funcionales

Primer grupo (de cara al cliente del cine):

- Página para venta de entradas y otros servicios del cine.
- Página de socio del cine (tanto para unirse y brindar sus credenciales, como para darse de alta).

- Página de sugerencias (aquí apareceran las películas más vistas, películas con mejores rating, promociones, ofertas del cine; cualquiera de estos criterios puede ser configurado por el administrador).
- Página para cancelar la compra de una entrada.

Segundo grupo (de cara al administrador del cine):

- Visión de estadísticas (películas más vendidas, filtrado por día, mes o año, más gustadas, actores, géneros, comparaciones entre cine cubano y extrangero).
- Edición de disponibilidad de películas y horarios de las salas.
- Edición de interfaz de inicio con ofertas con fines propagandísticos y económicos. ### Requerimientos no funcionales
- Usabilidad: La pltaforma está preparada para que todo el que interactúe con ella, sea capaz de sacar su máximo provecho. Por un lado tenemos la cara dispuesta al administrador, donde facilmente puede gestionar cada filme, asignando horarios, cambiando su estado de disponibilidad o no y editando la metadata, ya sea información de los actores, género, país de origen, etc.

  Por otro lado, tenemos la cara dispuesta al cliente del cine, quien puede hacer uso de la plataforma de manera sencilla, solo comprando la entrada que desea; luego, para usuarios que estén más motivados y quieran sacar más partido, pueden disfrutar de listas de recomendaciones, así como brindar sus credenciales para acumular puntos por compras y formar parte de la plataforma ayudando a calificar los filmes de los que disfruta.
- Seguridad:
  - Confidencialidad: La información manejada en la plataforma, respecto a credenciales de los Socios, es solo visible al administrador del cine, quien hará un uso de ella bajo su responsabilidad. Por otro lado, la información de filmes en la plataforma, no constituye información sensible, siempre y cuando hablemos de filmes que estén disponibles al público, los que no están en puesta o aún no han sido estrenados, constituyen información sensible y, al igual que las credenciales, están bajo la seguridad del administrador de la plataforma.
  - Integridad: Todos los datos manejados en la plataforma serán cuidadosamente protegidos bajo el uso del nombre y la contraseña del administrador de la plataforma.
  - Disponibilidad: Se garantiza el acceso a la infraestructura siempre y cuando el administrador proporcione su credencial de forma correcta.
- Diseño e implementación: De los lenguajes de programación, el protagonista fue C#, pues era un requisito en el proyecto, y utilizamos .net Framework. En cuanto a tecnologías, ya que implementamos una aplicación web con la arquitectura MVC, fue necesario desarrollar los modelos y para ello se

utilizó Entity Framework, corriendo este por detrás una base de datos en SQLite. Para la interfaz, Bootstrap. A continuación, una lista detallada de cada una de las tecnologías:

- `Antlr` version 3.5.0.2
- `Bootstrap` version 5.0.1
- `Entity Framework` version 6.4.4
- `jQuery` version 3.3.1
- `jQuery.Validation` version 1.17.0
- `Microsoft.AspNet.Mvc` version 5.2.7
- `Microsoft.AspNet.Razor` version 3.2.7
- `Microsoft.AspNet.Web.Optimization` version 1.1.3
- `Microsoft.AspNet.WebPages` version 3.2.7
- `Microsoft.CodeDom.Providers.DotNetCompilerPlatform` version 2.0.0
- `Microsoft.jQuery.Unobtrusive.Validation` version 3.2.11
- `Microsoft.Web.Infrastructure` version 1.0.0.0
- `Modernizr` version 2.8.3
- `Newtonsoft.Json` version 11.0.1
- `WebGrease` version 1.6.0

Requerimientos de Entorno

A partit del desarrollo de una aplicación web accesible desde cualquier navegador moderno y, por tanto, desde cualquier dispositivo de escritorio independientemente del sistema, se ha facilitado para que la infraestructura sea multiplataforma. Es necesario que cumpla además los requerimientos de seguridad planteados anteriormente en el presente informe, pues el administrador corre con la responsabilidad del correcto uso de sus credenciales. De esta forma, cualquier empleado que esté verificado por el dueño del negocio, puede adquirir credenciales de administrador y acceder a los datos y actualizarlos a conveniencia.

Se estima, para el correcto funcionamiento del proyecto planteado, la necesidad de un servidor dedicado a ejecutar la plataforma, para la base de datos SQLite y C# con .net Framework para el servicio de datos a la aplicación web; un basto almacenamiento para la base de datos; y que disponga de una velocidad de internet decente para la posibilidad de ser accedida por una cantidad representable de clientes. No se presentan requerimientos adicionales para los empleados.

IV. Funcionalidades del Producto

Presentar las funcionalidades usando un diagrama de casos de uso del sistema con la correspondiente descripción de los actores o storyboards mostrando en una secuencia de imágenes (prototipos) la evolución que va teniendo el sistema.

V. Enfoque Metodológico

En el desarrollo del proyecto nos asegurado de conocer los principios de diseño que aún no dominábamos del todo, así como desglosar sus principales características para asegurar su utilización en nuestro diseño.

Fue utilizado el Principio Solid, para el acote y especificación de las responsabilidades de cada programa, y extensibilidad. Principio Liskov, segregación de interfaces, y la no dependencia entre los distintos módulos. Principio Dry, el que se puede lograr con un correcto encapsulamiento. Principio Kiss, lográndose con una buena simplicidad, mantenida como objetivo en todo el proyecto. Principio Yagni, que se consigue con una objetividad clara, sin añadir características innecesarias.

También adaptamos la metodología Crystal Clear, ya brinda una serie de ventajas a los equipos y proyectos que la utilizan, principalmente equipos pequeños, además de que fue nuestra favorita de las estudiadas en el curso. Esta metodología da lugar a la auto-modificación y mejora de las características dentro de las propias etapas de desarrollo, así como a la discusión abierta y constante sobre como mejorar las características sin interrumpir severamente el proceso de desarrollo. Nos permitió una comunicación cercana y mejorada para el equipo desarrollador y promueve la interacción e intercambio entre miembros, así como la inclusión de reglas, pautas, convenciones y metodologías propias con los que los equipos tengan mas experiencia y facilidad, considerando que no todos los procedimientos o estructuras productivas funcionan por igual para todos los equipos.

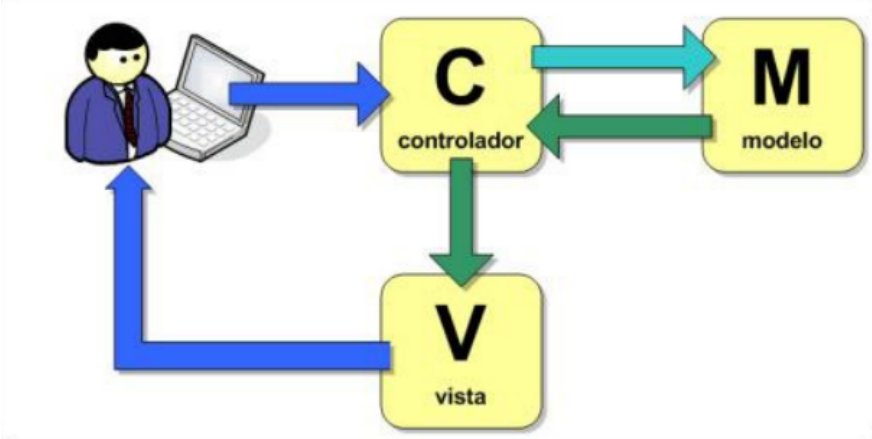
VI. Arquitectura

Utilizamos la arquitectura Modelo-Vista-Controlador (MVC) que permite estructurar la aplicación en componentes o capas con mínimo acoplamiento entre ellas, el Modelo encargado de garantizar la permanencia de los datos, la Vista responsable de generar la interfaz de usuario y el Controlador que funciona de intermediario entre ambas capas capturando las acciones del usuario en la Vista y llamando a acciones CRUD (Create, Read, Update, Delete) en el Modelo para consultar o actualizar los datos.

Elegimos esta arquitectura por su simplicidad (principio KISS) y por el desacoplamiento entre las capas que nos permite cumplir con el principio SOLID contribuyendo a una mayor rapidez en el desarrollo del producto, a que sea más sencillo de mantener y a mayor facilidad al realizar test unitarios. También usamos esta arquitectura por su factibilidad al ya existir varias herramientas en el entorno de desarrollo de .NET que nos facilitan el proceso, como EntityFramework y ASP.NET MVC, cumpliendo con el principio DRY.

Para plantear con un ejemplo cómo funcionaría este patrón de arquitectura en el proyecto, digamos que un usuario quiere comprar una entrada para cierto

filme en la página web de la aplicación. Primeramente el usuario al introducir el URL hacia la página el HomeController se encarga de mostrarle al usuario un view que representa el Index como página principal del sitio web. El usuario procede a escoger entre las películas con mayor calificación, el controlador de filme se encarga de acceder al modelo Filme y obtener los mejores filmes según algún criterio, en este caso calificación, y le muestra al usuario el view "Top" con los filmes ordenados según este criterio. Luego el usuario elige el filme del cual desea obtener entradas y el controlador le muestra un view para que inserte la cantidad de entradas a comprar y modifique las butacas si lo desea, una vez realizado esto el usuario debe completar su compra introduciendo sus datos de su tarjeta de crédito a través de una pasarela de pagos segura.



VII. Patrones de visualización y de datos

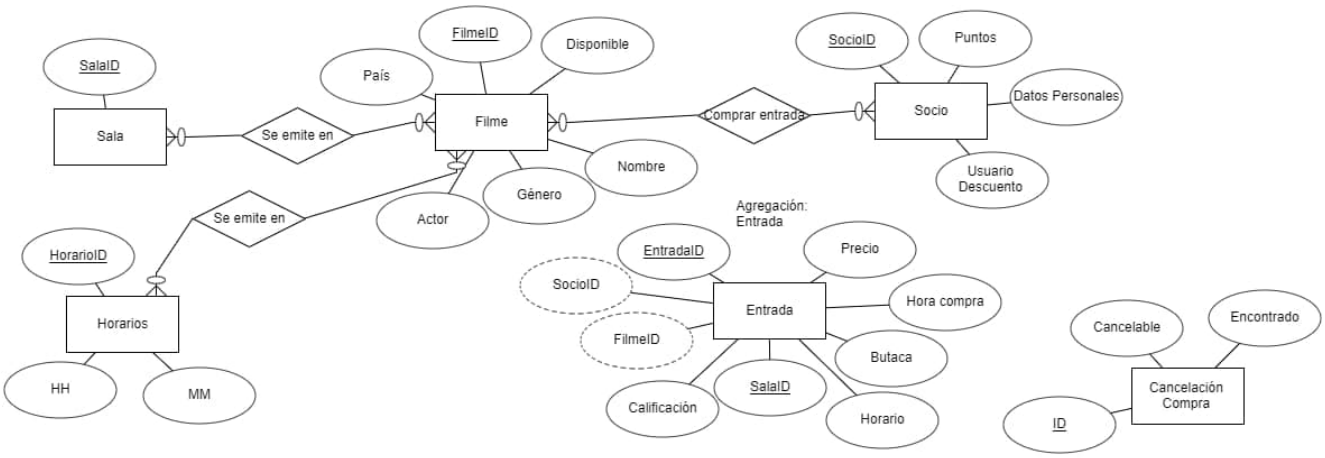
En este proyecto al usar la arquitectura Model-View-Controller, el acceso a datos se encuentra definido por las capas de Modelos y de Controladores. Se implementaría un patrón Data Mapper donde tenemos unas clases que representan las tablas de la base de datos (modelos) y otras clases que se encargan de las operaciones de manipulación de estos datos (controladores). Decidimos utilizar este patrón ya que permite separar los datos de la lógica para manipularlos y haciendo uso del Object-Relational Mapping (ORM) a través de Entity Framework nos facilita el mantenimiento del código y la limpieza del mismo, a la vez que aumenta la velocidad de desarrollo mediante el uso de Language Integrated Query (LINQ) con ASP.NET al no tener que escribir constantemente código SQL repetitivo.

Como patrón de visualización usamos el patrón Model-View-Controller (MVC) principalmente porque nos permite desacoplar la lógica de la aplicación de la interfaz gráfica de usuario, contribuyendo a facilitar los tests, crear datos arbitrarios para probar la interfaz y añadir un mayor nivel de limpieza en el

proyecto en general. Aquí el Model o Modelo se encarga de la lógica de negocio, la persistencia y consistencia de los datos. La Vista representa el estado del Modelo en un momento determinado y logra una independencia entre el modelo y la presentación de los datos, por ejemplo, en el caso de este proyecto la Vista se responsabiliza del HTML (HyperText Markup Language) necesario para ofrecer una interfaz gráfica amigable para el usuario, sin embargo si la comunicación es con un software o producto de terceros, estos datos podrían ser representados como JSON (Javascript Object Notation). También se hace uso del Observer Pattern o Patrón Observador donde al ocurrir un evento (un click en un botón de la página web) el Controlador realiza las acciones necesarias sobre el Modelo y se actualiza el View.

VIII. Modelo de datos

Presentar el modelo que determina la estructura de la base de datos de su proyecto.



IX Anexo 1: Manual de usuario

1. Nombre del software

- Cine ++

2. Autores

Equipo #3: - Carlos Luis Águila Fajardo - Eduardo García Maleta - Sergio Pérez Pantoja - Ricardo Piloto Marin

3. Objetivos del proyecto.

El proyecto desarrolla una plataforma para la admistración completa, óptima y sencilla de un cine, donde los administradores podrán tener una productiva forma de interactuar con la arquitectura.

4. Requerimientos técnicos. Características de la máquina (o red) donde se ejecutará la aplicación

• Es suficiente con que el dispositivo pueda ejecutar cualquier navegador moderno.

5. Requerimientos de software

- Antlr version 3.5.0.2
- Bootstrap version 5.0.1
- Entity Framework version 6.4.4
- jQuery version 3.3.1
- jQuery.Validation version 1.17.0
- Microsoft.AspNet.Mvc version 5.2.7
- Microsoft.AspNet.Razor version 3.2.7
- Microsoft.AspNet.Web.Optimization version 1.1.3
- Microsoft.AspNet.WebPages version 3.2.7
- Microsoft.CodeDom.Providers.DotNetCompilerPlatform version 2.0.0
- Microsoft.jQuery.Unobtrusive.Validation version 3.2.11
- Microsoft.Web.Infrastructure version 1.0.0.0
- Modernizr version 2.8.3
- Newtonsoft.Json version 11.0.1
- WebGrease version 1.6.0

6. Forma de instalar la aplicación

- Basta con descargar y ejecutar. Vale destacar que los paquetes asociados en `/packages` son relativamente pesados ($\approx$ 200Mb).

7. Breve explicación de cada una de las opciones del sistema (debe contener imágenes de las pantallas con datos)