

SEGUNDO RELEASE (SPRINT 7 Y 8)

1. E3 - HU5: Eliminar ejercicio.

2. E3 - HU2: Modificar plan de mejoramiento físico a clientes premium .

3. E3 - HU7: Consultar estadísticas sobre los clientes.

4. E3 - HU3: Consultar información personal de un entrenador del gimnasio.

ID	Componente	Valores	Tipo (+/-)	Resultado Esperado	Criticidad	Resultado Real
E3-HU5-CA1	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] // references public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } }</pre>	ID = null		Un mensaje de error: "La identificación no puede estar vacía".	Media	<pre>if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { TempData["ErrorMessage"] = "La identificación no puede estar vacía." ; // Immediato return View("EliminarEjercicio"); // Redirige a la página donde se muestra el formulario }</pre>
E3-HU5-CA2	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] // references public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } }</pre>	ID = 2323242332		Un mensaje advirtiendo que la id no puede tener más de 10 dígitos.	Media	<pre>// verificar que la longitud no sea mayor a 10 dígitos if (ejercicio.IdEjercicio.Length > 10) { TempData["ErrorMessage"] = "La identificación no puede tener más de 10 dígitos."; return View("EliminarEjercicio"); }</pre>
E3-HU5-CA3	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] // references public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } }</pre>	ID = asdad		El sistema mostrará un mensaje diciendo que solo puede contener números el ID.	Media	<pre>// Verificar que la identificación solo contenga números if (!ejercicio.IdEjercicio.All(char.IsDigit)) { TempData["ErrorMessage"] = "La identificación solo puede contener números."; return View("EliminarEjercicio"); }</pre>
	<pre>[Authorize(Roles = "Entrenador")]</pre>					

E3-HU5-CA4	<pre> [HttpPost] [Authorize] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } } </pre>	ID = -2323		El sistema deberá de mandar un mensaje advirtiendo que no se puede números negativos.	Media	<pre> // Convertir a entero y verificar que sea mayor a 0 if (!int.TryParse(ejercicio.IdEjercicio, out int identificacionNumero) identificacionNumero <= 0) { TempData["ErrorMessage"] = "La identificación debe ser un número positivo."; return View("EliminarEjercicio"); } </pre>
E3-HU5-CA5	<pre> [Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } } </pre>	ID = 232		El sistema informará al usuario que el ejercicio no existe.	Media	<pre> else { TempData["ErrorMessage"] = "Ejercicio no existente en el el PNF"; return View("EliminarEjercicio"); } </pre>
E3-HU5-CA6	<pre> [Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } } </pre>	ID = 2		El sistema realizará la eliminación del ejercicio.	Media	<pre> if (eliminar) { return RedirectToAction("modificarEjercicios"); } </pre>
E3-HU5-CA7	<pre> [Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } } </pre>	Recargar la página		El sistema deberá de elimina el campo donde se digita la identificación del ejercicio.	Baja	El sistema elimina el campo donde se digita la identificación del ejercicio.
E3-HU5-CA8	<pre> [Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("ClienteIdEjercicio")); ; ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) else } } </pre>	Salir del modulo		El sistema deberá de elimina el campo donde se digita la identificación del ejercicio.	Baja	El sistema elimina el campo donde se digita la identificación del ejercicio.
	<pre> [Authorize(Roles = "Entrenador")] </pre>					

E3-HU5-CA9	<pre>[HttpPost] [Authorize] public IActionResult EliminarEjercicio(EjercicioModel ejercicio) { // Verificar que la identificación no esté vacía if (string.IsNullOrEmpty(ejercicio.IdEjercicio)) { ejercicio.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); ejercicio.FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("fechaValoracion")); // Verificar si el ejercicio existe en la base de datos var existencia = consulta.EjercicioExistente(ejercicio); if (existencia) { // ... } else { // ... } } }</pre>	Presionar ESC		El sistema no hace nada, sigue el flujo correctamente.	Baja	El sistema no realiza ninguna acción por presionar ESC
E3-HU2-CA3	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public JsonResult AgregarPMF(PMFModel pmf) { // 4.000ms elapsed HttpContext.Session.SetString("FechaValoracion", pmf.FechaValoracion.ToString()); pmf.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); if (ModelState.IsValid) { // ... return Json(new { success = true, errors = ModelState }, JsonRequestBehavior.AllowGet, v => v.Value.Errors.Select(e => e.ErrorMessage).ToArray()); } }</pre>	Fecha 12/1/2024		El sistema deberá de crear un PMF	Media	<pre>var existencia = consulta.PMFExistente(pmf); if (existencia) { return Json(new { success = true, redirectUrl = Url.Action("modificarEjercicios") }); }</pre>
E3-HU2-CA4	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public JsonResult AgregarPMF(PMFModel pmf) { // 4.000ms elapsed HttpContext.Session.SetString("FechaValoracion", pmf.FechaValoracion.ToString()); pmf.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); if (ModelState.IsValid) { // ... return Json(new { success = false, errors = ModelState }, JsonRequestBehavior.AllowGet, v => v.Value.Errors.Select(e => e.ErrorMessage).ToArray()); } }</pre>	Fecha 12/1/2024		El sistema deberá de retornar una lista de los ejercicios creados si tiene, de lo contrario mostrará una tabla vacía.	Media	<pre>int idCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); DateTime FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); List<EjercicioModel> ejercicios = consulta.ListarEjercicios(idCliente, FechaValoracion); if (ejercicios.Count() > 0) { return ejercicios; } else { // ... }</pre>
E3-HU2-CA5	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public JsonResult AgregarPMF(PMFModel pmf) { // 4.000ms elapsed HttpContext.Session.SetString("FechaValoracion", pmf.FechaValoracion.ToString()); pmf.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); if (ModelState.IsValid) { // ... return Json(new { success = false, errors = ModelState }, JsonRequestBehavior.AllowGet, v => v.Value.Errors.Select(e => e.ErrorMessage).ToArray()); } }</pre>	Recarga página		El sistema deberá de llamar a las funciones previamente ejecutadas permitiendo hacer actualización de datos.	Baja	<pre>int idCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); DateTime FechaValoracion = DateTime.Parse(HttpContext.Session.GetString("FechaValoracion")); List<EjercicioModel> ejercicios = consulta.ListarEjercicios(idCliente, FechaValoracion); if (ejercicios.Count() > 0) { return ejercicios; } else { // ... }</pre>
E3-HU2-CA6	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] [ValidateAntiForgeryToken] public JsonResult AgregarPMF(PMFModel pmf) { // 4.000ms elapsed HttpContext.Session.SetString("FechaValoracion", pmf.FechaValoracion.ToString()); pmf.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); if (ModelState.IsValid) { // ... return Json(new { success = false, errors = ModelState }, JsonRequestBehavior.AllowGet, v => v.Value.Errors.Select(e => e.ErrorMessage).ToArray()); } }</pre>	Salir del módulo		El sistema deberá eliminar el cache de la información.	Baja	El sistema elimina el cache de la información.

E3-HU2-CA7	<pre>[Authorize(Roles = "Entrenador")] [HttpPost] // references public JsonResult AgregarPMF(PMFModel pmf) { < 400ms elapsed HttpContext.Session.SetString("FechaValoracion", pmf.FechaValoracion.ToString()); pmf.IdCliente = int.Parse(HttpContext.Session.GetString("clienteIdEjercicio")); if (ModelState.IsValid) { return Json(new { success = false, errors = ModelState .ToDictionary(k => k.Key, v => v.Value.Errors.Select(e => e.ErrorMessage).ToArray()) }); } }</pre>	Presionar ESC		El sistema no deberá de ejecutar o hacer alguna otra acción en el módulo.	Baja	El sistema no realiza ninguna acción que influya en su función.
E3-HU2-CA8	<pre>[Authorize(Roles = "Entrenador")] public IActionResult AgregarEjercicio() { List<NombreEjercicio> opciones = consulta.ObtenerOpciones(); var modelo = new EjercicioModel { Opciones = opciones }; return View("AgregarEjercicio", modelo); }</pre>	Módulo añadir ejercicio		El sistema deberá de redirigir a la vista del ejercicio	Media	<pre>[Authorize(Roles = "Entrenador")] // references public IActionResult AgregarEjercicio() { List<NombreEjercicio> opciones = consulta.ObtenerOpciones(); var modelo = new EjercicioModel { Opciones = opciones }; return View("AgregarEjercicio", modelo); < 2ms elapsed }</pre>
E3-HU2-CA9	<pre>[Authorize(Roles = "Entrenador")] public IActionResult EliminarEjercicio() { return View("EliminarEjercicio"); }</pre>	Módulo eliminar ejercicio		El sistema deberá de redirigir a la vista del ejercicio	Media	<pre>[Authorize(Roles = "Entrenador")] // references public IActionResult EliminarEjercicio() { return View("EliminarEjercicio"); < 1ms elapsed }</pre>
E3-HU2-CA10	<pre>[Authorize(Roles = "Entrenador")] // references public IActionResult BuscarActualizarEjercicio() { < 5ms elapsed return View("BuscarEjercicioActualizar"); }</pre>	Módulo modificar ejercicio		El sistema deberá de redirigir a la vista del ejercicio	Media	<pre>[Authorize(Roles = "Entrenador")] // references public IActionResult BuscarActualizarEjercicio() { return View("BuscarEjercicioActualizar"); < 1ms elapsed }</pre>
E3-HU2-CA11		Recargar página		El sistema hace el llamado a las funciones que dependen del módulo	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

E3-HU2-CA12		Salir del módulo		El sistema deberá de eliminar la información que se tenga.	Baja	El sistema borra el cache de la información buscada previamente.
E3-HU2-CA13		Presiona 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.
E3-HU7-CA1	<pre> [Authorize(Roles = "Entrenador")] public IActionResult DashboardEntrenador() { if (HttpContext.Session.GetString("ClienteIdEjercicio") != null) HttpContext.Session.Remove("ClienteIdEjercicio"); try { string idEntrenador = User.FindFirst(ClaimTypes.Name)?.Value; if (!int.TryParse(idEntrenador, out int entrenadorId)) { ViewBag.TotalClientes = consultaEntrenador.NumeroClientesAsignados(entrenadorId); ViewBag.DiasRestantes = consultaEntrenador.DiasRestantesContrato(entrenadorId); return View("DashboardEntrenador"); } catch (Exception ex) { TempData["ErrorMessage"] = \$"Error al cargar el dashboard: {ex.Message}"; return View("DashboardEntrenador"); } } } </pre>	Inicio login		El sistema muestra el inicio de sesión junto con las gráficas deseadas para el módulo.	Alta	<pre> try { string idEntrenador = User.FindFirst(ClaimTypes.Name)?.Value; if (!int.TryParse(idEntrenador, out int entrenadorId)) { ViewBag.TotalClientes = consultaEntrenador.NumeroClientesAsignados(entrenadorId); ViewBag.DiasRestantes = consultaEntrenador.DiasRestantesContrato(entrenadorId); return View("DashboardEntrenador"); } } </pre>
E3-HU7-CA2	<pre> [Authorize(Roles = "Entrenador")] public IActionResult DashboardEntrenador() { if (HttpContext.Session.GetString("ClienteIdEjercicio") != null) HttpContext.Session.Remove("ClienteIdEjercicio"); try { string idEntrenador = User.FindFirst(ClaimTypes.Name)?.Value; if (!int.TryParse(idEntrenador, out int entrenadorId)) { ViewBag.TotalClientes = consultaEntrenador.NumeroClientesAsignados(entrenadorId); ViewBag.DiasRestantes = consultaEntrenador.DiasRestantesContrato(entrenadorId); return View("DashboardEntrenador"); } catch (Exception ex) { TempData["ErrorMessage"] = \$"Error al cargar el dashboard: {ex.Message}"; return View("DashboardEntrenador"); } } } </pre>	Opción página principal entrenador		El sistema vuelve a hacer llamado a la función de estadísticas del entrenador.	Media	<pre> try { string idEntrenador = User.FindFirst(ClaimTypes.Name)?.Value; if (!int.TryParse(idEntrenador, out int entrenadorId)) { ViewBag.TotalClientes = consultaEntrenador.NumeroClientesAsignados(entrenadorId); ViewBag.DiasRestantes = consultaEntrenador.DiasRestantesContrato(entrenadorId); return View("DashboardEntrenador"); } } </pre>
E3-HU7-CA3	<pre> [Authorize(Roles = "Entrenador")] public IActionResult DashboardEntrenador() { if (HttpContext.Session.GetString("ClienteIdEjercicio") != null) HttpContext.Session.Remove("ClienteIdEjercicio"); try { string idEntrenador = User.FindFirst(ClaimTypes.Name)?.Value; if (!int.TryParse(idEntrenador, out int entrenadorId)) { ViewBag.TotalClientes = consultaEntrenador.NumeroClientesAsignados(entrenadorId); ViewBag.DiasRestantes = consultaEntrenador.DiasRestantesContrato(entrenadorId); return View("DashboardEntrenador"); } catch (Exception ex) { TempData["ErrorMessage"] = \$"Error al cargar el dashboard: {ex.Message}"; return View("DashboardEntrenador"); } } } </pre>	Minimizar la pestaña		El sistema no deberá de cambiar datos o dimensiones de la gráfica.	Baja	El sistema no cambia datos o dimensiones de la gráfica, continua su flujo normal.
	<pre> public DateTime ObtenerDateUltimoPMD(string idCliente) </pre>					

E3-HU3-CA1

```
public DateTime ObtenerDateUltimoPMF(string idCliente)
{
    bool isValid = int.TryParse(idCliente, out int newIdCliente);
    if (!isValid)
    {
        throw new Exception("El id del cliente no es valido");
    }
    DateTime result = default;
    try
    {
        return result;
    }
}
```

cliente asignado al entrenador donde se inicio la sesi



El sistema lo dirige al módulo donde se muestra la información respecto al PMF

Alta

```
public IActionResult PMF()
{
    //obtengo el id del cliente almacenado en HttpContext.Session
    string idCliente = HttpContext.Session.GetString("ClienteIdEjercicio");

    //Recupero el ultimo PMF desde la base de datos
    DateTime dateTimeUltimoPMF = consulta.ObtenerDateUltimoPMF(idCliente);
    HttpContext.Session.SetString("FechaUltimaEjecucion", dateTimeUltimoPMF.ToString());
    return View("PMF", ObtenerEjercicios()); // Time elapsed
}
```