UNIVERSIDAD NACIONAL DE LA PAMPA
FACULTAD DE INGENIERÍA

PROYECTO FINAL DE GRADO

"Diseño e Implementación de un Tablero de Gestión para el Ministerio Público de la Provincia de La Pampa"

Previa Obtención de Título de:

Ingeniero en Sistemas

Presentado por:

Rodriguez Martin Eduardo

Director de Proyecto:

Lic. Lafuente, Guillermo

General Pico, La Pampa – 2018

Agradecimientos

Quiero aprovechar este espacio para agradecer a todas las personas que han formado parte de mi carrera universitaria y de este proyecto final de grado y me han apoyado de una u otra forma para poder alcanzar este logro tan deseado.

A mis padres, que me dieron la oportunidad de poder realizar mis estudios, por su apoyo incondicional en todo momento y por los valores que me han inculcado desde pequeño.

A mis hermanos, que estuvieron siempre presentes.

A mis amigos y compañeros de estudio, con los cuales he compartido muchos momentos a lo largo de la carrera.

A los profesores de la Facultad de Ingeniería de la UNLPam, quienes dedican su tiempo y esfuerzo a enseñar y comparten su vocación con los alumnos.

A mi director del proyeto, Guillermo Lafuente, por ser mi tutor y guiarme en el desarrollo del trabajo, realizando un aporte desde su experiencia.

Al personal del Ministerio Público de la provincia de La Pampa, por permitirme ser parte de este proyecto de desarrollo, aportando a mi crecimiento profesional.

Y a todas las personas que me han acompañado a lo largo de este camino.

Muchas Gracias

Resumen

El presente proyecto describe todas las etapas del ciclo de vida del software empleadas para el desarrollo de un sistema informático para el Ministerio Público de la Provincia de La Pampa. Esta herramienta, denominada Tablero de Gestión, tiene por objetivo brindar información para el control y monitoreo de la organización, permitiendo así diagnosticar adecuadamente una situación y poder tomar decisiones a nivel gerencial en base a ella.

Como metodología de desarrollo de software, se optó por utilizar una Metodología Ágil, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque tiene una gran efectividad en proyectos con requisitos cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

La arquitectura planteada para el sistema fue una arquitectura híbrida, incluyendo los patrones MVC, MVVM y un servicio REST para realizar las consultas a la Base de Datos.

En la implementación del sistema se utilizaron tecnologías de desarrollo de la plataforma .NET como ASP.NET MVC, ASP.NET Web API y ADO.NET Entity Framework combinadas con tecnologías de código abierto basadas en JavaScript, entre ellas AngularJS y JQuery.

El proyecto requirió del esfuerzo de un equipo interdisciplinario y la coordinación con diferentes áreas del organismo para concretar su despliegue.

Tabla de contenido

Tabla de Figuras

Introducción

Hoy en día el mundo se rige por la tecnología, la información y la comunicación, estos tres conceptos están íntimamente relacionados y año a año se avanza a pasos agigantados sobre estos pilares.

En este contexto, las herramientas informáticas realizan un gran aporte, ayudando al perfeccionamiento de la gestión de las organizaciones y permitiendo el crecimiento institucional.

Desde hace varios años, se volvió indiscutible la importancia de contar con un sistema de control de gestión en organizaciones tanto públicas como privadas. En este sentido, el Ministerio Público de la Provincia de La Pampa coincide en el valor que otorga disponer de un sistema de este tipo y los beneficios que aportan a la institución.

Por tales razones, el presente proyecto pretende dar solución a la necesidad actual del Ministerio Público de contar con una herramienta informática que le permita tener al alcance información acerca de su situación actual, para poder monitorear y controlar de manera más efectiva la gestión del organismo.

La creación de un Tablero de Gestión para el Ministerio Público, es una tarea que requiere del esfuerzo de todas las partes del organismo para poder lograr su realización.

El presente informe describe, desde el punto de vista de la Ingeniería de Software los aspectos teóricos y prácticos involucrados en el desarrollo del Tablero de Gestión para el Ministerio Público, incluyendo todas las etapas del ciclo de vida del desarrollo de un sistema.

1. Capítulo I: Situación Problemática

1.1. Planteamiento del Problema

El Poder Judicial de la Provincia de La Pampa actualmente cuenta con un Sistema Informático de Gestión de Expedientes Penales (SIGeLP) desarrollado por la Secretaría de Sistemas del Poder Judicial e implementado en forma simultánea con el nuevo Código Procesal Penal, en 2011. Este sistema es utilizado tanto por el Ministerio Público como por el Superior Tribunal de Justicia. Allí se vuelcan todas las actuaciones que la ley les confiere a Fiscales y Jueces en el transcurso de una causa penal, desde su inicio hasta su finalización.

Con el paso del tiempo, se fueron incorporando al SIGeLP nuevos actores. Actualmente se relacionan a través del mismo, los Jueces, la Oficina Judicial, el Ministerio Público Fiscal, el Ministerio Público de la Defensa, las Asesorías de menores, la Oficina de Atención a la Víctima y al Testigo del delito (OAVyT), la Policía y los abogados particulares.

Este sistema, al ser un sistema a nivel operativo, no contempla la generación de información para monitoreo y toma de decisiones a niveles gerenciales.

El Ministerio Público requiere de una herramienta informática que permita recopilar y organizar información que refleje su situación global, dando información estadística para diagnosticar adecuadamente una realidad y facilitar la toma de decisiones.

Este proyecto pretende dar solución a estas necesidades insatisfechas, desarrollando una herramienta informática como lo es un **Tablero de Gestión** que ayude a la toma de decisiones a nivel gerencial.

1.2. Objetivos

El objetivo general del proyecto es desarrollar una herramienta informática, particularmente un tablero de gestión, siguiendo las etapas del ciclo de vida del software e integrando nuevas tecnologías de desarrollo para lograr un sistema que se adecúe a las necesidades del Ministerio Público antes mencionado.

El tablero de gestión deberá cumplir con los requerimientos planteados por el personal de Procuración del Ministerio Público y servirá como una herramienta de monitoreo y control, dando la información necesaria para aportar al proceso de toma de decisiones por parte del Nivel Gerencial de la Organización.

1.3. Justificación del Problema

La realización de una herramienta de este tipo es de suma importancia en una organización gubernamental como el Ministerio Público, ya que permite perfeccionar la gestión y el crecimiento institucional del mismo.

El control de la gestión es esencial y no se le debe dar menor importancia. Un tablero de Control permite entre otras cosas: planificar y establecer metas e indicadores, difundir objetivos estratégicos en las diferentes áreas y niveles de la organización, y proporciona información de gran utilidad para la toma de decisiones.

Ante esta demanda, se presenta a continuación el informe del trabajo realizado a fin de dar cumplido a este requerimiento.

1.4. Estructura del Informe

El informe correspondiente al proyecto final se organizó de manera que los conceptos se vayan introduciendo gradualmente y conteniendo tanto fundamentos teóricos como detalles de implementación.

En el Capítulo 1, se plantea el problema a resolver con sus objetivos y justificaciones.

El Capítulo 2, se centra en describir la organización sobre la cual fue desarrollado el sistema.

En el Capítulo 3, se introducen conceptos teóricos sobre los Tableros de Gestión y sus características.

El Capítulo 4, realiza una descripción del proyecto de desarrollo, describiendo la metodología usada y los principales requerimientos.

El Capítulo 5, se centra en aspectos de diseño del sistema, describiendo las arquitecturas y servicios web utilizados.

El Capítulo 6, describe de forma resumida las tecnologías y herramientas involucradas en el desarrollo del proyecto.

El Capítulo 7, se basa en la implementación del tablero de gestión, mostrando funcionalidades, estructura del código y un ejemplo de desarrollo.

En el Capítulo 8, se describen prácticas realizadas para el control de calidad y despliegue del software en cuestión.

Por último, se presentan las conclusiones obtenidas del desarrollo del presente trabajo final, teniendo en cuenta el problema y los objetivos planteados.

Como Anexo, se expone la Resolución N° 188/17 dictada por el Procurador General del Ministerio Público, que pone formalmente en conocimiento la herramienta desarrollada a todos los usuarios que les concierne.

2. Capítulo II: La Organización

Como etapa inicial de análisis de sistemas en todo proceso de desarrollo, es fundamental comprender el área en el cual se va a trabajar, el funcionamiento y estructura de la organización comprometida, así como sus procesos [1].

El presente capítulo, intenta describir brevemente la organización del Ministerio Público de La Pampa, para situarnos en contexto, y las etapas del sistema penal acusatorio pampeano.

2.1. Introducción

El Ministerio Público es un órgano que integra el Poder Judicial de la provincia de La Pampa y tiene como función principal promover la actuación de la justicia en defensa de la legalidad de los intereses generales de la sociedad. Su autoridad máxima está representada por el Procurador General como jefe de los Fiscales, Defensores y Asesores de Menores. [2]

A partir de la puesta en vigencia de la Ley Nº 2574 Orgánica del Poder Judicial, el Ministerio Público logra autonomía funcional, administrativa y financiera.

La estructura del MPF se conforma de unidades especializadas, que tienen como fin mejorar el desempeño en el trabajo. Ello requiere sí o sí un Ministerio ágil y abierto a los cambios que la realidad va demandando de modo constante.

El Ministerio Público, se corresponde con el sistema llamado acusatorio, donde el fiscal investiga y el juez decide. Esto garantiza un juez imparcial que, a la hora de tomar decisiones, deberá valorar lo que le presenten las partes.

2.2. Reseña Histórica

En el transcurso de los últimos años la provincia de La Pampa ha experimentado quizás los cambios más importantes de su historia judicial, resumidos en una reforma procesal penal que dio como resultado un nuevo Código Procesal Penal y una nueva Ley Orgánica del Poder Judicial.

La Reforma Procesal Penal es fruto de un gran esfuerzo en el que intervinieron tanto el Poder Ejecutivo, Legislativo y Judicial, como así también distintos sectores e instituciones de la sociedad pampeana que brindaron su tiempo y capacidad en estudiar, diseñar y mejorar nuestro sistema procesal penal.

En este marco, en el año 1999, un grupo de profesores de la Universidad Nacional de La Pampa redactó y presentó un proyecto de reforma del Código Procesal Penal vigente en aquel momento -Ley Nº 332-.

Este proyecto fue aceptado y aprobado con éxito en el transcurso del año 2000 y sirvió de base para la posterior sanción de la Ley Nº 2287 del Código Procesal Penal y la Ley Nº 2574 Orgánica del Poder Judicial.

De esta forma la provincia de La Pampa se incorporó a los estándares internacionales en materia de procedimiento penal institucionalizando el Sistema Acusatorio, por sobre el viejo de carácter Inquisitivo.

2.3. Estructura Organizacional

El Ministerio Público de La Pampa, se encuentra encabezado por la Procuración General, a cargo del Procurador General y está constituido por el Ministerio Público Fiscal, El Ministerio Público de la Defensa y la Asesoría de Menores.

Figura 2.1: Organigrama del Ministerio Público

2.3.1. Procuración General

La Procuración General es la sede de actuación del Procurador General, que además de llevar a cabo las funciones del Procurador, es la unidad de ejecución funcional, operativa y administrativa que lleva adelante la gestión del Ministerio Público.

Como mencionamos arriba, el Procurador General es el Jefe del cuerpo de Fiscales, Defensores y Asesores de Menores que componen el Ministerio Público, estableciendo su unidad de acción.

La función del Procurador General es representar al Ministerio Público ante el Superior Tribunal, continuando con la intervención que los representantes del Ministerio Público Fiscal tuvieron en las instancias inferiores e interviene en todas las causas de competencia originaria y exclusiva del Superior Tribunal y en las que éste deba conocer y decidir por vía de los recursos de casación, inconstitucionalidad, revisión y extraordinario.

Además, hay funciones previstas por la ley orgánica del Poder Judicial N° 2574 que se vinculan directamente con el desarrollo de una herramienta de gestión. Entre ellas el Procurador General debe "Fijar la planificación general del Ministerio Público y controlar su cumplimiento, optimizando los resultados de la gestión" (inc. 9 del art. 96), "Impartir instrucciones para cumplir con eficacia los deberes a cargo del Ministerio Público, tendientes a procurar la unidad de acción de los funcionarios al servicio del organismo" (inc 13 del art. 96), "Supervisar la tarea de los miembros del Ministerio Público, practicando visitas de inspección y auditorías, y organizar un adecuado sistema de control de gestión Permanente" (inc 14 del art. 96).

2.3.2. Ministerio Público Fiscal

El rol que le corresponde a los fiscales que conforman el MPF tiene fines y objetivos muy claros: defender los intereses generales de la sociedad, porque al Estado y a la sociedad le interesa particularmente que se persigan todos los delitos.

En el Ministerio Público los fiscales, en los procesos penales, son quienes llevan adelante la investigación de los delitos. Esa dinámica (fiscal que investiga y juez que decide) se corresponde con un sistema llamado "acusatorio", donde es el fiscal quien debe realizar la acusación durante todo el proceso.

El Ministerio Público Fiscal a través de la Oficina de Atención a la Víctima y a los Testigos brinda protección, contención y asesoramiento a la víctima o testigo de un delito durante todo el procedimiento penal.

2.3.3. Ministerio Público de la Defensa

El Ministerio Público de la Defensa es la institución encargada de asegurar la efectiva asistencia, el acceso a la Justicia y la defensa judicial de los derechos de las personas. El servicio es brindado por los defensores públicos, que integran el organismo, en todo el territorio de la Provincia de La Pampa.

En nuestra Provincia, el Ministerio Público de la Defensa está compuesto por Defensores Oficiales en lo Civil y Penal. El Defensor General es la máxima autoridad del cuerpo de Defensores y responsable de su buen funcionamiento, debiendo coordinar con el Procurador General los objetivos propuestos.

2.3.4. Asesoría de Menores

Su función es asesorar obligatoriamente a los jueces en aquellos casos en los que uno o más menores o personas incapaces, se encuentren involucrados. Su función es cuidar que los intereses del menor o los incapaces y las leyes de protección de menores se hayan cumplido.

2.4. Ministerio Público Fiscal

Antes de la reforma del Código Procesal Penal de La Pampa el Ministerio Público estaba conformado por un modelo rígido y era reflejo de los tribunales y juzgados ante los cuales intervenía, ello dificultaba dar respuestas directas a los distintos tipos de delitos sobre los que actuaba.

Con la puesta en funcionamiento del nuevo Código Procesal Penal a partir de marzo de 2011, se estableció el diseño organizacional del MPF redistribuyendo funciones sobre la base de los principios de flexibilidad, especialización, trabajo en equipo, responsabilidad personal en el caso y compartida en relación con el resultado de la gestión.

Manteniendo la base del diseño del MPF se avanzó en mayo de 2015 en la reestructuración de la sede Santa Rosa. La reforma tuvo como principal eje la creación de las fiscalías temáticas o agencias especializadas. El mismo año también se aplicó la reestructura en sede General Pico.

2.4.1. Organigrama del MPF

El Ministerio Público Fiscal se organiza por medio de Oficinas Únicas (OUMPF) distribuidas por sede en 5 puntos de la Provincia: Santa Rosa, General Pico, General Acha, 25 de Mayo y Victorica.

Las Jefaturas de estas oficinas son ejercidas por 4 Fiscales Generales: dos tienen a cargo las sedes Santa Rosa y Victorica, uno la sede General Pico y el restante Fiscal General tiene a cargo las sedes General Acha y 25 de Mayo.

Figura 2.2: Organigrama del Ministerio Público Fiscal

2.4.2. Estructura de Funcionamiento Interna del MPF

Las sedes de Santa Rosa y General Pico, están conformadas por:

- **Unidades de Base** (que interactúan con todas las unidades fiscales),

- Unidad de Atención Primaria (UAP):
  - Recibe las denuncias hechas personalmente en el MPF y aquellas que se realizaron en la policía.
  - Custodia y organiza los elementos secuestrados.
  - Efectúa un análisis jurídico de las causas ingresadas y realiza una selección temprana de los casos que pueden ser llevados con éxito a un juicio oral o aquellos en los que puede lograrse una sentencia desde la audiencia de formalización.
  - Determina una estrategia de investigación en las causas con autores ignorados.
  - Realiza las reservas y archivos de las causas que no prosperan en el procedimiento penal.
  - Tramita las salidas tempranas al proceso y deriva las restantes causas a las fiscalías temáticas según el ámbito de actuación.
- Unidad de Ejecución Penal:
  - Control en la ejecución de las sentencias de condena.
  - Dictaminar en pedidos de los condenados ante el Juez de Ejecución.
- Unidad de Expedición y Recepción de antecedentes penales (UER):
  - Informes sobre antecedentes penales.
- Oficina de Atención a la Víctima del Delito y a los Testigos (OAVyT):
  - Brinda asistencia integral en lo jurídico, psicológico y social a la víctima del delito y al testigo, facilitando su participación activa en el proceso penal.

- **Fiscalías Temáticas** (áreas especializadas que intervienen de acuerdo a su ámbito de actuación), integradas por:
  - Delitos Económicos y Delitos contra la Administración Pública
  - Delitos contra la Propiedad y Juicios Directos
  - Delitos contra las Personas
  - Delitos que impliquen Violencia Familiar y de Género
  - Causas No Penales y derivadas del Juzgado de la Familia y del Menor

2.5. Etapas del Proceso Penal

Etapas del procedimiento penal acusatorio de la Provincia de La Pampa [3]:

- **Etapa de Inicio:**

1. Denuncia: El proceso penal se inicia mediante denuncia ante la Policía, el MPF, o iniciada de oficio por el Fiscal. Se crea el legajo penal

2. Inicio de la Investigación Fiscal Preparatoria (IFP): A partir de la denuncia el Fiscal comienza a investigar y observa las circunstancias del hecho y los elementos probatorios con los que cuenta, realizando un análisis integral del caso.

   En esta etapa pueden surgir algunas alternativas:

   - ***Desestimación***: el Fiscal considera que la denuncia realizada no corresponde a un delito penal
   - ***Reserva****:* transcurrido un tiempo, y al no poder encontrar elementos probatorios para continuar con la IFP, el Fiscal reserva las actuaciones del legajo penal hasta que disponga de nuevas pruebas para retomar la investigación
   - ***Archivo****:* si la denuncia fue desestimada o hace un tiempo que las actuaciones están reservadas el Fiscal archiva el legajo penal y se lo comunica a la víctima.

- **Etapa de Investigación:**

3. Continuación de la IFP: El Fiscal profundiza la investigación y se auxilia con la fuerza policial dependiente del Poder Ejecutivo.

Además: toma declaraciones a víctimas, testigos e imputados; solicita informes periciales; a fin de recopilar la mayor cantidad de pruebas para poder fundamentar su acusación o solicitar el sobreseimiento.

i) aplicación de criterios de oportunidad

4. Formalización de la IFP: En este punto el Fiscal le comunica al Juez de Control que está investigando a una o varias personas, especificando los delitos y notificando a todas las partes (víctimas, imputados, defensor público, defensor particular y querellantes).

5. Finalización de la IFP: Una vez formalizada la IFP, el Fiscal cuenta con un determinado tiempo para definirse por la acusación o el sobreseimiento de los imputados del punto anterior. Si se define por la acusación se inicia la etapa de procedimiento Intermedio.

i) La IFP tiene un plazo ordenatorio de 90 días, sin perjuicio de la prorroga ordinaria o extraordinaria que se pueda acordar por la complejidad del caso.

ii) En esta instancia pueden surgir las medidas denominadas ***salidas tempranas*** del procedimiento penal, estas son: la ***suspensión del proceso a prueba*** y el ***juicio abreviado***.

- **Etapa de Procedimiento Intermedio:**

6. Etapa procesal siguiente a la IFP que consiste en la notificación a las partes de la decisión adoptada por el Fiscal, en tanto acuse o solicite el sobreseimiento del imputado, para que aquellas planteen sus oposiciones o peticiones ante el Juez de Control.

- **Etapa de Juicio:**

7. Se desarrolla el juicio oral y público, donde todas las partes exponen sus pruebas.

8. El Juez dicta sentencia condenatoria, absolutoria o mixta sobre los delitos en los que el Fiscal solicitó condena para los acusados. En esta instancia el Fiscal también puede solicitar el sobreseimiento.

i) Los tiempos del procedimiento penal son totalmente dinámicos (sin perder de vista que muchos están establecidos en el CPP). Podemos encontrarnos que existen causas que finalizan en menos de un mes (Ej. Un Juicio directo) y otras que demoran más de un año, dependiendo de la complejidad de cada caso.

A partir del estudio realizado, y habiendo reconocido el funcionamiento y estructura del Ministerio Público, se avanzará en el siguiente capítulo sobre Tableros de Gestión.

3. Capítulo III: El Tablero de Gestión

El presente Capítulo tiene por objetivo introducir los conceptos teóricos y características de los Tableros de Gestión y la justificación de aplicarlo en una organización como el Ministerio Público.

3.1. ¿Qué es un Tablero de Gestión?

El **Tablero de Gestión (TG)** o también llamado Tablero de Control [4] es una herramienta del campo de la administración de empresas, aplicable a cualquier organización y nivel de la misma, cuyo objetivo y utilidad básica es diagnosticar adecuadamente una situación y facilitar la toma de decisiones.

Se le define como el conjunto de indicadores cuyo seguimiento y evaluación periódica permitirá contar con un mayor conocimiento de la situación de su empresa o sector apoyándose en las tecnologías informáticas.

El diagnóstico y monitoreo permanente de información, es la base para mantener un buen control de situación. Es necesario generar metodologías gerenciales para que las empresas no se basen sólo en la intuición y conocimientos de cada directivo.

La mayoría de las empresas lo utilizan para la planeación estratégica, tener información actualizada y accesible para el control del cumplimiento de sus objetivos y metas.

El tablero de control es una metodología gerencial que sirve como herramienta para la planeación y administración estratégica de las empresas. [5]

Tal como señala Mas Sabaté [6]:

> *"El Tablero de Comandos es un instrumento que recoge de forma sintética y sistematizada la información relevante sobre la gestión, la realización de actuaciones y el logro de objetivos de una organización, con la finalidad de ser usado por los directivos y/o responsables de diferentes niveles jerárquicos, especialmente en la toma de decisiones. Es, por tanto, un instrumento de gestión orientado a facilitar la acción. Forma parte de un sistema de información y está concebido como instrumento de pilotaje de la organización."*

El soporte de esta información suele ser informática.

Jorge Hintze, señala que:

> *"Los tableros de control son herramientas que generan transparencia sobre ámbitos de otra manera opacos a la percepción directa. En tal sentido, no sólo son instrumentos para la toma de decisiones sino,*

también, para la articulación entre los actores y la rendición de cuentas. Para ello es preciso que los tableros (o cualquier otra fuente de información que pueda ser utilizada de tal manera) permitan llegar hasta el origen de los datos, en cuyo caso, se garantiza que la información sea observable y eventualmente auditable".

3.2. Tipos de Tableros

En función de las necesidades de las empresas se pueden aplicar cuatro tipos genéricos de Tablero de Control según el enfoque realizado por Alberto Ballve (2008) en su libro "Tablero de control". Ellos son:

- **Tablero de Control Operativo:** permite hacer un seguimiento, al menos diario, del estado de situación de un sector o procedimiento de la empresa para poder tomar a tiempo las medidas correctivas necesarias.
- **Tablero de Control Directivo:** permite monitorear los resultados de la empresa en su conjunto y de los diferentes temas clave en que puede segmentarse. Está más orientado al seguimiento de los resultados internos de la organización en su conjunto y en el corto plazo.
- **Tablero de Control Estratégico:** nos brinda la información interna y externa necesaria para conocer la situación y evitar llevarnos sorpresas desagradables con respecto al posicionamiento estratégico y a largo plazo de la empresa.
- **Tablero de Control Integral:** información relevante para que la alta dirección pueda conocer la situación integral de su empresa. Engloba las tres perspectivas anteriores.

3.3. Objetivos del tablero de Gestión

Los objetivos principales de un tablero de gestión son [7]:

- Medir los avances y cumplimiento de la visión, misión, valores, objetivos y estrategias de la organización.
- Difundir vertical y horizontalmente los objetivos estratégicos e indicadores en las diferentes áreas y niveles jerárquicos de la organización.
- Sincronizar los objetivos y metas de la dirección general con las demás áreas.
- Orientar los esfuerzos hacia la satisfacción de las necesidades de la organización en las diferentes áreas.
- Servir de guía para el diseño e implantación de sistemas de evaluación del personal basada en el desempeño.
- Proporcionar información para el control estratégico y operacional.
- Facilitar la toma de decisiones por parte de los directivos.

- Medir el desempeño de la organización y facilitar el control.
- Dar una visión global e integral de la organización.

Hay que tener en cuenta que el Tablero tiene determinado alcance que lo limita pero a su vez refuerza su utilidad [4]:

- Refleja solo información cuantificable: Si bien es útil para intentar cuantificar lo que antes considerábamos no cuantificable hay límites claros que indican que el tablero debe ser complementado con otras herramientas de control, formales e informales.
- No focaliza totalmente la acción directiva: en principio establece qué mirar para diagnosticar y generar un buen ambiente de análisis. Esto sin embargo es un gran avance.
- No reemplaza el juicio directivo: siempre habrá que aplicar el sentido común para emitir juicio a partir de la información. El uso de la herramienta debe estar fundamentado en el desarrollo de una estrategia organizacional previa y en la construcción de una propuesta de valor.

3.4. Un TG para el Ministerio Público

Todo proceso de gestión pública debe perseguir la creación de valor público, lo que tiene que reflejarse en resultados, cuantificables, que permitan evaluar el rendimiento en todas sus dimensiones. Para la determinación de resultados, se requiere analizar el desempeño y el nivel de alcance de las metas, trazadas con sistemas de información para el seguimiento, evaluación y control, que fundamenten la toma de decisiones y medidas correctivas [8]. La gestión requiere la implementación de sistemas de monitoreo, medición y control que permitan un adecuado control, y la toma de decisiones.

Dada la compleja organización que posee el Ministerio Público, tal como se introdujo en el capítulo anterior, éste requiere de una herramienta informática que permita recopilar y organizar información de la situación global del mismo, dando información estadística para diagnosticar adecuadamente una situación y facilitar la toma de decisiones. La realización de una herramienta de este tipo es de suma importancia, ya que permite perfeccionar la gestión y el crecimiento institucional del mismo.

El control de la gestión es esencial y no se le debe dar menor importancia. Un tablero de Control como describimos en la sección 3.3 permite entre otras cosas: planificar y establecer metas e indicadores, difundir objetivos estratégicos en las diferentes áreas y niveles de la organización y proporciona información de gran utilidad para la toma de decisiones.

4. Capítulo IV: Descripción del Proyecto de Desarrollo

El presente Capítulo detalla aspectos relacionados con el proyecto de desarrollo, describiendo el equipo de trabajo, la metodología de desarrollo aplicada y los requerimientos de software planteados para el Tablero de Gestión.

4.1. Equipo de Trabajo

Con el fin de concretar el proyecto, se creó un equipo de trabajo interdisciplinario, conformado por: por un lado la Secretaria de Política Criminal, Gestión y Planificación Estratégica Dra. Carolina Ghione y el empleado de la Procuración General Emanuel Soria, quienes definieron los requerimientos del sistema, cambios y mejoras y evacuaron las dudas que surgieron a lo largo del proceso y coordinaron los esfuerzos de trabajo; por otro lado los estudiantes avanzados de la carrera de Ingeniería en Sistemas de la Facultad de Ingeniería: Rodriguez Martin (quien escribe) y Juan Barbero - contratados como pasantes - avocados al desarrollo del proyecto del sistema.

Se contó también con el contacto continuo con personal de Secretaría de Sistemas y Organización de la Pampa por cuestiones de permisos de acceso y despliegue del software en sus servidores.

4.2. Metodología de Desarrollo de Software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros.

Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto.

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas [9]. Este enfoque tiene una gran efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Los métodos ágiles proponen una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

4.2.1. El Manifiesto Ágil

Los propulsores de las metodologías ágiles firmaron un manifiesto donde se expresaban las ideas fundamentales de la metodología [10]:

- **Valorar a las personas y las interacciones del equipo de desarrollo, por encima de los procesos y las herramientas**. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno.
- **Valorar el software que funciona, por encima de la documentación exhaustiva**. la documentación es necesaria dado que permiten la transferencia del conocimiento, pero su redacción debe limitarse a aquello que aporte valor directo al producto/servicio.
- **Valorar la colaboración con el cliente, por encima de la negociación contractual**. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- **Valorar la respuesta a cambios, más que seguir estrictamente un plan**. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto, determina también el éxito o fracaso del mismo. Por ende, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo. Los principios son [11]:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII. El software que funciona es la medida principal de progreso.
VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
X. La simplicidad es esencial.
XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

El esquema de desarrollo de software "tradicional" ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos). Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico.

Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

En este sentido a continuación de describirá el uso de esta metodología para abordar el desarrollo del TG.

4.2.2. Metodología ágil para el TG

Teniendo en cuenta las características y beneficios de las metodologías ágiles, y la naturaleza del proyecto del Tablero de Gestión, se optó por realizar un desarrollo basado en el enfoque de las metodologías ágiles.

Si bien hay una serie de propuestas muy populares como eXtreme Programming (XP), Scrum, DSDM, entre otras [9], para el proyecto presente, no se eligió una en particular, sino que se utilizó la filosofía de metodologías ágiles y se generó, podríamos decir, una metodología híbrida, tomando lo más conveniente de las propuestas mencionadas para este proyecto en particular.

En la siguiente imagen se puede ver el proceso base que se siguió en el desarrollo del proyecto, de forma resumida.

Figura 4.1: Metodología de Desarrollo Ágil

En una primera instancia se realiza un **Estudio del estado actual de la Organización**, donde se tienen en cuenta la estructura de la organización, los procesos existentes, el contexto y aspectos relacionados a la naturaleza del problema.

Una vez inmersos en el tema y contexto, se debe hacer un **Análisis de Requerimientos**, donde se identifican las necesidades propias del sistema a desarrollar, limitaciones, problemas y objetivos.

Luego la etapa de **Diseño**, en la cual se definen aspectos fundamentales del sistema (arquitectura, interfaz, prototipado, aspectos de implementación, herramientas, etc.).

Una vez tenido en cuenta aspectos de diseño y requerimientos, comienza la etapa de **Desarrollo**, sobre la cual se pone mucho énfasis en las metodologías ágiles, ya que se basa en realizar entregas periódicas del desarrollo para que el usuario final vaya utilizando el software, y solicite cambios. Es vital que el cliente sea parte del proceso de desarrollo y su feedback es de gran importancia. En esta etapa se va iterando sobre el sistema hasta obtener un entregable deseado.

Una vez desarrollada determinadas funcionalidades que cumplan con los requerimientos del cliente, se realiza el **Control de Calidad**, donde se identifican defectos y resuelven

errores que surjan con el testeo de la aplicación. En esta etapa el refactoring [1](reestructurar código fuente) también es muy importante para ir mejorando el producto software.

Por último, un vez que la aplicación fue testeada y cumple con lo solicitado, se da paso al **Despliegue** del software, es decir poner el sistema en producción. Este paso muchas veces requiere de la ayuda de soporte técnico para poner el sitio en funcionamiento en un entorno productivo.

Cabe destacar que estas etapas se realizan de forma iterativa e incremental [9]. Cada iteración del ciclo de vida incluye estas etapas y el objetivo de cada iteración no es agregar toda la funcionalidad para lanzar el producto completo, sino que pretende incrementar el valor por medio de "software que funciona".

4.3. Requerimientos de Software

Para el desarrollo del tablero de gestión, el proyecto se dividió en 3 etapas de desarrollo. El presente trabajo, se realizó sobre la implementación de la 1er etapa, conteniendo ésta, información primordial para el Ministerio Público (MP). Esta primera etapa permitió asentar las bases para seguir con las restantes.

Los requerimientos de esta etapa surgieron de las necesidades de Procuración del MP y a través de las reuniones realizadas a lo largo del proyecto se fueron refinando gracias a la interacción constante con el grupo de trabajo.

A continuación se describe una síntesis de los requerimientos funcionales.

4.3.1. Requerimientos Funcionales

> *"Un requerimiento funcional es un comportamiento requerido del sistema que expresa una capacidad de acción del mismo, una funcionalidad".*

1ra Etapa:

Esta primera fase se limita a la exposición de información de "**legajos en trámite**[2]".

- Seleccionar y exponer datos cuantitativos que permitan conocer cuestiones generales y realizar un control de la actividad de las cinco sedes del MPF, de exclusivo resorte del Procurador General.
- Exponer en forma diferenciada los datos sobre legajos radicados ante la UAP y aquellos que ya se encuentran asignados a un Fiscal de una Fiscalía Temática,

[1] Refactoring: técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.
[2] Legajo en trámite: legajo que no tiene a la fecha una resolución que le dé fin (ya sea por archivo, desestimación, sentencia o una salidas alternativa al proceso)

distinguiendo éstos últimos en tres grandes grupos: **legajos sin formalizar**, legajos **formalizados sin acusación** y **legajos con acusación**.

- Mostrar alertas ante situaciones problemáticas en los diferentes ámbitos, de utilidad para resolver el curso de acción.

2da etapa (por desarrollar)

Esta fase está enfocada en los resultados, se procura la obtención y exposición de información de legajos que integran el heterogéneo grupo denominado "TEXAS". Son aquellos en los que existió algún tipo de resolución fiscal o judicial que le puso fin, aunque la misma aún no se encuentre firme.

- Mostrar flujo de actividad de la UAP y de las Fiscalías Temáticas (entradas / salidas), carga de trabajo.
- Exposición de datos que permitan el análisis de la forma de conclusión de los procesos.
- Medir el tiempo que insumen las distintas etapas del proceso penal, la influencia en ello de la actuación de los distintos operadores (MPF, Poder Judicial, Peritos), los plazos entre las solicitudes formuladas y las resoluciones.
- Exponer datos representativos de la situación delictiva (cantidad de cierto tipo de delitos que se definirán s/ interés ej.: femicidios, abusos sexuales a menores, homicidios y lesiones culposas en accidentes de tránsito y ranking de delitos cometidos por N.N.), cantidad de denuncias que ingresan al MPF, etc.
- Mostrar información de los recursos disponibles en el MPF (humanos, tecnológicos, etc).

3ra etapa (por desarrollar)

Enfocada en obtener recursos para la investigación y datos de utilidad para la prevención.

- Análisis de información: Fecha/hora/lugar de comisión de los delitos, imputados, víctimas.
- Visualizar Redes delictivas y Mapa del delito.

4.3.2. Requerimientos No Funcionales

> *"Un requerimiento no funcional es una característica requerida del sistema que señala una restricción del mismo, no describe comportamiento sino características"*

Algunos requerimientos no funcionales fueron especificados por Procuración y otros fueron propuestos por el equipo de desarrollo para lograr un sistema con características deseables.

- **Rendimiento**

Es necesario que el rendimiento del sistema sea constante y los tiempos de respuesta no sean mayores a 7 segundos (dependiendo de la complejidad de la consulta que se realice), por lo que el diseño de sus componentes debe ser eficiente, siendo necesaria la aplicación de las mejores prácticas para diseño y construcción del sistema de información.

- **Disponibilidad**

El sistema debe soportar una alta disponibilidad, no debe presentar puntos de fallo, y se deben prever mecanismos o componentes que aseguren la continuidad del servicio. La disponibilidad es fundamental para un sistema de control y soporte a toma de decisiones.

- **Usabilidad**

Es la facilidad con que las personas pueden utilizar el sistema con el fin de alcanzar un objetivo concreto. En este punto se tiene en cuenta el tiempo de aprendizaje del sistema por un usuario, la tasa de errores cometidos por un usuario, mensajes del sistema, interfaces de usuario bien formadas.

- **Interoperabilidad**

El sistema debe funcionar junto con sistemas ya existentes, como lo es el SIGeLP, o futuros sistemas. La comunicación entre distintos sistemas es una característica muy deseable hoy en día.

- **Escalabilidad**

Es la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes. El rendimiento del sistema no debe decaer si aumenta la cantidad de usuarios que acceden a él, o si el sistema crece.

- **Mantenibilidad**

Es la facilidad con la que el sistema o componente de software puede ser modificado para corregir fallos, mejorar su funcionamiento u adaptarse a cambios. Es de suma importancia que el sistema sea fácil de mantener para que el esfuerzo y tiempo requerido para realizar cambios sean mínimos.

- **Seguridad**

El sistema debe ser confiable, es decir, la información manejada por el mismo debe estar protegida de acceso no autorizado y divulgación. Se deben implementar mecanismos de autenticación de usuario.

5. Capítulo V: Aspectos de Diseño

En el presente capítulo se dará una breve descripción de conceptos de arquitectura de software, estilos y patrones arquitectónicos que fueron utilizados en el desarrollo del Tablero de Gestión.

5.1. Arquitectura de Software

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema.

Una arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido para interactuar con el código fuente. La arquitectura se selecciona y diseña en base a los requerimientos y restricciones del sistema a desarrollar, es decir, en base a requerimientos funcionales y no funcionales y del tipo de sistema a desarrollar.

Bass, Clement y Kazman definen la arquitectura de software como la estructura del sistema, que incluyen los componentes de software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. [9]

Lo más común a la hora de definir qué tipo de arquitectura utilizar en un proyecto de desarrollo, es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para el caso concreto.

5.1.1. Arquitectura Cliente-Servidor

La arquitectura cliente-servidor, es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados “servidores”, y quienes demandan estos servicios, llamados “clientes”.

El cliente realiza peticiones al servidor y recibe la respuesta a dicha petición. El servidor recibe y procesa la petición, y devuelve la respuesta solicitada al cliente.

A continuación se presenta un esquema básico de una arquitectura cliente servidor, donde el medio de comunicación es Internet, podría ser una Intranet también, o una red local.

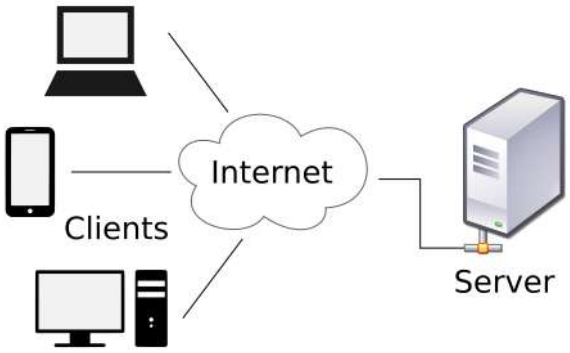

Figura 5.1: Arquitectura Cliente Servidor

La arquitectura cliente/servidor, ha evolucionado a lo largo del tiempo, pasando de arquitecturas C/S monolíticas (de una sola capa), hasta arquitecturas C/S de dos y tres capas.

En una arquitectura cliente/servidor de tres capas, el cliente implementa la capa de presentación, y tenemos un servidor que implementa la lógica de negocio (capa de negocio) y otro para el acceso a datos (capa de datos). Ésta última es la que más ventajas proporciona, destacándose que es un sistema débilmente acoplado, escalable y con mayor facilidad de mantener.

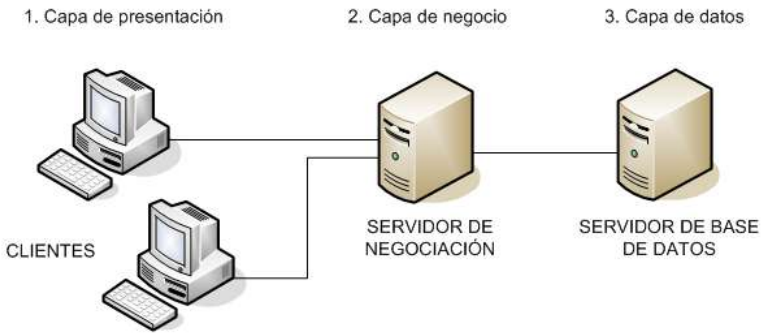

Figura 5.2: Arquitectura Cliente Servidor de 3 capas

5.1.2. Estilos Arquitectónicos

Dentro de la arquitectura de software, nos encontramos con estilos arquitectónicos [9]. Un **estilo arquitectónico** es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema.

Dentro de los estilos arquitectónicos tenemos:

- Arquitectura Centrada en Datos
- Arquitectura de Flujo de Datos
- **Arquitectura de Llamada y Retorno** (o Call/Return)
- Arquitectura Orientada a Objetos

- Arquitectura Estratificada

El presente proyecto, al ser un desarrollo orientado a la Web y utilizar un modelo Cliente-Servidor, queda definido como del tipo Llamada-Retorno, ya que el modelo cliente servidor está definido dentro de esta familia. El estilo llamada y retorno [12], enfatiza la escalabilidad y fácil modificación del sistema, y refleja la estructura del lenguaje de programación.

5.1.3. Patrones Arquitectónicos

A su vez contamos con **patrones arquitectónicos**, que al igual que un estilo, impone una transformación en el diseño de la arquitectura. Sin embargo, un patrón difiere de un estilo en varios aspectos: 1) el alcance de un patrón es menor, se concentra en un aspecto; 2) un patrón impone una regla sobre la arquitectura, pues describe la manera en que se manejará algún aspecto de su funcionalidad.

Los patrones arquitectónicos dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. [13]

Aunque un patrón arquitectónico comunica una imagen de un sistema, no es una arquitectura como tal, sino que es más un concepto que captura elementos esenciales de una arquitectura de software. Muchas arquitecturas diferentes pueden implementar el mismo patrón y por lo tanto compartir las mismas características.

Uno de los aspectos más importantes de los patrones arquitectónicos es que aportan a diferentes atributos de calidad. Por ejemplo, algunos patrones representan soluciones a problemas de rendimiento y otros pueden ser utilizados con éxito en sistemas de alta disponibilidad.

Ejemplos de patrones arquitectónicos incluyen los siguientes:

- Programación por capas
- **Tres niveles (Capa de presentación, Capa de lógica de negocio, Capa de datos)**
- Arquitectura Orientada a Servicios
- Arquitectura Dirigida por Eventos
- **Modelo-Vista-Controlador (MVC)**
- **Modelo-Vista-VistaModelo (MVVM)**
- Pipeline
- Arquitectura en Pizzarra
- Arquitectura Peer to peer

En el presente proyecto de desarrollo, al ser un sistema Web, se desarrolló bajo el modelo **cliente/servidor en 3 capas** para lograr una separación entre la capa de presentación, la capa lógica de negocio y la capa de datos.

En el proyecto web, se utilizaron los patrones **MVC**, **MVVM** y se creó un **servicio REST** con ASP.NET Web API. La combinación de estas tecnologías permitió obtener un alto grado de mantenibilidad y flexibilidad en el desarrollo del sistema.

A continuación se describen los patrones MVC y MVVM, para luego, en el Capítulo 7, poder describir con más detalle cómo interactúan estos componentes en la implementación.

5.1.3.1. Modelo-Vista-Controlador (MVC)

Es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones [14]. Para ello, MVC utiliza tres componentes distintos que son el Modelo, la Vista y el Controlador.

Este patrón de arquitectura se basa en las ideas de separación de conceptos y reutilización de código, facilitando tareas de mantenimiento y dando mayor flexibilidad a la hora de desarrollar.

Además el diseño modular permite a los desarrolladores y los diseñadores trabajar simultáneamente.

- El **Modelo** contiene una representación de la información que maneja el sistema por ende, gestiona los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio)
- La **Vista**, o interfaz de usuario, compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El **Controlador**, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno. Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo'.

Flujo de Control

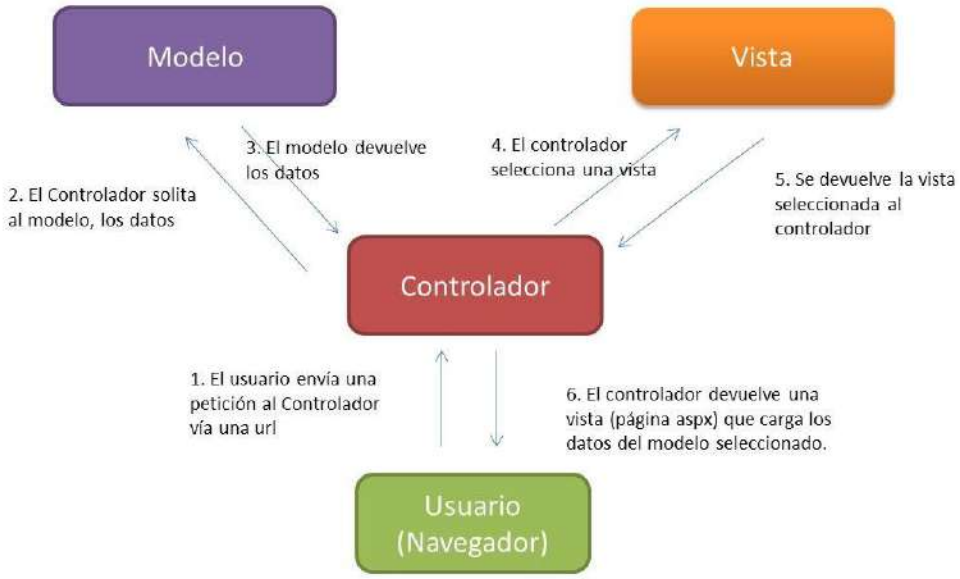

Figura 5.3: Flujo de Control MVC

El patrón MVC fue aplicado al Tablero de Gestión bajo la plataforma de ASP.NET MVC, permitiendo separar los tres conceptos del patrón de forma clara.

Algunos aspectos del patrón MVC han evolucionado dando lugar a ciertas variantes del concepto original, ya que las algunas partes del MVC clásico no tienen sentido para los clientes actuales.

Algunas de estas variantes son: MVA (Modelo-Vista-Adaptador), MVP (Modelo-Vista-Presentador), MVVM (Modelo-Vista Vista-Modelo), entre otras.

5.1.3.2. Modelo-Vista-VistaModelo (MVVM)

MVVM es un sucesor del patrón bien conocido y exitoso MVC. Como explicamos arriba una aplicación MVC se compone de tres componentes, con responsabilidades bien claras y diferenciadas: el modelo, la vista y el controlador.

Las aplicaciones MVVM agregan la capacidad de usar “bindings” para transferir datos desde y hacia la vista, por lo que el controlador sólo es responsable de implementar el comportamiento de la vista. En este caso, el controlador es llamado VistaModelo (o view model) dando origen al patrón MVVM [15].

La finalidad principal del patrón MVVM es tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación. Veamos a grandes rasgos sus partes principales:

- El **Modelo**. Lo mismo que en MVC.

- La **Vista**. Igual que en MVC. En MVVM la vista contiene enlaces a datos.
- El **Modelo-Vista**. Es responsable de exponer (convertir) los objetos de datos del modelo de tal manera que sean fácilmente administrados y presentados. Implementa el comportamiento de la vista para responder a las acciones del usuario y de exponer los datos del modelo de forma tal que sea fácil usar *bindings* en la vista. El ModeloVista es más modelo que vista y maneja la mayoría, si no toda la lógica de visualización de la vista.

  El ViewModel es un actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. La comunicación entre la vista y el viewmodel se realiza por medio de los enlaces de datos (binders).

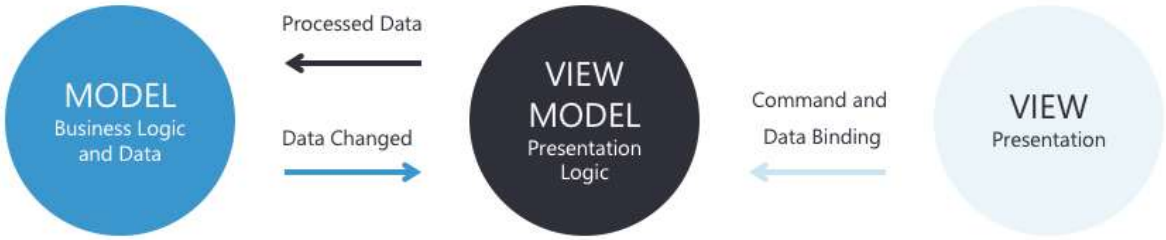

Figura 5.4: Esquema Modelo-Vista-VistaModelo

MVVM tiene un beneficio adicional: la simplificación que resulta de usar *binding* declarativo para transferir los datos desde y hacia el modelo a la vista.

Este patrón MVVM fue incluido en el desarrollo del TG para beneficiarse del uso de los data bindings y poder desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación. Permitiendo que el front-end se pueda desarrollar casi en paralelo con el back-end (esto lo veremos más en detalle en la sección 7.5).

Beneficios del MVVM:

Como mencionamos arriba MVVM permite un gran flujo de trabajo desarrollador-diseñador, proporcionando estos beneficios [16]:

- Durante el proceso de desarrollo, desarrolladores y diseñadores pueden trabajar de forma más independiente y en simultáneo en sus componentes. El diseñador puede concentrarse en la vista y los desarrolladores en el view model y el modelo.
- Los desarrolladores pueden crear test unitarios para el view model y el modelo sin utilizar la vista.
- Es más fácil rediseñar o modificar la interfaz de usuario de la aplicación sin necesidad de tocar el código.

5.1.3.3. MVC vs MVVM

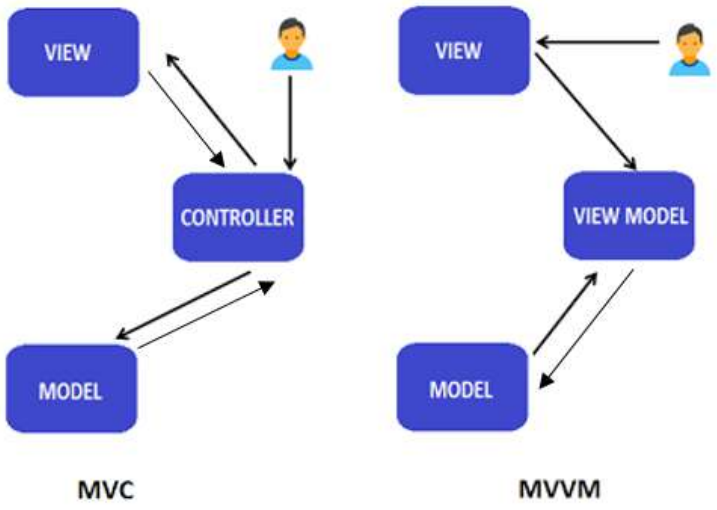

Figura 5.5: MVC vs MVVM

MVC	MVVM
Las entradas son dirigidas al controlador	La entrada es dirigida a la Vista
Un Controlador puede hacer muchas vistas diferentes dependiendo de la operación	Un ModelView puede soportar diferentes Vistas.
La Vista no tiene ningún conocimiento acerca del Controlador	La VistaModelo no tiene conocimiento de su Vista
La Vista no es consciente del Modelo.	La Vista no es consciente del Model. El ViewModel actualiza la Vista

5.2. Servicios Web

Un servicio web (en inglés, web service) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. [17]

Entre algunos de los estándares más conocidos nos encontramos con:

- **XML** (Extensible Markup Language): formato estándar para los datos que se vayan a intercambiar
- **SOAP** (Simple Object Access Protocol): protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

- **WSDL** (Web Services Description Language): es el lenguaje de la interfaz pública para los servicios web.
- **UDDI** (Universal Description, Discovery and Integration): protocolo para publicar la información de los servicios web.
- **Otros protocolos**: los datos también pueden enviarse de una aplicación a otra mediante protocolos normales como Hypertext Transfer Protocol (**HTTP**), File Transfer Protocol (**FTP**), o Simple Mail Transfer Protocol (**SMTP**).

5.2.1. Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares puedan ser combinados fácilmente para proveer servicios integrados.

Los Servicios Web albergan muchos tipos diferentes de sistemas, pero el caso común de uso se refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP.

Sin embargo, en los últimos años se ha popularizado un estilo de arquitectura Software conocido como **REST** (Representational State Transfer). Este nuevo estilo ha supuesto una nueva opción de uso de los Servicios Web.

Para el proyecto del Tablero de Gestión se optó por implementar un Servicio Web REST a través de la plataforma ASP.NET Web API. El principal motivo de esta decisión fue el de generar un servicio que aporte interoperabilidad entre aplicaciones y en un futuro poder proveer servicios integrados. Si en un futuro se desea compartir datos con otras aplicaciones programadas en diferentes plataformas, el servicio REST podría ser consumido desde diversas fuentes sin inconvenientes, lo que le da un valor agregado al software.

A continuación describiremos algunas características de la arquitectura REST como Servicio Web.

5.3. REST

REST (Representational State Transfer) es una arquitectura de software para sistemas hipermedias distribuidos tales como la Web, es una forma de proporcionar interoperabilidad entre los sistemas informáticos en Internet. El término se originó en el año 2000 por Roy Fielding, el padre de la especificación HTTP, y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo. [18]

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en un sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) sin las abstracciones adicionales de los protocolos de intercambio de mensajes, como SOAP.

5.3.1. Características de REST

- **Protocolo cliente/servidor sin estado**: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.
- **Conjunto de operaciones bien definidas** que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son **POST** (crear), **GET** (leer), **PUT** (editar) y **DELETE** (eliminar), que con frecuencia se equiparan a las operaciones CRUD[3] en bases de datos.
- **Los objetos en REST siempre se manipulan a partir de la URI**[4]. Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. cada recurso es direccionable únicamente a través de su URI.
- **Interfaz uniforme**: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- **El uso de hipermedios**, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

5.3.2. Ventajas que ofrece REST

1. **Separación entre el cliente y el servidor**: el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.
2. **Visibilidad, fiabilidad y escalabilidad**. La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las

[3] CRUD: acrónimo de Create, Read, Update y Delete, funciones básicas en bases de datos
[4] URI: Identificador de recursos uniforme, del inglés uniform resource identifier

peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

3. **La API REST siempre es independiente del tipo de plataformas o lenguajes**: la API REST se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.

5.3.3. Ejemplo de Petición REST

El siguiente ejemplo muestra una consulta de los usuarios registrados en un sistema. Se realiza una petición GET, la URI es http://laravel.app/user y el formato de intercambio de datos es JSON.

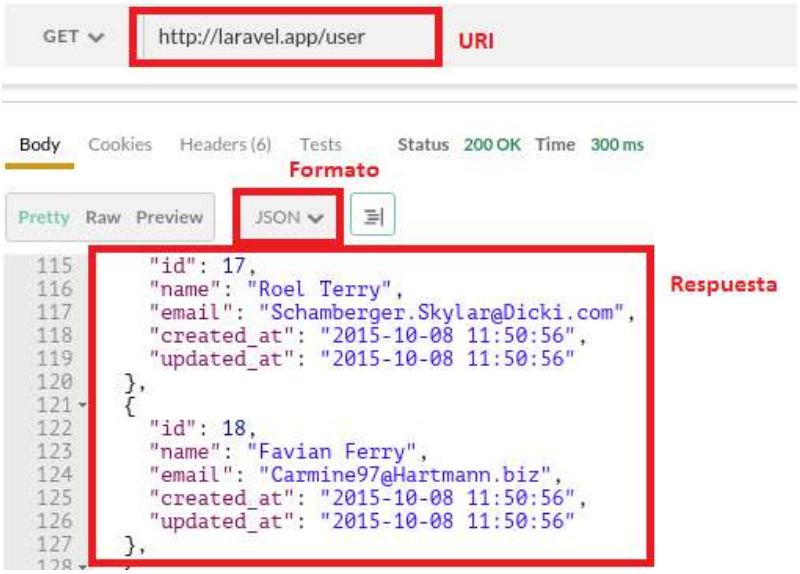

Figura 5.6: Ejemplo Petición REST

Como mencionamos en la sección 5.2.1, para el presente proyecto se implementó un servicio REST para realizar el intercambio de datos entre el back-end y el front-end, aportando independencia entre cliente y servidor, y mejorando la escalabilidad y portabilidad.

5.4. Diseño de Interfaz de Usuario

El diseño de la interfaz de usuario (UI) y la experiencia de usuario (UX) [19], no es un punto menor en el diseño del sistema, ya que un tablero de gestión debe mostrar información de manera que sea lo más fácil de comprender y utilizar para el usuario. Se pretende que la

información se comunique y aprenda lo más rápidamente posible, a través de recursos como los gráficos, pictogramas, estereotipos y simbología, todo ello sin afectar el funcionamiento del sistema.

Al no contar en el equipo de trabajo con una persona avocada al diseño gráfico, ni al diseño de Interfaces de Usuario, se optó por utilizar un Template (plantilla) para luego modificar en base a las necesidades. Para ello, se utilizó una plantilla de Panel de Control Open Source denominada AdminLTE[5], que se adaptó en base a las necesidades para la implementación del tablero de control (o dashboard).

El Template AdminLTE, está construido sobre Bootstrap 3, y provee una amplia gama de componentes web responsive[6], reutilizables y de uso general.

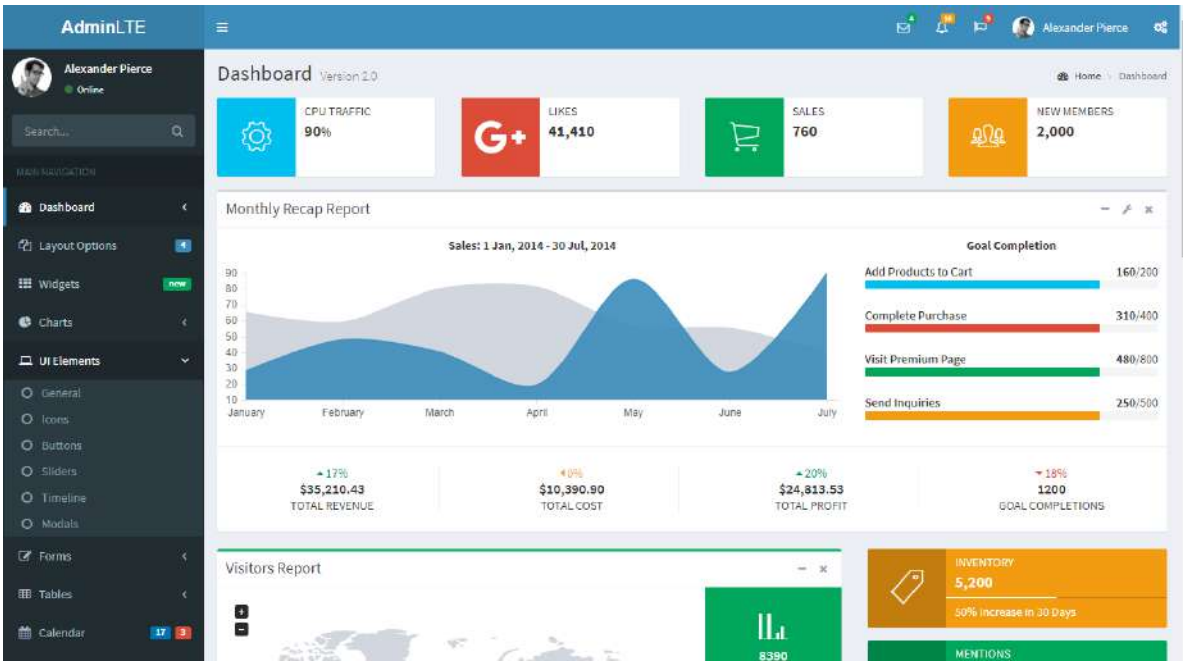

Figura 5.7: Template AdminLTE

[5] https://adminlte.io/

[6] Diseño web adaptativo o responsive, éste último del inglés responsive web design

6. Capítulo VI: Tecnologías y Herramientas de desarrollo

El presente Capítulo tiene como objetivo describir las tecnologías y herramientas utilizadas a lo largo del desarrollo del proyecto.

6.1. Tecnologías Involucradas

6.1.1. Microsoft ASP.NET

ASP.NET es un entorno para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime (CRL), permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework, entre ellos Microsoft Visual Basic, C#, JScript .NET y J#.

Modelos de programación en ASP.NET [20]

Actualmente, ASP.NET soporta tres modelos de programación para Sitios Web: ASP.NET Web Forms, ASP.NET MVC y ASP.NET Web Pages; y dos más para Servicios Web: ASP.NET Web API y SignalR, como se muestra en la siguiente figura:

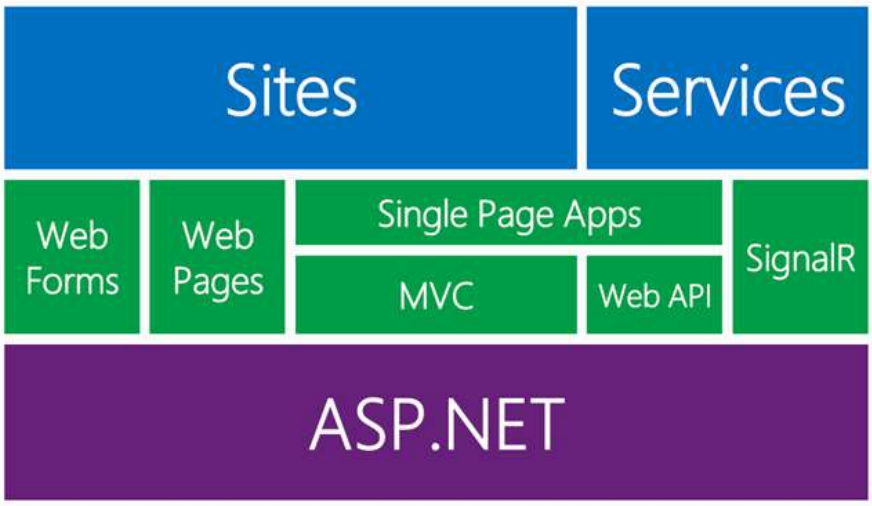

Figura 6.1: Microsoft ASP.NET

ASP.NET para Sitios Web:

Aunque los tres modelos de programación se ejecutan sobre la misma base de ASP.NET, cada uno de ellos estructura la aplicación de maneras completamente distintas, promueve metodologías de desarrollo diferentes y se adapta a perfiles de desarrolladores distintos. Algunas características que son virtudes en unos modelos de programación, pueden ser

consideradas debilidades en el otro. ¿Qué es más importante, desarrollar a un gran nivel de abstracción o tener control total cada uno de los aspectos de la aplicación? Simplicidad vs. Control. Flexibilidad vs. Eficiencia.

Es importante recalcar que el hecho de elegir uno de los modelos de programación al comenzar un proyecto de ASP.NET no excluye necesariamente a los otros, sino que es posible tener aplicaciones "híbridas".

- **ASP.NET Web Forms** fue el primero de los tres modelos de programación en existir, y proporciona un gran nivel de abstracción con un modelo de programación familiar basado en eventos y controles que favorece la productividad mediante la programación declarativa reduciendo la cantidad de código necesaria para implementar una determinada funcionalidad.
- **ASP.NET MVC** se concibió como alternativa a Web Forms y proporciona un modelo de programación basado en el popular patrón de arquitectura MVC. Entre sus principales características destacan su completa integración con pruebas unitarias y su separación más clara entre la lógica de presentación, la lógica de negocio y la lógica de acceso a datos.
- **ASP.NET Web Pages** es el más reciente de los tres modelos de programación, y fue creado como respuesta a una creciente demanda de desarrolladores web sin experiencia previa con ASP.NET, cuya iniciación en ASP.NET Web Forms o MVC les suponía una inversión inicial de tiempo demasiado grande. Web Pages proporciona un modelo de programación más simple y rápida de aprender, sin renunciar a toda la funcionalidad y flexibilidad de ASP.NET.

ASP.NET para Web Services

- **ASP.NET Web API** es un framework que hace más fácil construir servicios HTTP que lleguen a una amplia variedad de clientes, incluyendo navegadores y dispositivos móviles. Web API es una plataforma ideal para desarrollar aplicaciones RESTful sobre el framework ASP.NET
- **ASP.NET SignalR** es una nueva librería para desarrolladores de ASP.NET que facilita el desarrollo de funcionalidades para aplicaciones en tiempo real.

6.1.2. ADO.NET

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework. ADO.NET ofrece componentes para la creación de aplicaciones de uso compartido de datos distribuidas. Constituye una parte integral de .NET Framework y proporciona acceso a datos relacionales, XML y de aplicaciones.

Entity Framework (EF)

Entity Framework es un framework ORM (Object Relational Mapping) de código abierto para ADO.NET.

ADO.NET Entity Framework

ADO.NET Entity Framework permite a los desarrolladores crear aplicaciones de acceso a datos programando con un modelo de aplicaciones conceptuales en lugar de programar directamente con un esquema de almacenamiento relacional

ADO.NET Entity Data Model (EDM)

Entity Data Model (EDM) es un conjunto de conceptos que describen la estructura de los datos, independientemente del formato en el que estén almacenados. EDM se basa en el modelo entidad-relación (Entity-Relationship Model). Cabe destacar que EDM proporciona herramientas para ayudar a compilar aplicaciones de Entity Framework

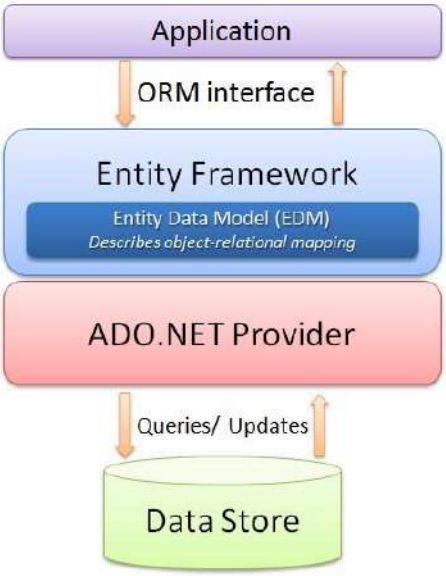

Figura 6.2: ADO.NET Entity Framework

6.1.3. C# (C Sharp)

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. C# corre en el Lenguaje Común en Tiempo de Ejecución (CLR, Common Language Runtime). Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, éste es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

6.1.4. LINQ (Language-Integrated Query)

LINQ es el nombre para un conjunto de tecnologías basadas en la integración de de las capacidades de consulta a datos de manera nativa para lenguajes de .NET.

Tradicionalmente, las consultas contra datos son expresadas como cadenas simples sin verificación de tipo en tiempo de compilación o soporte de IntelliSense[7].

Además, se debería utilizar un lenguaje de consulta diferente para cada tipo de origen de los datos: bases de datos SQL, documentos XML, WebService, etc. Con LINQ se ha adoptado un enfoque más general y se agregan facilidades de consulta de propósito general .NET que se aplican a todas las fuentes de información.

6.1.5. JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor.

6.1.6. JQuery

JQuery[8] es una librería de JavaScript pequeña, rápida y con muchas funciones, que permite simplificar la manera de interactuar con los documentos HTML, manipular el DOM, manejar eventos, desarrollar animaciones y agregar interacciones con AJAX a páginas web. Ofreciendo versatilidad y extensibilidad, JQuery es la biblioteca de JavaScript más utilizada.

jQuery es software libre y de código abierto, licenciado bajo la Licencia MIT y GNU GPL v2, permitiendo su uso en proyectos libres y privados. jQuery ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, permite lograr buenos resultados en menor tiempo y espacio.

6.1.7. AngularJS

AngularJS[9] es un framework de JavaScript de código abierto, mantenido por Google, utilizado para el desarrollo Web Front End, que permite crear y mantener aplicaciones SPA Single Page Application. Tiene capacidad de implementar el patrón MVC (Modelo, Vista, Controlador) o MVVM (Modelo, Vista, Vista-Modelo) y permite extender el vocabulario HTML. El entorno resulta muy expresivo, legible y rápido para desarrollar.

El framework trabaja leyendo primero la página HTML, que tiene incorporados atributos de etiquetas personalizadas. Angular interpreta estos atributos como directivas para enlazar partes de entrada o salida de la página a un modelo que es representado por variables JS. Los valores de estas variables, pueden establecerse manualmente dentro del código o recuperados de recursos estáticos o dinámicos JSON.

[7] https://es.wikipedia.org/wiki/IntelliSense
[8] https://jquery.com/
[9] https://angularjs.org/

AngularJS está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software, mientras que la programación imperativa es excelente para expresar la lógica de negocio. Este framework adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un data binding bidireccional que permite la sincronización automática de modelos y vistas.

6.1.8. Apache Subversion (SVN)

Apache Subversion[10] (abreviado como SVN) es una herramienta de **control de versiones** open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con la que lo haría si estuviese en su disco duro local.

Subversion puede acceder al repositorio a través de la red, lo que le permite ser usado por personas que se encuentran en distintas computadoras, fomentando así la colaboración y evita tener un único conducto por el cual pasan todas las modificaciones.

6.1.9. SQL Server

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL(TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

6.1.10. Bootstrap 3

Bootstrap[11] es un framework o conjunto de herramientas de Código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML, CSS y extensiones de JavaScript.

Bootstrap es el framework más popular de HTML, CSS y JS para el desarrollo web responsive.

[10] https://subversion.apache.org/
[11] https://getbootstrap.com/

6.2. Herramientas Utilizadas

6.2.1. SQL Server 2014 Management Studio

SQL Server Management Studio (SSMS) es un entorno integrado para gestionar cualquier infraestructura SQL. SSMS es usado para acceder, configurar, manejar, administrar y desarrollar todos los componentes de SQL Server, Azure SQL Database, y SQL Data Warehouse. SSMS provee una integración que combina un amplio grupo de herramientas gráficas con editores de scripts para proporcionar acceso a SQL Server para desarrolladores y administradores de Bases de Datos.

6.2.2. Visual Studio 2015

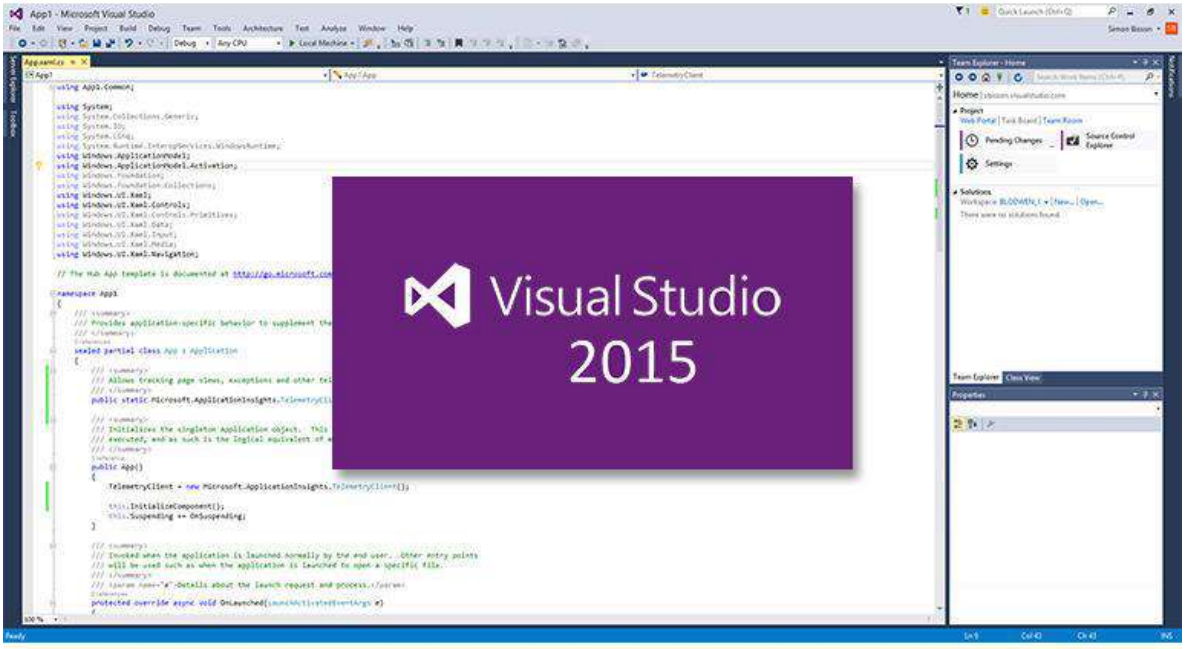

Figura 6.3: Visual Studio 2015

Es un entorno de desarrollo integrado (IDE) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET.

El IDE de Visual Studio presenta un conjunto de herramientas destinadas a ayudar a escribir y modificar el código para los programas, así como a detectar y corregir errores.

Visual Studio 2015 permite trabajar con los frameworks .NET Framework 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6 y 4.6.1.

6.2.3. TortoiseSVN

TortoiseSVN[12] es un cliente Subversion, implementado como una extensión al shell de Windows, que provee una agradable y fácil interfaz de usuario para SVN. Está desarrollado bajo licencia GNU GPL, Lo que significa que es totalmente gratuito para cualquier persona, incluso en un entorno comercial, sin ninguna restricción. Puede ser usado con la herramienta de desarrollo que se desee y con cualquier tipo de archivo.

6.2.4. Postman

Postman[13] está compuesto por diferentes herramientas y utilidades que permiten realizar diferentes tareas dentro del mundo de API REST, como creación de peticiones a APIs internas o de terceros, elaboración de tests para validar el comportamiento de APIs, posibilidad de crear entornos de trabajo diferentes (con variables globales y locales), y todo ello con la posibilidad de ser compartido con otros compañeros del equipo de manera gratuita (exportación de toda esta información mediante URL en formato JSON).

6.2.5. VisualSVN

VisualSVN[14] es un cliente de Apache Subversion, implementado como una extensión de paquete VS para Microsoft Visual Studio, que proporciona una interfaz para realizar las operaciones de control de versiones más comunes directamente desde dentro del IDE de Visual Studio.

[12] https://tortoisesvn.net/
[13] https://www.getpostman.com/
[14] https://www.visualsvn.com/

7. Capítulo VII: Construcción e Implementación del Tablero de Gestión

El Capítulo presente describe aspectos relacionados con la implementación del sistema, teniendo en cuenta el versionado de código utilizado, las funcionalidades desarrolladas, las páginas y accesos de usuarios, la estructura del código fuente y especificando el desarrollo del lado del cliente como del lado del servidor con las tecnologías planteadas y su integración.

7.1. Control de Versiones

Como se nombró en la sección 6.1.8, para el versionado del código fuente se utilizó Apache SVN (Subversion) porque el repositorio utilizado por Secretaría de Sistemas utiliza esta tecnología. Como el tablero de gestión se integró con el proyecto del SIGeLP, sólo se necesitó contar con los accesos al repositorio existente para comenzar a subir revisiones.

Además, como el desarrollo se realizó en computadoras con sistema operativo Windows, como cliente de SVN se utilizó TortoiseSVN (6.2.3) y VisualSVN (6.2.5) para poder realizar las operaciones de control de versiones directamente desde el IDE utilizado que fue Visual Studio 2014.

Cabe destacar que es indispensable utilizar una herramienta de control de versiones en cualquier desarrollo de software que se realice, ya sea un proyecto pequeño, mediano o de gran envergadura. Estos sistemas facilitan la administración de las distintas versiones del producto desarrollado, y crean un entorno de trabajo colaborativo, aumentando la productividad del equipo a la hora de integrar versiones y cambios en el código fuente.

A continuación se muestran imágenes de cómo se integra TortoiseSVN con Windows y VisualSVN con VisualStudio, y se muestra un ejemplo de un commit realizado al repositorio.

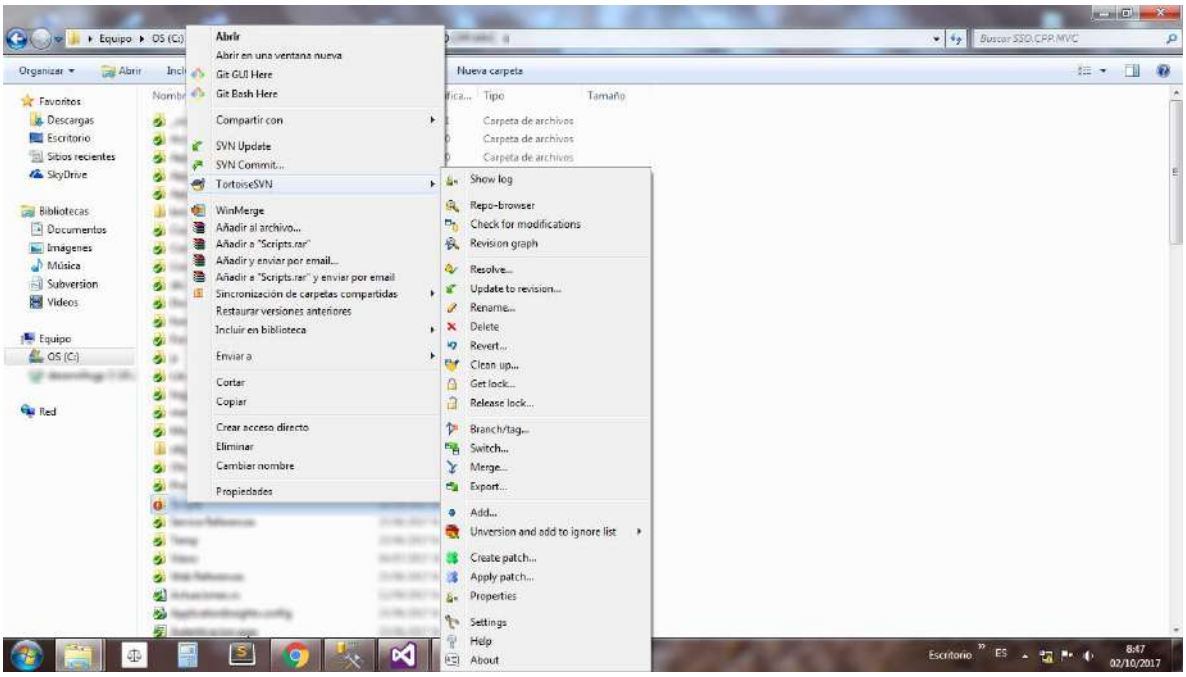

Figura 7.1: Captura de pantalla Tortoise SVN

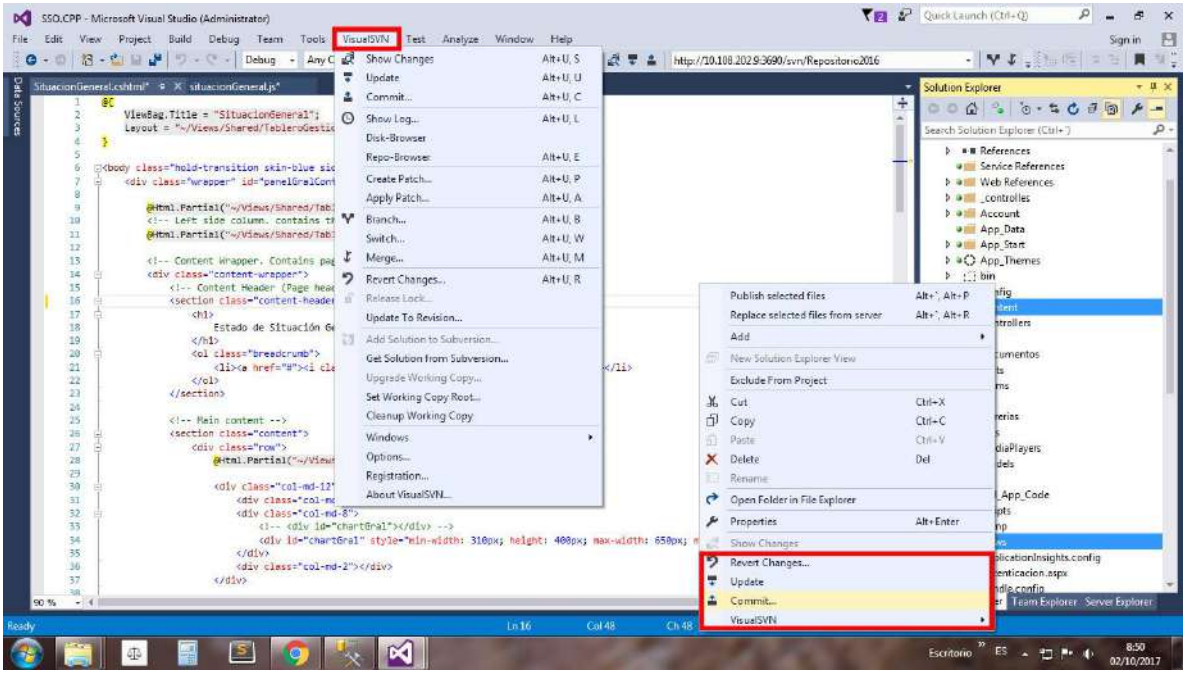

Figura 7.2: Captura de pantalla Visual SVN

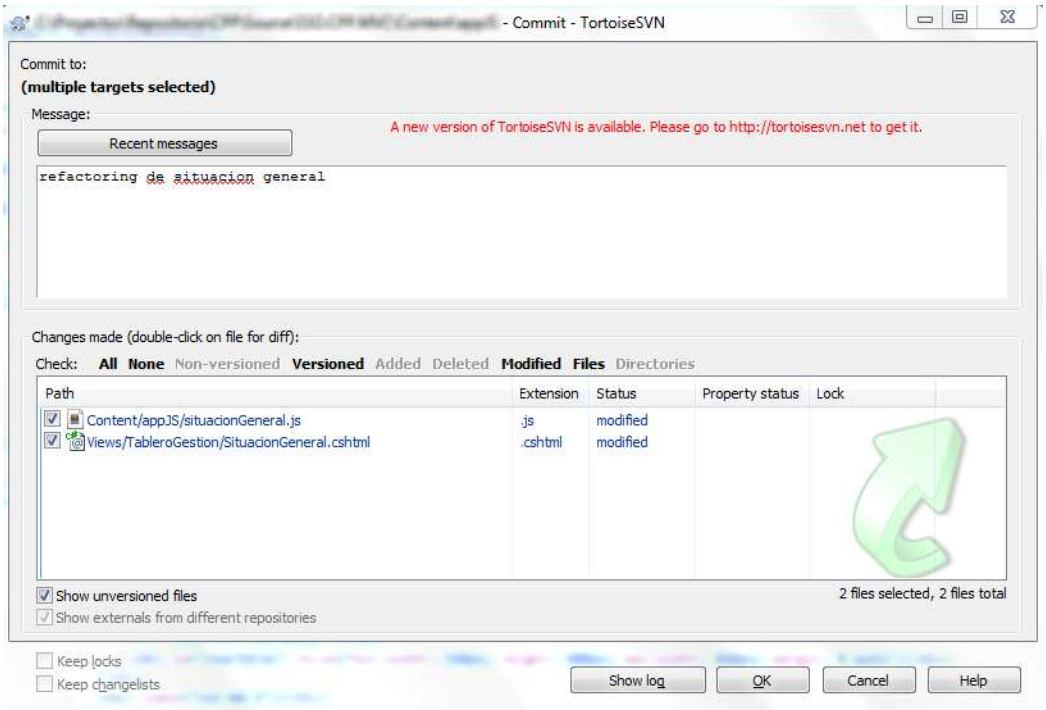

Figura 7.3: Ejemplo de Commit TortoiseSVN

7.2. Funcionalidades de la primera etapa

En base a los requerimientos funcionales de la primera etapa, descritos en la sección 4.3, el desarrolló se dividió en 3 módulos principales:

Nota: Los valores mostrados en las imágenes siguientes son meramente ilustrativos, no reflejando la realidad.

- **Estado de Situación General**:

Figura 7.4: Pantalla Estado de Situación General

Esta pantalla permite visualizar a través de un gráfico y una tabla, la cantidad de legajos en Trámite de cada Circunscripción y la cantidad de legajos Terminados por Archivo, Sentencia o Depuración (TEXAS), completando así el total de legajos que maneja cada sede y la provincia.

El gráfico permite visualizar en términos porcentuales la relación legajos en trámite vs TEXAS y la tabla expone información en valores absolutos comparativamente por sede y total provincial.

Este estado de situación general, se puede filtrar por fecha de creación de legajos y así poder hacer distintas lecturas y obtener la información necesaria de acuerdo al período elegido.

- **Estado de Situación UAP:**

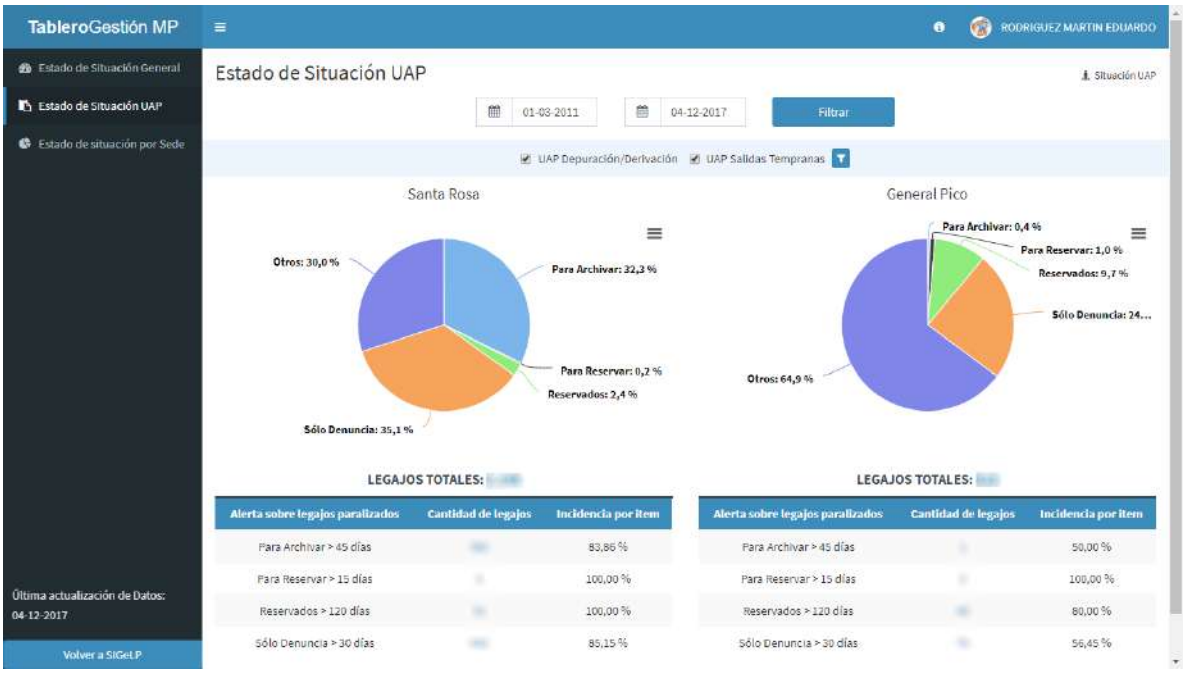

Figura 7.5: Pantalla Estado de Situación UAP

Esta pantalla muestra los legajos que se encuentran activos “En Trámite” en la Unidad de Atención Primaria (UAP) diferenciados en 4 categorías:

- Para Archivar: legajos con estado “para archivar”, que tienen pendiente la adopción de una decisión (efectivizar el archivo).
- Para Reservar: legajos con estado “para reservar”, que tienen pendiente la adopción de una decisión (efectivizar la reserva y/o archivo).
- Reservados: legajos reservados. Tanto como para el SIGeLP como para el TG, los legajos “reservados” son considerados como activos, es decir “en trámite”.
- Solo denuncia: legajos cuya única actuación es la denuncia.
- Otros: categoría residual.

A simple vista se puede observar la distribución de legajos que gestiona la UAP en las sedes principales de Santa Rosa y General Pico.

La tabla de Legajos Paralizados, permite visualizar alertas de legajos en situaciones no deseadas. Se expone la cantidad de legajos en términos absolutos que se encuentran dentro de la alerta definida y la incidencia por ítem, que da información representativa de la proporción de legajos que se encuentran paralizados para la categoría analizada del total de esa categoría.

Desde esta pantalla se puede acceder al listado de los legajos para cada categoría.

- **Estado de Situación por Sede**:

Figura 7.6: Pantalla Estado de Situación por Sede

En el estado de situación por sede, se ve un desglose de los legajos en trámite de una sede en particular (Ej: Santa Rosa), permitiendo ver la cantidad total de legajos en trámite, y cuantos se encuentran en UAP y UTC.

La tabla permite visualizar las fiscalías temáticas de la sede en cuestión, y los fiscales que se encuentran activos en cada una de ellas, con la cantidad de legajos que tienen en trámite en la temática y fuera de ella.

Recordemos que las fiscalías temáticas fueron implementadas en el año 2015, por ende hay legajos que los fiscales tienen asignados que no están en la temática en la cual están trabajando, son legajos antiguos. Éstos legajos figuran como “legajos en trámite fuera de su temática”.

Cuando expandimos una temática y seleccionamos una cantidad de legajos en particular de un fiscal, pasamos a otra pantalla que tendrá el detalle del mismo y la clasificación de esos legajos.

  - <u>Situación por Fiscal</u>:

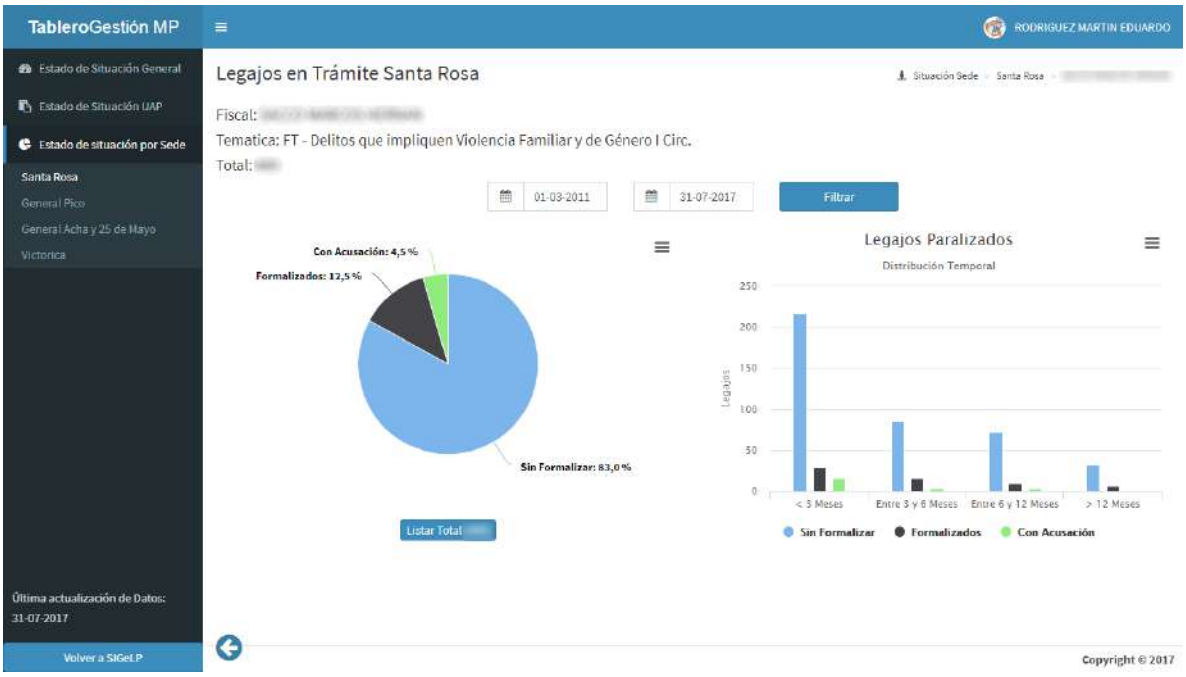

Figura 7.7: Pantalla Situación por Fiscal

En esta pantalla, se puede ver la clasificación y distribución de los legajos en trámite que tiene un determinado fiscal, que trabaja en determinada fiscalía temática.

Por un lado se expone un gráfico de torta con la clasificación de los legajos que tiene este fiscal. Las tres clasificaciones definidas para los legajos en trámite son:

- Sin formalizar: legajos en los que no se formalizó la IFP.
- Formalizados sin Acusación: legajos con formalización, pero sin acusación.
- Con acusación: legajos con acusación, pero sin resolución.

Haciendo clic en las porciones del gráfico, se puede acceder al listado de los legajos que componen cada una de las tres categorías.

Por otro lado, se puede ver un gráfico con información temporal sobre el tiempo que hace que estos legajos se encuentran sin movimiento, es decir “paralizados”.

Este gráfico de distribución temporal de legajos paralizados también está diferenciado en las tres categorías mencionadas, permitiendo tener información de gran relevancia y poder ir desglosando aún más la situación que se desee analizar (Por ejemplo: ver los legajos en trámite del fiscal XX de la temática YY, sin formalizar, paralizados hace más de 12 meses). Desde este gráfico también se permite ir al listado correspondiente.

  - Listado de Legajos:

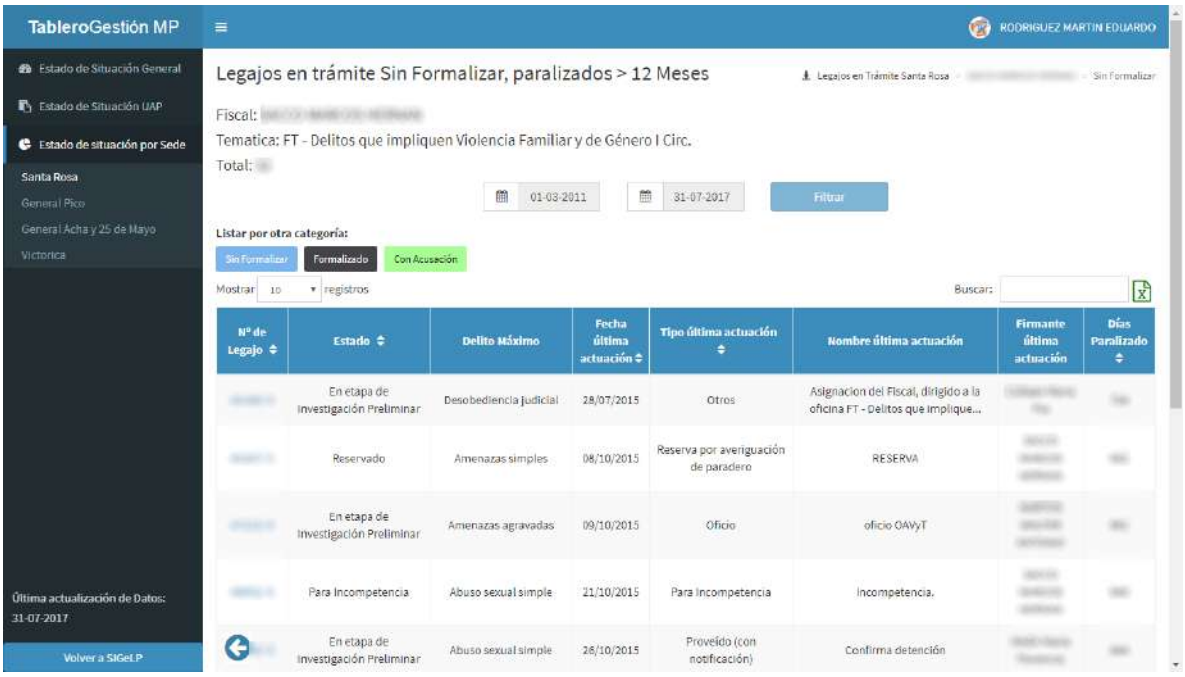

Figura 7.8: Pantalla listado de Legajos

Ahora sí llegamos a la pantalla de mayor desagregación del tablero de gestión, que es el listado de legajos que se encuentran en la situación a la que se fue llegando a través de los filtros y clasificaciones anteriores.

El listado de los legajos permite ver los datos básicos de cada legajo como (Nro de legajo, Estado, Delito Máximo, Fecha de última actuación, Tipo de última actuación, días paralizados), se puede realizar una búsqueda dentro del listado por texto por alguno de estos campos y permite reordenar listados por cualquier campo que se desee.

Además, cabe destacar que los listados son exportables en formato Excel o PDF para su posterior análisis con otra herramienta y/o impresión.

Cada legajo del listado se puede acceder a través del nro de legajo y se va al detalle del mismo del SIGeLP, para poder ver ahí si todas sus actuaciones, personas involucradas en la causa y demás detalles.

7.3. Esquema de páginas y Accesos de Usuarios

El siguiente esquema muestra cómo fue planteado el sistema para cumplir con los requerimientos de la primera etapa y refleja las relaciones entre las páginas descriptas en la sección anterior (7.2).

Como se puede ver, dentro del sistema del tablero de gestión, se diferencian claramente los 3 módulos de Situación General, Situación por Sede y Situación UAP, y las páginas que se desprenden de ellas.

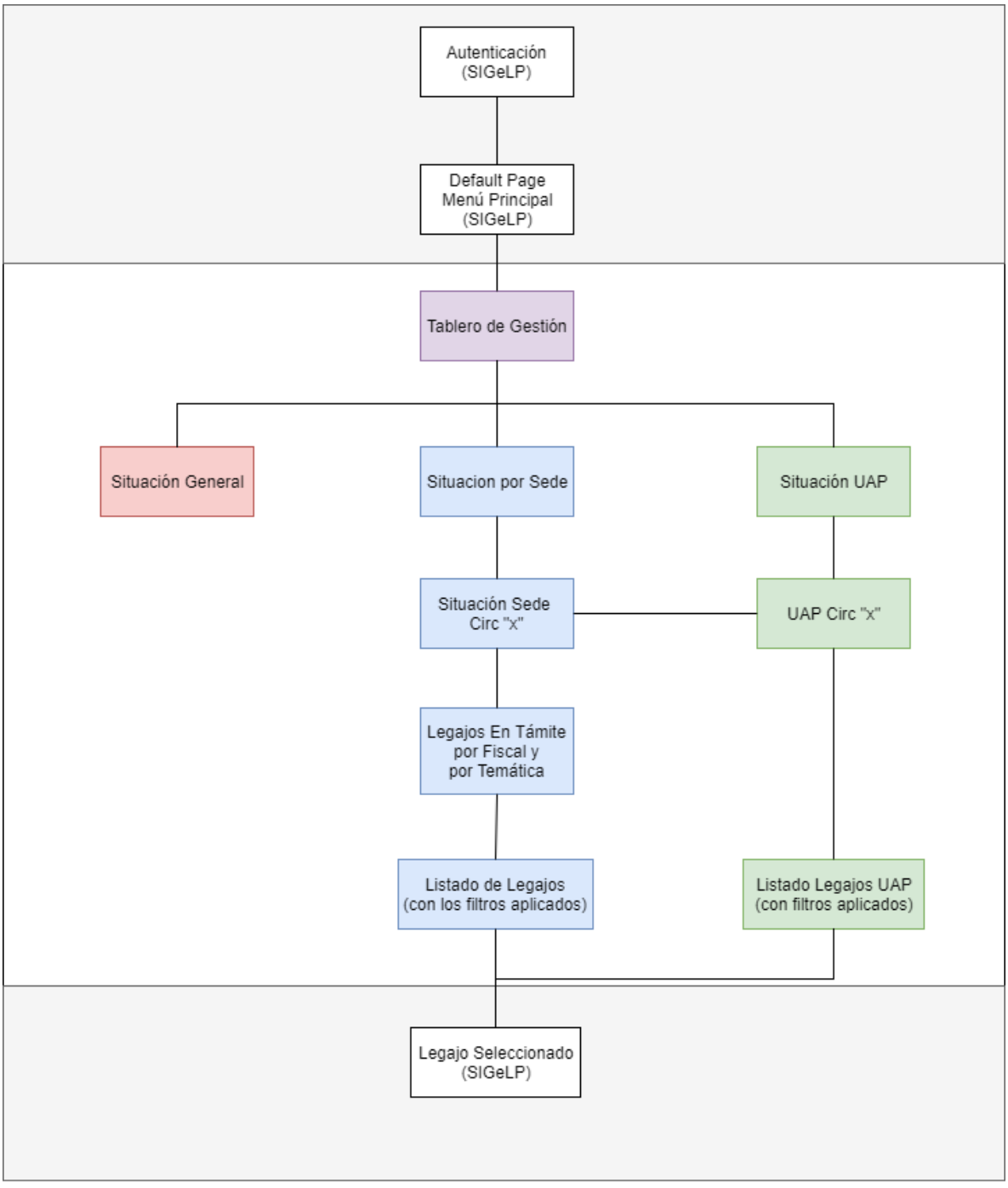

Figura 7.9: Esquema de módulos y páginas del Tablero de Gestión

Como vimos también en los requerimientos, el tablero de gestión, pretende ser de utilidad no sólo a nivel de la Procuración del Ministerio Púbico, sino que en etapas posteriores también sea una herramienta para los Fiscales Generales y los Fiscales. Por supuesto, el acceso a estos nuevos usuarios implica una restricción de las páginas accesibles por los mismos.

A continuación se muestra un esquema del acceso de los Fiscales Generales. La idea es que el Procurador tenga acceso completo y sin restricción al tablero de gestión, y a su vez los Fiscales Generales de cada Circunscripción tengan acceso a su sede y a la información relevante de la misma.

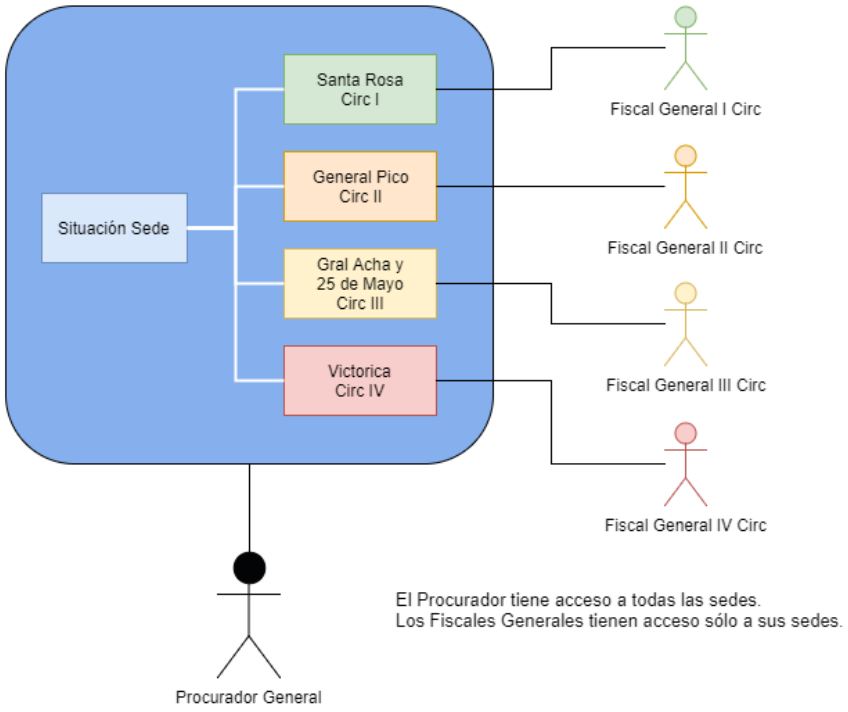

Figura 7.10: Esquema de Accesos de Usuarios

7.4. Estructura de carpetas del proyecto

Para la implementación se estructuró el proyecto conforme al Framework ASP.NET MVC, el cual contempla una estructura de organización predeterminada.

A continuación mostramos cómo queda el árbol de carpetas de un proyecto base MVC y qué contiene cada una [21]:

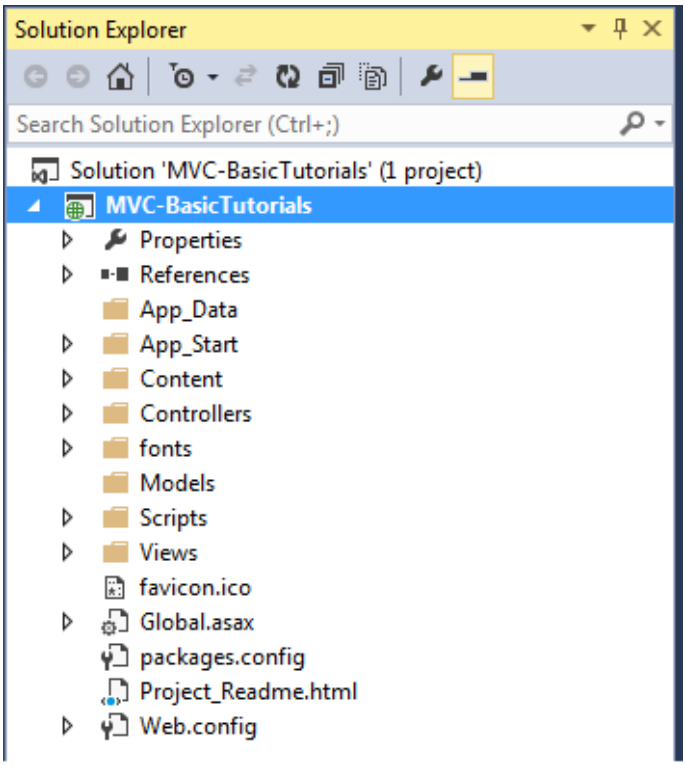

Figura 7.11: Estructura de Carpetas Proyecto MVC

- **App_Data**: puede contener archivos de datos de aplicaciones como LocalDB, archivos .mdf, archivos xml y otros archivos relacionados con datos.
- **App_Start**: contiene archivos de clase que se ejecutarán cuando se inicie la aplicación. Normalmente, estos serían archivos de configuración como AuthConfig.cs, BundleConfig.cs, FilterConfig.cs, RouteConfig.cs
- **Content**: contiene archivos estáticos como archivos css, imágenes e íconos.
- **Controllers**: contiene archivos de clase para los controladores. Los controladores manejan la solicitud de los usuarios y devuelven una respuesta. MVC requiere que el nombre de todos los archivos de controladores finalicen con "Controller".
- **Fonts**: contiene archivos de fuentes personalizados utilizados en la aplicación.
- **Models**: contiene archivos de clase para el modelo. Normalmente, las clases del modelo incluyen propiedades públicas, que serán utilizadas para la manipulación de datos de la aplicación. Por ejemplo los archivos DTO (Data Transfer Object)
- **Scripts**: contiene archivos JavaScript o VBScript para la aplicación.
- **Views**: contiene archivos html para la aplicación. Por lo general, el archivo de vista es extensión .cshtml donde se escribe html y código C# o VB.NET.
  La carpeta de Views incluye carpetas separadas para cada controlador.
  La carpeta Shared dentro de Views contiene todas las vistas que serán compartidas entre diferentes controladores, por ejemplo: archivos layout.

Adicionalmente, los proyectos MVC también incluyen los siguientes archivos de configuración:

- **Global.asax**: permite escribir código que se ejecuta en respuesta a eventos a nivel de aplicación, por ejemplo Application_BeginRequest, application_start, application_error, session_start, session_end etc.
- **Package.config**: este archivo es manejado por NuGet para realizar un seguimiento de qué paquetes y versiones están instaladas en la aplicación.
- **Web.config**: contiene configuraciones a nivel de aplicación.

7.5. Implementación con ASP.NET MVC Web API y AngularJS

Como comentamos en la sección 5, la aplicación fue desarrollada con ASP.NET MVC Web API y AngularJS. El proyecto involucró la combinación de varias tecnologías y un aspecto clave fue la integración de AngularJS (front-end) con ASP.NET MVC Web API (back-end).

En la sección 6.1.7 se describió brevemente la tecnología de AngularJS y en 5.1.3.2 el patrón arquitectónico empleado por la misma. También se introdujo el patrón MVC en 5.1.3.1 y la arquitectura REST en 5.3. Ahora, describiremos cómo interactúan estas tecnologías.

En el lado del servidor, tenemos el servicio REST implementado con ASP.NET Web API, que recibirá las peticiones HTTP de la aplicación a través de Ajax y responderá con los datos solicitados con un formato JSON a través del mismo medio HTTP.

Como Web API es compatible y se puede incluir dentro de ASP.NET MVC, tendremos los controladores (Controllers) que procesarán las peticiones a la API y los Modelos (Models) que representan los modelos de acceso a datos.

Del lado del cliente (front end) tenemos las Vistas (Views) con los data-binding y los controladores en JavaScript (JS) de AngularJS. Recordemos que los data-binding en AngularJS son la sincronización entre el modelo y la vista, ya que se trabaja sobre el patrón MVVM.

Los controladores de AngularJS tendrán una o más llamadas Ajax ($http.get …) que se comunicarán con la Web API y obtendrán y procesarán la respuesta otorgada por el servidor.

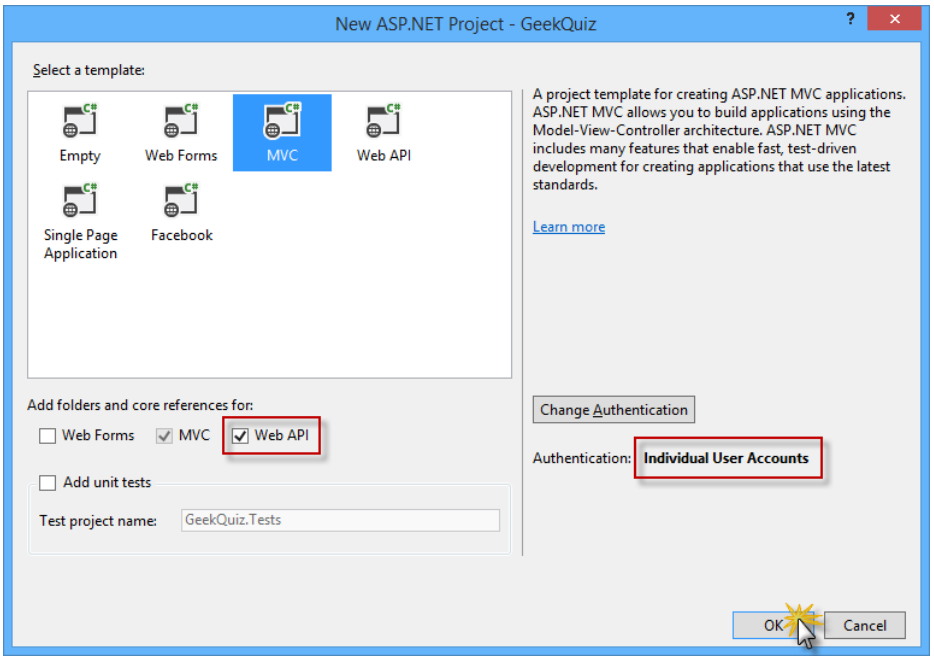

Figura 7.12: Creación de Proyecto MVC Web API

De forma resumida a lo dicho anteriormente la arquitectura a respondería al siguiente esquema:

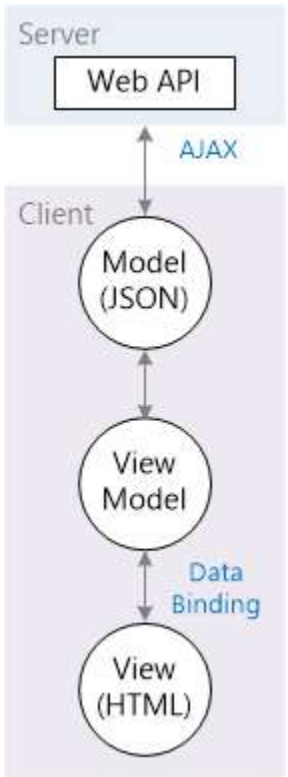

Figura 7.13: Arquitectura MVVM con Web API

En el esquema se puede ver claramente que la comunicación entre el cliente y el servidor es a través de llamadas AJAX, que intercambian datos formateados en JSON.

Como ejemplo de la implementación, se utilizó la creación de una página del tablero de gestión (Estado de Situación General). Este desarrollo del ejemplo permitirá al lector comprender mejor como se integran las tecnologías mencionadas y su funcionamiento base.

Los detalles de implementación serán omitidos por temas de confidencialidad, lo que se muestra es un ejemplo a modo ilustrativo de cómo funcionan las tecnologías.

7.5.1. Desarrollo de la Web API

Teniendo en cuenta que el proyecto MVC ya fue creado y el modelo de acceso a datos con ADO.NET también (ver punto 7.6), resta configurar la ruta de la WebApi y crear el controlador para la misma.

1. Configuramos la ruta de la WebAPI (AppStart > WebApiConfig)

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        // Web API routes
        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "SituacionGral",
            routeTemplate: "Legajos/{controller}/{action}/{fdesde}/{fhasta}"
        );
```

Figura 7.14: Configuración WebAPIConfig

2. Creamos el controlador en la carpeta Controllers

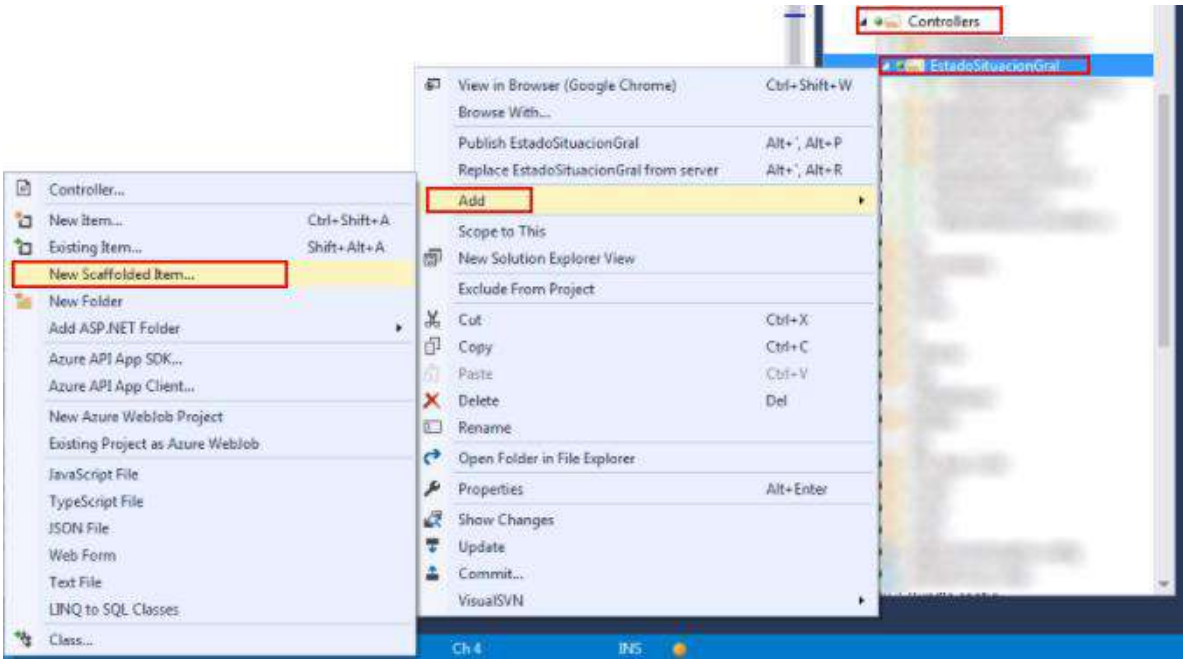

Figura 7.15: Creación de Controlrolador

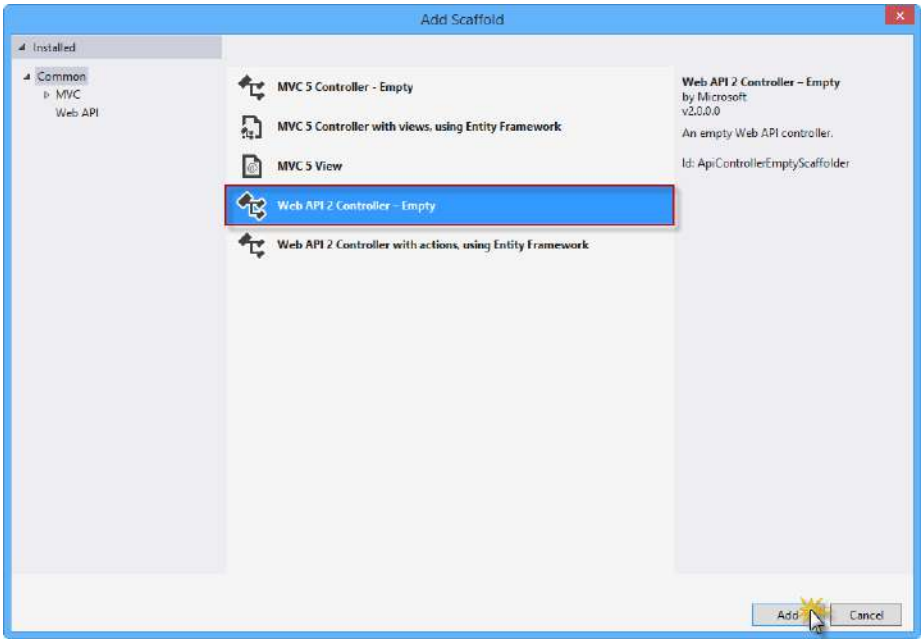

Figura 7.16: Crear Controlador Web API 2 Controller

Figura 7.17: Add Controller Web API

3. El controlador API que devuelve los legajos en trámite de cada circunscripción quedaría así:


```
namespace [redacted].Controllers.EstadoSituacionGral
{
    public class SituacionGralController : ApiController
    {
        private [redacted] db = new [redacted]
        private Legajo leg = new Legajo();

        [ActionName("totalEnTramite")]
        public IHttpActionResult GetTotalEnTramite(string fdesde, string fhasta)
        {
            DateTime d1 = Convert.ToDateTime(fdesde);
            DateTime d2 = Convert.ToDateTime(fhasta);

            var totalEnTramite = leg.obtenerLegEnTram();
            IQueryable<LegajoDTO> response = totalEnTramite
                .Where([redacted])
                .GroupBy([redacted])
                .Select(tab => new LegajoDTO
                {
                    Circ = tab.Key,
                    Clasificacion = "EnTramite",
                    Cantidad = tab.Count()
                });

            if (response == null)
                return NotFound();
            else
                return Ok(response);
        }
```

Figura 7.18: Extracto de Controlador Web API Situación General

Vemos que la ruta de la petición REST está definida por la configuración del WebApiConfig, el ActionName y los parámetros.

Al ser un WebService, la petición se puede probar a través del navegador colocando la ruta del Api REST o con una herramienta como Postman (6.2.4).

Ya tenemos configurada la ruta de la WebApi y el Controlador que será en encargado de procesar dicha petición y devolver los datos formateados en JSON. Ahora vamos a la implementación del Front End con AngularJS.

7.5.2. Desarrollo FrontEnd con AngularJS e integración con la WebApi

1. Creo la Vista que se encargará de mostrar los datos (en Views > TableroGestion):

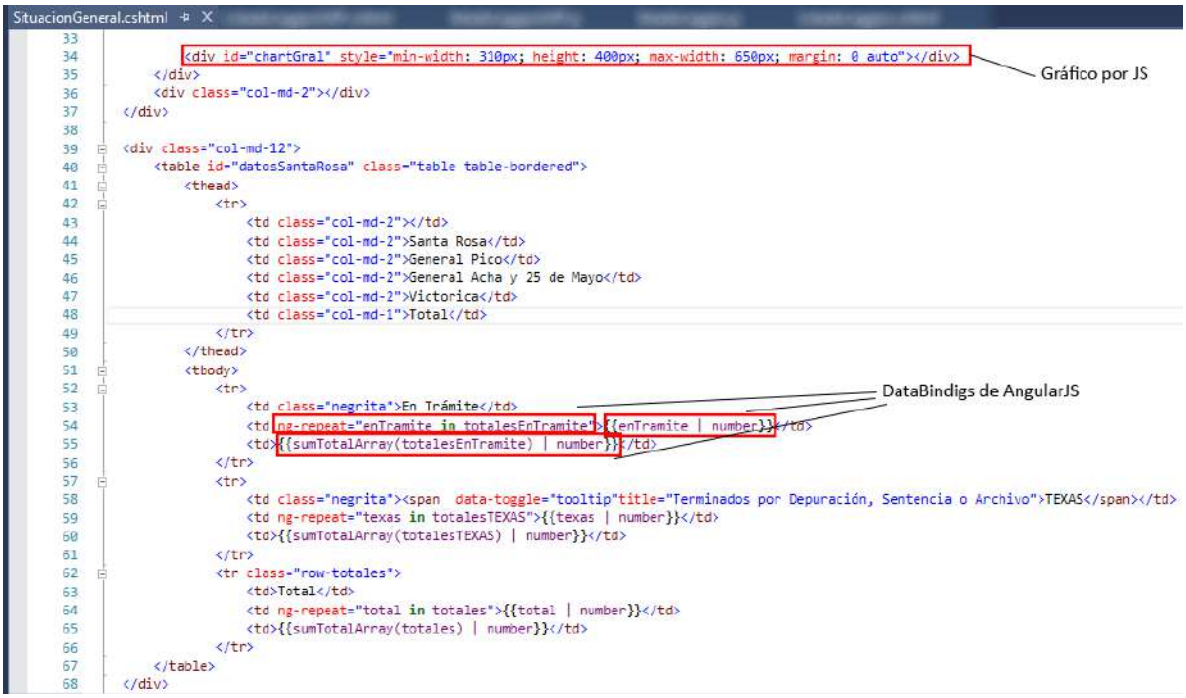

Figura 7.19: Extracto de Vista de página Situación General

Esta es una parte de la vista, donde podemos ver que armamos la tabla para mostrar los datos de SituaciónGeneral. En la parte de la interfaz gráfica, como mencionamos en 5.4 se utilizó el framework Bootstrap 3 y un Template base.

2. Creamos el controlador de JavaScripts que contendrá la lógica del frontEnd en JS utilizando Angular.

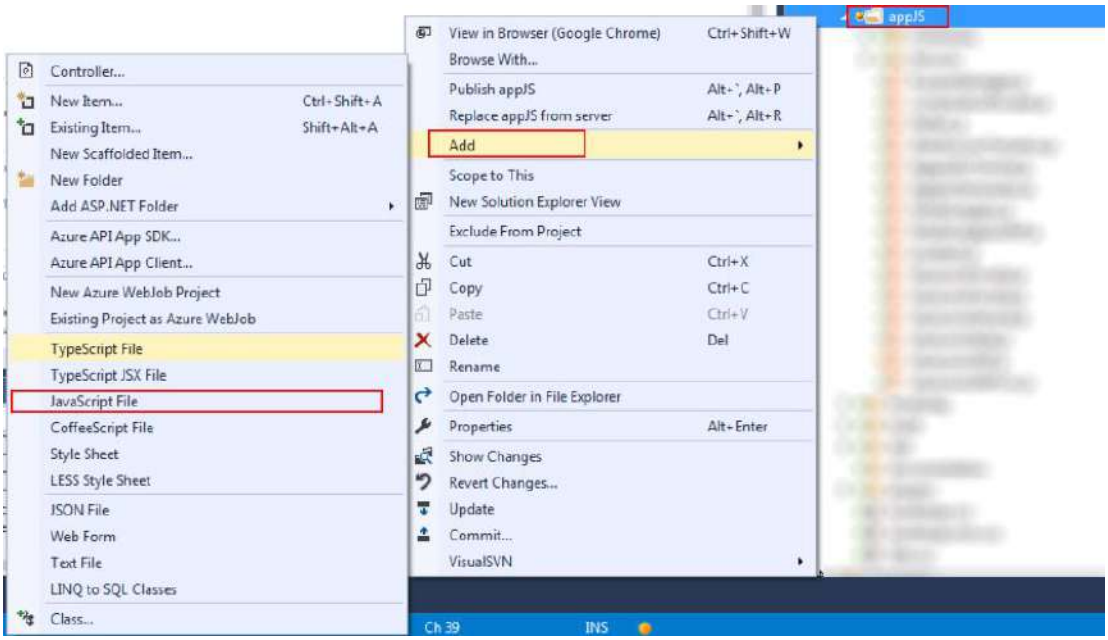

Figura 7.20: Creación de controlador JavaScript

3. Ahora, implementamos el Controlador de JS que hará el enlace entre el Cliente y la Web Api.

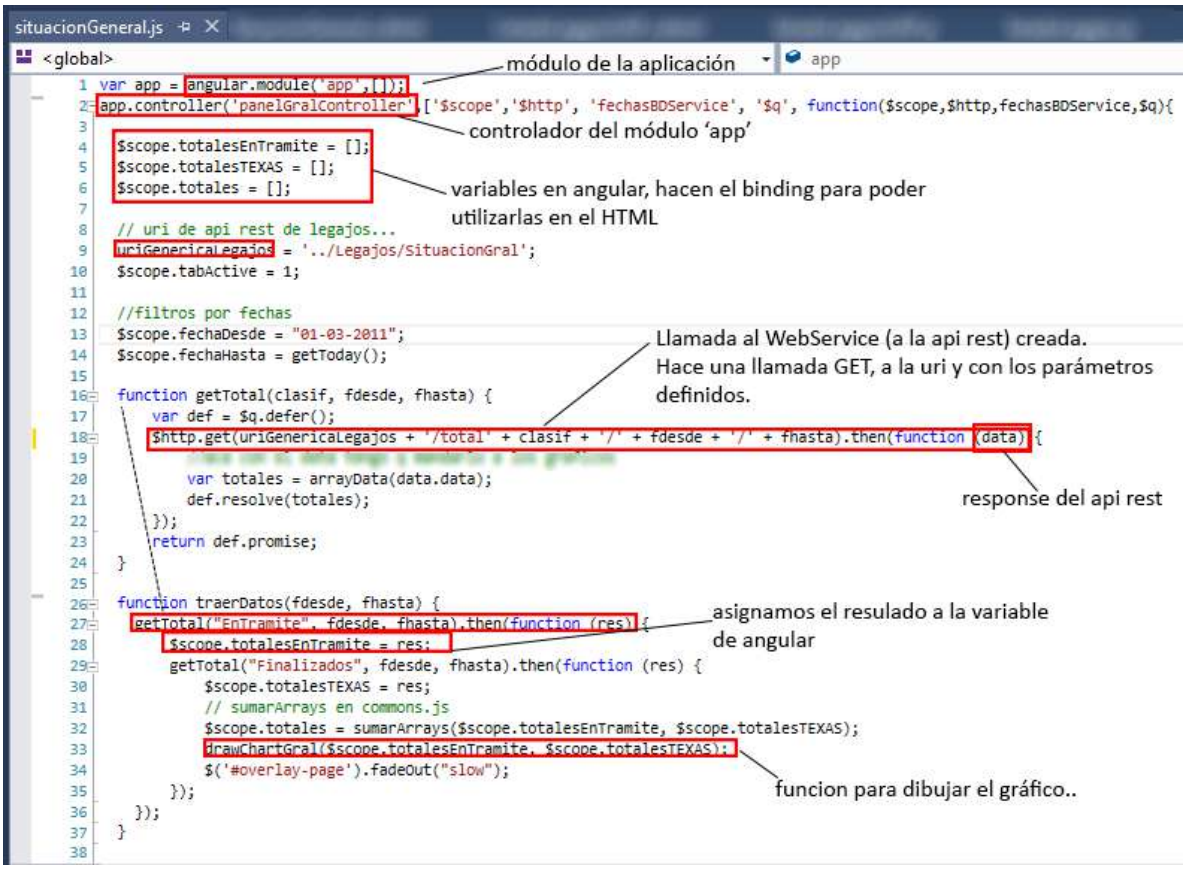

Figura 7.21: Controlador de angularJS de Situación General

Así queda el Controlador de JS, que se encarga de llamar a la WebApi y traer los datos necesarios para mostrar en la vista.

En la Figura 7.4 se puede ver cómo queda la página presentada como ejemplo y en la siguiente imagen se ve cómo queda la petición realizada por la misma para traer datos.

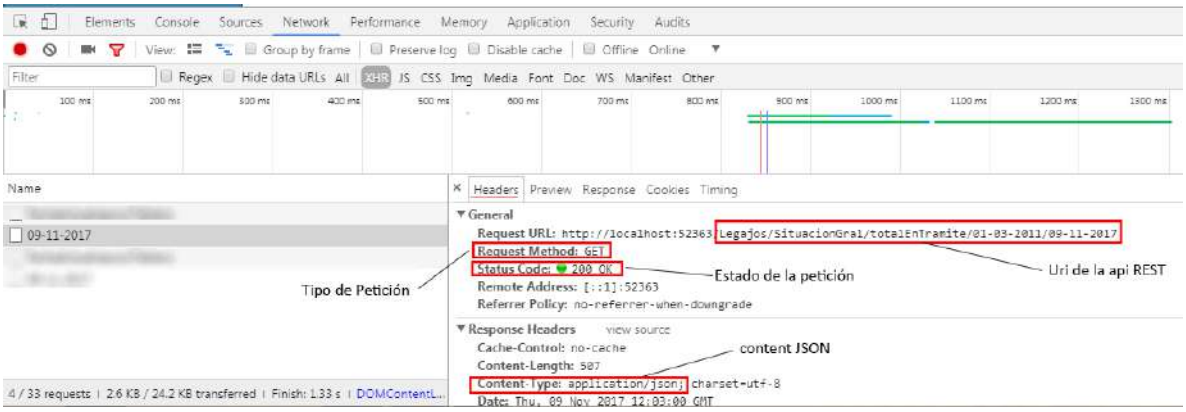

Figura 7.22: Ejemplo petición HTTP

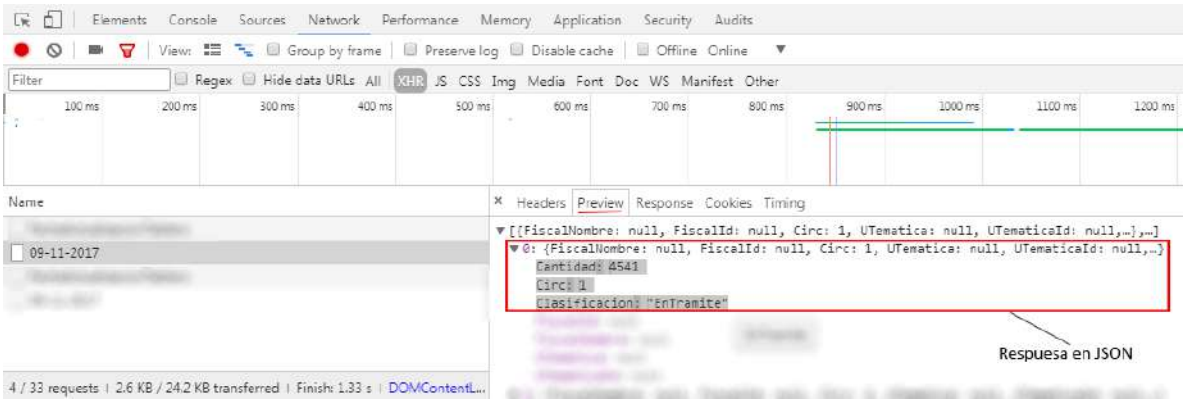

Figura 7.23: Ejemplo response petición HTTP

7.6. Acceso a Datos: ADO.NET Entity Framework

Para el modelado de datos (capa de acceso a datos) se utilizaron las herramientas proporcionadas por la plataforma .NET: ADO.NET Entity Framework y ADO.NET Entity Data Model. En 6.1.2 se presentaron brevemente estas tecnologías.

Para el desarrollo del TG se utilizó la versión 6.1.3 del Framework ORM Entity Framework (EF).

Herramientas de ADO.NET Entity Data Model

Las herramientas de Entity Data Model están diseñadas para ayudar a compilar aplicaciones de Entity Framework. Con estas herramientas se puede crear un *modelo conceptual* a partir de una base de datos existente. Después, puede visualizar y modificar gráficamente dicho modelo. O bien, primero puede crear gráficamente un modelo conceptual y, a continuación, generar una base de datos que admita su modelo. En cualquier caso, puede actualizar el modelo automáticamente cuando la base de datos subyacente cambie y generar automáticamente el código de capa de objeto para la aplicación. La generación de bases de datos y la generación del código de capa de objeto son personalizables. [22]

En la siguiente lista se describen las herramientas específicas que constituyen las herramientas de Entity Data Model:

- **ADO.NET Entity Data Model Designer (Entity Designer)** le permite crear y modificar visualmente entidades, asociaciones, asignaciones y relaciones de herencia.
- El **Asistente para Entity Data Model** le permite generar un modelo conceptual a partir de una base de datos existente y agrega información de conexión de la base de datos a la aplicación.
- El **Asistente para crear base de datos** le permite crear primero un modelo conceptual y, a continuación, crear una base de datos que lo admita.
- El **Asistente para actualizar modelo** le permite actualizar el modelo conceptual, el modelo de almacenamiento y las asignaciones cuando se efectúen cambios en la base de datos subyacente.

Las herramientas generan o modifican un archivo .**edmx**, que contiene información que describe el modelo conceptual, el modelo de almacenamiento y las asignaciones entre ellos.

A continuación mostramos de forma resumida cómo se crea el modelo conceptual a partir de una base de datos existente a través del Asistente para EDM.

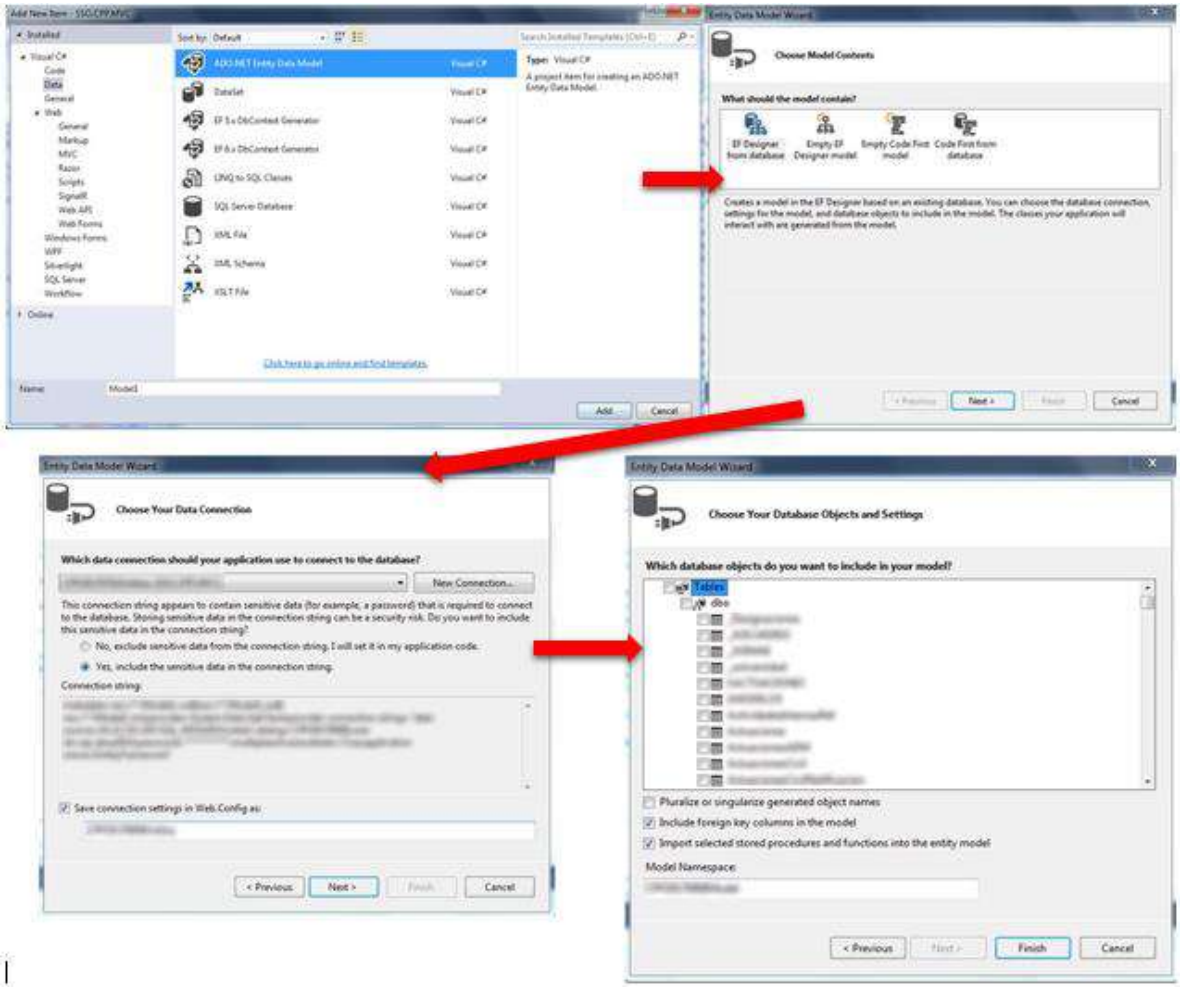

Figura 7.24: Creación de Modelo de Datos a partir de Base de Datos existente

Una vez creado el modelo conceptual se tiene una vista del mismo en el IDE como la siguiente:

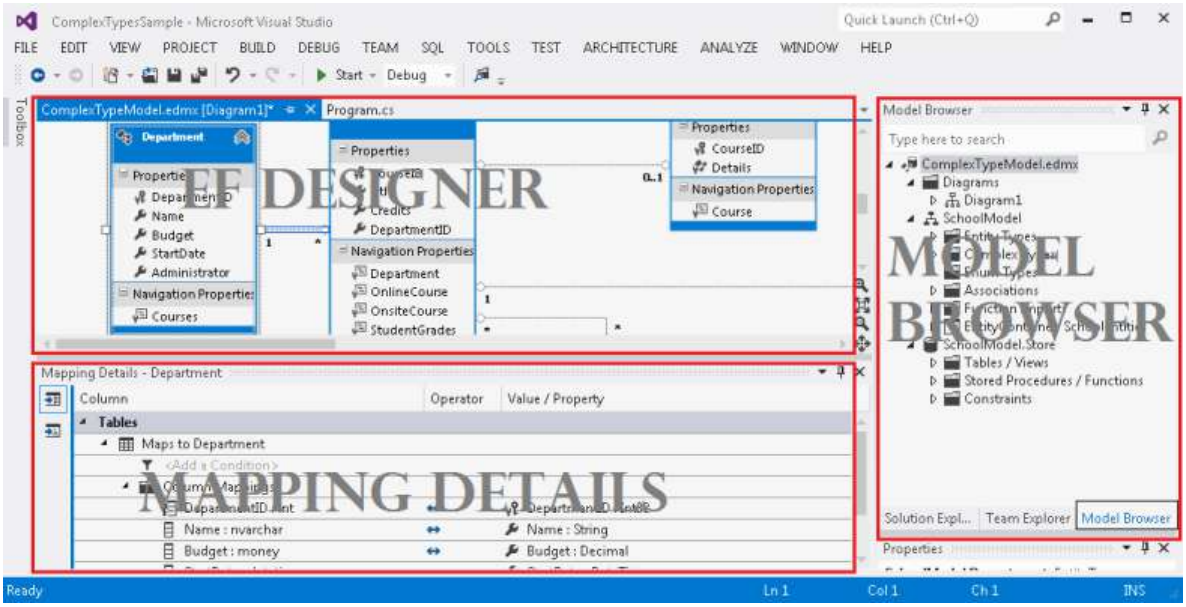

Figura 7.25: Vista del Modelo de Datos en Visual Studio

Cabe señalar que la imagen anterior no corresponde al modelo del proyecto ya que algunos detalles de la implementación se deben mantener privados.

7.7. Interacción de componentes y tecnologías

En el punto 7.5, se desarrolló un ejemplo de la implementación adoptada. Ahora, mostraremos un esquema resumiendo cómo interactúan los componentes desarrollados cuando comienza una petición de una página del TG.

La secuencia de interacciones es la siguiente:

1. Al ingresar desde el menú del SIGeLP al TG, la aplicación redirige al usuario al sitio principal del Tablero. Para ello actúa el ruteo MVC del proyecto, devolviendo la vista correspondiente y validando el perfil de usuario que intenta acceder.
2. Una vez que la vista se renderiza en el cliente, se carga tanto el HTML propio de la vista, los CSS (estilos), y el JavaScript que se encargará de realizar las peticiones al servidor para traer los datos necesarios.
3. Una vez cargado el HTML y el JS, por defecto se realiza una petición a la API REST para traer datos. Esta petición se realiza mediante AJAX desde el JavaScript (en este caso con AngularJS).
4. El controlador WebAPI encargado de procesar esa petición, la recibe y la procesa. En el procesamiento de la misma, el controlador interactúa con el modelo de datos para realizar la consulta necesaria para traer los datos.
5. El modelo de datos, como sabemos es el encargado de gestionar los accesos a dicha información (Entity Framework).

6. Una vez que el controlador tiene la respuesta de la petición, la envía por HTTP al cliente.
7. El JavaScript que realizó la petición a la WebApi ahora tiene los datos de la respuesta en formato JSON, y con ayuda de AngularJS y del modelo planteado en el controlador de js, los enlaza en la capa de presentación (es decir, en la Vista).
8. Al actualizar las variables de AngularJS la vista se refresca con estos datos y se obtiene la vista deseada con los datos requeridos.
9. Finalmente, el sitio queda a la espera de alguna nueva acción por parte del usuario que vuelva a desencadenar la secuencia de pasos para mostrar la información necesaria.

La imagen siguiente muestra la integración de las tecnologías utilizadas en el TG, tanto del lado del cliente como del lado del servidor:

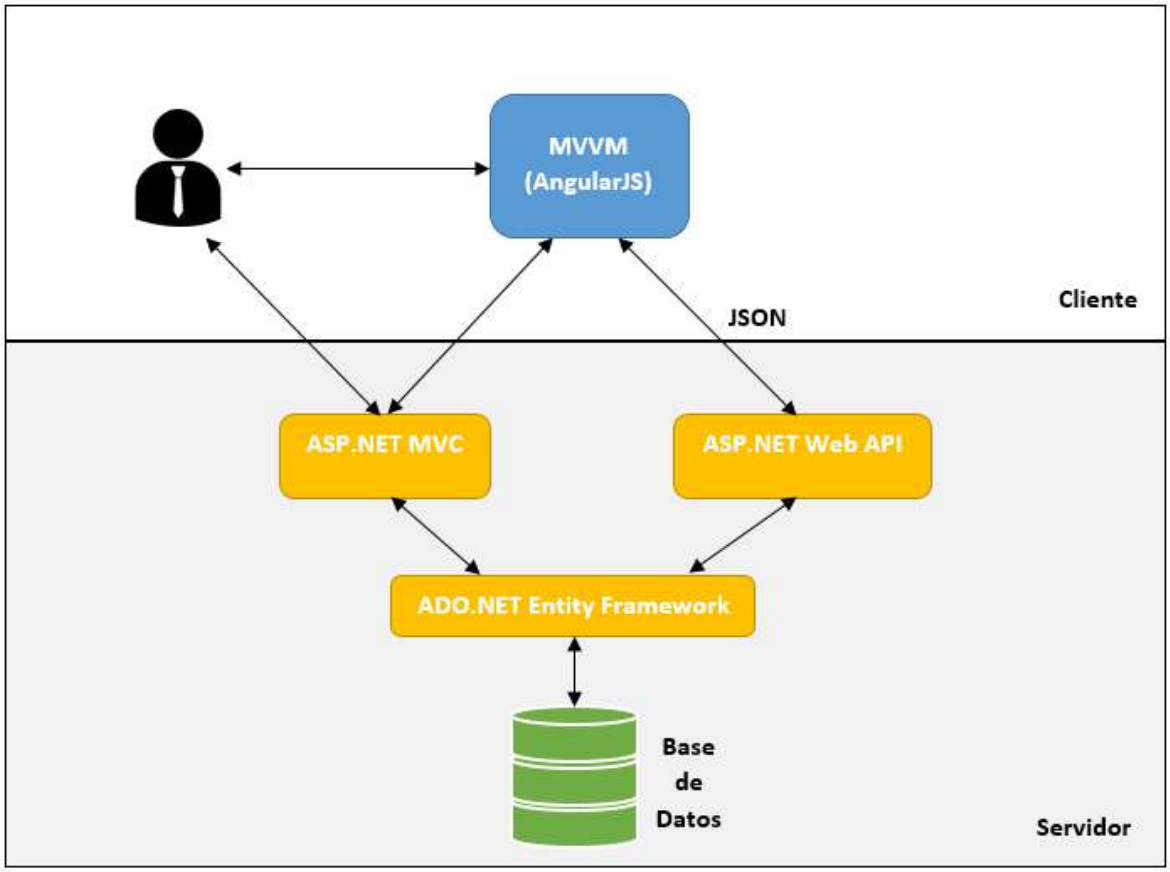

Figura 7.26: Integración de Tecnologías del TG

7.8. Integración del TG con SIGeLP

Cabe destacar que el Tablero de Gestión fue pensado para integrarse al sistema de gestión de expedientes penales SIGeLP, como una herramienta extra que proporciona información a determinados niveles de la organización.

El SIGeLP, está desarrollado sobre la plataforma ASP.NET y dentro de ésta en ASP.NET Web Forms. Como se indicó en la sección 6.1.1, ASP.NET Web Forms es uno de los 3 modelos de programación que pueden usarse para crear aplicaciones web ASP.NET, los otros son ASP.NET MVC, ASP.NET Web Pages.

Web Forms fue el primero de los tres modelos en existir y utiliza un tipo de programación basado en controles y eventos. En Web Forms, las páginas solicitadas a través del navegador por los usuarios pueden ser escritas usando una combinación de HTML, scripts del lado del cliente, controles de servidor, y código de servidor. Cuando los usuarios solicitan una página, ésta es compilada y ejecutada en el servidor, y luego el framework genera el HTML que el navegador va a mostrar.

El Tablero de Gestión, como fuimos describiendo en el presente informe, fue desarrollado bajo el modelo ASP.NET MVC e implementando una Web API.

La siguiente imagen, muestra los diferentes modelos de programación de ASP.NET para creación de sitios y servicios. Cada uno de ellos tiene sus ventajas y desventajas y se puede utilizar uno en particular o una combinación de ellos en base a las necesidades y características del proyecto a desarrollar.

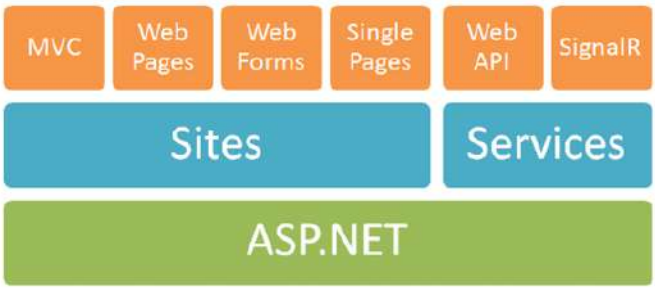

Figura 7.27: Modelos de Programación de ASP.NET

En nuestro caso podemos ver que las dos herramientas (SIGeLP y TG) fueron desarrolladas con diferentes modelos de programación, en base a las necesidades y características particulares de cada proyecto. Sin embargo, esto no fue un impedimento a la hora de integrarlos, ya que al ser desarrollados ambos sobre la plataforma de ASP.NET Framework, los proyectos pueden convivir sin tener significativos inconvenientes.

Los modelos desarrollados bajo la plataforma ASP.NET son compatibles, aunque posean diferencias a la hora de programar en ellas.

La siguiente imagen, muestra el acceso al tablero de gestión desde el Sistema de Gestión de Expedientes penales:

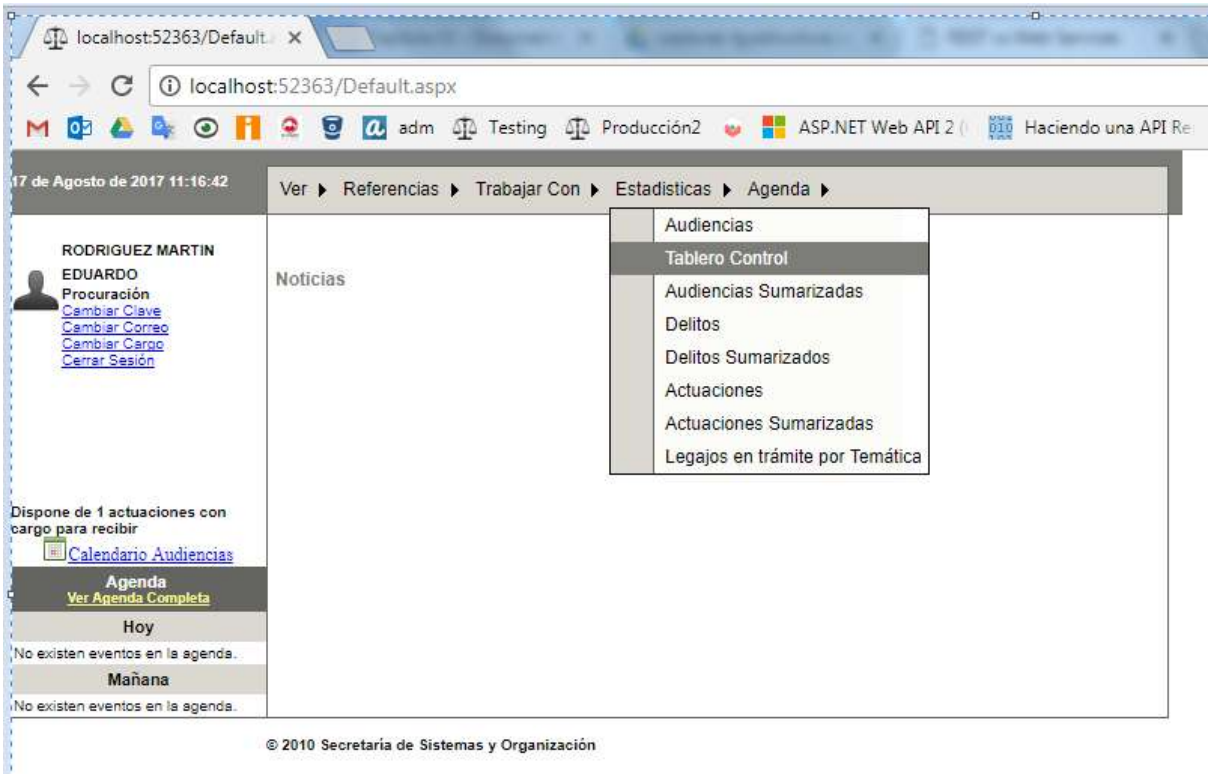

Figura 7.28: Menú del Sistema de Gestión de Expedientes Penales (SIGeLP)

Cabe destacar que para ingresar al Tablero de Gestión, se debe ingresar sí o sí a través del SIGeLP, con los usuarios y contraseñas asignados por el mismo a los usuarios.

Una vez en el SIGeLP, determinados perfiles de usuario pueden acceder al tablero desde el menú de Estadísticas > Tablero de Gestión.

La siguiente imagen muestra la interfaz del TG una vez ingresado desde el SIGeLP.

Figura 7.29: Interfaz del Tablero de Gestión

8. Capítulo VIII: Control de Calidad y Despliegue del Software

Si bien el presente proyecto no se centró en aspectos de control de calidad (QC) o aseguramiento de calidad (QA), el Capítulo pretende mostrar las prácticas básicas utilizadas en etapas de Testing y Mantenimiento para poder lograr una mejora del producto software desarrollado y así aportar a la calidad del mismo.

8.1. Testing

El Testing de software pertenece a las actividades de verificación y validación, que pretenden asegurar que el software respeta su especificación y satisface las necesidades de los usuarios. [23]

El Testing o pruebas de software, es una actividad desarrollada para evaluar la calidad del producto, y para mejorarlo al identificar defectos y problemas. El testing de software consiste en la verificación dinámica del comportamiento de un sistema sobre un conjunto de casos de prueba en relación con el comportamiento esperado.

Es una técnica dinámica en el sentido de que el programa se verifica poniéndolo en ejecución de la forma más parecida posible a como ejecutará cuando esté en producción – esto se contrapone a las técnicas estáticas las cuales se basan en analizar el código fuente.

8.1.1. Testing de Caja Blanca y Caja Negra

- **Testing Estructural o de Caja Blanca**

  Las pruebas de caja blanca se centran en los detalles procedimentales del software, por lo que el diseño de los casos de prueba está fuertemente ligado al código fuente. En las pruebas se escogen distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y asegurarse de que se devuelven los valores de salida adecuados.

  Como los casos de prueba se crean en base a la estructura del código fuente, no es posible generarlos hasta tanto no haya sido terminado el módulo a probar. Y, si la implementación cambia, debido a errores o cambios en las estructuras de datos o algoritmos, puede ser necesario volver a rediseñar los casos de prueba.

- **Testing Basado en Modelos o de Caja Negra**

  Es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software.

  En las pruebas de caja negra, nos enfocamos solamente en las entradas y salidas del sistema, sin preocuparnos en tener conocimiento de la estructura interna del programa

de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos en los requerimientos de software y especificaciones funcionales.

La distinción entre técnicas de pruebas de caja negra y pruebas de caja blanca es la clasificación clásica de las pruebas de software.

Para el Tablero de Gestión, la mayoría de las pruebas realizadas fueron pruebas de caja negra, que se centran en verificar la funcionalidad del mismo, sin la necesidad de diseñar casos de prueba en base al código, que puede llegar a modificarse.

8.1.2. Pruebas Funcionales y No Funcionales

Las **pruebas funcionales** se centran en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente. Se basan en funciones y su interoperabilidad con sistemas específicos, y puede llevarse a cabo en todos los niveles del Testing.

Las **pruebas no funcionales** incluyen pruebas de rendimiento, carga, estrés, usabilidad, entre otros requerimientos no funcionales definidos para el sistema. Se centran en características que debe cumplir el sistema y puede ejecutarse en todos los niveles de pruebas.

8.1.3. Niveles de Pruebas

Podemos considerar el proceso de pruebas funcionales como un proceso donde se va probando inicialmente lo de más bajo nivel y se van integrando y probando paulatinamente componentes hasta lograr un sistema completo totalmente probado. Por eso se dice que hay distintos niveles de prueba [24]. Se empieza por las Pruebas Unitarias, luego las Pruebas de Integración, las Pruebas del Sistema y finalmente las de Aceptación.

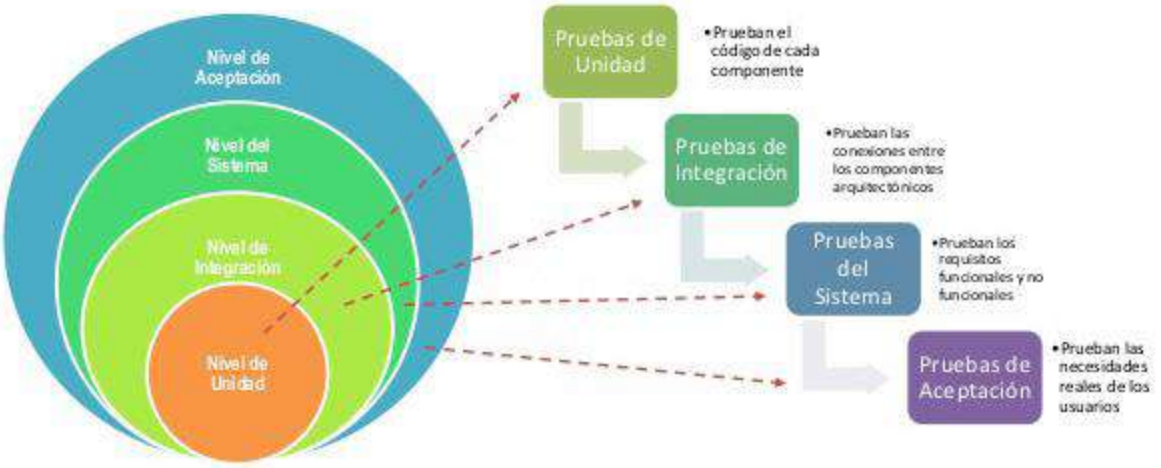

Figura 8.1: Niveles de Pruebas de Software

- **Pruebas Unitarias**: Se prueba cada componente tras su realización/construcción. Sólo se prueban componentes individuales y por lo general se producen con acceso al código bajo prueba y con el apoyo del entorno de desarrollo
- **Pruebas de Integración**: Verifica las interfaces entre componentes, interacciones entre diferentes partes de un sistema. Se realizan una vez que se han aprobado las pruebas unitarias y pretende probar que todos los componentes unitarios funcionan correctamente probándolos en grupo (a nivel de subsistema). Las estrategias pruebas de integración pueden estar basadas en la arquitectura del sistema.
- **Pruebas de Sistema**: Pueden incluir pruebas basadas en las especificaciones sobre los requerimientos, procesos de negocio, casos de uso, u otras descripciones de alto nivel del comportamiento del sistema, las interacciones con el sistema operativo y los recursos del sistema. Deben investigar requerimientos funcionales y no funcionales del sistema.
- **Pruebas de Aceptación**: Se verificará que el software satisface los requisitos del cliente. Son pruebas con respecto a las necesidades del usuario, requerimientos y procesos de negocio, realizadas para determinar si un sistema satisface los criterios de aceptación que permitan que el usuario, cliente u otra entidad autorizada pueda determinar si acepta o no el sistema. Encontrar defectos no es el foco principal en las pruebas de aceptación.

En base a los requerimientos planteados para el sistema del tablero de gestión, se realizaron las pruebas en los diferentes niveles, para poder llegar a un software probado íntegramente y que las pruebas de aceptación cumplan con los requisitos del cliente.

8.2. Gestión de Incidencias

Una vez que los usuarios comenzaron a utilizar el sistema en producción y también como resultado de la ejecución de pruebas, comenzaron a surgir errores, ajustes y modificaciones. Para la gestión de estos problemas denominados incidentes o issues (en inglés), se utilizó un sistema de seguimiento de incidentes o issue tracking system. Un issue tracker es una herramienta usada para realizar un seguimiento de requerimientos de desarrollo, errores (bugs) reportados y otras tareas de administración de proyectos surgidas durante el proceso de desarrollo.

Para el presente proyecto se utilizó el issue tracker de Bitbucket[15] de Atlassian Inc. Esta herramienta permite crear issues, asignando prioridades, tipo de incidente, asignarlo a una persona del proyecto y realizar comentarios y actualizaciones sobre el mismo.

A continuación mostramos algunas imágenes del funcionamiento del mismo:

[15] https://bitbucket.org/

Figura 8.2: Listado de Issues - Bitbucket

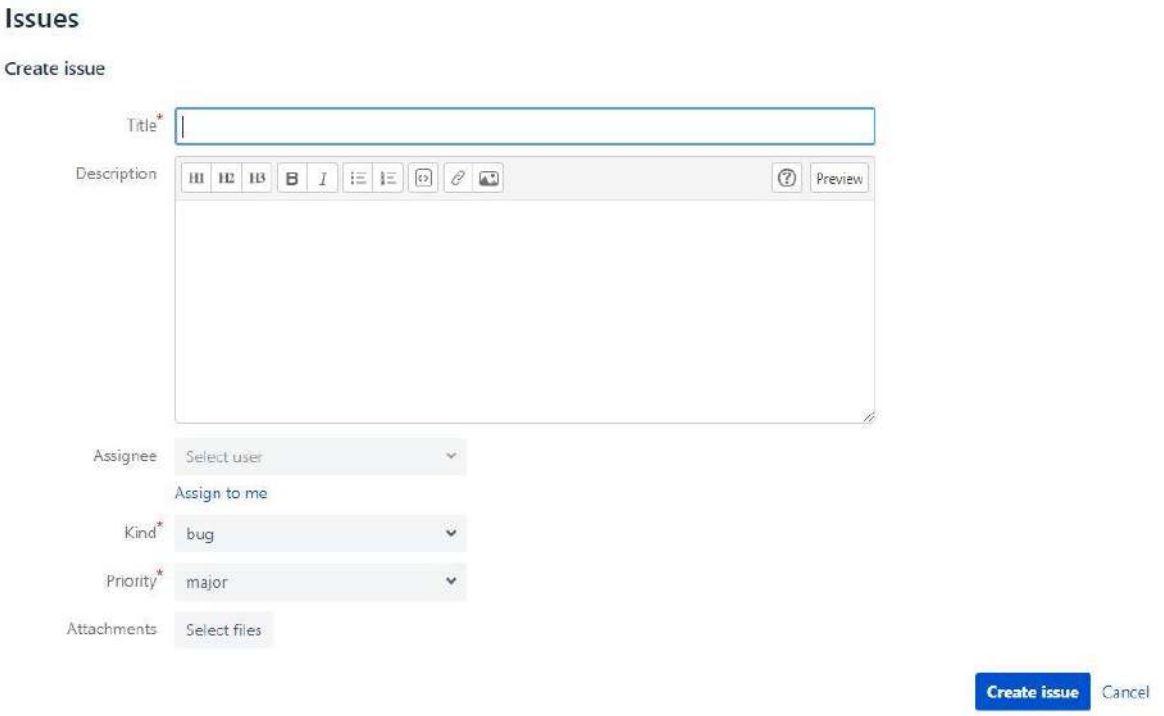

Figura 8.3: Creación de Issue - Bitbucket

Figura 8.4: Listado de Issues abiertos - Bitbucket

Además, como herramienta extra se utilizó una hoja de Cálculo de Google[16] compartida con el equipo de trabajo, donde los clientes iban anotando con sus palabras y según su prioridad, modificaciones y ajustes que iban surgiendo. Como sabemos las metodologías ágiles integran al cliente en el proyecto.

8.3. Refactoring de código fuente

El refactoring (o refactorización) es una técnica de la ingeniería de software para reestructurar código fuente, alterando su estructura interna, pero sin modificar su comportamiento externo. [25]

Esta práctica es muy utilizada en las metodologías ágiles, que tiene como objetivo remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar sus posteriores cambios y mantenimiento.

El refactoring es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad.

En este proyecto se dio mucha importancia a esta técnica para mejorar la estructura del código y facilitar su mantenimiento. A medida que se iban desarrollando nuevas funcionalidades y realizando modificaciones, se iba haciendo refactoring.

8.4. Integración Continua

La integración continua [26] es una práctica de desarrollo de software donde los miembros de un equipo realizan integraciones con frecuencia, generalmente cada persona integra al menos una vez al día, pudiendo haber múltiples integraciones en un día. Esto permite

[16] https://www.google.com/intl/es_ar/sheets/about/

detectar errores de integración tan pronto como sea posible. La integración puede incluir la compilación y ejecución de pruebas de todo un proyecto.

En proyecto como este que adopta la metodología ágil, la integración continua es uno de los pilares de la agilidad, asegurando que el sistema funciona en cada integración.

El proceso utilizado para la integración continua en nuestro caso fue, cada cierto tiempo (horas) hacer una actualización de los fuentes (update) del control de versiones SVN, realizar un merge en caso de haber conflicto con la versión local, compilarlo y ejecutar pruebas. De la misma manera cuando se realizan cambios locales se realiza el proceso para subir al servidor de control de versiones. En nuestro caso no se utilizó ninguna herramienta adicional que automatice estas tareas, pero en un futuro se podría considerar incorporar una.

8.5. Políticas de Publicación

En la publicación de sitios web en ASP.NET a través de Visual Studio, se compila el sitio y se copia el resultado en una ubicación especificada, como puede ser un servidor de testing o de producción. Al publicar, se llevan a cabo las siguientes tareas:

- Se compilan las páginas, el código fuente de la carpeta App_Code, etc., obteniendo un ejecutable.
- Se escribe el resultado ejecutable en una carpeta de destino.

Cuando publicamos, lo hacemos sobre un servidor Internet Information Server (IIS) que pueda ejecutar correctamente nuestra aplicación.

Al publicar un sitio web, se tienen las siguientes caracetísticas:

- Se precompila el código, en busca de cualquier error y evitando así que el código se publique con errores.
- La velocidad de respuesta inicial para las páginas individuales es más rápida porque ya están compiladas.
- Ningún código de programa se implementa con el sitio, lo que constituye una medida de seguridad para los archivos. Al publicar, quedan los binarios y no los fuentes.

Como sabemos el tablero de gestión se integró con el SIGeLP, el cual se encuentra alojado en los servidores de Secretaría de Sistemas y Organización de la Provincia.

Los servidores utilizados están centralizados en Santa Rosa y se posee un esquema como el siguiente para la publicación de versiones del sistema:

Cada desarrollador tiene su configuración y su entorno de desarrollo en su máquina, de manera local, donde realiza los cambios y pruebas pertinentes antes de subir revisiones al controlador de versiones.

Como dijimos antes, al realizar integración continua, todos los integrantes del equipo tienen el mismo código funcionando. Una vez que se han desarrollado determinados requerimientos o se han implementado cambios en el sistema, se pasa al entorno de Testing. En el servidor de testing, es el paso previo a lanzar el sistema en producción y es donde se pueden realizar todo tipo de pruebas para asegurarse que el software cumple con lo solicitado y no posee errores.

Una vez que esta versión que se encuentra en Testing es aprobada, se puede pasar al sitio de producción. En esta instancia, el software no debería tener errores, ya que aquí ya es utilizado por todos los usuarios del sistema y es crítico que un error llegue a esta etapa, ya que puede interferir con la forma de operar diaria del sistema en producción y puede generar conflictos en la organización.

Cabe destacar que las publicaciones del sitio son realizadas por el personal de la Secretaría de Sistemas en horarios fuera del horario de trabajo para que no se interfiera con la utilización del mismo. Las publicaciones no se realizan con una frecuencia fija predefinida, sino que depende de los cambios realizados y la necesidad de actualizar el sitio.

Conclusiones

En el presente informe, como trabajo final de la carrera de Ingeniería en Sistemas, se han desarrollado todos los aspectos teóricos y prácticos empleados en la creación de una herramienta informática denominada Tablero de Gestión para el Ministerio Público de la Provincia de La Pampa.

El objetivo planteado inicialmente en el punto 1.2 pudo cumplirse en su totalidad siguiendo una metodología ágil de desarrollo de software, con la cual se obtuvieron muy buenos resultados, basándose en las premisas de la metodología de: priorizar software que funciona, por encima de la documentación exhaustiva; la interacción constante con el cliente y el equipo de desarrollo; la importancia de construir un buen equipo y poder responder a los cambios a través de una planificación flexible y abierta.

A través de la utilización de este tipo de metodologías de desarrollo de software que son iterativas e incrementales, se obtuvo un producto a través de mejoras y entregas continuas, que satisface plenamente las necesidades del cliente, en un corto tiempo.

Cabe destacar que, como parte del Análisis de Sistemas, se requirió de un esfuerzo por parte del equipo de desarrollo para comprender e introducirse en un área desconocida como lo es el proceso penal, conceptos legales y nociones propias del dominio del área para el cual se realizó el desarrollo. Esto personalmente considero que es siempre un desafío por parte de los ingenieros en sistemas, que deben aventurarse en áreas en las cuales no son expertos para poder comprender el dominio y poder hacer un buen relevamiento y posterior desarrollo.

Otro desafío fue la integración del Tablero de Gestión con el Sistema de Gestión de Expedientes Penales (SIGeLP). Ambos sistemas fueron desarrollados bajo la plataforma de ASP.NET pero con diferentes modelos de programación. El SIGeLP como comentamos en 7.8 está desarrollado en ASP.NET Web Forms y el TG fue implementado con ASP.NET MVC y ASP.NET Web API. Sin embargo, ambos sistemas se pudieron compatibilizar y convivir dentro de un mismo proyecto.

Un punto no menor fue la implementación de una arquitectura híbrida para lograr una separación bien definida entre las tres capas del sistema (la capa de presentación, la capa de lógica de negocio y la capa de datos), logrando así un aumento en la escalabilidad y mantenibilidad del sistema.

En cuanto a la utilización de tecnologías para el desarrollo del sistema, debo mencionar que se utilizaron diversas tecnologías de desarrollo web actuales, como: ASP.NET, Entity Framework, JavaScript, JQuery, AngularJS, ApacheSVN, SQL Server, Bootstrap 3, entre otras. Este abanico de tecnologías combinadas, permiten obtener sistemas robustos, sin perder de foco la usabilidad de los mismos.

Gracias al desarrollo de esta herramienta, destinada a brindar información para gestionar el trabajo, diagnosticar una situación y tomar decisiones en base a datos, se pudo comprender y vivenciar la importancia de contar con este tipo de sistemas en organismos tanto públicos como privados, ya que proporcionan información fundamental para la gestión del mismo.

Por último, en relación al desarrollo del presente Proyecto Final de Grado, considero que éste ha sido un trabajo integral, que ha plasmado los conocimientos adquiridos a lo largo de la carrera y que ha aportado al desenvolvimiento como futuro profesional del área.

Referencias Bibliográficas

[1] K. E. Kendall y J. E. Kendall, *Análisis y Diseño de Sistemas*, 8va Ed. México: Pearson Education, 2011.

[2] «Ministerio Público Provincia de La Pampa. Sitio Web Institucional». [En línea]. Disponible en: http://www.mplapampa.gov.ar/. [Accedido: 27-jul-2017].

[3] «Procedimiento Penal en el Sistema Acusatorio de la Provincia de La Pampa». [En línea]. Disponible en: http://www.mplapampa.gov.ar/archivos/sitio/procedimiento_penal.pdf. [Accedido: 10-ago-2017].

[4] «Tablero de control - Wikipedia». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Tablero_de_control. [Accedido: 27-jul-2017].

[5] J. Fleitman, «LA IMPORTANCIA DE LOS TABLEROS DE CONTROL», 2010.

[6] J. M. Sabaté, «El cuadro de mando: un instrumento clave para la administración integrada», 2001.

[7] M. P. Fiscal, «Pre-proyecto Tablero de Gestión para el Ministerio Público». 2012.

[8] R. Vieytes, «Monitoreo, medición y comunicación de los estándares de calidad de los organismos adheridos al Programa Carta Compromiso».

[9] R. S. Pressman, *Ingeniería de Software: Un enfoque práctico*, 6ta Ed. Mc Graw Hill, 2005.

[10] «Metodologías ágiles de gestión de proyectos (Scrum, DSDM, Extreme Programming – XP)», 2008. [En línea]. Disponible en: https://www.marblestation.com/?p=661. [Accedido: 19-nov-2017].

[11] J. H. Canós, P. Letelier, y C. M. Penadés, «Metodologías Ágiles en el Desarrollo de Software», 2003.

[12] D. Zúñiga, «Estilo de arquitectura de llamada y retorno de en Prezi», 2016. [En línea]. Disponible en: https://prezi.com/gjtimc-be3c_/estilo-de-artquitectura-de-llamada-y-retorno/. [Accedido: 19-nov-2017].

[13] «Patrones de Arquitectura». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Patrones_de_arquitectura. [Accedido: 17-sep-2017].

[14] «Modelo–vista–controlador». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo–vista–controlador. [Accedido: 22-oct-2017].

[15] J. P. Sarco, «Una introducción simple al patrón Model View ViewModel para construir aplicaciones Silverlight y Windows Presentation Foundation». [En línea]. Disponible en: https://fernandomachadopiriz.com/2010/06/09/una-simple-introduccin-al-patrn-model-view-viewmodel-para-construir-aplicaciones-silverlight-

y-windows-presentation-foundation/. [Accedido: 22-oct-2017].

[16] «The MVVM Pattern», 2012. [En línea]. Disponible en: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Accedido: 19-nov-2017].

[17] «Servicio Web». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Servicio_web. [Accedido: 20-sep-2017].

[18] R. Navarro Marset, «Rest vs Web Service. Modelado, Diseño e Implementación de Servicios Web», 2006.

[19] J. Gothelf, *Learn UX: Applying Learn Principles to Improve User Experiences*. O'Reilly, 2013.

[20] D. Mossberg, «Modelos de programación en ASP.NET: Web Forms, MVC y Web Pages». [En línea]. Disponible en: https://blogs.msdn.microsoft.com/daniem/2012/05/10/modelos-de-programacin-en-asp-net-web-forms-mvc-y-web-pages/. [Accedido: 19-nov-2017].

[21] «MVC Folder Structure». [En línea]. Disponible en: http://www.tutorialsteacher.com/mvc/mvc-folder-structure. [Accedido: 19-nov-2017].

[22] «Herramientas de ADO.NET Entity Data Model». [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/bb399249(v=vs.100).aspx. [Accedido: 08-nov-2017].

[23] M. Cristiá, «Introducción al Testing de Software», 2009.

[24] «ISTQB – CAP 2 – TESTING A TRAVÉS DEL CICLO DE VIDA DEL SOFTWARE – II». [En línea]. Disponible en: https://josepablosarco.wordpress.com/2012/05/25/istqb-cap-2-testing-a-traves-del-ciclo-de-vida-del-software-ii/. [Accedido: 20-nov-2017].

[25] «Refactorización». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Refactorización. [Accedido: 22-nov-2017].

[26] D. Gomez, «Integración Continua», *2008*. [En línea]. Disponible en: https://dosideas.com/noticias/metodologias/309-integracion-continua. [Accedido: 19-nov-2017].

ANEXOS

Anexo I: Resolución PG N° 188/17 – Tablero de Gestión del MPF

Ministerio Público
Poder Judicial de la Provincia de La Pampa
Procuración General

Santa Rosa, 12 de diciembre de 2017.

VISTO:

Que la Ley Orgánica del Poder Judicial N° 2574 establece en su art. 96 los deberes y atribuciones del Procurador General; y

CONSIDERANDO:

Que entre dichos deberes, se encuentra el de fijar la planificación general del Ministerio Público y controlar su cumplimiento, optimizando los resultados de la gestión (inc. 9) y el de supervisar la tarea de los miembros del Ministerio Público, practicando visitas de inspección y auditorías, y organizar un adecuado sistema de control de gestión permanente (inc. 14).

Desde la implementación del Código Procesal Penal (Ley N° 2287) en marzo de 2011, se proyectaron y efectivizaron diversas acciones con el objeto de contribuir al cumplimiento de aquella manda legal y supervisar la labor desplegada por los integrantes del Ministerio Público en sus respectivos ámbitos de competencia.

Uno de los cursos de acción desplegados, que forma parte de un plan integral, se centró en la planificación y desarrollo de un Tablero de Gestión de las tareas a cargo del Ministerio Público Fiscal, como una herramienta destinada a brindar información completa y veraz que en tiempo real permitiera gestionar la labor de los distintos operadores, como también diagnosticar adecuadamente la situación y tomar decisiones fundadas en datos ciertos.

Con el fin de concretar dicho proyecto, en el mes de diciembre de 2016 se conformó un equipo de trabajo interdisciplinario, integrado por Juan Barbero y Martín Rodríguez - dos estudiantes avanzados de la carrera de Ingeniería en Sistemas, contactados a través de un convenio de pasantías firmado con la Universidad Nacional de La Pampa-, la Secretaria Sustituta de Política Criminal, Gestión y Planificación Estratégica Dra. Carolina Ghione y el empleado de la Procuración General, Emanuel Soria.

Durante seis meses, se proyectó y programó el software, basado en

1

MARIO OSCAR BONGIANINO
PROCURADOR GENERAL

dos principios:

1) Su base de datos es el Sistema Informático de Gestión de Legajos Penales (SIGeLP).

El SIGeLP fue desarrollado y es administrado por la Secretaría de Sistemas y Organización del Poder Judicial de la Provincia de La Pampa. Se implementó en marzo de 2011 en forma paralela con la entrada en vigencia del nuevo Código Procesal Penal y contemplando exclusivamente la interacción de operadores internos del fuero judicial. En forma progresiva, se amplió el acceso a operadores externos, como las fuerzas policiales y los letrados particulares.

La íntima vinculación entre el SIGeLP y el Tablero de Gestión, acentúa la necesidad de perfeccionar la utilización del primero por parte de todos los usuarios, ya que son quienes en definitiva generan los datos que luego expondrá el segundo.

En el marco de este principio, es que en forma paralela al trabajo de desarrollo del Tablero de Gestión, desde Procuración General en diciembre de 2016 comenzaron a remitirse mensualmente a los Fiscales Generales y Fiscales de todas las Sedes, listados de los que surgían los procesos en los que estos figuraban como responsables, con datos de interés como su estado, cantidad de días paralizados y una comparativa con los períodos anteriores.

Como parte de lo que podríamos denominar un "programa de reordenamiento de legajos en el SIGeLP", se procedió a regularizar en la faz informática la situación de legajos radicados ante UAP y ante UTC, para lo cual fue necesaria la creación de nuevas actuaciones y estados, el requerimiento de ajustes a la Secretaría de Sistemas, la reorganización de usuarios en las Fiscalías Temáticas, la elaboración de informes personalizados antes mencionados con base en uno más amplio emitido por la Secretaría de Sistemas, etc.

Además, en el marco de este principio también fueron dictadas las Resoluciones PG N° 163/17 y N° 185/17, tendientes a la difusión de instrucciones básicas para el uso del SIGeLP a usuarios clave del mismo: por un lado, quienes se

encargan de generar legajos en el SIGeLP (en el caso de Santa Rosa y General Pico, integrantes de las Unidades de Atención Primaria), y por el otro la Policía, que en el marco de su función judicial genera en el Sistema el sumario policial, que posteriormente da origen a un legajo de investigación Fiscal. Asimismo, se persiguió transmitir la trascendencia de la labor de dichos usuarios, y el valioso aporte que implica una carga completa, correcta y oportuna de datos en el SIGeLP.

2) La planificación y desarrollo del Tablero de Gestión se realiza por etapas, cada una centrada en grupos clave de información.

Si bien los objetivos concretos son dinámicos y eventualmente pueden requerir ajustes sobre la marcha -al tener en consideración las necesidades propias del Ministerio Público Fiscal-, a grandes rasgos pueden definirse tres etapas de desarrollo del Tablero de Gestión, cada una con una finalidad central. Para su comprensión, es fundamental repasar la estructura de la plataforma.

El Tablero de Gestión trabaja con grandes grupos o "categorías" de legajos, creadas al fin de la herramienta. Así, como una división primaria, pueden distinguirse por un lado los llamados legajos "*en trámite*" (aquellos que aún no han obtenido una resolución fiscal o judicial que les ponga fin) de los denominados "*TEXAS*[1]" (que son los que han obtenido una decisión de algún tipo que los concluye: archivo/desestimación fiscal, resolución de incompetencia, sentencia judicial de sobreseimiento, absolución, condena, aplicación de alguna salida alternativa como Suspensión de Juicio a Prueba o principio de oportunidad).

Asimismo, los legajos se distinguen según quién figura como su titular responsable, en legajos "*UAP*", que son aquellos que no tienen aún asignado un fiscal responsable o bien tienen asignado el Fiscal Unidad Atención Primaria o el Fiscal de Salidas Tempranas de las sedes Santa Rosa y General Pico, respectivamente, siendo ello indicativo de que se encuentran en el ámbito de dicha

[1] El nombre tiene origen en "*TErminados X Archivo o Sentencia*" como las actuaciones que con mayor frecuencia ponen fin a los legajos que se inician.

MARIO OSCAR BONGIANINO
PROCURADOR GENERAL

Unidad en las Sedes en la que la misma se encuentra formalmente constituida.

Por otro lado, los legajos agrupados como "*UTC*" engloban aquellos que tienen asignado un fiscal responsable, en el marco o no de una determinada Fiscalía Temática.

A su vez, los legajos "*en trámite*", se distinguen en sub-categorías, según se los analice desde el ámbito de UAP *(solo denuncia/ para archivar/para reservar/ reservados/ otros)* o desde la órbita de la UTC *(sin formalizar / formalizados sin acusación / con acusación sin sentencia),* ya que las labores son esencialmente diferentes y por ende también la problemática que puede presentar cada una.

Los legajos *"en trámite"* son el centro de la primera etapa de desarrollo del Tablero de Gestión, tendiente a destacar legajos en los que el o la Fiscal aún tiene tareas pendientes que realizar vinculadas con la investigación preparatoria. Sin perjuicio de la función continua de los Fiscales (art. 72 C.P.P.), en esta fase el foco está puesto en la etapa inicial del proceso penal.

La segunda etapa del Tablero de Gestión –actualmente en desarrollo- apunta en cambio a los resultados y se centraliza en los legajos denominados "*TEXAS*", es decir –como arriba se explicara- aquellos que ya se encuentran concluidos de alguna de las formas previstas por la normativa vigente. Con los avances que se producirán al implementarse esta etapa, se podrá contar con información sobre los diversos modos de conclusión de los procesos, pudiendo distinguirlos, consultar su frecuencia, como también los tiempos transcurridos entre ciertos actos fundamentales. Progresivamente se apunta a que el Tablero de Gestión además de exhibir información sobre legajos en concreto (si bien agrupados de acuerdo a diferentes criterios), también exponga información general.

La tercera etapa, busca la obtención de información útil para la investigación, por ejemplo, trabajando con los datos que surgen de los legajos sin autor identificado, y proyectando la elaboración de mapas del delito, días y horarios de comisión de los diferentes delitos, etc.

La primera fase del Tablero de Gestión fue examinada en modo prueba durante los meses de Julio y Agosto de 2017, con resultados exitosos. Superada dicha instancia, con la colaboración de la Secretaría de Sistemas y Organización dependiente del Superior Tribunal de Justicia, se dio acceso a la herramienta al Procurador General a fines de dicho mes.

Desde su perfil y en el marco de esta primera fase, puede visualizarse información sobre todos los legajos que se encuentran "*en trámite*" en todas las Sedes del Ministerio Público Fiscal de la Provincia de La Pampa, distinguiéndose en "UAP" y "UTC" en las sedes Santa Rosa y General Pico.

En Septiembre de 2017, se otorgó acceso a los cuatro Fiscales Generales, pudiendo cada uno visualizar lo vinculado con la Sede del Ministerio Público Fiscal sobre la que ejercen su Jefatura, de conformidad con las Res. P.G. Nros. 9/11, 10/11, 60/15, 122/15 y 97/17.

Asimismo, encontrándose a la fecha disponible la posibilidad de otorgar acceso al Tablero de Gestión a los y las Fiscales de toda la Provincia, y en el convencimiento de que dicha herramienta les facilitará el contralor de su labor diaria, y contribuirá a un mejor desempeño, como también los instrumentos que paulatinamente se vayan incorporando en el marco de la segunda y tercera etapa de desarrollo,

El Procurador General de la Provincia de La Pampa

RESUELVE:

1°) Hacer saber a los Fiscales Generales y de los Fiscales de la Provincia de La Pampa, que cuentan con un acceso al Tablero de Gestión ajustado a la Sede y de acuerdo a la función que en la misma desempeñan, pudiendo acceder a través de la opción "*Tablero de Gestión*", que se encuentra en el menú "*Estadísticas*", del Sistema Informático de Gestión de Legajos Penales.

2°) Incorporar el documento titulado *"Instructivo conceptual y de uso. Tablero de Gestión"* como "**Anexo I**" de la presente

3°) Regístrese. Póngase la presente y su "Anexo I" en conocimiento de los Fiscales Generales y por su intermedio de los Sres. y Sras. Fiscales de toda la Provincia, del Superior Tribunal de Justicia y de la Secretaría de Sistemas y Organización dependiente del mismo. Cumplido, archívese.

Resolución P.G. N° 188/17

MARIO OSCAR BONGIANINO
PROCURADOR GENERAL