Universidad Tecnológica Nacional

Facultad Regional Paraná

Tecnicatura Universitaria en Programación

Informe Final Práctica Profesional Supervisada

Docente: Zapata Icart, Ernesto Andrés

Alumno: Ghiringhelli, Livio Alejandro

Empresa: Grandi y Asociados – Ertic S.R.L.

2024

Índice

Resumen Ejecutivo

Este trabajo se enmarca en la cátedra Práctica Profesional que es una materia integradora de la carrera de la Tecnicatura Universitaria en Programación, cuyo objetivo es que el estudiante pueda plasmar todo lo visto en el transcurso de la tecnicatura, tanto académico-conceptual como profesional y humano. Se desarrolla en una institución o empresa determinada que es elegida entre el alumno y el docente responsable de la cátedra.

Se busca que el estudiante se apropie del ecosistema laboral del mundo del desarrollo de software y/o consiga experimentar de primera mano las metodologías de trabajo en la institución o empresa en que se realiza la Práctica Profesional Supervisada (en adelante PPS), la cual se llevará a cabo mediante la supervisión y colaboración entre los profesionales de aquellas y el docente. De esta manera el practicante puede adquirir herramientas relevantes para la carrera profesional, fomentando el desarrollo de habilidades interpersonales, de trabajo en equipo y con capacidad de adaptación a distintos contextos de la profesión.

La PPS se estructura entorno a un plan de actividades y proyectos que permiten que el estudiante pueda abordar problemáticas reales y aplicar soluciones concretas utilizando todos los conocimientos adquiridos durante la formación académica y personal. Esto conlleva a que éste se enfrente a nuevos desafíos y que contribuya de manera significativa al trabajo y los proyectos de la institución o empresa elegida.

Por lo dicho, en esta PPS se busca y espera que el practicante pueda aplicar todo lo adquirido durante la tecnicatura en un entorno laboral posible de desarrollarse como futuro profesional, y que éste sirva al estudiante como puntapié inicial para la inserción en el mundo laboral del desarrollo de software.

Introducción

En el desarrollo de este informe se intenta proporcionar un análisis descriptivo de las Prácticas Profesionales Supervisadas llevadas a cabo por quien suscribe, el estudiante Livio Alejandro Ghiringhelli, en la empresa Grandi y Asociados por un período de seis meses.

La empresa Grandi y Asociados, es una empresa de desarrollo de software e ingeniería web que brinda soluciones informáticas integrales para comercios y PyMEs. Tiene más de treinta y cinco años de trayectoria en la Argentina y en otros países latinoamericanos.

Debido a la pandemia de COVID-19 las Prácticas Profesionales Supervisadas que antes se realizaban de forma presencial tuvieron que pasar a ser remotas, lo que perduró para las prácticas venideras. Esto implicó la adopción de tecnologías que, si bien no eran nuevas, pudieron ser un gran soporte para la realización de estas prácticas. Por tal motivo llevé adelante mis Prácticas Profesionales Supervisadas de forma remota desde el 21 de junio hasta el 15 de diciembre de 2022, con una dedicación de cuatro horas semanales, establecidas entre las 14:00 y las 18:00 horas.

Durante mis prácticas, se adoptaron tecnologías que transformaron significativamente la dinámica laboral. La aplicación Discord se utilizó tanto para facilitar reuniones de trabajo virtuales como para la supervisión, y se integró la plataforma Redmine en el horario laboral, siendo esta última fundamental para la gestión de proyectos.

La supervisión de las tareas a distancia se llevó a cabo mediante Discord, donde se compartían actualizaciones y progresos del proyecto asignado, evidenciando así la adaptabilidad y la implementación de herramientas tecnológicas. Este cambio resalta la capacidad de mantener la eficiencia y la colaboración en un entorno laboral que antes se basaba principalmente en la presencialidad.

1. Aspectos Generales

1.1. Antecedentes

Antes de adentrarme en las Prácticas Profesionales Supervisadas (en adelante PPS) dentro de la empresa Grandi y Asociados contaba con experiencia:

- trabajé independiente en tareas de reparación y mantenimiento de computadoras. Esto me permitió aprender a relacionarme con los clientes y que éstos me recomendaran. También me permitió orientarme en cuáles serían algunos de mis intereses a posteriori, lo que llevó a que me interesara por esta tecnicatura.
- trabajé en forma remota en cursos que realicé relacionados a la programación como fueron las dos etapas del programa nacional Argentina Programa, la primera llamada "#SeProgramar" y la segunda "#YoPrograma, Perfil Web Full Stack Junior"; y capacitaciones dictadas por el Instituto Nacional de Formación Docente y por la Fundación Sadosky llamadas "Estrategias para la enseñanza de la programación I" y "Estrategias para la enseñanza de la programación II".
- trabajé en equipo, valor que considero importante a la hora de desempeñarme como futuro programador ya que de esta forma hay un feedback que permite desarrollarme profesional y personalmente. Me desempeño dentro de la administración pública provincial, específicamente en la Secretaría de Economía Social del Ministerio de Desarrollo Social de la provincia de Entre Ríos.

Por mi bagaje anteriormente mencionado, fue que tomé la decisión de realizar las PPS. Esto se debió a que considero importante la experiencia que me puede aportar el trabajo en equipo en diferentes áreas y ámbitos profesionales en los cuales puede desempeñarse un programador como así también realizar una praxis y adquirir herramientas como futuro técnico en programación.

Por último, considero importante resaltar que mi decisión fue producto del deseo de conocer de primera mano la forma con la que trabaja una empresa del sector y medir los conocimientos que adquirí durante la Tecnicatura Universitaria en Programación (en adelante TUP).

1.2. Objetivos

1.2.1. General

- Proporcionar al estudiante la oportunidad de una experiencia práctica mediante la prestación de un servicio de carácter competente a una Institución o Empresa para que desarrolle sus habilidades y destrezas, y conocimiento adquirido en el proceso de aprendizaje durante su carrera universitaria, para complementar su formación integral y profesional.

1.2.2. Específicos

- Brindar al estudiante la oportunidad de desarrollar sus conocimientos adquiridos durante su carrera universitaria.

- Fomentar el desarrollo del sentido de responsabilidad social y ético en el desempeño profesional del estudiante.

- Lograr que la experiencia adquirida por el estudiante en su práctica profesional estimule el desarrollo integral y profesional del mismo.

- Mejorar la calidad de vida tanto profesional como personal.

1.3. Delimitación de la práctica profesional

En la empresa Grandi y Asociados los proyectos se desarrollaban con el lenguaje C#, lenguaje con el cual estaba familiarizado en mi primer año académico de la TUP, por lo que mi competencia se adecuaba a los requerimientos de esta. Sin embargo, dentro de la empresa también se utilizaban otras tecnologías que no conocía en profundidad como el motor de base de datos SQL Server, el ORM Entity Framework, los framework Flutter para el desarrollo de aplicaciones móviles y Angular para el desarrollo frontend.

Al inicio de las PPS me asignaron un ABM (Alta, Baja, Modificación) de una entidad de un proyecto de prueba que tenía en mente la empresa. Esta etapa fue necesaria para analizar el nivel en el que me encontraba para luego, con más precisión, designarme el proyecto con el que finalmente desarrollé mis PPS durante el período de tiempo estipulado.

1.4. Limitaciones

La empresa requería que tuviese conocimiento del lenguaje C#, si bien contaba con conocimiento de este, desconocía otras tecnologías con las que la misma trabajaba, como lo son:

- Angular

- Flutter
- EntityFramework
- SQL Server.

Razón por la cual tuve entre tres y cuatro semanas de lectura de documentación y visualización de videos aportados por la empresa para familiarizarme con estas tecnologías y poder desempeñarme de forma pertinente en el proyecto que esta me asignaría con posterioridad.

2. Evaluación Institucional

2.1. Descripción General de la Empresa

- Razón social: Grandi y Asociados - Ertic SRL

- Localidad: Paraná

- Provincia: Entre Ríos

- Domicilio: Pablo Lorentz 2977

- Código postal: 3100

2.2. Objetivos de la Empresa

- Dar soluciones informáticas integrales para comercios y PyMEs en la Argentina y en otros países latinoamericanos.

- Proponer sistemas de gestión abiertos y configurables, lo que permite una amplia adaptación, desarrollados por personal de la empresa, lo que asegura la continuidad y soporte post venta.

- Desarrollar sitios web a cargo de profesionales informáticos y diseñadores gráficos, que trabajan de forma conjunta para lograr sitios web estéticamente atractivos y técnicamente operativos.

- Brindar seguridad y confianza en sus clientes.

- Ofrecer soluciones rápidas, seguras y elásticas que se adapten a todas las etapas de los proyectos de sus clientes.

- Brindar servicio de gestión y soporte técnico 24/7.

2.3. Visión

La visión de Grandi y Asociados es ser una empresa de desarrollo de Software personalizado, para las pequeñas y medianas empresas de más éxito y con los mejores estándares de calidad en el mercado. Aportando con esto a que sus clientes sean más rentables y competitivos.

2.4. Misión

La misión de la empresa es brindar soluciones tecnológicas que se encuentren al alcance de sus clientes a un costo accesible y de alta calidad.

Su misión especial es exceder constantemente, con sus productos y servicios, todas las expectativas de sus clientes, satisfaciendo sus necesidades y ayudándolos en el logro de sus metas y objetivos.

Además, proponen a sus clientes soluciones integrales, que abarquen los procesos de negocios dentro de su empresa, ayudando a lograr las diferencias competitivas que les permitan perpetuarse y crecer. Esto lo hacen con personal calificado y comprometido con la empresa.

Presentan a sus socios las utilidades esperadas y proporcionan a los empleados posibilidades de desarrollo que les permita alcanzar crecimiento personal y profesional.

2.5. Actividad de la Empresa

Grandi y Asociados es una empresa de software en Argentina que brinda soluciones informáticas integrales para comercios y PyMEs.

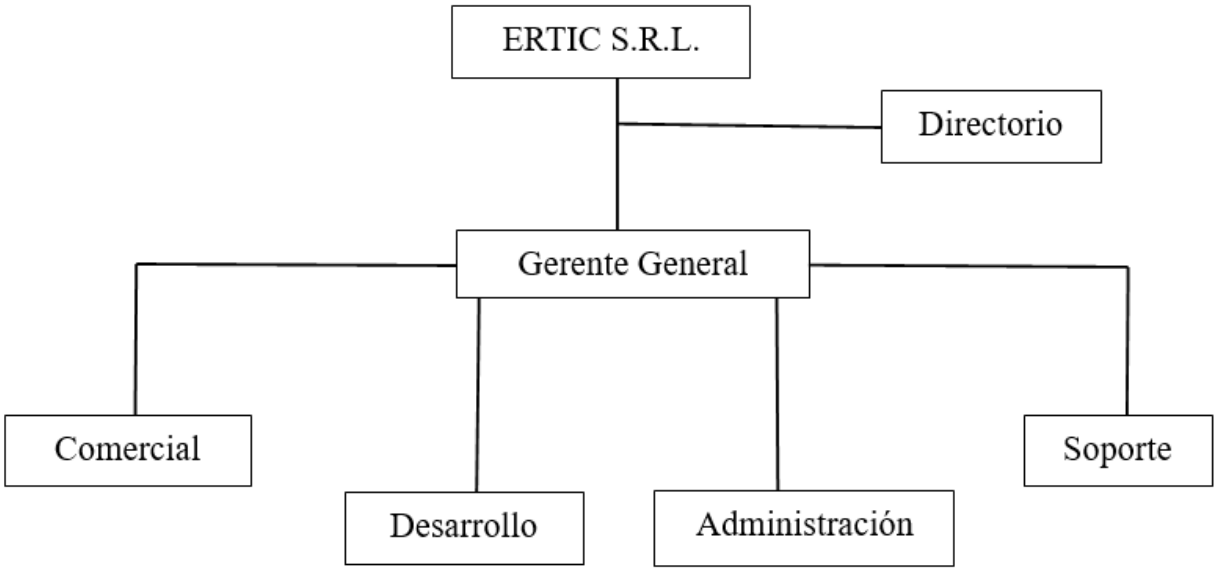
Por su experiencia adquirida al haber trabajado en diversidad de rubros y sumado al equipo de profesionales de la informática y el diseño web del que forman parte, es que avalan y garantizan sus trabajos, brindando seguridad y confianza a las necesidades de sus clientes.

Se encuentra conformada por un equipo de profesionales que acompañan a emprendedores y empresas en el crecimiento de sus negocios online mediante soluciones rápidas, seguras y elásticas que se adaptan a todas las etapas de sus proyectos.

Permanentemente, crean nuevas ideas ampliando su cartera de productos de Hosting, Cloud VPS, Nubes y Servidores Dedicados con Servicio de Gestión y Soporte Técnico 24/7.

El desarrollo de los sitios web se encuentra a cargo de profesionales de la informática y diseñadores gráficos, que trabajan mancomunadamente para que éstos sean estéticos y funcionales.

2.6. Estructura Organizacional



2.7. Descripción del grupo humano

El grupo humano al momento de iniciar las PPS se encontraba conformado por uno de los cofundadores de la empresa, una coordinadora, cinco ex alumnos de la TUP y tres compañeros practicantes de esta misma tecnicatura.

Cofundador:

- Ing. en Sistemas Lucas Grandi

Coordinadora del área:

- Lic. en Sistemas de Información María de los Ángeles Gabas

Desarrolladores:

- Mateo Schultheis
- Agustín Lamboglia
- Gustavo Toobe
- Luciano Forastieri
- Joan Sabotig.

En cuanto al equipo de trabajo de la empresa, todos se encontraban a disposición para responder dudas o consultas.

Quienes acompañaron y supervisaron el desarrollo del proyecto asignado durante las PPS fueron Gustavo Toobe y Lucas Grandi. Agustín Lamboglia y Mateo Schultheis también acompañaron este proceso, en particular en la instalación de librerías que serían necesarias para poder trabajar en el mismo. Luciano Forastieri y Joan Sabotig se encontraban trabajando en el desarrollo de otro proyecto para la empresa, sin embargo, cuando se presentaba un problema en el desarrollo del cual me encontraba trabajando, se mostraron predispuestos a responder inquietudes y orientarme en una solución o bien resolver el problema para no atrasar el tiempo de trabajo de este.

En cuanto a asuntos administrativos María de los Ángeles Gabas fue quien acompañó este proceso y quien nos capacitó en la utilización de la plataforma Redmine para poder marcar en esta las horas trabajadas y una descripción de lo realizado de las mismas.

Los compañeros practicantes que estuvieron realizando estas PPS conmigo fueron:

- Martín Estrada
- Rodrigo Sosa
- Adriel Miño

Al iniciar estas PPS los cuatro estuvimos trabajando en conjunto sobre el proyecto asignado y nos comunicábamos constantemente cuando encontrábamos un problema que no nos permitía continuar con el desarrollo.

Cuando finalizamos con la primera etapa del desarrollo del proyecto nos dividieron en dos grupos, uno conformado por Rodrigo y Adriel y que se encontraban acompañados en el proceso por Mateo; y el otro grupo se conformó entre Martín y quien suscribe el presente informe, nos supervisaba Gustavo para las consultas y Lucas para cuestiones técnicas y de análisis del proyecto. Continuamos trabajando de esta manera hasta que finalizamos estas PPS.

Junto a Martín trabajamos de manera mancomunada, resolviendo problemas propios del desarrollo del proyecto y, de manera conjunta, cuando no encontrábamos una solución por nosotros mismos le realizábamos la consulta a Gustavo o Agustín.

2.8. Descripción del escenario de trabajo

El escenario de trabajo fue cien por ciento remoto. Me conectaba al canal de “Entrenamiento” de la plataforma Discord y en este delineábamos el día de trabajo.

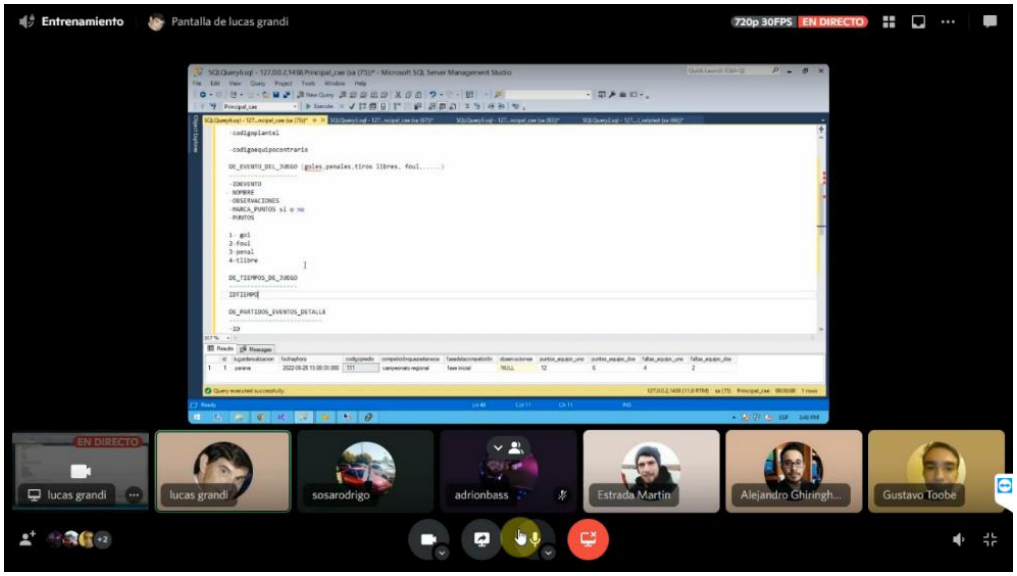
Todas las tecnologías y herramientas digitales que necesité utilizar fueron instaladas en mi computadora personal.

Una de estas herramientas fue la plataforma Discord, la misma era utilizada por todos los y las empleadas de la empresa. Tenían varios canales para reuniones de trabajo, en los cuales nos podíamos conectar ya sea para una reunión privada con Lucas o cuando nos dividieron para poder trabajar en equipo.

La plataforma fue de mucha utilidad a la hora de exponer consultas ya que de haber necesitado resolver algunas por correo electrónico hubiera significado una demora en el proceso del desarrollo del proyecto.

Un problema que se nos presentaba con esta plataforma era que podíamos perder la conexión a internet y, por ende, perdernos de escuchar algo importante en alguna reunión o no poder conectarnos al servidor de la base de datos para continuar con el desarrollo del proyecto. Esta situación se presentó en cinco oportunidades y se resolvió a partir de la reconexión en otro horario, por lo general podíamos conectarnos los días sábado, de esta manera podíamos recuperar esas horas y así no perder tiempo en el desarrollo del proyecto.

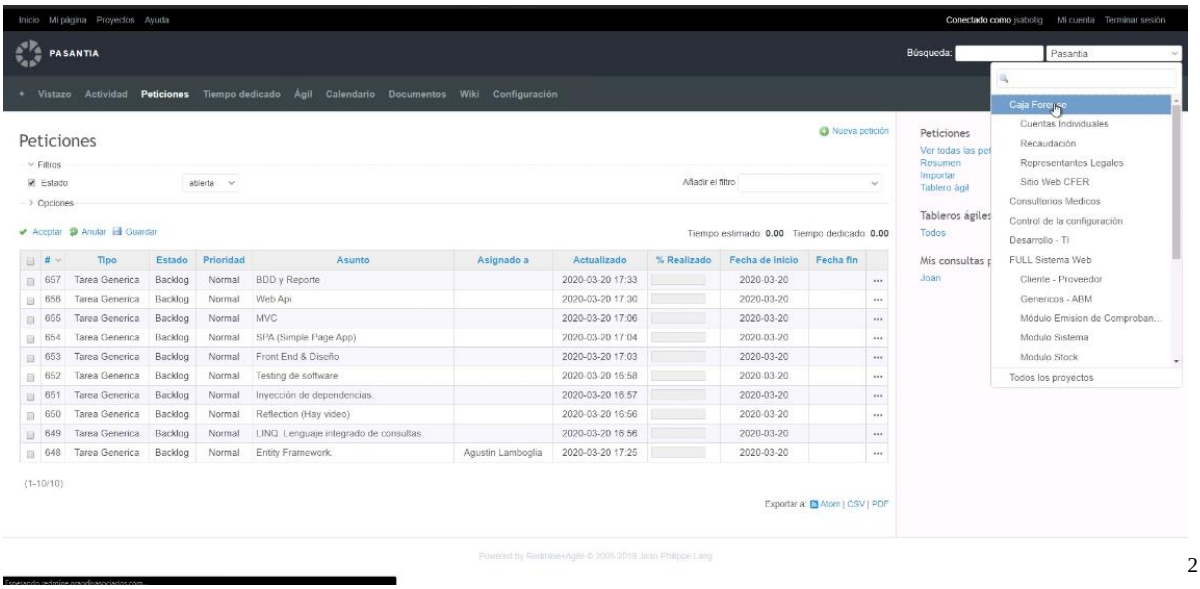
Otro problema que podía llegar a presentarse con la plataforma, pero que rara vez sucedía, era que el servidor de esta se encontrara caído y, por consiguiente, no podíamos conectarnos a una reunión. Esta situación se presentó dos veces y no implicó que no se pudiera continuar con el desarrollo del proyecto.



[1]

En cuanto a la plataforma Redmine, la misma se utilizó para la gestión de proyectos y el registro de las horas de conexión a la plataforma Discord, así como también una descripción de las tareas realizadas en las horas de conexión para que la empresa y nosotros como practicantes pudiéramos tener un registro cronológico de las PPS.

La descripción debía ser corta y concreta con relación a lo trabajado durante el día.

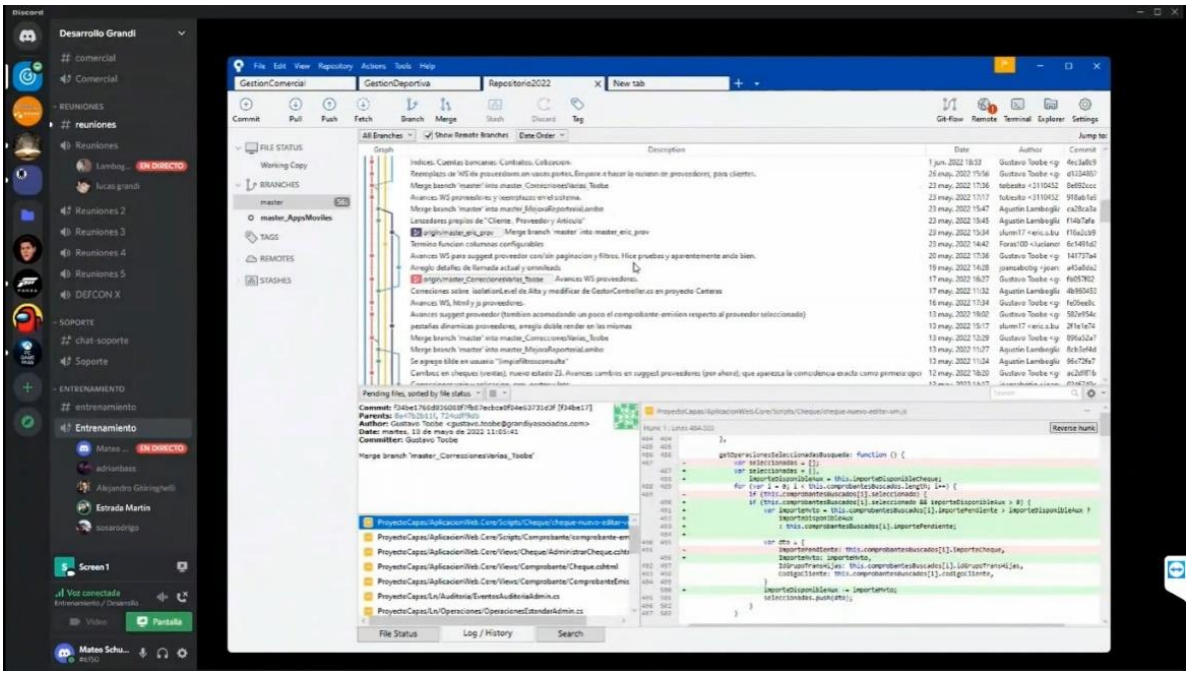


[2]

Para el proyecto asignado utilizamos un cliente Git gratuito para el seguimiento y control de versiones, el entorno Sourcetree fue el utilizado durante las PPS.

[1]Reunión virtual
[2]Grabación de una capacitación a pasantes del año 2021 en el uso de Redmine

La rama principal era la rama “master”, por lo que para el desarrollo de las PPS se creó una nueva rama a partir de esta a la que llamamos “master_pasantias”. De esta última rama, creamos ramas específicas para cada practicante, en mi caso “alejandro_ghiringhelli”.

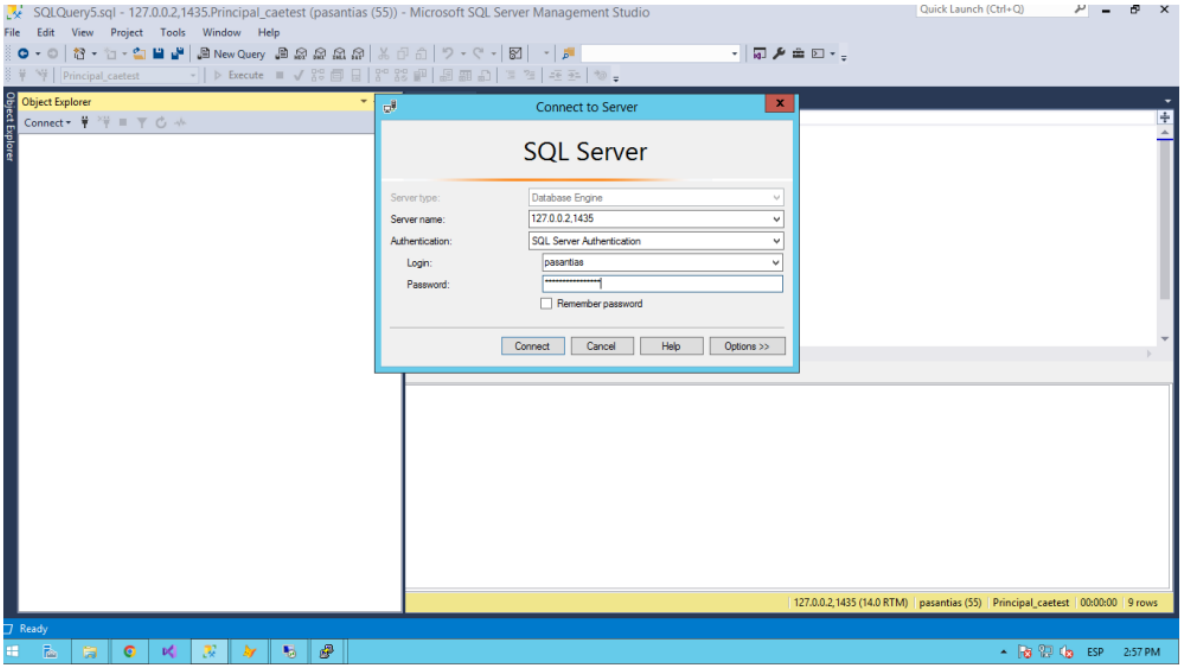


[3]

En cuanto al motor de base de datos implementado, SQL Server fue el elegido por la empresa para el manejo de todos sus proyectos. Se utilizó un servidor específico de prueba para estas PPS y que nos permitía poder realizar consultas a la base de datos sin afectar a los datos del resto de los proyectos de la empresa.

El usuario con el que nos conectábamos llevaba el nombre de “pasantias". Con este teníamos acceso a todas las tablas de todos los proyectos de la empresa.

[3]Extracción de una grabación de la capacitación en la utilización de la versión de escritorio Sourcetree



[4]

Para el desarrollo del proyecto asignado utilizamos las tablas:

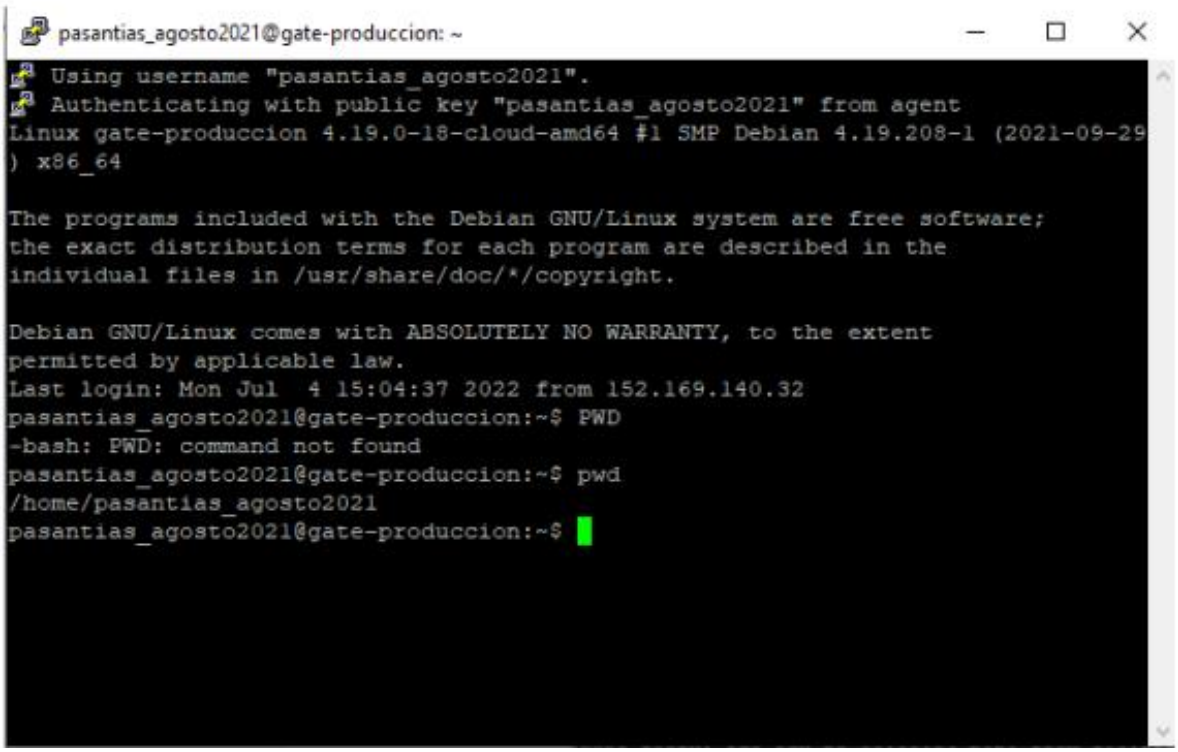
- DE_partidos
- DE_eventos_deportivos_tipo
- Estados
- DE_nivel_de_juego
- DE_campeonato_liga_fase
- DE_campeonato_liga
- Localidad
- DE_entidad_deportiva_predios
- DE_entidades_deportivas
- DE_planteles_por_temporada
- DE_categorias_deportivas
- Imagenes_multimedia_asoc
- Imagenes_multimedia
- DE_partidos_eventos
- DE_partido_eventos_participantes
- DE_situaciones_de_juego
- DE_partido_etapas

[4]Capacitación sobre la instalación de SQL Server y su conexión a la base de datos de la empresa

- DE_partido_duelos_resultados
- DE_posicion_campo_juego
- DE_puestos_deportivos
- DE_partidos_personas_involucradas

A partir del DER (Diagrama Entidad-Relación) [5] es que entendimos, como grupo de practicantes, la gran diferencia de lo que era realizar el análisis, diseño e implementación de código en ejercicios de carácter académico con proyectos a nivel empresarial.

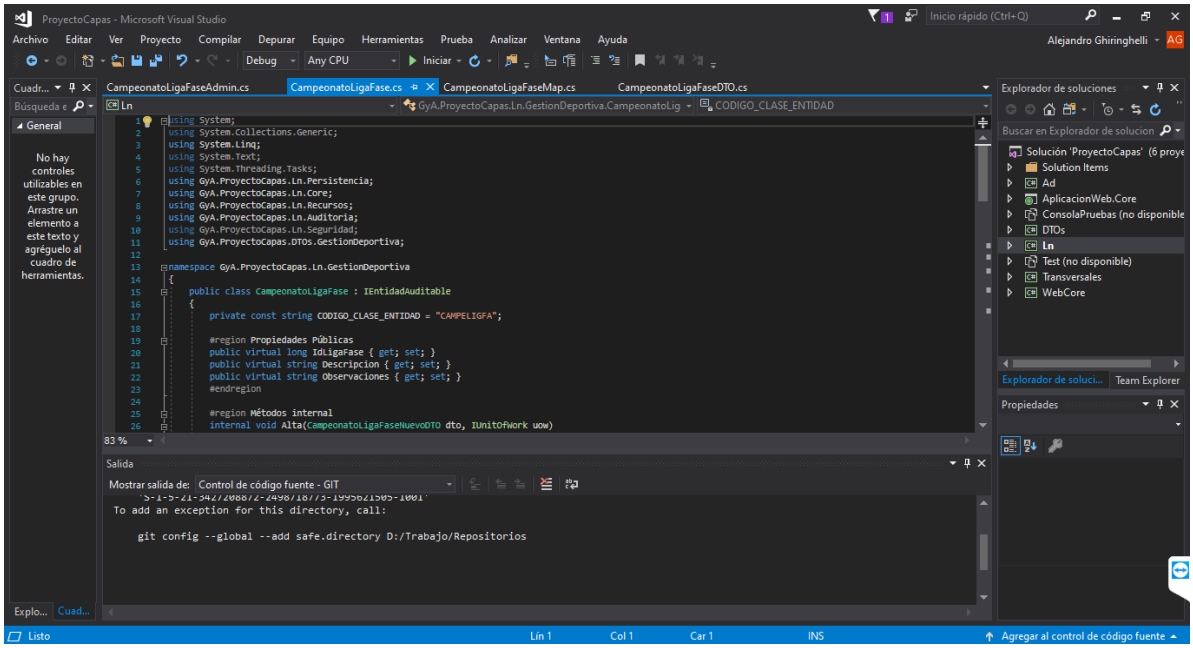
También se implementó el cliente PuTTY, de licencia libre, que sirvió para poder conectarnos al servidor de prueba de la base de datos de la empresa.



[6]

En cuanto al desarrollo del proyecto fue necesario utilizar, al menos, la versión actualizada al 2017 del IDE Visual Studio. Al iniciar estas PPS me encontraba utilizando que para proyectos de la TUP la versión 2010, por lo cual, actualicé el IDE a la versión solicitada por la empresa.

[5] Por razones de confidencialidad de información de la empresa es que este diagrama no puede ser plasmado en el presente informe.
[6] Conexión al servidor de la empresa



[7]

En cuanto a las tecnologías que se utilizaron, C# fue el lenguaje predominante para el desarrollo backend del proyecto mientras que para el desarrollo frontend utilicé la versión de Angular 1.0 para el JavaScript y HTML y CSS para el maquetado y estilado. Es importante señalar en este punto que dentro del proyecto existían archivos .css que sirvieron para realizar esta tarea.

[7]Extracción del proyecto trabajado para la PPS

3. Desarrollo de la Práctica Profesional

3.1. Descripción de actividades desarrolladas en la práctica profesional

Las primeras semanas me conectaba en la plataforma Discord y realizaba la lectura y familiarización de la documentación con relación a las tecnologías que no conocía, sumando a esta lectura la visualización de videos de reuniones virtuales generados por la empresa cuando nos encontrábamos en pandemia y que les permitió a otros y otras estudiantes de la TUP poder realizar sus prácticas.

Realicé la lectura a partir de una guía de temas generada por la empresa, la misma se detalla de forma general de la siguiente manera:

- Lógica de negocios .NET C#
- Frontend para aplicaciones FULL WEB – AngularJS (V1 y ANGULAR)
- Frontend para aplicaciones APP – Flutter
- BDD SQL SERVER
- Reportería
- Aplicaciones de Negocios

Luego de esas primeras semanas, me asignaron realizar el CRUD de una de las entidades de un proyecto de gestión deportiva, con el cual se pretendía administrar todo lo relacionado a un club, particularmente el Club Estudiantes de nuestra ciudad.

La entidad con la que comenzamos a desarrollar el proyecto asignado a modo de capacitación la llamamos “CampeonatoLigaFase”, la cual administraba todas aquellas fases correspondientes a una liga o un campeonato en particular. Por ejemplo, cuartos de final de un campeonato regional o fecha N de una liga regional.

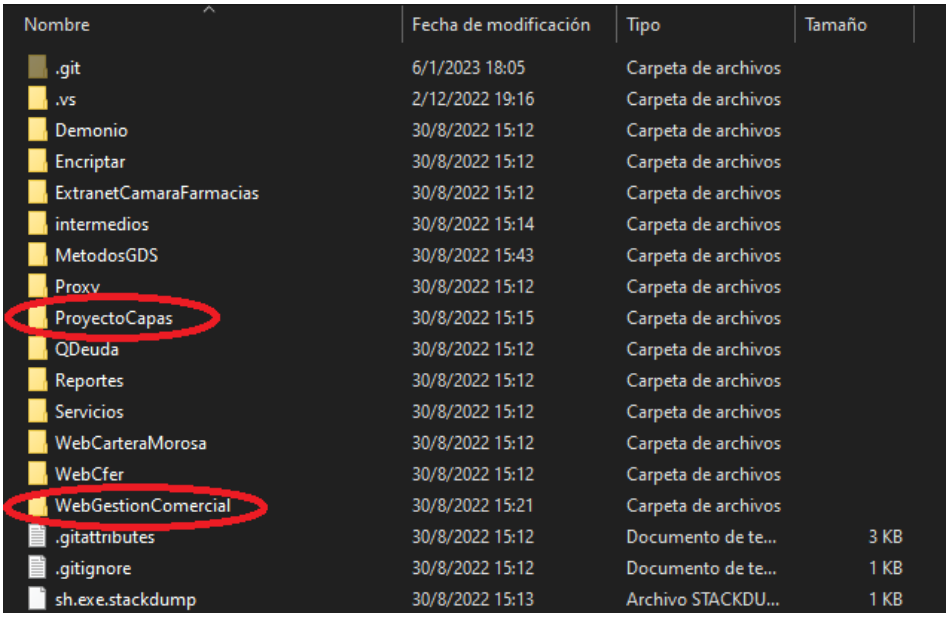
Después del CRUD de esta entidad pasamos a mejorar el desarrollo entidades ya creadas dentro del sistema como Partido, PartidoEvento, PartidoEtapas, PartidoPersonasInvolucradas, PosicionCampoJuego y SituacionDeJuego.

La gestión de este proyecto consistía en registrar a los deportistas del club, staff técnico, los deportes que realizaban o podían llegar a realizar y los eventos que sucedían dentro de un partido, ya sea de rugby, fútbol o de cualquier otra disciplina en la que el club participe o practique. Para esta gestión nos centramos en el rugby como disciplina deportiva.

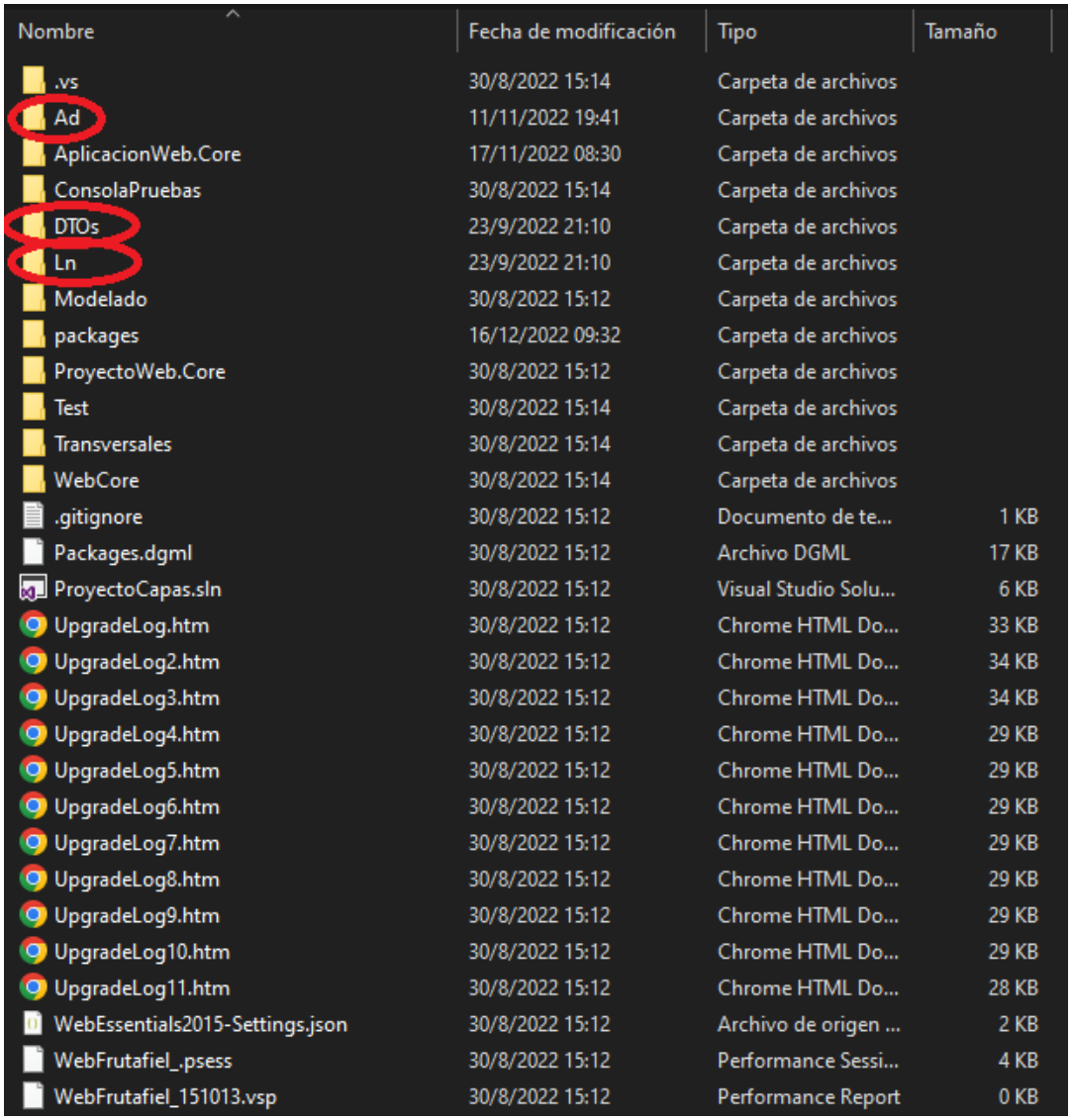
En cuanto al CRUD que realicé, el mismo consistió en las posiciones que ocupa un deportista dentro del campo de juego. Por lo que, para la prueba de concepto utilicé las posiciones que un jugador de rugby debía ocupar en la cancha. Sin embargo, al no estar familiarizado con las reglas y posiciones de este deporte tuve que realizar una investigación sobre los nombres de estas y el nombre que tiene cada sector de la cancha para poder especificar atributos relacionados a la posición que podía ocupar un deportista de esta disciplina.

Una vez familiarizado con estas posiciones (Pilar izquierdo, Hooker, Pilar derecho, dos Segundas línea, Ala lado ciego, Ala lado abierto, Número 8, Medio scrum, Apertura, Wing izquierdo, Centro interior, Centro exterior, Wing derecho y Full back) y con cada sector de la cancha (líneas de pelota muerta y líneas de touch-in-goal, líneas de goal, líneas de 22 metros, línea de mitad de cancha, líneas de touch), comencé a desarrollar esta entidad a partir del backend utilizado en otra entidad. Esto se fue así porque lo que se pretendía es que se respete la lógica de negocio implementada dentro de todo el sistema. De esta manera generé todos los archivos necesarios para comenzar con el CRUD al que llamamos "PosicionCampoJuego".

Generé un total doce (12) archivos y la edición de otro ya generado, los mismos se subdividían en dos directorios distintos, "ProyectoCapas" y "WebGestionComercial".

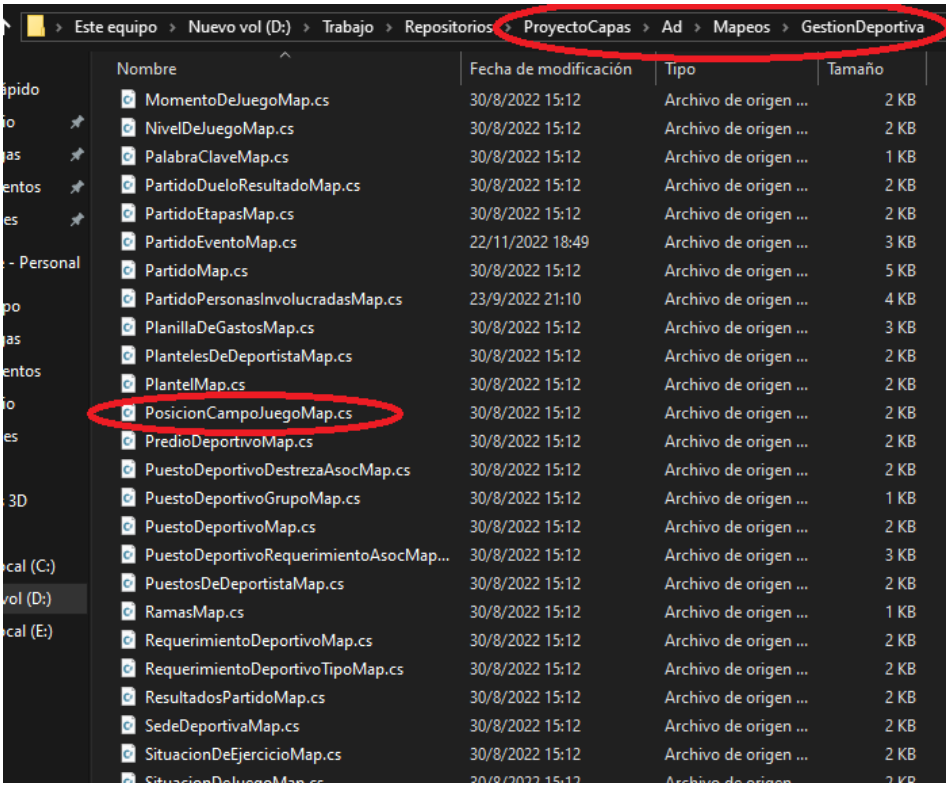


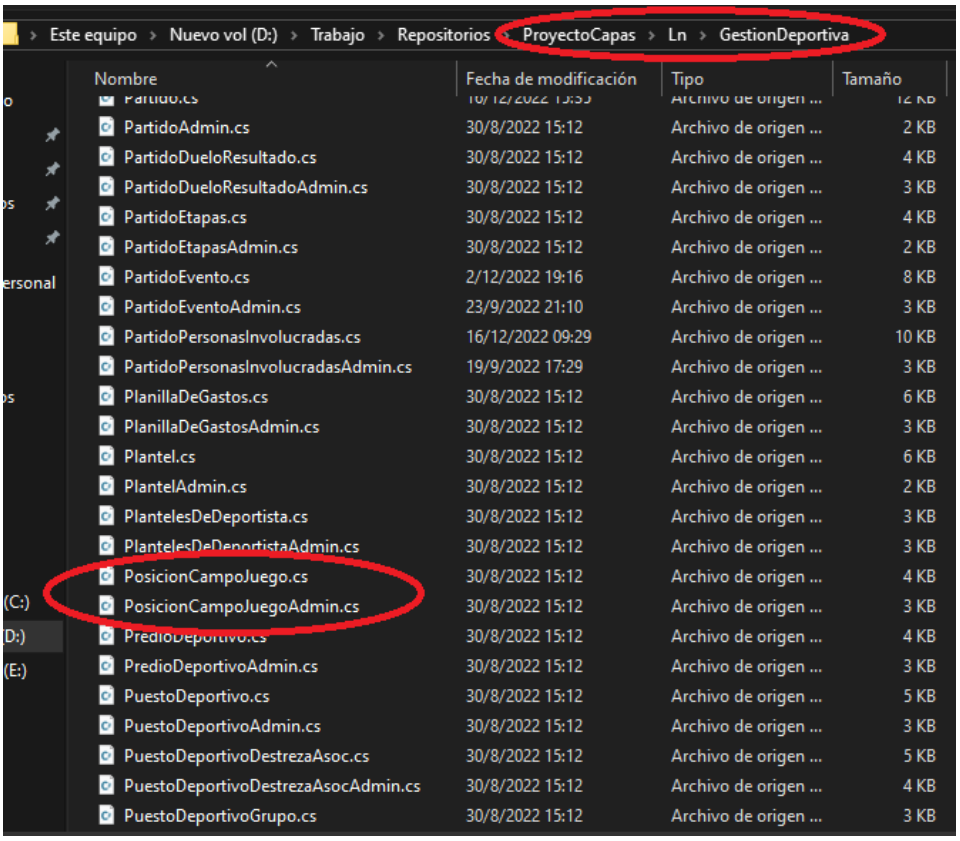
El directorio "ProyectoCapas", en el que existían tres directorios, "Ad", "DTOs" y "Ln".

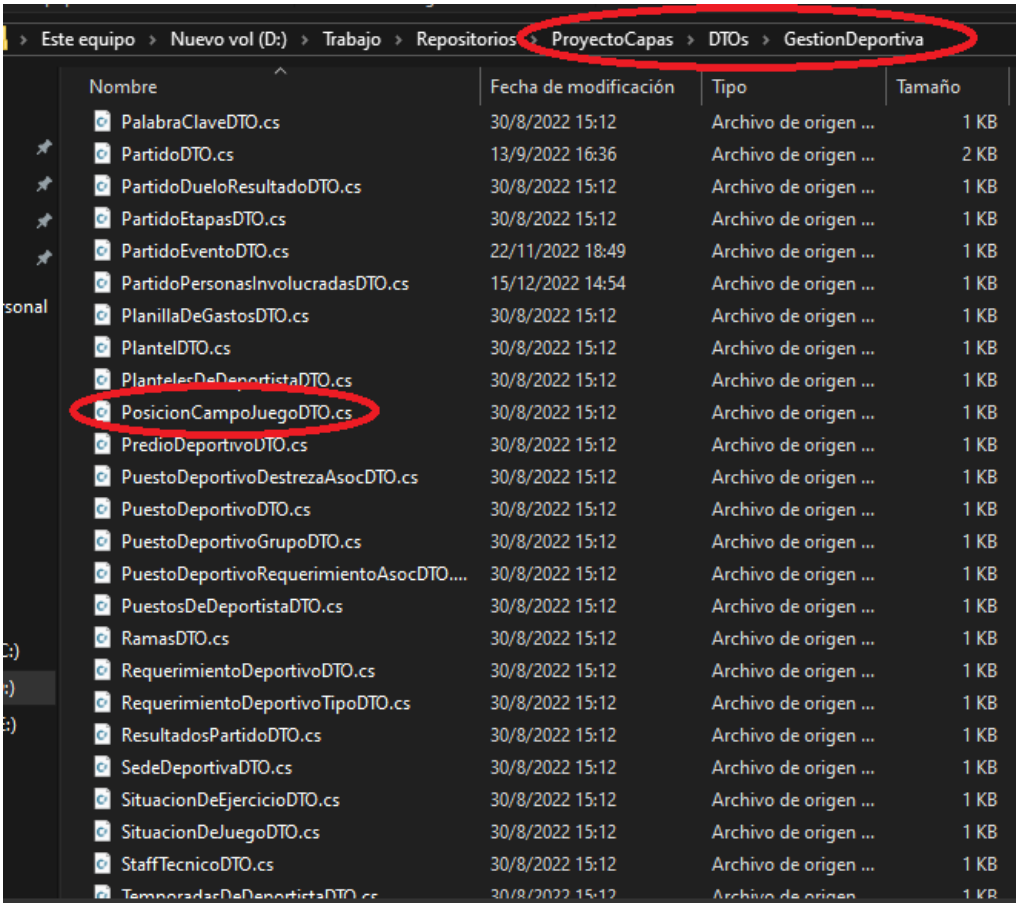


Nombre	Fecha de modificación	Tipo	Tamaño
.vs	30/8/2022 15:14	Carpeta de archivos	
Ad	11/11/2022 19:41	Carpeta de archivos	
AplicacionWeb.Core	17/11/2022 08:30	Carpeta de archivos	
ConsolaPruebas	30/8/2022 15:14	Carpeta de archivos	
DTOs	23/9/2022 21:10	Carpeta de archivos	
Ln	23/9/2022 21:10	Carpeta de archivos	
Modelado	30/8/2022 15:12	Carpeta de archivos	
packages	16/12/2022 09:32	Carpeta de archivos	
ProyectoWeb.Core	30/8/2022 15:12	Carpeta de archivos	
Test	30/8/2022 15:14	Carpeta de archivos	
Transversales	30/8/2022 15:14	Carpeta de archivos	
WebCore	30/8/2022 15:14	Carpeta de archivos	
.gitignore	30/8/2022 15:12	Documento de te...	1 KB
Packages.dgml	30/8/2022 15:12	Archivo DGML	17 KB
ProyectoCapas.sln	30/8/2022 15:12	Visual Studio Solu...	6 KB
UpgradeLog.htm	30/8/2022 15:12	Chrome HTML Do...	33 KB
UpgradeLog2.htm	30/8/2022 15:12	Chrome HTML Do...	34 KB
UpgradeLog3.htm	30/8/2022 15:12	Chrome HTML Do...	34 KB
UpgradeLog4.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog5.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog6.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog7.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog8.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog9.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog10.htm	30/8/2022 15:12	Chrome HTML Do...	29 KB
UpgradeLog11.htm	30/8/2022 15:12	Chrome HTML Do...	28 KB
WebEssentials2015-Settings.json	30/8/2022 15:12	Archivo de origen ...	2 KB
WebFrutafiel_.psess	30/8/2022 15:12	Performance Sessi...	4 KB
WebFrutafiel_151013.vsp	30/8/2022 15:12	Performance Report	0 KB

En cada uno de estos existía otro directorio más con el nombre “Gestión deportiva” en el cual se encontraban cada uno de estos archivos, el de mapeo, que llamamos “PosicionCampoJuegoMap.cs”, se encontraba dentro del directorio “Ad → Mapeos → GestionDeportiva”, dos archivos correspondientes a la lógica de negocio que llevaban el nombre “PosicionCampoJuego.cs” y “PosicionCampoJuegoAdmin.cs”, estos se encontraban en “Ln → Gestión deportiva”, y finalmente el archivo del DTO, que llevaba la nomenclatura “PosicionCampoJuegoDTO.cs”, y como en el caso de los tres primeros, se encontraba en el directorio “DTOs → Gestión deportiva”.


Este equipo › Nuevo vol (D:) › Trabajo › Repositorios › ProyectoCapas › Ad › Mapeos › GestionDeportiva
Nombre
Fecha de modificación
Tipo
Tamaño
MomentoDeJuegoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
NivelDeJuegoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PalabraClaveMap.cs 30/8/2022 15:12 Archivo de origen ... 1 KB
PartidoDueloResultadoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PartidoEtapasMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PartidoEventoMap.cs 22/11/2022 18:49 Archivo de origen ... 3 KB
PartidoMap.cs 30/8/2022 15:12 Archivo de origen ... 5 KB
PartidoPersonasInvolucradasMap.cs 23/9/2022 21:10 Archivo de origen ... 4 KB
PlanillaDeGastosMap.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PlantelesDeDeportistaMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PlantelMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PosicionCampoJuegoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PredioDeportivoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PuestoDeportivoDestrezaAsocMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PuestoDeportivoGrupoMap.cs 30/8/2022 15:12 Archivo de origen ... 1 KB
PuestoDeportivoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PuestoDeportivoRequerimientoAsocMap... 30/8/2022 15:12 Archivo de origen ... 3 KB
PuestosDeDeportistaMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
RamasMap.cs 30/8/2022 15:12 Archivo de origen ... 1 KB
RequerimientoDeportivoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
RequerimientoDeportivoTipoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
ResultadosPartidoMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
SedeDeportivaMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
SituacionDeEjercicioMap.cs 30/8/2022 15:12 Archivo de origen ... 2 KB


Este equipo › Nuevo vol (D:) › Trabajo › Repositorios › ProyectoCapas › Ln › GestionDeportiva
Nombre
Fecha de modificación
Tipo
Tamaño
PartidoAdmin.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PartidoDueloResultado.cs 30/8/2022 15:12 Archivo de origen ... 4 KB
PartidoDueloResultadoAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PartidoEtapas.cs 30/8/2022 15:12 Archivo de origen ... 4 KB
PartidoEtapasAdmin.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PartidoEvento.cs 2/12/2022 19:16 Archivo de origen ... 8 KB
PartidoEventoAdmin.cs 23/9/2022 21:10 Archivo de origen ... 3 KB
PartidoPersonasInvolucradas.cs 16/12/2022 09:29 Archivo de origen ... 10 KB
PartidoPersonasInvolucradasAdmin.cs 19/9/2022 17:29 Archivo de origen ... 3 KB
PlanillaDeGastos.cs 30/8/2022 15:12 Archivo de origen ... 6 KB
PlanillaDeGastosAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
Plantel.cs 30/8/2022 15:12 Archivo de origen ... 6 KB
PlantelAdmin.cs 30/8/2022 15:12 Archivo de origen ... 2 KB
PlantelesDeDeportista.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PlantelesDeDeportistaAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PosicionCampoJuego.cs 30/8/2022 15:12 Archivo de origen ... 4 KB
PosicionCampoJuegoAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PredioDeportivo.cs 30/8/2022 15:12 Archivo de origen ... 4 KB
PredioDeportivoAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PuestoDeportivo.cs 30/8/2022 15:12 Archivo de origen ... 5 KB
PuestoDeportivoAdmin.cs 30/8/2022 15:12 Archivo de origen ... 3 KB
PuestoDeportivoDestrezaAsoc.cs 30/8/2022 15:12 Archivo de origen ... 5 KB
PuestoDeportivoDestrezaAsocAdmin.cs 30/8/2022 15:12 Archivo de origen ... 4 KB
PuestoDeportivoGrupo.cs 30/8/2022 15:12 Archivo de origen ... 3 KB



Una vez generados estos archivos comencé a adaptar el backend en cuanto a las necesidades de la entidad.

El archivo “PosicionCampoJuegoMap.cs” se encargaba del mapeo de la entidad “PosicionCampoJuego” con la tabla “DE_posicion_campo_juego” usando Entity Framework como ORM.

Como se observa en la siguiente imagen, la clase “PosicionCampoJuegoMap” hereda de “EntityTypeConfiguration” para la configuración de la tabla en la base de datos a partir de un tipo específico de entidad que se pasa entre los picos, en este caso, “PosicionCampoJuego”. A partir de ahí se observa en la declaración del constructor de esta la palabra reservada “this” seguido de los métodos heredades de “EntityTypeConfiguration” para poder armar el objeto y así configurar la entidad que se va a crear. Los métodos que utiliza son: “ToTable”, donde se configura el nombre de la tabla al que se asigna la entidad “PosicionCampoJuego”; “HasKey”, para configurar la clave primaria de esta entidad; “Property”, para determinar los atributos según el tipo especificado de los atributos de la entidad, el método “IsRequired” para que la propiedad sea obligatoria y el método

“HasColumnName” para asignar el nombre que llevará la columna en la base de datos; y para la cardinalidad y las relaciones los métodos “WithMany” y “HasForeignKey” respectivamente, así como también la llamada al método “HasRequired” para indicar la obligatoriedad de este atributo como relación en la base de datos. En resumen, se declara y configura cómo es que se va a conformar la tabla dentro de la base de datos con sus atributos, relaciones y cardinalidad.

```
public class PosicionCampoJuegoMap : EntityTypeConfiguration<PosicionCampoJuego>
{
    public PosicionCampoJuegoMap(string nombreEsquema)
    {
        //Table
        this.ToTable("DE_posicion_campo_juego", nombreEsquema);

        //Clave principal
        this.HasKey(t =>t.CodigoPosicionCampoJuego);

        //Propiedades
        this.Property(t => t.CodigoPosicionCampoJuego)
        .IsRequired()
        .HasColumnName("Codigoposicioncampojuego");

        this.Property(t => t.Descripcion)
        .IsRequired()
        .HasColumnName("Descripcion");

        this.Property(t => t.CodigoDisciplina)
        .IsRequired()
        .HasColumnName("codigodiciplina");

        //Relaciones
        this.HasRequired(t => t.DisciplinaDeportiva)
        .WithMany()
        .HasForeignKey(t => t.CodigoDisciplina);
    }
}
```

“PosicionCampoJuegoDTO.cs” define tres clases, necesarias para representar el DTO (Data Transfer Object), la clase “PosicionCampoJuegoDTO”, que contiene dos propiedades públicas: “Descripcion” y “CodigoDisciplina”, para representar la descripción de la posición y a qué disciplina corresponde la posición para armar el objeto que se está consultando desde la base de datos. Por su parte, la clase PosicionCampoJuegoNuevoDTO hereda de la anterior y declara la propiedad pública “CodigoPosicionCampoJuego” para crear, el conjunto con las propiedades heredadas, el objeto de una nueva posición en el campo de juego. Finalmente, la clase “PosicionCampoJuegoModificarDTO”, que también hereda de la primera, pero sin agregar nuevas propiedades, la modificación de una posición en el campo de juego.

```
public class PosicionCampoJuegoDTO
{
    public string Descripcion;
    public string CodigoDisciplina;
}

public class PosicionCampoJuegoNuevoDTO : PosicionCampoJuegoDTO
{
    public string CodigoPosicionCampoJuego;
}

public class PosicionCampoJuegoModificarDTO : PosicionCampoJuegoNuevoDTO
{

}
```

"PosicionCampoJuego.cs" define la entidad "PosicionCampoJuego" y representa, justamente, una posición dentro del campo de juego. Esta clase implementa la interfaz "IEntidadAuditable" para auditar una o más acciones que puedan realizarse sobre ella. Dentro de esta clase se declara la constante privada de tipo string "CLASE_CODIGO_ENTIDAD" y se le asigna un nombre para utilizar posteriormente sobre el frontend.

```
public class PosicionCampoJuego : IEntidadAuditable
{
    private const string CODIGO_CLASE_ENTIDAD = "POSCAJUEGO";
```

Luego, en el código, se visualizan cuatro regiones. La primera es la región en donde se declaran cuatro propiedades públicas, tres de tipo string y que además pueden ser sobrescritas por clases derivadas de esta, y otra de tipo "DisciplinaDeportiva" para poder identificar el código de la disciplina al que pertenece la posición.

```
#region Propiedades Públicas
public virtual string CodigoPosicionCampoJuego { get; set; }
public virtual string Descripcion { get; set; }
public virtual string CodigoDisciplina { get; set; }
public virtual DisciplinaDeportiva DisciplinaDeportiva { get; set; }
#endregion
```

La región en que se encuentran declarados tres métodos internal de tipo void, "Alta", "Modificar" y "Eliminar", los dos primeros reciben dos parámetros y el último recibe un único parámetro.

"Alta": recibe un objeto de tipo "PosicionCampoJuegoNuevoDTO" con los datos necesarios para la creación de una nueva posición y un objeto de tipo "IUnitOfWork" para realizar el registro de una instancia del tipo "PosicionCampoJuego" para dar de alta en la

base de datos. En cuanto las acciones, implementa los métodos “RegisterNew” de la interfaz “IUnitOfWork” para realizar el registro en la base de datos de esta nueva posición y “setPropiedadesNuevo” que recibe dos objetos como argumentos, el objeto “dto” de tipo “PosicionCampoJuegoNuevoDTO” y el objeto “uow” de tipo “IUnitOfWork” para ejecutar esta transacción.

“Modificar”: recibe un objeto de tipo “PosicionCampoJuegoModificarDTO” con los datos de la posición a modificar en la base de datos y también un objeto “IUnitOfWork” para realizar esta modificación. A diferencia del método “Alta”, este implementa el método “setPropiedadesModificar” que recibe dos objetos como argumentos, el objeto “dto” de tipo “PosicionCampoJuegoModificarDTO” con los datos que se van a modificar de la posición y el objeto “uow” de tipo “IUnitOfWork” para ejecutar la transacción.

“Eliminar”: recibe como parámetro un objeto de tipo “IUnitOfWork” encargado de llamar al método “RegisterDeleted” y que permite poder realizar la baja de la instancia de una posición en la base de datos.

```
#region Métodos internal
internal void Alta(PosicionCampoJuegoNuevoDTO dto, IUnitOfWork uow)
{
    uow.RegisterNew<PosicionCampoJuego>(this);
    this.setPropiedadesNuevo(dto, uow);
}

internal void Modificar(PosicionCampoJuegoModificarDTO dto, IUnitOfWork uow)
{
    this.setPropiedadesModificar(dto, uow);
}

internal void Eliminar(IUnitOfWork uow)
{
    uow.RegisterDeleted<PosicionCampoJuego>(this);
}
#endregion
```

En la siguiente región se encuentran cinco métodos privados, “setPropiedadesNuevo” “setPropiedadesModificar”, “setCodigoPosicionCampoJuego”, “setDescripcion” y “setCodigoDisciplina”:

“setPropiedadesNuevo”: establece las propiedades de una nueva posición y recibe dos parámetros, “dto” de tipo “PosicionCampoJuegoNuevoDTO” y “uow” de tipo

“IUnitOfWork” para responder al llamado desde el método internal “Alta” y así poder ejecutar el registro de una nueva posición en la base de datos.

“setPropiedadesModificar”: establece las propiedades de una nueva posición y recibe dos parámetros, “dto” de tipo “PosicionCampoJuegoModificarDTO” y “uow” de tipo “IUnitOfWork” para responder al llamado desde el método internal “Modificar” y así poder ejecutar el registro de la modificación de una posición en la base de datos.

“setCodigoPosicionCampoJuego”: este método setea la propiedad “CodigoPosicionCampoJuego” de la clase “PosicionCampoJuego”.

“setDescripcion”: este setea la propiedad “Descripcion” de esta misma clase y lanza una excepción en caso de que el parámetro recibido de tipo string sea nulo o se encuentre vacío

“setCodigoDisciplina”: setea la propiedad “CodigoDisciplina” siempre que no se lancen alguna de las dos excepciones especificadas, la que se lanza en caso de que el parámetro de tipo string sea nulo o se encuentre vacío bien la excepción que se lanza en caso de que el código de esa disciplina no exista en la base de datos.

```
#region Métodos privados
private void setPropiedadesNuevo(PosicionCampoJuegoNuevoDTO dto, IUnitOfWork uow)
{
    this.setCodigoPosicionCampoJuego(dto.CodigoPosicionCampoJuego);
    this.setDescripcion(dto.Descripcion);
    this.setCodigoDisciplina(dto.CodigoDisciplina, uow);
}

private void setPropiedadesModificar(PosicionCampoJuegoModificarDTO dto, IUnitOfWork uow)
{
    this.setCodigoPosicionCampoJuego(dto.CodigoPosicionCampoJuego);
    this.setDescripcion(dto.Descripcion);
    this.setCodigoDisciplina(dto.CodigoDisciplina, uow);
}
private void setCodigoPosicionCampoJuego(string CodigoPosicionCampoJuego)
{
    this.CodigoPosicionCampoJuego = CodigoPosicionCampoJuego;
}

private void setDescripcion(string Descripcion)
{
    if (string.IsNullOrWhiteSpace(Descripcion))
    {
        throw new ExcepcionPropiedadRequerida("Una posición de campo de juego requiere una descripción.", "Descripcion");
    }
    this.Descripcion = Descripcion;
}

private void setCodigoDisciplina(string codigoDisciplina, IUnitOfWork uow)
{
    if (string.IsNullOrWhiteSpace(codigoDisciplina))
    {
        throw new ExcepcionPropiedadRequerida(Mensajes.SituacionDeJuegoRequiereCodigoDisciplina, "CodigoDisciplina");
    }
    if (!uow.Queryable<DisciplinaDeportiva>().Any(s => s.CodigoDisciplina == codigoDisciplina))
    {
        throw new ExcepcionEntidadInexistente(Mensajes.DisciplinaDeportivaInexistente, typeof(DisciplinaDeportiva));
    }
    this.CodigoDisciplina = codigoDisciplina;
}
#endregion
```

Finalmente, la región de “Miembros IRecursoAuditable” implementa la propiedad de la interfaz “CodigoEntidad” que devuelve el código con el que se identifica a la clase “PosicionCampoJuego” y el método “EstadoAuditar” que devuelve el objeto PosicionCampoJuego con la información que se debe auditar sobre el estado de esta entidad.

```
#region Miembros IRecursoAuditable
public string CodigoEntidad
{
    get { return CODIGO_CLASE_ENTIDAD; }
}

public object EstadoAuditar()
{
    return new
    {
        this.CodigoPosicionCampoJuego,
        this.Descripcion,
        this.CodigoDisciplina,
    };
}
#endregion
```

Finalmente, en el archivo “PosicionCampoJuegoAdmin.cs” se declara la clase “PosicionCampoJuegoAdmin” la cual se encarga de proporcionar las operaciones para la administración de las posiciones dentro del campo de juego. Implementa la interfaz “IUnitOfWork” para el registro de las posiciones en la base de datos, realizando la instancia de un objeto del tipo de esta interfaz.

```
public class PosicionCampoJuegoAdmin
{
    private IUnitOfWork _uow;
```

Se implementa un constructor de esta clase que recibe como parámetro un objeto del tipo “IUnitOfWork” que se asigna sobre la instancia “_uow”.

```
public PosicionCampoJuegoAdmin(IUnitOfWork uow)
{
    this._uow = uow;
}
```

El método “AltaPosicionCampoJuego” de tipo “PosicionCampoJuego” recibe como parámetro un objeto “dto” de tipo “PosicionCampoJuegoDTO” con la información requerida para crear una nueva posición. Dentro se crea el objeto “PosicionCampoJuego” del tipo “PosicionCampoJuego” estableciendo las propiedades con los datos del objeto “dto” para luego llamar al método “Alta” que recibe los argumentos “dto” y “_uow” para finalmente realizar el registro de la posición y devolver la misma.

```
public PosicionCampoJuego AltaPosicionCampoJuego(PosicionCampoJuegoNuevoDTO dto)
{
    PosicionCampoJuego PosicionCampoJuego = _uow.CreateNew<PosicionCampoJuego>();
    PosicionCampoJuego.Alta(dto, _uow);
    _uow.Save();
    return PosicionCampoJuego;
}
```

“ModificarPosicionCampoJuego” tiene una acción similar al método anterior, ya que recibe como parámetro un objeto “dto” pero de tipo “PosicionCampoJuegoModificarDTO”. Sin embargo, antes de realizar el llamado al método “Modificar”, se realiza la creación de un objeto “PosicionCampoJuego” del tipo “PosicionCampoJuego” a partir de la asignación de la instancia producida por el objeto “_uow” al llamar al método “TryEnsureRegistered” que trae la consulta de la base de datos en base al código de la posición devolviendo el objeto con toda la información en caso de existir un registro en la base de datos o null en caso contrario. Consiguientemente se produce el manejo de la excepción “ExcepcionLn” que verifica si el

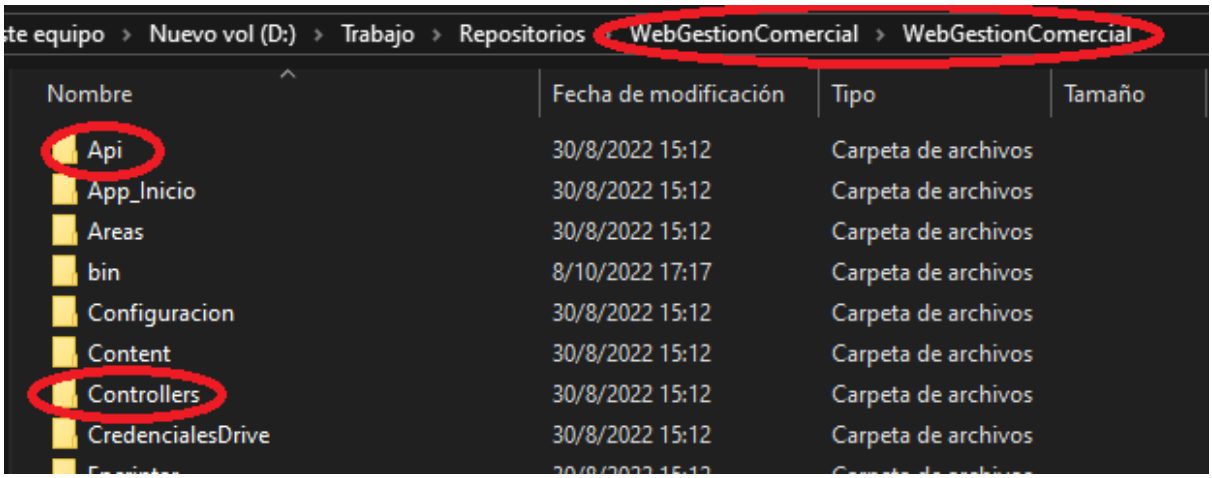
objeto creado es null, en este caso se lanza la excepción y se informa que la posición no existe, y si no lo es, se produce el llamado al método “Modificar” donde se pasan dos argumentos, el “dto” con la información a modificar y el objeto “_uow”, este último registrará con el llamado al método “Save” la actualización del registro en la base de datos. Finalmente se devuelve el objeto “PosicionCampoJuego” para continuar con la transacción.

```
public PosicionCampoJuego ModificarPosicionCampoJuego(PosicionCampoJuegoModificarDTO dto)
{
    PosicionCampoJuego PosicionCampoJuego = _uow.TryEnsureRegistered<PosicionCampoJuego>(dto.CodigoPosicionCampoJuego);
    if (PosicionCampoJuego == null)
    {
        throw new ExcepcionLn(string.Format("Posición de campo de juego inexistente", dto.CodigoPosicionCampoJuego));
    }
    PosicionCampoJuego.Modificar(dto, _uow);
    _uow.Save();
    return PosicionCampoJuego;
}
```

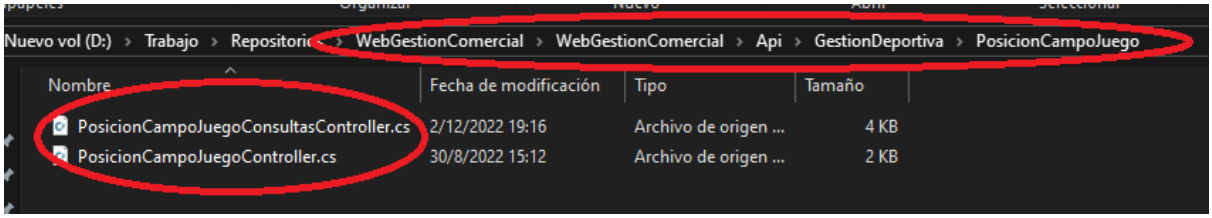
El método “EliminarPosicionCampoJuego” tiene un decorador que se utiliza para indicar una acción que debe ser auditada, lleva la nomenclatura “AccionAuditableSobreEntidad”, a la cual se le pasa como argumento un “código de acción”, en este caso puntual, “ELIMINAR”. Este método recibe como parámetro una variable de tipo string “CodigoPosicionCampoJuego”, que no es ni más ni menos que el código de la posición que se desea eliminar del sistema. Como se produjo en el método “ModificarPosicionCampoJuego”, se asigna a un objeto “PosicionCampoJuego” del tipo “PosicionCampoJuego” la instancia producida por el objeto “_uow” al llamar al método “TryEnsureRegistered” que trae la consulta de la base de datos. Se llama al método eliminar y se produce un manejo de excepción en el caso de que la posición a eliminar del sistema esté vinculada a otra entidad, en este caso se lanza la excepción y se emite el mensaje “Existe entidades relacionada a la entidad que desea eliminar, elimine esas primero.”, de lo contrario se elimina el registro de la base de datos con el llamado del objeto “_uow” al método “Save”.

```
[AccionAuditableSobreEntidad(Accion.Codigos.ELIMINAR)]
public void EliminarPosicionCampoJuego(string CodigoPosicionCampoJuego)
{
    PosicionCampoJuego PosicionCampoJuego = _uow.TryEnsureRegistered<PosicionCampoJuego>(CodigoPosicionCampoJuego);
    try
    {
        PosicionCampoJuego.Eliminar(_uow);
        _uow.Save();
    }
    catch (Exception e)
    {
        throw new ExcepcionLn(string.Format(Mensajes.EntidadesRelacionadasAlEliminar));
    }
}
```

Por otro lado, fue necesario trabajar sobre otro directorio dentro del proyecto, “WebGestionComercial” y, dentro de este, se encontraba otro directorio más con el mismo nombre. En este último se utilizaron dos directorios más para terminar con el backend, Api y Controllers.



Dentro del directorio Api, específicamente en el directorio “GestionDeportiva”, fue necesario generar dos archivos .cs, uno que especificaba la clase “PosicionCampoJuegoConsultasController” para realizar consultas a la base de datos y el otro archivo que especificaba la clase “PosicionCampoJuegoController” encargada de realizar la ejecución del ABM de la entidad “PosiciónCampoJuego”. La ruta completa era “WebGestionComercial → WebGestionComercial → Api → GestionDeportiva”



Con relación al código, el archivo “PosicionCampoJuegoConsultasController.cs” declara la clase “PosicionCampoJuegoConsultasController” que hereda de “ApiController”. En ella se instancia un objeto “_consultasRepositorio” del tipo de la interfaz “IConsultasRepositorio” que se encargará de ejecutar las consultas a la base de datos.

```
public class PosicionCampoJuegoConsultasController : ApiController
{
    private IConsultasRepositorio _consultasRepositorio;
```

El constructor de esta clase recibe como parámetro un objeto del tipo de la interfaz mencionada con anterioridad y simplemente lo asigna al objeto "_consultasRepositorio".

```
public PosicionCampoJuegoConsultasController(IConsultasRepositorio consultasRepositorio)
{
    _consultasRepositorio = consultasRepositorio;
}
```

Dentro de esta clase se declara otra clase "PosicionCampoJuegoDatos" para armar el objeto que contendrá toda la información relacionada a una posición del campo de juego.

```
public class PosicionCampoJuegoDatos
{
    public string CodigoPosicionCampoJuego;
    public string Descripcion;
    public string CodigoDisciplina;
    public string NombreDisciplina;
}
```

El método "GetPosicionCampoJuego" recibe como parámetro el código de la posición buscada, además cuenta con tres decoradores, "[AutorizarApi]" para verificar que el usuario tenga el permiso para realizar esa consulta a la base de datos, "[AutorizarWebApi]" para verificar que la acción realizada por el usuario esté permitida, en este caso para realizar la consulta a la base de datos sobre una posición en particular dentro del campo de juego, y "[HttpGet]" es para indicar la solicitud HTTP GET y ejecutar la acción de devolver los datos indicados como consulta dentro del método. Finalmente, retorna el resultado de la consulta a la base de datos a partir de la expresión lambda utilizando el llamado a los métodos "Queryable<T>.Where()" y "Queryable<T>.Select()", donde el primero filtra la colección de posiciones de campo de juego para que solamente queden las que coincidan con el código especificado, mientras que el segundo convierte las posiciones de campo de juego que coinciden en objetos del tipo "PosicionCampoJuegoDatos", y el método "FirstOrDefault" devuelve el primer elemento de la colección o null si la misma se encuentra vacía.

```
[AutorizarApi]
[AutorizarWebApi(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.CONSULTAR)]
[HttpGet]
public PosicionCampoJuegoDatos GetPosicionCampoJuego (string codigo)
{
    return _consultasRepositorio.Queryable<PosicionCampoJuego>()
                                .Where(u => u.CodigoPosicionCampoJuego == codigo)
                                .Select(a => new PosicionCampoJuegoDatos
                                {
                                    CodigoPosicionCampoJuego = a.CodigoPosicionCampoJuego,
                                    Descripcion = a.Descripcion,
                                    CodigoDisciplina = a.CodigoDisciplina,
                                    NombreDisciplina = a.DisciplinaDeportiva.Nombre
                                ,
                                }).FirstOrDefault();
}
```

El siguiente código utiliza los mismos decoradores utilizados en el método “GetPosicionCampoJuego”, en este caso recibe tres parámetros, “hint” de tipo string, “inicio” y “cantidad”, ambos de tipo int. Se llama al método “Palabras” y se asigna el resultado a la variable “palabras”, en caso de que el parámetro “hint” sea nulo se asigna una cadena vacía. Si la longitud de la variable es cero se lanza una excepción. Luego se establecen en cero los valores de los parámetros “inicio” y “cantidad” en caso de que estos sean menos a cero; seguidamente se ejecuta el llamado al método “Queryable<PosicionCampoJuego>()” para obtener todas las instancias de este tipo y consiguientemente la estructura “foreach” para filtrar aquellas instancias del tipo “PosicionCampoJuego” que contengan alguna de las palabras en “Descripcion”, “CodigoPosicionCampoJuego” o “CodigoDisciplina”; después se seleccionan la instancias filtradas del tipo “PosicionCampoJuego” y se asignan a los atributos del objeto de tipo “PosicionCampoJuegoDatos” para finalmente aplicar los límites de inicio y cantidad de los elementos que se van a retornar. Por último, se ordenan las posiciones del campo de juego de menor a mayor y se convierte a una lista. En resumen, la funcionalidad de este método es devolver un listado de posiciones del campo de juego por medio de una búsqueda por palabras.

El método “Palabras”, definido como privado y estático, recibe como parámetro un string y retorna un arreglo de este mismo tipo. El parámetro recibido llama al método “Split” de la clase String para armar una subcadena utilizando el carácter espacio como separador, luego se ejecuta una estructura “for” que itera sobre la colección “palabras” para eliminar aquellos espacios en blanco iniciales y finales de cada elemento del arreglo de tipo string y, finalmente, devuelve la colección. La funcionalidad de este método es exclusivamente para ser utilizado en una consulta de búsqueda, es por eso que se utiliza en el método

"GetPosicionCampoJuego" sobre la variable "palabras" al comienzo de la ejecución de este método.

```
[AutorizarApi]
[AutorizarWebApi(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.CONSULTAR)]
[HttpGet]
public ICollection<PosicionCampoJuegoDatos> PosicionCampoJuegos(string hint, int inicio, int cantidad)
{
    var palabras = Palabras(hint ?? "");
    if (palabras.Length == 0)
    {
        throw new ExcepcionWeb(string.Format(GyA.ProyectoCapas.WebCore.Recursos.Mensajes.AccionArgumentoNoValido, "hint"));
    }
    if (inicio < 0) inicio = 0;
    if (cantidad < 0) cantidad = 0;

    var todas = _consultasRepositorio.Queryable<PosicionCampoJuego>();
    foreach (var palabra in palabras)
    {
        todas = todas.Where(a => a.Descripcion.ToString().Contains(palabra) || a.CodigoPosicionCampoJuego.ToString().Contains(palabra)
        ||  a.CodigoDisciplina.Contains(palabra));
    }

    var ItemsADevolver = todas.Select(a => new PosicionCampoJuegoDatos
    {
        CodigoPosicionCampoJuego = a.CodigoPosicionCampoJuego,
        Descripcion = a.Descripcion,
        CodigoDisciplina = a.CodigoDisciplina,
        NombreDisciplina = a.DisciplinaDeportiva.Nombre,
    });

    if (inicio > 0)
        ItemsADevolver = ItemsADevolver.Skip(inicio);
    if (cantidad > 0)
        ItemsADevolver = ItemsADevolver.Take(cantidad);

    var response = ItemsADevolver.OrderBy(o => o.Descripcion).ToList();
    return response;
}
```

```
private static string[] Palabras(string texto)
{
    var palabras = texto.Split(' ');
    for (int i = 0, l = palabras.Length; i < l; ++i)
    {
        palabras[i] = palabras[i].Trim();
    }
    return palabras;
}
```

Finalmente, respecto a la clase "PosicionCampoJuegoController" del archivo "PosicionCampoJuegoController.cs" dentro del directorio "Api", se declara esta clase que también hereda de "ApiController". En ella se instancia un objeto "_PosicionCampoJuegoAdmin" del tipo "PosicionCampoJuegoAdmin" que se encargará, como ya mencioné, de administrar todas aquellas operaciones relacionadas de las posiciones del campo de juego.

La clase cuenta con un constructor, que inicializa el objeto instanciado de tipo "PosicionCampoJuegoAdmin" con el valor de un objeto del mismo tipo que es pasado por parámetro. También cuenta con otros tres métodos para realizar las operaciones del ABM:

- "Alta" que recibe como parámetro un objeto de tipo "PosicionCampoJuegoNuevoDTO" y que llama al método "AltaPosicionCampoJuego" para crear una nueva posición del campo de juego
- "Baja" que recibe un string como parámetro y que llama al método "EliminarPosicionCampoJuego" para eliminar una posición del campo de juego
- "Modificar" que recibe un objeto de tipo "PosicionCampoJuegoModificarDTO" como parámetro y que modifica los datos de una posición del campo de juego.

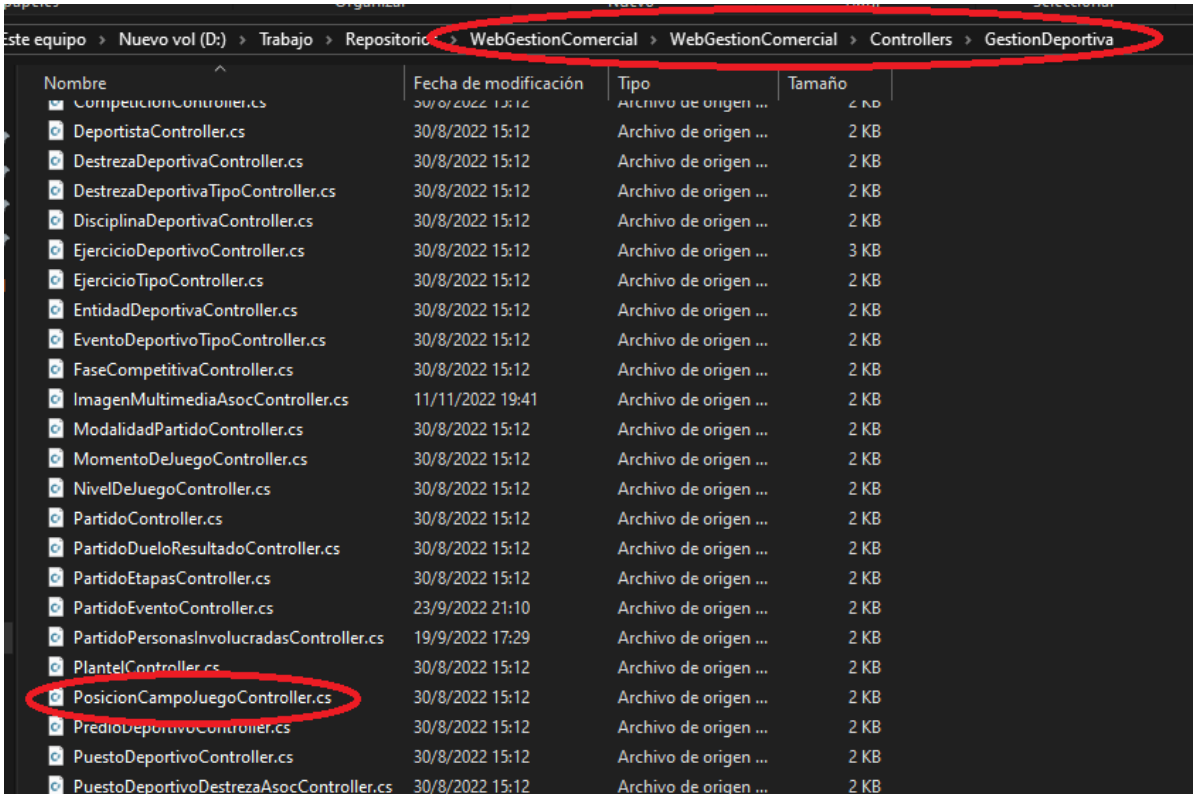
```
public class PosicionCampoJuegoController : ApiController
{
    private PosicionCampoJuegoAdmin _PosicionCampoJuegoAdmin;

    public PosicionCampoJuegoController (PosicionCampoJuegoAdmin PosicionCampoJuegoAdmin)
    {
        _PosicionCampoJuegoAdmin = PosicionCampoJuegoAdmin;
    }

    [AutorizarApi]
    [AutorizarWebApi(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.ALTA)]
    public void Alta(PosicionCampoJuegoNuevoDTO datos)
    {
        _PosicionCampoJuegoAdmin.AltaPosicionCampoJuego(datos);
    }
    [HttpDelete]
    [AutorizarApi]
    [AutorizarWebApi(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.ALTA)]
    public void Baja(string codigo)
    {
        _PosicionCampoJuegoAdmin.EliminarPosicionCampoJuego(codigo);
    }

    [AutorizarApi]
    [AutorizarWebApi(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.ALTA)]
    public void Modificar(PosicionCampoJuegoModificarDTO nuevosdatos)
    {
        _PosicionCampoJuegoAdmin.ModificarPosicionCampoJuego(nuevosdatos);
    }
}
```

Dentro del directorio Controller, así como en el anterior en "GestionDeportiva", generé un archivo .cs en el que se declaraba la clase "PosicionCampoJuegoController" (misma nomenclatura que el archivo creado dentro de la carpeta Api) encargada de administrar y controlar las vistas de la aplicación.



Respecto al código, la clase “PosicionCampoJuegoController” que hereda de “Controller” representa un controlador MVC para manejar aquellas acciones relacionadas a la administración, creación y edición de una posición del campo de juego, asegurando la autorización de estas acciones por medio del decorador [AutorizarMvc(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.VER)]”.

Cada método declarado es de tipo “ActionResult” y devuelven un modelo de vista con el nombre especificado. En el caso del método “AdministrarPosicionCampoJuego” crea un objeto “modelo” de tipo “EtiquetasModel” para retornar una vista parcial “PartialView” con el “modelo” creado.

Los métodos “NuevoPosicionCampoJuego” y “EditarPosicionCampoJuego”, si bien también crean un objeto “modelo” de tipo “EtiquetasModel”, lo que devuelven es una vista “View” con el nombre de la vista y el modelo creado.

```
public class PosicionCampoJuegoController : Controller
{
    [AutorizarMvc(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.VER)]
    public ActionResult AdministrarPosicionCampoJuego()
    {
        var modelo = new EtiquetasModel();
        return PartialView(modelo);
    }
    [AutorizarMvc(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.VER)]
    public ActionResult NuevoPosicionCampoJuego()
    {
        var modelo = new EtiquetasModel();
        return View("NuevoEditarPosicionCampoJuego", modelo);
    }
    [AutorizarMvc(CodigoPrograma = Programa.Codigos.POSICIONCAMPOJUEGO, CodigoAccion = Accion.Codigos.VER)]
    public ActionResult EditarPosicionCampoJuego()
    {
        var modelo = new EtiquetasModel();
        return View("NuevoEditarPosicionCampoJuego", modelo);
    }
}
```

Del lado del frontend fue necesario generar los archivos restantes, siguiendo la misma lógica estructural de directorios. En este contexto, se trabajó sobre dos directorios Scripts y Views. Los mismos se encontraban en “WebGestionComercial → WebGestionComercial → Scripts” y “WebGestionComercial → WebGestionComercial → Views” respectivamente.

Para el directorio Views generé dos archivos HTML, ambos se localizaban en la ruta “WebGestionComercial → WebGestionComercial → Views → PosicionCampoJuego”, uno, “AdministrarPosicionCampoJuego.cshtml”, visualizaba la lista de posiciones dentro del campo de juego con los botones necesarios para la administración de estas posiciones vinculando estos archivos HTML con los archivos JavaScript que detallaré en el siguiente párrafo. Y el otro “NuevoEditarPosicionCampoJuego.cshtml” que se encargaba de mostrar al usuario el formulario para la creación o edición de una posición dentro del campo de juego.

Para el directorio Scripts, generé cuatro archivos JavaScript. Tres correspondientes a la posición dentro del campo de juego: “posicioncampojuego-administrar-vm.js”, “posicioncampojuego-nuevo-editar-vm.js” y “posicioncampojuego-mod.js”. Y el otro “aplicación-gestion-comercial.js”.

Los tres primeros se encontraban en la ruta “WebGestionComercial → WebGestionComercial → Scripts → PosicionCampoJuego”, mientras que el restante se encontraba en “WebGestionComercial → WebGestionComercial → Scripts → AppGestionComercial”.

El archivo JavaScript “aplicación-gestion-comercial.js” vinculaba el backend de todas las entidades del sistema de la empresa con el frontend de las mismas.

Por el lado de los archivos JavaScript de las posiciones dentro del campo de juego “posicioncampojuego-administrar-vm.js” se encargaba de manejar la lógica de administración de las posiciones. Este archivo administraba las funciones de los botones del HTML “AdministrarPosicionCampoJuego.cshtml” para eliminar una posición, editarla o bien un botón para generar una nueva, llamando al archivo “posicioncampojuego-nuevo-editar-vm.js” quien manejaba el formulario dedicado a la creación o edición de estas.

Personas involucradas en un partido: “PartidoPersonasInvolucradas”

Luego de este ABM trabajé en equipo con otro compañero de las prácticas en las entidades necesarias para administrar un partido, desde las personas involucradas como lo son los deportistas y el staff técnico de ambos equipos y personas involucradas que no pertenecieron a ninguno de los equipos participantes del partido en sí, como lo son las situaciones de juego o sucesos que se produjesen dentro del partido, como pueden ser las conversiones, trys o penales o las faltas. Sumando también a estas entidades mencionadas, otras como la competición a la que se encontraba afectado el partido, es decir, un partido de copa, un partido de liga, un partido amistoso u otro.

Además de estas situaciones de juego mencionadas, también existían eventos que afectaban al estado del partido, es decir, si el mismo se encontraba en juego, o si ya había finalizado o si, por el contrario, se encontraba suspendido.

En este contexto fue necesario que nos dividamos entidades para poder avanzar con la tarea que nos asignaron como equipo. En mi situación generé la entidad “PartidoPersonasInvolucradas”, es decir, todas aquellas personas que se involucraron de alguna forma dentro de un partido.

Estas personas eran los jugadores, el staff técnico y aquellas personas que no pertenecían al staff, es decir, personas ajenas a los equipos. Por lo que para esta entidad fue necesario separar a los jugadores del equipo local del equipo visitante, lo mismo sucedió con los integrantes del staff técnico.

Las entidades de los deportistas y del staff técnico se encontraban generadas, razón por la cual fueron utilizadas para la creación de esta entidad “PartidoPersonasInvolucradas”.

Además de estas, fue necesario analizar que no solamente deportistas y staff eran parte de esta entidad, sino también personas que no estaban vinculadas a los equipos como el árbitro, por ejemplo. Y, en caso de que el evento sea de carácter amistoso o de competición de una liga, también debíamos considerar la posibilidad de que el partido o evento podría llegar a ser un partido práctica, en cuyo caso el encargado de arbitrar el mismo podía ser alguien del mismo equipo, así como también los deportistas participantes.

La complejidad de esta entidad recaía en los atributos que eran necesarios para su creación. Es decir, se tenía en cuenta que la persona podía existir o no dentro del sistema, por lo que cada persona, además de tener su número y su tipo de documento, tenía su código único, lo que nos llevaba a tener que identificar si ésta no existía que se genere un nuevo código de persona.

También teníamos que considerar que esta persona podía ser un o una deportista o no, como por ejemplo un empleado de la institución, alguien que forma parte del staff técnico o, por el contrario, si es alguien ajeno a la institución. Por lo que se tuvo que estimar el código de deportista o el código de profesión de esta persona.

Otro atributo fundamental era el código del partido al que se debía vincular la persona que se involucraría en el mismo, así como también resultarían fundamentales atributos como lo son el código del puesto que ocupase una persona en caso de ser un deportista, es decir, la posición dentro del campo de juego, el número de camiseta, su nombre y apellido, el código de la entidad deportiva a la que perteneciera, local o adversaria, y el atributo “Observaciones” en caso de ser necesario resaltar alguna observación sobre esta persona involucrada dentro del partido.

En este contexto, como con la entidad “PosicionCampoJuego”, se continuó con la misma lógica estructural de archivos. Generé doce archivos y edité el archivo “aplicacion-gestion-comercial.js”.

Esta tarea llevó varias semanas, por todas las consideraciones mencionadas anteriormente, razón por la cual fue necesario revisar las entidades ya generadas, cómo se habían implementado, leer el código escrito por otro programador o programadora, repasar la documentación brindada por la empresa para que la entidad pudiera mostrar a las personas involucradas en un determinado partido.

Una vez concluida esta tarea, fue necesario realizar ajustes sobre la entidad “Partido” para poder visualizar sobre ella a estas personas involucradas, así como también su creación y/o edición y eliminación en caso de existir algún error al momento de la carga.

Otra modificación que realizamos sobre esta entidad fue, en la parte administrativa, poder filtrar los partidos por su estado, es decir, si el mismo se encontraba suspendido o si ya se había jugado, por ejemplo. Así como también al momento de editar o crear un nuevo partido, poder ver la lista de jugadores del equipo local y su staff técnico y los jugadores del equipo visitante junto a su staff.

Finalmente, unificamos las ramas con mi compañero de lo hecho por separado y testeamos que las entidades sobre las cuales trabajamos se ejecutaran correctamente para su posterior publicación.

Concluida la etapa de testeo pudimos coordinar una reunión con nuestro supervisor de prácticas para presentar lo trabajado. En la misma se verificó que las entidades y el sistema se estuvieran ejecutando adecuadamente para poder publicarlo.

3.2. Conclusiones

3.2.1. Aprendizaje obtenido de la realización de la práctica

El aprendizaje fue contínuo, lo que me permitió darle dimensión a las metodologías y formas de trabajo que tienen en una empresa.

En lo particular pude implementar el ORM EntityFramework, el motor de base de datos SQL Server y Angular. También pude desenvolverme frente a problemas que surgieron dentro del desarrollo del proyecto y que sin la ayuda de mis compañeros o la de los desarrolladores hubieran resultado difíciles de resolver en el tiempo que disponía en estas prácticas.

El aprendizaje conseguido me llevó a realizar un análisis mucho más detallado del que podemos realizar del dominio del problema en el ámbito académico, así como también la arquitectura computacional que, en lo particular esta empresa, implementa para el desarrollo de software.

La experiencia lograda fue enriquecedora, tanto por el grupo humano de la empresa como por mis compañeros practicantes, quienes fueron de vital importancia a la hora de enfrentar frustraciones que tuvimos durante estas PPS y que tendremos en nuestra carrera profesional.

3.2.2. Comentarios personales del trabajo realizado

En lo personal, me llevo una grata experiencia. Tanto de los desarrolladores con los que pude trabajar, algunos de ellos ya culminaron la tecnicatura en nuestra casa de estudios, como de mis compañeros practicantes con quienes trabajamos codo a codo para que todos podamos concluir estas prácticas.

También quiero resaltar el trato humano de Lucas y de María, ambos tuvieron una cordialidad y atención ante cualquier inquietud que surgiera de estas PPS, a la hora de contactarme por email para poder realizar mis prácticas, en el durante para cuestiones administrativas y para el desarrollo del proyecto, y luego de culminadas para solicitar información relevante a la redacción del presente informe.

Finalmente, el trabajo que hice cumplió con las expectativas que tenía de estas PPS. El proyecto de gestión deportiva en el que la empresa estaba trabajando me permitió conocer nuevas tecnologías y son la punta del hilo para poder implementarlas en futuros proyectos personales.

Considero de especial importancia que sin la colaboración y el trabajo en equipo los resultados esperados en el desarrollo del proyecto asignado hubiesen sido muy difíciles de conseguir. Entiendo y creo firmemente que el trabajo en equipo es fundamental para poder concretar resultados a la altura de las circunstancias, no solo en esta empresa, sino en toda nuestra carrera profesional.

3.2.3. Conclusiones generales

En conclusión, el recorrido de estos casi seis meses de PPS, me permitió conocer un sistema informático a nivel empresarial que a nivel académico no llegaba a dimensionar en su totalidad.

La experiencia fue enriquecedora y me sirvió para entender que, si bien lo tenía presente, en nuestra profesión aprendemos nuevas tecnologías todo el tiempo e incorporamos nuevas metodologías de trabajo que nos permiten crecer en nuestras relaciones laborales, profesionales y personales.

3.3. Recomendaciones

Quiero agradecer a todo el cuerpo académico de nuestra casa de estudios por lo aprendido a lo largo de esta tecnicatura, particularmente a Ernesto Zapata Icart, quien supervisó estas PPS y que sin su ayuda hubiera sido difícil de finalizar, tanto en el desarrollo de estas como en el cursado de distintas materias, por otro lado quiero hacer una mención especial a los contenidos teórico-prácticos que impartió en las mismas, siendo muy pedagógico a la hora de dictar cada clase y plasmando su experiencia profesional en las mismas.

3.3.1. Aportes

- **Zapata Icart**, Ernesto Andrés. Profesor supervisor de las prácticas
- **Grandi**, Lucas. Ingeniero en sistemas Socio fundador de Grandi y Asociados - Ertic SRL
- **Gabas**, María de los Ángeles. Lic. en Sistemas de Información Grandi y Asociados - Ertic SRL

Firmado y visado del presente Informe

Es obligatorio que el informe esté firmado por el autor y contenga además el visado del responsable técnico y el responsable de la administración donde se desarrolló la práctica profesional.