***Informe Individual Ingeniería de Software***

***Alejandro Escobar Giraudy C-312 Movie++***

**1-Modelación de la base de datos.**

Junto con Airelys realicé esta parte del proyecto que nos fue bastante complicado, hubo que pensar y trabajar mucho pues tenemos que tener una perspectiva de lo que nos puede hacer falta en el futuro.

Después de mucho pensar se me ocurrió formar un horario donde este la Película, la Sala y la Fecha, con otras ideas que ya se tenían, Airelys llego a modelar completamente nuestra base de datos, la cual solo sufrió pequeños cambios en el camino.

Aquí dejo el código de la entidad BuyTicket que es uno de los más importantes dentro de la base de datos.

public class Buy\_Ticket : BaseEntity

{

public string ApplicationUserId { get; set; }

public int? HoraryId { get; set; }

public int? CreditCardId { get; set; }

public int? NumberOfEntrance { get; set; }

public int? Payment { get; set; }

public string Voucher { get; set; }

public string VoucherSeats { get; set; }

public string VoucherUserTypes { get; set; }

public DateTime Date { get; set; }

public bool? PayWithPoints { get; set; }

public bool? PayCompleted { get; set; }

public virtual CreditCard CreditCard { get; set; }

public virtual ApplicationUser ApplicationUser { get; set; }

public virtual Horary Horary { get; set; }

public virtual ICollection<Reserved\_Seats> Reserved\_Seats { get; set; }

}

**2-Funcionalidades de los Gerentes.**

**-Crear, editar, eliminar y listar películas, salas y horarios.**

Como sabemos el gerente es el encargado de tener actualizado el listado de películas, salas y los horarios disponibles. Para ello creamos un controlador con vistas de listar Movie, Movie\_Local y Horaries que se encargará de realizar estas funcionalidades.

Como las vistas de estos controladores solo pueden ser accesibles por el gerente y el Admin que creamos que tiene acceso a todo, encima del método Index del controlador ponemos [Authorize (Roles = “Admin, Manager”)] para solo darle permiso a estos dos usuarios.

Aquí dejo el controlador de horarios para crear, editar, eliminar y listar ya que los de películas y salas son semejantes.

public class HorariesController : Controller

{

IHoraryService \_HoraryService;

IMovieService \_MovieService;

IMovieLocalService \_MovieLocalService;

IPagerService<Horary> \_PagerService;

public HorariesController(IHoraryService HoraryService, IMovieService MovieService,

IMovieLocalService MovieLocalService, IPagerService<Horary> PagerService)

{

\_HoraryService = HoraryService;

\_MovieService = MovieService;

\_MovieLocalService = MovieLocalService;

\_PagerService = PagerService;

}

// GET: Horaries

public IActionResult Index(int id, string \_title, string \_localName, DateTime \_minDate, DateTime \_maxDate,

int \_price, int \_priceInPoints, int page = 1)

{

var horaries = \_HoraryService.GetAllHoraries()

.Include(h => h.Movie)

.Include(h => h.Movie\_Local).ToList();

if (id != 0)

horaries = horaries.Where(h => h.MovieId == id).ToList();

horaries = \_HoraryService.Filters(\_title, \_localName, \_minDate, \_maxDate, \_price, \_priceInPoints, horaries).ToList();

return View(\_PagerService.GetPager(horaries, page));

}

// GET: Horaries/Details/5

public IActionResult Details(int? id)

{

if (id == null)

{

return NotFound();

}

var horary = \_HoraryService.GetAllHoraries()

.Include(h => h.Movie)

.Include(h => h.Movie\_Local)

.FirstOrDefault(m => m.Id == id);

if (horary == null)

{

return NotFound();

}

return View(horary);

}

// GET: Horaries/Create

[Authorize(Roles = "Admin,Manager")]

public IActionResult Create()

{

ViewData["Movie"] = new SelectList(\_MovieService.GetAllMovies(), "Title", "Title");

ViewData["Movie\_Local"] = new SelectList(\_MovieLocalService.GetAllMovie\_Locals(), "Local\_Name", "Local\_Name");

return View();

}

// POST: Horaries/Create

[Authorize(Roles = "Admin,Manager")]

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Create(string Movie, string Movie\_Local, DateTime Date, DateTime Time, int Price,

int PriceInPoints, int PointsForBuying, int Id)

{

int MovieId = \_MovieService.GetAllMovies().FirstOrDefault(x => x.Title == Movie).Id;

int Movie\_LocalId = \_MovieLocalService.GetAllMovie\_Locals().FirstOrDefault(x => x.Local\_Name == Movie\_Local).Id;

if(Date.AddSeconds(Time.TimeOfDay.TotalSeconds) < DateTime.Now.AddHours(2))

ModelState.AddModelError("", "Not valid Date.");

Horary horary = new Horary()

{

MovieId = MovieId,

Movie\_LocalId = Movie\_LocalId,

Date = Date,

Time = Time,

Price = Price,

PriceInPoints = PriceInPoints,

PointsForBuying = PointsForBuying

};

if (ModelState.IsValid && \_HoraryService.DuplicateHorary(horary))

ModelState.AddModelError("", "There is already an horary with the same data.");

if (ModelState.IsValid)

{

\_HoraryService.InsertHorary(horary);

return RedirectToAction(nameof(Index));

}

ViewData["Movie"] = new SelectList(\_MovieService.GetAllMovies(), "Title", "Title", horary.MovieId);

ViewData["Movie\_Local"] = new SelectList(\_MovieLocalService.GetAllMovie\_Locals(), "Local\_Name", "Local\_Name", horary.Movie\_LocalId);

return View(horary);

}

// GET: Horaries/Edit/5

[Authorize(Roles = "Admin,Manager")]

public IActionResult Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var horary = \_HoraryService.GetHorary(id.Value);

if (horary == null)

{

return NotFound();

}

ViewData["Movie"] = new SelectList(\_MovieService.GetAllMovies(), "Title", "Title", horary.MovieId);

ViewData["Movie\_Local"] = new SelectList(\_MovieLocalService.GetAllMovie\_Locals(), "Local\_Name", "Local\_Name", horary.Movie\_LocalId);

return View(horary);

}

// POST: Horaries/Edit/5

[Authorize(Roles = "Admin,Manager")]

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Edit(int id, string Movie, string Movie\_Local, DateTime Date, DateTime Time, int Price,

int PriceInPoints, int PointsForBuying)

{

var horary = \_HoraryService.GetHorary(id);

int MovieId = \_MovieService.GetAllMovies().FirstOrDefault(x => x.Title == Movie).Id;

int Movie\_LocalId = \_MovieLocalService.GetAllMovie\_Locals().FirstOrDefault(x => x.Local\_Name == Movie\_Local).Id;

if (Date.AddSeconds(Time.TimeOfDay.TotalSeconds) < DateTime.Now.AddHours(2))

ModelState.AddModelError("", "Not valid Date.");

horary.MovieId = MovieId;

horary.Movie\_LocalId = Movie\_LocalId;

horary.Date = Date;

horary.Time = Time;

horary.Price = Price;

horary.PriceInPoints = PriceInPoints;

horary.PointsForBuying = PointsForBuying;

if (ModelState.IsValid && \_HoraryService.DuplicateHorary(horary))

ModelState.AddModelError("", "There is already an horary with the same data.");

if (ModelState.IsValid)

{

try

{

\_HoraryService.UpdateHorary(horary);

}

catch (DbUpdateConcurrencyException)

{

if (!\_HoraryService.ExistsHorary(horary.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["Movie"] = new SelectList(\_MovieService.GetAllMovies(), "Title", "Title", horary.MovieId);

ViewData["Movie\_Local"] = new SelectList(\_MovieLocalService.GetAllMovie\_Locals(), "Local\_Name", "Local\_Name", horary.Movie\_LocalId);

return View(horary);

}

// GET: Horaries/Delete/5

[Authorize(Roles = "Admin,Manager")]

public IActionResult Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var horary = \_HoraryService.GetAllHoraries()

.Include(h => h.Movie)

.Include(h => h.Movie\_Local)

.FirstOrDefault(m => m.Id == id);

if (horary == null)

{

return NotFound();

}

return View(horary);

}

// POST: Horaries/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public IActionResult DeleteConfirmed(int id)

{

var horary = \_HoraryService.GetHorary(id);

\_HoraryService.DeleteHorary(horary);

return RedirectToAction(nameof(Index));

}

}

**-Elegir el criterio usado en las sugerencias**

En una vista solo accesible por los gerentes o el admin ponemos un select donde están las cuatro opciones a escoger: más vistas, más gustadas, propagandísticas y económicas y aleatorios.

Cuando el gerente elige uno de estos criterios, en la vista de sugerencias que ven todos los usuarios van a estar listadas 10 películas correspondientes al criterio escogido.

Para hacer el aleatorio simplemente escogemos 10 películas random de la base de datos; para el resto de los criterios hacemos consultas en la base de datos, de las más vistas escogemos las 10 películas que se han comprado más entradas, de las más gustadas escogemos las 10 que más ranking(método hecho por Henri donde una persona califica la película) tienen y de las propagandísticas y económicas escogemos 10 que tengan esta característica, pues cuando el gerente crea una película marca si esta tiene este carácter propagandístico y económico.

Aquí dejo el código del controlador de sugerencias.

public class SuggestionController : Controller

{

private ISuggestionService \_SuggestionService;

private IBuyTicketService \_BuyTicketService;

private IMovieService \_MovieService;

public SuggestionController(ISuggestionService SuggestionService, IBuyTicketService BuyTicketService, IMovieService MovieService)

{

\_SuggestionService = SuggestionService;

\_BuyTicketService = BuyTicketService;

\_MovieService = MovieService;

}

[Authorize(Roles = "Admin,Manager")]

public IActionResult Index()

{

ViewData["Suggestions"] = new SelectList(Enum.GetValues(typeof(\_suggestions)).Cast<\_suggestions>().ToList());

return View();

}

[Authorize(Roles = "Basic\_User,Admin,Manager")]

public IActionResult Show(string Title)

{

if (Title != null)

{

try

{

var sugestion = \_SuggestionService.GetAny();

sugestion.Vigentsuggestion = Title;

\_SuggestionService.UpdateVigentSuggestion(sugestion);

}

catch

{

Suggestion s = new Suggestion() { Vigentsuggestion = Title };

\_SuggestionService.InsertSuggestion(s);

}

}

else

{

try

{

Title = \_SuggestionService.GetAny().Vigentsuggestion;

}

catch

{

Suggestion s = new Suggestion() { Vigentsuggestion = \_suggestions.Random.ToString() };

\_SuggestionService.InsertSuggestion(s);

}

}

ViewData["Suggestions"] = new SelectList(Enum.GetValues(typeof(\_suggestions)).Cast<\_suggestions>().ToList());

return View(\_SuggestionService.SuggestionMovies(Title, \_MovieService.GetAllMovies(), \_BuyTicketService.GetAllBuy\_Tickets()));

}

}

**-Consultar estadísticas**.

Para esto se listan las películas en dependencia de la venta de entradas que ha tenido cada una, donde luego se le aplicaran unos filtros hechos por Henri como son por título, actores, país, entre otros.

Estas estadísticas tienen la opción de ser llevadas a Pdf para que el gerente si lo desea pueda descargarlas e imprimirlas localmente. Para esto se usa el paquete Rotativa.

Aquí dejo el código del controlador.

public class EstadisticsController : Controller

{

IBuyTicketService \_BuyTicketService;

IEstadisticService \_EstadisticService;

IMovieService \_MovieService;

IPagerService<EstadisticViewModel> \_PagerService;

private static List<EstadisticViewModel> \_estadistics;

public EstadisticsController(IBuyTicketService BuyTicketService, IEstadisticService EstadisticService,

IMovieService MovieService, IPagerService<EstadisticViewModel> PagerService)

{

\_BuyTicketService = BuyTicketService;

\_EstadisticService = EstadisticService;

\_MovieService = MovieService;

\_PagerService = PagerService;

}

// GET: Estadistics

[Authorize(Roles = "Admin,Manager")]

public IActionResult Index(string \_title, string \_country, string \_kindOfMovie, int \_duration,

DateTime \_minDate, DateTime \_maxDate, string \_actor, int \_ranking, int page = 1)

{

\_estadistics = new List<EstadisticViewModel>();

var \_movieIds = \_EstadisticService.MovieIds(\_minDate, \_maxDate, \_BuyTicketService.GetAllBuy\_Tickets());

foreach (var item in \_movieIds)

{

var \_movie = \_MovieService.GetMovie(item.Item1);

EstadisticViewModel e = new EstadisticViewModel()

{

Id = \_movie.Id,

Title = \_movie.Title,

Duration = \_movie.Duration,

KindOfMovie = \_movie.KindOfMovie,

Country = \_movie.Country,

Actors = \_movie.Actors,

Ranking = \_movie.Ranking,

TotalOfEntrance = item.Item2

};

\_estadistics.Add(e);

}

\_estadistics = \_EstadisticService.Filters(\_title, \_country, \_kindOfMovie, \_duration, \_actor, \_ranking, \_estadistics).ToList();

return View(\_PagerService.GetPager(\_estadistics, page));

}

public IActionResult ViewPDF()

{

return View(\_estadistics);

}

public IActionResult PDF()

{

return new ViewAsPdf("ViewPDF", \_estadistics)

{

FileName = "Estadistic.pdf",

PageOrientation = Rotativa.AspNetCore.Options.Orientation.Portrait,

PageSize = Rotativa.AspNetCore.Options.Size.A4

};

}

}

En el informe técnico realicé el modelo de datos y desarrollé los patrones de datos y visualización.