Proyecto Cine+

# Informe Individual

Jessica Núñez Hernández

Equipo 5
Grupo C312
Profesor: José L. Castañeda

En el proyecto se fue encomendado por el jefe del equipo enfrentar la tarea de la confección de los controladores trabajando estrechamente con los desarrolladores visuales y con el desarrollador de las distintas capas de repositorios y la base de datos.

- CinemaController (./Controllers/CinemaController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los cinemas contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `[HttpPost]`<br>`public IActionResult Create(Cinema obj)` | Captura un cinema creado desde la interfaz visual y lo agrega a la base de datos |
| `public IActionResult Update(Cinema obj)` | Captura un cinema modificado desde la interfaz visual y lo actualiza en la base de datos |
| `public IActionResult CinemaList()` | Proporciona la lista de Cinemas existentes en la base de datos a la vista CinemaList |

- DiscountController (./Controllers/DiscountController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los descuentos contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `[HttpPost]`<br>`public IActionResult Create(Discount obj)` | Captura un descuento creado desde la interfaz visual y lo agrega a la base de datos |
| `public IActionResult Update(Discount obj)` | Captura un descuento modificado desde la interfaz visual y lo actualiza en la base de datos |
| `public IActionResult DiscountList()` | Proporciona la lista de descuentos existentes en la base de datos a la vista DiscountList |

- MainController (./Controllers/MainController.cs):
  Este controlador está dedicado solamente a mostrar la página principal para usuarios regulares

- ManagerController (./Controllers/ManagerController.cs):
  Este controlador está dedicado solamente a mostrar la página principal para usuarios Administradores

| *Función* | *Objetivo* |
|---|---|
| `public IActionResult Main()` | Muestra la página principal |
| `[HttpPost] Public IActionResult Main(IFormCollection querys)` | Captura un formulario desde la interfaz visual y analiza la consulta devolviendo una respuesta según lo solicitado |

- MovieController (./Controllers/MovieController.cs):
  Este controlador se encarga de todas las vistas que interactúan con las películas contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `[HttpPost] public IActionResult Create(Movie obj, IFormCollection createform)` | Captura un Movie y un formulario creados desde la interfaz visual para agregar una película a la base de datos |
| `public IActionResult Update(Movie obj, IFormCollection updateform)` | Captura un Movie y un formulario creados desde la interfaz visual para actualizar una película en la base de datos |
| `public IActionResult MainMovies(int page = 1, string search_by = "Title", string order_by = "Title", string search = "")` | Recibe los valores del formulario de la interfaz visual para construir la información necesaria en este caso la lista de películas que cumple con los parámetros de búsquedas para la vista MainMovie |
| `public IActionResult MovieList()` | Genera una lista con todas la películas existentes en la base de datos para ser usada en la vista MovieList |

- ProducerController (./Controllers/ ProducerController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los actores y directores contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|

| | |
|---|---|
| `public IActionResult CreateActor(Actor obj)` | Captura un Actor creado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult CreateDirector(Director obj)` | Captura un Director creado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult UpdateDirector(Director obj)` | Captura un Director modificado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult UpdateActor(Actor obj)` | Captura un Actor modificado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult ProducerList()` | Genera una lista con todos los actores y directores existentes en la base de datos para ser usada en la vista ProducerList |

- ShowController (./Controllers/ ShowController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los Shows contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `public IActionResult Create(Show obj)` | Captura un Show creado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult Update(Show obj)` | Captura un Show modificado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult MainShowDetails(int id)` | Captura un Show desde la base de datos utilizando el *id* para buscarlo y brindárselo a la vista MainShowDetails |
| `[HttpPost]`<br>`public IActionResult MainShowDetails (IFormCollection seatsform)` | Captura el formulario desde la vista MainShowDetails con los asientos seleccionados por el usuario y los agrega a la base de datos |
| `public IActionResult ShowList()` | Genera una lista con todos los Show existentes en la base de datos para ser usada en la vista ShowList |
| `public IActionResult MainShows(int page = 1)` | Genera una lista con todos los Show existentes en la base de datos para ser usada en la vista MainShow |

- TheaterMemberController (./Controllers/TheaterMemberController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los Miembros(ViP) contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `public IActionResult MemberList()` | Genera una lista con todos los Miembros existentes en la base de datos para ser usada en la vista MemberList |
| `public IActionResult Create(IFormCollection user)` | Captura un formulario desde la interfaz visual para crear y añadir un miembro a la base de datos |

- TheaterUserController (./Controllers/TheaterUserController.cs):
  Este controlador se encarga de todas las vistas que interactúan con los usuarios contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `public IActionResult Register()` | Renderiza la vista de registro |
| `[HTTP Post]`<br>`public async Task<IActionResult> Register(RegisterViewModel model)` | Crear un usuario nuevo y lo logea. |
| `public async Task<IActionResult> Logout()` | Cierra la sesión del usuario actual |
| `public IActionResult Login()` | Renderiza la vista de inicio de sesión |
| `[HttpPost]`<br>`public async Task<IActionResult> Login(LoginViewModel model)` | Logea a un usuario que ya tenga una cuenta. |
| `public async Task<IActionResult> BecomeClubMember()` | Convierte a un usuario ya logeado en miembro del club |
| `[HttpPost]`<br>`public async Task<IActionResult> Create(TheaterUser user)` | Lista todos los usuarios de la base de datos |
| `public IActionResult UserList()` | Genera una lista de todos los usuarios existentes en la base de datos para ser utilizados en la vista UserList |
| `public IActionResult Main Profile()` | Renderiza la vista del perfil de usuario |

- TicketController (./Controllers/ TicketController.cs):

Este controlador se encarga de todas las vistas que interactúan con los Tickets contando con los siguientes métodos (funciones):

| *Función* | *Objetivo* |
|---|---|
| `public IActionResult Create(Ticket obj)` | Captura un Ticket creado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult Update(Ticket obj)` | Captura un Ticket modificado desde la interfaz visual para agregarlo a la base de datos |
| `public IActionResult TicketList()` | Genera una lista con todos los Tickets existentes en la base de datos para ser usada en la vista TicketList |
| `public IActionResult CancelTicket(int? id)` | Cancela el ticket referente al *id* |