

UNIVERSIDAD TECNOLÓGICA NACIONAL  
FACULTAD REGIONAL PARANÁ

# Informe Práctica Profesional

TUP – UTN FRP

Alumno: **Ulrich, Maximiliano Alejandro**

Organización: **Grandi y Asociados ERTIC SRL.**

Docente: **Ernesto Zapata Icart**

Fecha: **30/04/2024**

## Resumen ejecutivo

Este informe documenta la Práctica Profesional Supervisada llevada a cabo en la empresa Ertic S.R.L. La práctica se realizó durante el período comprendido entre octubre de 2023 y marzo de 2024, en la que participe junto a un grupo de otros cuatro pasantes.

La práctica se dividió en tres etapas: la capacitación inicial, la asignación de tareas grupales y la resolución de tareas individuales. Durante este tiempo, los desarrollos se llevaron adelante en dos proyectos principales y relacionados entre si: el proyecto "Capas", que engloba la lógica de negocios para todos los proyectos de la empresa, y el proyecto web gestión comercial.

Nuestra tarea principal fue la creación del módulo de aforadores para estaciones de servicios. Este módulo incluye un conjunto de ABM (Alta, Baja y Modificación) para las entidades de Tanque, Manguera, Isla y Varillaje. Además de las funciones básicas de un ABM, se implementaron filtros avanzados, como coincidencias de palabras y rangos de fecha, así como opciones de consulta a través de reportes en formato XLS y PDF.

## Índice

<b>RESUMEN EJECUTIVO .....</b>	<b>2</b>
<b>INTRODUCCIÓN.....</b>	<b>4</b>
<b>I – ASPECTOS GENERALES.....</b>	<b>5</b>
1.1 ANTECEDENTES .....	5
1.2 OBJETIVOS .....	5
1.2.1 General:.....	5
1.2.2 Específicos:.....	6
1.3 DELIMITACIÓN DE LA PRÁCTICA PROFESIONAL .....	6
1.4 LIMITACIONES: .....	7
<b>II – EVALUACIÓN INSTITUCIONAL .....</b>	<b>8</b>
2.1 DESCRIPCIÓN GENERAL DE LA INSTITUCIÓN (EMPRESA) .....	8
2.2 MISIÓN.....	8
2.3 VISIÓN .....	9
2.4 VALORES DE LA INSTITUCIÓN .....	9
2.5 ACTIVIDAD INSTITUCIONAL (EMPRESA) .....	10
2.6 DESCRIPCIÓN DEL GRUPO HUMANO (CON EL CUÁL EL ALUMNO TRABAJÓ).....	10
2.8 ESTRUCTURA ORGANIZACIONAL .....	11
<b>III – DESARROLLO DE LA PRÁCTICA PROFESIONAL .....</b>	<b>12</b>
3.1 DESCRIPCIÓN DE ACTIVIDADES DESARROLLADAS EN LA PRÁCTICA PROFESIONAL .....	12
<i>Primera etapa: Período de capacitación teórico-práctica.....</i>	<i>12</i>
<i>Segunda etapa: Asignación de peticiones al equipo para el desarrollo del módulo de aforadores para Estaciones de Servicio:.....</i>	<i>14</i>
<i>Tercera etapa: Asignación de peticiones individuales de Reporteria para el módulo de Hotelería: .....</i>	<i>34</i>
3.2 CONCLUSIONES .....	37
3.2.1 Aprendizaje obtenido de la realización de las prácticas .....	37
3.2.2 Comentarios personales del trabajo realizado .....	38
3.2.3 Conclusiones generales .....	40
3.3 RECOMENDACIONES Y APORTES.....	42
<b>ANEXOS.....</b>	<b>43</b>

## Introducción

En el dinámico mundo de la tecnología, las oportunidades profesionales a menudo implican una adaptación constante y el desarrollo de nuevas habilidades. En octubre de 2023, tuve el privilegio ingresar como pasante a Grandi y Asociados, una destacada empresa argentina con una amplia trayectoria en el mercado y una sólida base de clientes en el país y América Latina. Mi participación en esta empresa como pasante me brindó una experiencia apasionante en el desarrollo de software.

Como pasante, me sumergí en el desarrollo de nuevas funcionalidades para el software comercializado por la empresa, las cuales formarán parte de futuras ofertas comerciales. Nuestra experiencia comenzó con un período de capacitación que se extendió por poco más de un mes y medio. Posteriormente, nos centramos en el desarrollo del módulo de aforadores para estaciones de servicio, el cual se integra con el resto de la aplicación. Finalmente, nos dedicamos a tareas individuales de reportería para el módulo de hotelería.

En este informe, compartiré mi experiencia, los desafíos enfrentados, los logros alcanzados y cómo mi formación académica en la UTN ha contribuido al éxito de los desafíos que surgieron durante la pasantía. En particular, destacaré el desarrollo de una pieza de software específica: el Módulo aforadores para Estaciones de Servicio. Esta tarea no solo requirió la mayor cantidad de horas de trabajo, sino que también fue la más enriquecedora en términos de conocimientos aplicados y trabajo en equipo.

## I – Aspectos generales

### 1.1 Antecedentes

Mi nombre es Maximiliano Alejandro Ulrich, nacido en la ciudad de Paraná. A lo largo de mi vida, he tenido la oportunidad de residir en diversas partes del país por motivos laborales, hasta establecerme nuevamente en Paraná en diciembre de 2019.

Mi primer contacto con el mundo de la programación se remonta a mi primer año de colegio secundario, allá por el año 2002. Fue gracias a una profesora de informática que pude adentrarme en este fascinante campo. Durante ese tiempo, realizamos pequeños programas de consola en el lenguaje Pascal, el cual, al ser fuertemente tipado, constituye una herramienta valiosa para iniciarse en la programación.

Después de finalizar mis estudios secundarios, decidí iniciar la carrera de Contador Público Nacional. Sin embargo, mi interés por la programación nunca desapareció. No fue hasta finales de 2020 que, gracias al crecimiento en la demanda de programadores y por la recomendación de un amigo que había iniciado sus estudios en la UTN, decidí inscribirme en la Tecnicatura Universitaria en Programación de la Universidad Tecnológica Nacional. Finalicé el cursado de la misma en noviembre de 2022.

### 1.2 Objetivos

#### 1.2.1 General:

Ampliar mis conocimientos de programación con el objetivo de convertirme en un desarrollador de software competente y poder ingresar al mercado laboral, ya sea como profesional independiente o como parte de una empresa del sector.

### 1.2.2 Específicos:

1. Diseñar y desarrollar el módulo de Estaciones de Servicio con funcionalidades básicas para el alta de las distintas entidades, así como funcionalidades avanzadas de filtrado y generación de informes.
2. Garantizar la integración adecuada del módulo con el resto de la aplicación, asegurando su funcionamiento sin problemas y su compatibilidad con otras partes del sistema.
3. Realizar pruebas exhaustivas del módulo de Estaciones de Servicio para identificar y corregir posibles errores o fallos de funcionamiento, asegurando su fiabilidad y calidad.
4. Investigar y aplicar buenas prácticas de programación y diseño de software para garantizar la escalabilidad, mantenibilidad y eficiencia del módulo a largo plazo.
5. Colaborar activamente con otros desarrolladores y profesionales del equipo para intercambiar conocimientos, resolver problemas y optimizar el desarrollo del proyecto en su conjunto.

### 1.3 Delimitación de la práctica profesional

Alcance temporal: Esta práctica profesional se encuentra limitada a la duración establecida en el contrato con la empresa, el cual abarca el periodo desde el 17 de Octubre de 2023 al 17 de Marzo de 2024.

Alcance funcional: El desarrollo de la práctica profesional divide en tres momentos:

- El periodo de capacitación: donde se abordaron una serie de contenidos de carácter teórico-prácticos estructurados en 6 temas: Introducción, conceptos generales y

herramientas a utilizar, .NET, Programación Orientada a Objeto General y NETCORE, Front End & Diseño web, Flutter, SQL SERVER – BDD y finalmente Reporteria.

- Desarrollo del módulo de Estaciones de Servicio: Con un ABM para la creación de cada una de sus entidades, funcionalidades de filtrado y reporteria.
- Diseño y creación de reportes: Para el módulo de hotelería, del reporte en formato PFD, de arribos y partidas previstas para el día posterior al de la consulta, incluyendo los cálculos de totales a facturar para las partidas.

Alcance tecnológico: Si bien en el período de capacitación se abordó una amplia lista de tecnologías, para el desarrollo del módulo de estaciones y reporteria se utilizó un stack más acotado, incluyendo C#, .NET Framework, AngularJS, HTML, Razor, SQL Server, Report Viewer, etc.

Alcance de aplicación: El módulo de Estaciones de Servicio está pensado para ser una solución que se integra a los módulos de gestión y forman parte de la oferta comercial de la empresa para gestión de Estaciones de Servicio.

Alcance Organizacional: La Práctica Profesional se llevó a cabo dentro de la estructura organizativa de Grandi y Asociados Ertic SRL.

#### 1.4 Limitaciones:

- El horario establecido es de 14 a 18Hs, de lunes a viernes.
- Las horas de trabajo son bajo la modalidad de trabajo remota y se deben registrar las tareas realizadas y el tiempo insumido a través de la aplicación Redmine, cuyo usuario y password son provistos por la empresa.

- Los desarrollos se deben realizar con las tecnologías que ya utiliza la empresa y respetando la arquitectura y patrones de diseño de la aplicación.

## **II – Evaluación Institucional**

### **2.1 Descripción general de la Institución (Empresa)**

Grandi y Asociados Ertic S.R.L. es una empresa de software argentina situada en la ciudad de Paraná en la calle Pablo Lorentz 2977, posee una trayectoria de más de 35 años en el rubro de desarrollo de software, brindando soluciones para empresas, pymes y comercios.

Ofrece soluciones de software que son íntegramente desarrolladas por el personal de la empresa, asegurando a sus clientes continuidad y soporte post venta, con gran capacidad de adaptabilidad para mejoras o ampliación de funcionalidades.

Posee un equipo de desarrolladores y diseñadores gráficos que posibilitan el desarrollo de aplicaciones estéticamente atractivas y técnicamente operativas<sup>1</sup>.

### **2.2 Misión**

Poner soluciones tecnológicas al alcance de nuestros clientes a un costo accesible y alta calidad. Tenemos como misión especial exceder constantemente, con nuestros productos y servicios, todas las expectativas de nuestros clientes, satisfaciendo sus necesidades y ayudándolos en el logro de sus metas y objetivos. Brindar a nuestros clientes soluciones integrales, que abarquen los procesos de negocios dentro de su empresa, ayudando a lograr las diferencias competitivas que les permitan perpetuarse y

---

<sup>1</sup> Fuente: Sitio Web Grandi y Asociados. - ¿Quiénes Somos? –  
Sitio Web: <https://www.grandiyasociados.com/quienes-somos/>

crecer. Esto lo hacemos con personal calificado, satisfecho y comprometido con la empresa. Ofreciendo a nuestros socios las utilidades esperadas y proporcionar a los empleados posibilidades de desarrollo que les permita alcanzar crecimiento personal y profesional<sup>2</sup>.

### 2.3 Visión

Ser la empresa de desarrollo de Software personalizado, para la pequeña y mediana empresa, de más éxito y con los mejores estándares de calidad en el mercado; ayudando con esto a que nuestros clientes sean más rentables y competitivos<sup>3</sup>.

### 2.4 Valores de la Institución

- Integridad: Ser honestos, justos y coherentes en todas nuestras acciones.
- Responsabilidad: Asumir compromisos y cumplirlos.
- Compromiso: Esfuerzo continuo para elevar la calidad de nuestro trabajo.
- Espíritu de Equipo: Conservar un ambiente de trabajo confortable que inspire el trabajo en equipo y fomente las relaciones humanas.
- Respeto Mutuo: Mantener el respeto y cordialidad tanto entre los integrantes de la empresa como con los clientes y asociados.
- Superación Continua: Permanecer renovando nuestras habilidades profesionales.
- Vocación de Servicio: Predecir y exceder, oportunamente, las expectativas de nuestros clientes.<sup>4</sup>

---

<sup>2</sup> Extraído: Sitio Web Grandi y Asociados. – *Misión Grandi y Asociados* – Sitio Web: <https://www.grandiyasociados.com/quienes-somos/>

<sup>3</sup> Extraído: Sitio Web Grandi y Asociados. – *Visión Grandi y Asociados* – Sitio Web: <https://www.grandiyasociados.com/quienes-somos/>

<sup>4</sup> Extraído: Sitio Web Grandi y Asociados. – *Nuestros Valores* – Sitio Web: <https://www.grandiyasociados.com/quienes-somos/>

## 2.5 Actividad Institucional (Empresa)

Grandi y asociados, en su página institucional, se define de la siguiente manera: “Somos una Empresa Desarrolladora de Software e Ingeniería Web en Argentina, con más de 35 años de trayectoria brindando soluciones informáticas integrales en la república y países Latinoamericanos. La experiencia adquirida al haber trabajado en diversidad de rubros sumada al equipo de profesionales de la informática y el diseño web que formamos parte, son los que avalan y garantizan nuestros trabajos, brindando la seguridad y confianza que usted necesita.

Los sistemas de gestión que ofrecemos son desarrollados íntegramente por personal de la empresa, lo que asegura la continuidad y soporte post venta. Son sistemas abiertos y configurables, lo que permite una amplia adaptación.

El desarrollo de sitios web está a cargo de profesionales de la informática y diseñadores gráficos, los que en forma conjunta trabajan para lograr sitios web estéticamente atractivos y técnicamente operativos, conjunción fundamental a tener en cuenta al momento de evaluar el diseño.<sup>5</sup>”

## 2.6 Descripción del grupo humano (con el cuál el alumno trabajó)

La empresa se conforma actualmente por 4 Áreas que dependen de la Gerencia General, El desarrollo de mi práctica profesional se llevó a cabo como integrante de un grupo de 5 pasantes (todos alumnos de la tecnicatura universitaria en programación) y nuestro contacto para coordinar actividades propias de la práctica era con el Gerente de desarrollo y la coordinadora de TI y Desarrollo de Software. En cuanto a consultas técnicas

---

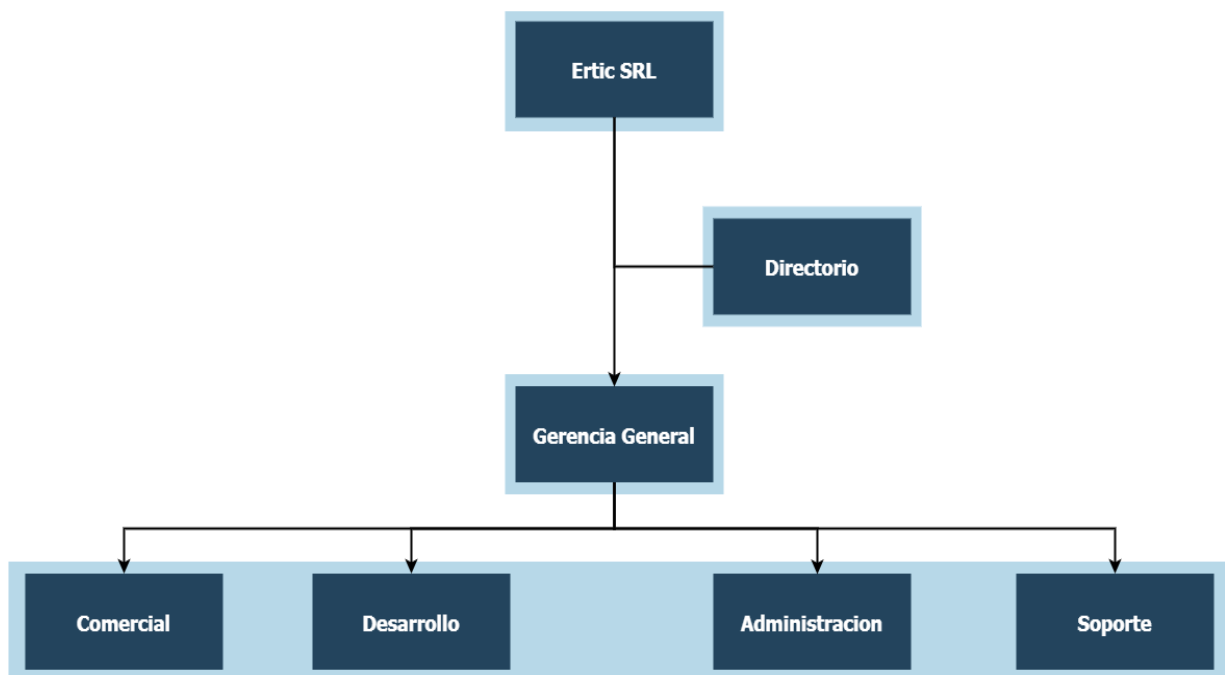
<sup>5</sup> Extraído: Sitio Web Grandi y Asociados. – ¿Quiénes somos? – Sitio Web:  
<https://www.grandiyasociados.com/quienes-somos/>

sobre las peticiones a cumplimentar, nuestro contacto era con el grupo de desarrolladores de la empresa.

## 2.7 Descripción del escenario de trabajo

Como pasantes dependíamos del Área de Desarrollo, la cual trabaja en forma coordinada con las otras tres Áreas y dependen de la Gerencia General. Diariamente nos conectábamos utilizando la plataforma Discord, al canal creado por la Empresa, donde a su vez existen distintos sub-canales, entre los cuales estaba el de Entrenamiento que utilizábamos para reunirnos y coordinar las distintas tareas que se nos asignaban.

## 2.8 Estructura Organizacional



### III – Desarrollo de la Práctica Profesional

#### 3.1 Descripción de Actividades desarrolladas en la práctica Profesional

##### Primera etapa: Período de capacitación teórico-práctica

A fin de nivelar los conocimientos del grupo de pasantes, la primer tarea que me fue asignada, fue estudiar y repasar los puntos contenidos en un temario provisto por la empresa, los cuales resulta necesario tener noción para luego abordar peticiones específicas sobre el software de la empresa. El mencionado temario que recorría los siguientes tópicos:

- Herramientas de desarrollo y de documentación: SQL Server, Visual Studio 2015/17, GIT (Source tree), Redmine, VScode (Flutter y Angular), Figma - Prototipado APP y WEB y Android STUDIO.
- Arquitectura de las soluciones: Full web, Cliente servidor, Saas, Paas, IaaS , multi-tenant (una app, N clientes), CrossBrowsers, Responsive, Componentes en Servidores Windows para despliegue de Soluciones NET y JAVA (**IIS, Wildfly, Tomcat**)
- Aplicaciones Funcionales: Entidades, Tablas maestras y de movimientos, Grupos de tablas nomenclatura, Tablas por módulos.
- Funcionalidades comunes: ABM simples y con pestañas, funcionalidades simples, medias y avanzadas, Lanzadores de consultas.
- Funcionalidades resaltadas por módulo: ERP, CRM, Omnicanalidad, Chat Boot-Inteligencia Artificial(Chat Boot conceptos, DialogFlow by Google).
- Testing: TDD (Test Driven Development), NUnit., Fakes, Mocks. Moq.
- Temas comunes a NET: .NET Framework, CLR (Common Language Runtime),

DLL (Dynamic Link Libraries), MSIL (Microsoft Intermediate Language), ¿Cómo maneja la RAM .NET?, pila de la memoria, Heap dinámico de .NET, Recolector de basura, Código manejado y no manejado, lenguajes soportados por .NET. (C#, VB.NET, C++) F#, etc. (Principalmente C#), Introducción MVC y Web Api, entre otros.

- Lógica de negocios: Dentro de esta se hizo un recorrido por temas básicos y avanzados en el uso de C# que incluía temas tratados durante el desarrollo de las materias de Programación I y II, así como otras que nunca habíamos abordado, entre estas, SignalR, Manejo de paquetes NuGet, Owin, Entity Framework, Patrón Unit of Work, Concepto de Persistence Ignorance, etc.
- LINQ. Lenguaje integrado de consultas: LINQ y expresiones lambda, Clase Enumerable, Interfaz IQueryable., Clase Queryable, Clase Expression y Árboles de expresión.
- Reflection e inyección de dependencias.
- 1. Front End & Diseño web: Document Object Model (DOM), JavaScript, Bootstrap, Require JS, HTML 5, Razor, vistas enlazadas, CSS y LESS, AngularJs (Controllers, Module, Application, Filter, Factory, Directives, ng-model, ng-change, ng-repeat, ng-class, ng-bind, ng- etc, \$state, \$watch, \$location, Bindings, Values, Router, state.
- Web Api: Controladores, Metodos (GET, POST, DELETE), Protocolo http y https en .NET, Llamadas async, Invocación de Web Services (WS), Manejo de Excepciones, Manejo de parámetros de Web Services, Armado de DTOs.
- Flutter: Dentro de este apartado se abordaron cuestiones básicas de la documentación oficial, configuración del editor, creación de una APP básica,

Layouts básicos, diversos tutoriales, widgets, Navegación y enrutado, Modelo MVVM para el desarrollo vistas (IS), Providers (IS) y Conexión con Back-End (IS).

- Reporteria en Report Viewer: Parámetros, Manejo de Dataset dentro del reporte, Cortes de control (tablas) (en profundidad), Encabezado, Pie de página, Imágenes, Gráficos y Totales.
- Aplicaciones de Negocios: Dentro de este apartado se recorrieron ejemplos soluciones específicas de los distintos productos que integran la oferta comercial de la empresa.

Cabe destacar que para facilitar el abordado de la guía, fui provisto acceso a una carpeta de Google Drive en la que contenía gran cantidad de videos de clases grabadas elaboradas por el personal de la empresa, y esto a su vez fue complementado con clases en tiempo real a través de la plataforma Discord, donde tuve la posibilidad de evacuar dudas y consultas.

### Segunda etapa: Asignación de peticiones al equipo para el desarrollo del módulo de aforadores para Estaciones de Servicio:

Antes de abordar el desarrollo de esta etapa, es importante conocer brevemente las características del stack tecnológico utilizado para este desarrollo puntual, las paso a enumerar:

- C#: Es un lenguaje de programación desarrollado por Microsoft, potente y versátil, de propósito general y orientado a objetos que se utiliza ampliamente en una variedad de industrias y aplicaciones, gracias a su combinación de simplicidad,

potencia y soporte para el ecosistema .NET de Microsoft.

- JavaScript: JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, ampliamente utilizado en el desarrollo web. Aunque inicialmente fue diseñado para agregar interactividad a las páginas web, JavaScript ha evolucionado para convertirse en un lenguaje de programación completo que se ejecuta en diversos entornos, incluyendo navegadores web, servidores y dispositivos móviles, convirtiéndose en un componente esencial en el desarrollo de aplicaciones web dinámicas y de una sola página (SPA).

Una de las características distintivas de JavaScript es su capacidad para manipular el DOM (Document Object Model) de una página web de forma dinámica, lo que permite la creación de experiencias interactivas y ricas en contenido. Además, JavaScript es un lenguaje de scripting flexible que permite a los desarrolladores crear funciones y aplicaciones complejas con relativa facilidad.

- AngularJS: (a menudo llamado Angular.js o AngularJS 1) es una librería de JavaScript de código abierto mantenida por Google, diseñada para facilitar la creación y mantenimiento de aplicaciones web de página única (SPA). Su objetivo principal es mejorar las aplicaciones basadas en navegador mediante el patrón Modelo-Vista-Controlador (MVC), lo que simplifica el desarrollo y la prueba de las mismas. En nuestra aplicación, AngularJS ofrece una sólida base para el desarrollo de una interfaz dinámica e interactiva. Gracias a su arquitectura basada en componentes y su sistema de enlace de datos bidireccional, AngularJS simplifica la manipulación del DOM (Document Object Model) y la gestión del estado de la aplicación. Esto se logra a través de la implementación de directivas, controladores y servicios, que permiten una comunicación fluida entre el código JavaScript y el

HTML, lo que resulta en una experiencia de usuario más rica y receptiva.

- .NET Framework: .NET Framework es un entorno de desarrollo y tiempo de ejecución creado por Microsoft. Proporciona una plataforma integral para construir y ejecutar una variedad de aplicaciones, desde aplicaciones de escritorio hasta servicios web y aplicaciones móviles. .NET Framework nos brinda una infraestructura robusta y segura para la lógica de negocio de nuestra aplicación, facilitando la comunicación con la base de datos y garantizando un alto nivel de confiabilidad y eficiencia.
- Razor: Es un motor de plantillas utilizado en el desarrollo web con ASP.NET, especialmente en el contexto de ASP.NET Core. Permite a los desarrolladores combinar código C# con HTML para crear vistas dinámicas de manera sencilla y eficiente. Utiliza una sintaxis basada en signos de "@" para delimitar el código C# dentro del HTML, lo que facilita la transición entre el código de servidor y el código de cliente.
- SQL Server: SQL Server es un sistema de gestión de bases de datos relacional (RDBMS) desarrollado por Microsoft. Es una plataforma completa para el almacenamiento, administración y análisis de datos estructurados y relacionales. SQL Server es ampliamente utilizado en entornos empresariales y corporativos debido a su robustez, escalabilidad y capacidad para manejar grandes volúmenes de datos.

Una de las características distintivas de SQL Server es su lenguaje de consulta, Transact-SQL (T-SQL), que permite a los usuarios interactuar con la base de datos para realizar consultas, insertar, actualizar y eliminar datos, y administrar la

seguridad y la integridad de los datos.

### **Petición #1921:**

Durante la segunda semana de enero, en una reunión con el Gerente de Desarrollo, se nos encargó al grupo la tarea de desarrollar el módulo de aforadores para estaciones de servicio. Este módulo debía ser habilitado en el menú lateral de la aplicación y constar de cuatro ABM correspondientes a las cinco entidades principales: Tanque, Surtidor, Manguera, Isla y Modelos de Surtidores.

Para iniciar esta petición, nuestro primer desafío fue estudiar la estructura de la aplicación existente. Identificamos los archivos que debíamos crear y modificar, los métodos que podríamos reutilizar y el esquema de permisos necesario para visualizar los nuevos componentes en la aplicación.

Una vez completada esta fase inicial y con los pasos a seguir definidos, nos dividimos en grupos de dos personas para comenzar a escribir el código necesario para cada ABM. A continuación, detallaré el proceso de desarrollo para una mejor comprensión:

### **En el Proyecto Capas:**

1. Creación de las clases principales: como primer paso creé los archivos Tanque.cs, Surtidor.cs, ModelosSurtidor.cs, Isla.cs y Manguera.cs, que corresponden a las entidades principales del sistema. Estos archivos se ubicaron en la lógica de negocios del proyecto, dentro de una nueva carpeta identificada como "Estaciones".

Cada archivo constituye una clase principal que contiene las propiedades y métodos necesarios para la creación y manipulación de objetos de dicha clase. Estas clases servirán como la base fundamental para el desarrollo del módulo de aforadores de estaciones de servicio.

```
public class Isla : IEntidadAuditable
{
    private const string CODIGO_CLASE_ENTIDAD = "ISLAS_____";

    #region Propiedades Públicas
    7 referencias
    public string CodigoIsla { get; set; }
    4 referencias
    public string Descripcion { get; set; }
    5 referencias
    public string codsuc { get; set; }
    1 referencia
    public virtual SucursalEmpresa Sucursal { get; set; }
    4 referencias
    public string observaciones { get; set; }
    #endregion

    #region Métodos internal
    1 referencia
    internal void Alta(IslaNuevoDTO dto, IUnitOfWork uow)
    {
        uow.RegisterNew<Isla>(this);
        this.setPropiedadesNuevo(dto, uow);
    }

    1 referencia
    internal void Modificar(IslaModificarDTO dto, IUnitOfWork uow)
    {
        this.setPropiedadesModificar(dto, uow);
    }

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

*Ilustración 1 - Propiedades y métodos de Isla.cs*

2. Creación de los archivos TanquesAdmin.cs, SurtidoresAdmin.cs, ModelosSurtidoresAdmin.cs, IslasAdmin.cs y ManguerasAdmin.cs. Estos archivos se encuentran también dentro de la carpeta Estaciones en la lógica de negocios.

Estos archivos contienen los métodos encargados de realizar las operaciones de creación, modificación y eliminación de registros de la entidad en la base de datos. Estos métodos llaman a los correspondientes métodos de las clases principales creadas anteriormente.

Es importante destacar que los métodos para modificar registros de estas clases

reciben como parámetro un objeto de tipo DTO (Data Transfer Object), el cual comparte varios de los atributos del objeto principal.

```
0 referencias
public Isla AltaIsla(IslaNuevoDTO dto)
{
    Isla isla = _uow.CreateNew<Isla>();
    isla.Alta(dto, _uow);
    _uow.Save();
    return isla;
}

0 referencias
public Isla ModificarIsla(IslaModificarDTO dto)
{
    Isla isla = _uow.TryEnsureRegistered<Isla>(dto.Codigoisla);
    if (isla == null)
        throw new ExceptionLn(string.Format(Mensajes.IslaInexistente, dto.Codigoisla));
    isla.Modificar(dto, _uow);
    _uow.Save();
    return isla;
}

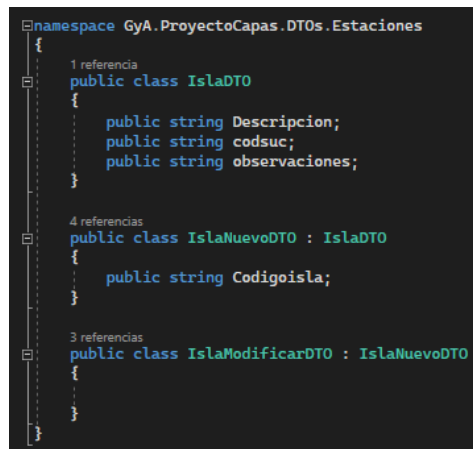
[AccionAuditableSobreEntidad(Accion.Codigos.ELIMINAR)]
0 referencias
public void EliminarIsla(string codigo)
{
    if (string.IsNullOrEmpty(codigo))
        throw new ExceptionLn(Mensajes.IslaRequiereCodigo);
    Isla isla = _uow.TryEnsureRegistered<Isla>(codigo);
    if (isla == null)
        throw new ExceptionLn(string.Format(Mensajes.IslaInexistente, codigo));
    isla.Eliminar(_uow);
    try
    {
        _uow.Save();
    }
    catch (Exception e)
    {
        throw new ExceptionLn(string.Format(Mensajes.EntidadesRelacionadasAlEliminar));
    }
}
```

*Ilustración 2 - Métodos de IslasAdmin.cs*

**3. Creación de los Data Transfer Objects (DTOs):** Para facilitar la transferencia de datos entre las capas de la aplicación, se procedió a la creación de los archivos TanquesDTO.cs, SurtidoresDTO.cs, ModelosSurtidorDTO.cs, IslasDTO.cs y ManguerasDTO.cs. Estos archivos se alojan en una carpeta destinada específicamente para los DTOs, similar a la organización utilizada para los archivos de administración de entidades.

Estas clases replican la estructura de las entidades principales, pero contienen solo un subconjunto sus propiedades. Estos DTOs se utilizan en los archivos de administración mencionados anteriormente para pasar como parámetro a los métodos. De esta manera, los métodos de administración pueden interactuar con las tablas de la base de datos

correspondientes a las entidades de manera eficiente y coherente.



```
namespace GyA.ProyectoCapas.DTOs.Estaciones
{
    1 referencia
    public class IslaDTO
    {
        public string Descripcion;
        public string codsuc;
        public string observaciones;
    }

    4 referencias
    public class IslaNuevoDTO : IslaDTO
    {
        public string Codigoisla;
    }

    3 referencias
    public class IslaModificarDTO : IslaNuevoDTO
    {
    }
}
```

*Ilustración 3 - IslasDTO.cs*

4. Creación de los archivos Map: Se crean los archivos TanqueMap.cs, SurtidorMap.cs, ModeloSurtidorMap.cs, IslasMap.cs y MangueraMap.cs, dentro de la carpeta de Mapeos. Estos archivos son utilizados por Entity Framework (ORM) para establecer las relaciones entre las clases principales del proyecto y las tablas previamente creadas en la base de datos.

En estos archivos, es fundamental establecer no solo una concordancia entre el nombre de los atributos de las clases y los nombres de las columnas de la base de datos, sino también indicar las claves primarias, las claves foráneas y el tipo de relación con otras tablas.

Durante el proceso de desarrollo, surgió un desafío relacionado con los errores al intentar hacer el mapeo a las tablas de la base de datos. Este problema se resolvió al respetar el orden del mapeo de las propiedades en los archivos Map.

Además, la configuración de las relaciones entre las entidades requirió una comprensión más profunda del problema, así como el uso de los métodos provistos por

Entity Framework para definir correctamente estas relaciones.

```
class IslaMap : EntityTypeConfiguration<Isla>
{
    1 referencia
    public IslaMap(string nombreEsquema)
    {
        // Primary Key
        this.HasKey(t => t.Codigoisla);

        // Properties
        this.Property(t => t.Codigoisla)
            .IsRequired()
            .HasMaxLength(10);

        this.Property(t => t.Descripcion)
            .IsRequired()
            .HasMaxLength(100);

        this.Property(t => t.codsuc)
            .HasMaxLength(10);

        this.Property(t => t.observaciones)
            .HasMaxLength(100);

        // Table & Column Mappings
        this.ToTable("E_islas", nombreEsquema);
        this.Property(t => t.Codigoisla).HasColumnName("Codigoisla");
        this.Property(t => t.Descripcion).HasColumnName("Descripcion");
        this.Property(t => t.codsuc).HasColumnName("codsuc");
        this.Property(t => t.observaciones).HasColumnName("observaciones");

        //Relationships
        this.HasOptional(t => t.Sucursal)
            .WithMany()
            .HasForeignKey(t => t.codsuc);
    }
}
```

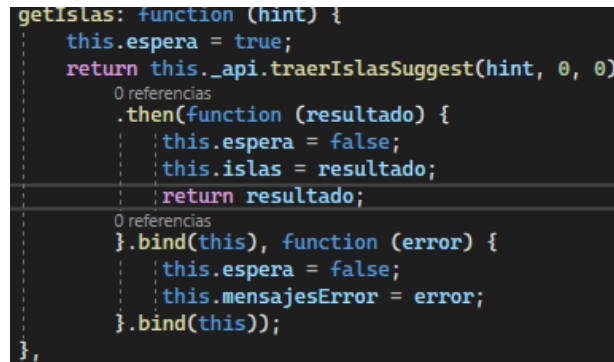
*Ilustración 4 - IslaMap.cs*

5. Creación de los View Model para la vista principal: se crean los archivos Tanque-administrar-vm.js, Surtidor-administrar-vm.js, ModeloSurtidor-administrar-vm.js, Isla-administrar-vm.js y Manguera-administrar-vm.js. Estos archivos están enlazados con el proyecto de gestión comercial y se utilizan para declarar los View Models que manejan la lógica de la vista principal de cada entidad.

Cada archivo define un View Model que contiene el código JavaScript necesario para interactuar con la API y realizar las operaciones requeridas por la vista principal. Por lo general, estos métodos incluyen llamadas a la API para obtener los datos de la entidad que la vista necesita mostrar, como por ejemplo el método `getIslas`, que devuelve un listado de islas con los atributos necesarios para mostrar en una lista. También pueden incluir métodos para navegar a otras vistas, como el método `irAltaIsla`, que activa la vista para

agregar o editar una isla (Isla-nuevo-editar-vm.js).

Es importante destacar que cada uno de estos archivos está relacionado con una vista específica, por ejemplo, AdministrarIsla.cshtml, lo que facilita la organización y mantenimiento del código.



```
getIslas: function (hint) {  
    this.espera = true;  
    return this._api.traerIslasSuggest(hint, 0, 0)  
        .then(function (resultado) {  
            this.espera = false;  
            this.islas = resultado;  
            return resultado;  
        })  
        .bind(this, function (error) {  
            this.espera = false;  
            this.mensajesError = error;  
        }).bind(this));  
},
```

*Ilustración 5 – metodo getIslas*

6. Creación de los View Model de la vista para agregar o modificar: Se crean los archivos Tanque-nuevo-editar-vm.js, Surtidor-nuevo-editar-vm.js, ModeloSurtidor-nuevo-editar-vm.js, Isla-nuevo-editar-vm.js y Manguera-nuevo-editar-vm.js. Estos archivos tienen la misma función que los View Models para la vista principal, pero están destinados a la vista secundaria utilizada para el alta o modificación de registros de cada entidad.

Cada archivo define un View Model que contiene el código JavaScript necesario para manejar la lógica de la vista de agregar o modificar registros. La dificultad en esta vista radica en distinguir si se está cargando un nuevo registro o editando un registro existente. Para lograr esto, se utilizan una serie de variables booleanas que indican el estado actual de la vista.

Estos archivos son fundamentales para asegurar la correcta funcionalidad de las vistas de agregar o modificar registros, ya que manejan la interacción con la API y la

presentación de la información correspondiente a cada entidad.

```
self.getDto = function(){
  return {
    Codigoisla: self.isla.Codigoisla,
    Descripcion: self.isla.Descripcion,
    observaciones: self.isla.observaciones,

    codsuc: (self.editandoSucursal && self.sucursal !== null) ?
      self.isla.codsuc :
      (!self.editandoSucursal && !self.nuevo ?
        self.isla.codsuc :
        (self.nuevo && self.sucursal === null) ?
          null :
          (self.editandoSucursal && self.sucursal === null ?
            null :
            self.isla.codsuc)),
  }
}
```

*Ilustración 6 - Ejemplo de método getDto utilizado en Isla-nuevo-editar-vm.js*

7. Creación de los archivos “mod”: Se crean los archivos Tanque-mod.js, Surtidor-mod.js, ModeloSurtidor-mod.js, Isla-mod.js y Manguera-mod.js. Estos archivos tienen la función de controlar los View Models de administrar y el de nuevo-editar para cada entidad del proyecto.

En estos archivos, nos encontramos con diversos métodos útiles para la manipulación de datos y la interacción con las vistas:

- **Métodos ".controller"**: Se utilizan para definir los controladores que gestionan la lógica de las vistas y se comunican con los View Models correspondientes.
- **Métodos ".factory"**: Son utilizados para crear u obtener objetos que serán utilizados en los View Models, facilitando la gestión de la información.
- **Métodos ".filter"**: Se utilizan para mostrar información filtrada en las vistas, como por ejemplo, atributos concatenados de determinados objetos que se muestran en los sugeridos de los inputs.

Estos archivos desempeñan un papel crucial en la organización y funcionamiento del

proyecto, ya que centralizan la lógica de control de las vistas y facilitan la manipulación de los datos en la interfaz de usuario.

```
.filter('isla', function () {
  0 referencias
  return function (islas) {
    var array = [];
    0 referencias
    return islas.then(function (resultado) {
      3 referencias
      angular.forEach(resultado, function (item) {
        array.push({
          objeto: item,
          label: item.Codigoisla + ' - ' + item.Descripcion
        });
      });
      return array;
    });
  };
})
```

*Ilustración 7 -. filter de isla*

8. Creación de las vistas principales: se crearon las vistas principales AdministrarTanque.cshtml, AdministrarSurtidor.cshtml, AdministrarModeloSurtidor.cshtml, AdministrarIslas.cshtml y AdministrarManguera.cshtml. Estas vistas constituyen la vista principal de cada entidad y muestran una lista de elementos cargados, como surtidores, modelos de surtidores, islas o mangueras. Además, proporcionan botones para borrar el registro a la izquierda de la lista.

Para acceder a la vista de nuevo-editar, se debe seleccionar un elemento de la lista para editarlo o hacer clic en el vínculo ubicado en la esquina superior izquierda de la vista para crear un nuevo registro.

Estas vistas están asociadas con un View Model, como se explicó en el punto 5, y con un controlador correspondiente, por ejemplo IslaController.cs, ya que el proyecto sigue una arquitectura de tipo MVC (Modelo-Vista-Controlador).

Una particularidad de estas y el resto de las vistas del sistema es que para el texto de los label, títulos, botones, etc., se utilizan etiquetas que se definen en el controlador y se

almacenan en un archivo de tipo .resx, lo que permite una gestión eficiente del contenido textual de la aplicación.



*Ilustración 8 - Vista principal AdministrarTanque.cshtml*

9. Creación de las vistas secundarias para carga o edición: se crean las vistas secundarias NuevoEditarTanque.cshtml, NuevoEditarSurtidor.cshtml, NuevoEditarModeloSurtidor.cshtml, NuevoEditarIslas.cshtml y NuevoEditarManguera.cshtml. Estas vistas permiten visualizar los campos de carga de datos para dar de alta un nuevo registro u editar uno existente.

Estas vistas presentaron un nivel de dificultad superior a las vistas principales, ya que en muchos casos se necesitaba traer datos de otras entidades, darles tratamiento y mostrarlos en elementos como suggest o comboBox. Para lograr esto, se hizo uso de diferentes web services y métodos ".filter" para mostrar la información de manera adecuada.

Cuando el usuario presiona el botón de guardar, se llama al método correspondiente en el View Model, al cual se le pasan por parámetro los datos del objeto. Este método contiene el código encargado de llamar a los métodos para la creación o modificación del objeto, según sea el caso, y finalmente redireccionar a la vista principal.

Estas vistas desempeñan un papel fundamental en la interacción del usuario con el sistema, permitiendo la inserción y modificación de datos de manera intuitiva y eficiente.

Administrar / Buscar / Nuevo  
AGREGAR TANQUE

Grabar Cancelar

Datos principales Varillajes Lanzador de consultas

Código de tanque 4

Descripción Ingrese la descripción

Observaciones Ingrese observaciones

Código de artículo Ingrese código de artículo

Capacidad en litros Ingrese la capacidad en litros

Id de sistema externo Ingrese Id de sistema externo

Diámetro Ingrese el diámetro

Ángulo de inclinación Ingrese el ángulo de inclinación

Altura Ingrese la altura

Largo Ingrese el largo

Habilitado ☐

*Ilustración 9 - NuevoEditarTanque.cshtml*

### **En el Proyecto Web Gestión Comercial:**

1. Creación de los controladores de la API: Se crean los controladores de la API `TanqueController.cs`, `SurtidorController.cs`, `ModeloSurtidorController.cs`, `IslaController.cs` y `MangueraController.cs`. Estos controladores contienen los servicios web necesarios para gestionar las operaciones de alta, modificación o eliminación de registros relacionados con cada entidad del proyecto.

Cada uno de estos controladores define métodos que reciben como parámetro un objeto creado en el View Model correspondiente. Estos objetos encapsulan los datos necesarios para realizar las operaciones solicitadas, como agregar un nuevo registro, modificar uno existente o eliminarlo.

La creación de estos controladores es fundamental para proporcionar una interfaz de

programación de aplicaciones (API) robusta y eficiente que permita la comunicación entre el cliente y el servidor de la aplicación.

```
[AutorizarApi]
[AutorizarWebApi(CodigoPrograma = Programa.Codigos.ISLA, CodigoAccion = Accion.Codigos.MODIFICAR)]
0 referencias
public void Modificar(IslaModificarDTO nuevosDatos)
{
    _IslaAdmin.ModificarIsla(nuevosDatos);
}
```

*Ilustración 10 - Método Modificar de IslaController.cs*

2. Creación de los controladores para consulta: Se crean los controladores para consulta TanqueConsultasController.cs, SurtidorConsultasController.cs, ModeloSurtidorConsultasController.cs, IslasConsultasController.cs y MangueraConsultasController.cs. Estos archivos contienen servicios web dedicados exclusivamente a la recuperación de información de la base de datos relacionada con cada entidad del proyecto.

Cada uno de estos controladores define métodos que permiten realizar consultas específicas, como la obtención de datos filtrados según ciertos criterios, la recuperación de todos los registros que coincidan con un ID específico o la obtención de la lista completa de registros.

La creación de estos controladores para consulta es esencial para proporcionar una forma estructurada y eficiente de acceder a los datos almacenados en la base de datos, lo que permite a la aplicación recuperar y mostrar la información solicitada de manera rápida y precisa.

```
[AutorizarApi]
[AutorizarWebApi(CodigoPrograma = Programa.Codigos. ISLA, CodigoAccion = Accion.Codigos.VER)]
[HttpGet]
0 referencias
public IslaDatos GetIsla(string codigo)
{
    return _consultasRepositorio.Queryable<Isla>()
        .Where(u => u.Codigoisla == codigo)
        .Select(a => new IslaDatos
        {
            Codigoisla = a.Codigoisla,
            Descripcion = a.Descripcion,
            observaciones = a.observaciones,

            codsuc = a.codsuc,
            sucursalDatos = new SucursalDatos {
                CodigoSucursal = a.Sucursal.CodigoSucursal,
                Nombre = a.Sucursal.Nombre
            },
        })
        .FirstOrDefault();
}
```

*Ilustración 11 - Método GetIsla de IslaConsultasController.cs*

3. Creación de los controladores de la vista: Se crean los controladores de la vista EstacionTanqueController.cs, EstacionSurtidorController.cs, EstacionModeloSurtidorController.cs, EstacionIslasController.cs y EstacionMangueraController.cs. Estos controladores son responsables de controlar las vistas de administrar y nuevo-editar (.cshtml) asociadas a cada entidad del proyecto.

Estos controladores se almacenan en la carpeta de controladores dentro de la estructura de Modelo-Vista-Controlador (MVC).

Cada uno de estos controladores define métodos que manejan las solicitudes entrantes de las vistas correspondientes, realizando operaciones como la recuperación y presentación de datos, el procesamiento de formularios y la gestión de acciones del usuario.

```
public class EstacionIslaController : Controller
{
    [AutorizarMvc(CodigoPrograma = Programa.Codigos.ISLA, CodigoAccion = Accion.Codigos.VER)]
    0 referencias
    public ActionResult AdministrarIsla()
    {
        var modelo = new EtiquetasModel();
        return PartialView(modelo);
    }

    [AutorizarMvc(CodigoPrograma = Programa.Codigos.ISLA, CodigoAccion = Accion.Codigos.ALTA)]
    0 referencias
    public ActionResult NuevoIsla()
    {
        var modelo = new EtiquetasModel();
        return View("NuevoEditarIsla", modelo);
    }

    [AutorizarMvc(CodigoPrograma = Programa.Codigos.ISLA, CodigoAccion = Accion.Codigos.MODIFICAR)]
    0 referencias
    public ActionResult EditarIsla(string codigo)
    {
        var modelo = new EtiquetasModel();
        return View("NuevoEditarIsla", modelo);
    }
}
```

*Ilustración 12 - EstacionIslaController.cs*

4. Modificación del archivo MenuLateral.cshtml y menu-lateral.js: Procedo a realizar la modificación del archivo MenuLateral.cshtml y menu-lateral.js para agregar accesos a las vistas de los ABM recientemente creados al menú lateral de la aplicación.

En el archivo MenuLateral.cshtml, agrego componentes li que representan ítems del menú lateral correspondientes a los nuevos ABM. Estos ítems proporcionan enlaces a las vistas asociadas a cada uno de los ABM, permitiendo a los usuarios acceder fácilmente a estas funcionalidades desde el menú lateral.

Además, creo los métodos necesarios en el View Model de la vista encargada de mostrar el menú lateral. Estos métodos se encargan de gestionar la lógica necesaria para mostrar los nuevos ítems del menú lateral y asegurar una navegación fluida y coherente dentro de la aplicación.

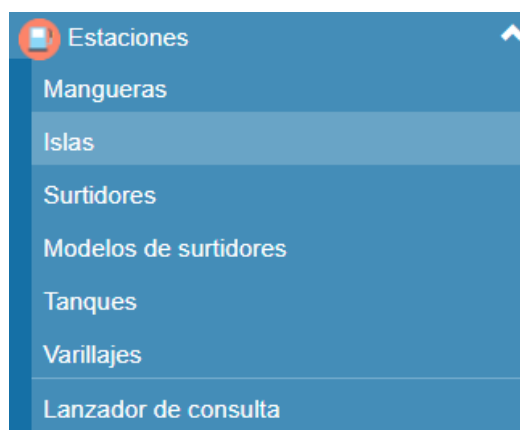
```
<li ng-if="listadoDePermisos.permisosIslas == true"
    ng-click="irIsla()"
    ng-class="{submenuDesplegado: muestraEstaciones}">
  <div>
    <span>{{Etiquetas.GetString("Islas")}}</span>
  </div>
</li>
```

*Ilustración 13 - ítem de menú Isla*

5. Configuración de la aplicación para dar de alta los ítems del menú lateral: finalmente configuro la aplicación, para agregar los ítems del menú lateral anteriormente creados. Estos ítems se han dado de alta desde la aplicación utilizando un esquema de permisos específico.

La implementación de este esquema de permisos permite controlar el acceso a los nuevos ítems del menú lateral, asegurando que solo los usuarios autorizados tengan la capacidad de visualizar y utilizar estas funcionalidades.

Con esta configuración finalizada, se da por cumplida la petición 1921, satisfaciendo así los requisitos de la petición y garantizando una correcta integración de los nuevos ítems del menú lateral en la aplicación.



*Ilustración 14 - Menú Lateral modulo Estaciones*

#### **Petición #1924:**

En esta petición se me encomendó, junto al resto de los pasantes, agregar otro ABM al módulo de estaciones, el ABM de “Varillajes”, el cual es encargado de registrar las mediciones periódicas que se realizan a los tanques, por lo cual no solo tiene relación con al ABM de tanques, sino también se relaciona con un parte de caja, determinándose así, por diferencia entre los distintos varillajes, la variación en litros de combustible de un determinado tanque.

Para dar inicio a esta tarea, procedí a cargar un script (provisto por el gerente de desarrollo), para dar de alta la nueva tabla y modificar la tabla de tanques, agregando las referencias a la tabla de varillajes y parte diario de caja.

Luego, una vez creada la tabla, el proceso de creación del ABM fue el mismo que para los anteriores, dando como resultado una disminución notable en los tiempos de desarrollo, consecuencia del aprendizaje de la petición anterior.

Esta petición tenía dos novedades respecto a la anterior. La primera novedad era que para la edición del registro, debíamos verificar si el parte de caja asociado estaba abierto y, en caso contrario, no permitir la edición del registro y mostrar una notificación al usuario. En segundo lugar, debíamos asegurarnos que al seleccionar el tanque al que correspondía el varillaje, este traiga el producto (tipo de combustible) asociado al mismo y complete automáticamente este campo, no permitiendo la modificación del mismo y manteniéndose constante aun cuando con posterioridad se modifique el combustible del tanque sobre el que se realiza el varillaje.

Una vez finalizada la tarea encomendada, el Gerente de Desarrollo realizo una

ampliación sobre la misma, en la cual se nos solicitó dos tareas. Para la primera debíamos agregar una pestaña en el ABM de tanques donde se muestren los varillajes asociados a un determinado tanque, el cual era previamente seleccionado de la pestaña principal de tanques. En segundo lugar debíamos agregar, en el ABM de varillajes, dos tipos de filtros, uno en la parte superior de la lista donde filtrara en tiempo real la lista según las coincidencias con las palabras que fuéramos escribiendo y un segundo filtro, el cual se desplegara y permitiera filtrar según rango de fechas, parte de caja y usuario.

Para llevar adelante estas implementaciones, el principal desafío radicó en comprender el código de otros módulos donde ya se encontraban implementadas pestañas y filtros. Una vez superada esta instancia, agregar la pestaña solicitada no reportó grandes inconvenientes, ya que en su mayoría, el desarrollo implicaba la modificación del archivo .cshtml nuevo-editar de tanques para agregar el código de la pestaña y la información que se debía visualizar, así como también modificar el view model asociado a este para agregar los métodos necesarios, junto con algunas de modificaciones menores en otros archivos. En cuanto a la implementación de los filtros, dada la complejidad de los métodos que debía implementar junto con la necesidad de crear nuevos filtros, tornaron el proceso de desarrollo bastante más extenso y tedioso, implicando un proceso de debugging para eliminar errores, recuperar y mostrar la información en la forma requerida.

Una vez cumplidos con estos requerimientos y para finalizar con el módulo de aforadores, se me solicitó implementar, junto al resto de los pasantes, dos lanzadores de consultas, uno en el menú lateral para consultas del módulo en general, y otro en forma de pestaña dentro del ABM de tanques para consultas específicas de estos. Esta última petición también solicitaba la creación de reportes en formato XLS y PDF (varillajes por

tanque, tanques existentes).

Para llevar adelante este requerimiento, el Gerente de Producción, dictó vía Discord, una capacitación en la cual nos explicó el funcionamiento de los lanzadores de consulta, el esquema de permisos y como darlos de alta en el menú lateral de la aplicación.

Dado que estaba implementado ya en otros módulos, la tarea de crear los lanzadores de consulta no implicó un gran desafío más allá de analizar el código y replicarlo con las adaptaciones necesarias en el módulo de aforadores de estaciones de servicio.

En cuanto a la creación de las consultas, debemos hacer dos distinciones, las consultas en formato XLS y las que son en formato PDF. Para ambas consultas se debía crear en el código de la aplicación las variables (que se le asignaban valores tipo <<codigoNombre>> que luego son remplazados por valores que se establecen en el filtro) que se utilizarían para filtrar las consultas e incorporarlas a una tabla especial en la BDD de SQL server donde se almacenan los valores (<<codigoNombre>>) de este tipo de variables y, finalmente se crea en lenguaje Transact-SQL (T-SQL) la consulta que se debe hacer a la BDD para recuperar los datos necesarios para volcar en el informe. Para el caso de los reportes en formato PDF la complejidad era mayor, ya que además de los pasos indicador en el punto anterior debíamos crear un modelo de informe en formato RDLC, el cual se relacionaba con un archivo, en este caso particular “GenericoUno” que contenía todas las variables donde se iban a almacenar los datos de la consulta SQL y que después se volcarían en el informe.

Finalmente, una vez creados los informes, procedí a agregarlos, desde la interfaz de

la aplicación, a sus respectivos lanzadores de consulta y con los filtros a utilizar.

### Tercera etapa: Asignación de peticiones individuales de Reporteria para el módulo de Hotelería:

#### **Petición #1931:**

Durante los primeros días de marzo, faltando dos semanas para la finalización de la PPS, se me asigna una petición individual de Reporteria, donde se me solicita, para el módulo de Hotelería, desarrollar un informe PDF donde se puedan visualizar por separado, los arribos y las partidas previstos para el día siguiente al del que se realiza la consulta. Tanto para los arribos como para las consultas, debía contener información sobre el número de habitación, el tipo, el nombre y apellido del pasajero, en caso de pertenecer a una el nombre de la empresa, grupo o agencia (puede darse el caso de que pertenezca a todas, una o ninguna), la fecha de ingreso, egreso y la hora prevista de arribo, cantidad de pax alojados, numero de reserva y el monto total a facturar. También debía mostrar un subtotal, al final de la tabla, donde muestre la cantidad de partidas, la cantidad de arribos, la cantidad de pax para arribos y partidas y el monto total a facturar para las partidas.

Para llevar adelante esta petición, lo primero que hice fue estudiar las tablas de donde debía obtener la información, para luego comenzar a diseñar la consulta SQL necesaria para obtener los datos que necesitaba. Una vez determinadas las tablas necesarias, comencé a escribir el script de la consulta, para la primera columna, escribí la condición para que tome como referencia la fecha actual en la que se realizaba la consulta, le sume un día y la compare con las fechas de arribos y las fechas de partidas, en caso de coincidir con la fecha de arribo se trataba de un arribo y en el segundo caso de una partida, con esto ya tenía la columna que posteriormente iba a utilizar en mi rdlc para agrupar los

registros. Esta misma lógica la utilice en una cláusula where para que solo me muestre los registros que coincidían con arribos o partidas.

Obtener los datos de las columnas del número de habitación, el tipo, el nombre y apellido del pasajero, la fecha de ingreso, egreso, la hora prevista de arribo, cantidad de pax alojados, observaciones y número de reserva, no reportó gran complejidad, ya que conociendo las tablas que intervenían y como se vinculaban, fue suficiente con escribir algunos INNER JOIN y LEFT JOIN.

Para saber si el pasajero pertenecía a alguna empresa, con una concatenación de tres expresiones CASE verificaba si en la tabla de movimientos de hoteles, el valor de la tupla que contenía el código de empresa, el código de agencia y el código de grupo eran diferente de null, y en caso afirmativo, concatenaba las tuplas de las tablas de empresa, grupo y agencia que contenían el nombre de la misma, con el formato “Empresa: , Grupo: , Agencia: ”.

En cuanto a la columna donde se mostraba el total a facturar, fue posiblemente la que me reportó más problemas, en primer lugar, para definir que conceptos iba a incluir en dicha columna para el caso de los arribos y luego los conceptos que incluiría para el caso de las partidas y en segundo lugar como iba a establecer las relaciones y uniones de tablas para obtener estos valores. Luego de definir esto con el gerente de producción e investigar las tablas de donde debía obtener los valores (valor de la reserva, cargos repetitivos, etc), pude obtener finalmente la consulta con las columnas que necesitaba para mí reporte rdlc mediante la suma de tres subconsultas donde obtenía los valores que necesitaba.

Finalmente me aboqué a diseñar el reporte con Report Viewer, teniendo en la

primera columna diferenciado si se trataba de un arribo o una partida, utilicé ese dato para que el reporte haga el agrupamiento. El cálculo de los subtotales los efectué dentro del reporte, con expresiones que usa Report Viewer escritas en Visual Basic.

Ya con mi rdlc y consulta SQL lista, procedí a cargar el informe en el sistema, y luego de algunas pruebas y corrección de errores menores, la petición estuvo lista para ser presentada al gerente de desarrollo.

## **3.2 Conclusiones**

### **3.2.1 Aprendizaje obtenido de la realización de las prácticas**

La realización de mi práctica profesional supervisada me proporcionó una valiosa experiencia práctica en el desarrollo de software, así como importantes lecciones sobre trabajo en equipo, resolución de problemas y gestión de proyectos, conocimientos que considero, serán fundamentales para mi futuro desarrollo como profesional.

Durante mi participación en la práctica profesional supervisada (PPS), adquirí una amplia gama de habilidades y conocimientos que contribuyeron significativamente a mi desarrollo como profesional.

Algunos de los aprendizajes más destacados incluyen:

- Desarrollo de módulos completos: A través del desarrollo del módulo de aforadores para estaciones de servicio, aprendí a diseñar e implementar sistemas completos de administración de datos, desde la creación de clases y entidades principales hasta la configuración de controladores de API y vistas asociadas.
- Uso efectivo de tecnologías y herramientas: Me familiaricé con una variedad de tecnologías y herramientas utilizadas en el desarrollo de aplicaciones web, incluyendo ASP.NET MVC, Entity Framework, JavaScript, Report Viewer y SQL Server.
- Trabajo en equipo y colaboración: Trabajar en equipo con otros pasantes me permitió aprender a comunicarme de manera efectiva y resolver problemas de manera conjunta. Esto me ayudó a desarrollar habilidades de trabajo en equipo y a entender la importancia de la colaboración en entornos profesionales.

- Resolución de problemas y debugging: Enfrentarse a desafíos técnicos y bugs durante el desarrollo de los módulos me permitió mejorar mis habilidades de resolución de problemas y debugging. Aprendí a identificar y abordar problemas de manera sistemática, utilizando herramientas de debugging y analizando el código.
- Gestión de proyectos y seguimiento de tareas: Participar en la planificación y ejecución de proyectos me proporcionó una comprensión más profunda de los procesos de gestión de proyectos, incluyendo la definición de requisitos, la asignación de tareas y el seguimiento del progreso. Aprendí a gestionar mi tiempo de manera efectiva y a priorizar tareas según las necesidades del proyecto.

### 3.2.2 Comentarios personales del trabajo realizado

Como primer comentario debo agradecer a las personas que hicieron posible haber llegado a esta instancia de mi carrera, a mi familia, mi esposa y mi hija que siempre me apoyaron y muchas veces resignaron tiempo para que pueda estudiar, asistir al cursado de mis materias y poder cumplir con las exigencias que me demandó la práctica profesional.

A mis Padres, quienes han sido mi mayor fuente de inspiración y apoyo a lo largo de mi vida. Desde una edad temprana, me inculcaron el valor del estudio y la perseverancia como herramientas fundamentales para alcanzar mis metas y superarme como persona. Su dedicación y ejemplo me han enseñado la importancia de esforzarme por lo que quiero lograr, incluso cuando los desafíos parecen abrumadores.

Gracias a su constante aliento y orientación, he aprendido a enfrentar los obstáculos con determinación y a nunca renunciar ante las dificultades. Su apoyo incondicional ha sido fundamental para mi desarrollo académico y profesional, brindándome la confianza

necesaria para perseguir mis sueños y aspiraciones.

Respecto a las actividades desarrolladas y los conocimientos obtenidos en las PPS, debo manifestar el orgullo que siento por los logros que he alcanzado durante todo este proceso, como la creación exitosa de módulos completos de software, la resolución de desafíos técnicos y la colaboración efectiva en equipos de desarrollo.

Trabajar en equipo con otros pasantes ha sido una experiencia muy enriquecedora. Tuve el honor de haber formado parte de un grupo de personas excepcionales, tanto desde lo humano como en lo técnico. He aprendido a comunicarme de manera efectiva, a colaborar en proyectos complejos y a apoyar a mis compañeros en la resolución de problemas.

Durante la práctica, me he dado cuenta del esfuerzo y la dedicación que se necesita en el mundo del desarrollo de software, tanto para mantenerse actualizado en las tecnologías utilizadas como la necesidad de estar permanentemente investigando y estudiando nuevas tecnologías para brindar más y mejores soluciones en nuestros desarrollos. También he identificado áreas en las que debo mejorar, como la gestión del tiempo y la resolución de problemas técnicos. Estoy comprometido a seguir aprendiendo y creciendo en estas áreas.

Debo agradecer también a todo el equipo de Grandi y Asociados por la oportunidad de realizar esta PPS en su empresa, la predisposición y el acompañamiento durante todo el proceso. Ha sido una experiencia invaluable que me ha ayudado a crecer tanto personal como profesionalmente, y estoy emocionado por aplicar lo que he aprendido en futuros proyectos.

También quiero expresar mi sincero agradecimiento a todos los docentes de la Tecnicatura Universitaria en Programación por su dedicación, compromiso y pasión en la enseñanza. Su invaluable guía y conocimientos compartidos han sido fundamentales en mi formación profesional y personal. Cada clase, cada explicación y cada consejo han contribuido significativamente a mi desarrollo como profesional en el campo de la programación. Agradezco especialmente su paciencia y disposición para responder a nuestras preguntas, resolver nuestras dudas y brindarnos orientación en cada paso del camino. Su compromiso con la excelencia educativa y su capacidad para inspirar y motivar a sus estudiantes han dejado una huella indeleble en mí, y estoy profundamente agradecido por haber tenido la oportunidad de aprender de su vasta experiencia y conocimiento. Sin su dedicación y apoyo, no habría sido posible alcanzar los logros y superar los desafíos que he enfrentado durante mi formación académica. Gracias por su constante inspiración y por ayudarme a alcanzar mis metas y aspiraciones.

### 3.2.3 Conclusiones generales

La realización de la práctica profesional supero ampliamente mis expectativas, durante el desarrollo de la misma se ha empleado un conjunto diverso de tecnologías, incluyendo C#, JavaScript, AngularJS, .NET Framework, Razor y SQL Server. Cada una de estas tecnologías ha sido seleccionada por sus capacidades específicas para las distintas partes del desarrollo.

Se ha seguido una metodología estructurada que ha implicado la creación de clases principales, administradoras, DTOs, archivos de mapeo, View Models, controladores de API, controladores para consulta y controladores de vista. Esta metodología ha permitido una organización clara y efectiva del desarrollo.

Muchas veces, la naturaleza del desarrollo no permitió la división en grupos pequeños para maximizar la eficiencia, teniendo que limitarnos a realizar las tareas de desarrollo en conjunto. Para desarrollos en los que logramos dividirnos en grupos, pudimos centrarnos en tareas específicas y trabajar en paralelo, logrando así resultados más rápidos.

Se han enfrentado desafíos técnicos durante el desarrollo, como errores en el mapeo a las tablas de la base de datos y la comprensión de la lógica de otros módulos para implementar nuevas funcionalidades. Sin embargo, estos desafíos se han superado con éxito mediante el análisis detallado y la colaboración entre los miembros del equipo.

Se logró integrar nuevas funcionalidades en el sistema existente, como la creación de los ABM para entidades del módulo de aforadores de estaciones de servicio. Esta integración se ha realizado de manera coherente y siguiendo las pautas de diseño establecidas.

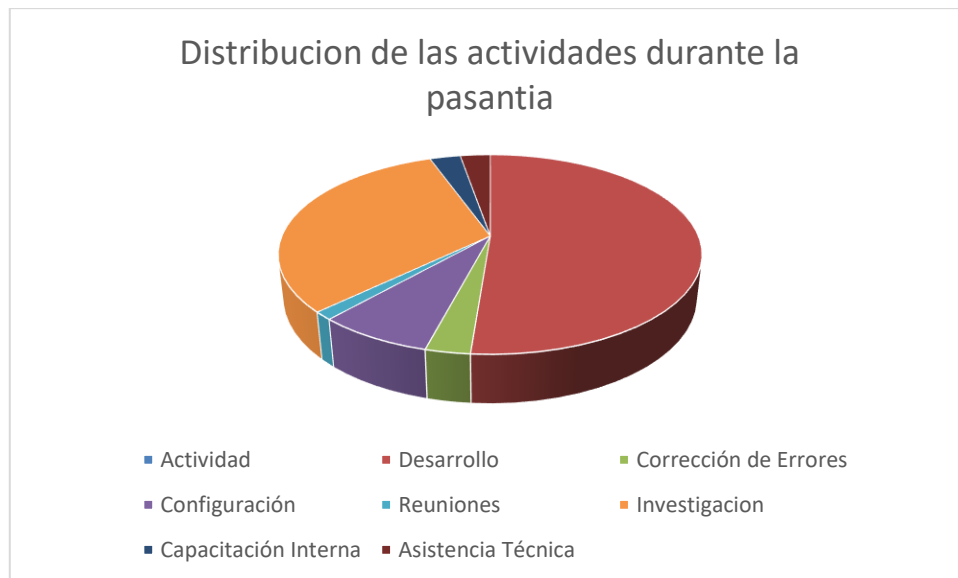
Se desarrollaron informes para el lanzador de consultas, tanto del módulo antes mencionado como para el módulo de hotelería, los cuales permiten visualizar información relevante para la toma de decisiones de los usuarios del sistema.

Para cerrar, esta práctica profesional no solo fue una experiencia profesional invaluable, sino que también complementó y amplió mis conocimientos adquiridos durante mi carrera. Me permitió aplicar lo aprendido en un entorno práctico y desafiante, fortaleciendo mis habilidades técnicas y mi comprensión de los procesos de desarrollo de software. Esta oportunidad ha sido fundamental para mi crecimiento tanto personal como profesional, consolidando mi pasión por la tecnología y preparándome para futuros desafíos en el campo de la programación.

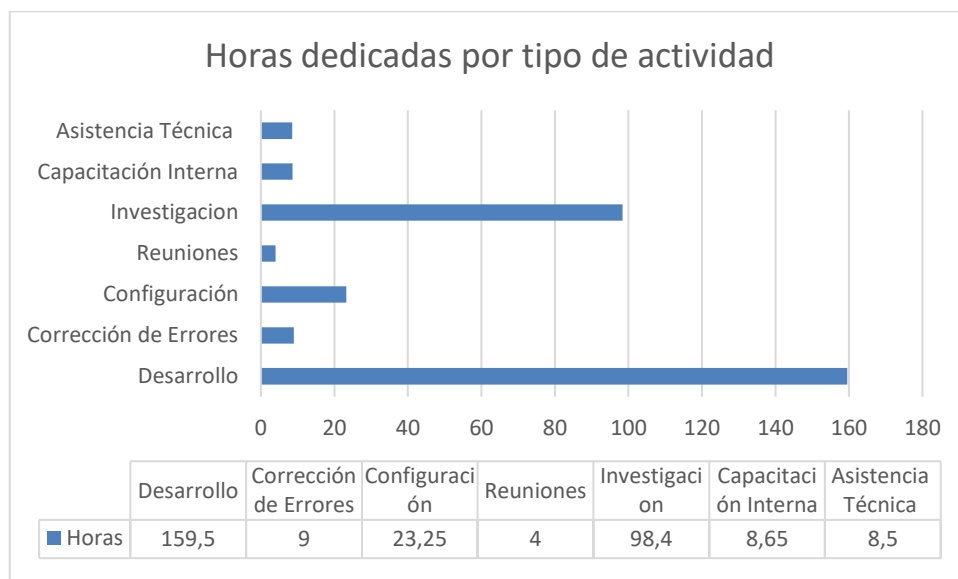
### 3.3 Recomendaciones y aportes

1. Continuar explorando nuevas tecnologías: Durante el desarrollo del proyecto, hemos utilizado un stack tecnológico sólido pero que debe ser actualizado en un futuro próximo. Recomiendo explorar nuevas tecnologías y herramientas que puedan mejorar aún más la eficiencia y efectividad de futuros proyectos.
2. Fomentar la comunicación y colaboración: La comunicación efectiva entre los miembros del equipo ha sido fundamental para el éxito del proyecto. Aconsejo seguir fomentando un ambiente de colaboración donde se promueva el intercambio de ideas y la resolución conjunta de problemas.
3. Incorporar retroalimentación continua: La retroalimentación constante durante el desarrollo del proyecto nos ha permitido realizar ajustes y mejoras de manera oportuna. Es importante seguir incorporando este enfoque de retroalimentación continua en futuros proyectos para garantizar su éxito.
4. Priorizar la documentación: La documentación detallada de los procesos y decisiones tomadas durante el desarrollo del proyecto es fundamental para el equipo de desarrolladores. Recomiendo priorizar la documentación para facilitar la comprensión y mantenimiento del sistema en el futuro.

## Anexos



*Ilustración 15 - Grafico de torta con distribución de las actividades desarrolladas durante la pasantía*



*Ilustración 16 - Grafico de barras con distribución horas por actividad*