***Informe Individual Ingeniería de Software***

***David Cabrera García C-31 Movie++***

**1-Autenticación y Roles.**

Esta tarea la llevé a cabo junto con Henri, yo hice para que cada usuario que se registre tenga por defecto el rol de Basic\_User. También puse para que un usuario pusiera más datos a la hora de registrarse.

En lo siguiente podemos ver todos los campos que se piden a la hora de registrarse.

public class InputModel

{

[Required]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Confirm password")]

[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

}

Y como puse para que cada usuario tenga por defecto Basic\_User, se puede ver en la última línea del código siguiente.

var result = await \_userManager.CreateAsync(user, Input.Password);

if (result.Succeeded)

{

\_logger.LogInformation("User created a new account with password.");

await \_userManager.AddToRoleAsync(user, Roles.Roles.Basic\_User.ToString());

**2-Tipos de descuentos**

También sembré en la base de datos un usuario normal que tiene descuento 0, ya después el manager puede añadir otros criterios para los diferentes descuentos, pero tenemos ese por defecto.

public static async Task SeedNormalUserType(ApplicationDbContext \_context)

{

if (\_context.UserType.Count() == 0)

{

var \_normalUser = new UserType() { Type = "Normal User", Discount = 0 };

await \_context.AddAsync(\_normalUser);

await \_context.SaveChangesAsync();

}

}

Para añadir el resto de los tipos de descuentos para los diferentes usuarios hice el siguiente controlador.

public class UserTypesController : Controller

{

IUserTypesService \_UserTypesService;

public UserTypesController(IUserTypesService UserTypesService)

{

\_UserTypesService = UserTypesService;

}

// GET: UserTypes

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

public IActionResult Index()

{

return View(\_UserTypesService.GetAllUserTypes().ToList());

}

// GET: UserTypes/Details/5

public IActionResult Details(int? id)

{

if (id == null)

{

return NotFound();

}

var userType = \_UserTypesService.GetUserType(id.Value);

if (userType == null)

{

return NotFound();

}

return View(userType);

}

// GET: UserTypes/Create

public IActionResult Create()

{

return View();

}

// POST: UserTypes/Create

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Create([Bind("Type,Discount,Id")] UserType userType)

{

if (\_UserTypesService.DuplicateUserType(userType))

ModelState.AddModelError("", "There is already a UserType with the same type");

if (ModelState.IsValid)

{

\_UserTypesService.InsertUserType(userType);

return RedirectToAction(nameof(Index));

}

return View(userType);

}

// GET: UserTypes/Edit/5

public IActionResult Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var userType = \_UserTypesService.GetUserType(id.Value);

if (userType == null)

{

return NotFound();

}

return View(userType);

}

// POST: UserTypes/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Edit(int id, [Bind("Type,Discount,Id")] UserType userType)

{

if (id != userType.Id)

{

return NotFound();

}

if (\_UserTypesService.DuplicateUserType(userType))

ModelState.AddModelError("", "There is already a UserType with the same type");

if (ModelState.IsValid)

{

try

{

\_UserTypesService.UpdateUserTypes(userType);

}

catch (DbUpdateConcurrencyException)

{

if (!\_UserTypesService.ExistsUserType(userType.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(userType);

}

// GET: UserTypes/Delete/5

public IActionResult Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var userType = \_UserTypesService.GetUserType(id.Value);

if (userType == null)

{

return NotFound();

}

return View(userType);

}

// POST: UserTypes/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public IActionResult DeleteConfirmed(int id)

{

var userType = \_UserTypesService.GetUserType(id);

\_UserTypesService.DeleteUserType(userType);

return RedirectToAction(nameof(Index));

}

}

**3-Cancelar compra.**

El usuario va a tener una opción donde podrá ver todas las compras que ha hecho y consultar sus comprobantes, pero también tendrá la opción de cancelar sus compras hasta dos horas antes del comienzo de la misma. Para ello, desarrollé el controlador CancelBuy con los siguientes métodos.

public IActionResult Index(string \_code, string \_title, string \_localMovie, DateTime \_minDate, DateTime \_maxDate, int page = 1)

{

var user = \_UserService.GetUser(\_UserManager.GetUserId(User));

var userBuys = \_BuyTicketService.GetAllBuy\_Tickets()

.Include(u => u.Horary)

.Include(u => u.Horary.Movie\_Local)

.Include(u => u.Horary.Movie)

.Where(m => m.ApplicationUserId == user.Id && m.PayCompleted == true).ToList();

userBuys = \_CancelBuyService.Filters(\_code, \_title, \_localMovie, \_minDate, \_maxDate, userBuys).ToList();

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

}

public IActionResult Cancel(int id)

{

var Buy = \_BuyTicketService.GetAllBuy\_Tickets()

.Include(u => u.Horary)

.Include(u => u.CreditCard)

.Include(u => u.ApplicationUser)

.Include(u => u.Horary.Movie\_Local)

.Include(u => u.Horary.Movie)

.Include(u => u.Reserved\_Seats)

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

Buy.Horary.ReservedTickets -= (int)Buy.NumberOfEntrance;

\_HoraryService.UpdateHorary(Buy.Horary);

\_ReservedSeatsService.RemoveReservedSeats(Buy.Reserved\_Seats);

if (Buy.PayWithPoints.Value)

{

Buy.ApplicationUser.Puntuation += Buy.Payment;

\_UserService.UpdateUser(Buy.ApplicationUser);

}

else

{

Buy.CreditCard.Money += (int)Buy.Payment;

\_CreditCardService.UpdateCreditCardMoney(Buy.CreditCard);

}

\_BuyTicketService.RemoveBuyTicket(Buy);

TempData["Success"] = "Buy Canceled";

return RedirectToAction(nameof(Index));

}

En el informe técnico del equipo desarrollé los requerimientos funcionales.