

## TERCER RELEASE (SPRINT 5 y 6)

1. E1-HU4: Asignar entrenador a un cliente premium.

2. E2-HU2: Consultar información personal de los entrenadores del gimnasio

3. E2-HU3: Consultar información personal de un entrenador del gimnasio

ID	Componente	Valores	Tipo (+/-)	Resultado Esperado	Criticidad	Resultado Real
E1-HU4 - CA1	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExiste = consultaEntrenador.EntrenadorExiste(identificacionEntrenador);     if (!entrenadorExiste)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = null		El sistema deberá de mostrar un mensaje diciendo que la identificación no puede estar vacía.	Media	<pre> if (string.IsNullOrWhiteSpace(identificacionEntrenador)) {     TempData["ErrorMessage"] = "La identificación no puede estar vacia.";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1-HU4 - CA2	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExiste = consultaEntrenador.EntrenadorExiste(identificacionEntrenador);     if (!entrenadorExiste)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = 23213		El sistema mostrará un mensaje diciendo que el entrenador no existe.	Media	<pre> if (!entrenadorExiste) {     TempData["ErrorMessage"] = "Entrenador no existente";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1-HU4 - CA3	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExiste = consultaEntrenador.EntrenadorExiste(identificacionEntrenador);     if (!entrenadorExiste)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = 1		El sistema mostrará un mensaje diciendo que el entrenador no está disponible.	Media	<pre> if (identificacionEntrenador != null) {     TempData["ErrorMessage"] = "Entrenador no disponible";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1-HU4 - CA4	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExiste = consultaEntrenador.EntrenadorExiste(identificacionEntrenador);     if (!entrenadorExiste)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = 2		El sistema mostrará un mensaje exitoso de registro.	Media	<pre> if (auxRegistrarPersona == true) {     var auxRegistrarCliente = consultaCliente.RegistrarCliente(auxCliente);     TempData["ClienteDatos"] = null;     TempData["SuccessMessage"] = "Cliente registrado correctamente";     return RedirectToAction("DashboardAdministrador", "Admin"); } </pre>

E1- HU4 - CA5	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);     if (!entrenadorExistente)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varid = 0; varid &lt; entrenadoresDisponibles.Count; varid++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors =         ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = -1232	El sistema deberá de mostrar un mensaje advirtiendog que el numero debe de ser positivo.	Media	<pre> if (identificacionNum &lt; 0) {     TempData["ErrorMessage"] = "La identificación debe ser un número positivo.";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1- HU4 - CA6	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);     if (!entrenadorExistente)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varid = 0; varid &lt; entrenadoresDisponibles.Count; varid++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors =         ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = 102002323	El sistema deberá de mostrar un mensaje advirtiendog que el numero no debe de tener más de 10 dígitos.	Media	<pre> // verificar que la identificación no tenga más de 10 dígitos if (identificacionEntrenador.Length &gt; 10) {     TempData["ErrorMessage"] = "La identificación no puede tener más de 10 dígitos.";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1- HU4 - CA7	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);     if (!entrenadorExistente)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varid = 0; varid &lt; entrenadoresDisponibles.Count; varid++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors =         ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	ID = ;??!	El sistema deberá de mostrar un mensaje diciendog que la identificación debe ser un número válido.	Media	<pre> // Verificar si la identificación es numérica y convertirla a entero if (!int.TryParse(identificacionEntrenador, out int identificacionNum)) {     TempData["ErrorMessage"] = "La identificación debe ser un número válido.";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); } </pre>
E1- HU4 - CA8	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);     if (!entrenadorExistente)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varid = 0; varid &lt; entrenadoresDisponibles.Count; varid++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors =         ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	Recargar la página	El sistema vuelve a llamar a la función de listar entrenadores disponibles	Baja	El sistema vuelve a llamar a la función de listar entrenadores disponibles
E1- HU4 - CA9	<pre> public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     if (string.IsNullOrWhiteSpace(identificacionEntrenador))     if (identificacionEntrenador.Length &gt; 10)     if (int.TryParse(identificacionEntrenador, out int identificacionNum))     if (identificacionNum &lt; 0)     bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);     if (!entrenadorExistente)     List&lt;EntrenadorModel&gt; entrenadoresDisponibles = ObtenerEntrenadoresDisponibles();     for (int varid = 0; varid &lt; entrenadoresDisponibles.Count; varid++)     if (identificacionEntrenador != null)     return Json(new { success = false, errors =         ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) }); } </pre>	Salir del modulo	El sistema no guarda al cliente en la base de datos hasta que no se le asigne un entrenador	Baja	El sistema guarda la información previa en el módulo de registrar cliente para poder seguir con el proceso, no se guarda hasta asignar correctamente al entrenador.

E1- HU4 - CA10	<pre>public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     {         if (string.IsNullOrWhiteSpace(identificacionEntrenador))         {             if (identificacionEntrenador.Length &gt; 10)             {                 if (int.TryParse(identificacionEntrenador, out int identificacionNum))                 {                     if (identificacionNum &lt; 0)                     {                         bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);                         if (!entrenadorExistente)                         {                             List&lt;EntrenadorModel&gt; entrenadoresDisponibles = obtenerEntrenadoresDisponibles();                             for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)                             {                                 if (identificacionEntrenador != null)                                 {                                     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) });                                 }                             }                         }                     }                 }             }         }     } }</pre>	Presiona ESC	El sistema no hace nada, sigue el flujo correctamente.	Baja	El sistema no realiza ninguna accion por presionar ESC																					
E1- HU4 - CA11	<pre>public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     {         if (string.IsNullOrWhiteSpace(identificacionEntrenador))         {             if (identificacionEntrenador.Length &gt; 10)             {                 if (int.TryParse(identificacionEntrenador, out int identificacionNum))                 {                     if (identificacionNum &lt; 0)                     {                         bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);                         if (!entrenadorExistente)                         {                             List&lt;EntrenadorModel&gt; entrenadoresDisponibles = obtenerEntrenadoresDisponibles();                             for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)                             {                                 if (identificacionEntrenador != null)                                 {                                     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) });                                 }                             }                         }                     }                 }             }         }     } }</pre>	ID = 130	El sistema deberá de mostrar un mensaje de que el entrenador no existe.	Media	<pre>if (!entrenadorExistente) {     TempData["ErrorMessage"] = "Entrenador no existente";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); }</pre>																					
E1- HU4 - CA12	<pre>public ActionResult AsignarEntrenadorCliente(string identificacionEntrenador) {     var a = auxCliente;     if (auxCliente.IdEntrenador != 0)     {         if (string.IsNullOrWhiteSpace(identificacionEntrenador))         {             if (identificacionEntrenador.Length &gt; 10)             {                 if (int.TryParse(identificacionEntrenador, out int identificacionNum))                 {                     if (identificacionNum &lt; 0)                     {                         bool entrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacionEntrenador);                         if (!entrenadorExistente)                         {                             List&lt;EntrenadorModel&gt; entrenadoresDisponibles = obtenerEntrenadoresDisponibles();                             for (int varId = 0; varId &lt; entrenadoresDisponibles.Count; varId++)                             {                                 if (identificacionEntrenador != null)                                 {                                     return Json(new { success = false, errors = ModelState.ToDictionary(k =&gt; k.Key, v =&gt; v.Value.Errors.Select(e =&gt; e.ErrorMessage).ToArray()) });                                 }                             }                         }                     }                 }             }         }     } }</pre>	ID = 2	El sistema deberá de mostrar un mensaje : ""El cliente ya tiene un entrenador asignado".	Media	<pre>if (auxCliente.IdEntrenador != 0) {     TempData["ErrorMessage"] = "El cliente ya tiene un entrenador asignado";     return RedirectToAction("AsignarEntrenadorCliente", "Cliente"); }</pre>																					
E2-HU2-CA1	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return Json(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Todos"	Una lista de EntrenadorModel con 3 entrenadores creados previamente.	Media	<table><tr><td>filter</td><td>"all"</td><td>Q View</td></tr><tr><td>entrenadoresFiltrados</td><td>Count = 3</td><td>Q View</td></tr><tr><td>[0]</td><td>(NET_MVC.Models.EntrenadorModel)</td><td></td></tr><tr><td>[1]</td><td>(NET_MVC.Models.EntrenadorModel)</td><td></td></tr><tr><td>[2]</td><td>(NET_MVC.Models.EntrenadorModel)</td><td></td></tr><tr><td>Raw View</td><td></td><td></td></tr><tr><td>IdSede</td><td>"1"</td><td>Q View</td></tr></table>	filter	"all"	Q View	entrenadoresFiltrados	Count = 3	Q View	[0]	(NET_MVC.Models.EntrenadorModel)		[1]	(NET_MVC.Models.EntrenadorModel)		[2]	(NET_MVC.Models.EntrenadorModel)		Raw View			IdSede	"1"	Q View
filter	"all"	Q View																								
entrenadoresFiltrados	Count = 3	Q View																								
[0]	(NET_MVC.Models.EntrenadorModel)																									
[1]	(NET_MVC.Models.EntrenadorModel)																									
[2]	(NET_MVC.Models.EntrenadorModel)																									
Raw View																										
IdSede	"1"	Q View																								
E2-HU2-CA2	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return Json(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Crossfit"	Una lista de EntrenadorModel con 1 entrenador.	Media	<table><tr><td>this</td><td>(NET_MVC.Controllers.EntrenadorController)</td><td></td></tr><tr><td>filter</td><td>"crossfit"</td><td>Q View</td></tr><tr><td>entrenadoresFiltrados</td><td>Count = 1</td><td>Q View</td></tr><tr><td>IdSede</td><td>"1"</td><td>Q View</td></tr></table>	this	(NET_MVC.Controllers.EntrenadorController)		filter	"crossfit"	Q View	entrenadoresFiltrados	Count = 1	Q View	IdSede	"1"	Q View									
this	(NET_MVC.Controllers.EntrenadorController)																									
filter	"crossfit"	Q View																								
entrenadoresFiltrados	Count = 1	Q View																								
IdSede	"1"	Q View																								
	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;(); }</pre>				<table><tr><td>this</td><td>(NET_MVC.Controllers.EntrenadorController)</td><td></td></tr></table>	this	(NET_MVC.Controllers.EntrenadorController)																			
this	(NET_MVC.Controllers.EntrenadorController)																									

E2-HU2-CA3	<pre>String IdSede = User.FindFirst(ClaimTypes.Name)?.Value; entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);  return JsonResult(new {     entrenadores = entrenadoresFiltrados,     totalEntrenadores = entrenadoresFiltrados.Count });</pre>	Filtro = "Fuerza"		Una lista de EntrenadorModel con 2 entrenadores.	Media	<div> <div> this filter entrenadoresFiltrados IdSede </div> <div> [NET_MVC.Controllers.EntrenadorController] "fuerza" Count = 2 "1" </div> <div> Q View Q View Q View Q View </div> </div>
E2-HU2-CA4	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return JsonResult(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Reducción de peso"		Una lista de EntrenadorModel vacia.	Media	<div> <div> NET_MVC.Datos.Adm... this filter entrenadoresFiltrados IdSede </div> <div> Count = 0 [NET_MVC.Controllers.EntrenadorController] "reduccion" Count = 0 "1" </div> <div> Q View Q View Q View Q View </div> </div>
E2-HU2-CA5	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return JsonResult(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Culturismo"		Una lista de EntrenadorModel vacia.	Media	<div> <div> Name Value this filter entrenadoresFiltrados IdSede </div> <div> [NET_MVC.Controllers.EntrenadorController] "culturismo" Count = 2 "1" </div> <div> Q View Q View Q View Q View </div> </div>
E2-HU2-CA6	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return JsonResult(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Género Masculino"		Una lista de EntrenadorModel con 1 entrenador.	Media	<div> <div> NET_MVC.Datos.Adm... this filter entrenadoresFiltrados IdSede </div> <div> Count = 1 [NET_MVC.Controllers.EntrenadorController] "masculino" Count = 0 "1" </div> <div> Q View Q View Q View Q View </div> </div>
E2-HU2-CA7	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return JsonResult(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Género Femenino"		Una lista de EntrenadorModel con 2 entrenadores.	Media	<div> <div> this filter entrenadoresFiltrados IdSede </div> <div> [NET_MVC.Controllers.EntrenadorController] "femenino" Count = 2 "1" </div> <div> Q View Q View Q View Q View </div> </div>
E2-HU2-CA8	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return JsonResult(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); }</pre>	Filtro = "Género no especificado"		Una lista de EntrenadorModel vacia.	Media	<div> <div> this filter entrenadoresFiltrados IdSede </div> <div> [NET_MVC.Controllers.EntrenadorController] "no-especificado" Count = 0 "1" </div> <div> Q View Q View Q View Q View </div> </div>

	<pre> entrenadores = entrenadoresFiltrados, totalEntrenadores = entrenadoresFiltrados.Count }); </pre>				
E2-HU2-CA9	<pre> [Authorize(Roles = "Administrador")] [HttpPost] // references public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return Json(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); } </pre>	Recargar página	El sistema hace el llamado a las funciones que dependen de la consulta de entrenadores ya que pudo haber una nueva actualización de entrenadores disponibles.	Baja	Al recargar la página en cualquier filtro, el sistema hace el llamado nuevamente a las funciones dependientes del filtro y vuelve a cargar la información
E2-HU2-CA10	<pre> [Authorize(Roles = "Administrador")] [HttpPost] // references public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return Json(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); } </pre>	Salir del módulo	El sistema borrará el cache de los entrenadores buscados	Baja	El sistema borra el cache de los entrenadores buscados
E2-HU2-CA11	<pre> [Authorize(Roles = "Administrador")] [HttpPost] // references public JsonResult Filtrar(string filter) {     List&lt;EntrenadorModel&gt; entrenadoresFiltrados = new List&lt;EntrenadorModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     entrenadoresFiltrados = consultaEntrenador.ListarEntrenadores(IdSede, filter);      return Json(new     {         entrenadores = entrenadoresFiltrados,         totalEntrenadores = entrenadoresFiltrados.Count     }); } </pre>	Tecla [ESC]	El sistema sigue su flujo normal, no se saldrá de la página o algún otro comportamiento.	Baja	Al Presionar ESC en cualquier filtro, el sistema no realiza ninguna acción.
E2-HU3-CA1	<pre> [Authorize(Roles = "Administrador")] [HttpPost] // references public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     if (string.IsNullOrEmpty(identificacion))     {         if (int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         // Verificar que la identificación no esté vacía                         if (string.IsNullOrEmpty(identificacion))                         {                             TempData["ErrorMessage"] = "La identificación no puede estar vacía.";                             return RedirectToAction("InformacionEntrenador");                         }                     }                 }             }         }     } } </pre>	ID = null	El sistema deberá de mostrar un mensaje diciendo que la identificación no puede estar vacía.	Media	<pre> // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(identificacion)) {     TempData["ErrorMessage"] = "La identificación no puede estar vacía.";     return RedirectToAction("InformacionEntrenador"); } </pre>
E2-HU3-CA2	<pre> [Authorize(Roles = "Administrador")] [HttpPost] // references public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     if (string.IsNullOrEmpty(identificacion))     {         if (int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         // Verificar que la identificación no esté vacía                         if (string.IsNullOrEmpty(identificacion))                         {                             TempData["ErrorMessage"] = "La identificación no puede estar vacía.";                             return RedirectToAction("InformacionEntrenador");                         }                     }                 }             }         }     } } </pre>	ID = 23242	El sistema deberá de mostrar un mensaje diciendo "Entrenador no encontrado"	Media	<pre> else {     TempData["ErrorMessage"] = "Entrenador no encontrado.";     return RedirectToAction("InformacionEntrenador"); } </pre>

E2-HU3-CA3	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>	ID = 1		El sistema mostrará la información del entrenador.	Media	<pre> // Verificar si se pudo obtener la información del cliente if (entrenador != null) {     return View("InformacionEntrenadorEspecifico", entrenador); } else {     return View("EliminarEjercicio"); } </pre>
E2-HU3-CA4	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>	ID = 232345345634		El sistema mostrará un mensaje diciendo que el número debe ser menor a 10 dígitos.	Media	<pre> // Verificar que la longitud no sea mayor a 10 dígitos if (ejercicio.IdEjercicio.Length &gt; 10) {     TempData["ErrorMessage"] = "La identificación no puede tener más de 10 dígitos.";     return View("EliminarEjercicio"); } </pre>
E2-HU3-CA5	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>	ID = asdasdr		El sistema mostrará un mensaje diciendo que el número debe de ser válido.	Media	<pre> // Verificar si la identificación es numérica if (!int.TryParse(identificacion, out int identificacionNumero)) {     TempData["ErrorMessage"] = "Debe ser un número válido.";     return RedirectToAction("InformacionEntrenador"); } </pre>
E2-HU3-CA6	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>	ID = -233453		El sistema mostrará un mensaje diciendo que el número de identificación debe de ser positivo	Media	<pre> // Verificar que la identificación sea un número positivo if (identificacionNumero &lt;= 0) {     TempData["ErrorMessage"] = "Debe ser un número positivo.";     return RedirectToAction("InformacionEntrenador"); } </pre>
E2-HU3-CA7	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>	Recargar la página		El sistema deberá de llamar a la funcionalidad nuevamente para actualizar la información del entrenador buscado.	Baja	<pre> // Verificar si se pudo obtener la información del cliente if (entrenador != null) {     return View("InformacionEntrenadorEspecifico", entrenador); } else {     return View("EliminarEjercicio"); } </pre>
	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name).Value;     if (string.IsNullOrWhiteSpace(identificacion))     {         if (!int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         return View("InformacionEntrenadorEspecifico", entrenador);                     }                     else                     {                         return View("EliminarEjercicio");                     }                 }             }         }     } } </pre>					

E2-HU3-CA8	<pre> public ActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     if (string.IsNullOrEmpty(identificacion))     {         if (int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         // ...                     }                     else                     {                         // ...                     }                 }             }         }     } } </pre>	Salir del módulo		El sistema deberá de eliminar la información que se tenga.	Baja	El sistema borra el cache de la información buscada previamente.
E2-HU3-CA9	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public IActionResult BuscarEntrenador(string identificacion) {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     if (string.IsNullOrEmpty(identificacion))     {         if (int.TryParse(identificacion, out int identificacionNumero))         {             if (identificacion.Length &gt; 10)             {                 if (identificacionNumero &lt;= 0)                 {                     bool EntrenadorExistente = consultaEntrenador.EntrenadorExistente(identificacion);                     if (EntrenadorExistente)                     {                         // ...                     }                     else                     {                         // ...                     }                 }             }         }     } } </pre>	Presionar ESC		El sistema no deberá de realizar ninguna acción.	Baja	Al Presionar ESC el sistema no realiza ninguna acción.
E3-HU1-CA1	<pre> public ActionResult ListarClientesAsignados(string identificador) {     var clientes = new List&lt;ClienteModel&gt;();     try     {         if (Conexion.abrirConexion())         {             using (OracleCommand cmd = new OracleCommand("pkg_Procedimientos.LISTAR_CLIENTES_ASSIGNADOS", conexionMD))             {                 cmd.CommandType = CommandType.StoredProcedure;                 cmd.Parameters.Add("ID_ENTRENADOR", OracleDbType.Int32).Value = Convert.ToInt32(identificador);                 cmd.Parameters.Add("CONexion", OracleDbType.RefCursor).Direction = ParameterDirection.Output;                 using (OracleDataAdapter reader = cmd.ExecuteReader())                 {                     while (reader.Read())                     {                         ClienteModel objCliente = new ClienteModel();                         clientes.Add(objCliente);                     }                 }             }         }     }     return clientes; } </pre>	ID = null		El sistema deberá de advertir que no se permiten campos vacíos.	Media	<pre> if (string.IsNullOrEmpty(identificacion)) {     TempData["ErrorMessage"] = User.IsInRole("Administrador") ? "Por favor diligenciar los campos marcados como obligatorios." : "La identificación no puede estar vacía.";     return RedirectToAction("InformacionCliente"); } </pre>
E3-HU1-CA2	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt;= 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             // ...                         }                         else                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = 12212		El sistema mandará un mensaje de que el cliente no existen en el sistema.	Media	<pre> if (!clienteExiste) {     TempData["ErrorMessage"] = User.IsInRole("Administrador") ? "Cliente no encontrado." : "El cliente no existe en el sistema.";     return RedirectToAction("InformacionCliente"); } </pre>
E3-HU1-CA3	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt;= 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             // ...                         }                         else                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = 23222322		El sistema mandará un mensaje de que el cliente no está asignado a el entrenador.	Media	<pre> else {     TempData["ErrorMessage"] = "El cliente no está asignado a este entrenador."; } </pre>
E3-HU1-CA4	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     // ...                 }             }         }     } } </pre>	ID = 106		El sistema mostrará la información del cliente buscado.	Media	<pre> if (clienteAsignadoResult) {     TempData["ClienteId"] = identificacion;     return View("InformacionClienteAsignado", cliente); // Mostrar la información del cliente } else {     // ... } </pre>

	<pre> if (int.TryParse(identificacion, out _)) {     if (int.Parse(identificacion) &lt; 0)     {         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);         if (clienteExiste)         {             return RedirectToAction("InformacionCliente");         }     } } </pre>					<pre> TempData["ErrorMessage"] = "El cliente no está asignado a este entrenador."; } </pre>
E3-HU1-CA5	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt; 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = sdads		El sistema mandará un mensaje "La identificación debe ser un número válido"	Media	<pre> if (int.TryParse(identificacion, out _)) {     TempData["ErrorMessage"] = "La identificación debe ser un número válido.";     return RedirectToAction("InformacionCliente"); } </pre>
E3-HU1-CA6	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt; 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = 1212121212232232		El sistema mostrará una advertencia de que no puede tener un ID más de 10 dígitos.	Media	<pre> if (identificacion.Length &gt; 10) {     TempData["ErrorMessage"] = "La identificación no puede tener más de 10 dígitos.";     return RedirectToAction("InformacionCliente"); } </pre>
E3-HU1-CA7	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt; 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = -23232		El sistema mostrará una advertencia de que no puede tener un ID negativo.	Media	<pre> if (int.Parse(identificacion) &lt; 0) {     TempData["ErrorMessage"] = "La identificación debe ser un número positivo.";     return RedirectToAction("InformacionCliente"); } </pre>
E3-HU1-CA8	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt; 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	Recargar la página		El sistema deberá de volver a llamar a la función de buscar el cliente	Baja	<pre> if (User.IsInRole("Administrador")) {     return View("InformacionClienteEspecifico", cliente); } </pre>
E3-HU1-CA9	<pre> [HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") != null &amp;&amp; User.IsInRole("Entrenador"))     {         String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         {             if (identificacion.Length &gt; 10)             {                 if (int.TryParse(identificacion, out _))                 {                     if (int.Parse(identificacion) &lt; 0)                     {                         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);                         if (clienteExiste)                         {                             return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	Salir del modulo		El sistema deberá de eliminar el cache de la información buscada.	Baja	El sistema deberá de eliminar el cache de la información buscada.



E3-HU1-CA10	<pre>[HttpGet] [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public IActionResult BuscarCliente(string identificacion) {     if (HttpContext.Session.GetString("ClienteIdEjercicio") !=         null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdSede = User.FindFirst(ClaimTypes.Name)?.Value;         if (string.IsNullOrEmpty(identificacion))         if (identificacion.Length &gt; 10)         if (!int.TryParse(identificacion, out _))         if (int.Parse(identificacion) &lt; 0)         bool clienteExiste = consultaCliente.ClienteExiste(identificacion);         if (clienteExiste)         else         return RedirectToAction("InformacionCliente");     } }</pre>	Presionar ESC		El sistema no debería de hacer ninguna acción.	Baja	El sistema no realiza ninguna acción, continua su flujo.
-------------	--	---------------	--	--	------	--