

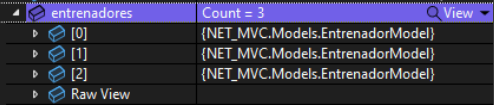
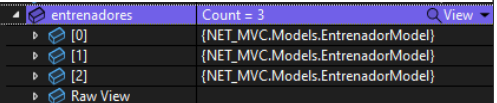
## SEGUNDO RELEASE (SPRINT 3 Y 4)

1. E1 - HU3: CONSULTAR INFORMACION DE LOS ENTRENADORES DISPONIBLES.

2. E1 - HU7: CONSULTAR ESTADISTICAS SOBRE LOS CLIENTES.

3. E1 - HU5: CONSULTAR INFORMACION PERSONAL DE LOS CLIENTES DEL GIMNASIO

4. E1 - HU6: CONSULTAR INFORMACION PERSONA DE UN CLIENTE DEL GIMANSIO.

ID	Componente	Valores	Tipo (+/-)	Resultado Esperado	Criticidad	Resultado Real
E1-HU3-CA1	<pre>[Authorize(Roles = "Administrador")] references private List&lt;EntrenadorModel&gt; ObtenerEntrenadoresDisponibles() {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     var entrenadores = consultaEntrenador.ListarEntrenadoresDisponibles(IdSede);      try     {         if (entrenadores.Count() &gt; 0)         {             return entrenadores;         }         else         {             // Mensaje no hay entrenadores disponibles             return new List&lt;EntrenadorModel&gt; { };         }     }     catch (Exception ex)     {         // Mensaje error         return new List&lt;EntrenadorModel&gt; { };     } }</pre>			Una lista de EntrenadorModel con 3 entrenadores creados previamente que no superan los 5 clientes asignados.	Alta	
E1-HU3-CA2	<pre>[Authorize(Roles = "Administrador")] references private List&lt;EntrenadorModel&gt; ObtenerEntrenadoresDisponibles() {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     var entrenadores = consultaEntrenador.ListarEntrenadoresDisponibles(IdSede);      try     {         if (entrenadores.Count() &gt; 0)         {             return entrenadores;         }         else         {             // Mensaje no hay entrenadores disponibles             return new List&lt;EntrenadorModel&gt; { };         }     }     catch (Exception ex)     {         // Mensaje error         return new List&lt;EntrenadorModel&gt; { };     } }</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	
E1-HU3-CA3	<pre>[Authorize(Roles = "Administrador")] references private List&lt;EntrenadorModel&gt; ObtenerEntrenadoresDisponibles() {     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;     var entrenadores = consultaEntrenador.ListarEntrenadoresDisponibles(IdSede);      try     {         if (entrenadores.Count() &gt; 0)         {             return entrenadores;         }         else         {             // Mensaje no hay entrenadores disponibles             return new List&lt;EntrenadorModel&gt; { };         }     }     catch (Exception ex)     {         // Mensaje error         return new List&lt;EntrenadorModel&gt; { };     } }</pre>	Salir del módulo		El sistema deberá de eliminar el cache de la información buscada.	Baja	El sistema deberá de eliminar el cache de la información buscada.

E1-HU3-CA4		Tecla [ESC]		El sistema sigue su flujo normal, no se saldrá de la página o algún otro comportamiento.	Baja	No se detectaron cambios dentro del sistema.
E1 - HU7 - CA1	<pre>[Authorize(Roles = "Administrador")] 0 references public IActionResult DashboardAdministrador() {     AdmCliente consultaCliente = new AdmCliente();     AdmEntrenador consultaEntrenador = new AdmEntrenador();     AdmPersona consultaPersona = new AdmPersona();     string usuarioActual = User.Identity.Name;     int idSede = consultaPersona.ObtenerIdSedePorUsuario(usuarioActual);     ViewBag.TotalClientes = consultaCliente.ObtenerTotalClientesPorSede(idSede);     ViewBag.TotalEntrenadores = consultaEntrenador.ObtenerTotalEntrenadoresPorSede(idSede);     ViewBag.TotalPersonas = consultaPersona.ObtenerTotalPersonasPorSede(idSede);     ViewBag.ClientesPremium = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "Premium");     ViewBag.ClientesGeneral = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "General");     return View(); }</pre>		Estadísticas de clientes, entrenadores y personas por sede.	Media	<div><div><div>Microsoft.AspNetCo...Count = 3</div><div>NET_MVC.Datos.Ad...16</div><div>System.Func&lt;T1, T2,...16</div></div><div><div>this{NET_MVC.Controllers.AdminController}</div><div>consultaCliente{NET_MVC.Datos.AdmCliente}</div><div>consultaEntrenador{NET_MVC.Datos.AdmEntrenador}</div><div>consultaPersona{NET_MVC.Datos.AdmPersona}</div><div>usuarioActual"1"</div></div><div>Q View</div></div>	
E1 - HU7 - CA2	<pre>[Authorize(Roles = "Administrador")] 0 references public IActionResult DashboardAdministrador() {     AdmCliente consultaCliente = new AdmCliente();     AdmEntrenador consultaEntrenador = new AdmEntrenador();     AdmPersona consultaPersona = new AdmPersona();     string usuarioActual = User.Identity.Name;     int idSede = consultaPersona.ObtenerIdSedePorUsuario(usuarioActual);     ViewBag.TotalClientes = consultaCliente.ObtenerTotalClientesPorSede(idSede);     ViewBag.TotalEntrenadores = consultaEntrenador.ObtenerTotalEntrenadoresPorSede(idSede);     ViewBag.TotalPersonas = consultaPersona.ObtenerTotalPersonasPorSede(idSede);     ViewBag.ClientesPremium = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "Premium");     ViewBag.ClientesGeneral = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "General");     return View(); }</pre>		Estadísticas de clientes, entrenadores y personas por sede.	Media	<div><div><div>Microsoft.AspNetCo...Count = 3</div><div>NET_MVC.Datos.Ad...16</div><div>System.Func&lt;T1, T2,...16</div></div><div><div>this{NET_MVC.Controllers.AdminController}</div><div>consultaCliente{NET_MVC.Datos.AdmCliente}</div><div>consultaEntrenador{NET_MVC.Datos.AdmEntrenador}</div><div>consultaPersona{NET_MVC.Datos.AdmPersona}</div><div>usuarioActual"1"</div></div><div>Q View</div></div>	
E1 - HU7 - CA3	<pre>[Authorize(Roles = "Administrador")] 0 references public IActionResult DashboardAdministrador() {     AdmCliente consultaCliente = new AdmCliente();     AdmEntrenador consultaEntrenador = new AdmEntrenador();     AdmPersona consultaPersona = new AdmPersona();     string usuarioActual = User.Identity.Name;     int idSede = consultaPersona.ObtenerIdSedePorUsuario(usuarioActual);     ViewBag.TotalClientes = consultaCliente.ObtenerTotalClientesPorSede(idSede);     ViewBag.TotalEntrenadores = consultaEntrenador.ObtenerTotalEntrenadoresPorSede(idSede);     ViewBag.TotalPersonas = consultaPersona.ObtenerTotalPersonasPorSede(idSede);     ViewBag.ClientesPremium = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "Premium");     ViewBag.ClientesGeneral = consultaCliente.ObtenerTotalClientesPorTipo(idSede, "General");     return View(); }</pre>	Minimizar pantalla	El sistemá deberá de seguir mostrando las gráficas sin ningun cambio o actualización.	Baja	El sistema no hace ninguna acción o cambio, las gráficas siguen siendo las mismas.	
E1.HU5-CA1	<pre>[Authorize(Roles = "Administrador")] [HttpPost] 0 references public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); }</pre>	Filtro = "Todos"	Lista de todos los clientes registrados en la sede.	Media	<div><div><div>NET_MVC.Datos.Adm...Count = 6</div><div>this{NET_MVC.Controllers.ClienteController}</div><div>filter"all"</div></div><div><div>clientesFiltradosCount = 6</div><div>IdSede"1"</div></div><div>Q View</div></div>	
	<pre>[Authorize(Roles = "Administrador")] [HttpPost] 0 references public JsonResult Filtrar(string filter)</pre>					

E1.HU5-CA2	<pre> public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); } </pre>	Filtro = "Clientes Premium"		Lista de todos los clientes premium registrados en la sede.	Media	<div> <div>  NET_MVC.Datos.Adm... Count = 4 Q View </div> <div>  this {NET_MVC.Controllers.ClienteController} </div> <div>  filter "premium" Q View </div> <div>  clientesFiltrados Count = 4 Q View </div> <div>  IdSede "1" Q View </div> </div>
E1.HU5-CA3	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); } </pre>	Filtro = "Clientes Generales"		Lista de todos los clientes generales registrados en la sede.	Media	<div> <div>  NET_MVC.Datos.Adm... Count = 1 Q View </div> <div>  this {NET_MVC.Controllers.ClienteController} </div> <div>  filter "general" Q View </div> <div>  clientesFiltrados Count = 1 Q View </div> <div>  IdSede "1" Q View </div> </div>
E1.HU5-CA4	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); } </pre>	Filtro = "Género Masculino"		Lista de todos los clientes de género masculino registrados en la sede.	Media	<div> <div>  NET_MVC.Datos.Adm... Count = 4 Q View </div> <div>  this {NET_MVC.Controllers.ClienteController} </div> <div>  filter "masculino" Q View </div> <div>  clientesFiltrados Count = 4 Q View </div> <div>  IdSede "1" Q View </div> </div>
E1.HU5-CA5	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); } </pre>	Filtro = "Género Femenino"		Lista de todos los clientes de género femenino registrados en la sede.	Media	<div> <div>  NET_MVC.Datos.Adm... Count = 2 Q View </div> <div>  this {NET_MVC.Controllers.ClienteController} </div> <div>  filter "femenino" Q View </div> <div>  clientesFiltrados Count = 2 Q View </div> <div>  IdSede "1" Q View </div> </div>
E1.HU5-CA6	<pre> [Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter) {     List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     }); } </pre>	Filtro = "Género no especificado"		Lista de todos los clientes de género no especificado registrados en la sede.	Media	<div> <div>  NET_MVC.Datos.Adm... Count = 0 Q View </div> <div>  this {NET_MVC.Controllers.ClienteController} </div> <div>  filter "no-especificado" Q View </div> <div>  clientesFiltrados Count = 0 Q View </div> <div>  IdSede "1" Q View </div> </div>

E1.HU5-CA7		Recargar página		El sistema hace el llamado a las funciones que dependen de la consulta de clientes ya que pudo haber una nueva actualización de datos.	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
E1.HU5-CA8	<pre>[Authorize(Roles = "Administrador")] [HttpPost] public JsonResult Filtrar(string filter)      List&lt;ClienteModel&gt; clientesFiltrados = new List&lt;ClienteModel&gt;();     String IdSede = User.FindFirst(ClaimTypes.Name)?.Value;      // Ejecuta la consulta y obtiene la lista filtrada de clientes     clientesFiltrados = consultaCliente.ListarClientes(filter, IdSede);      return Json(new     {         clientes = clientesFiltrados,         totalClientes = clientesFiltrados.Count     });</pre>	Salir del módulo		El sistema borra automáticamente la consulta que se retorna en formato Json.	Baja	Al salir del módulo, la información de los clientes con los filtros se elimina.
E1.HU5-CA9		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.
E1-HU6-CA1	<pre>[HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != 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 _))                 {                     var clienteExistente = consultaCliente.ClienteExistente(Identificacion);                     if (clienteExistente)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdSede);                         if (cliente != null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clienteAsignados = ObtenerClientesAsignados();                             bool clienteAsignadoExistente = clienteAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (clienteAsignadoExistente)                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     }     return RedirectToAction("InformacionCliente"); }</pre>	ID = null		Mensaje de error	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"); // 2ms elapsed }</pre>
E1-HU6-CA2	<pre>[HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != 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 _))                 {                     var clienteExistente = consultaCliente.ClienteExistente(Identificacion);                     if (clienteExistente)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdSede);                         if (cliente != null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clienteAsignados = ObtenerClientesAsignados();                             bool clienteAsignadoExistente = clienteAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (clienteAsignadoExistente)                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     }     return RedirectToAction("InformacionCliente"); }</pre>	ID = ?#"		El sistema deberá de mandar un mensaje diciendo que el número debe de ser uno válido.	Media	<pre>// 3. Verificar que sea numérica if (!int.TryParse(Identificacion, out _)) {     TempData["ErrorMessage"] = "La identificación debe ser un número válido.";     return RedirectToAction("InformacionCliente"); // 1ms elapsed }</pre>
	<pre>[HttpPost] [Authorize(Roles = "Administrador, Entrenador")]</pre>					

E1-HU6-CA3	<pre>[HttpPost] [Authorize] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdCliente = User.FindFirst(ClaimTypes.Name).Value;         if (string.IsNullOrEmpty(Identificacion))         {             if (Identificacion.Length &gt; 10)             {                 if (int.TryParse(Identificacion, out _))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (!clienteExiste)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdCliente);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                             else if (User.IsInRole("Entrenador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     } }</pre>	ID = 102342345734	El sistema mandará mensaje diciendo que la identificación no puede tener más de 10 dígitos.	Media	<pre>// 2. Verificar que la longitud no supere los 10 caracteres if (Identificacion.Length &gt; 10) {     TempData["ErrorMessage"] = "La identificación no puede tener más de 10 dígitos.";     return RedirectToAction("InformacionCliente"); // 3 ms elapsed }</pre>																
E1-HU6-CA4	<pre>[HttpPost] [Authorize] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdCliente = User.FindFirst(ClaimTypes.Name).Value;         if (string.IsNullOrEmpty(Identificacion))         {             if (Identificacion.Length &gt; 10)             {                 if (int.TryParse(Identificacion, out _))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (!clienteExiste)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdCliente);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                             else if (User.IsInRole("Entrenador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     } }</pre>	ID = 103430	El sistema mandará un mensaje "Cliente no encontrado."	Media	<pre>if (!clienteExiste) {     TempData["ErrorMessage"] = User.IsInRole("Administrador")         ? "Cliente no encontrado."         : "El cliente no existe en el sistema.";     return RedirectToAction("InformacionCliente"); // 2 ms elapsed }</pre>																
E1-HU6-CA5	<pre>[HttpPost] [Authorize] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdCliente = User.FindFirst(ClaimTypes.Name).Value;         if (string.IsNullOrEmpty(Identificacion))         {             if (Identificacion.Length &gt; 10)             {                 if (int.TryParse(Identificacion, out _))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (!clienteExiste)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdCliente);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                             else if (User.IsInRole("Entrenador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     } }</pre>	ID = 109	El sistema mostrará la información del cliente.	Media	<table><tr><th>cliente</th><th>[NET_MVC.Models.ClienteModel]</th></tr><tr><td>DiasRestantes</td><td>22</td></tr><tr><td>FechaNacimiento</td><td>{1/01/0001 12:00:00 a. m.}</td></tr><tr><td>Genero</td><td>"M" <a href="#">Q View</a></td></tr><tr><td>IdEntrenador</td><td>0</td></tr><tr><td>IdSede</td><td>null</td></tr><tr><td>Identificacion</td><td>"109" <a href="#">Q View</a></td></tr><tr><td>Nombre</td><td>"Camilo Castro" <a href="#">Q View</a></td></tr></table>	cliente	[NET_MVC.Models.ClienteModel]	DiasRestantes	22	FechaNacimiento	{1/01/0001 12:00:00 a. m.}	Genero	"M" <a href="#">Q View</a>	IdEntrenador	0	IdSede	null	Identificacion	"109" <a href="#">Q View</a>	Nombre	"Camilo Castro" <a href="#">Q View</a>
cliente	[NET_MVC.Models.ClienteModel]																				
DiasRestantes	22																				
FechaNacimiento	{1/01/0001 12:00:00 a. m.}																				
Genero	"M" <a href="#">Q View</a>																				
IdEntrenador	0																				
IdSede	null																				
Identificacion	"109" <a href="#">Q View</a>																				
Nombre	"Camilo Castro" <a href="#">Q View</a>																				
E1-HU6-CA6	<pre>[HttpPost] [Authorize] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdCliente = User.FindFirst(ClaimTypes.Name).Value;         if (string.IsNullOrEmpty(Identificacion))         {             if (Identificacion.Length &gt; 10)             {                 if (int.TryParse(Identificacion, out _))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (!clienteExiste)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdCliente);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                             else if (User.IsInRole("Entrenador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     } }</pre>	Recargar página	El sistema llamará nuevamente a las funciones encargadas y mostrará a el cliente.	Baja	<table><tr><th>cliente</th><th>[NET_MVC.Models.ClienteModel]</th></tr><tr><td>DiasRestantes</td><td>22</td></tr><tr><td>FechaNacimiento</td><td>{1/01/0001 12:00:00 a. m.}</td></tr><tr><td>Genero</td><td>"M" <a href="#">Q View</a></td></tr><tr><td>IdEntrenador</td><td>0</td></tr><tr><td>IdSede</td><td>null</td></tr><tr><td>Identificacion</td><td>"109" <a href="#">Q View</a></td></tr><tr><td>Nombre</td><td>"Camilo Castro" <a href="#">Q View</a></td></tr></table>	cliente	[NET_MVC.Models.ClienteModel]	DiasRestantes	22	FechaNacimiento	{1/01/0001 12:00:00 a. m.}	Genero	"M" <a href="#">Q View</a>	IdEntrenador	0	IdSede	null	Identificacion	"109" <a href="#">Q View</a>	Nombre	"Camilo Castro" <a href="#">Q View</a>
cliente	[NET_MVC.Models.ClienteModel]																				
DiasRestantes	22																				
FechaNacimiento	{1/01/0001 12:00:00 a. m.}																				
Genero	"M" <a href="#">Q View</a>																				
IdEntrenador	0																				
IdSede	null																				
Identificacion	"109" <a href="#">Q View</a>																				
Nombre	"Camilo Castro" <a href="#">Q View</a>																				
E1-HU6-CA7	<pre>[HttpPost] [Authorize] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != null &amp;&amp; User.IsInRole("Entrenador"))     {         string IdCliente = User.FindFirst(ClaimTypes.Name).Value;         if (string.IsNullOrEmpty(Identificacion))         {             if (Identificacion.Length &gt; 10)             {                 if (int.TryParse(Identificacion, out _))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (!clienteExiste)                     {                         ClienteModel cliente = consultaCliente.ObtenerClientePorIdentificacion(Identificacion, IdCliente);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModel&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                             else if (User.IsInRole("Entrenador"))                             {                                 return RedirectToAction("InformacionCliente");                             }                         }                     }                 }             }         }     } }</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.																
	<pre>[HttpPost] [Authorize] [Authorize(Roles = "Administrador, Entrenador")]</pre>																				

E1-HU6-CA8	<pre> [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != 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 ))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (clienteExiste)                     {                         ClienteModelo cliente = consultaCliente.ObtenerClientePorId(Identificacion, idSede);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModelo&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                                 return RedirectToAction("InformacionCliente");                             else if (User.IsInRole("Entrenador"))                                 return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	Tecla ESC		El sistema no deberá de realizar ninguna acción.	Baja	Al Presionar ESC el sistema no realiza ninguna acción.
E1-HU6-CA9	<pre> [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != 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 ))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (clienteExiste)                     {                         ClienteModelo cliente = consultaCliente.ObtenerClientePorId(Identificacion, idSede);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModelo&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                                 return RedirectToAction("InformacionCliente");                             else if (User.IsInRole("Entrenador"))                                 return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = 1		El sistema deberá de mostrar un mensaje, diciendo que el cliente no existe 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>
E1-HU6-CA10	<pre> [HttpPost] [Authorize(Roles = "Administrador, Entrenador")] public ActionResult BuscarCliente(string Identificacion) {     if (HttpContext.Session.GetString("ClienteId") != 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 ))                 {                     bool clienteExiste = consultaCliente.ClienteExiste(Identificacion);                     if (clienteExiste)                     {                         ClienteModelo cliente = consultaCliente.ObtenerClientePorId(Identificacion, idSede);                         if (cliente == null)                         {                             HttpContext.Session.SetString("ClienteId", Identificacion);                             List&lt;ClienteModelo&gt; clientesAsignados = ObtenerClientesAsignados();                             bool clientesAsignadosResult = clientesAsignados != null &amp;&amp; clientesAsignados.Any(c =&gt; c.Identificacion == cliente.Identificacion);                             if (User.IsInRole("Administrador"))                                 return RedirectToAction("InformacionCliente");                             else if (User.IsInRole("Entrenador"))                                 return RedirectToAction("InformacionCliente");                         }                     }                 }             }         }     } } </pre>	ID = 120		El sistema deberá de mostrar un mensaje, diciendo que el cliente no existe en el sistema de la sede.	Media	<pre> if (cliente == null) {     TempData["ErrorMessage"] = "El cliente no existe en esta sede. Por favor intente nuevamente.";     return RedirectToAction("InformacionCliente"); } </pre>