Maestría en Tecnología Informática y de Comunicaciones

Cohorte 2019 – 2020

Título del Trabajo Final:

“SAICo: Software automatizado para inspecciones y control”

Autor: *Juan Martín Roldán*

Director del Trabajo Final: ***Mg. Pablo Ezequiel Inchausti***

Institución a la que pertenece: UADE FAIN

Fecha de entrega: 10/03/2021

AGRADECIMIENTOS

En primer lugar, dentro del cuerpo docente de la UADE, al Mg. Pablo Inchausti y al Mg. Adrián de Armas por su contribución al desarrollo de este trabajo.

A los entrevistados, Ivan Pavloff y Carlos Rodriguez, por el tiempo brindado y por aportar su opinión, destacando aspectos fundamentales que se tuvieron en cuenta para el desarrollo del software.

A Daniela y a Roma, por su amor y apoyo incondicional en todos mis proyectos.

A mi madre Silvia, a marito y a mi tío Hugo por la crianza recibida, los valores transmitidos y el respaldo constante, pilares fundamentales para mi desarrollo como persona y profesional.

ABSTRACT

The inspections allow a control that, depending on the work context, could be a critical aspect. However, these controls require the use of resources and consequently an effort, which many times limit the quantity and quality of the inspections carried out on the objectives to be audited.

In most work areas, controls are carried out whose variety and specificity are as wide as there are objects of inspection. These characteristics add difficulty to establish a single tool that adapts to the different possible scenarios. In this work, a basic control flow is proposed and software is developed that includes customizable modules to adapt to the different work realities. In addition, this software has visualization and statistics tools, with an algorithm for automatic assignment of objectives and analysis rules that allow the system to carry out actions without the manual intervention of a user.

RESUMEN

La realización de inspecciones permite llevar cabo un control que, dependiendo del contexto de trabajo, puede ser un aspecto crítico. Sin embargo, estos controles requieren la involucración de recursos y consecuentemente un esfuerzo, que muchas veces actúan como factores limitantes en cuanto a la cantidad y la calidad de las inspecciones que se deben ejecutar sobre los objetivos que se deben fiscalizar.

En la mayoría de las áreas de trabajo, se requiere llevar cabo controles, cuya variedad y especificidad son tan amplias como objetos de inspección existen. Estas características, plantean cierta dificultad a la hora de establecer una única herramienta que se adapte a los distintos escenarios posibles. En el presente trabajo, se propone un flujo básico de control y, se desarrolla un software que incluye módulos parametrizables para adecuarse a las distintas realidades de trabajo. Además, este sistema cuenta con herramientas de visualización y estadísticas, junto con un algoritmo de asignación automática de objetivos y reglas de análisis que, permiten que el sistema realice las acciones que en caso contrario requerirían manual de la intervención de un usuario.

ÍNDICE

1. INTRODUCCIÓN

En la actualidad, existen distintos entes u organismos tanto públicos como privados, que realizan inspecciones con la finalidad de ejercer un control sobre diferentes tipos de entidades. El objetivo de este control, es llevar a cabo una evaluación sobre determinados puntos de interés, para obtener un resultado que permita determinar si se requiere de cambios o ajustes para subsanar los defectos o fallas que pudieran haberse detectado.

Estas inspecciones o controles, se llevan a cabo utilizando diversos medios, que pueden ser tanto manuales/físicos como digitales. El primer caso puede ser representado simplemente con una hoja de papel, donde se muestra un checklist que enlista una cierta cantidad de ítems a verificar. En cambio, el segundo caso, varía según el alcance del producto utilizado, es decir, puede tratarse de un software cuyas funciones están orientadas simplemente a digitalizar el trabajo en papel, o una herramienta con funciones adicionales como la automatización del trabajo, obtención de estadísticas, generación de reportes, interacción con otros sistemas, etc.

Tomando como ejemplo el contexto actual de pandemia, planteado por el COVID-19, el hecho de controlar que las distintas actividades habilitadas para operar, cumplan con el protocolo de seguridad e higiene correspondiente, contribuye a evitar la propagación de la enfermedad. Sin embargo, para llevar a cabo este control, se requieren diversos tipos de recursos que lo transforman en algo complejo y costoso, si se analiza el universo de entidades que se deben controlar.

El presente trabajo aporta un software que contribuye a disminuir los esfuerzos necesarios para llevar a cabo una inspección, presentando una solución de respaldo al proceso de control, que automatiza tareas en determinados puntos críticos.

Por lo tanto, si se emplea un software para la realización de inspecciones, se obtienen numerosas ventajas:

- Mayor rapidez en la ejecución de tareas
- Mayor trazabilidad
- Control y segmentación del acceso a la información
- Generación de reportes y estadísticas
- Automatización de tareas

1.1. Descripción del problema

Pressman (2013) señala que “es imposible operar el mundo moderno sin software. Las infraestructuras nacionales y los servicios públicos se controlan mediante sistemas basados en computadoras, y la mayoría de los productos eléctricos incluyen una computadora y un software de control”.

De acuerdo a este planteo, se vuelve indispensable la idea de conducir y gestionar cada vez más actividades a través de herramientas de software que faciliten y agilicen las tareas. Sin embargo, dada la complejidad y diversidad del mundo actual, se torna esencial que estos sistemas tengan una gran amplitud y flexibilidad, para adaptarse a las distintas circunstancias que se plantean.

Otro aspecto importante a considerar, es la inclusión de algoritmos que analicen la información recibida y apliquen una lógica programada sobre determinadas variables, para tomar una decisión evitando la intervención y el posible error humano.

La mayoría de las herramientas de software disponibles para realizar inspecciones, se crean pensando en una problemática en particular, por lo cual permanecen demasiado rígidas a la hora de tener que responder ante una situación diferente a aquella para la cual fue desarrollada. Esta circunstancia, genera que se deban utilizar sistemas específicos para cada tipo de inspección, a pesar de que existen ciertos puntos en común que sirven como como base para realizar un planteo generalista, orientado a la flexibilidad. Por ejemplo: no es lo mismo la fiscalización de una obra en construcción, a cargo de un municipio, que la inspección de un casino, a cargo de otro organismo

gubernamental, responsable de la administración y explotación de la actividad vinculada a los juegos y apuestas. En el primer caso, se buscan diferencias entre lo habilitado y lo realmente construido, el cumplimiento de determinadas medidas de seguridad, etc. En cambio, en el segundo caso, se busca detectar irregularidades respecto al funcionamiento de las máquinas tragamonedas, controlar el inventario de los equipos en funcionamiento, etc. Si bien ambos casos apuntan a distintos objetivos de control, comparten ciertos aspectos en común:

- Ubicación: siempre existe una geolocalización, donde se sitúa el establecimiento inspeccionado
- Checklist: si bien las preguntas y las posibles respuestas son distintas, en ambos casos existe una estructura de cuestionario, donde se refleja la información relevada
- Área: el trabajo es realizado por distintos sectores de trabajo, dentro de un determinado organismo.
- Motivo: existe múltiples causas que pueden dar origen a un control, por ejemplo, una simple inspección de rutina o verificar la subsanación de una falta detectada anteriormente.
- Resultado: toda fiscalización concluye con un resultado especifico, que puede ser favorable o desfavorable, y disparar acciones en consecuencia.

La falta de una herramienta de software, que pueda soportar las diferentes características de las entidades a inspeccionar, provoca que cada empresa u organismo realice un esfuerzo orientado a resolver únicamente su propia problemática. En el caso del Estado, esta situación se traduce un en mal gasto de recursos públicos, ya que cada municipio u ente gubernamental que tenga entre sus funciones la realización de controles, requerirá de un sistema propio adecuado a su situación particular.

1.2. Objetivos

Desarrollar un software de administración y seguimiento de inspecciones que, mediante módulos parametrizables, permita llevar a cabo diferentes tipos de controles, aplicando algoritmos y reglas orientadas a la automatización de las tareas involucradas.

1.3. Objetivos particulares

Dentro de los objetivos particulares de este trabajo, se plantean los siguientes:

- Evaluar el framework Xamarin para el desarrollo de la aplicación mobile.
- Evaluar la plataforma .Net Core y el motor de base de datos MySQL para el desarrollo del back end.
- Evaluar el framework Angular para el desarrollo del front end.
- Evaluar el entorno de Node Js y MongoDB para la construcción de un manejador de archivos.
- Desarrollar un algoritmo que permita asignar automáticamente los objetivos de inspección programados
- Desarrollar reglas que permitan analizar el resultado obtenido en una inspección, y apliquen una acción automática en consecuencia.
- Desarrollar un software en base a módulos parametrizables, para lograr un producto adaptable a distintas circunstancias de trabajo.
- Realizar entrevistas a expertos en el uso de sistemas de inspección para validar el aporte del software desarrollado.

1.4. Marco metodológico

El enfoque utilizado en este trabajo, es del tipo cualitativo ya que las decisiones que se toman con respecto al desarrollo del software, son subjetivas de acuerdo al conocimiento del investigador y a los estudios previos. Es decir, la “realidad” se define a través de las interpretaciones de los participantes en la investigación de acuerdo a su propia visión. Lógicamente, las mismas pueden variar conforme avance el estudio y a

su vez, constituyen las fuentes de datos. A partir de este enfoque, se adquiere un punto de vista interno, manteniendo una perspectiva analítica pero no se definirán variables con el objetivo de operarlas experimentalmente (Hernández Sampieri y otros. 2014).

Respecto al tipo de diseño, es del tipo descriptivo ya que busca especificar las propiedades y características del proceso de inspección realizado a través de un software, es decir, se describen las decisiones tomadas respecto a la arquitectura, diseño y desarrollo del mismo. Se eligen una serie de conceptos y variables, y se pretende recoger información al respecto, sin considerar como se relacionan entre ellas (Hernández Sampieri y otros. 2014).

Por otro lado, se realizan entrevistas con individuos con una amplia experiencia en el mundo de la fiscalización, con el objetivo de realizar una validación de las funcionalidades del software y de las tareas automatizadas. Es decir, se pretende corroborar la utilidad de la herramienta, consultando con expertos que trabajen o hayan trabajado, con aplicaciones similares.

El tipo de entrevista utilizado es la “entrevista cualitativa” del tipo “semiestructurada”. La entrevista cualitativa es más íntima, flexible y abierta que la cuantitativa. Es en buena medida anecdótica y tiene un carácter más amistoso, es decir, se define como una reunión para conversar e intercambiar información entre el entrevistador y uno o varios entrevistados. Las entrevistas del tipo semiestructuradas, se basan en una guía de asuntos o preguntas, pero el entrevistador tiene la libertad de introducir preguntas adicionales, para obtener mayor información o detalle (Hernández Sampieri y otros. 2014).

2. MARCO CONCEPTUAL Y ESTADO DEL ARTE

El concepto de automatización, hace referencia a los trabajos realizados por un operario humano, que en la Industria 4.0 pasan a ser automatizados y sustituidos por una máquina, un software informático o por un robot. El proceso de automatización

implica mejorar los tiempos de ciclo, la productividad, la calidad del proceso y la competitividad de una empresa u organismo, pudiendo lograr además un aprendizaje automático. (Basco y otros. 2018)

En este trabajo, se propone aplicar este concepto de automatización sobre un circuito de inspección, como el que se observa en la Figura 2.1, donde se incluyen las siguientes actividades:

- Carga: implica la identificación y agrupación del conjunto de datos relativos al entorno de la inspección.
- Programación: una vez constituido el objeto de control, se procede a la anulación (en caso de error) o bien a la asignación los usuarios responsables que llevarán a cabo el control.
- Anulación: es un estado final, representa la cancelación de un objeto de inspección cargado previamente.
- Inspección: es la instancia en la que se procede a realizar el control programado, evaluando un conjunto de ítems que dependerán del tipo de entidad que se esté fiscalizando. En caso de que la inspección no pueda llevarse a cabo por algún motivo, el objetivo se devuelve al área de programación para que sea reasignado posteriormente.
- Análisis: consiste en la evaluación de los resultados obtenidos, para determinar las acciones que se deben llevar cabo. En algunos casos, deriva en la solicitud de una nueva inspección, en caso de que haya que verificar posteriormente la corrección de las irregularidades detectadas.

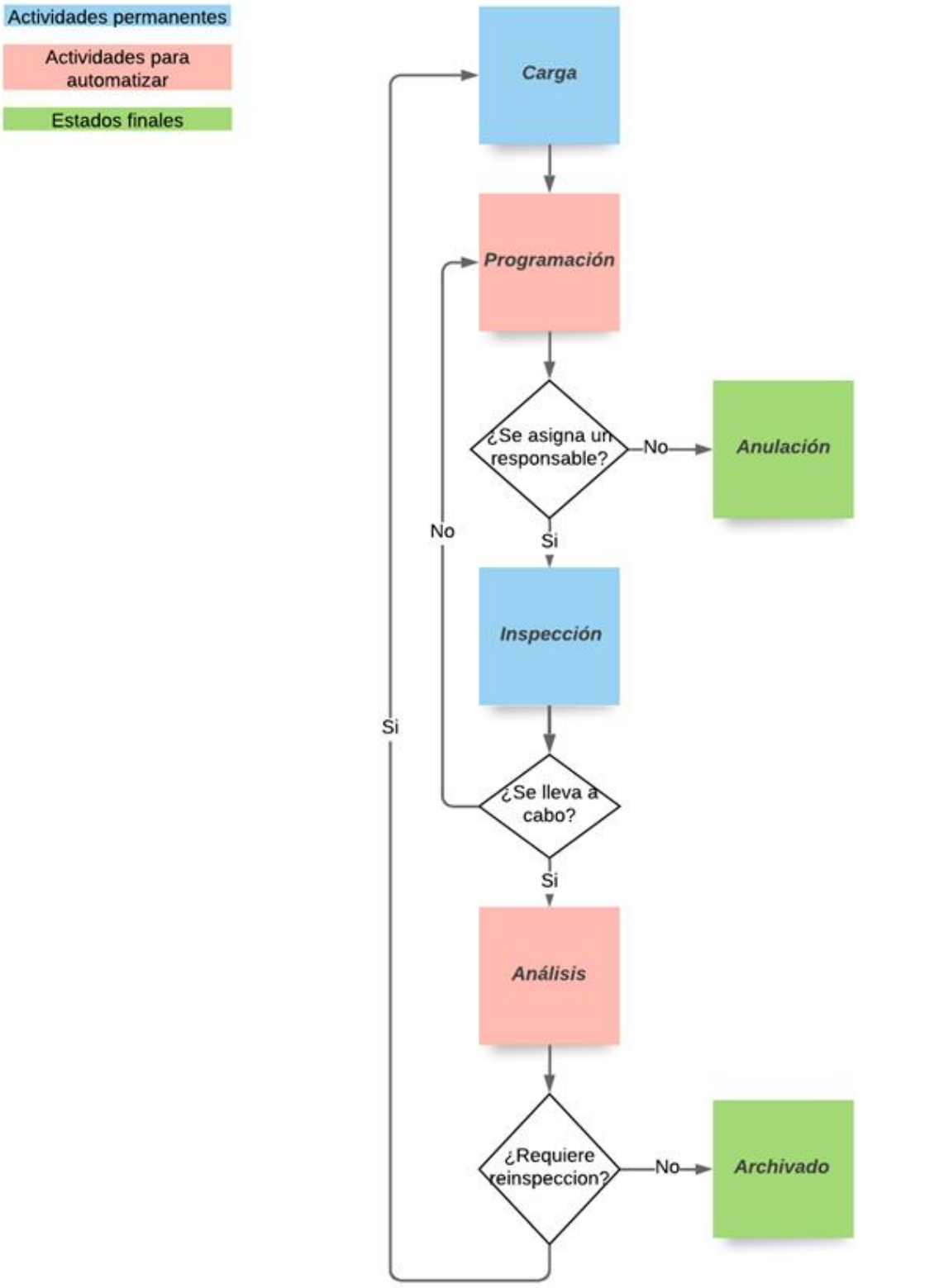

Figura 2.1: Circuito básico de control

De acuerdo a la estructura del circuito básico descripto en la figura 2.1, se propone automatizar las actividades de Programación y Análisis utilizando el siguiente criterio:

- Programación: se incorpora un algoritmo que permite realizar la asignación de manera automática, identificando aquellos casos que tengan mayor criticidad.
- Análisis: se desarrolla un módulo, donde el usuario podrá definir reglas de evaluación, que especifican la lógica utilizada para tomar las decisiones en esta instancia. Es decir, se definen un conjunto de casos y acciones que el sistema debe realizar en consecuencia.

Además, para dar respuesta a las distintas circunstancias y contextos donde se lleva a cabo una inspección, se incluyen una serie de módulos parametrizables que permiten que el software se adapte, a través de la configuración de los mismos.

Por otra parte, para permitir el control transparente y susceptible de modificación, que forma parte esencial del concepto de software libre (Stallman y Lessig. 2007), se plantea la utilización de herramientas acorde a estos lineamientos. De esta forma, se podrán llevar a cabo modificaciones y mejoras en futuras líneas de investigación o en implementaciones del software en algún organismo, sin necesidad de incurrir en gastos de licencias. Lógicamente estos cambios deberán ser realizados y gestionados por el/los responsables que pretendan utilizar el software.

Si bien existen soluciones de software que permiten la carga de cuestionarios tipo “checklist”, están orientados a los ítems de verificación que se deben llevar a cabo durante la inspección, es decir, no están orientados a la fácil parametrización y a la automatización de tareas, como se plantea en este trabajo. Por este motivo, el tema propuesto presenta interés para diversos organismos, tanto públicos como privados, que tienen dentro de sus funciones la realización de inspecciones para llevar a cabo diversos tipos de controles.

2.1. Industria 4.0 y automatización

Existe un factor común entre todas las épocas por las que fue transitando la humanidad: la búsqueda de una superación para progresar, es decir, dar saltos cualitativos que mejoren la calidad de vida. Siguiendo con este lineamiento y, en relación a la producción de bienes y servicios, surge el concepto de “Industria 4.0” o “Cuarta Revolución Industrial”, que plantea a la tecnología y a la innovación como dos factores fundamentales. La eficiencia en la cadena de valor, mejora claramente a través del análisis de datos y la toma de decisiones. Los mercados se amplían con las plataformas digitales, que permiten compartir información entre distintos actores del sistema productivo. Como consecuencia de esta colaboración entre empresas y actores, aparecen nuevos modelos de negocios (Basco y otros. 2018).

De acuerdo con esta nueva industria, la automatización de flujos de trabajo y la gestión de procesos de negocios, forman parte de las organizaciones públicas y privadas. Gradualmente las empresas avanzan en la automatización de sus procesos, articulando las tareas realizadas por diferentes individuos, compartiendo información y documentación con el objetivo de ser más eficaces y eficientes. A su vez, ésta automatización se aplica en los distintos niveles de la pirámide de conocimiento, hasta el punto tal que un experto que analiza y decide, de acuerdo a su experiencia, pueda ser asesorado y hasta relevado en algunos casos (Moreno. 2016).

Los nuevos enfoques para los próximos cambios de automatización están impulsando nuevas soluciones. El vínculo laboral entre humanos y máquinas, va evolucionando en conjunto con las habilidades requeridas, por tal motivo, la automatización traerá grandes cambios, desplazando a millones de trabajadores solo en la próxima década. En este sentido, se prevé que la robótica y la automatización podrían provocar el desplazamiento de 20 millones de puestos de trabajo de fabricación para 2030. Sin embargo, este cambio no tiene que considerarse como algo negativo, sino como una necesidad de nuevos roles y perfiles de trabajo, que lógicamente requieren de capacitaciones para realizar un tareas más técnicas y difíciles de automatizar. Por tal

motivo, es necesario proteger a los trabadores llevando a cabo tareas vinculadas a la readaptación necesaria para el cambio, apoyando a las necesidades de las futuras empresas de fabricación moderna. (Amar Hanspal. 2020).

En el presente trabajo, además del desarrollo de la plataforma de software para la realización de inspecciones, se plantea la automatización de dos actividades que forman parte del proceso, que se observa en la figura 2.1. De esta forma se pretende sentar las bases para una mejora en la eficiencia y la eficacia del proceso, liberando a los recursos que antes eran destinados a estas tareas, de modo tal que puedan avocarse a otras.

2.2. Softwares de inspecciones y control

Analizando las distintas alternativas presentes en el mercado, lógicamente existen soluciones tanto a nivel global como a nivel local. El factor común que se observa en ambos casos es la orientación hacia soluciones enfocadas en un cuestionario de preguntas y respuestas que constituyen la inspección, generando un informe como resultado. Es decir, no se centran en la automatización de las tareas involucradas en el proceso de control, y en algunos casos tampoco plantean la flexibilidad de adaptación mediante módulos parametrizables (externos al cuestionario de inspección), como se propone en este trabajo.

A nivel mundial, se encuentran disponibles soluciones como “ProntoForms”, que se puede observar en la figura 2.2.1. Es una solución para el campo de automatización de formularios sin papeles, permite que los empleados técnicos capturen rápidamente datos para las inspecciones y auditorías mediante el uso de una aplicación móvil. Este software se encuentra disponible en países como Estados Unidos, Canadá, Reino Unido, Australia y la India (ProntoForms. 2020).

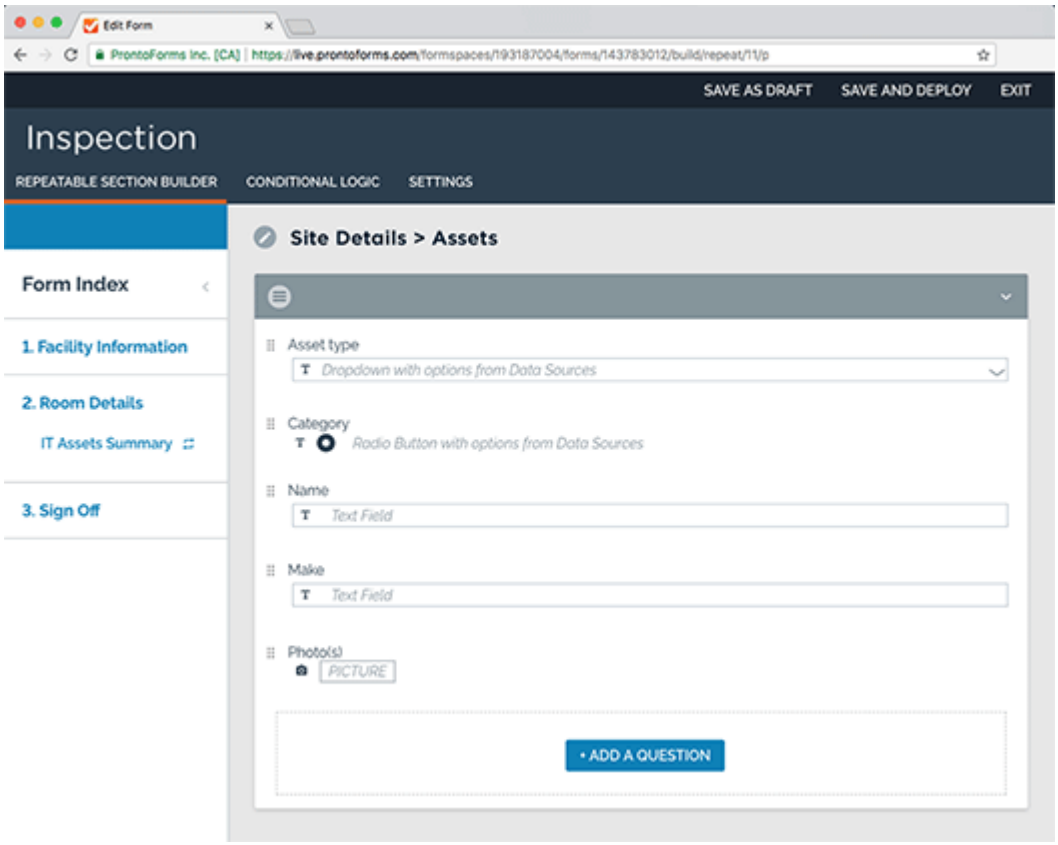

Figura 2.2.1: ProntoForms Software (ProntoForms. 2020)

Otra opción disponible es “Certainty Software”, que es una solución de administración de auditoría e inspección para grandes empresas que permite ingresar datos usando papel, navegadores o la aplicación Checkit. Es utilizada por más de 100.000 profesionales para completar más de 2.000.000 de auditorías e inspecciones anualmente (Certainty Software. 2019).

Figura 2.2.2: Certainty Software (Certainty Software. 2019)

Un tercer caso que se puede mencionar es MaintainX, que es un software de digitalización de procedimientos móvil, que permite realizar inspecciones, comprobaciones de seguridad, listas de verificación, etc, todo con una auditoría digital. Esta solución está disponible en idioma Ingles, en países como Estados Unidos, Canadá, Reino Unido, Australia, China, India, Japón, Alemania, Brasil y México (MaintainX. 2020).

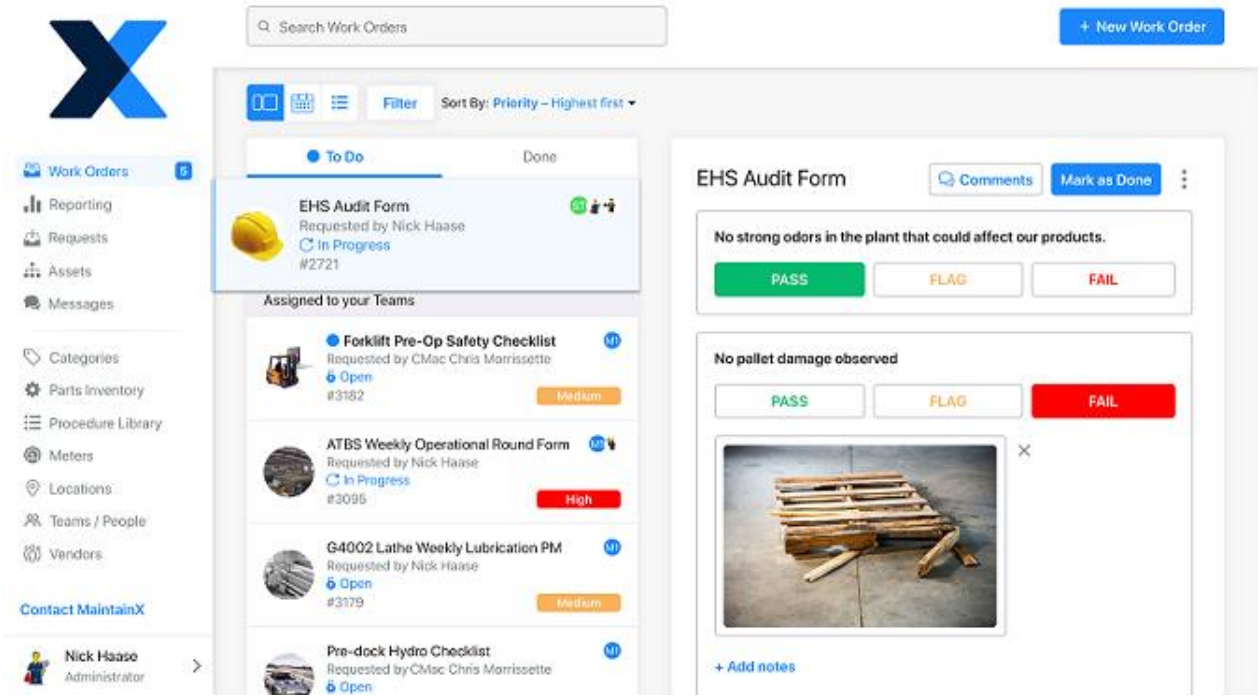

Figura 2.2.3: MaitainX Software (MaintainX. 2020)

En Argentina, dentro de los organismos públicos, se encuentra el software “Liza”, desarrollado por la empresa Softing SRL para la Agencia Gubernamental de control (AGC). Esta solución posibilita el monitoreo, la fiscalización y los relevamientos, incluso a través de dispositivos móviles. Asimismo, garantiza la programación y la carga de datos, incluyendo imágenes, archivos adjuntos, lecturas de códigos QR, etc. Dicha herramienta fue cedida por el Gobierno de la Ciudad de Buenos Aires al Ministerio de Salud, Desarrollo Social y Deportes, de la provincia de Mendoza (Buenos Aires Ciudad. 2018) y también a la Secretaria de Asuntos Políticos e Institucionales del Ministerio del Interior, Obras Públicas y Vivienda de la Nación Argentina para la fiscalización de los

procesos electorales realizados en el 2019 (Clarin.com. 2018). En adición a estos organismos, también fue cedida en el año 2018 al Instituto Provincial de Lotería y Casinos de la provincia de Buenos Aires (Lanuse. 2018).

Figura 2.2.4: Liza Software (AGC. 2020)

Dentro del mismo Gobierno de la Ciudad de Buenos Aires, otras dependencias utilizan Oracle Field Service Cloud (OFSC) que es una solución configurable, que programa y optimiza automáticamente el trabajo en función de sus requisitos y mantiene una vista del campo en tiempo real, tiene tantos módulos web como móvil (Oracle Argentina. 2020)

Por otro lado, también se encuentra el software “Documenta”, que se utiliza en el ámbito de inspecciones para el control de plagas. Fue desarrollado por el “Grupo Ginebra” y tiene como caso de éxito la empresa “Clean Garden”. Este sistema permite a los inspectores rellenar un formulario con fotos y firma electrónica, cuando se termina el trabajo, el supervisor recibe el detalle y luego de ser aprobado se envía el informe al cliente (Grupo Ginebra. 2020).

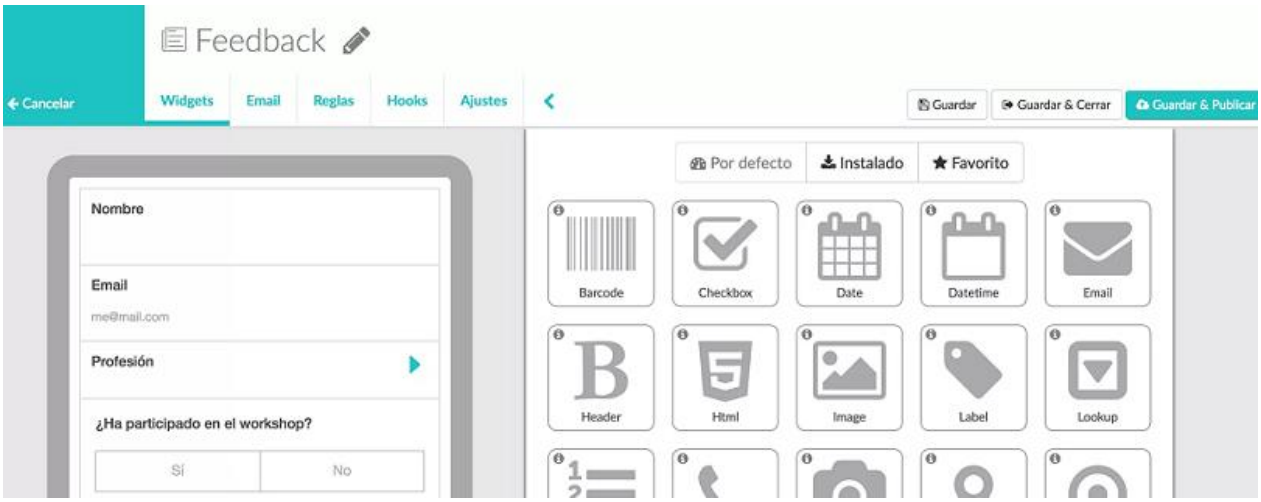

Figura 2.2.5: Documenta Software (Grupo Ginebra. 2020)

3. DESARROLLO DEL TRABAJO

El objetivo principal del trabajo, es el desarrollo de la plataforma SAICo (Software Automatizado de Inspecciones y Control), que permite llevar a cabo inspecciones y diversos tipos de controles, incluyendo la automatización de las actividades mencionadas en la sección 2.

3.1. Ciclo de vida del software

Según la definición del estándar ISO/IEC/IEEE 12207 (2017), el ciclo de vida de un producto de software, es un modelo de referencia que abarca desde la etapa inicial donde se definen los requisitos hasta su retiro, e incluye los procesos, actividades, explotación y mantenimiento.

En este trabajo, se utiliza el ciclo de vida “incremental”, cuya filosofía se basa en construir productos parciales y por cada iteración se van ampliando las funcionalidades del sistema. Es decir, se aumentan gradualmente las capacidades del software, por lo cual permite obtener versiones previas a la entrega final, con el objetivo de reducir los riesgos asociados a desvíos, que en caso contrario serian detectados en forma tardía al

obtener el producto final (PRESSMAN. 2013). A continuación, se incluye un diagrama donde se puede observar el esquema de este ciclo de vida:

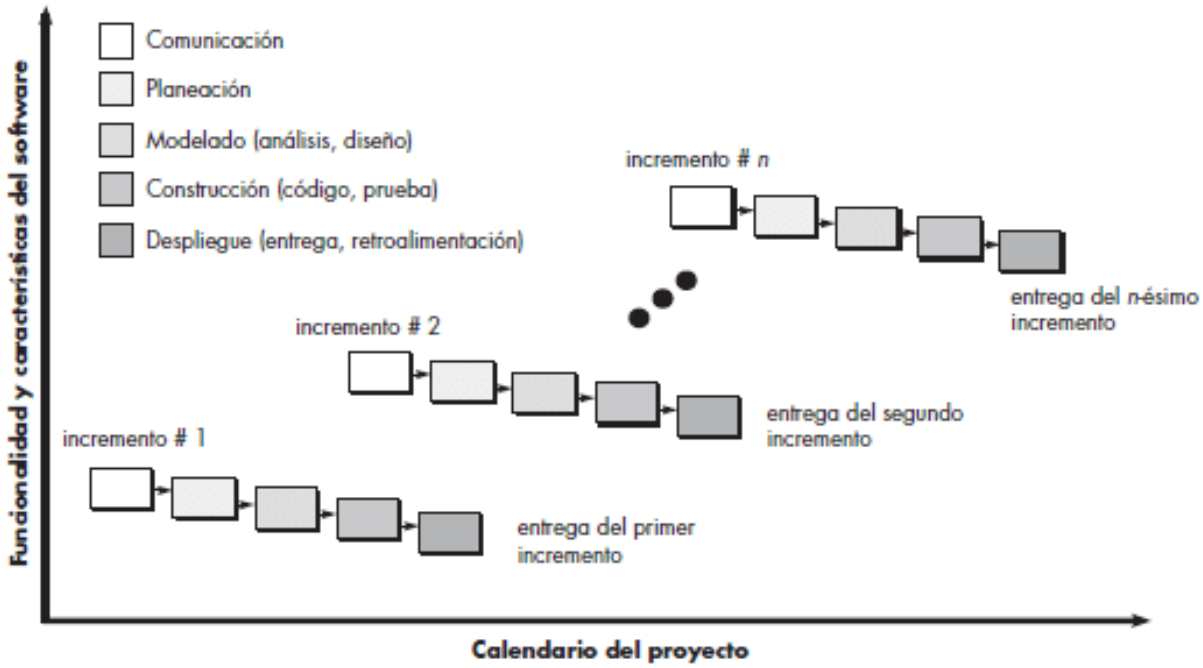

Figura 3.1.1: Ciclo de vida incremental (PRESSMAN. 2013)

3.1.1. Metodología de desarrollo

En los procesos de desarrollo de software estrictos y controlados, donde se realiza una minuciosa planificación y un aseguramiento de calidad formalizada, se incluyen costos operativos que son demasiado grandes para sistemas pequeños y medianos. Es decir, se dedica más tiempo al diseño que al desarrollo y la prueba del software. Por otra parte, si se modifican los requisitos es indispensable ajustar la especificación y el diseño (Sommerville. 2012).

Con el objetivo de mejorar el producto a partir de la práctica y la observación, en este trabajo se utilizan los principios de las metodologías agiles para el desarrollo del software, que establecen iteraciones incrementales donde se pueden detectar rápidamente los desvíos, para evitar llevar a cabo un esfuerzo sobre un trabajo que luego deba deshacerse. Cuando se realizan estas revisiones, se busca corroborar que

las tareas llevadas a cabo, contribuyen a satisfacer alguno de los objetivos del sistema. Estas comprobaciones, disminuyen en el riesgo en comparación a un desarrollo en donde se plantea una idea y se lleva a cabo el 100% del trabajo, sin validaciones intermedias (Schwaber. 2012).

Para realizar estas validaciones, se planifica la división del producto en dos partes:

- Versión 1.0.0 / “Gadget” (en alusión a la famosa serie de dibujos animados): el objetivo está centrado en los “Conceptos fundamentales” del sistema, es decir, dejar preparados los distintos ABMs, la carga de datos necesarios para el uso de la herramienta y realizar el planteamiento de las pantallas donde confluye esta información.
- Versión 2.0.0 / “Hiperion” (en referencia al dios griego, considerado como el “dios de la observación”): una vez establecidas las bases, sobre las cuales lleva a cabo el trabajo de campo, el objetivo está puesto en la ejecución de las inspecciones y, posteriormente en la visualización de la información registrada.

Por tal motivo, las entrevistas con los expertos son realizadas al finalizar cada una de estas versiones, con el propósito de validar los paquetes del producto y adquirir nuevas ideas como resultado de esta interacción.

3.1.2. Plan de trabajo

Para el desarrollo del producto se establece un cronograma total de seis meses, dividido en iteraciones de un mes. Este plazo se estableció teniendo en cuenta la cantidad de tareas a llevar a cabo y, la cantidad de recursos disponibles con una dedicación semanal de 20 (veinte) horas.

Nombre de tarea	Duración	Predecesoras	Entregable
Sprint 1	**30 días**		
Análisis de requisitos V1	15 días		
Diseño de mockups V1	15 días	2	
Sprint 2	**30 días**		
Generación de repositorios	1 día		
Desarrollo Front End: login y pantallas de ABMs	12 días	5	
Desarrollo Back End: conexión con base de datos, modelado de entidades, servicios para el alta-baja-modificación de los conceptos fundamentales	14 días	5	
Desarrollo FileManager	7 días	5	
Sprint 3	**30 días**		**Version 1.0.0 "Gadget"**
Desarrollo Front End: bandeja de trabajo	5 días	6	
Desarrollo Back End: bandeja de trabajo	5 días	7	
Desarrollo Móvil: login,menu lateral, descarga y visualización de tickets en bandeja de trabajo	14 días	5	
Pruebas V1	7 días		
Implementación version 1.0.0 en ambiente de demo	2 días		
Entrevista a experto 1	1 día	14	
Sprint 4	**30 días**		
Análisis de requisitos V2	1 día		
Diseño de mockups V2	1 día?	17	
Sprint 5	**30 días**		
Desarrollo Móvil: carga de checklist, adjuntado de archivos y envio de tickets finalizados	7 días		
Desarrollo Front End: visualización de datos asociados a la ubicacion	5 días		
Desarrollo Back End: algoritmos de automatizacion e integracion con servicio de Google Maps	10 días		
Sprint 6	**30 días**		**Version 2.0.0 "Hiperion"**
Desarrollo Back End: exportaciones, estadísticas e informes	7 días	22	
Desarrollo Front End: exportaciones, estadísticas e informes	7 días	21	
Pruebas V2	7 días		
Implementación version 2.0.0 en ambiente de demo	2 días		
Entrevista a experto 2	1 día		

Figura 3.1.2.1: Plan de trabajo

3.1.3. Implementación de prácticas DevOps

El término “DevOps” es un acrónimo ingles de Development (Desarrollo) y Operations (Operaciones), es una cultura estrechamente vinculada a las metodologías agiles, que tiene por objetivo acortar la brecha y las diferencias de enfoques entre los diversos sectores que participan en el ciclo de vida de las aplicaciones, para dar un mejor servicio al usuario final. Es decir, DevOps influye en el ciclo de vida del software a lo largo de las fases de planeamiento, desarrollo, entrega y uso. Cada fase depende de las demás y las fases no son específicas de un rol. Implementando una cultura DevOps, con las prácticas y herramientas que involucra, se consigue una respuesta mejor a los requerimientos de cliente, alcanzado los objetivos más rápido e incrementando la confianza en las aplicaciones desarrolladas (Hüttermann. 2012).

Con respecto a las practicas DevOps, Microsoft Azure (2020) menciona las siguientes:

- Integración y entrega continuas (CI/CD): la integración continua consiste en combinar los cambios realizados en el código del proyecto, de manera frecuente y en un repositorio central, de forma que se puedan realizar pruebas y descubrir fallos con mayor rapidez. Luego de llevar a cabo esta práctica, aparece el concepto de entrega continua, que hace referencia a la entrega de actualizaciones de la aplicación a los usuarios, sobre una base sólida y constante.
- Control de versiones: facilita el trabajo en equipo, gestionando el código a través de versiones, que permite registrar e identificar los cambios realizados, facilitando la recuperación del código, en caso de que sea necesario.
- Desarrollo ágil de software: fomenta la retroalimentación entre los distintos perfiles involucrados en el proyecto, promoviendo lanzamientos de nuevas versiones a través de iteraciones cortas de trabajo. Un ejemplo muy utilizado actualmente es Scrum.
- Infraestructura como código: trata a la infraestructura de configuración como un software de programación, por lo cual permite reducir riesgos, costos y agiliza las implementaciones.

- Administración de configuración: consiste en identificar y controlar los elementos en el sistema, para garantizar la integridad y la calidad del producto en el transcurso del desarrollo.
- Supervisión continua: consiste en el monitoreo constante de las aplicaciones y la infraestructura con el objetivo de evitar los tiempos de inactividad.

En el presente trabajo, se decide implementar la cultura y las practicas DevOps, con el objetivo de obtener un producto de software de mayor calidad a través de un desarrollo incremental.

Herramienta	Etapa	Descripción
GitHub	Planificación, Codificación	Entorno que facilita la planificación, ya que incluye una herramienta para gestionar proyectos, y su vez sirve como repositorio del código fuente (GitHub Inc. 2020).
Azure Devops Pipelines	Construcción	Permite automatizar las compilaciones y las implementaciones (Microsoft Azure. 2020).
Docker	Distribución	Plataforma que automatiza el despliegue de aplicaciones dentro de contenedores de software, es decir, proporciona una capa adicional de abstracción y automatización con el objetivo de facilitar la implementación en diversos entornos (Docker. 2020).
Amazon Aws	Despliegue	Es una plataforma líder en servicios basados en la nube, que cuenta con multiples características que van desde tecnologias de infraestructura hasta

		servicios de aprendizaje automatico, inteligencia artificial, etc (Amazon Aws. 2020).

Tabla 3.1.3: Herramientas DevOps

3.1.4. Seguridad

La Seguridad Informática se puede definir como cualquier acción destinada a impedir la realización de operaciones no permitidas sobre un sistema o red informática. En un sentido más amplio, la norma ISO/IEC 17799 considera a la Seguridad de la Información como la protección de los principios de confidencialidad, integridad y disponibilidad (conocidos como “CIA” por sus términos en inglés). Por consiguiente, ambos términos no son exactamente iguales siendo el concepto de Seguridad de la Información mas abarcatativo ya que la información puede estar contenida en diversos medios, que pueden no ser digitales (Vieites. 2010).

En relación a la Seguridad, que permite proteger los activos de la plataforma desarrollada de manera transversal a los distintos módulos, se busca conseguir el cumplimiento los tres principios básicos de la seguridad de la información mencionados anteriormente:

- Confidencialidad: es la forma de evitar la divulgación de información, a personas o sistemas que no se encuentran autorizados
- Integridad: consiste en mantener los datos intactos, es decir, sin modificaciones o alteraciones por parte de terceros, por lo tanto, la información que se dispone es válida y consistente.
- Disponibilidad: la información debe estar accesible cuando el usuario o el sistema necesite realizar la consulta.

Para lograr y reforzar el cumplimiento de los tres principios fundamentales de la Seguridad, en SAICo se implementan los siguientes mecanismos:

- Autenticación: se utiliza un mecanismo basado en JSON Web Tokens (JWT), que es un estándar basado en JSON para crear un token que sirve para enviar datos entre las aplicaciones, garantizando que sean válidos y seguros. Es decir, cuando un usuario quiere autenticarse, envía sus datos de inicio de sesión al back-end, que genera el JWT y lo devuelve a la aplicación cliente (front-end o modulo móvil). Luego, en cada petición, el cliente envía este token que se utiliza para verificar que el usuario está debidamente autenticado y poder identificarlo. Un JWT, tiene una estructura definida que se basa en tres partes, la primera de ellas se denomina “header” e indica el algoritmo (SHA-256) y el tipo de token, la segunda de ellas es el “payload” y contiene los datos del usuario, por último la tercera se conoce como “signature” y corresponde a la firma para verificar que el token es válido (auth0.com. 2020).
- Autorización: para acceder a los diversos recursos, se define un esquema de roles asociados a cada usuario, cuyo detalle se encuentra definido en la sección 3.4.1.2
- Auditoria: con el objetivo de verificar el correcto funcionamiento de las políticas de seguridad, y además poder conocer el detalle de los usuarios responsables de las transacciones, se guarda un registro de auditoria de las distintas transacciones que se realizan en el sistema, el detalle especifico de este módulo se define en la sección 3.7.
- Realización de copias de seguridad: en la implementación en el servidor de demo, se define un esquema de generación de copias de respaldo que se realiza diariamente.

3.1.5. Licenciamiento

El software “SAICo”, se desarrolló dentro del marco de este Trabajo Final de Maestría como un proyecto “open source”, es decir, de código abierto y con acceso libre para beneficio de la comunidad científica y académica. Dentro de este marco de trabajo, se establecen las bases y funcionalidades para la “Community Edition”, posibilitando la derivación a futuro de una versión “Comercial.”

El código fuente de la plataforma, se encuentra disponible en cuatro repositorios públicos de GitHub, divididos de acuerdo al nombre del módulo que representan:

- SAICo_BackEnd
- SAICo_FrontEnd
- SAICo_Mobile
- SAICo_FileManager

El sistema se distribuye bajo licencia Apache 2.0, creada por The Apache Software Foundation y muy utilizada en el mundo del software libre. Proporciona libertad para el uso, distribución, modificación y redistribución de versiones modificadas del software, sin tener que pagar regalías, siempre y cuando se incluya un aviso para notificar que en la distribución se está usando código con licencia Apache. Es decir, es bastante permisiva ya que no incluye que las versiones derivadas se distribuyan bajo la misma licencia (ASF. 2019).

3.2. Arquitectura de software

SAICo está compuesto por una arquitectura dividida en varios módulos, cada uno de estos cumple una función específica dentro de la plataforma. Dicha división está pensada para dar una mejor respuesta a cada uno de los requerimientos, obteniendo como resultado una estructura modular y escalable.

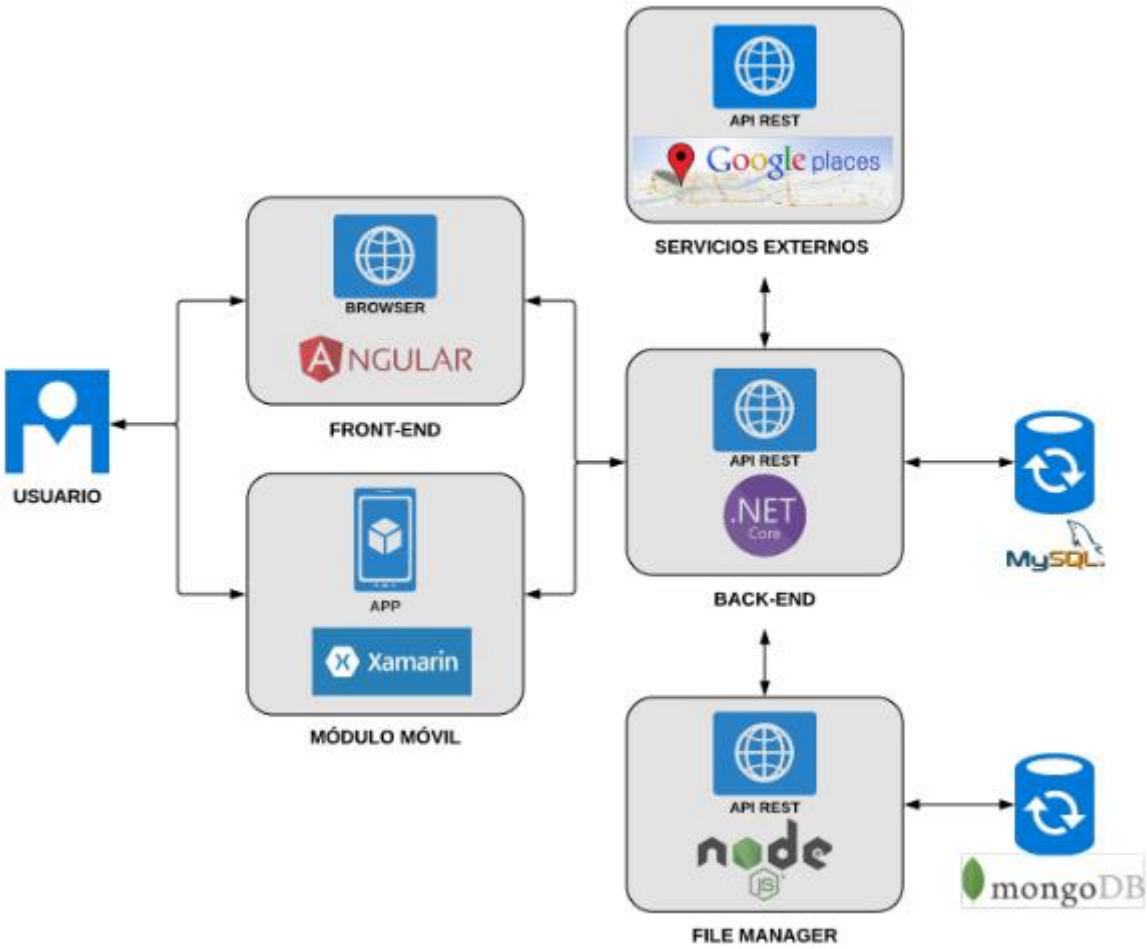

Figura 3.2.1: Arquitectura de software

3.2.1. Front-end

Este módulo está destinado a la interacción con el usuario, es decir, incluye todo lo que se ve por pantalla cuando se accede a la web: tipos de letra, colores, efectos del cursor, desplazamientos, efectos visuales, etc.

Esta parte del software es esencial para que el usuario tenga una buena experiencia con el sistema. Por este motivo, cada uno de los componentes involucrados fue desarrollado pensando en la simplicidad, para obtener como resultado una interfaz de usuario clara y amigable. Como muestra del diseño utilizado, se incluyen algunas

imágenes de las pantallas principales del front-end, los conceptos que se visualizan en cada una de ellas son explicados en detalle en la sección 3.4.

En la figura 3.2.1.1 se observa una barra de menú lateral que permite el acceso a cada uno de los distintos módulos del sistema, esta barra se puede expandir para visualizar los textos de cada uno de los botones o contraer para que ocupe menos porcentaje de la pantalla, dejando visible únicamente los botones de acceso. Por otro lado, en el margen superior del contenido principal de la página, se ubica una barra de búsqueda que permite filtrar los ítems visualizados. Con respecto a los elementos de información, se utiliza un diseño basado en “cards” para mostrar los datos de cabecera. Además, en la parte inferior de cada “card”, se ubican los botones de acciones que aplican a nivel individual, es decir, cada botón realiza una operación únicamente sobre el elemento desde el cual fue accionado:

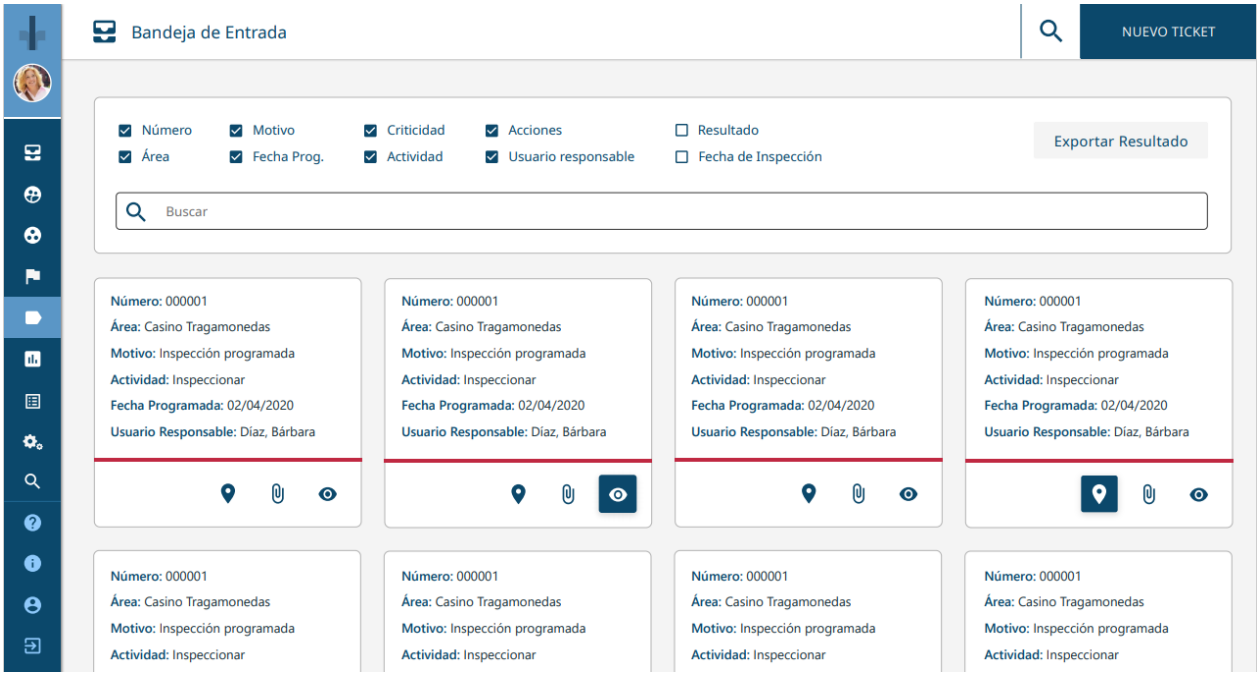

Figura 3.2.1.1: Bandeja de entrada

El formulario que se visualiza en la Figura 3.2.1.2, corresponde a la carga de uno de los elementos de información, que se encuentran contenidos dentro de “cards” en la figura 3.2.1.1. Para este caso, se realiza una agrupación de los ítems de información en

distintas secciones, que están divididas de acuerdo a su funcionalidad en el sistema para que su carga sea intuitiva. Algunas de estas secciones son colapsables, es decir, permiten que el usuario oculte rápidamente su contenido con el objetivo de reducir la cantidad de elementos desplegados en pantalla, para focalizarse únicamente en aquellos componentes sobre los cuales se va a trabajar:

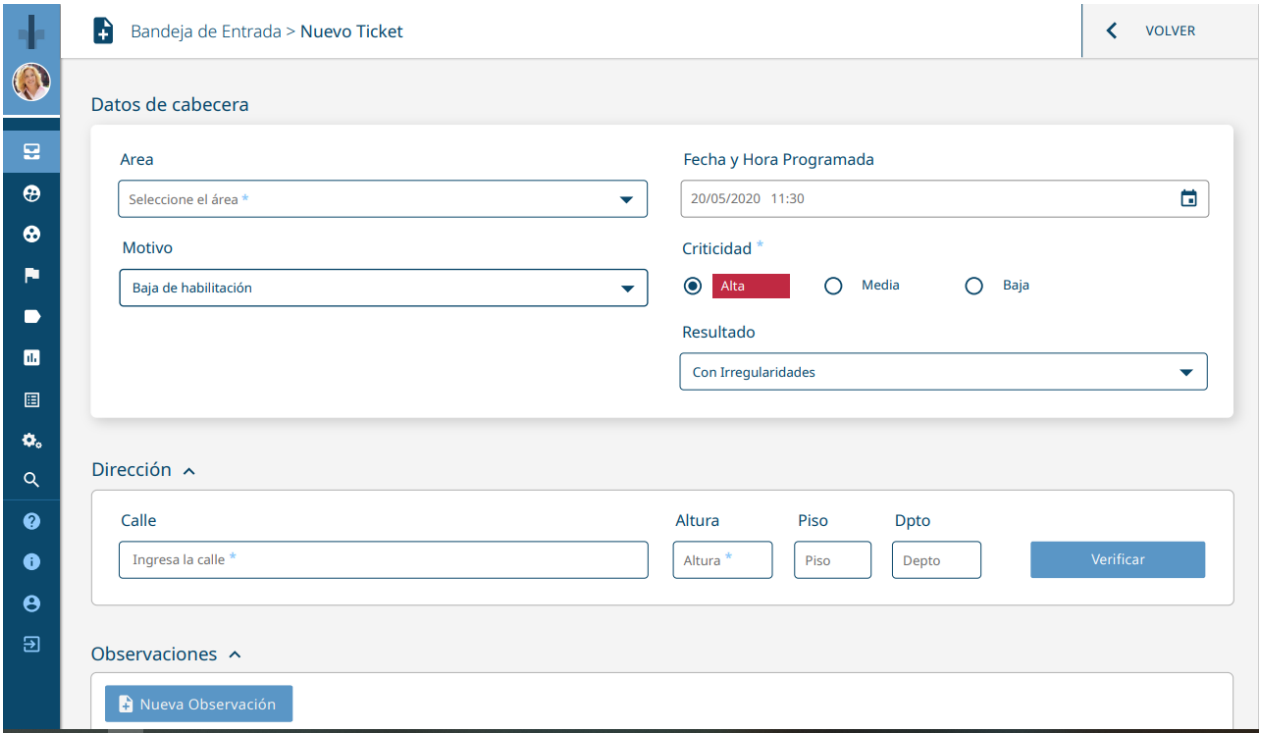

Figura 3.2.1.2: Formulario nuevo Ticket

Como confirmación de la carga del formulario, se muestra el mensaje de la figura 3.2.1.3, donde se incluye un resumen de la información ingresada, con el objetivo que el usuario pueda validar fácilmente los datos ingresados, antes de pasar a la instancia siguiente:

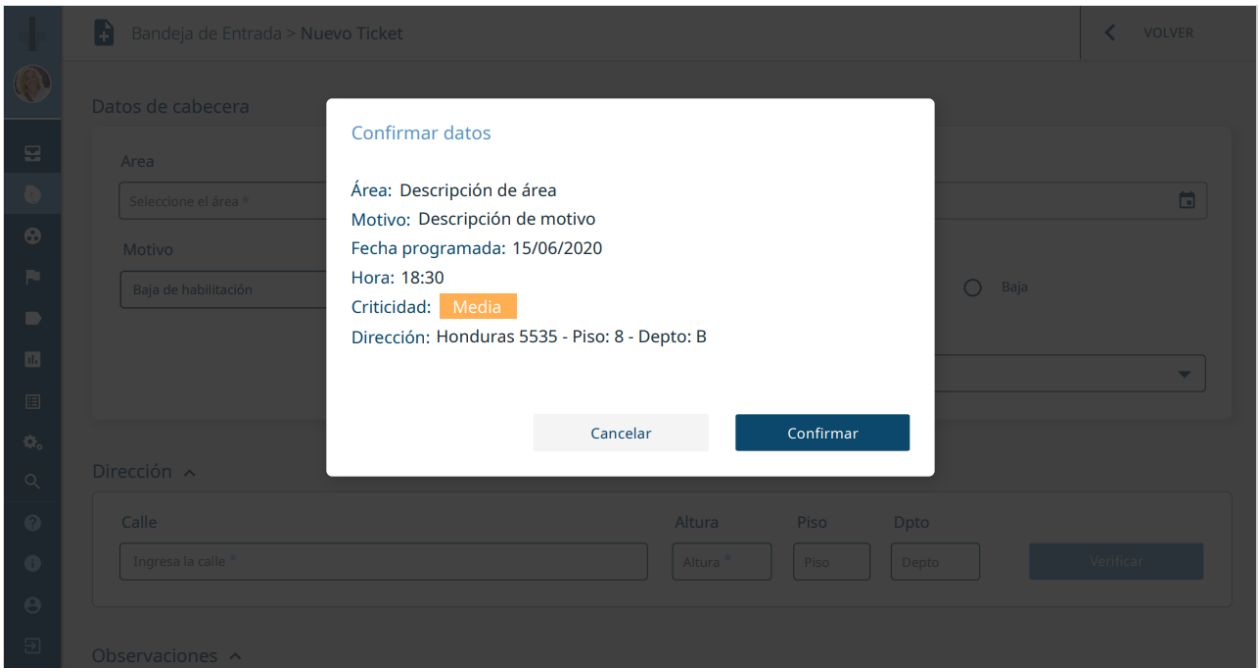

Figura 3.2.1.3: Confirmación nuevo Ticket

Por último, en la figura 3.2.1.4, se incluye una pantalla que refleja el estilo y la disposición de elementos utilizados en todos los módulos de administración del sistema. Si bien el contenido específico de las propiedades que se muestran en pantalla, dependerá de las características propias de la entidad que se esté administrando, en todos los módulos se respeta el mismo componente visual basado en “cards”, la misma ubicación de los botones para aplicar acciones, etc. Esta homogeneidad en la disposición de los elementos, fue realizada con el objetivo que el usuario se familiarice respecto a la distribución de la información:

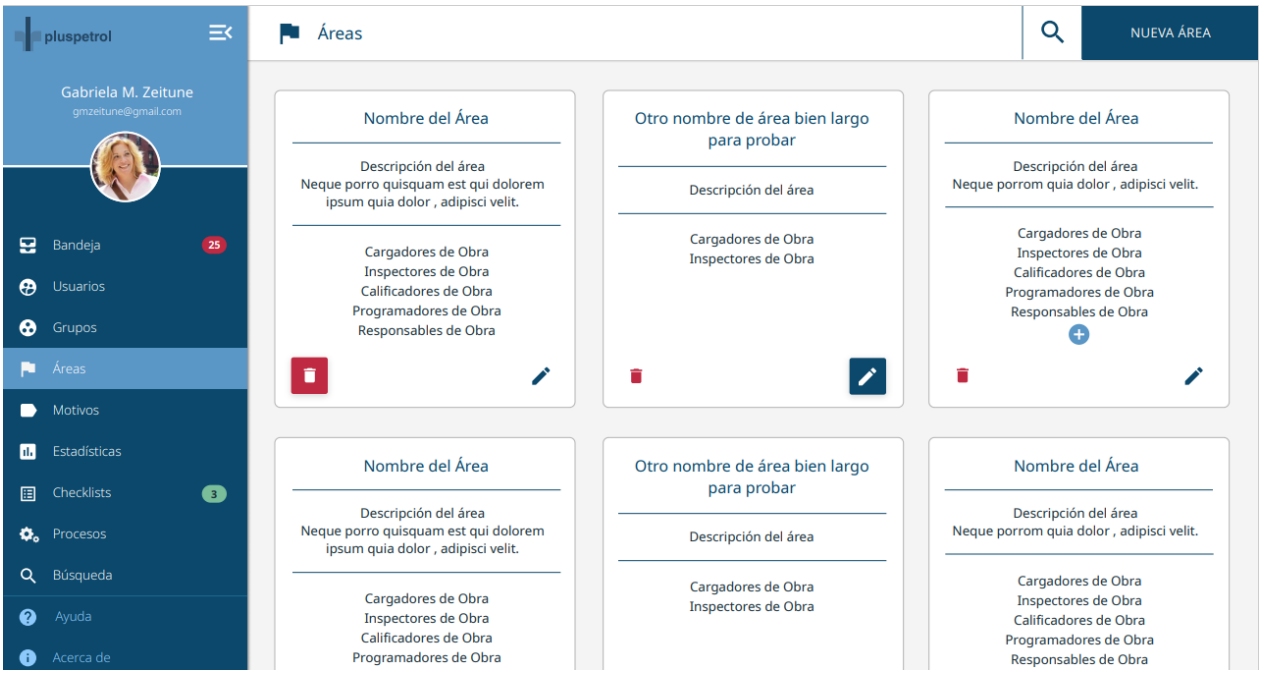

Figura 3.2.1.4: Administración de Áreas

3.2.2. Back-end

Es la capa que se encuentra por debajo del front-end, a la cual los usuarios finales no tienen acceso. En esta parte de la arquitectura se incluyen las reglas de negocio, la seguridad, los roles de los usuarios, etc.

Este módulo es el intermediario entre la comunicación que se realiza entre la aplicación Mobile y el front-end del módulo web, contra la base de datos. Además, resuelve la interacción con los diversos sistemas externos, a través de los cuales se envía o se recibe información.

3.2.2.1. “Clean Architecture”

Con el objetivo de facilitar el entendimiento, la escalabilidad y futuras modificaciones de frameworks y tecnologías de base de datos, para el proyecto del back end se utiliza una arquitectura “Clean”.

Como plantea Martin (2017), “The Clean Architecture” establece un conjunto de delimitaciones en cuanto a las responsabilidades de las capas de la aplicación, y plantea una dependencia que va de afuera hacia adentro, es decir, las capas exteriores pueden depender de las capas interiores pero no al revés. El objetivo es evitar el acoplamiento a un determinado framework o una base de datos, facilitando la realización de cambios en estas estructuras. En la figura 3.2.2.1 se observan las capas definidas para este tipo de arquitectura:

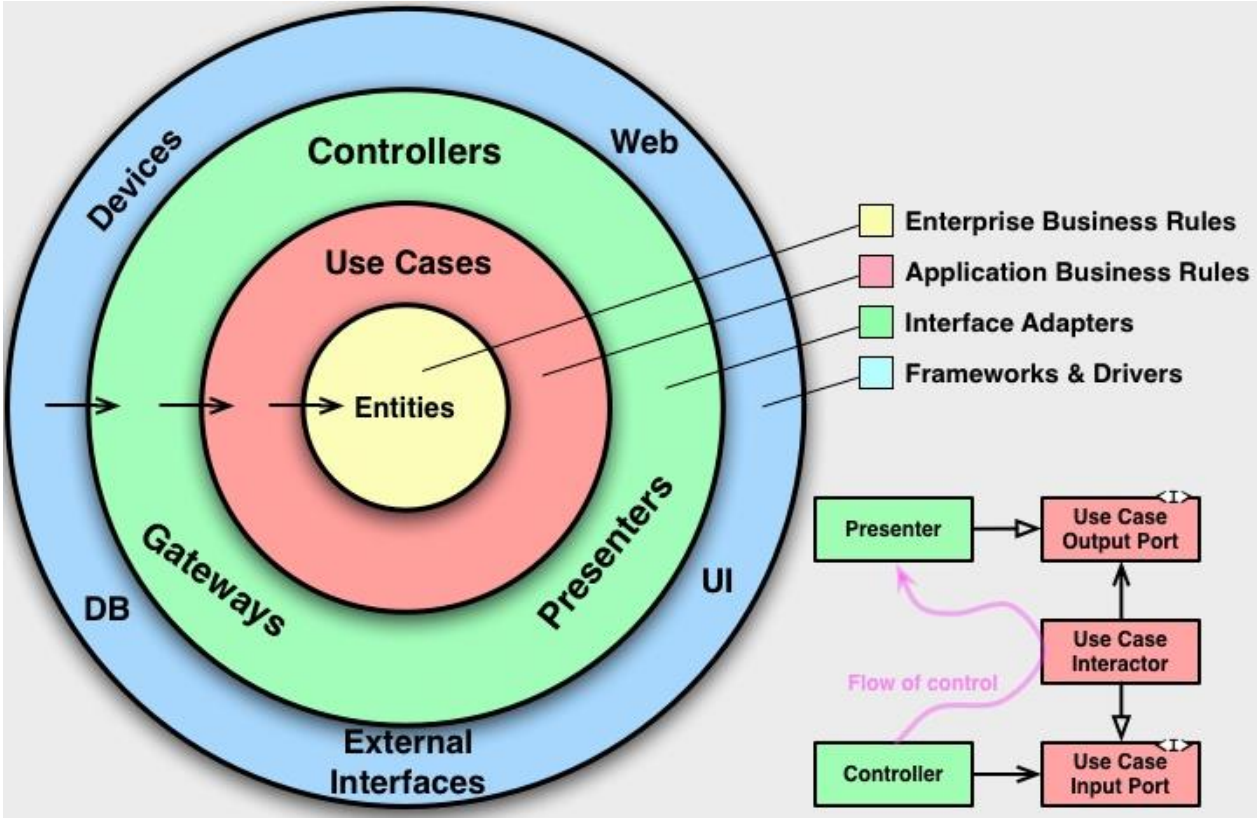

Figura 3.2.2.1: Arquitectura Clean (Martin. 2017)

Como se puede apreciar en la imagen, el dominio, donde se definen las entidades, es la capa más importante de la que dependen todas las demás, pero a su vez el dominio no depende de ninguna. La comunicación entre estas capas se consigue haciendo uso de del principio SOLID de Inversión de dependencia. Como menciona Martin (2017), cada una de sus letras tiene el siguiente significado:

S – Single Responsibility Principle (SRP): una clase debe tener una sola razón para cambiar.

O – Open/Closed Principle (OCP): el comportamiento de una clase se debe poder extender sin modificarla.
L – Liskov Substitution Principle (LSP): los objetos deben poder reemplazarse por instancias de sus subtipos sin alterar el buen funcionamiento del software.
I – Interface Segregation Principle (ISP): ninguna clase debería depender de métodos que no usa. Es importante que todas las clases que implementan una interfaz, sean capaces de agregar comportamiento a todos los métodos, en caso contrario, es conveniente tener varias interfaces más pequeñas.
D – Dependency Inversion Principle (DIP): permite que el núcleo de la aplicación no dependa de los detalles de implementación. Para conseguir esto, los módulos de alto nivel no deberían depender de los módulos de bajo nivel, ambos deberían depender de las abstracciones. Por otro lado, las abstracciones no deberían depender de los detalles, debería ser al revés.

Como resultado de aplicar esta arquitectura y estos principios, se obtienen las siguientes ventajas:

- Las reglas de negocio son independientes de frameworks, es decir, no están acopladas a librerías, lo que permite utilizar estas librerías como herramientas que son fácilmente sustituibles.
- Las reglas de negocio son fácilmente testables sin utilizar la interfaz de usuario, base de datos, servidor web.
- La interfaz de usuario es fácilmente modificable, sin afectar a las capas más cercanas al núcleo.
- Es fácil sustituir una base de datos por otra sin afectar a las reglas de negocio.

3.2.3. Módulo Móvil

Este módulo, al igual que el front-end, se comunica directamente con el back-end para acceder a la base de datos y a las distintas funciones que ofrece el sistema.
La aplicación móvil, permite realizar las mismas acciones que el modulo front end con respecto a la creación, edición y transición de tickets. Sin embargo, en esta aplicación

no se incluyen las funciones de administración de la plataforma, dado que su objetivo está orientado al trabajo en campo.

En la figura 3.2.1.1, se observa la barra de menú lateral, donde se incluye un botón de acceso a cada una de las funcionalidades disponibles, junto con la imagen personal y los datos que identifican al usuario que tiene sesión iniciada. Además, para mejorar la familiarización y personalización del usuario con la herramienta, se incluye el logo del organismo especifico que está utilizando la aplicación:

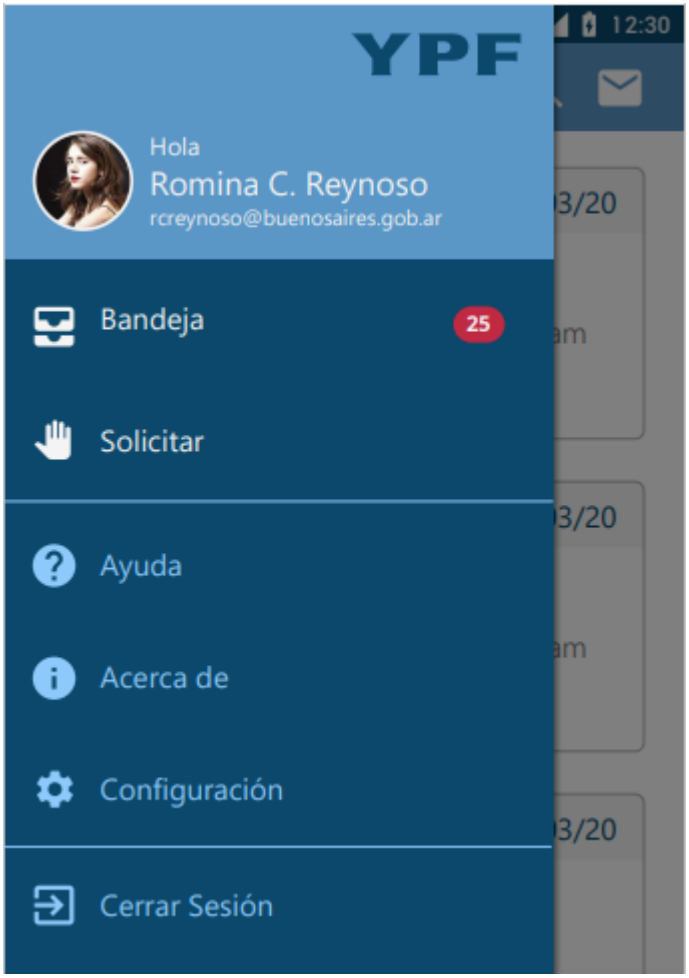

Figura 3.2.3.1: Menú lateral módulo Móvil

La pantalla principal del módulo Móvil, es la bandeja de entrada que se observa en la figura 3.2.3.2, donde se accede a los elementos de trabajo propios de cada usuario. Al igual que en front end, se utilizan “cards” para agrupar la información de cada elemento.

Con el objetivo de destacar los ítems más importantes se utiliza una paleta de colores del estilo “semáforo”, que se observa en el margen superior izquierdo de cada “card”. Por otro lado, en el margen superior de la pantalla, se incluye una barra de búsqueda rápida, que permite filtrar los elementos visualizados:

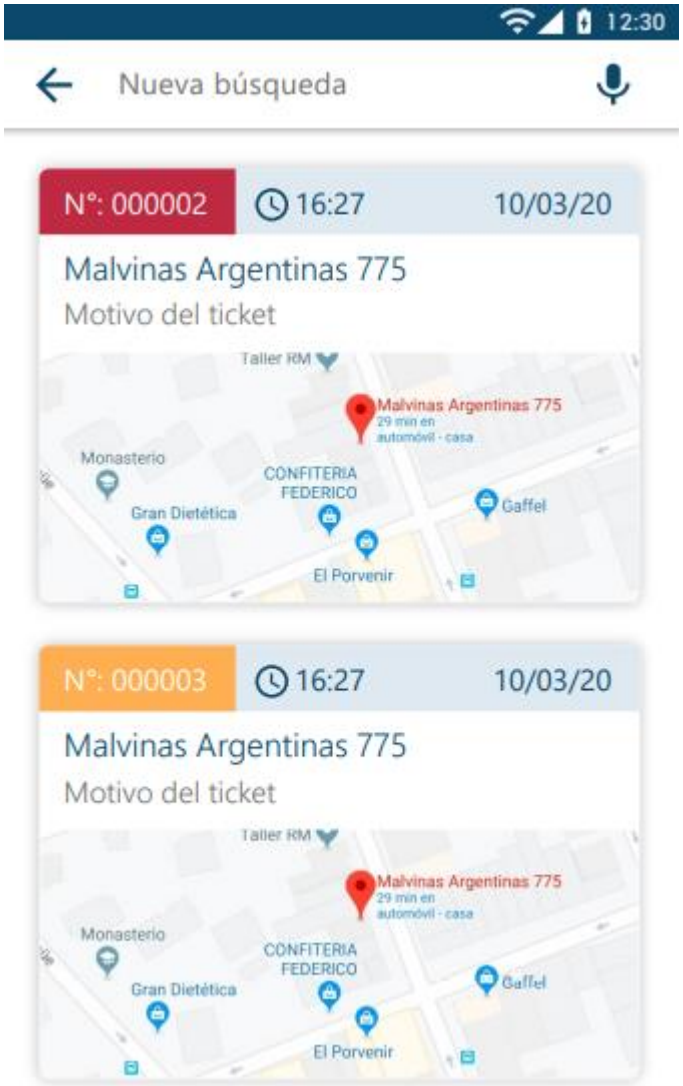

Figura 3.2.3.2: Bandeja de entrada módulo Móvil

Por último, cabe destacar que el modo de funcionamiento de la aplicación móvil es “online”, es decir, las operaciones son realizadas en el momento, asumiendo que se dispone de conectividad a internet.

3.2.4. File Manager

Este módulo es el responsable de guardar los distintos tipos de archivos que se cargan en el software, identificándolos de manera univoca. También permite la lectura de los mismos, utilizando para esto el identificador asignado previamente.

Se decide implementarlo como una estructura independiente, dado que se encuentra desarrollado en una tecnología diferente a la del back-end, especialmente seleccionada para el manejo de archivos (ver sección 3.1.4).

3.2.5. Integración con Google Maps

Para obtener los datos específicos de la ubicación, se utiliza la plataforma de “Google Maps”, mediante el servicio denominado “Places”. De acuerdo a Google (2020), dicha herramienta posee las siguientes características:

- Permite acceder al detalle de más de 150 millones de puntos de interés distribuidos en todo el mundo.
- Recibe constantes actualizaciones diarias.
- Tiene una infraestructura escalable, de acuerdo a las necesidades de uso.
- Ofrece diferentes funcionalidades, como el acceso a la ubicación actual, conversión de direcciones en coordenadas, autocompletado de direcciones, etc.

Con respecto a los costos de utilización, Google (2020) ofrece un crédito gratis de 200 USD al mes, sobre los cuales se ira restando saldo, de acuerdo a la cantidad de consultas que se vayan realizando, estos valores pueden observarse en la tabla 3.2.5.1.

Servicio	Costo en USD (por 1000 solicitudes)
Geocoding	5
Autocomplete	17
Place Details	17
Find Place	17
Place Photos	7
Current Place	30
Geolocation	5
Time Zone	5

Tabla 3.2.5.1: Costos Servicio Places de Google (Google. 2020)

Considerando esta escala de valores, la cantidad de consultas que se pueden realizar de forma gratuita utilizando el crédito mensual y, el contexto de este trabajo, se dispone de una cantidad suficiente de consultas, para poder utilizar el servicio tanto para desarrollo como para demo sin incurrir en gastos adicionales.

3.3. Tecnologías seleccionadas

La plataforma de software propuesta, está compuesta por diversos módulos, cada uno de ellos utiliza un conjunto específico de tecnologías, que fueron seleccionadas de acuerdo a las características y condiciones de uso que ofrecen.

3.3.1. Angular

Para la parte del front-end, se decide utilizar Angular, que es un framework desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, que se destacan por su simplicidad y eficiencia (Acosta. 2019).

De acuerdo a Wohlgethan (2018), las principales características de este framework son las siguientes:

- Es open-source.

- Utiliza TypeScript como lenguaje principal, en reemplazo de JavaScript.
- Utiliza la arquitectura Modelo-Vista-Controlador (MVC).
- Posee una interfaz de línea de comandos.
- Utiliza un archivo independiente para especificar las rutas, gestionando los componentes a mostrar según la URL activa.
- Es estable, teniendo en cuenta su política respecto al manejo de las nuevas versiones y, que es promovido y desarrollado por Google.
- Tiene una curva de aprendizaje un poco elevada hasta conocerlo completamente, pero no para desarrollar y configurar un proyecto básico.

3.3.2. Net Core y MySQL

Para el desarrollo de back-end se elige como lenguaje de programación C#, que se utiliza con el framework .NET Core, cuyas características principales según Microsoft (2020) son las siguientes:

- Código abierto: con licencias de MIT y Apache 2. Es un proyecto de .NET Foundation.
- Multiplataforma: tiene compatibilidad con los sistemas operativos Windows, macOS y Linux.
- Implementación flexible: se puede instalar la plataforma como una instalación separada o como parte propia de la aplicación. Su diseño modular implica que solo incluyas las dependencias que necesites.
- Moderno: utiliza conceptos actuales como programación asincrónica
- Herramientas de línea de comandos: incorpora estas herramientas sencillas que se pueden utilizar para el desarrollo local y la integración continua (Microsoft. 2020).
- Rendimiento: proporciona alto rendimiento.
- Es perfectamente compatible con el framework seleccionado para el front-end, AngularJS.

Con respecto al motor de la base de datos que tendrá interacción con este back-end, se decide utilizar MySQL, que es el RDBMS (sistema de gestión de bases de datos relacionales) más extendido en la actualidad, por tal motivo cuenta con una comunidad muy grande que ofrece soporte a otros usuarios (MySQL. 2020). Las principales características que detalla Robledano (2019), son las siguientes:

- Desarrollado inicialmente por MySQL AB, fue adquirida por Sun MicroSystems en 2008 y está su vez comprada por Oracle Corporation en 2010, que ya disponía de un motor propio llamado InnoDB para MySQL.
- Código abierto: se ofrece bajo la licencia GNU GPL, además también tiene una versión comercial gestionada por la compañía Oracle, pero este último caso no es el que involucra a este trabajo.
- Arquitectura Cliente y Servidor: basa su funcionamiento en un modelo cliente y servidor. Es decir, clientes y servidores se comunican entre sí de manera diferenciada para un mejor rendimiento.
- Compatibilidad con SQL: este lenguaje está muy generalizado dentro de la industria.
- Vistas: Desde la versión 5.0 de MySQL se ofrece compatibilidad para poder configurar vistas personalizadas, este recurso se vuelve imprescindible en bases de datos de gran tamaño.
- Procedimientos almacenados: no procesa las tablas directamente, sino que a través de procedimientos almacenados es posible incrementar la eficacia de la implementación.
- Desencadenantes: permite automatizar ciertas tareas dentro de la base de datos. Es decir, en el momento que se produce un evento otro es lanzado para actualizar registros u optimizar su funcionalidad.
- Transacciones: el sistema de base de registros comprueba que todo el proceso se realice correctamente o en caso contrario se prefiere preservar la integridad de la base de datos.

3.3.3. Xamarin

Para el desarrollo de la app móvil, se plantea la utilización de una herramienta que permita crear tanto aplicaciones para Android como iOS, con el objetivo de obtener un software que sea compatible con la mayoría de los dispositivos disponibles en el mercado.

En este sentido, Xamarin es una herramienta utilizada para el desarrollo de aplicaciones multiplataforma que permite a los desarrolladores compartir alrededor del 90 por ciento del código entre las principales plataformas. Sus principales características son:

- Lenguaje C#: utiliza un único lenguaje, C#, para crear aplicaciones de todas las plataformas móviles. Las aplicaciones compiladas mediante Xamarin aprovechan la aceleración de hardware específica de la plataforma y se compilan para tener un rendimiento nativo. Esto no se consigue con soluciones que interpretan el código en tiempo de ejecución (Microsoft. 2018).
- Basado en.NET framework. C# es un lenguaje muy desarrollado y como es uno de los lenguajes del framework.NET, es compatible con un gran número de características útiles de .NET como Lambdas, LINQ, y programación Asynchronous (Asynk).
- Compilación. La plataforma cuenta con dos productos principales: Xamarin.iOS y Xamarin.Android. En el caso de iOS, el código fuente se compila directamente en código ARM nativo (compilación Ahead-of-Time), mientras que las aplicaciones Xamarin para Android se compilan primero en Lenguaje Intermedio y posteriormente en AOT. Sin embargo, en ambos casos el proceso está automatizado y adaptado para solucionar problemas como la asignación de memoria. (APM. 2017)
- Es de código abierto y está disponible con licencia MIT, es decir, los SDK de Xamarin y el entorno de ejecución Mono son de código abierto en GitHub (Microsoft. 2018).

3.3.4. Node JS y MongoDB

Con el objeto de obtener un mejor rendimiento al manejar múltiples solicitudes, para el caso del File Manager se decide emplear tecnologías diferentes a las anteriores, que responden mejor a las necesidades de este módulo.

Por tal motivo se decide utilizar Node JS, que es un entorno en tiempo de ejecución multiplataforma cuyas principales características, según mencionan Ojamaa y Düüna (2012), son las siguientes:

- Es de código abierto
- Usa Javascript como lenguaje de programación
- Utiliza un solo hilo para ejecutar la aplicación, que maneja todas las solicitudes que ingresan, por lo cual permite simplificar la implementación de servicios rápidos y escalables
- Maneja los recursos de una forma más eficiente que aquellos modelos que utilizan subprocesos, ya que utiliza entradas y salidas asíncronas que pueden ejecutarse concurrentemente
- Emplea una máquina virtual confiable y extremadamente rápida

Por otra parte, con respecto a la base de datos, se toma la determinación de utilizar MongoDB, que utiliza un modelo NoSQL. Tal como describen Satheesh y otros (2015), las particularidades de esta base de datos son las siguientes:

- Es de código abierto
- En lugar de guardar los datos en registros, los guarda en documentos que son almacenados en BSON, que es una representación binaria de JSON
- Al trabajar con JSON, proporciona una mayor flexibilidad ya que se utiliza el mismo formato de datos entre el cliente, el servidor y la base de datos.
- No es necesario seguir un esquema, es decir, los documentos de una misma colección, concepto similar al de una tabla, pueden tener distintos esquemas.
- Está diseñada para ser altamente escalable, tanto horizontal como verticalmente

- Como no existen los “Joins”, típicos de los modelos relacionales, no es recomendable utilizarla cuando se necesiten consultar datos vinculados entre varias colecciones. Como en nuestro caso solo vamos a almacenar “Archivos”, esta situación no representa un problema.

3.4. Conceptos fundamentales

Para poder adaptarse a las distintas situaciones, que tienen dependencia directa con el ámbito donde se implementa el software, se definen una serie de conceptos y módulos parametrizables, con el objetivo de proporcionarle al sistema un mayor grado de flexibilidad y amplitud.

3.4.1. Usuarios

Este módulo permite la administración de los diversos “Usuarios” y perfiles que utilizan el sistema.

Un perfil de usuario está compuesto por la siguiente información:

Propiedad	Descripción
Nombre	Información personal de cabecera
Apellido	Información personal de cabecera
Email	Se utilizará posteriormente para ingresar en la aplicación
Teléfono	Es opcional, puede ser un celular o un teléfono fijo
Imagen de perfil	Archivo que se muestra en las barras de menú para mayor personalización
Roles	Define los permisos del usuario en el sistema
Grupos	Permite asociar el usuario a los “Grupos” de trabajo cargados en el sistema

Tabla 3.4.1.1: Propiedades de los Usuarios

Cuando se da de alta un nuevo “Usuario”, el sistema genera una clave aleatoria que la envía a la casilla de email ingresada. De esta forma el único que tiene conocimiento de la contraseña, es el usuario que debe tener acceso.

Posteriormente, un usuario podrá editar ciertos datos de su cuenta, accediendo al panel de “Mi Perfil”. Sin embargo, esta opción no permitirá la modificación de información crítica como el email, los roles y los grupos. Estos datos solo son modificados por los “Administradores de Usuarios” del sistema.

3.4.1.1. Grupos

En este módulo se configuran los “Grupos” de trabajo, es decir, la agrupación de usuarios que comparten una misma función en el sistema. Bajo esta definición, un “Usuario” tiene asociados uno o varios “Grupos”, en donde cada uno de ellos le permite acceder a diferentes tipos de “Tickets” de trabajo.

Los “Grupos” de trabajo están directamente vinculados a las “Actividades” dentro de los diferentes “Procesos” o flujos de trabajo. Esta asociación le permite al sistema determinar que usuarios se encuentra habilitados para acceder a los “Tickets”, que se encuentran en una determinada “Actividad”.

3.4.1.2. Roles

Los “Roles” de un “Usuario”, definen los permisos para acceder a los distintos módulos y funcionalidades del sistema.

La funcionalidad del sistema se divide en cuatro “Roles” que se pueden observar en la tabla 3.4.1.2.1.

Rol	Descripción
Usuario Simple	Solo puede acceder a la bandeja de entrada para gestionar su trabajo (tickets)
Usuario Avanzado	Puede acceder a las funciones del usuario simple y además a las pantallas de búsqueda y estadísticas.
Administrador de Negocio	Puede usar todas las funciones de un usuario avanzado y, acceder a los ABMs de los módulos del dominio de negocio, es decir, aquellos que no son reservados para administración del sistema sino para la parametrización.
Administración del Sistema	Puede utilizar todas las funciones del software

Tabla 3.4.1.2.1: Roles

3.4.2.Tickets

El “Ticket” es el elemento básico de trabajo sistema, sobre el cual se cargan distintas propiedades con información. Es decir, es el contenedor elemental de información, que va pasando por distintas “Actividades”, incorporando en cada una de ellas información adicional para llegar a un estado final, en donde todos los datos requeridos fueron completados.

Con respecto al contenido del ticket, se deberá introducir la información mínima e indispensable, que está compuesta por las propiedades que se detallan en la tabla 3.4.2.1.

Propiedad	Descripción
Área	Solo son elegibles aquellas que están vinculadas a los “Grupos” asociados a la “Actividad de Inicio” del “Proceso” actual
Motivo	Solo son elegibles aquellos que estén asociados al “Área” de trabajo actual o que estén libres (sin vinculación a un área específica)
Fecha y Hora Programada	En formato dd/mm/yyyy hh:mm
Criticidad	Permite seleccionar entre 3 valores prestablecidos (Alta, Media o Baja)
Ubicación	Permite ingresar información relacionada a la geolocalización
Archivos adjuntos	Se pueden asociar todos los tipos de archivos básicos, como PDF, DOC, DOCX, XSLSX, JPG, JPEG, PNG, etc
Resultado	Es la conclusión final con la que se cierra el “Ticket”

Tabla 3.4.2.1: Propiedades de los Tickets

Además de esta información mínima que se solicita de forma fija, cada “Ticket” puede tener definido un “Elemento” con propiedades dinámicas, que se configuran en el módulo de “Proceso” (ver sección 3.4.4). Es decir, que dependiendo de esta configuración, se solicitan adicionalmente más datos, que se irán mostrando en las distintas “Actividades”, según corresponda.

3.4.2.1. Áreas

Este módulo permite definir las distintas “Áreas” de trabajo que operan el sistema, cada una de ellas tiene asociado uno o más “Grupos” de usuarios.

Dependiendo de los “Grupos” a los que pertenezca el usuario que está cargando o editando el “Ticket”, se filtra el listado de “Áreas” seleccionables, para que solo se visualicen aquellas que vinculadas a sus “Grupos” de trabajo.

3.4.2.2. Motivos

Este concepto permite definir cuál es la motivación que originó la realización del “Ticket” de trabajo. Por ejemplo, en el caso de un sector que se dedique a la supervisión de obras en construcción, lo motivos podrían ser: “Control Obra Etapa I”, “Control Obra Etapa II”, “Control Obra Etapa III”, etc.

Los “Motivos” tienen asociación con las “Áreas” trabajo, es decir, mediante esta vinculación puede definirse que un “Motivo” de trabajo solo sea visible para un “Área”. En cambio, si no se carga ninguna vinculación “Motivo”-“Área”, se considera que el “Motivo” es libre para cualquier carga de tickets que se efectué en el sistema.

3.4.2.3. Criticidades

Este tipo de dato está destinado a destacar fácilmente el “Ticket”, para poder dar un tratamiento diferenciado, en caso de que sea requerido. Presenta tres valores posibles:

- Alta
- Media
- Baja

Se considera que, con estas opciones, se abarcan la mayoría de los casos, por tal motivo, no se estima necesario desarrollar un módulo específico para administrar cambios.

3.4.2.4. Ubicaciones

Esta entidad contiene a su vez múltiples propiedades, relacionadas a la localización donde debe hacerse la inspección.

Hay dos opciones posibles, dependiendo de la información disponible a la hora de la carga del “Ticket”:

- Ubicación definida: partir de una calle y una altura, que se validan a través del servicio de Google Maps, se obtienen los datos de la ubicación:
    - Calle
    - Puerta
    - País
    - Provincia
    - Ciudad
    - Barrio
    - Código Postal
    - Coordenada X
    - Coordenada Y
- Ubicación libre: simplemente se ingresa un texto que describe la ubicación y además permite cargar los datos de las coordenadas (en caso que se disponga de dicha información)

3.4.2.5. Observaciones

Por cada “Ticket” se podrán cargar múltiples “Observaciones”, registrándose por cada una de ellas la siguiente información:

- Fecha y Hora
- Usuario
- Texto descriptivo

3.4.2.6. Resultados

Este módulo permite administrar los distintos “Resultados” que quedan vinculados como conclusión en cada “Ticket”.

Cada “Resultado” puede estar asociado a una o más “Áreas”. De esta forma, ese “Resultado” solo será visible, cuando se esté trabajando con alguna de las “Áreas” vinculadas en la configuración.

3.4.2.7. Bandeja de entrada

Este módulo, le permite al usuario acceder rápidamente al trabajo que tenga asignado actualmente o bien generar un nuevo “Ticket” (en caso de que disponga de permisos).

Respecto a la primera funcionalidad, se muestran en la bandeja de trabajo todos los tickets activos que estén dentro de una actividad donde el usuario tenga permisos. El criterio para conformar la bandeja de trabajo del usuario es el siguiente:

- Tickets de pertenencia directa: son aquellos que lo tienen directamente como “Usuario Responsable”, siempre y cuando se encuentren en alguna de las actividades sobre las cuales el usuario tiene permisos.
- Tickets de pertenencia por grupo de trabajo: son aquellos que se encuentran en una actividad relacionada con algún “Grupo” del cual el usuario es miembro. Esta relación se configura desde el módulo de “Proceso”, mediante la asignación de “Grupos” en las distintas “Actividades”.

Respecto a la segunda funcionalidad, se habilita a dar de alta nuevos “Tickets”, solo si se dispone de permisos en la “Actividad de Inicio” de algún “Proceso”. Es decir, que dinámicamente se muestra la opción para cargar nuevos “Tickets”, dependiendo del perfil del usuario. En caso de disponer de privilegios para cargar “Tickets” en más de un “Proceso”, el sistema muestra el listado de opciones disponibles.

3.4.3. Checklists

Este módulo permite configurar los distintos cuestionarios que definen la estructura y las condiciones de la información que se desea relevar, o lo puntos que se controlan durante la realización de una inspección o auditoria.

Se incluyen una gran variedad de propiedades y funcionalidades, que permiten que esta herramienta sea lo suficientemente flexible para adaptarse a distintas situaciones.

3.4.3.1. Categorías

Básicamente un “Checklist” está compuesto por una o más “Categorías”, cada una de las cuales posee una o más “Preguntas”. Dentro de esta estructura, se denomina “Categoría Inicial”, a aquella que contiene las “Preguntas” que se encuentran dentro de la raíz.

En la figura 3.4.3.1.1 se muestra un Diagrama de Entidad Relación, que refleja las asociaciones entre las clases vinculadas al “Checklist”.

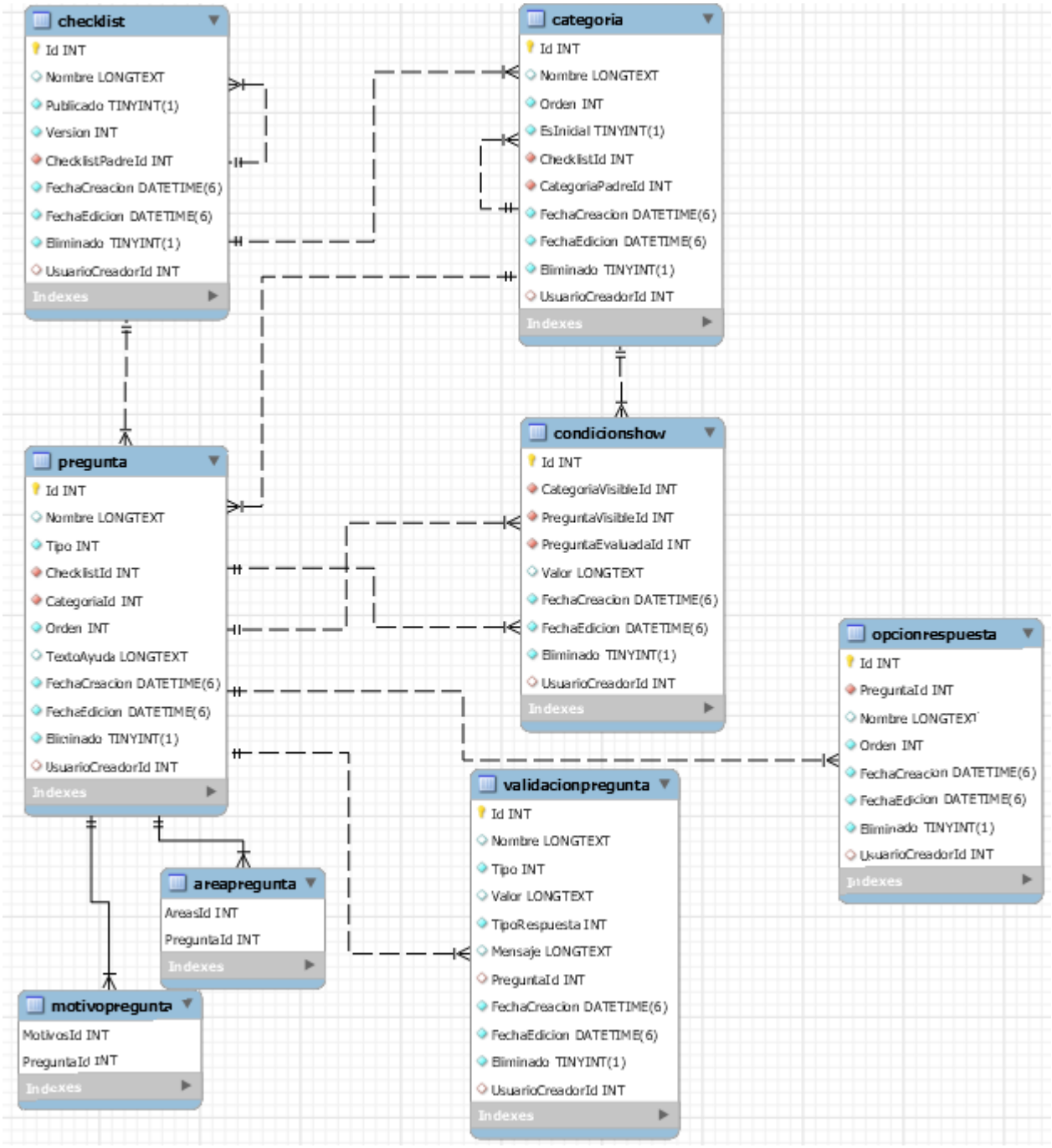

Figura 3.4.3.1.1: DER Checklist

3.4.3.2. Preguntas

Por cada “Pregunta” que forma parte de una “Categoría”, dentro del “Checklist”, se deben definir una serie de propiedades:

Propiedad	Descripción
Nombre	Es el texto que se visualiza como cabecera de la pregunta
Tipo de respuesta	Define el tipo de dato que se acepta como respuesta
Condiciones de visualización	Definen si la visualización de la “Pregunta”, depende de que se cumpla una o varias condiciones previas
Validaciones	Permite garantizar que la “Respuesta” obtenida cumpla con determinadas condiciones
Orden	Define la posición en la que se visualiza la “Pregunta”
Texto de ayuda	Permite ingresar información adicional que podría ser de utilidad para que el “Usuario” conteste la “Pregunta”. Dicha información es visualizada a demanda, mediante un botón de acceso rápido
Area y Motivo	Permite condicionar la visualización de la pregunta a estas dos propiedades, que forman parte de la cabecera del “Ticket”. Es decir, permite definir que una “Pregunta” sea visible solo si el “Ticket” tiene un “Motivo” o “Área” especifica
Valor por defecto	Su comportamiento varía de acuerdo al TipoRespuesta de la “Pregunta”, por ejemplo, si su tipo es “Decimal”, permite ingresar solo números

Tabla 3.4.3.2.1: Propiedades de las Preguntas

3.4.3.3. Publicaciones y versiones

El módulo de “Checklist”, tiene definido un esquema de versionado, que permite trabajar de forma segura durante la edición, y recuperar rápidamente una versión anterior en caso de detectar algún error o inconveniente en la nueva versión. Es decir, a medida que se edita la versión inicial de un “Checklist”, se agregan registros independientes, de forma que se puede publicar cualquier versión. Es decir, al editar un “Checklist”, el sistema no trabaja directamente sobre la base de datos sino en una copia en memoria, que recién se guarda al presionar al “Finalizar”. Cuando se realiza esta acción, el sistema genera una nueva versión de “Checklist”, donde todas las categorías y preguntas son creadas nuevamente para reflejar los últimos cambios ingresados.

Por cada versión de “Checklist” se dispone de un botón de “Publicar”, cuya acción es llevar esa versión de “Checklist” a producción. La versión inicial de un “Checklist” es la numero 1, posteriormente se empieza a incrementar el número en una unidad.

Desde la bandeja de entrada de “Checklist”, donde se enlistan todos los cuestionarios que se fueron cargando, se permite consultar todas las versiones existentes y, en caso de ser necesario, se puede editar y publicar cualquiera de ellas. Por cada “Checklist” puede haber más de una versión, pero solo una de ellas puede estar publicada.

Cuando se está creando/editando un “Checklist”, hay tres acciones posibles:

- Previsualizar: simula la ejecución del “Checklist” cargado hasta el momento.
- Publicar: envía a producción la versión seleccionada. Para poder aplicar esta acción, se detecta si se hicieron cambios sobre el “Checklist” en curso, en este caso previamente se genera la nueva versión y luego se publica. En cambio, si no se hicieron cambios sobre el “Checklist” actual, quiere decir que se puede tomar exactamente esa versión y publicarla, no hace falta generar una nueva.
- Finalizar/Guardar: si se hicieron cambios sobre el “Checklist” actual, se genera una nueva versión. Esta acción no envía el “Checklist” como “publicado”.

3.4.4. Procesos

El módulo de “Proceso” permite configurar el flujo de trabajo sobre el cual se gestiona el ticket. Es decir, vincula diferentes tipos de “Actividades” sobre las cuales va transitando el ticket, y las respectivas “Transiciones” entre ellas.

Por cada “Proceso”, se debe completar la siguiente información:

- Datos de cabecera: Nombre, Descripción, Tipo de Proceso
- Actividades y transiciones
- Elemento

Existen distintos “Tipos de Procesos” seleccionables, con el objetivo de realizar una rápida diferenciación entre los distintos flujos de trabajo, por ejemplo:

- Administrativo: el proceso no incluye una actividad de inspección, es decir, es un “Proceso” donde se gestiona información que puede estar relacionada a un acto control, pero no incluye la tarea específica de fiscalización.
- Inspectivo: indica que el proceso involucra una actividad de inspección/control, donde se realiza una fiscalización sobre una entidad.

3.4.4.1. Actividades y transiciones

Las “Actividades” son las distintas estaciones de trabajo por las que transita un “Ticket” a lo largo del “Proceso”. Están relacionadas entre sí, mediante el concepto de “Transición”, que básicamente es un nexo entre una actividad de origen y una de destino.

Cuando se finaliza un “Ticket”, si el software detecta que la “Actividad” actual tiene más de una “Transición” posible, se enlistan por pantalla las opciones disponibles para que el “Usuario” determine el camino a seguir. En cambio, cuando solo hay una “Transición” aplicable hacia la “Actividad” siguiente, el sistema realiza la acción automáticamente.

3.4.4.2. Elementos

A medida que se avanza en las “Actividades” del “Proceso”, se solicita al “Usuario” diversos tipos de datos correspondientes al “Elemento”, que es un contenedor de información especifico de cada “Proceso”.

Los datos a completar del “Elemento”, se muestran o se ocultan en función de la “Actividad” actual con el objetivo de segmentar el llenado de información, de acuerdo a los distintos perfiles de usuarios que van interviniendo a lo largo del flujo de trabajo.
Para definir cada una de las propiedades del “Elemento”, se debe registrar la siguiente información:

Propiedad	Descripción
Nombre	Es la denominación que se visualiza al solicitar el dato en el formulario
Tipo	Define el tipo de dato
Actividades	Define en que instancias del flujo de un “Ticket”, se muestra la propiedad para su carga

Tabla 3.4.4.2.1: Propiedades de los Elementos

Cuando las actividades llegan a su punto final, donde se dispone de un “Elemento” de información completo, se considera que dicha información pasa a ser “publica” dentro del sistema, pudiendo ser consultada por cualquier usuario.

3.5. Módulos parametrizables y algoritmos

Para dar respuesta a las distintas situaciones que se pueden plantear en el ámbito de una inspección, se seleccionan y desarrollan una serie de módulos parametrizables, de manera que el “Usuario” pueda cargar la información específica de su contexto, sin tener que recurrir a modificar directamente la base de datos o el código de la aplicación.

Por otro lado, para automatizar las “Actividades” mencionadas en la sección 2, se desarrolla un algoritmo de asignación de “Tickets” y un módulo de “Reglas” de análisis de resultados.

3.5.1. Selección de módulos

Los módulos seleccionados para aportar un mayor grado de apertura al software, son los siguientes:

- Área
- Grupo
- Motivo
- Checklist
- Resultado
- Usuario

Por cada uno de estos, el “Usuario” dispone de una interfaz gráfica para cargar, editar o borrar los registros asociados, de una forma ágil y rápida. Además, el acceso a cada uno de estos módulos, está validado según la lógica de roles explicada en la sección 3.4.1.2.

3.5.2. Algoritmo de asignación automática

Para contribuir en el trabajo de automatización de tareas, el sistema incluye un algoritmo que permite asignar “Tickets” de inspección a aquellos usuarios que “solicitan” trabajo. Es decir, al quedarse sin elementos de trabajo, un “Inspector” puede utilizar esta funcionalidad, que automáticamente le asignará nuevos “Tickets”.

Este algoritmo tiene en cuenta algunas de las propiedades cargadas para cada inspector, y determina cuales son los casos más críticos que aún están pendientes de resolver y, que aplican a las competencias del inspector. Para poder resolver esta situación, el sistema tiene en cuenta los siguientes factores:

- Ubicación
- Criticidad
- Fecha programada
- Áreas de especialidad del inspector
- Motivos de especialidad del inspector

Por cada inspector, se establece un punto de origen, que actúa como coordenada de referencia a partir de la cual se empieza a realizar la búsqueda. En una primera etapa, se buscan los casos que están en un radio de 2000 mts, si no se alcanza el total de “Tickets” buscados, comienza a incrementarse el radio, aumentando de a 500 mts hasta alcanzar la cantidad máxima de tickets de trabajo establecida para el usuario que realiza la solicitud. Dentro de este criterio de ubicación, se toman aquellos casos que tengan la “Criticidad” más alta, con una “Fecha Programada” más pronta a vencer. En caso de que el inspector tenga configuradas “Áreas” y “Motivos” en los que es especialista, se les dará preferencia a los casos detectados que más se ajusten a estas propiedades. Dentro de esta ponderación, si no se configuran “Áreas” y “Motivos” de especialidad, el sistema asume que el “Inspector” puede tratar “Tickets” de cualquier tipo. En cambio, si se configuran dichas propiedades, el sistema le otorga trabajo específicamente para la especialidad configurada, priorizando esto por sobre la ubicación.

3.5.3. Reglas de análisis de resultados

Estas reglas permiten dar respuesta automática, ante alguno de los “Resultados” que tiene un “Ticket” de trabajo. Es decir, una vez que son cargadas en el sistema, quedan esperando a que se cumplan los condicionales configurados para ejecutarse, desencadenando la acción definida.

Para fijar cada una de estas reglas, es necesario especificar una serie de atributos, que se detallan en la tabla 3.5.3.1

Propiedad	Descripción
Actividad de origen	Es el punto de partida, es decir, el lugar desde donde se ejecuta la regla. Por ejemplo: de acuerdo al circuito básico de control planteado en la figura 2.1, donde el objetivo es evitar la actividad de “Análisis” posterior a la “Inspección”, la actividad de origen tiene que ubicarse en “Archivado”, es decir, la instancia final del flujo de trabajo
Área	La carga de esta propiedad es opcional, permite limitar la ejecución de la regla teniendo en cuenta, además de la actividad de origen, un “Área” de trabajo específica
Motivo	Es exactamente igual que el caso anterior, pero teniendo en cuenta la propiedad “Motivo”
Resultados	Es de carácter obligatorio, indica cuales son los “Resultados” que deben obtenerse para que se dispare la regla
Tipo de acción	Existen dos operaciones posibles, la primera de ellas es la generación de un nuevo “Ticket”, ya sea dentro del mismo “Proceso” o en otro, en este caso habrá que especificar los datos de cabecera con los cuales se va a crear el nuevo “Ticket”. La segunda operación, corresponde a disparar una alerta a un determinado grupo de trabajo o un usuario. En este último caso, la notificación se recibe a través del email asociado a la cuenta de usuario del sistema, donde se envía adjunto el PDF del informe que se explica en la sección 3.5.1

Tabla 3.5.3.1: Propiedades de las Reglas de Análisis de Resultados

3.6. Informes, búsquedas y estadísticas

Lógicamente una de las finalidades principales de realizar inspecciones o ejercer un control a través de un software, es obtener estadísticas y realizar un monitoreo que permita conocer en detalle las fortalezas y debilidades del esquema de trabajo actual, para realizar ajustes que posibiliten mejorar la eficiencia del modelo utilizado.

Para permitir la consulta, visualización y exportación de la información, tanto de la actual como de la histórica, SAICo incluye distintos módulos que ofrecen estas funcionalidades.

3.6.1. Informe PDF

En cualquier instancia del flujo de trabajo, se puede obtener un informe PDF con la información cargada hasta ese momento. Estos datos están organizados en diferentes secciones:

- Datos de cabecera: Área, Motivo, Fecha, Criticidad, Ubicación, Resultado
- Elemento: abarca toda la información configurada para el elemento del flujo de trabajo actual
- Checklist y datos del inspector (en caso de corresponder): estos datos son aplicables en el caso de los procesos del tipo “Inspectivo”.

Este PDF es generado temporalmente, a modo de que el usuario pueda compartirlo a través de diversos medios (email, whatsapp, etc). Sin embargo, en la actividad final, donde se dispone de toda la información necesaria para completar cada una de las secciones del informe, se generar un archivo que quedará adjunto en el ticket de trabajo. De esta forma, cuando se consulta un ticket finalizado, se puede acceder rápidamente al reporte con toda la información necesaria de manera resumida.

3.6.2. Búsqueda y exportación

El objetivo de este módulo, es permitir la consulta de registros históricos (tickets cerrados) y registros actuales (tickets aún no finalizados). En esta sección del software, el “Usuario” realiza distintos tipos de búsquedas en base a filtros predefinidos, y luego se habilita la exportación de los resultados obtenidos a un documento Excel.

Existen 3 tipos de búsquedas permitidas en el sistema:

- Por Datos de cabecera: Número de ticket, Área, Motivo, Criticidad, Fecha programada, Resultado.
- Por Ubicación: Calle, Altura, Fecha programada, Usuario responsable.
- Por Proceso: Proceso, Actividad, Fecha programada, Usuario responsable.

3.6.3. Estadísticas

Para monitorear y obtener reportes dinámicos y parametrizables, SAICo incorpora una sección de estadísticas, donde se incluyen funcionalidades que contribuyen con estas tareas:

- Tiempo promedio por actividad: seleccionando un “Proceso” como filtro, se muestra el tiempo promedio que permanecen los “Tickets”, en cada una de las actividades del flujo de trabajo seleccionado.
- Ranking de inspectores: muestra una tabla de todos los inspectores con la cantidad de “Tickets” finalizados por cada uno de ellos. Adicionalmente permite agregar como filtro un “Área”, una “Fecha Desde” y una “Fecha Hasta” para acotar la evaluación a periodos y grupos de trabajo específicos.
- Mapa de inspecciones: despliega un plano donde se identifican los distintos puntos de inspección. Permite realizar un filtro, para obtener detalles específicos sobre una determinada fecha, un determinado estado del “Ticket” o un “Área” de trabajo en particular. Por cada objetivo identificado, se puede acceder a la información asociada como el número de ticket, inspector asignado, datos de la ubicación, etc.
- Ubicación de inspectores: exhibe un mapa donde se pueden visualizar las últimas ubicaciones detectadas de los inspectores. También permite realizar un filtro para obtener los datos de un “Usuario” específico o de un “Área” de trabajo en particular. El objetivo en este caso, es poder realizar un monitoreo para corroborar si los inspectores se desplazan según sus objetivos de trabajo.

3.7. Auditoria

Por cada entidad que crea en SAICo, se guarda un detalle completo de la transacción, a modo de poder verificar posteriormente que usuario realizó la carga. Por otra parte, también se guarda un registro si la entidad es modificada posteriormente. De esta forma se tiene una trazabilidad completa de los cambios que ocurren en el sistema, en caso que sea necesario realizar una revisión.

En la tabla 3.7.1 se puede ver el detalle de las propiedades almacenadas para el seguimiento de las transacciones que se realizan en el software:

Propiedad	Descripción
Fecha de creación	Es la fecha de carga de la entidad, en formato dd/mm/yyyy hh:mm:sss
Usuario creador	Incluye la información asociada al usuario que da de alta la entidad
Fecha de modificación	Es la fecha de modificación del registro, en formato dd/mm/yyyy hh:mm:sss
Usuario modificador	Incluye la información asociada al usuario que modifica la entidad
Entidad modificada	Se guardan en formato JSON todas propiedades correspondientes a la entidad modificada

Tabla 3.7.1: Propiedades de auditoria

4. ENTREVISTAS CON EXPERTOS

En las siguientes secciones, se incluye el análisis de las entrevistas realizadas a los expertos en sistemas de inspecciones y control. Como se indica en la sección 1.4, estas entrevistas fueron realizadas utilizando la metodología cualitativa semistructurada, tomando como referencia el cuestionario de preguntas que se detalla en el Anexo 8.2.1. Las entrevistas tienen dos objetivos principales: una primera instancia donde se busca conocer las falencias y debilidades de los sistemas conocidos por los expertos, y una segunda instancia donde se pretende validar la herramienta desarrollada en este Trabajo Final.

Estas entrevistas fueron grabadas con el consentimiento de los participantes, y dichas transcripciones se encuentran detalladas en el Anexo 8.2.2 y 8.3.3 respectivamente.

4.1. Análisis de entrevista a experto N° 1

La entrevista fue realizada el 26/10/2020 y tuvo una duración de 33 minutos. Se llevó a cabo manera remota, mediante una videollamada, donde se compartió la pantalla para exponer el primer prototipo del software.

Durante la primera parte, el entrevistado menciona que tiene una formación orientada a sistemas, que hace diez años que trabaja para la Agencia Gubernamental de Control (AGC), cuyas funciones son la realización de fiscalizaciones e inspecciones de todas las actividades comerciales de la Ciudad de Buenos Aires.

A lo largo de su experiencia en este organismo público, utilizo dos sistemas de inspecciones. El primero de ellos estaba orientado a una carga totalmente manual de información, donde utilizaban perfiles de “data entry”, en cambio, el segundo de ellos, más reciente, utiliza conceptos de estaciones de trabajo a través de las cuales fluye la información mediante un elemento contenedor. Los principales problemas que observa de ambos, son el alcance orientado únicamente a una registración de información, y por

otro lado la escalabilidad. Menciona además, que el sistema que utilizan actualmente sufrió una merma considerable en el desempeño, debido a la cantidad creciente de usuarios.

Debido a la cantidad de entes gubernamentales, que interactúan con el organismo para el cual trabaja el entrevistado, hace especial hincapié en la escalabilidad, la performance y la capacidad de integración como factores críticos a tener en cuenta en este tipo de sistemas. Por otro lado, también destaca que al desempeñar funciones para el sector público, las áreas de trabajo se ven afectadas frecuentemente por los cambios de normativa. Por este motivo, considera como un factor crítico que estas modificaciones a nivel de estructura organizacional puedan realizarse a través de módulos parametrizables, sin la necesidad de tener que realizar los cambios por scripts de base de datos.

Del circuito básico de control presentado, el entrevistado expresa su conformidad con respecto a las actividades incluidas y, destaca que automatizando las actividades de programación y de análisis final, seguramente se consiga una reducción considerable en el tiempo y el esfuerzo humano requerido para llevar a cabo una fiscalización.

Con respecto a la presentación del prototipo destaca el concepto dinámico del “Elemento”, que permite agregar o quitar propiedades especificas por cada “Proceso”, dado que no todos los controles registran el mismo tipo de información, de la forma en la que se desarrolla este concepto en el presente TF, se expone la flexibilidad suficiente para contemplar estos casos. Por otro lado, considera que el diseño utilizado para mostrar los elementos de trabajo, basado en “cards” y no en tablas, conlleva a un mejor despliegue en pantalla de la información.

En referencia al módulo de “Checklist”, resalta que el modulo expuesto en el prototipo, presenta un gran avance respecto al que utiliza en su trabajo, ya que incorpora los conceptos de “versiones” y “publicación”, que permiten editar un cuestionario sin que esto impacte directamente en las preguntas que se muestran en la implementación

productiva. Es decir, permite trabajar en una nueva “versión”, mientras se sigue utilizando la anterior, y a su vez permite retroceder rápidamente entre una “versión” y otra, en caso de que se detecte algún inconveniente. Esta flexibilidad, permite adecuar los cuestionarios rápidamente ante los cambios de normativas, que menciona como constantes en la diaria de su trabajo.

Por último, la interfaz gráfica presentada le resulta agradable y conductiva, concluye que la encuentra más “moderna” respecto al software que utiliza actualmente que está muy basado en “tablas”.

4.2. Análisis de entrevista a experto N° 2

La entrevista fue realizada el 18/11/2020 y tuvo una duración de 28 minutos. Al igual que en el caso anterior, se llevó a cabo manera remota a través de una videollamada, donde se compartió la pantalla para exponer el flujo de control propuesto junto con el segundo prototipo del software.

Durante la primera parte, el entrevistado menciona que en los últimos treinta años trabajo en el área de sistemas, específicamente a cargo de proyectos de desarrollo y gestión de la tecnología. En los últimos doce años trabajo específicamente en el sector público, en los primeros años desempeñaba sus funciones en la Agencia de Sistemas de Información de la Ciudad de Buenos Aires, y actualmente trabaja a cargo del área de sistemas de la Agencia Gubernamental de Control.

Debido a su puesto laboral actual, tiene contacto tanto con sistemas de inspecciones como con sistemas periféricos que interactúan entre sí. Hace especial hincapié que, en el organismo actual, se presenta una situación muy particular, referida al espectro de objetos que tienen que inspeccionar/controlar, que van desde locales comerciales, obra privada hasta la elaboración y expendio de alimentos.

Menciona la trazabilidad como factor importante para poder demostrar el correcto desempeño del inspector, en caso de que sea requerido. Por otro lado, menciona como aspectos variables las políticas públicas relativas al control, que se van ajustando de acuerdo a una realidad cambiante y a su vez afectan tanto al proceso de control como a los aspectos puntuales que se relevan.

Luego de presentar y explicar el circuito de control básico, expresa su conformidad respecto a la cantidad de actividades y afirma con seguridad que las actividades automatizadas de programación y análisis, son las apropiadas para reducir el esfuerzo humano y para mejorar la transparencia del proceso.

Por último, se presenta el segundo prototipo de la plataforma, y el entrevistado aprueba la interfaz gráfica visualizada, destacando que la tendencia en gobierno es que el ciudadano realice un autocontrol, por lo cual es muy importante que la herramienta que se utilice tenga una interfaz clara y amigable.

4.3. Comparación de entrevistas

En la tabla 4.3.1 se incluye una comparación resumida entre las entrevistas realizadas a los dos expertos entrevistados:

Dimensión	Experto Nº 1	Experto Nº 2
Versión	1.0.0 / “Gadget”	2.0.0 / “Hiperion”
Visión	Los cambios constantes de contexto, que se dan frecuentemente sobre todo en el ámbito público, requieren que la herramienta proporcione facilidades para su adaptación a las diferentes circunstancias.	La realización de inspecciones a través de un software aporta transparencia al proceso. Este aspecto es fundamental para garantizar que se respeten los derechos del inspeccionado y que se cumplan con las normativas vigentes, que por otro lado están sujetas a modificaciones frecuentes.

Dimensión	Experto Nº 1	Experto Nº 2
Recomendación	Hacer hincapié en la performance, la escalabilidad y la facilidad de integración con otros sistemas.	Hacer hincapié en la trazabilidad, flexibilidad y en la reducción del esfuerzo requerido para gestionar una inspección, a modo de optimizar el uso de los recursos, que son escasos en comparación al espectro de lugares que se deben fiscalizar.
Aprobación de interfaz gráfica	Si	Si

Tabla 4.3.1: Comparativa de entrevistas

5. RESULTADOS

5.1. SAICo en entorno de demo

En la figura 5.1.1 se observa la página de inicio del back-end, donde se incluye la documentación de los métodos disponibles, para lo cual se utilizó la herramienta Swagger.

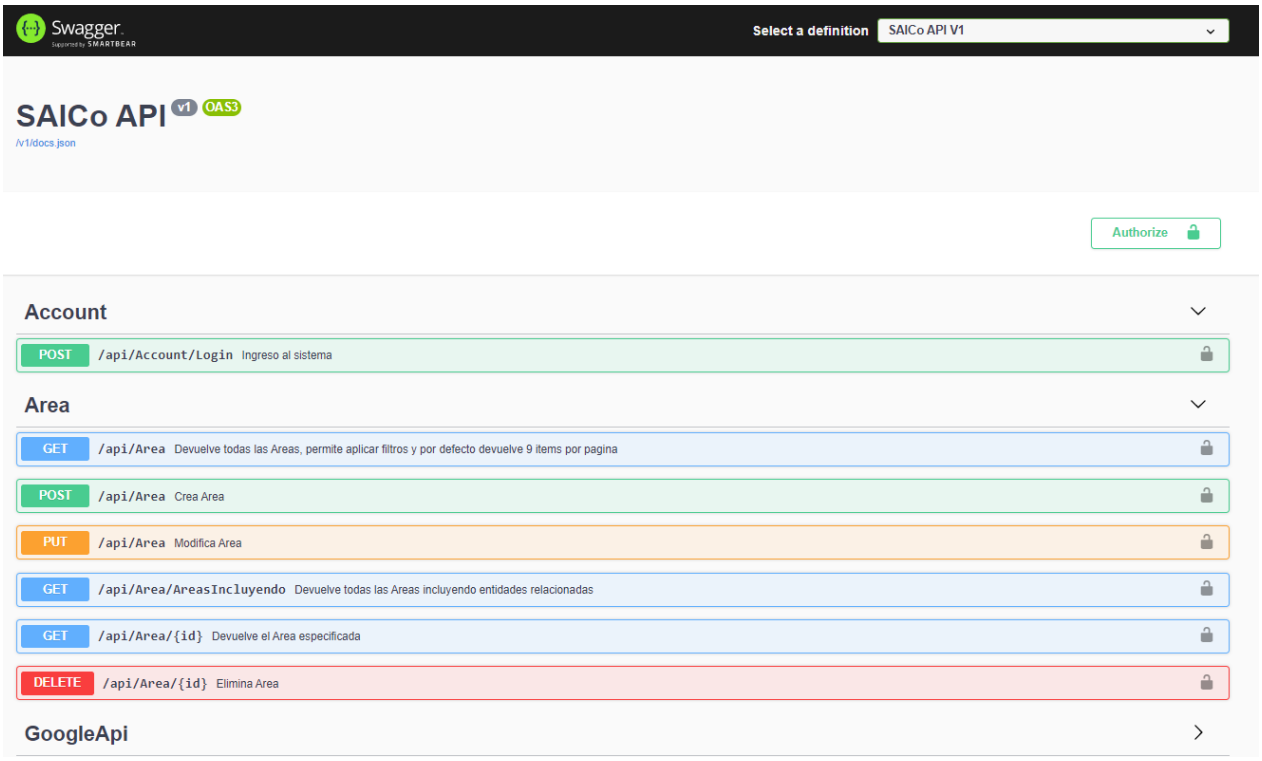

Figura 5.1.1: Back end - Pantalla de inicio

En la figura 5.1.2 se puede visualizar la pantalla principal del front-end, la bandeja de trabajo del usuario:

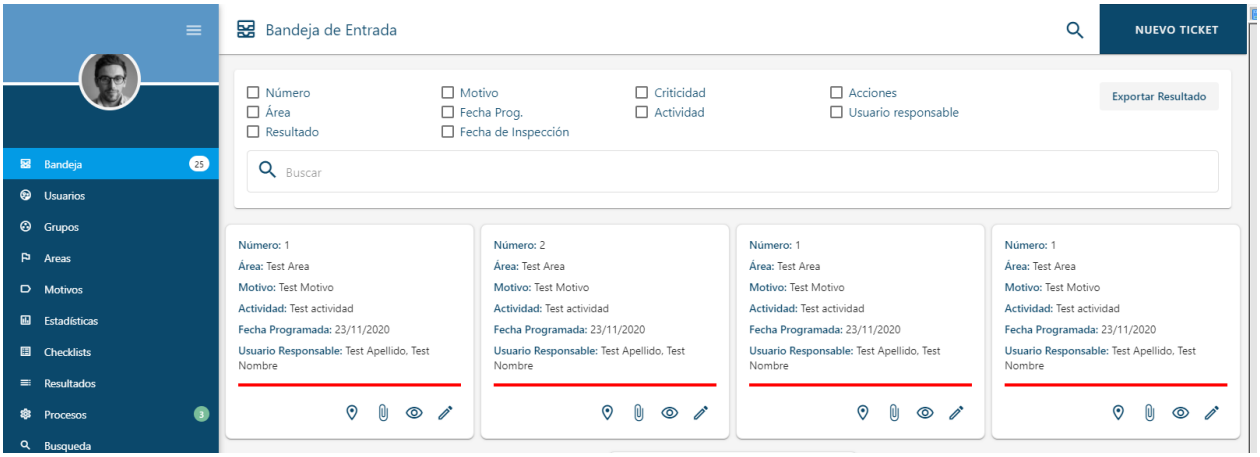

Figura 5.1.2: Front end – Bandeja de trabajo

En las figuras 5.1.3, 5.1.4 y 5.1.5 se observan las distintas secciones de parametrización de un checklist, destacado como uno de los módulos que deber ser

más flexible y amigable, debido a la frecuencia con la cual se realizan cambios sobre los cuestionarios que se deben completar en la inspección:

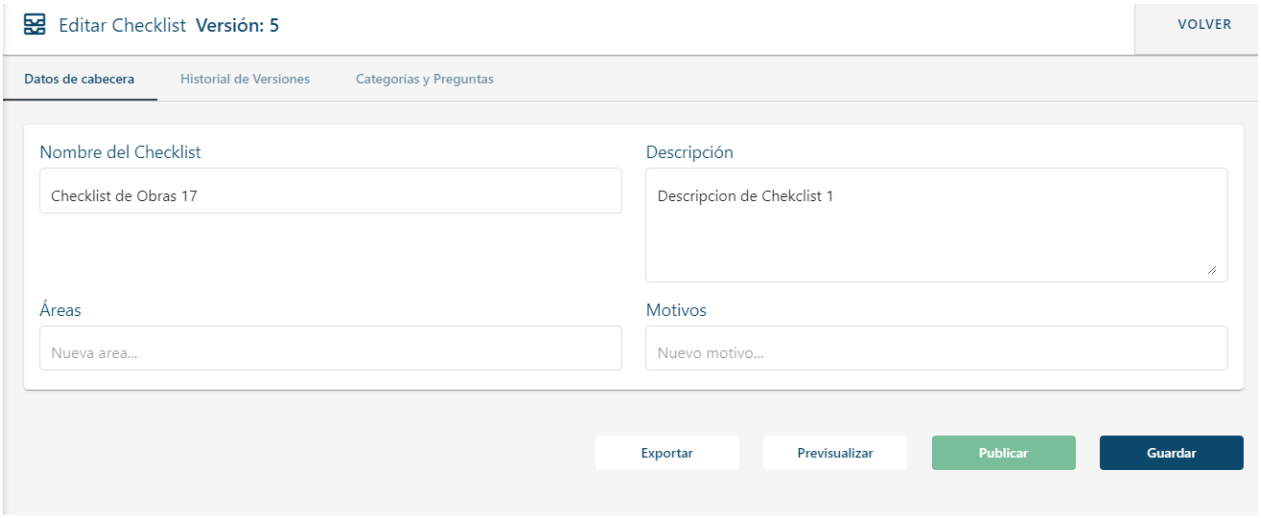

Figura 5.1.3: Front end – Cabecera de checklist

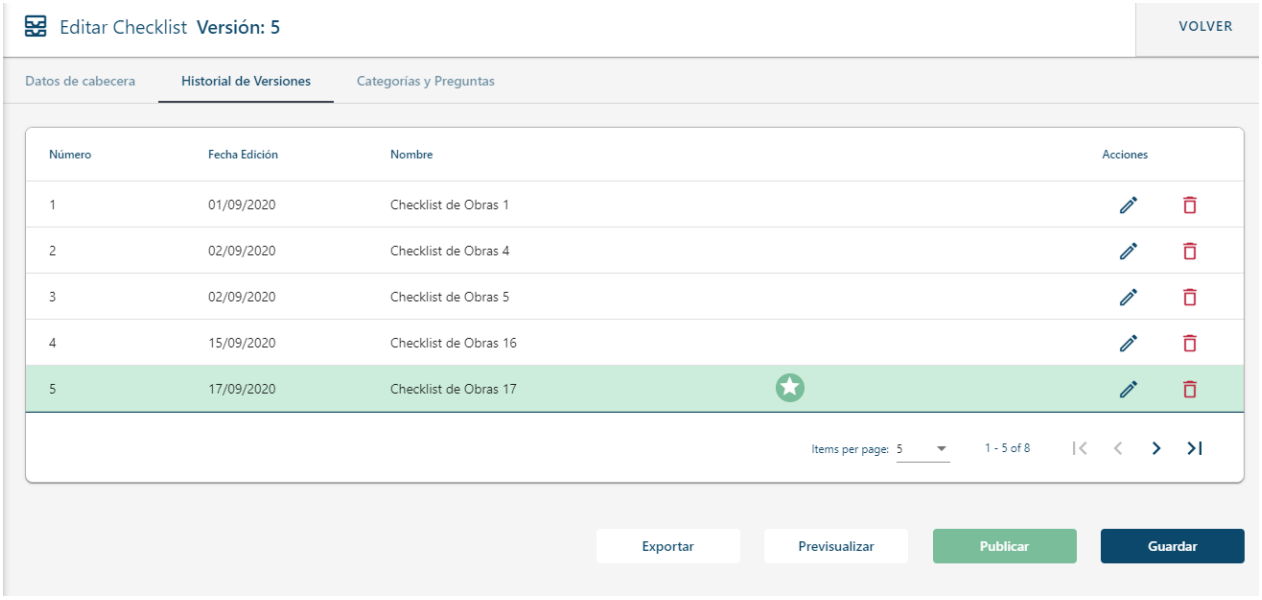

Figura 5.1.4: Front end – Versiones de checklist

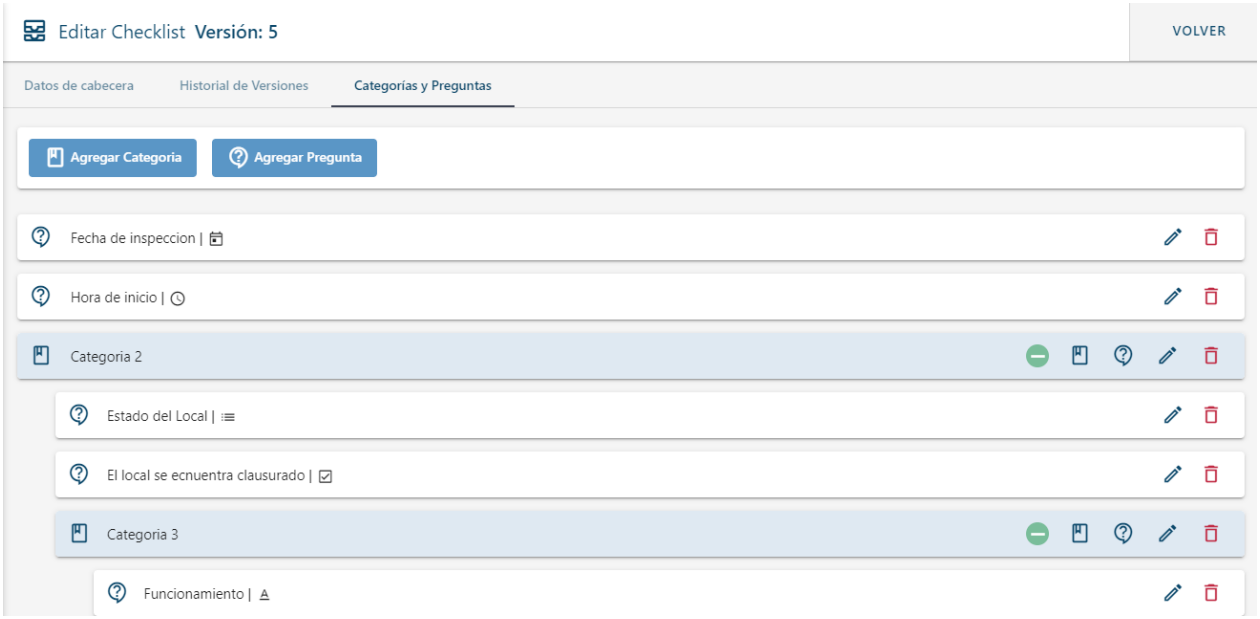

Figura 5.1.5: Front end – Categorias y preguntas de checklist

En la figura 5.1.6 y 5.1.7 se observan la sección de datos de un “Ticket” y el despliegue de un checklist desde la aplicación móvil, utilizada para llevar a cabo el trabajo en campo:

Figura 5.1.6: Sección de datos en app móvil

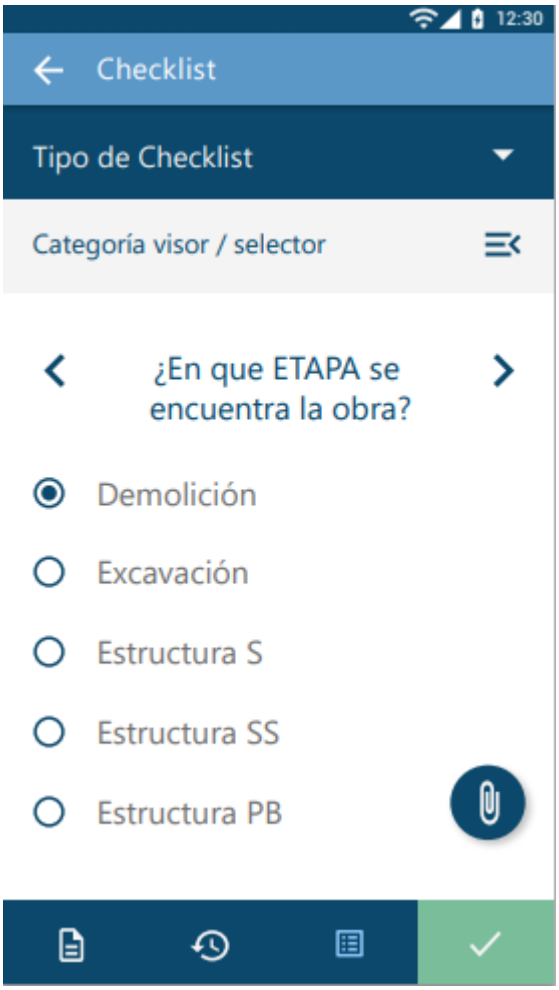

Figura 5.1.7: Pregunta en checklist en app móvil

5.2. Evaluación de las tecnologías seleccionadas

En la plataforma SAICo, se utilizaron diversas tecnologías, cada una de ellas fue seleccionada teniendo sus características particulares y, el objetivo específico del módulo sobre el cual se utilizaron.

En la sección 3.1, se describe el ciclo de vida y la metodología utilizada para el desarrollo, junto con cuestiones de seguridad transversales a la plataforma y, se define un conjunto de herramientas para llevar a cabo las practicas DevOps (GitHub, Azure Devops Pipelines, Docker y Amazon Aws). Además, en la sección 3.2 se describe la arquitectura del software, detallando los diversos módulos que forman parte del

sistema, junto con el detalle del servicio externo utilizado, correspondiente a los datos de geolocalización provistos por Google Maps.

Como complemento, en la sección 3.3 se describen las tecnologías seleccionadas, con las cuales se implementa la arquitectura mencionada en la sección anterior:

- Font-end: Angular
- Back-end: .Net Core y MySQL
- Modulo móvil: Xamarin
- File Manager: Node JS y MongoDB

5.3. Automatización de tareas

Para conseguir el objetivo de automatizar las actividades de “programación” y “análisis”, conforme a lo detallado en la sección 2, se utilizaron dos técnicas:

- Programación – Algoritmo de asignación automática: se elaboró un algoritmo que tiene en cuenta diversas propiedades, para poder determinar que inspector es el más apropiado para un determinado objetivo, asociándolo consecuentemente como responsable del mismo. De esta forma se evita la asignación manual por parte de un usuario operador, por lo cual no solo que se optimiza el esfuerzo involucrado para llevar a cabo la tarea, sino que también se vuelve más transparente el proceso. El detalle de dicho algoritmo se encuentra especificado en la sección 3.5.2.
- Análisis – Reglas automáticas: se desarrolló un módulo, detallado en la sección 3.5.3, que permite definir condiciones que hacen que se ejecute automáticamente una acción en consecuencia. Este procedimiento, permite al sistema emular la conducta de un operador, que revisa el trabajo de los inspectores para llevar a cabo una tarea según sea el resultado de la inspección.

5.4. Validación del software con entrevistas

Se llevaron a cabo dos entrevistas, con expertos en el uso de sistemas de inspecciones y control. Estas entrevistas fueron del tipo cualitativas-semiestructuradas, y se realizaron con el objetivo de validar los prototipos de la herramienta desarrollada, y obtener aspectos críticos y variables a tener en cuenta. Para el desarrollo de las mismas, se utilizó como guía el cuestionario detallado en la sección 3.2.1.

Como resultado de estos encuentros, se validó la selección de las actividades incluidas en el flujo de control básico, junto con el agregado de valor de las actividades automatizadas. Además, en ambos casos calificaron la interfaz gráfica de SAICo como “amigable” y “clara”, lo cual es un indicio de que los módulos incluidos se entienden con facilidad.

Por último, se obtuvieron diversos factores críticos y variables que los entrevistados destacaron como fundamentales en este tipo de sistemas:

- Performance
- Escalabilidad
- Apertura de interconexión con otros sistemas
- Módulos parametrizables para “Áreas” y “Checklist”

6. CONCLUSIONES

Teniendo en cuenta los objetivos planteados para este TF, definidos en la sección 1.3, se realiza a continuación una conclusión sobre el cumplimiento de los mismos, que se vincula con los conocimientos adquiridos durante la carrera, al mismo tiempo que se destaca el aporte de la herramienta obtenida y se dejan establecidos los lineamientos de base para futuras mejoras.

6.1. Discusión - Resumen final

Como resultado de este trabajo, se obtuvo una herramienta de software que permite llevar a cabo un control dinámico sobre diversos puntos de interés, tomando como atributos de base aquellos que se describen en la sección 3.4.

Con los diversos módulos y propiedades incluidos en el software, se busca contemplar los factores comunes al ámbito de las inspecciones y controles, sin profundizar en detalles muy puntuales que pueden variar según el contexto de uso. Se prevé que, en una problemática específica, puede plantearse la necesidad de incorporar aún más funcionalidades o información que puede provenir, por ejemplo, de un sistema externo. Por este motivo, se utilizó una arquitectura en capas explicada en la sección 3.2 que, separa la lógica en diversos módulos, lo cual facilita la modificación de frameworks y tecnologías, en caso de ser requerido. Las tecnologías y la arquitectura seleccionada, dotan al sistema de la robustez necesaria para permitir su uso bajo distintos tipos de volúmenes de información.

En adición a esto, se definieron y se incluyeron en la herramienta distintos mecanismos de automatización, explicados en la sección 3.5.1 y 3.5.2, que permiten reducir la necesidad de que un recurso humano intervenga en dos de las actividades planteadas como parte del flujo de trabajo básico de control, que se detalla en el módulo 2 de este TF.

Por consiguiente, como el software fue concebido con la idea de sentar una base, que luego pueda ser modificada para adaptarse a un contexto de negocio especifico, se obtuvo un producto con la flexibilidad y la escalabilidad requerida para facilitar esta adaptación. Además, se considera que se logró una interfaz gráfica clara y amigable, siendo esta validada en las entrevistas.

Esta herramienta gratuita, desarrollada con fines académicos, puede facilitar la realización de diversos controles, por parte de organismos públicos o privados que no

utilizan ningún software o que utilizan sistemas más limitados, que no ofrecen la parametrización y la automatización que se incluye en este producto.

6.2. Desafíos

Para la elaboración de este TF, se vincularon conocimientos técnicos con funcionales, es decir, conocimientos de las TIC con los del dominio del negocio, asociados a los procedimientos de inspección y control. Para poder obtener el producto final, el sistema SAICo, se tuvo que abordar el problema desde diferentes enfoques, utilizando diversas herramientas y metodologías. Todo este procedimiento guiado a través de un marco de investigación.

Por esta razón, se estableció un plan de proyecto para guiar el desarrollo y se llevaron a cabo actividades de distinta índole, como la investigación sobre tecnologías para la creación de software, la adquisición de conocimientos de desarrollo y de arquitectura de software y, la definición de diversos módulos y conceptos del dominio del negocio, con el objetivo de promover la automatización de tareas. Además, se llevaron a cabo entrevistas con expertos con el objetivo de validar la herramienta construida.

Todo el conocimiento adquirido, permitió concluir con la herramienta de software implementada en un servidor de demo, quedando el código fuente a disposición de la comunidad científica y académica, o del organismo que lo requiera para implementarlo y utilizarlo en su propio contexto.

6.3. Futuras líneas de investigación

Con el objeto de agregar valor y nuevas funcionalidades, a las ya existentes en el software SAICo, se plantean los temas que se observan en la Tabla 6.3.1.

Tema	Descripción
Algoritmos de IA para automatizar la generación de inspecciones	El objetivo es realizar un aprendizaje en base a las inspecciones realizadas con resultados negativos, para detectar patrones comunes entre estos casos y generar nuevos “Tickets” automáticamente, en los sucesos que se detecten como potencialmente peligrosos. De esta forma, se evita la dependencia de la carga manual de los “Tickets”, asegurando una base de objetivos fiscalizables relacionada a los casos que se identifican como más críticos, de acuerdo a la información obtenida en experiencias anteriores
Módulo para la gestión de descargos y generación de tienda de aplicaciones	Como consecuencia de una inspección o control, pueden detectarse situaciones que presentan irregularidades, que deben ser subsanadas por el presunto infractor. Para aquellos organismos que generen infracciones, se plantea la creación de un nuevo módulo para la plataforma, dedicado a la interacción con el público inspeccionado, de forma que puedan realizar descargos. De este modo, se evitaría tener que recurrir nuevamente al lugar, para verificar la corrección de una falta, sorteando el esfuerzo humano que esto conlleva. Este módulo, da lugar a lo que se conoce como “tienda de aplicaciones”, donde la implementación de los sistemas específicos dependerá del contexto del cliente donde se ejecute el software.
Arquitectura “multi-tenant”	Modificar la arquitectura actual, para que una sola instancia del software pueda dar servicio a muchos clientes. Lógicamente cada cliente utiliza su propia parametrización de datos, pero el código que se ejecuta es el mismo en todos los casos.

	De esta forma se optimizan los esfuerzos de implementación y mantenimiento, permitiendo que el software sea ejecutado en la nube, dando respuesta a múltiples organismos que lo utilizan a un bajo costo.

Tabla 6.3.1: Futuras líneas de investigación

7. BIBLIOGRAFÍA

ACOSTA, Victor Manuel, **2019**. Frontend con Angular: Todo lo que debes saber sobre esta herramienta. *Canal Informática y TICS* [en línea]. [Consulta: 12 septiembre 2020]. <https://revistadigital.inesem.es/informatica-y-tics/frontend-con-angular-todo-lo-que-debes-saber-sobre-esta-herramienta/>.

AGC, **2020**. Log in - LIZA. [en línea]. [Consulta: 10 octubre 2020]. <http://liza.agcontrol.gob.ar/Account/Login>.

AMAR HANSPAL, **2020**. In the age of automation, technology will be essential to reskilling the workforce. *World Economic Forum* [en línea]. [Consulta: 7 septiembre 2020]. <https://www.weforum.org/agenda/2020/03/how-tech-can-lead-reskilling-in-the-age-of-automation/>.

AMAZON AWS, **2020**. ¿Qué es AWS? *Amazon Web Services, Inc.* [en línea]. [Consulta: 11 octubre 2020]. <https://aws.amazon.com/es/what-is-aws/>.

APM, **2017**. ¿Qué es Xamarin? [en línea]. [Consulta: 6 septiembre 2020]. </que-es-xamarin-caracteristicas-desarrollo-apps-multiplataforma/>.

ASF, THE APACHE SOFTWARE FOUNDATION, **2019**. Licencia Apache, versión 2.0. [en línea]. [Consulta: 10 octubre 2020]. <https://www.apache.org/licenses/LICENSE-2.0>.

AUTH0.COM, **2020**. JWT.IO - JSON Web Tokens Introduction. [en línea]. [Consulta: 19 noviembre 2020]. <http://jwt.io/>.

BASCO, Ana Inés, BELIZ, Gustavo, COATZ, Diego y GARNERO, Paula, **2018**. *Industria 4.0: Fabricando el Futuro* [en línea]. S.l.: Inter-American Development Bank. [Consulta: 7 septiembre 2020]. <https://publications.iadb.org/handle/11319/9015>.

BUENOS AIRES CIUDAD, **2018**. Gestiones gubernamentales más eficientes, también en Mendoza. *Buenos Aires Ciudad - Gobierno de la Ciudad Autónoma de Buenos Aires* [en línea]. [Consulta: 11 septiembre 2020]. <https://www.buenosaires.gob.ar/gobierno/intercambioba/gestiones-gubernamentales-mas-eficientes-tambien-en-mendoza>.

CERTAINTY SOFTWARE, **2019**. Certainty Software - Manage Performance, Not Data. [en línea]. [Consulta: 11 septiembre 2020]. <https://www.certaintysoftware.com/>.

CLARIN.COM, **2018**. La Ciudad «exporta» su sistema de fiscalización a otros organismos y municipios. [en línea]. [Consulta: 11 septiembre 2020]. <https://www.clarin.com/ciudades/ciudad-exporta-sistema-fiscalizacion-organismos-municipios_0_BJ7ZcMcUQ.html>.

DOCKER, **2020**. ¿Qué es un contenedor? | Containerización de aplicaciones | Estibador. [en línea]. [Consulta: 11 octubre 2020]. <https://www.docker.com/resources/what-container>.

GITHUB INC, **2020**. Build software better, together. *GitHub* [en línea]. [Consulta: 11 octubre 2020]. <https://github.com>.

GOOGLE, **2020**. Places | Google Maps Platform. *Google Cloud* [en línea]. [Consulta: 23 septiembre 2020]. <https://cloud.google.com/maps-platform/places?hl=es>.

GRUPO GINEBRA, **2020**. Documenta en Clean Garden. *Grupo Ginebra* [en línea]. [Consulta: 11 septiembre 2020]. <https://grupoginebra.com/casos-de-exito/documenta/documenta-en-clean-garden/>.

HERNÁNDEZ SAMPIERI, Roberto, FERNÁNDEZ COLLADO, Carlos, BAPTISTA LUCIO, Pilar, MÉNDEZ VALENCIA, Sergio y MENDOZA TORRES, Christian Paulina, **2014**. *Metodología de la investigación*. México, D.F.: McGraw-Hill Education. ISBN 978-1-4562-2396-0.

HÜTTERMANN, Michael, **2012**. *DevOps for Developers*. New York: s.n. 214 p. ISBN 978-1-4302-4569-8.

ISO/IEC/IEEE, **2017**. ISO/IEC/IEEE 12207:2017, First Edition: Systems and software engineering - Software life cycle processes. .

LANUSE, Matias, **2018**. Decreto Provincia de Buenos Aires. [en línea]. [Consulta: 21 septiembre 2020]. <https://normas.gba.gob.ar/documentos/BdaRDaHO.pdf>.

MAINTAINX, **2020**. MaintainX – Manage your Maintenance and Operations. *MaintainX* [en línea]. [Consulta: 11 septiembre 2020]. <https://www.getmaintainx.com/>.

MARTIN, Robert, **2017**. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. London, England: s.n. 432 p. ISBN 978-0-13-449416-6.

MICROSOFT, **2018**. Xamarin App Development with Visual Studio. *Visual Studio* [en línea]. [Consulta: 7 septiembre 2020]. <https://visualstudio.microsoft.com/es/xamarin/>.

MICROSOFT, **2020**. Información general sobre .NET Core. [en línea]. [Consulta: 12 septiembre 2020]. <https://docs.microsoft.com/es-es/dotnet/core/about>.

MICROSOFT AZURE, **2020a**. Azure Pipelines | Microsoft Azure. [en línea]. [Consulta: 11 octubre 2020]. <https://azure.microsoft.com/es-es/services/devops/pipelines/>.

MICROSOFT AZURE, **2020b**. ¿Qué es DevOps? Explicación de DevOps | Microsoft Azure. [en línea]. [Consulta: 10 octubre 2020]. <https://azure.microsoft.com/es-es/overview/what-is-devops/>.

MORENO, Juan José, **2016**. La automatización de los procesos y el trabajo humano. *IEEM Revista de Negocios*, vol. 19, no. 3, pp. 38-38. ISSN 23011181.

MYSQL, **2020**. MySQL :: Zona de desarrolladores. *https://dev.mysql.com/* [en línea]. [Consulta: 12 septiembre 2020]. <https://dev.mysql.com/>.

OJAMAA, Andres y DÜÜNA, Karl, **2012**. Assessing the security of Node.js platform. *2012 International Conference for Internet Technology and Secured Transactions*. S.l.: s.n., pp. 348-355.

ORACLE ARGENTINA, **2020**. Field Service Cloud | Customer Experience | Oracle Argentina. [en línea]. [Consulta: 11 septiembre 2020]. <https://www.oracle.com/ar/cx/service/field-service-management/>.

PRESSMAN, ROGER S., **2013**. *Ingeniería del software: un enfoque práctico*. S.l.: s.n. ISBN 978-1-4562-1836-2.

PRONTOFORMS, **2020**. ProntoForms: Mobile Forms & Field Data Collection Apps. *ProntoForms* [en línea]. [Consulta: 11 septiembre 2020]. <https://es.prontoforms.com/>.

ROBLEDANO, Angel, **2019**. Qué es MySQL: Características y ventajas. *OpenWebinars.net* [en línea]. [Consulta: 12 septiembre 2020]. <https://openwebinars.net/blog/que-es-mysql/>.

SATHEESH, Mithun, D'MELLO, Bruno Joseph y KROL, Jason, **2015**. *Web Development with MongoDB and NodeJS - Second Edition: Build an interactive and full-featured web application from scratch using Node.js and MongoDB*. Birmingham, UK: s.n. 300 p. ISBN 978-1-78528-752-7.

SCHWABER, Ken, **2012**. *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust*. Hoboken, N.J: s.n. 216 p. ISBN 978-1-118-20666-9.

SOMMERVILLE, Ian, **2012**. *Ingeniería de software*. S.l.: s.n. 792 p. ISBN 978-607-32-0603-7.

STALLMAN, Richard M. y LESSIG, Lawrence, **2007**. *Software libre para una sociedad libre*. Madrid: Traficantes de Sueños. ISBN 978-84-933555-1-7.

VIEITES, Alvaro Gomez, **2010**. *Seguridad Informática. Básico*. Madrid: s.n. 122 p. ISBN 978-84-92650-36-1.

WOHLGETHAN, Eric, **2018**. Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js. , pp. 84.

8. ANEXOS

8.1. Anexo 1: Glosario

Término	Definición
AGC	Agencia Gubernamental de Control de la Ciudad de Buenos Aires
ASI	Agencia de Sistemas de Información de la Ciudad de Buenos Aires
API	Application Programming Interface
CD	Despliegue Continuo (en inglés: Continuous Delivery)
CI	Integración Continua (en inglés: Continuous Integration)
CIA	Confidentialy, Integrity, Availability
DevOps	Combinación de las palabras en inglés “Development” y “Operation”
DIP	Dependency Inversion Principle
IA	Inteligencia Artificial
IEC	Comisión Electrotécnica Internacional (en inglés: International Electrotechnical Commission)
IEEE	Instituto de Ingeniería Eléctrica y Electrónica (en inglés: Institute of Electrical and Electronics Engineers)
ISO	Organización Internacional de Normalización (en inglés: International Organization for Standardization)
ISP	Interface Segregation Principle
JSON	JavaScript Object Notation
JWT	Json Web Tokens
LSP	Liskov Substitution Principle
MIT	Instituto de Tecnología de Massachusetts (en inglés: Massachusetts Institute of Technology)
MVC	Modelo-Vista-Controlador
OCP	Open/Closed Principle
OFSC	Oracle Field Service Cloud
RDBMS	Sistema de gestión de bases de datos relacionales (en inglés:

	Relational Database Management System)
SAICo	Software Automatizado para Inspecciones y Control
SDK	Kit de desarrollo de software (en inglés: Software Development Kit)
SRP	Single Responsibility Principle
TF	Trabajo Final
TIC	Tecnologías de la Información y la Comunicación
UADE	Universidad Argentina de la Empresa

8.2. Anexo 2: Entrevistas con expertos

8.2.1. Guía de entrevistas

Fecha: __/__/____.

Nombre y apellido del entrevistado:

Educación/Formación:

Experiencia laboral:

1. ¿Cuál es o cual fue su relación con un sistema de inspecciones y/o control?
2. ¿Cuáles son los principales problemas que observa del sistema con el cual tiene o tuvo relación?
3. ¿Cuáles son los aspectos que considera como “críticos” en este tipo de sistemas?
4. ¿Cuáles son los conceptos que considera como más variables dentro de este contexto?
5. Con respecto al circuito básico de control planteado para este trabajo. ¿Está de acuerdo con las actividades incluidas o considera que se deben modificar?
6. Las actividades que se plantean para automatizar, ¿Considera que reducen el esfuerzo humano requerido para llevar a cabo una inspección o control?
7. Del prototipo presentado, ¿Qué aspecto considera que se debe mejorar? ¿Podría sugerir alguna forma?
8. ¿La interfaz gráfica le resulta clara y amigable u observa dificultad en su comprensión?

8.2.2. Entrevista a experto N° 1

Fecha: 26/10/2020

Nombre y apellido: Jorge Iván Pavloff Miranda

Educación/Formación: Técnico en Análisis de Sistemas

Experiencia laboral: Agencia Gubernamental de Control (10 años)

1. ¿Cuál es o cual fue su relación con un sistema de inspecciones y/o control?

Hace diez años empecé a trabajar en el ámbito de inspecciones en el Gobierno de la Ciudad, con un perfil técnico y operativo, teniendo en cuenta las misiones y funciones de la AGC para que los inspectores fiscalicen e inspeccionen todas las actividades comerciales de la Ciudad.

2. Para la realización de estas inspecciones, ¿Utiliza algún software?

Sí, claro. En todo este tiempo hemos utilizado dos, en un principio el denominado “Fisca” que era más que nada un sistema registral, me acuerdo que trabajábamos con 15 data entries que cargaban todo lo que informaba el inspector mediante un informe en papel, después se guardaba y archivaba para consulta. Posteriormente buscando una evolución, pasamos al denominado “Liza”, donde el objetivo era trabajar en estaciones, o sea, en “Tickets” que van pasando de un lugar a otro hasta que se cierra la inspección.

3. ¿Cuáles son los principales problemas que observa del sistema actual o de los sistemas con los cuales tuvo experiencia?

Del sistema original, es decir, de “Fisca”, que era solo registral, no se podía interactuar. En el último sistema, ya hace unos años, más que nada el problema de performance, a medida que se fueron generando nuevas áreas, nuevas direcciones generales o se fueron integrando con diversas áreas de la Ciudad que se fueron uniendo a la AGC lo cual escaló. Pasamos de tener 100 usuarios a tener 2700 lo cual generó un conflicto de

escalabilidad y de performance, se puso todo muy lento, presentando dificultades para trabajar en la diaria.

4. ¿Cuáles son los aspectos que considera críticos en los sistemas de inspecciones?

Como te decía, hoy en día es muy importante la performance, la escalabilidad, la integración a través de estos años nos hemos dado cuenta que una entidad gubernamental o X organización puede o amerita que se conecte con otra organización, y esto siempre ha sido un caos o hemos estado limitados para esa integración, y también para la automatización de las acciones a llevar a cabo en cada inspección.

5. ¿Cuáles son los aspectos que considera más variables dentro de este contexto?

Bueno, al trabajar para el Estado, aunque entiendo que para las empresas privadas en otro aspecto puede ser también, es muy dinámico el cambio de normativa a la cual está atada el accionar inspectivo y también el procedimiento administrativo que lleva a cabo esa inspección. En nuestro caso hemos tenido muchos cambios de áreas, muchos cambios de direcciones generales, se han articulado las uniones de diversas entidades en una sola y ahí siempre hemos caído en tener que pedirle al proveedor que genere scripts para estar acorde a los cambios normativos o estructurales.

6. Es decir, ¿Considera critico que el módulo de "Áreas" sea parametrizable por un usuario final para que pueda ir adecuándose a estos cambios de normativa?

Si, tal cual. Sumamente crítico teniendo cuenta el contexto y la coyuntura diaria que vivimos en el país y de los cambios estructurales que acontecen.

Se presenta el circuito básico de control, que se detalla en la sección 2.

7. De acuerdo a su experiencia, ¿Considera que las actividades incluidas contemplan un circuito de control básico?

Si, contemplo que las actividades son las necesarias para poder desarrollar la actividad de una manera aconsejable. Me parece perfecto que se busque una automatización, tanto en la programación como en la parte final de la inspección, que de alguna manera retroalimente la carga o esa programación, me parece adecuado.

8. ¿Consideras que automatizando estas dos actividades contribuiría a reducir el esfuerzo humano requerido para llevar a cabo una inspección?

Si, seguramente, optimizarías recursos y obviamente el tiempo, más que nada si estas teniendo en cuenta la reprogramación también.

Se presenta el primer prototipo del software, versión "Gadget", y además se explican los criterios de automatización.

9. Con respecto a la interfaz gráfica presentada, ¿Resulta clara/amigable u observa alguna dificultad para la comprensión?

Dejando de lado, lo que te comentaba de mi costumbre a la interfaz gráfica de mi sistema que esta un paso más atrás, me parece agradable, intuitiva. En cierta forma te va conduciendo y te va relacionando con todas las funciones que utilizas del sistema. Haciendo esa salvedad que estoy muy acostumbrado a trabajar con planillas, tablas, me parece mejor una vez que entro en la afinidad de verlo. Me parece una vuelta de rosca a lo que estoy acostumbrado. Vi que tiene colores para la criticidad, y en todo caso se podría reacomodar a cada organización el tema de los filtros, acomodar un poco la información más como la quiere el usuario. Hablando de una bandeja de trabajo, me parece que está bien, es amigable, conductivo, te proporciona la información de una manera clara, y si queres entrar tenes los botones de acceso. Hablando de los botones, reemplazaría el del "ojito" por un lápiz, como para tener algo más unificado en ese sentido, pero es algo menor.

8.3.3. Entrevista a experto N° 2

Fecha: 18/11/2020

Nombre y apellido: Carlos Raúl Rodríguez

Experiencia laboral: 30 años en el sector de sistemas. Últimos 12 años en el sector público (7 años en ASI y 5 años como director de sistemas de la AGC)

1. ¿Cuál es su relación con un sistema de inspecciones y/o control?

Bueno, específicamente la misión que tengo en la AGC es la administración de los sistemas que utiliza la agencia, y en esto están incluidos los sistemas de fiscalización pero también otra serie de aplicaciones que tienen relación con las misiones y funciones de la agencia y que se relacionan entre sí. La agencia tiene un espectro de responsabilidades que abarcan todas las actividades comerciales de la Ciudad de Buenos Aires, también de obra civil privada y de elaboración y expendio de alimentos. El alcance de la agencia en cuanto a cuestiones de fiscalización y control es muy amplio, es una situación prácticamente única a nivel nacional

2. Con respecto a estos sistemas, ¿cuáles son los principales problemas que observa?

Como toda actividad de control está supeditada a la gestión de personas y a la interrelación de esas personas con los ciudadanos por un lado, y con los funcionarios y las políticas de gobierno por otro. Los sistemas de fiscalización se han creado, y se siguen mejorando, en lo posible, para que esa gestión sea lo más transparente posible y que sea lo más justo desde el punto de vista del cumplimiento de las normativas y también de los derechos de los ciudadanos. Esta doble situación de controlar pero a la vez respetar los derechos de los ciudadanos, hace que las aplicaciones pongan énfasis en la trazabilidad de los actos y en la transparencia de los mismos.

3. ¿Cuáles son los aspectos que considera críticos en este tipo de sistemas de inspecciones?

Bueno, uno de los casos más importantes es que el universo de locales o de objetos a inspeccionar es realmente muy amplio, muy grande, y hace falta realizar una selección de aquello que se va a fiscalizar de forma permanente. Estamos dirigiendo a donde hay que ir a inspeccionar y a controlar a los efectos que la población esté cubierta sobre los riesgos que puede generar la acción de los comerciantes, de las empresas constructoras o de las empresas que elaboran alimentos. Por lo tanto, hay que hacer un análisis de donde y con qué riesgos deben ser considerados los entes a fiscalizar, y por otro lado hay que asegurarse que esa fiscalización sea lo mejor posible. Para eso es indispensable que sea trazable y que sea transparente.

5. ¿Cuáles son los aspectos que considera más variables?

El tema es que las políticas públicas respecto el control varían en forma permanente, eso conlleva a que deba modificarse no solo el proceso sino también aspectos que se van incorporando al control. Esto implica que los checklist utilizan para sistematizar la fiscalización deban ser modificados permanentemente. Por lo tanto, las aplicaciones que se utilicen para esto deberán tener esa flexibilidad, de editar y reditar esos checklist y mantener la consistencia de la información que se releva. Ese es un tema importantísimo, la flexibilidad de la aplicación, las políticas cambian en forma permanente y los procesos cambian de igual forma porque es la realidad la que cambia y la que obliga a hacer un seguimiento de esto.

Se presenta el circuito básico de control y se explican los criterios de automatización que se detallan en la sección 2.

6. Con respecto al flujo de trabajo y de acuerdo a su experiencia, ¿Considera que las actividades incluidas abarcan un circuito de control básico de control o inspección?

Sí, creo que está cubierto todo. En grandes rasgos está absolutamente cubierta la actividad de fiscalización tal como la llevamos adelante en la agencia de control. Prácticamente están todos los pasos, no creo que este faltando nada.

7. Con respecto a las actividades que se plantean para automatizar, ¿Considera que reducirían el esfuerzo requerido para llevar a cabo una inspección?

Totalmente, creo que son justamente los dos puntos donde es posible mejorar el procesamiento y en lo posible automatizarlo, lo mejor que se pueda. Tanto en la programación como en el análisis, en la programación mediante la asignación automática de los objetos fiscalizables al responsable, eso es clave para reducir el esfuerzo humano y para mejorar la transparencia. En el análisis hay mucho para hacer, es un área donde podemos obtener mucha más información y eso generaría mejores decisiones respecto a la gestión. Me parece que son dos puntos donde vale la pena hacer el esfuerzo extra de automatización.

Se presenta el segundo prototipo del software, versión "Hiperion".

8. Con respecto a la interfaz gráfica presentada, ¿Resulta clara/amigable u observa alguna dificultad para la comprensión?

No, al contrario, me parece que está clara, que está despojada. Me parece que no tiene elementos que puedan confundir. Hay que cuidar mucho este aspecto porque la tendencia en gobierno es que muchas de estas actividades las haga directamente el ciudadano, es decir, que responda el checklist o el cuestionario como lo haría el inspector. Esto fuerza a que haya una especial preocupación por la experiencia y las interfaces, ya que no se va a tratar solo de capacitar a un grupo de agentes para que maneje la aplicación, sino que esa aplicación va a tener que llegar en alguna versión simplificada al ciudadano, en caso de corresponder. Ese es un aspecto que se debe tener muy en cuenta, esto que estoy diciendo es una tendencia que ya la estamos viendo a partir de toda esta situación especial que genero la pandemia, y es muy probable que vaya a crecer. En la agencia ya hay muchas aplicaciones que se están planteando que sean de autogestión, y si bien sigue estando solo en manos de los inspectores o de personal profesionalmente capacitado para hacer una inspección, muy probablemente algunas de esas actividades pasen a manos de los ciudadanos mismos, que en forma de declaración jurada le presenten los datos a la repartición correspondiente.