Universidad Nacional de Río Negro

Sede Atlántica

Licenciatura en Sistemas

Trabajo Final de Carrera

Aplicación para la administración de riesgos y controles de seguridad asociados a activos informáticos

Tesista: T.U.P. PEÑA; Ricardo Luis

Directores: Mg. LUGANI; Carlos Fabián

Lic. MUÑOZ ABBATE; Horacio Andrés

{rlpena; clugani; hmunoz}@unrn.edu.ar

Viedma, Río Negro. Año 2020

Agradecimientos

Es mi deseo hacer llegar mis más sinceros agradecimientos a quienes, de una manera u otra, han aportado su ayuda para hacer posible la realización de este trabajo:

Al director de la tesina y de la carrera de Lic. en Sistemas, Mg. Carlos Lugani, por la constante supervisión del trabajo, sus consejos y por permitirme conocer el apasionante campo de la seguridad informática, pero sobre todo por su motivación para siempre aspirar a más.

Al codirector del trabajo, Lic. Horacio Muñoz Abbate, por aportar sus conocimientos en desarrollo de software para orientarme a crear un producto de alta calidad, así como también su buena predisposición a prestar ayuda en cualquier momento.

A Mg. Luis Vivas y el Laboratorio de Informática Aplicada, que me abrieron sus puertas para poder llevar a cabo esta labor allí, además de brindar continuas capacitaciones que me permitieron crecer como desarrollador.

A la Dra. Paola Britos, que desinteresadamente prestó su colaboración para estructurar este trabajo y por motivarme a exigirme constantemente.

A la Prof. María Silvia Alasio, por aportar sus conocimientos en la escritura académica que me permitieron redactar este documento.

A mi familia, que ha sido el pilar desde el comienzo, acompañando mi trayectoria estudiantil y apoyándome en todo momento, sobre todo en aquellos más difíciles.

Resumen

Es innegable que la información constituye un recurso esencial para toda organización, puesto que de ella depende su desempeño. En la actualidad, se incluye cada vez más el uso de herramientas tecnológicas destinadas a tratar la información que circula en la organización. De ello es que estas herramientas conocidas como activos informáticos son de suma importancia y por lo tanto es menester protegerlos contra las amenazas de seguridad, gestionando estos riesgos de forma proactiva. No obstante, la mayoría de las organizaciones aún no han comprendido esta importancia y en consecuencia, no invierten lo suficiente en materia de seguridad de la información. Este inconveniente se debe principalmente al costo elevado de implementar el proceso, y a la falta de madurez dentro de la comunidad organizacional con respecto a la gestión de riesgos y su importancia. Por ello se desarrolló un sistema web destinado a introducir la gestión de riesgos informáticos en aquellas organizaciones que aún no la han implementado, y que integra la administración de activos informáticos, los riesgos a los que estos se exponen, así como también los controles de seguridad desarrollados para reducir o mitigar tales amenazas. La capacidad de asociar los controles con normas y políticas permite a este sistema orientarse hacia el cumplimiento de la Gobernanza, Riesgo y Cumplimiento, principio que permitirá a la organización generar valor.

Palabras clave: gestión de riesgos, activos informáticos, software, Gobernanza, Riesgo y Cumplimiento.

Índice

1. Introducción

Este documento corresponde al proyecto de Tesina de Grado, requisito necesario para la obtención del título de Licenciado en Sistemas, expedido por la Universidad Nacional de Río Negro.

El objetivo de la misma es demostrar la gestión e implementación de un sistema software para la gestión de riesgos y la aplicación de controles de seguridad asociados a los activos informáticos. Para ello, el aplicativo mantiene un inventario de los activos informáticos presentes en la organización y los riesgos a los que estos están expuestos.

Esta tesina se estructura mediante los siguientes apartados:

- Estado de la cuestión. En él se explican los fundamentos teóricos con respecto a los activos informáticos, gestión de riesgos y Gobernanza, Riesgo y Cumplimiento. Además, expone el estado actual del desarrollo de software relacionado con esta cuestión.
- Problema a resolver. Describe la problemática actual de la gestión de riesgos en las organizaciones, el modo en que esta se intentará resolver y los objetivos de dicha solución.
- Solución propuesta. En este capítulo se detalla el desarrollo de la solución de software propuesta, incluyendo requerimientos, herramientas y metodología.
- Verificación y validación. Menciona las pruebas realizadas para asegurar la calidad del sistema desarrollado.
- Conclusiones y líneas futuras. Enuncia las ideas principales del presente trabajo así como las futuras investigaciones a realizar para mejorar la solución desarrollada.
- Referencias. La bibliografía consultada para la redacción del presente documento.

2. Estado de la cuestión

Para una mejor comprensión de este trabajo, es necesario precisar el entorno bajo el cual fue desarrollado. En este capítulo se define el concepto de activo informático y su clasificación, el proceso de gestión de riesgos y su aplicación a tales activos. A su vez, este proceso conforma uno de los engranajes del modelo Gobernanza, Riesgo y Cumplimiento, concepto que también se define aquí. Por último, se menciona el estado actual del desarrollo de software relacionado con esta temática.

2.1. Marco teórico

2.1.1. Definición y clasificación de activos informáticos

La aplicación de la Gestión de Riesgos exige conocer primero cuáles son los activos que se han de analizar. En el desarrollo del presente trabajo se utilizó el concepto de activo informático de acuerdo con Lugani y Peña (2018, p.171):

> *Los activos informáticos –o activos TI- son aquellos recursos (hardware/software) que tiene o explota una organización para el desarrollo de sus actividades de negocio, y que hacen uso de la información concerniente a dicha organización, a través de la tecnología. La información, al estar contenida en estos recursos, es por tanto otro activo informático, de igual (o incluso mayor) importancia que los otros recursos. Además de estos recursos tecnológicos están los humanos relacionados con los activos TI de la organización.*

En la misma línea que estos autores, los activos informáticos se clasifican en: hardware (servidores, PCs, smartphones, impresoras, monitores, etc), software de aplicación (aplicaciones del negocio, sistemas de gestión empresarial, sistemas operativos, sistemas de oficina, etc.), comunicación (redes de comunicación, enlaces de fibra óptica, routers, etc.), datos (bases de datos, información del personal, claves, manuales de usuario, material de capacitación, documentación de sistemas), proveedores, terceros, proyectos de TI e información organizacional no contenida en software. En esta clasificación se incluyen también los dispositivos relacionados con el Internet de las Cosas (Internet of Things o IoT).

2.1.2. Gestión de riesgos

La información es esencial para una organización y por eso debe estar siempre protegida. Toda organización requiere y utiliza constantemente información para conocer su desempeño, optimizar recursos y tomar decisiones para cumplir con sus objetivos. Hoy en día, se hace uso de herramientas tecnológicas para el tratamiento de la información en los procesos de negocio. Estos recursos o activos informáticos están continuamente expuestos a amenazas que comprometen la seguridad de la información[1] que administran. Por esto es que se necesita un proceso para reducir o mitigar los riesgos de que se produzcan amenazas de seguridad identificándolos, clasificándolos, cuantificándolos e implementando controles en consecuencia. Así, la protección de los activos informáticos exige la identificación, análisis y tratamiento de los riesgos para garantizar la seguridad de la información.

La Gestión de Riesgos involucra a toda la organización, en consecuencia la misma en su totalidad se verá beneficiada con este proceso. El estándar ISO/IEC 31000 recomienda el desarrollo, implementación y mejora de un marco de trabajo para integrar la Gestión del Riesgo en los procesos de negocio, las políticas y la cultura de toda la organización (International Organization for Standardization, 2009). Por esto, la protección de los activos informáticos y la información requiere de la cooperación de todo el personal de la organización, incluyendo el compromiso y apoyo de la dirección con el proyecto en todo momento. Al decir de Solarte, Enriquez y Benavides (2015, p.496):

> *En general, se puede afirmar que el tema de seguridad de la Información no es sólo un tema eminentemente técnico, sino que también involucra procesos del negocio y actividades de gobierno corporativo, que aseguren una continua gestión de los riesgos y aseguramiento de los niveles de seguridad requeridos por la organización.*

Los autores afirman que para cumplir con la ISO/IEC 27001 se requieren factores no técnicos como el apoyo de la dirección, objetivos de seguridad y negocio alineados, controles compatibles con la cultura organizacional, conocimiento de requerimientos de seguridad y administración de riesgos, canales adecuados de comunicación con los empleados acerca de

---

1 De acuerdo con la Oficina Nacional de Tecnologías de Información (2015, p.46), "La seguridad de la información es la protección de la información de un rango amplio de amenazas para poder asegurar la continuidad del negocio, minimizar el riesgo de la operación y la operación normal del organismo."

la seguridad de la información, disposición y revisión de políticas de seguridad de la información. No obstante, se puede afirmar que estos factores son determinantes no sólo para cumplir con la ISO/IEC 27001, sino con cualquier norma, y con los objetivos de seguridad requeridos. Los beneficios, por lo tanto, se verán reflejados en toda la organización. De acuerdo con Alemán y Rodríguez (2014), el análisis de riesgos informáticos es una parte fundamental en la administración de la seguridad y otorga beneficios como la identificación de debilidades en la estructura de TI relacionada con los procesos críticos, la determinación de dónde aplicar esquemas de recuperación de desastres y continuidad de negocio, y la definición de políticas de seguridad mejor adaptadas a la organización. En conclusión, es a través de la gestión de riesgos como la organización podrá conocer las amenazas y las oportunidades para alcanzar sus objetivos de negocio y de seguridad.

2.1.2.1. Ciclo de vida de la gestión de riesgos

A la hora de gestionar los riesgos, se siguen cinco etapas definidas. El proceso inicia con la planificación de la actividad, donde se definen los plazos a cumplir, el alcance de la actividad y los objetivos a cumplir. A continuación se realiza la etapa de relevamiento de los activos informáticos, sus vulnerabilidades, y las amenazas de seguridad. En base a este relevamiento, se analizan cualitativa y cuantitativamente los riesgos en términos de probabilidad de ocurrencia de las amenazas y su impacto. Este impacto puede ser medido sobre el tiempo de retraso en el funcionamiento del negocio, el desempeño del activo afectado, y el costo económico para la organización. Tal análisis conduce a la posterior priorización de estos riesgos. Luego de estas primeras etapas (consideradas de análisis), se prosigue a la planificación e implementación de acciones (controles) para mitigar o reducir los riesgos. Por último, se realiza un monitoreo y control sobre los riesgos identificados y las acciones propuestas en la etapa anterior. En la Figura 2.1 se grafica el proceso de gestión de riesgos:

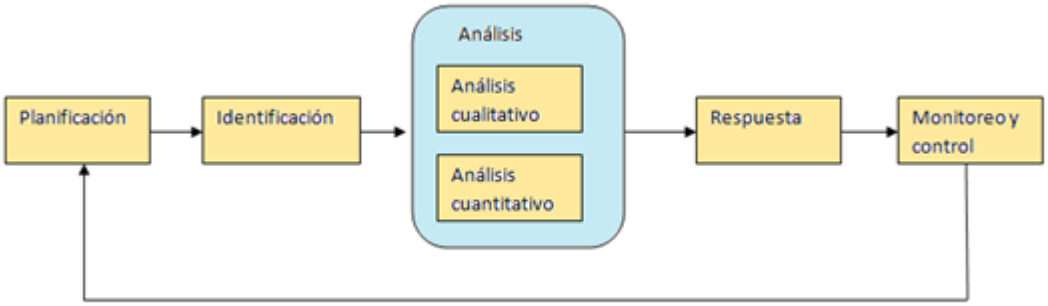

Figura 2.1. Diagrama del proceso de Gestión de Riesgos. (Lugani y Peña, 2018, p.174).

Cabe señalar que este proceso es cíclico y continuo. Puede suceder que se identifiquen nuevos riesgos o un determinado control requiera ser modificado, por consiguiente, estos eventos deberán ser evaluados a fin de determinar los riesgos asociados y actuar en consecuencia. De esta manera, se favorece la mejora continua del proceso y por lo tanto la adaptación de la organización a las necesidades cambiantes de seguridad.

2.1.3. Gobernanza, Riesgo y Cumplimiento

Como se mencionó anteriormente, los activos informáticos están presentes en gran parte de las organizaciones y sus procesos de negocio. Tal ubicuidad conduce a la necesidad de alinear el uso de las herramientas de TI con los objetivos de la organización, a la vez que se gestionan los riesgos de negocio y se verifica el cumplimiento de los controles de seguridad con diferentes regulaciones. Esta visión integral corresponde al concepto de Gobernanza, Riesgo y Cumplimiento (GRC), término introducido por la compañía PricewaterhouseCoopers en el año 2004 luego de, según Gartner (2016), varios desastres financieros corporativos altamente divulgados que dieron lugar a empresas que luchaban para mejorar sus procesos de control interno y gobernanza (como se cita en Lindros, 2017). Gobernanza, Riesgo y Cumplimiento es un modelo de gestión que describe diversas actividades organizacionales, desde preparar una auditoría hasta establecer procedimientos de monitoreo continuo de los controles internos, definir roles y responsabilidades de los procesos de negocio y usuarios de sistemas, y hasta procedimientos de análisis de datos (Papazafeiropoulou y Spanaki, 2016). En el contexto de TI, los tres componentes de este modelo deben satisfacer:

- **Gobernanza**: Las actividades relacionadas con la tecnología y/o que impliquen su uso deben estar alineadas a los procesos de negocio de la organización para facilitar así el cumplimiento de los objetivos de la misma.
- **Riesgo:** La organización debe disponer de un proceso continuo de gestión de riesgos informáticos, que identifique los activos disponibles, las amenazas a las que estos están expuestos para luego determinar la probabilidad de ocurrencia y el impacto de tales eventualidades. Este proceso además debe poder integrarse a las actividades realizadas en la organización y mejorarse continuamente para adaptarse a las necesidades cambiantes de seguridad de la información.
- **Cumplimiento:** Deben aplicarse controles de seguridad para reducir o mitigar los riesgos identificados en los activos informáticos, y asegurar en consecuencia el buen desempeño de los mismos así como el cumplimiento de normas y políticas organizacionales.

A través de la gestión de riesgos informáticos, la organización tendrá conocimiento de las amenazas reales a las que se encuentra expuesta su información, lo que le permitirá definir controles para tratar esas eventualidades, para luego monitorear su funcionamiento y en consecuencia, tomar decisiones eficientes que la ayuden a cumplir con sus objetivos, con normas y políticas. Es entonces GRC una excelente herramienta para alinear los activos informáticos a los procesos y objetivos de negocio para así mejorar la toma de decisiones generando valor a la organización.

2.2. Software relacionado con la gestión de riesgos

Como se sostiene a lo largo de este proyecto, gestionar los riesgos se ha convertido en objeto de gran relevancia para proteger la información de las organizaciones, incluso es una herramienta fundamental para el desarrollo de las mismas (Lugani y Peña, 2018). Es por ello que organizaciones como Omega, Symantec y Aloka UK se han dedicado en los últimos años a esta labor. Como resultado de ello, se han desarrollado sistemas como los mencionados a continuación.

2.2.1. Conformidad de Riesgos de Seguridad (Assurance Risk Compliance)

Esta es una herramienta web de gestión de riesgos informáticos, alineada a los principios establecidos en la norma ISO/IEC 27001:2013. Assurance Risk Compliance (ARC) no sólo gestiona riesgos sino que además colabora en la administración de activos, análisis del impacto en el negocio, y administración de usuarios. Entre sus ventajas se encuentra el soporte para la segregación de entornos, alineación con los principios de la norma ISO/IEC 27001, y facilidad para auditar las acciones del usuario (Aloka UK, 2017).

2.2.2. Administración de Activos de Software (Software Asset Management)

Desarrollado por la empresa estadounidense Symantec Corporation, este sistema está orientado a la administración de activos de software (Symantec Corporation, 2018). Para ello, define 5 procesos:

Descubrimiento de activos de software, parches instalados, versiones y comunicaciones.

Registro en la Base de Datos de Gestión de Configuraciones (CMDB) de información como paquetes de software, licencias, reglas de detección y datos de terceros.

Reconciliación del software, esto es, investigar el estado de las licencias y versiones de aplicaciones. De esta manera, la organización conocerá el estado de sus aplicaciones y podrá decidir renovar o revocar licencias.

Implementación de políticas para la instalación segura de software a partir del estado de los programas obtenido en la fase anterior.

Ejecución del software para evaluar la conformidad con el mismo e identificar oportunidades de ahorro teniendo en cuenta la frecuencia con que se utiliza el software.

2.2.3. Gestión de Riesgos en PIMS (PIMS Risk Management)

Desarrollado por la empresa noruega Omega, este software está enfocado en la identificación, evaluación y posterior reporte de riesgos. Permite definir y editar riesgos, así como generar reportes de fácil entendimiento sobre los mismos (Omega, 2018). Los riesgos identificados y las respectivas acciones correctivas o preventivas están asociados a responsables específicos dentro de la organización. Los riesgos identificados son mostrados a través de una matriz de probabilidad e impacto.

3. Problema a resolver e importancia de resolverlo

En este capítulo se detalla la problemática actual en relación a la gestión de riesgos informáticos y la necesidad que justifica el desarrollo de esta aplicación, junto con los objetivos que esta última pretende alcanzar.

En la actualidad, muchas organizaciones siguen sin comprender los riesgos de seguridad informática y en consecuencia el impacto de los mismos en su negocio. Tal como lo demuestra la firma internacional de consultoría PricewaterhouseCoopers (2018) en su Encuesta Global de Seguridad de la Información 2018, el 44% de las empresas a nivel mundial (53% en Argentina) no dispone de una estrategia general de seguridad de la información, sólo el 52% cuenta con programas de capacitación para sus empleados acerca de este aspecto (54% en Argentina), y el 54% (61% en Argentina) carece de un plan de contingencia ante la ocurrencia de incidentes. En la Figura 3.1 se muestra tales resultados y su comparación a nivel global, en Sudamérica y Argentina.

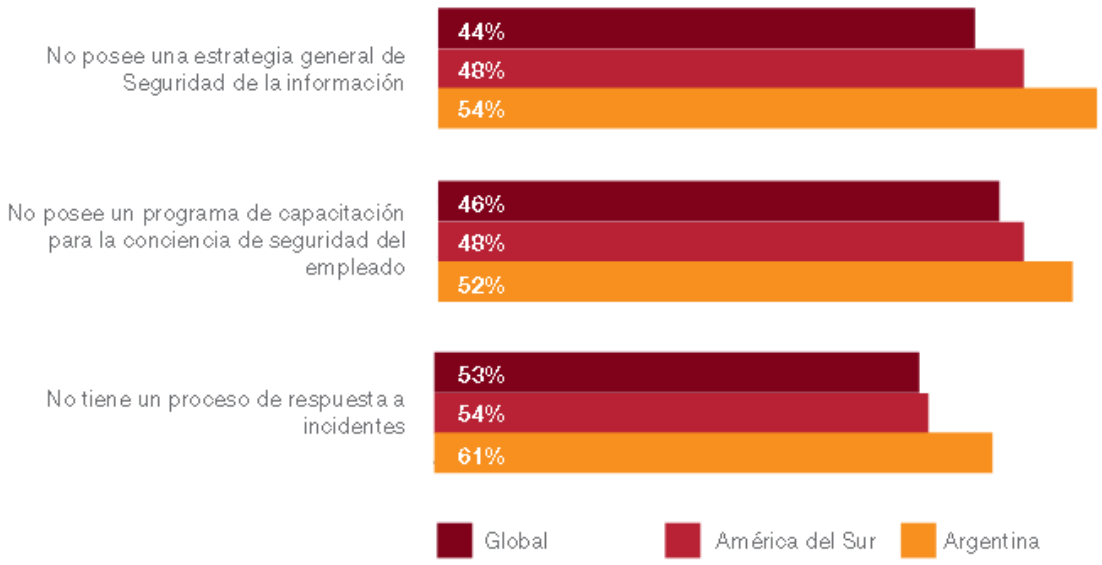

Figura 3.1. Nivel de protección de organizaciones contra amenazas de seguridad. (PricewaterhouseCoopers, 2018).

En nuestro país, muy pocas organizaciones (39%) consideran que el riesgo por sí solo impulsa a invertir en ciberseguridad. Esta situación conduce a la carencia de actividades de gestión de riesgos y consigo la no comprensión de las mismas, lo que resulta en desconocimiento del impacto de los riesgos en el negocio y por consiguiente no invertir en seguridad. Muchas veces la complejidad y costo (económico y de tiempo) que implica el implementar medidas de seguridad son causantes de esta poca inversión por parte de las organizaciones. Por lo tanto, surge la necesidad de optimizar la actividad de gestión de riesgos informáticos. Es por ello, que se requiere una solución que facilite la manera de gestionar los riesgos e identificar aquellos de mayor criticidad, que proporcione una interfaz intuitiva y que agilice el proceso de gestión de riesgos. Actualmente, las herramientas existentes suelen suministrarse a través de la filosofía Software como un Servicio (Software as a Service o SaaS en inglés). Sin embargo, debido al alto nivel de confidencialidad y criticidad de la información que estos sistemas deben tratar, implica un riesgo considerable de seguridad el confiar la misma a servidores ajenos a la organización. Benlian y Hess (2011) señalan que los principales riesgos asociados a SaaS son económicos, estratégicos y de pérdida de datos (como se cita en Jaramillo, E., 2017). Durante el año 2018, más de la mitad de los incidentes de seguridad gestionadas por la empresa israelí Check Point.se relacionan con la nube y los sistemas SaaS (Europa Press, 2018). Por consiguiente, es importante que la

solución sea ofrecida bajo la filosofía Sobre la Premisa (On Premise en inglés), es decir, debe ser requerida su instalación en los servidores de la organización cliente, sean estos ubicados físicamente allí o en la nube. Pese a que existe a nivel internacional software relacionado con la gestión de riesgos, estos implican un costo considerable, su interfaz es poco intuitiva y no abarcan los activos informáticos es la totalidad de su concepto, lo que demanda una solución mucho más simple de utilizar y que de esta forma permita al usuario enfocarse completamente a la gestión de riesgos informáticos en lugar de consumir tiempo en capacitaciones intensivas.

3.1. Importancia de la aplicación

Con la realización de este sistema se pretende contribuir al campo de la ciberseguridad, además de aportar al desarrollo de la región y la Argentina en esta área, puesto que no existe, al momento de redacción de este trabajo, software de este tipo en Río Negro. El constante crecimiento de amenazas informáticas cada vez más sofisticadas plantea la necesidad de tomar medidas de protección y concientizar a las organizaciones acerca de los riesgos que esta situación implica en el desempeño de su negocio cuando la información no está debidamente protegida. Por consiguiente, la seguridad informática cuenta con un mercado amplio y con gran potencial para seguir creciendo. El sistema desarrollado pretende facilitar a las organizaciones la administración de sus activos informáticos, los riesgos a los que éstos están expuestos y la aplicación de controles de seguridad, a la vez de ayudar a incorporar la gestión de riesgos informáticos a aquellas organizaciones que aún no han invertido en este aspecto.

3.2. Objetivos de la aplicación

Son objetivos globales de este proyecto:

- Optimizar la actividad de gestión de riesgos informáticos, facilitando la manera de identificar aquellos de mayor criticidad y la implementación de controles para reducirlos.

- Integrar los procesos de negocio, riesgos informáticos y cumplimiento de objetivos de la organización para mejorar la toma de decisiones y con ello generar valor.
- Aportar a la mejora de las organizaciones, y en particular las de Argentina, en materia de ciberseguridad.

Como objetivos específicos, se tienen los siguientes:

- Registrar y monitorear los activos informáticos de la organización así como los riesgos asociados a éstos.
- Definir, implementar y revisar controles de seguridad para los riesgos identificados.
- Permitir asociar políticas y normas organizacionales a los controles definidos.

4. Solución

Este capítulo detalla el proceso de desarrollo de la solución. Se introduce a la metodología ágil Scrum, y luego se describen las distintas etapas o sprints que se llevaron a cabo para lograr la implementación del sistema.

4.1. Metodología ágil Scrum

La globalización de la economía a comienzos de siglo, acompañada por el auge del internet, ha causado que las necesidades de negocio sean cada vez más dinámicas y por lo tanto cambien rápidamente. Esta situación exige un rápido y efectivo desarrollo de software, capaz de garantizar al cliente la competitividad en el mercado y la flexibilidad suficiente para adaptarse a sus necesidades en todo momento. Cabe señalar que las metodologías tradicionales como el modelo de desarrollo en cascada[2], caracterizadas por la rigurosidad de sus controles y el uso exhaustivo de la documentación, han demostrado ser demasiado prohibitivas y rígidas a la hora de adaptarse a los cambios en el negocio. Por ello, utilizar un enfoque tradicional de desarrollo no resulta efectivo en entornos tan cambiantes. Es así como surgieron las metodologías ágiles, modelos enfocados en el proceso incremental en el desarrollo de software, el uso reducido de documentación, la participación activa del cliente en todo el proceso y tiempos de entrega más cortos. En el año 2001 se creó la Alianza Ágil (Agile Alliance), una organización sin fines de lucro orientada a promover la filosofía ágil y ayudar a las organizaciones en su adopción (Alaimo, 2013, p.10). La mencionada organización basó esta filosofía en cuatro pilares:

1. El equipo es más importante que los procesos y herramientas, por lo que debe ser el primero quien construya su entorno de trabajo.

2. El software funcionando es más valioso que una documentación detallada, por consiguiente, esta última debe centrarse en lo esencial y ser breve.

2 Modelo de desarrollo secuencial y lineal, donde cada etapa comienza inmediatamente después de finalizar la etapa precedente.

3. La interacción constante con el cliente será lo que asegure el éxito del proyecto, en lugar de los asuntos contractuales.

4. La planificación del proyecto debe ser flexible para adaptarse satisfactoriamente a los cambios.

Pese a no reemplazar a las metodologías tradicionales en proyectos de gran ambición o que requieran controles rigurosos, el desarrollo ágil ha logrado adaptarse a las necesidades de negocio del mundo actual y su ritmo acelerado, aumentando la tasa de éxito en proyectos de software.

Dentro de este contexto de desarrollo ágil, una de las metodologías más utilizadas en la actualidad es la denominada Scrum. Este es un marco de trabajo creado para gestionar el desarrollo de productos complejos, manteniendo a su vez la calidad de los mismos y su adaptación a los cambios. Scrum no proporciona detalles acerca de cómo realizar cada tarea dentro del proyecto, sino que en su lugar genera un contexto relacional e iterativo, de constante inspección y adaptación de forma que sean los involucrados en el proyecto quienes vayan creando su propio proceso (Alaimo, 2013, p. 15). El proceso Scrum comienza con la elaboración de la Pila de Producto (Product Backlog), herramienta que contiene los requerimientos del sistema ordenados por prioridad en forma descendente. De la resultante, el equipo de desarrollo estima y selecciona aquellos requerimientos que abordará en la siguiente etapa (Sprint), a partir de una reunión de planificación (Sprint Planning). Los requerimientos elegidos conforman la Pila del Sprint (Sprint Backlog), y a partir de la misma el equipo de desarrollo acuerda las actividades a realizar. Durante la etapa de desarrollo, se realizan reuniones diarias (Daily Scrum) para revisar lo hecho el día anterior y definir lo que se hará en ese día. Una vez terminado el Sprint, se realiza una reunión de revisión con el cliente y aquellos relacionados con el negocio para detallar lo realizado en esa etapa y obtener una retroalimentación por parte de éstos. Por último, el equipo de desarrollo mantiene una reunión de retrospectiva, donde analiza su funcionamiento como tal. La Figura 4.1 representa el proceso mencionado.

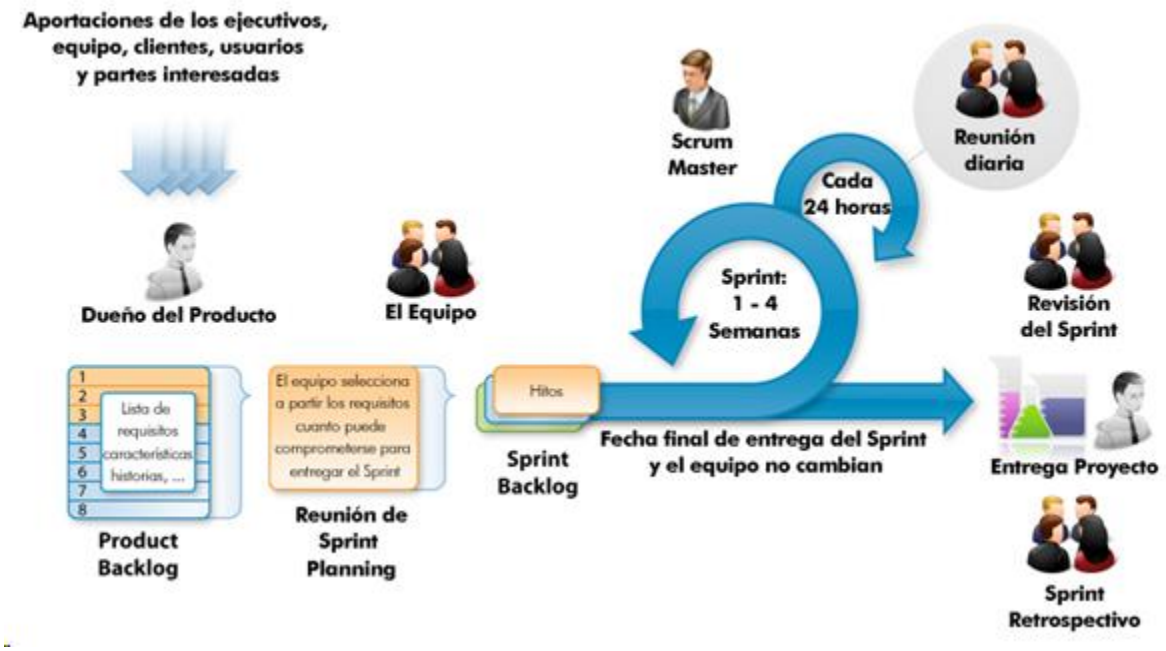

Figura 4.1. Proceso Scrum. Copyright 2012 por Isla Visual.

Scrum define los siguientes roles:

- **Dueño del Producto (Product Owner)**: representa al cliente y es quien define y prioriza los requerimientos de negocio.
- **Scrum Master**: es quien se encarga de hacer cumplir la metodología, de liderar al equipo, y ayudar al Product Owner a priorizar los requerimientos de negocio.
- **Equipo de desarrollo**: Se encarga de implementar el sistema y asegurar el cumplimiento de los requerimientos establecidos.

A través de este proceso, Scrum ha demostrado eficiencia, productividad y flexibilidad, y por esta razón es una de las metodologías de desarrollo ágil más utilizadas en la actualidad. En el libro *La guía definitiva de Scrum: Las reglas del juego* de Ken Schwaber y Jeff Sutherland se explica esta metodología de forma más detallada.

4.2. Planificación del proyecto

A continuación, se detallan los aspectos que se evaluaron durante la planificación del trabajo, que permitieron seguir un camino preciso para resolver el trabajo y alcanzar los objetivos deseados.

4.2.1. Justificación de la metodología elegida

Las características que motivaron la elección de Scrum fueron:

- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

4.2.2. Etapas del proyecto

La Tabla 4.1 muestra las fases del proyecto junto con la duración aproximada en días.

Etapa	Actividades	Entregables	Duración
Fase preliminar	Iniciación del proyecto	• Objetivos, alcances y roles del proyecto	2 días
	Relevamiento de requerimientos	• Requerimientos	7 días
Implementación del sistema	Análisis	• Product backlog priorizado • Casos de uso	7 días
	Diseño	• Modelo de datos • Arquitectura del sistema • Diagrama de clases • Plan de pruebas • Wireframes[3] • Mockups[4]	14 días
	Programación	• Sistema desarrollado	60 días

3 El *wireframe* es un boceto de la estructura de una interfaz gráfica, que hace uso sólo de texto y cuadros.

4 El *mockup* cumple una función similar al wireframe pero incluye detalle visual (imágenes).

	Verificación y validación	• Informe de pruebas realizadas • Sistema validado	10 días
Duración total del proyecto			100 días

Tabla 4.1. Cronograma de actividades para desarrollar el sistema. Elaboración propia.

Es necesario aclarar que los días aquí indicados son aproximados, debido al solapamiento de ciertas actividades. Un ejemplo de ello son las actividades de verificación y validación, que han sido distribuidas durante, al final de cada Sprint, así como en la etapa posterior al desarrollo. Por lo tanto, el desarrollo conjunto entre las actividades de programación y evaluación dificultó la definición de los días transcurridos, así como los entregables, como el informe de pruebas realizadas que se comenzó a elaborar durante la etapa de programación.

4.2.3. Roles y responsabilidades

Debido a algunos factores tales como la escasa cantidad de integrantes para conformar un equipo de desarrollo (por el carácter individual de la presente tesina), fue necesario adaptar la metodología a las características del proyecto. Por tanto, se distinguieron los siguientes roles:

- Ricardo Luis Peña: Desarrollador (Team Member) - Jefe de Proyecto (Scrum Master)
- Horacio Muñoz Abbate: Desarrollador (Team Member)
- Carlos Fabián Lugani: Desarrollador (Team Member)
- Universidad: Cliente (Product Owner)

Es importante señalar que, si bien se ha indicado específicamente un dueño del producto, el mismo sólo fue escogido para el desarrollo ágil del sistema. Este último está pensado para ser utilizado por cualquier organización que así lo desee.

4.2.4. Comunicación

La comunicación con el cliente se dio a través de las diferentes reuniones llevadas a cabo en su oficina y en la universidad, y de conversaciones en la app móvil Whatsapp, ya que esta tecnología permite una comunicación rápida y constante. La comunicación con los miembros del Laboratorio de Informática Aplicada de la UNRN se dio a través de encuentros presenciales en el lugar mencionado.

4.2.5. Riesgos del proyecto

- Terminar el proyecto fuera de los tiempos establecidos. Plan de prevención: utilizar funcionalidades ya desarrolladas para aspectos generales (login[5], manejo de permisos, ABMC[6]), estimar el esfuerzo en unidades de puntos historia en lugar de horas para gestionar los Sprints eficazmente según la complejidad de los requerimientos.
- Requerimientos ambiguos o mal definidos.
- Pérdida del código fuente del programa por asuntos del equipo. Plan de prevención: mantener copia del programa en un repositorio en la nube, a fin de disponer del código en todo momento independientemente de la computadora utilizada.
- Poca experiencia del tesista en gestión de proyectos de desarrollo de software. Plan de prevención: capacitación brindada por los miembros del Laboratorio de Informática Aplicada de la UNRN acerca de herramientas y proyectos desarrollados allí.

4.3. Seguimiento del proyecto

Para dar seguimiento a este proyecto durante su realización, se utilizaron las siguientes herramientas.

5 Login (de las palabras en inglés “log” e “in”) es el ingreso al sistema de un usuario registrado.
6 ABMC es la sigla correspondiente a las funcionalidades Alta, Baja, Modificación y Consulta.

- Google Drive para la documentación del trabajo realizado y la consecuente redacción de la presente tesina.
- Trello para seguir el estado de las actividades a través de un tablero de desarrollo, visualizando las actividades en sus diferentes estados: pendiente, en realización y finalizada.
- Diagrama de Trabajo Pendiente (Burndown) para visualizar el progreso del Sprint. Permite saber cuánto trabajo ha de realizarse para completar cada Sprint, además de controlar el cumplimiento del esfuerzo previsto y comparar con el desempeño deseado.

4.4. Desarrollo del sistema

Para el sistema desarrollado en este trabajo se utilizó una metodología ágil basada en Scrum y, por tanto, se llevaron adelante Sprints. Se destinó la primera etapa conocida como Sprint 0 para resolver cuestiones preliminares al desarrollo, tales como la formalización y priorización de los requerimientos del sistema, la selección de los recursos tecnológicos a utilizar y la previsión de historias de usuario a abordar en cada iteración.

4.4.1. Sprint 0 – Iniciación

Antes de comenzar el desarrollo de una aplicación de software, es necesario definir claramente cuáles son los requerimientos a abordar y qué herramientas son más apropiadas para tal fin. Es por eso que se destinó una etapa preliminar o Sprint 0 para ello, y en la que se definieron diferentes aspectos que a continuación se mencionan.

4.4.1.1. Historias de usuario

Cada requerimiento del sistema (funcional y no funcional) fue preservado en una historia de usuario, herramienta característica de las metodologías ágiles. En cada historia de usuario se detalla:

- **Fecha de creación:** fecha en la que se escribió la historia.
- **Autor:** el o los autor/es de la historia.
- **Versión:** versión de la historia en caso de haber sido actualizada.

- **Referencia del cambio:** en caso de ser un requerimiento actualizado.
- **Número:** identificador de la historia (ver Anexo 1 - Convenciones).
- **Usuario:** rol de usuario que solicita ese requerimiento.
- **Nombre de historia:** nombre que sintetice el objetivo de la historia.
- **Prioridad en negocio:** importancia de la historia para el cliente.
- **Puntos estimados:** esfuerzo requerido para cumplir con el requerimiento.
- **Iteración asignada:** el Sprint en el que se abordará la historia.
- **Descripción:** sigue el formato: "Como [rol de usuario], quiero [requerimiento], con el objetivo de [finalidad]" (Ver Anexo 1 - Convenciones).
- **Observaciones:** consideraciones respecto a la historia.
- **Criterio de aceptación:** la condición a cumplirse para que el usuario apruebe la funcionalidad.

A modo de ejemplo, la Tabla 4.2 muestra la historia de usuario para el requerimiento "Acceso seguro al sistema". Consultar el Anexo A para mayor información sobre todas las historias de usuario del sistema.

<table>
<tr><th>Fecha</th><th>Autor</th><th>Versión</th><th>Referencia del cambio</th></tr>
<tr><td>22/04/2019</td><td>Ricardo Luis Peña</td><td>1.0</td><td>-</td></tr>
<tr><td colspan="2">Número: HU-US001</td><td colspan="2">Usuario: Responsable de Sistemas, Responsable de Seguridad, Director Ejecutivo.</td></tr>
<tr><td colspan="4">Nombre historia: Acceso seguro al sistema</td></tr>
<tr><td colspan="4">Prioridad en negocio: Muy alta</td></tr>
<tr><td colspan="2">Puntos estimados: 8</td><td colspan="2">Iteración asignada: 1</td></tr>
<tr><td colspan="4">Descripción:

Como Responsable de Sistemas, Responsable de Seguridad y Director Ejecutivo queremos acceder al sistema a mediante un nombre de usuario y contraseña, con el objetivo de garantizar un uso más restringido y protegido de intrusos.</td></tr>
<tr><td colspan="4">Observaciones:

• El nombre de usuario debe ser único en el sistema.
• La contraseña debe ser encriptada (con algoritmo sha-1 o md5).</td></tr>
</table>

Información del usuario: • Nombre • Apellido • DNI • CUIL • Email
Criterio de Aceptación: El sistema aceptó la combinación usuario/contraseña correcta, y rechazó el par incorrecto, informando en este último caso del error.

Tabla 4.2. Historia de usuario para el requerimiento de "Acceso seguro al sistema". Elaboración propia.

Una vez definidas las historias de usuario, las mismas fueron ordenadas por prioridad en forma descendente. Así, la Tabla 4.3 muestra el Product Backlog resultante.

Nº	ID	Nombre Historia	Prioridad	Esfuerzo
1	HU-US001	Acceso seguro al sistema	Muy alta	8
2	HU-SI001	ABMC de activos	Muy alta	5
3	HU-RS001	ABMC de amenazas	Muy alta	3
4	HU-RS002	ABMC de riesgos	Muy alta	5
5	HU-DE001	Visualización de matriz de riesgos	Alta	13
6	HU-SI002	ABMC de usuarios	Alta	5
7	HU-RS004	ABMC de controles	Alta	3
8	HU-RS005	Asignación de control a riesgo	Alta	3
9	HU-RS007	Prueba de controles de seguridad	Alta	3
10	HU-DE002	Vencimiento de control sobre riesgo	Alta	5
11	HU-RS008	Historial de ejecución de controles	Alta	5
12	HU-RS009	Consulta de usuarios registrados	Alta	3
13	HU-RS003	Cálculo de la criticidad del riesgo	Media	1
14	HU-RS006	ABMC de políticas	Media	3
15	HU-DE003	Matriz dinámica de riesgos	Media	8
16	HU-US002	Diseño adaptable a cualquier dispositivo	Baja	1
Esfuerzo total (puntos historia)				74

Tabla 4.3. Pila de producto (Product Backlog) del sistema de gestión de riesgos. Elaboración propia.

4.4.1.2. Herramientas y arquitectura del sistema

Antes de desarrollar una aplicación de software es imprescindible definir una correcta arquitectura que permita estructurar lógica y físicamente el sistema de forma que sus componentes satisfagan los requisitos de calidad (ej.: desempeño, seguridad, modificabilidad) y sirvan como guía durante el desarrollo (Cervantes, 2010). Para el presente proyecto se definió una arquitectura cliente-servidor y en la que las responsabilidades se dividen en dos elementos bien definidos. El primero es el denominado Interfaz Frontal o Frontend, que se encarga de la representación visual de la información, la interacción con el usuario y su lógica; el segundo elemento es el Motor Dorsal o Backend, responsable de gestionar las peticiones recibidas del frontend, de la lógica del negocio y el manejo de los datos. La Figura 4.2 resume la arquitectura descrita.

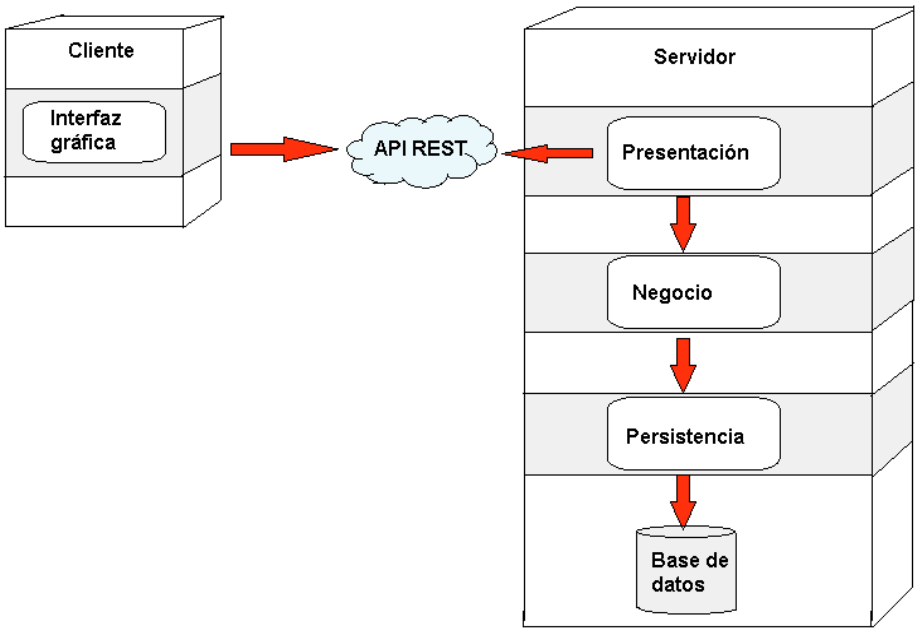

Figura 4.2. Arquitectura definida para el sistema de gestión de riesgos. Elaboración propia.

- **Frontend**: Para la interfaz de usuario se definió una arquitectura basada en componentes, que son elementos contenedores de una estructura (plantilla o template), estilo, y lógica, favoreciendo así la reutilización de los mismos. Se utilizó Angular 8 (migrado luego a la versión 9), framework[7] de código abierto desarrollado por Google, para aplicaciones web del lado del cliente. Encapsulando plantillas HTML, hojas de estilo CSS y lógica TypeScript (transpilado luego al lenguaje

7 Un framework (marco de trabajo) es una estructura con funcionalidades definidas que aceleran el desarrollo de software.

JavaScript) en cada uno de sus componentes, Angular permite la realización de aplicaciones SPA[8] totalmente dinámicas. Para los elementos visuales se utilizó Angular Material, una librería de componentes gráficos para Angular.

- **Backend**: Se utilizó para este elemento una arquitectura basada en 3 capas, lógicamente definidas:
    1. **Presentación**: Se encarga de interactuar con la capa de negocio a partir de peticiones del cliente, y representar la información resultante de tales servicios. Esta capa hace uso del patrón de diseño conocido como patrón Modelo-Vista-Controlador o MVC, que permite separar las responsabilidades de la lógica del negocio (modelo), la interfaz de usuario (vista) y el manejo de eventos y comunicaciones (controlador).
    2. **Negocio**: Implementa la lógica del negocio y brinda servicios de acuerdo al modelo de dominio.
    3. **Persistencia**: Es la responsable del acceso y tratamiento de los datos necesarios para el buen desempeño de la aplicación.

  El backend fue desarrollado en el lenguaje de programación Java debido a su estabilidad (más de 20 años) y popularidad, primero según el índice TIOBE (TIOBE, 2019), pero principalmente por su enfoque orientado a objetos. Se utilizó el framework Spring 5, que facilita la implementación de aplicaciones Java empresariales en cualquier plataforma de despliegue, y el IDE[9] de código abierto Eclipse JEE. Para la persistencia de los datos se utilizó el gestor base de datos relacional MySQL y la interfaz de persistencia JPA[10]. Para la visualización de la base de datos se utilizó la herramienta MySQL Workbench de la organización Oracle. Finalmente, la aplicación de backend se desplegó en un servidor de aplicación Apache Tomcat 9.
- **Interfaz de servicios**: Para la comunicación entre la aplicación de frontend y la de backend se utilizó una Interfaz de Desarrollo de Aplicaciones (API) que brinda servicios denominados de Transferencia de Estado Representacional (REST). En una API REST, los elementos de información que se transfieren se denominan recursos y se identifican unívocamente a través de su Identificador de Recursos Uniforme (URI). La comunicación entre el cliente y el servidor se da a través del protocolo de comunicación web HTTP[11], y se hace uso de los verbos GET, POST, PUT y DELETE para manipular los recursos (consulta, alta, modificación y baja, respectivamente). El uso de HTTP facilita el almacenamiento en caché de los servicios, otro principio de la filosofía REST. Otra característica de los servicios REST es que son sin estado (stateless), es decir, el servidor no guarda información alguna acerca del cliente y viceversa, toda la información requerida se encuentra en la petición HTTP y en las respuestas del servidor, generalmente en formato simple de JSON[12]. De esta manera,

8 Aplicación de Única Página (Single Page Application o SPA) es aquella aplicación que carga los componentes necesarios una única vez y luego solicita dinámicamente aquellos que necesite sin tener que recargar la página.

9 IDE (Integrated Development Environment) o Entorno de Desarrollo Integrado.

10 JPA Java Persistence API o API de Persistencia de Java.

11 HTTP Hypertext Transfer Protocol o Protocolo de Transferencia de Hipertexto.

12 JSON JavaScript Object Notation o Notación de Objeto JavaScript.

esta interfaz de servicios uniforme permite la escalabilidad de la aplicación, extendiendo la misma a otras plataformas cliente, como celulares smartphones o tablets. El uso de REST es muy habitual en la mayoría de las aplicaciones web.

Durante la actividad de programación se hizo uso de un VCS[13], específicamente Git, para controlar el versionado de ambas aplicaciones. Los repositorios correspondientes fueron resguardados de forma remota en la plataforma web GitLab. Otra herramienta que se utilizó fue ArgoUML, para la elaboración de los diagramas de clases, casos de uso y contexto, en UML[14].

4.4.1.3. Definición de Sprints

A partir de las historias de usuario definidas y priorizadas en el Product Backlog, se llevó adelante la última actividad de esta fase preliminar, donde se definieron las iteraciones a cubrir en el proyecto. Considerando la importancia de construir un producto mínimo viable que pueda introducirse en el mercado y en un corto periodo de tiempo, se definió un total de 3 iteraciones o Sprints: inventario de activos y riesgos, controles de seguridad y mejoras o misceláneas. Para las primeras 2 iteraciones se estableció una duración máxima de 1 mes (30 días), y la última de 2 semanas. La Tabla 4.4 detalla la información concerniente a cada iteración.

Sprint	Nombre	Descripción	Puntos Historia	Duración
1	Inventario de activos y riesgos	Administración y visualización de activos y riesgos	34	1 mes
2	Controles de seguridad	Aplicación de controles de seguridad	24	1 mes
3	Misceláneas	Mejoras en el sistema	16	2 semanas

Tabla 4.4. Iteraciones para el desarrollo del sistema de gestión de riesgos. Elaboración propia.

13 VCS Version Control System o Sistema de Control de Versiones.
14 UML Unified Modeling Language o Lenguaje de Unificado de Modelado.

4.4.2. Análisis – Casos de uso

Tal como se muestra en la Figura 4.2, se realizó en primer lugar el diagrama de contexto, que indica cuál es la relación de cada rol de usuario con el sistema.

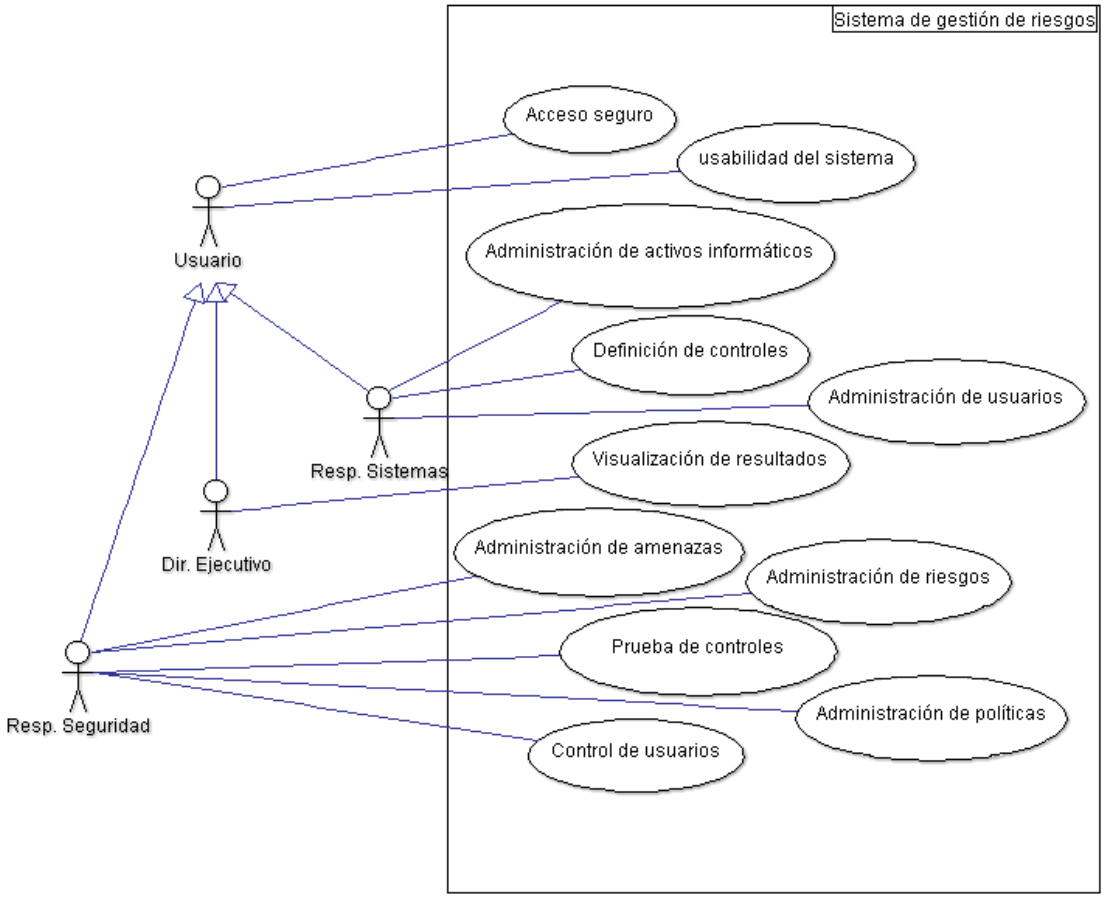

Figura 4.2. Diagrama de contexto del sistema de gestión de riesgos. Elaboración propia.

Las Figuras, 4.3, 4.4 y 4.5 presentan los casos de uso definidos para cada usuario: el responsable de sistemas, el responsable de seguridad y el director ejecutivo respectivamente.

Figura 4.3. Diagrama de casos de uso para el responsable de sistemas. Elaboración propia.

Figura 4.4. Diagrama de casos de uso para el responsable de seguridad. Elaboración propia.

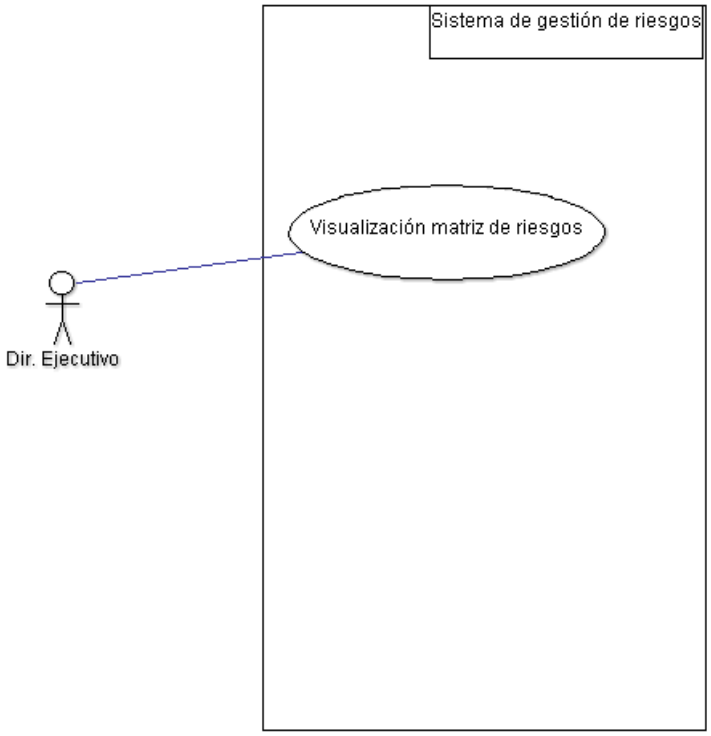

Figura 4.5. Diagrama de casos de uso para el director ejecutivo. Elaboración propia.

Para cada caso de uso se definió el escenario correspondiente, que cuenta con la siguiente información:

- **Nombre**: el nombre que identifica unívocamente al caso de uso.
- **Referencia a requerimiento**: la/s historia/s de usuario relacionada/s con el caso de uso.
- **Descripción**: resumen de qué se trata el caso de uso.
- **Actores**: el o los usuarios que interactúan con el sistema en el caso de uso.
- **Precondiciones**: las condiciones que se deben cumplir antes de la ejecución del caso de uso.
- **Flujo normal**: la interacción representada de forma secuencial y bajo condiciones normales.
- **Flujo anormal**: la interacción que se produce al ocurrir una excepción o error en el flujo normal.

- **Postcondiciones**: las condiciones que se cumplen al finalizar la ejecución del caso de uso.

A modo de ejemplo se presenta en la Tabla 4.5 el escenario para el caso de uso "Asignación de control a un riesgo". Los escenarios definidos se pueden consultar en el Anexo B de este documento.

Nombre: Asignación de control a un riesgo [CU-RS013]
Referencia a Requerimiento: [HU-RS005: Asignación de control a riesgo]
Descripción: Asignación de un control de seguridad a un riesgo determinado.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción "Asignar un control". Paso 2: El sistema presenta los riesgos registrados. Paso 3: El responsable de seguridad selecciona un riesgo. Paso 4: El sistema presenta los controles registrados. Paso 5: El responsable de seguridad selecciona un control. Paso 6: El responsable de seguridad selecciona la opción "Asignar". Paso 7: El sistema asigna el control al riesgo. Paso 8: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 2: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación. Paso 4: El sistema no encontró controles registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad asignó un control a un riesgo determinado.

Tabla 4.5. Escenario del caso de uso "Asignación de control a riesgo". Elaboración propia.

4.4.3. Diseño

Posterior al desarrollo de las historias de usuario y casos de uso, se realizó el diseño lógico y físico de la base de datos, el diseño preliminar del modelo de negocio, y el plan de pruebas a realizar.

4.4.3.1. Modelo de datos

Con la información obtenida del dominio, se diagramó el modelo de datos utilizando para ello un esquema relacional. A continuación, se detallan las tablas definidas, junto con sus campos y relaciones. Estas últimas son posibles gracias a las claves foráneas (*foreign keys*), que permiten asociar un campo de una tabla al identificador de otra, logrando así relacionar ambas entidades. Estos campos especiales se mencionan con la notación FK. Los campos subrayados indican aquellos que identificarán cada fila.

- Activo = (<u>id</u>, nombre, id_tipo(FK), descripción, ubicación, estado, versión, propietario, id_responsable (FK))
- Amenaza = (<u>id</u>, nombre, descripción, id_categoria (FK))
- Categoria_amenaza = (<u>id</u>, nombre, descripción)
- Control = (<u>id</u>, nombre, periodicidad, id_responsable (FK), descripción, valor_reduccion_riesgo)
- Definicion_control = (id, id_control (FK), id_riesgo (FK), proximo_vencimiento, estado, porcentaje_reduccion)
- Director_ejecutivo = (<u>id</u>, id_perfil(FK))
- Ejecucion_control = (<u>id</u>, fecha, estado, nombre_control, descripcion_riesgo, nombre_responsable, cuil_responsable, fecha_vencimiento id_responsable_seguridad (FK), id_definicion_control (FK))
- Perfil = (<u>id</u>, nombre, apellido, cuil, email, id_usuario(FK))
- Política = (<u>id</u>, nombre, descripción, año_readccion)
- Responsable = (<u>id</u>, id_perfil (FK))
- Responsable_sistemas = (<u>id_responsable</u>)
- Responsable_seguridad = (<u>id_responsable</u>)
- Riesgo = (<u>id</u>, descripción, valor_criticidad, probabilidad_ocurrencia, impacto_cuantitativo, impacto_cualitativo, valor_monetario, porcentaje_reduccion, id_activo(FK), id_amenaza(FK))
- Rol = (<u>id</u>, nombre, descripción)
- Usuario = (<u>id</u>, nombre_usuario, password, avatar, id_rol(FK))
- Tipo_activo = (<u>id</u>, nombre, descripción)

Relaciones muchos a muchos:

- Activos_relacionados = (<u>id_activo1, id_activo2</u>)

Control_politica = (id_control, id_politica)

4.4.3.2. Modelo de negocio

A partir de los requisitos y casos de uso analizados previamente, se realizó el diagrama de clases del modelo de negocio, tal como se muestra en la Figura 4.7. En el mismo se obviaron los métodos de recuperación y modificación de propiedades (métodos “getters y setters”).

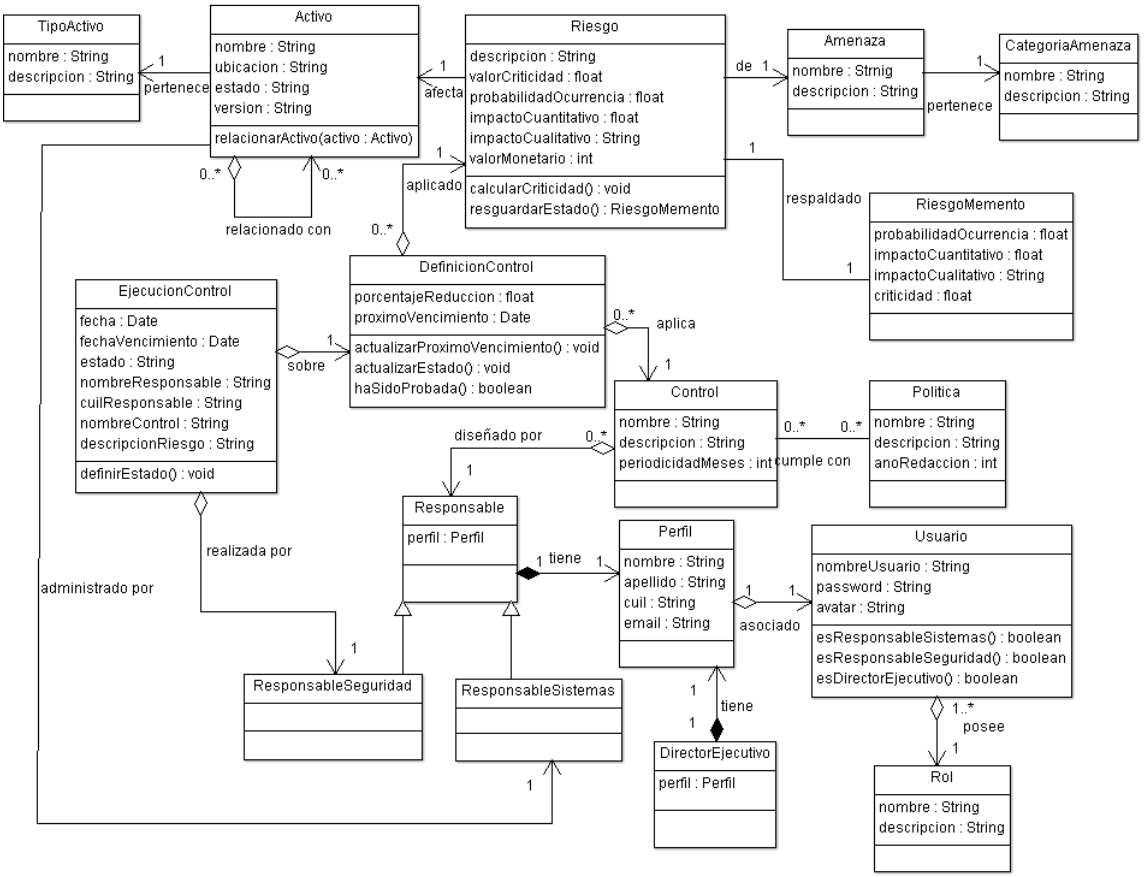

Figura 4.7. Diagrama de clases del modelo de negocio. Elaboración propia.

4.4.3.3. Plan de pruebas

A fin de garantizar la calidad del sistema, es necesario definir antes del desarrollo del mismo la forma en que se llevará a cabo la actividad de pruebas y qué aspectos estarán sujetos a revisión. Por ello se elaboró el plan de pruebas que a continuación se detalla.

Objetivos:

- Verificar y validar las funcionalidades del sistema.
- Descubrir requisitos mal especificados o ambiguos.
- Comprobar el cumplimiento de aspectos no funcionales del sistema.
- Identificar, analizar y dar seguimiento a los fallos del sistema, a fin de dar solución a los mismos.

Componentes a probar:

- Componentes del frontend.
- Controladores.
- Servicios.
- Repositorios de persistencia.
- Elementos del modelo de negocio.

Características a probar:

- Acceso restringido al sistema mediante autenticación por token.
- Manejo de roles de usuario y permisos.
- ABMC de:
    - Activos.
    - Amenazas.
    - Riesgos.
    - Controles.
    - Políticas.
- Validación de formularios.
- Implementación de controles sobre los riesgos.
- Manejo de excepciones.
- Correcta utilización de las peticiones HTTP sobre la API REST.
- Interacción con la matriz de riesgos.

Estrategia de prueba:

Se realizaron durante el desarrollo de cada iteración, pruebas unitarias destinadas a verificar las funcionalidades a nivel de clase. Estas pruebas fueron luego integradas de forma incremental para dar lugar a pruebas a nivel de componente. Una vez finalizado el desarrollo del sistema, se llevaron adelante las pruebas de desempeño, así como las de aceptación. Las pruebas unitarias e integración fueron, en su mayoría, automatizadas a través de las herramientas provistas por la dependencia Starter Test del framework Spring. Las pruebas de sistema y aceptación fueron realizadas de forma manual.

A partir de este plan de pruebas, se llevaron a cabo las actividades de verificación y validación sobre el sistema, y que se detallan en el siguiente capítulo.

4.4.4. Sprint 1-Inventario de activos y riesgos

En esta primera iteración se desarrolló la funcionalidad núcleo de la aplicación, es decir, el ingreso autenticado al sistema y el registro y visualización de activos informáticos, así como de los riesgos asociados a estos. Las historias de usuario contempladas en este Sprint fueron:

1. HU-US001-Acceso seguro
2. HU-SI001-ABMC de activos
3. HU-RS001-ABMC de amenazas
4. HU-RS002-ABMC de riesgos
5. HU-DE001-Visualización de la matriz de riesgos

Para el seguimiento de esta etapa, se implementó el correspondiente tablero en Trello, en el que se detallan las tareas específicas relacionadas con los requisitos abordados, tal como ejemplifica la Figura 4.6.

Figura 4.6. Tablero del Sprint 1. Recuperado de https://trello.com/b/2VB1nJqs

4.4.4.1. Acceso seguro

El primer requerimiento abordado fue la historia de usuario "Acceso seguro al sistema", que se detalla en la Tabla 4.2 y se resume a continuación:

- **ID**: HU-US001.
- **Nombre**: Acceso seguro al sistema.
- **Prioridad**: muy alta.
- **Esfuerzo estimado**: 8 puntos de historia.
- **Criterio de aceptación**: aceptar par usuario-contraseña válido y rechazar el incorrecto.
- **Observaciones**: nombre de usuario único y contraseña encriptada. Otra información: nombre, apellido, CUIL, email.

En primer lugar, se definieron en el modelo las clases `Usuario` y `Rol`, siendo entre ellas una relación de uno a muchos (como se puede apreciar en el apartado 4.4.3.2 -Modelo de negocio). Estas clases (y las demás clases del modelo) fueron definidas mediante la anotación `@Entity`, perteneciente a la especificación JPA. De esta forma, cada clase del

modelo es considerada una entidad persistente (tabla) en la base de datos, y es la propia API quien se encarga de crear las entidades, mapearlas a tablas y relacionarlas. Por lo tanto, las dos clases mencionadas son utilizadas para validar la autenticidad del usuario y autorizar su acceso en función de los permisos que disponga según su rol. Las clases del modelo fueron agrupadas en el paquete `com.tfc.riesgos.models` del proyecto Java.

Para la implementación de la seguridad del backend se hizo uso del framework Spring Security, que provee herramientas para la implementación de filtros de autenticación y control de permisos, además de protección contra ataques como CSRF[15]. El mecanismo de seguridad provisto por este framework para esta aplicación se vale principalmente de las siguientes clases:

- `WebSecurityConfig`: se encarga de configurar el comportamiento del sistema en relación a las peticiones HTTP recibidas. En esta aplicación, sólo se permite el acceso libre a la dirección de ingreso ("`/login`"), mientras que cualquier otra petición debe pasar por los filtros de autenticación y autorización definidos. El uso de la anotación `@EnableWebSecurity` permite la integración del framework con el entorno web, mientras que la anotación `@EnableGlobalMethodSecurity(prePostEnabled = true)` habilita la autorización basada en roles de usuario.
- `UsernamePasswordAuthenticationFilter` y `OncePerRequestFilter`: el primero es un filtro básico que se encarga de validar la autenticidad de un usuario según su nombre y contraseña, mientras que el segundo asegura una única ejecución del filtro para cada consulta, útil para la autorización.
- `UserDetailsService`: permite obtener, valiéndose de un objeto DAO[16], los detalles de un usuario en particular. La información obtenida se representa mediante la clase `UserDetails`.

15 CSRF Cross Site Request Forgery o Falsificación de Petición en Sitios Cruzados.

16 DAO Data Access Object u Objeto de Acceso a Datos

- `AuthenticationManager`: es una interfaz que se encarga de procesar la solicitud de autenticación.

Por lo tanto, el uso de estas clases ya proporciona seguridad al acceso a la API REST, evitando la intrusión de individuos no deseados.

Al uso de Spring Security, se añadió la autenticación mediante tokens de acceso. Debido a que la arquitectura REST es sin estado, no existen sesiones de usuario en el servidor, y por consiguiente este es incapaz de recordar al usuario entre las llamadas a la API (Álvarez, 2018). Para evitar entonces añadir el usuario y contraseña en cada llamada, se utiliza un token de acceso, que es una cadena de texto encriptada. El token es autocontenido, es decir, que dispone de la información necesaria para garantizar la autenticidad del usuario. Por ello, el servidor genera el token en la primera petición del cliente, y en las siguientes peticiones de este sólo debe revisar el token generado. En la Figura 4.7 se muestra esta interacción.

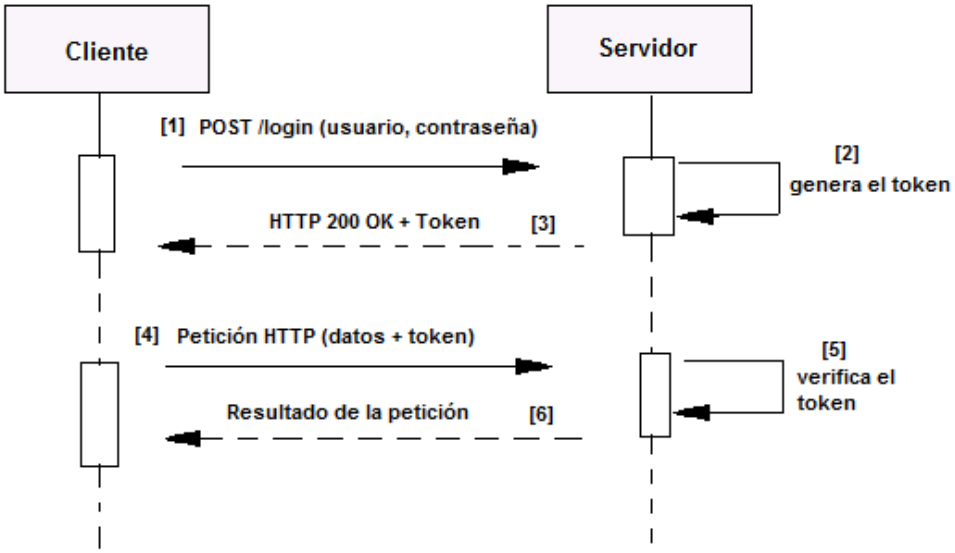

Figura 4.7. Proceso de autenticación por token. Elaboración propia.

En este proyecto, se utilizó la especificación JSON Web Token (JWT). Esta consiste en un objeto JSON definido en el estándar RFC 7519 como una forma segura de representar un conjunto de información entre dos partes (Stecky Efantis, 2016). El token JWT se compone de tres partes: una cabecera (header) con información sobre el algoritmo de cifrado a utilizar, un cuerpo (payload) que contiene la información del usuario, y una firma (signature) que se

utiliza para validar la autenticidad del token. El uso de JWT facilita la autenticación de un usuario dado. Sin embargo, al no estar encriptado, el token puede ser accedido por intrusos, por lo que no es aconsejable incluir información sensible (ej.: la contraseña). Para la autenticación basada en tokens, se utilizaron las siguientes clases:

- `JWTAuthenticationFilter`: este filtro extiende a `UsernamePasswordAuthenticationFilter` y añade la implementación de JWT a la autenticación.
- `JWTAuthorizationFilter`: extiende a `OncePerRequestFilter`, y se encarga de verificar la existencia del token, validarlo e implementar el filtro.
- `TokenProvider`: genera el token mediante el uso de la librería JWT para Java. Esta clase es utilizada en los dos filtros descritos anteriormente.
- `UsuarioService`: implementa `UserDetailsService`, y se encarga de ejecutar los servicios relacionados con el usuario.

En conclusión, el proceso de autenticación y manejo de permisos utilizando Spring Security y la seguridad por tokens JWT es el siguiente:

1. La solicitud de acceso es interceptada por el filtro de autenticación (`JWTAuthenticationFilter`), quien obtiene las credenciales indicadas por el usuario e invoca al gestor de autenticación (`AuthenticationManager`) para que realice la verificación de su identidad, haciendo uso del servicio `UsuarioService`.
2. Si la autenticación falla por credenciales incorrectas, el filtro retorna un error de código 401 (no autorizado). En caso contrario, solicita al proveedor de token (`TokenProvider`) el JWT correspondiente a ese usuario. El token incluye (entre otros datos) el rol del usuario, mediante la propiedad (claim) "CLAIM_TOKEN". El nombre del rol está acompañado del prefijo "ROLE_".
3. El token generado es insertado y enviado por el filtro en la cabecera Authentication de la respuesta HTTP. El token JWT se acompaña del prefijo "Bearer ", que es un estándar para este tipo de token.

4. Cuando el usuario autenticado realiza una petición al servidor, el filtro de autorización (`JWTAuthorizationFilter`) verifica la existencia y validez de la cabecera que contiene al token, y en caso de no ser correcta, genera un error. En caso contrario, obtiene el nombre del usuario del token y luego solicita el objeto `Usuario` al servicio `UsuarioService`.
5. Entonces, solicita al proveedor de token la validación del mismo con respecto a la información del usuario y su rol. Por último, prepara el contexto de seguridad con los permisos indicados, y aplica el filtro correspondiente. Este filtro verifica que la acción solicitada en la petición esté permitida para el rol del usuario, para lo que hace uso de la anotación `@PreAuthorize("hasRole('ROLE_xxxxx')")` en la acción (método) del controlador correspondiente.

De esta manera concluye la implementación del requisito de acceso seguro en el backend. Las clases descritas se encuentran en el paquete `com.tfc.riesgos.security`, a excepción de `UsuarioService` que, como todos los servicios de esta aplicación, pertenece al paquete `com.tfc.riesgos.services` del proyecto Java.

Para validar y verificar la funcionalidad implementada, se llevaron a cabo tests de integración en las capas de persistencia, servicios y controladores. Cabe mencionar que en este requisito no se realizaron pruebas de unidad, puesto que no hubo necesidad de probar funcionalidad de forma unitaria. El framework Spring facilita el desarrollo de las pruebas a través de la anotación `@SpringBootTest`, que establece automáticamente el contexto de aplicación, donde se ejecutan las pruebas indicadas con la anotación `@Test`. La Figura 4.8 muestra un caso de prueba del DAO relacionado con la gestión de usuarios, en el que se verifica la búsqueda de un usuario por su nombre. La ejecución automatizada del caso utilizando la herramienta JUnit fue exitosa, corroborando así la validez de la funcionalidad sometida a verificación.

```java
@Test
public void traerUsuarioPorNombre() {

    Usuario resultado = usuarioDAO.findByUsername("lpena");

    // condición para aprobar el test
    assertThat(resultado).isNotNull();
}
```

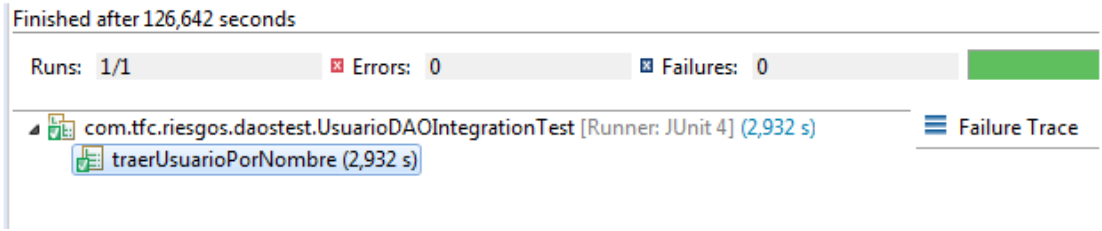

Figura 4.8. Ejecución de caso de prueba del DAO de usuarios. Elaboración propia.

De esta manera se realizaron todos los tests de integración de la capa de persistencia, que se encuentra en el paquete `com.tfc.riegos.daostest`, en la carpeta `src/test/java` del proyecto Java. Para verificar la capa de servicios, se utilizó la dependencia Mockito, con la que se simuló el uso de la capa de persistencia. De esta manera, fue probada la integración del servicio con el modelo; por último, se habilitó la capa de persistencia y se integró al servicio. La Figura 4.9 muestra la ejecución exitosa del caso en el que se solicita al servicio retornar la información de un usuario inexistente. Como este caso se realizaron todos los tests de la capa de servicio.

```java
@Test
public void traerUsuarioInexistente()
{
    String nombreUsuario = "no existo";
    Usuario resultado = usuarioService.findByUsername(nombreUsuario);

    assertThat(resultado).isNull();
}
```

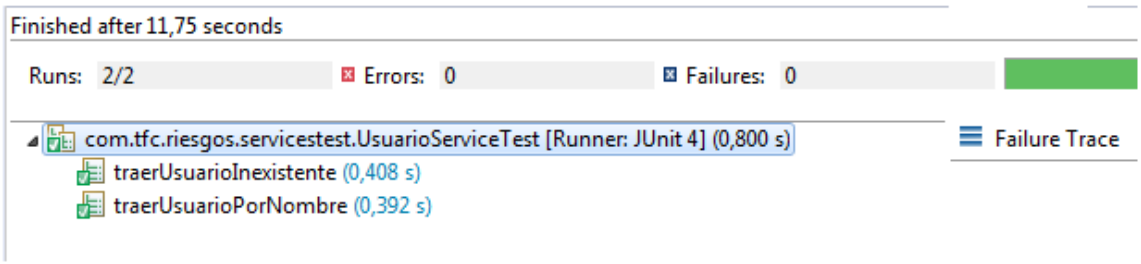

Figura 4.9. Ejecución de un caso de prueba del servicio de usuarios. Elaboración propia.

La integración con el controlador se probó con la herramienta RESTClient en el navegador Firefox. La misma permite elaborar peticiones HTTP, comunicarse con el servidor, y analizar la respuesta de este. En la Figura 4.10, se realizó una petición de ingreso al sistema, proporcionando un nombre de usuario y contraseña correctos. La operación fue

exitosa y el servidor generó el token y lo devolvió en la cabecera Authentication, además del código de estado 200 (OK).

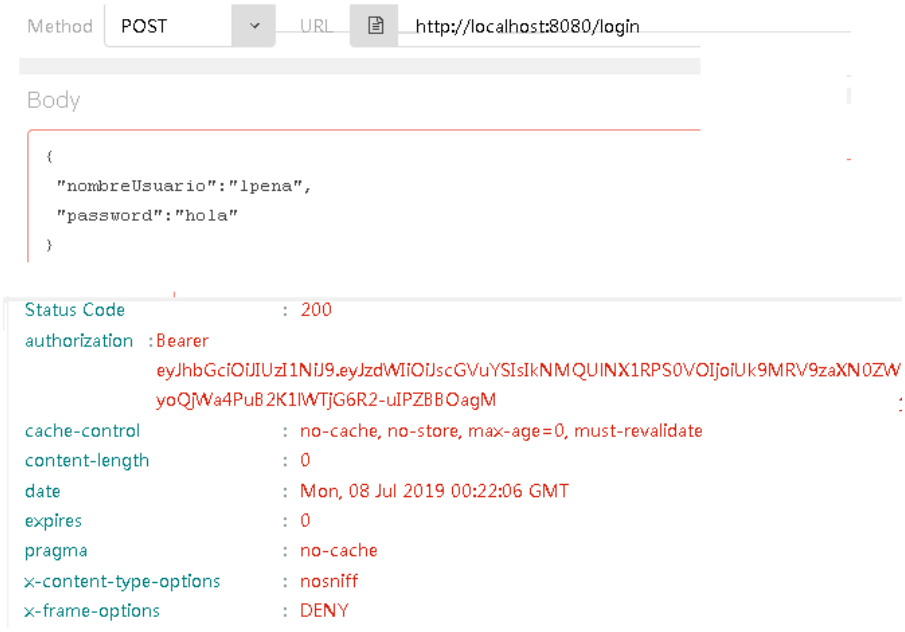

Figura 4.10. Verificación de ingreso al sistema. Elaboración propia.

Para la parte del frontend, el acceso autenticado se implementó a través de los siguientes elementos principales:

- `LoginComponent`: es quien presenta el formulario de ingreso al usuario y, una vez completo, verifica que los valores suministrados sean potencialmente válidos (no vacíos, dentro de una cantidad de caracteres, etc.). Luego, envía el formulario al servicio de autenticación (`AutenticacionService`) para solicitar el ingreso al sistema al servidor. Si la autenticación es correcta, este componente redirige al usuario a la página principal (home), cargando el componente `HomeComponent`.
- `AutenticacionService`: este servicio se encarga de enviar las credenciales del usuario al servidor, y en caso de acceso permitido, guarda el token recibido en la memoria local del navegador. De esta forma, la sesión iniciada se mantiene aún cuando se cierre la pestaña.
- `SesionModel` y `UsuarioModel`: se encargan de modelar la sesión de usuario y el usuario autenticado respectivamente.
- `JwtInterceptor`: su función es interceptar las peticiones HTTP al servidor, recuperar el token de la memoria del navegador (a través de `AutenticacionService`) e insertarlo en la cabecera de la solicitud.
- `AutorizacionGuard`: verifica que el usuario tenga los permisos necesarios para acceder a una determinada ruta.

Una vez que el usuario logra autenticarse, es redirigido al componente `HomeComponent`. Este corresponde al módulo `HomeModule` y es el núcleo de la SPA, puesto que carga una única vez los componentes de la cabecera (`HeaderComponent`), el menú (`SidebarComponent`) y el pie (`FooterComponent`). Luego, carga dinámicamente el contenido principal en `MainComponent`. A continuación, la Figura 4.11 muestra un ejemplo de usuario autenticado al sistema.

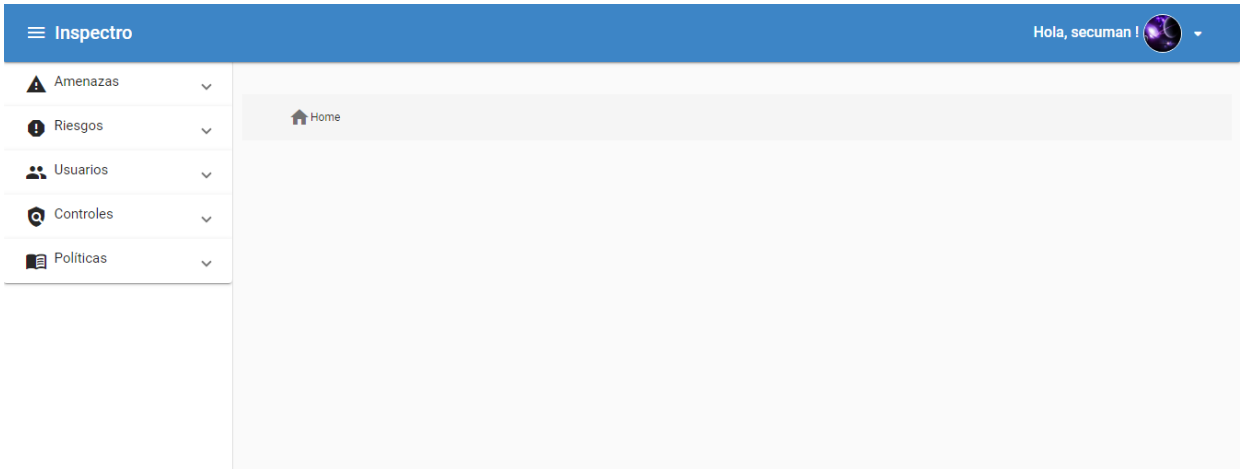

Figura 4.11. Ejemplo de ingreso al sistema. Elaboración propia.

En conclusión, con los elementos explicados en esta sección se completa el desarrollo del requerimiento de acceso autenticado al sistema.

4.4.4.2. ABMC de activos

La segunda historia a abordar fue la de ABMC de activos, con la siguiente información:

- **ID**: HU-SI001.
- **Nombre**: ABMC de activos.
- **Prioridad**: muy alta.
- **Esfuerzo estimado**: 5 puntos de historia.
- **Criterio de aceptación**: se cargó, modificó, eliminó y consultó un activo correctamente.
- **Observaciones**: datos del activo: tipo, nombre único, descripción, ubicación, estado, versión, responsable, activos relacionados.

De la misma manera que en el requisito anterior (y que en los subsiguientes), se definieron inicialmente las clases relacionadas con el modelo de negocio. Puesto que el activo tiene asociado un responsable a su cargo, fue necesario definir la clase `ResponsableSistemas`, que extiende a la clase `Responsable`. Esta última tiene asociado un

usuario y rol específicos, por consiguiente, existe una relación de uno a uno con Usuario. Luego, se definió la clase `Activo`, asociada con `ResponsableSistemas` y una nueva clase denominada `TipoActivo`, que representa la clasificación del activo informático descrita en el capítulo 2 (hardware, software, comunicación, etc.). Ambas relaciones son de uno a muchos, es decir, un activo pertenece a un único responsable/tipo mientras que un responsable/tipo puede tener asociado uno o varios activos. Los activos, además, pueden estar relacionados entre sí.

La capa de persistencia de activos está definida por el repositorio `ActivoDAO`, que extiende a `JpaRepository`, interfaz provista por Spring Data JPA y que dispone de métodos genéricos de alta, baja, modificación y consulta de entidades persistidas. Tal interfaz es extendida por todos los repositorios de esta aplicación, debido a su facilidad de uso, donde cada repositorio sólo debe indicar la entidad que representa, el tipo de dato del identificador de los objetos de esta, y la anotación `@Repository`. No obstante, las consultas específicas (personalizadas) fueron definidas manualmente utilizando la anotación `@Query` de Spring. Para la persistencia de los tipos de activo se implementó el repositorio `TipoActivoDAO`, mientras que `ResponsableDAO` se encarga de la entidad responsable. Adicionalmente, se creó el repositorio `ResponsableSistemasDAO` para consultas específicas relacionadas con este tipo de responsable; este repositorio extiende a `ResponsableDAO`.

La capa de servicio cuenta con la clase `ActivoService`, mientras que a nivel de API REST se encuentra el controlador `ActivoController`, del que se enumeran sus principales métodos en la Tabla 4.6. Este último se encarga de transferir los recursos solicitados por el cliente en forma de Objetos de Transferencia de Datos o DTOs. Estos son objetos planos (sin lógica de negocio) que permiten retornar la información necesaria para el cliente, sin necesidad de exponer el recurso tal como se encuentra persistido. De esta manera, se logra una mayor separación entre la persistencia de un objeto y su representación en el negocio. En este caso de uso se utilizaron los objetos `ActivoDTO` y `ActivoUpdateDTO`, que representan la información de un activo a leer y actualizar respectivamente. Los objetos DTO de esta aplicación se encuentran en el paquete `com.tfc.riesgos.dtos`; y son generados automáticamente utilizando la dependencia ModelMapper, que permite convertir objetos de una clase a otra (Halterman,2019). La lógica de convertir DTOs a activos y viceversa

(haciendo uso de la dependencia mencionada) se delegó a la clase `ActivoDTOMapper`, que implementa la interfaz genérica `DtoMapper`, definida para este fin; estas clases se encuentran en `mappers`, dentro del paquete de DTOs. El uso de estos objetos favorece además la mejora en el rendimiento del sistema, puesto que permite agrupar información de diferentes clases en un único objeto, evitando así sucesivas llamadas al servidor para obtener la información por separado.

URI	Método HTTP	Descripción	Rol autorizado	Respuesta
/activos	GET	Consulta de activos	Resp.Seguridad/Sistemas	200 - OK
/activos/id	GET	Consulta de un activo	Resp.Seguridad/Sistemas	200 - OK
/activos	POST	Alta de activo	Resp. Sistemas	201 - Created
/activos/id	PUT	Modificación de activo	Resp. Sistemas	200 – OK
/activos/id	DELETE	Baja de un activo	Resp. Sistemas	200 - OK

Tabla 4.6. Métodos ABMC de activos de la API REST. Elaboración propia.

En relación a la verificación de la funcionalidad, la Figura 4.12 muestra la ejecución exitosa de un caso de prueba unitaria sobre la clase `Activo` del modelo de negocio. Este caso consiste en relacionar un activo con otro una única vez, es decir, evitando relacionar un activo repetidas veces en un mismo activo destinatario.

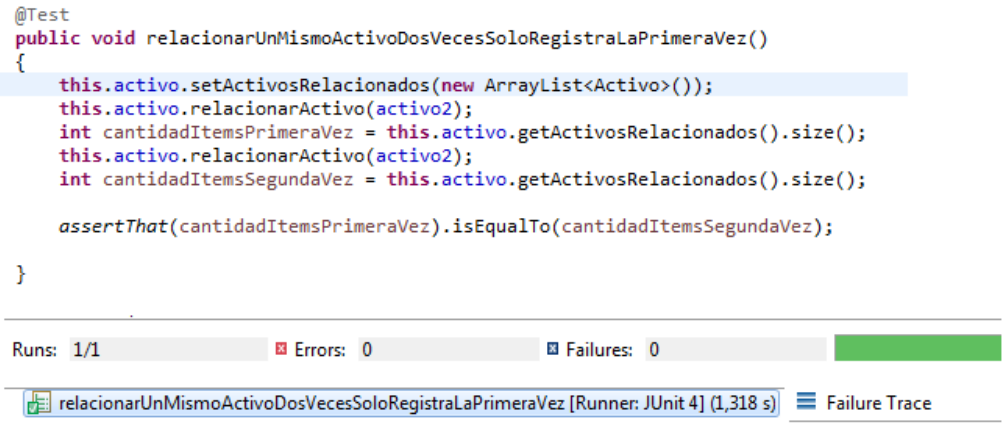

Figura 4.12. Ejecución de caso de prueba unitaria sobre un activo. Elaboración propia.

A continuación, se describe la implementación de cada operación del ABMC de activos, principalmente su integración con el frontend. Estas funcionalidades se encuentran en el módulo `ActivosInformáticosModule` del proyecto Angular.

1. **Alta**: para este caso se definió un componente llamado `NuevoActivoComponent`, que presenta un formulario a completar con los datos del nuevo activo. Los campos de este formulario están sujetos a validadores, de manera que la información ingresada debe cumplir con diversos criterios para que el formulario pueda ser enviado. Es necesario destacar que un activo debe tener un nombre único para evitar así inconsistencias en el sistema y confusión en los usuarios. En consecuencia, el nombre ingresado es verificado asincrónicamente (sin refrescar la pantalla) en el servidor mediante una consulta HTTP GET de un activo con ese nombre. Si la respuesta contiene un recurso, entonces el nombre existe y por ello es inválido. Considerando la reducción del rendimiento del sistema al enviar una solicitud al servidor cada vez que se pulsa una tecla con la única finalidad de verificar si un nombre existe, tal operación sólo se efectúa cuando se han aprobado los demás validadores sobre este campo y se pierde el foco (se hizo click en otro campo). El usuario también puede relacionar activos seleccionando de una lista aquellos que desee asociar al nuevo activo. Para evitar que se proporcione información incorrecta al servidor, se deshabilitó el botón de envío mientras existan campos incorrectos en el formulario, tal como se muestra en la Figura 4.13.

Figura 4.13. Formulario de registro de activo con campos vacíos. Elaboración propia.

No obstante, el componente vuelve a verificar la información en el momento de ser enviada, por lo que de esta manera se aplica un doble control sobre la misma. El formulario es redirigido al servidor mediante el servicio `ActivosService`, que efectúa una petición capturada por el controlador de activos. Adicionalmente, el servidor también verifica la información recibida antes de proceder a su registro. Esta forma de control sobre la información proporcionada por el usuario se repite para todos los requerimientos implementados. Por último, luego de registrar el activo, el servidor retorna un DTO con la información de este, junto con un enlace en la cabecera Location de la respuesta HTTP, que indica la ruta hacia el nuevo recurso. Una vez recibida esta respuesta (con código de estado 201 - Created), el componente despliega una notificación confirmando el éxito de la operación y reinicia los valores del formulario en caso de que el usuario desee registrar otro activo.

2. **Consulta**: puesto que cada responsable de sistemas tiene activos a su cargo, se implementó el componente `MisActivosComponent`, que muestra los activos registrados por ese responsable. Para ello el componente verifica que el usuario activo disponga del rol mencionado y recupera entonces su información para luego solicitar, mediante una petición de consulta (GET) al servidor, los activos asignados a este responsable. Finalmente, la información obtenida es presentada en una tabla facilitada por el componente `MatTable` de Angular Material. Esta tabla permite además mostrar la información por páginas y en forma ordenada. En la última columna de la tabla se disponen las opciones de modificar o dar de baja un determinado activo, tal como lo muestra la Figura 4.14.

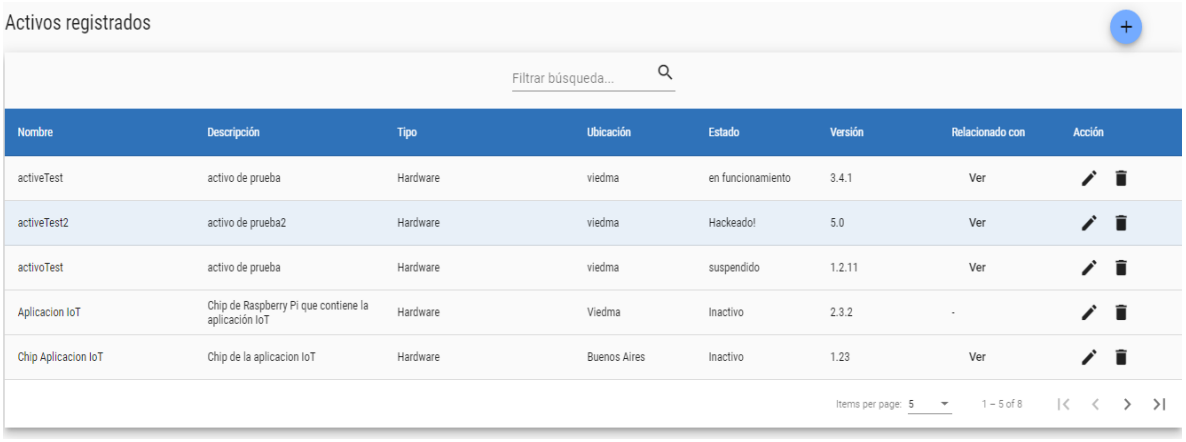

Figura 4.14. Ejemplo de tabla con activos informáticos registrados. Elaboración propia.

3. **Baja**: el componente `EliminarActivoComponent` es quien se encarga de dar de baja un determinado activo, y para ello, sólo despliega un diálogo solicitando la confirmación de la operación. En caso afirmativo, envía el identificador del activo a `ActivosService`, quien a su vez solicita la baja al servidor. Este último efectúa la eliminación del activo del sistema y una vez respondida la solicitud con código 200 (OK), el componente informa del éxito de la operación y cierra el diálogo. Por último, se actualiza automáticamente la tabla de activos.

4. **Modificación**: para actualizar un activo se implementó el componente `ModificarActivoComponent`, que es accedido seleccionando la opción de modificar en la tabla de activos registrados. Este componente despliega la información del activo elegido en un formulario. El usuario puede modificar cualquier información

del activo a excepción del responsable, y para la validación del nombre se procede de la misma manera que en el alta del activo (mediante consultas asincrónicas al servidor). También puede relacionar más activos o desasociar los existentes seleccionando/deseleccionando de la lista. Una vez modificada y validada la información del activo, esta es enviada en una consulta al servidor a través de un DTO de tipo `ActivoUpdate` para su actualización. Una vez validada la información, el servidor transforma el DTO a una entidad `Activo` y la persiste. Luego de registrada la operación, el sistema retorna un DTO con la información actualizada del activo y un código de estado 200 (OK). Por último, desde el componente notifica al usuario acerca del éxito de la operación y se actualiza la información del activo correspondiente en la tabla.

En conclusión, de esta manera se implementó el requerimiento de alta, baja, modificación y consulta de los activos informáticos registrados en el sistema. Para las demás historias de usuario relacionadas con el ABMC, se realizó el mismo procedimiento. En consecuencia, estas historias no serán descritas con el mismo nivel de detalle que la presente, puesto que no agregan valor al contenido de esta tesina.

4.4.4.3. ABMC de amenazas

El siguiente requerimiento a desarrollar fue la administración de amenazas, como se indica en la siguiente historia de usuario:

- **ID**: HU-RS001.
- **Nombre**: ABMC de amenazas.
- **Prioridad**: muy alta.
- **Esfuerzo estimado**: 3 puntos de historia.
- **Criterio de aceptación**: se cargó, modificó, eliminó y consultó una amenaza correctamente.
- **Observaciones**: datos de la amenaza: nombre único, descripción, categoría.

En la capa de negocio, se creó la clase `Amenaza`, que dispone de un nombre único, una descripción y la categoría a la que pertenece. Para esta última propiedad se creó la clase `CategoriaAmenaza`, asociándose de esta manera a `Amenaza` en una relación de uno a muchos. En la capa de persistencia, se definieron los repositorios `AmenazaDAO` y `CategoriaAmenazaDAO`, para el tratamiento de la amenaza y su categoría respectivamente. En la siguiente capa se desarrolló la clase `AmenazaService`, que se encarga de gestionar los

servicios relacionados con las amenazas y sus categorías. A nivel de API REST se implementó el controlador `AmenazaController`. En este caso no se utilizan DTOs para intercambiar la información entre el cliente y el servidor, puesto que una amenaza está representada por información básica, indivisible. Por consiguiente, no tiene demasiado sentido generar un DTO con la misma información que el objeto persistido en la base de datos (entidad), puesto que se estaría duplicando la clase y se reduciría el rendimiento de la aplicación.

Para verificar y validar este requerimiento, se implementaron pruebas de unidad para la clase del modelo de amenaza, así como también se evaluó la integración con el repositorio, el servicio y el controlador de amenazas. La Figura 4.15 muestra los casos de prueba de integración con el repositorio de amenazas. Como se aprecia en la misma, estos han respondido satisfactoriamente, por ello es correcto afirmar que la capa de persistencia funciona de acuerdo a lo deseado. La Figura 4.16 muestra un ejemplo de prueba de integración añadiendo la capa de servicio de amenazas.

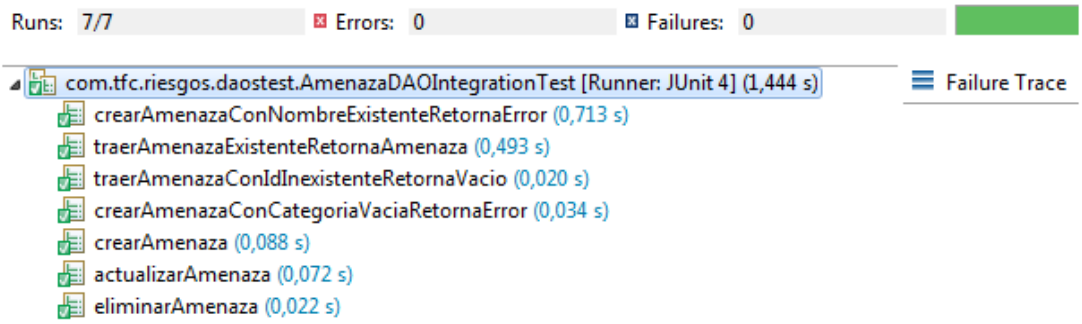

Figura 4.15. Casos de prueba de integración del repositorio de amenazas. Elaboración propia.

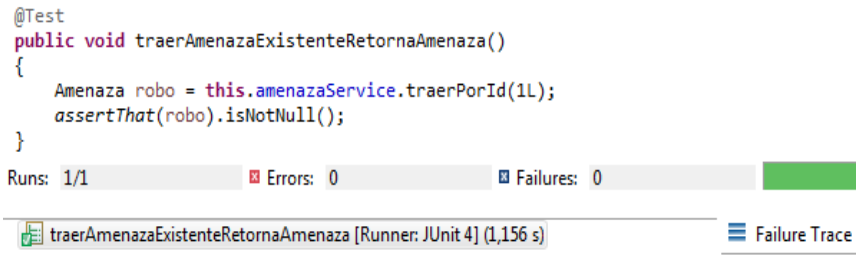

Figura 4.16. Caso de prueba de integración del servicio de amenazas. Elaboración propia.

4.4.4.4. ABMC de riesgos

Los detalles de este requerimiento son los que se mencionan a continuación:

- **ID**: HU-RS002.
- **Nombre**: ABMC de riesgos.
- **Prioridad**: muy alta.
- **Esfuerzo estimado**: 5 puntos de historia.
- **Criterio de aceptación**: se registró, actualizó, consultó y dio de baja un riesgo satisfactoriamente.
- **Observaciones**: un riesgo debe estar asociado a una amenaza y un activo. El valor de criticidad de un riesgo debe ser calculado en función de su probabilidad de ocurrencia e impacto. Los valores de ocurrencia e impacto deben ser probabilísticos, y no deben ser 0 o 1.

En primer lugar, se implementó en el modelo de negocio la clase `Riesgo`, que está asociada a un activo y una amenaza. Además de una descripción, el riesgo se compone de un valor de criticidad, probabilidad de ocurrencia, impacto cuantitativo y porcentaje de reducción (este último para la aplicación de controles). Adicionalmente, un riesgo puede disponer de un valor de impacto cualitativo e incluso un valor monetario asociado. La entidad creada es persistida en la base de datos utilizando el repositorio `RiesgoDAO`, que es accedido a través del servicio `RiesgoService`. En la capa a nivel de API se implementó el controlador `RiesgoController`, que se comunica con el cliente a través de objetos DTO según la operatoria a realizar. Para la generación de estos objetos se definió la clase `RiesgoDTOMapper`, que hace uso de `ModelMapper` para tal fin. Puesto que el riesgo es actualizado luego en función de los controles que le serán aplicados, es importante mantener persistida la información original del mismo. Para ello, se aplicó el patrón de diseño Memento, que permite resguardar los estados de un objeto para luego restaurarlos cuando este lo requiera. Para tal fin se definió la clase `RiesgoMemento`, que mantiene los valores de impacto, ocurrencia y criticidad del riesgo al momento de crearse este último.

Como se mencionó anteriormente, es importante que un riesgo no disponga de valores de probabilidad extremos, esto es, no debe permitirse la inclusión de los valores 0 o 1 en el impacto y la ocurrencia de la eventualidad. Esto se debe a que en caso de haber un evento con probabilidad de 1 (100%), entonces no sería un riesgo sino una certeza de que la amenaza se concretará; y en el caso de un evento con una probabilidad nula (0%), entonces no es un

riesgo sino un imposible, y por consiguiente, no ha de considerarse. Además, es muy difícil (cuando no imposible) tener la completa certeza de que una determinada eventualidad no sucederá de ninguna manera. En consecuencia, se realizaron principalmente pruebas de unidad sobre el modelo, tal como se ejemplifica en la Figura 4.17, que muestra el resultado de ejecutar el caso donde se establece el valor de probabilidad e impacto a 1, arrojándose una excepción.

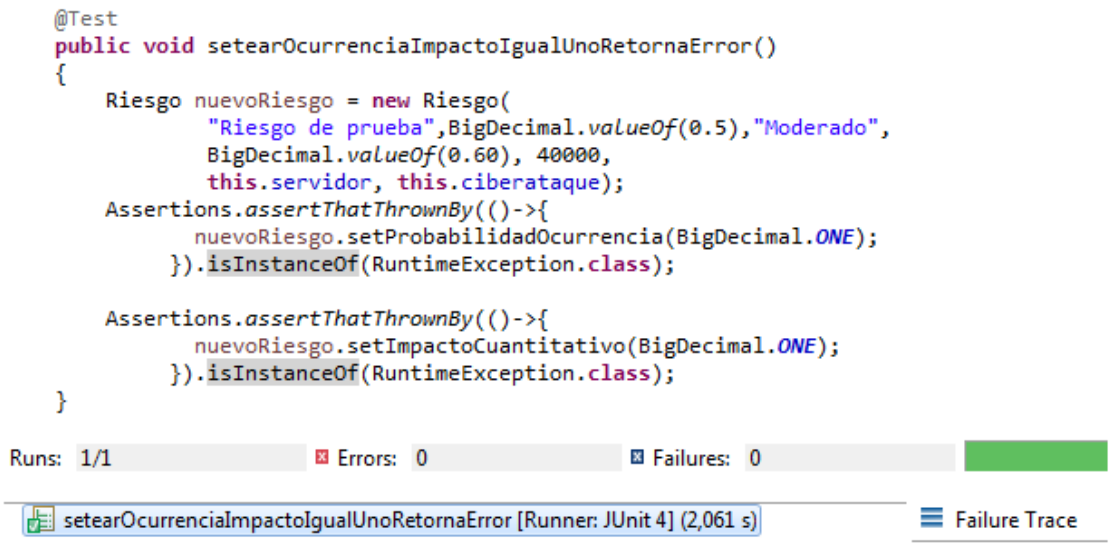

Figura 4.17. Ejecución de prueba unitaria la probabilidad e impacto de un riesgo. Elaboración propia.

En la Figura 4.18 se presenta la ejecución del caso de prueba para un riesgo con ocurrencia e impacto con valores nulos (es decir, de 0), que también supera el test satisfactoriamente. Por lo tanto, se ha comprobado que la verificación de la lógica de negocio relacionada con el riesgo fue exitosa.

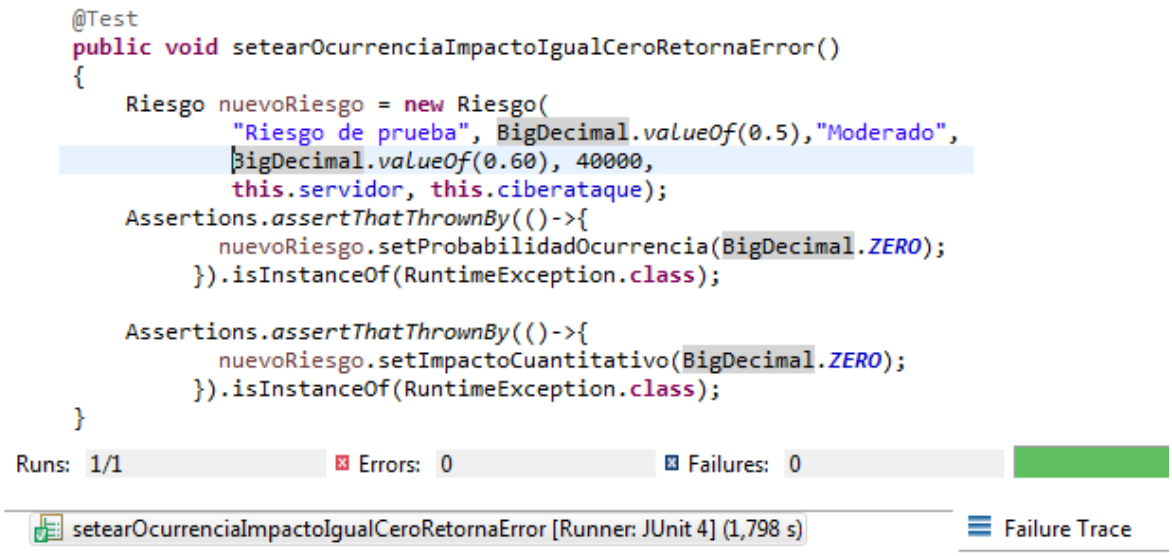

Figura 4.18. Ejecución de prueba unitaria la probabilidad e impacto de un riesgo. Elaboración propia.

En la Figura 4.19 se muestra el formulario de registro de un riesgo desde la experiencia de usuario en Angular. Este formulario se encuentra en el componente `NuevoRiesgoComponent`, dentro del módulo `RiesgosModule`. Para reducir la complejidad al usuario en el momento de ingresar los valores de probabilidad de ocurrencia e impacto, se definieron escalas de valores, obtenidos desde la base de datos a través del servidor. Esta información se representa en la clase `Escala`, que dispone de un nombre, una descripción y una lista de ítems, representados en la clase `ItemEscala`. Cada ítem presenta una descripción, un valor y un color (este es opcional). Estas clases se encuentran en el paquete `com.tfc.riesgos.utils` en la aplicación Java. De esta manera, el usuario puede ingresar valores predefinidos y con una breve descripción para facilitar su comprensión y elección. Adicionalmente, se permite seleccionar un valor de impacto en tres posibles factores: costo, tiempo y desempeño. Esta forma de selección de valores aplica también a la modificación de un determinado riesgo.

Figura 4.19. Formulario de registro de un nuevo riesgo al sistema. Elaboración propia.

4.4.4.5. Visualización de la matriz de riesgos

El siguiente requerimiento (y último de este Sprint) corresponde a la historia de usuario:

- **ID**: HU-DE001.
- **Nombre**: Visualización de matriz de riesgos.
- **Prioridad**: muy alta.
- **Esfuerzo estimado**: 13 puntos de historia.
- **Criterio de aceptación**: el sistema cargó la matriz con los valores y colores correctamente definidos.
- **Observaciones**: eje X: impacto. Valores posibles: despreciable, marginal, moderado, crítico, catastrófico. Eje Y: ocurrencia. Valores posibles: excepcional, improbable, probable, posible y cierto. Valores de criticidad y color asociado: mínimo (<10% - verde), moderado (entre 10% y 40% - amarillo), crítico (>10% - rojo).

Para este requerimiento, se definió en primer lugar la matriz de riesgos desde el backend. De esta manera, se logra una uniformidad en el armado de la matriz en caso de que

esta sea implementada por diferentes clientes (navegador, dispositivos móviles, etc.), puesto que todos harán uso de la misma información. Para ello, se implementó la clase `MatrizRiesgos`, que reutiliza la clase Escala para definir los ejes y la escala de criticidad. La matriz además se compone de una lista descendente de filas (`FilaMatrizRiesgos`), es decir, que la primera fila de la lista es la que se corresponde con los valores más altos, mientras que la última es aquella con los valores mínimos. En la Figura 4.20 se muestra el diagrama de clases de la matriz de riesgos y las relaciones descritas. Cada fila se compone además de una lista de celdas, que contienen la información necesaria para clasificar los riesgos.

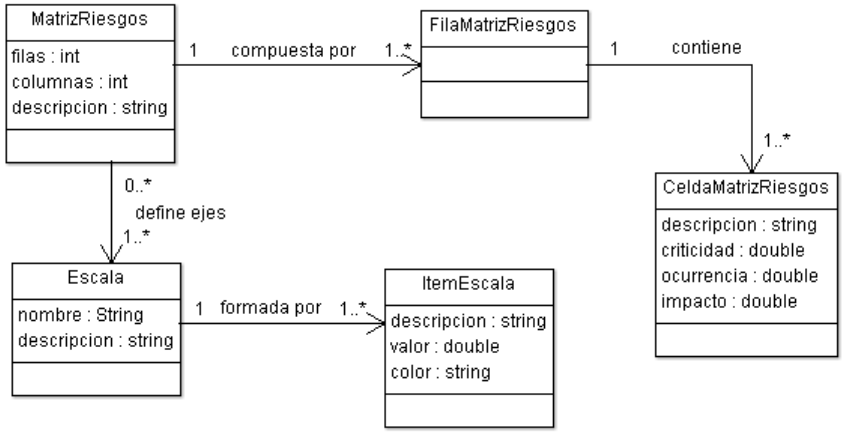

Figura 4.20. Diagrama de clases de la matriz de riesgo y sus relaciones. Elaboración propia.

Esta matriz es obtenida de la base de datos a través del repositorio `MatrizRiesgosDAO`, del que hace uso el servicio de riesgos. Desde la API, es el controlador `RiesgoController` quien envía la matriz al cliente.

Desde el frontend, se definió el componente `MatrizRiesgosComponent` dentro del módulo de riesgos. Este hace uso del diseño basado en grillas (Grid List) de Angular Material para visualizar la matriz de riesgos. Esta estructura permite mostrar contenido en cuadrículas dinámicas e interactivas, por tal motivo fue elegida para representar la matriz en lugar de la tabla HTML. Al iniciar el componente mencionado, este solicita, a través del servicio de riesgos, la información de la matriz al servidor. Este último carga la información de la matriz de la base de datos y la retorna al cliente. Una vez obtenida la matriz, esta se carga en el

componente, quien primero define las escalas de probabilidad, impacto y criticidad. Adicionalmente, añade una etiqueta del valor del eje Y en cada fila. Esto se debe a que la estructura de grillas de Angular se genera automáticamente en función de la cantidad de columnas y filas definidas, y lo hace en forma horizontal (por filas, de izquierda a derecha). Por lo tanto, no es un inconveniente para los valores del eje X (se genera una fila recorriendo la lista de valores de la escala), pero sí lo es para la escala Y. En consecuencia, es necesario indicar a qué valor del eje Y se corresponde cada fila, para visualizar finalmente el eje vertical en la matriz. Al momento de cargar la matriz, por cada celda aplica el color correspondiente según su criticidad, muestra su descripción al colocar el mouse sobre ella, y en caso de tener riesgos asociados muestra un botón para ver el detalle de los mismos. Para facilitar la comprensión al usuario, se muestra información de referencia de cada escala y color. La Tabla 4.7 muestra la referencia utilizada para asignar color en función de la criticidad, mientras que la Figura 4.21 muestra un ejemplo de la matriz de riesgos resultante.

Nivel de riesgo	**Porcentaje de criticidad**	**Color asociado**
Mínimo	<10%	
Moderado	>10% y <40%	
Crítico	>40%	

Tabla 4.7. Referencia de colores y valores de criticidad de riesgos. Elaboración propia.

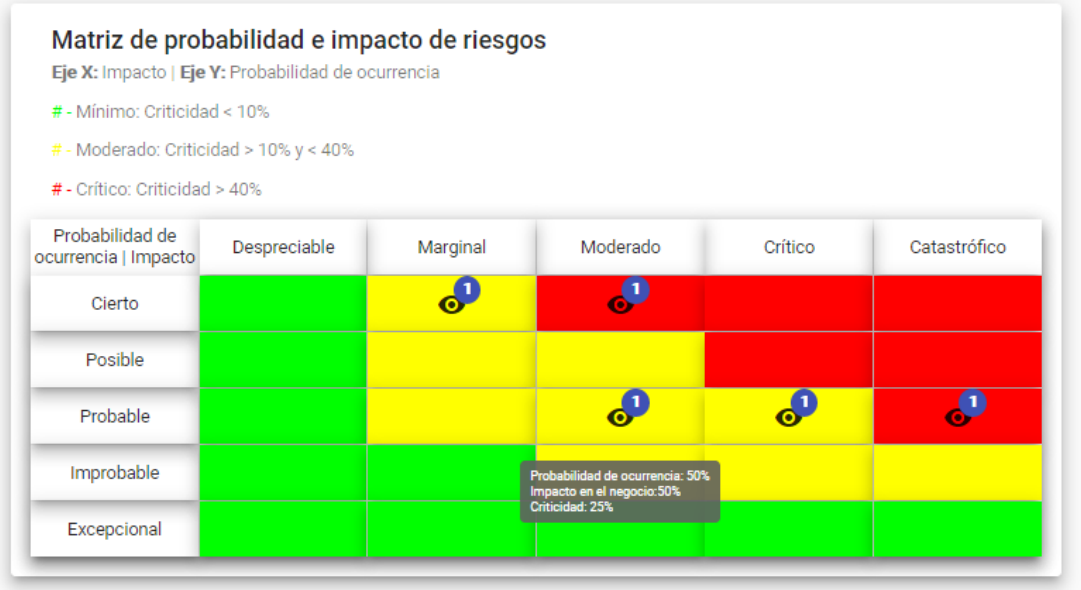

Figura 4.21. Ejemplo de matriz de riesgos presentada en la aplicación. Elaboración propia.

4.4.5. Sprint 2 – Controles de seguridad

En esta segunda etapa se desarrollaron las funcionalidades relacionadas con la definición y aplicación de controles de seguridad, así como también la gestión de usuarios. Tal como en el Sprint anterior, se implementó un tablero en Trello para el seguimiento de las tareas definidas para esta iteración.

4.4.5.1. ABMC de usuarios

El primer requerimiento a abordar en esta etapa fue la administración de usuarios del sistema, resumida a continuación.

- **ID**: HU-SI002.
- **Nombre**: ABMC de usuarios.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 5 puntos de historia.
- **Criterio de aceptación**: se creó, consultó, actualizó y borró un usuario del sistema.
- **Observaciones**: además de las credenciales de usuario, este debe contar con información de perfil: nombre, apellido, CUIL y un email. El nombre de usuario, el CUIL y el email deben ser únicos en el sistema.

En relación al modelo de negocio, se implementó la clase `Perfil`, que reúne la información profesional del personal de la organización (nombre, apellido, CUIL e email). Un perfil además está asociado a un usuario específico, y por ello, ya no es necesario que los responsables de la organización (sistemas, seguridad y ejecutivo) estén directamente relacionados con su cuenta de usuario. De esta manera, se desacopló el modelo de negocio de la representación en un sistema informático, puesto que, en caso de ya no requerir de usuarios, estos se pueden quitar sin afectar de forma alguna a un responsable o ejecutivo, puesto que sólo se elimina la relación del usuario con el perfil profesional. Este último, no obstante, es indivisible del sujeto, y es la información que este conoce, por lo que un miembro de la organización está relacionado con su perfil.

Cada usuario puede actualizar su información de cuenta con excepción del rol, que sólo puede ser modificado por el responsable de Sistemas. Este último puede, además, crear nuevos usuarios en el sistema. La Figura 4.22 muestra el formulario de registro de un nuevo usuario. En este proceso se verifica la disponibilidad del nombre de usuario mediante consultas asíncronas al servidor, así como la validez del CUIL y el email. En el caso de la contraseña, esta debe cumplir con las siguientes reglas: contar con un mínimo de ocho

caracteres, al menos una letra mayúscula y minúscula, un dígito y algún carácter especial. Esta es una forma de mantener claves seguras y, en consecuencia, menos propensas a ser vulneradas por terceros. Adicionalmente, se aplica otro control durante el cambio de contraseña, donde se solicita el ingreso de la clave anterior y la repetición de la nueva. La validación de la información del usuario también se realiza en el servidor. A fin de mantener información como el avatar (imagen de perfil) de los usuarios, se definió un proyecto en Google Firebase, que destina un servicio llamado Cloud Firestore, dedicado a almacenar archivos de forma remota (Google, 2019). Por lo anterior, al momento de registrar un nuevo usuario, este también es registrado en este servicio proporcionando su email y contraseña. Luego, al ingresar al sistema, el usuario se conecta automáticamente a Firebase y obtiene información como su imagen de perfil, la que además puede cambiar.

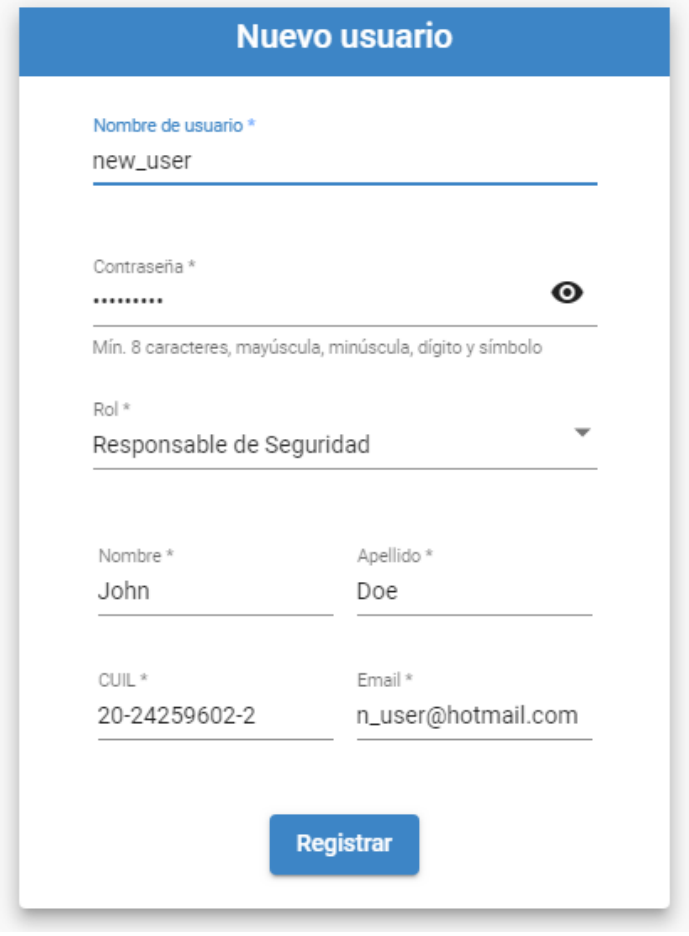

Figura 4.22. Formulario de registro de un usuario. Elaboración propia.

4.4.5.2. ABMC de controles

En este requerimiento se abordó la definición, consulta, modificación y eliminación de controles de seguridad. La historia de usuario es la siguiente:

- **ID**: HU-RS004.
- **Nombre**: ABMC de controles.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 3 puntos de historia.
- **Criterio de aceptación**: se creó, consultó, actualizó y borró un control del sistema.
- **Observaciones**: además de un nombre, un identificador y una descripción, el control debe mantener la información de la periodicidad en meses y del responsable que lo creó. La periodicidad del control debe ser a lo sumo de un año (12 meses). Un control, opcionalmente, puede estar asociado a una o varias políticas.

Para abordar este requerimiento, se implementó en primer lugar la clase `Control` en el modelo de negocio. Esta se encuentra asociada con la clase `Responsable`, puesto que tanto el responsable de Sistemas como de Seguridad pueden registrar un control. Es importante destacar que un control de seguridad es una actividad genérica, es decir, puede ser aplicado en diferentes riesgos que involucran diferentes activos, en escenarios similares. Por ello, el control no conoce riesgo alguno. Puesto que la administración de políticas corresponde al siguiente Sprint, no fue necesario relacionarlas con los controles. En las capas de repositorio, servicio y API, se implementaron las clases `ControlDAO`, `ControlService` y `ControlController` respectivamente.

Con respecto a la validación de la lógica de negocio, se desarrollaron casos de prueba unitaria sobre el modelo, donde se analizaron las propiedades de este, con mayor énfasis en los valores de periodicidad del control. Tal como se muestra en la Figura 4.23, estos casos han sido ejecutados satisfactoriamente y por ello, la lógica de negocio relacionada con los controles de seguridad actúa de acuerdo a lo esperado.

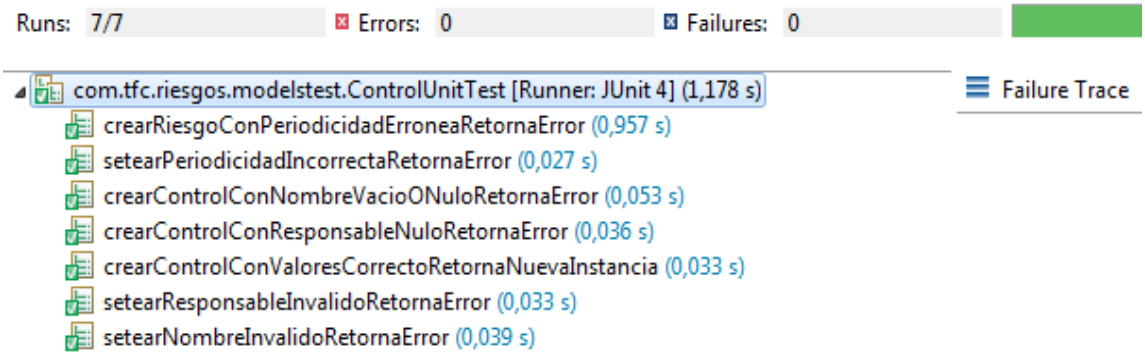

Figura 4.23. Ejecución de pruebas unitarias sobre controles. Elaboración propia.

4.4.5.3. Asignación de controles a los riesgos

En este requerimiento, el responsable desea poder especificar qué control aplicar a un riesgo determinado. A continuación, se resume la historia de usuario definida para ello:

- **ID**: HU-RS005.
- **Nombre**: Asignación de control a riesgo.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 3 puntos de historia.
- **Criterio de aceptación**: se asoció un control a un riesgo, y se puede observar la definición creada.
- **Observaciones**: un control no puede ser asignado a un mismo riesgo más de una vez.

Es preciso indicar que se consideraron diferentes etapas para la definición y aplicación de controles de seguridad. Como se mencionó en el requerimiento anterior, un control constituye una actividad genérica, es decir, puede ser aplicado en escenarios similares pero con diferentes activos y/o amenazas. Ejemplo de esto es la autenticación de personal, puesto que puede ser aplicado para reducir el riesgo de intrusión en un sistema informático, así como también del ingreso físico no autorizado a una sala de servidores. En ambos casos se realiza el mismo control, pero con diferentes acciones y recursos, así como el porcentaje de reducción del riesgo también podría variar. En consecuencia, un control puede ser definido para múltiples riesgos y, con ello, habrá múltiples definiciones para ese control.

Asignar un control de seguridad a un riesgo consiste, por consiguiente, en registrar una nueva definición de ese control. Esta información se encuentra contenida en la clase `DefinicionControl` implementada en el modelo. La misma mantiene un registro del control y riesgo asociados, así como el porcentaje en que este último se verá reducido mediante la aplicación del control. Adicionalmente, esta clase dispone del estado de la definición

(probado o pendiente) y de la fecha del próximo vencimiento de la misma, calculada a partir de la periodicidad del control. El repositorio `DefinicionControlDAO` es quien se encarga de la persistencia de la clase descrita, accedida mediante el servicio de controles, invocado desde el controlador. En la Figura 4.24 se muestra la asignación del control al riesgo desde la interfaz de usuario. La lógica para registrar una nueva definición de un control es la siguiente:

1. Se verifica que el control no se encuentre asignado a ese riesgo. En caso afirmativo, se cancela la operación e informa de la situación.
2. Se crea la nueva definición de control y se persiste en la base de datos.
3. Aumenta el porcentaje de reducción de ese riesgo en función del valor indicado en la definición. Este porcentaje es acumulativo, y en caso de llegar a 100 o superar tal cifra, se cancela la operación y se genera una excepción. Esto es debido a que no es posible tener la completa certeza de que un evento de seguridad no sucederá nunca.
4. Por último, se informa al usuario del éxito de la operación y se retorna un DTO con la información de la nueva definición (asignación) de un control sobre un riesgo.

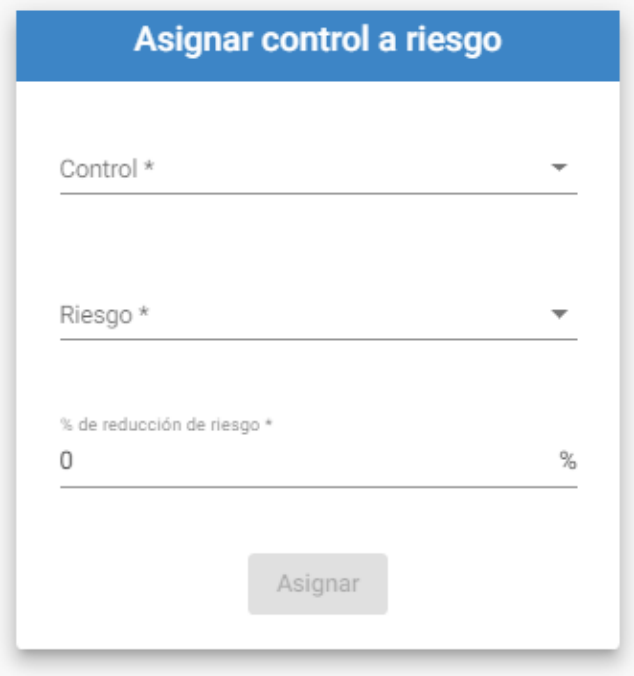

Figura 4.24. Asignación de un control de seguridad a un riesgo. Elaboración propia.

4.4.5.4. Prueba de los controles de seguridad

El responsable desea justificar la ejecución de un control sobre un riesgo, indicando al sistema que efectivamente este fue aplicado. El resumen de la historia de usuario es el siguiente:

- **ID**: HU-RS007.

- **Nombre**: Prueba de controles de seguridad.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 3 puntos de historia.
- **Criterio de aceptación**: al indicar un control como probado, el sistema registró el historial de ejecución, y por ello figura en las ejecuciones de los controles.
- **Observaciones**: se debe indicar el responsable que ejecutó el control, la fecha de la misma y la fecha de vencimiento. El sistema debe permitir ejecuciones automáticas y efectivas de controles programadas por el usuario.

Así como se distinguió al control de su definición (asignación a un riesgo), esta última es diferente a la aplicación efectiva del mismo, esto es, el riesgo no se verá reducido hasta que se haya ejecutado (probado) el control. Es necesario entonces que el responsable indique la aplicación del control para que esta se haga efectiva en el sistema. En consecuencia, es en ese momento cuando la ejecución del control es generada.

Esta ejecución es representada en la clase del modelo `EjecucionControl`, que se corresponde a una definición de control específica (`DefinicionControl`). La nueva clase registra además el responsable que la generó, la fecha en que se realizó la operación y la fecha de vencimiento del control. A partir de estas últimas, es posible indicar el estado de la ejecución, es decir, si se realizó en término o fuera de este. Es necesario destacar que esta clase representa el resultado de una transacción y, por ello, debe poder ser susceptible a posteriores consultas y auditorías. Por lo tanto, debe persistir aún cuando se haya dado de baja alguna de sus dependencias (definición de control, responsable, etc.). En consecuencia, la ejecución del control registra además la información relevante sobre el riesgo, el control y el responsable. De esta manera, al recuperar de la base de datos la ejecución del control, la misma permitirá acceder a su información aun cuando el control, riesgo o responsable se hayan dado de baja en el sistema. Las clases `EjecucionControlDAO`, `ControlService` y `ControlController` son las encargadas de las operaciones relacionadas con este requerimiento en las capas de persistencia, servicio y controlador, respectivamente. La información recuperada de la ejecución del control es enviada a través del DTO `EjecucionControlDTO`. En la Figura 4.25 se muestra el listado de aquellas definiciones de controles que necesitan ser probadas por el responsable (pendientes), para ello en cada fila de la tabla se encuentra la opción de ejecutar el respectivo control. Cuando esta acción se ejecuta, sucede la siguiente lógica:

1. Se verifica que la definición del control no disponga de una ejecución vigente. En caso afirmativo se cancela la operación, puesto que no es necesario volver a ejecutar una definición de control en vigencia.
2. Se crea la nueva ejecución del control y se calcula su estado.
3. La definición de control correspondiente establece su estado como “probado” y actualiza la fecha de su próximo vencimiento.
4. El riesgo correspondiente reduce su criticidad de acuerdo al porcentaje de reducción de la definición del control. Con esta reducción, el riesgo cambia su probabilidad de ocurrencia e impacto, y por ello es reubicado en la matriz de riesgos.
5. Por último, se persisten la ejecución creada y los cambios efectuados, y se envía la primera a través de un DTO al cliente.

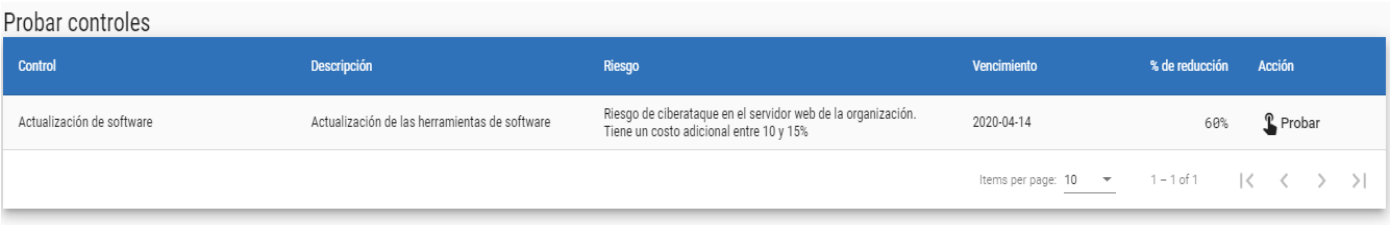

Figura 4.25. Vista de la aplicación de controles a riesgos. Elaboración propia.

Adicionalmente, el usuario puede programar la ejecución automática de un control determinado, de forma que el mismo sea efectivamente aplicado sobre el activo en cuestión. Para tal funcionalidad, se utilizó Quartz, una librería de código abierto que facilita la programación de tareas (Software AG, 2019). De esta manera, la tarea generada por el usuario es delegada a esta dependencia, que hace uso de expresiones especiales denominadas expresiones Cron. Las mismas permiten planificar períodos flexibles de ejecución de eventos, como se ejemplifica en la Figura 4.26. Esta funcionalidad se encuentra en el paquete `jobs` de la aplicación Java.

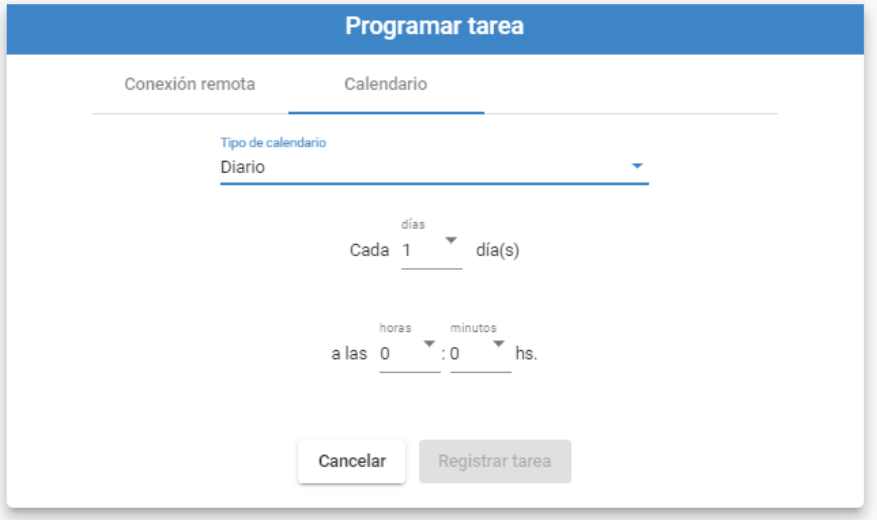

Figura 4.26. Ejemplo de programación de una tarea. Elaboración propia.

Para la comunicación remota entre la aplicación y los activos se utilizó el software de mensajería RabbitMQ a través de Spring AMQP (Pivotal Software, 2020). Este es un proyecto de código abierto ampliamente utilizado y que permite el envío de mensajes de forma asíncrona. De esta manera, el emisor publica un mensaje sin esperar su recepción. El funcionamiento de RabbitMQ se vale de los siguientes conceptos:

- Intercambio (Exchange). Es el intermediario entre el emisor y el receptor del mensaje, es decir, recibe el mensaje publicado y se encarga de remitirlo a su destino. Por lo tanto, no existe una comunicación directa entre los participantes del mensaje.
- Cola (Queue). Almacena los mensajes recibidos del exchange. Cada participante debe tener asociada una cola si desea recibir y leer mensajes.
- Clave de enlace (Binding key). Sirve de identificación para una cola determinada. Esta clave debe ser provista por el receptor al momento de asociar (registrar) una cola al exchange.
- Clave de ruteo (Routing key). Es la clave que debe proporcionar el emisor del mensaje e indica hacia quién va dirigido este último. Esta clave debe coincidir con la clave de enlace de la cola receptora.

La Figura 4.27 muestra la arquitectura del sistema de mensajería para la aplicación desarrollada. La funcionalidad de mensajería se encuentra en el paquete `amqp` del proyecto.

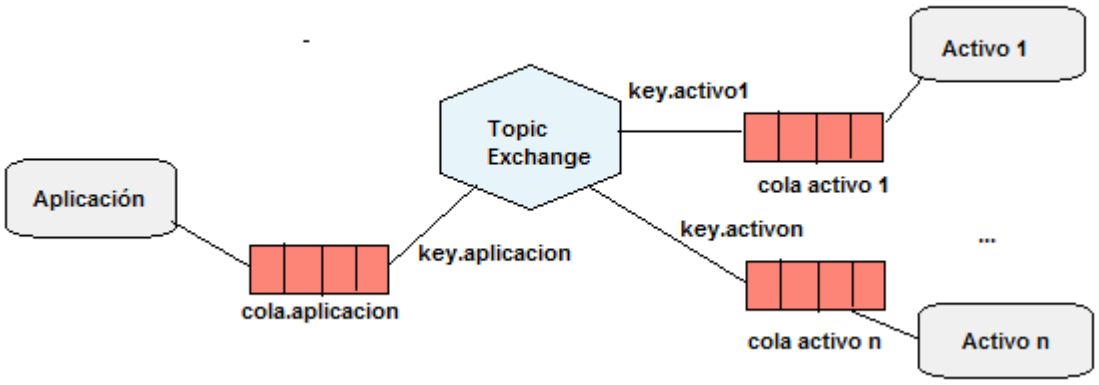

Figura 4.27. Infraestructura de mensajería para el sistema de riesgos. Elaboración propia.

Por lo tanto, el proceso de ejecución automática de controles consta de los siguientes pasos:

1. El usuario registra, para una definición de control (control + riesgo), una tarea a ejecutarse en un período o intervalo determinados.
2. Cuando el evento de la tarea se activa, el sistema genera un mensaje con la acción a realizar y lo envía al Exchange de RabbitMQ.
3. El activo remoto recibe el mensaje de la cola, ejecuta el comando recibido y envía un mensaje de respuesta al sistema.
4. El sistema recibe (de forma asincrónica) esta respuesta y verifica si la acción el control se ejecutó correctamente.
    a. En caso afirmativo, genera una ejecución de control, reduce el riesgo y actualiza la información en la base de datos.

4.4.5.5. Vencimiento de controles

El siguiente requerimiento abordado fue el vencimiento de los controles sobre los riesgos, es decir, de las definiciones de control. La historia de usuario correspondiente es la siguiente:

- **ID**: HU-DE002.
- **Nombre**: Vencimiento de control sobre riesgo.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 5 puntos de historia.
- **Criterio de aceptación**: el sistema actualizó el valor del riesgo al vencerse el control aplicado.

Para cumplir con este requisito, resultó necesario definir en el servidor un mecanismo de verificación constante sobre las definiciones de control vencidas. Esto fue fácilmente implementado utilizando la anotación `@Scheduled` de Spring, que permite definir tareas a ejecutarse en segundo plano en intervalos de tiempo. Para ello, el método no debe recibir parámetros ni retornar valor alguno. Para este caso de uso, se definió un método en `ControlController`, que verifica cada 24 horas la existencia de definiciones de control vencidas y actualiza la información en consecuencia. La Figura 4.28 muestra el código de este método. El mismo verifica la existencia de definiciones vencidas y ejecutadas, y en caso de hallar alguna, aplica la siguiente lógica:

1. Cambia el estado de la definición a “pendiente”.
2. Verifica si el riesgo correspondiente a esa definición está afectado por la aplicación de otros controles. En caso afirmativo, simplemente aumenta la criticidad del riesgo de acuerdo al porcentaje de reducción de la definición, y actualiza los valores de ocurrencia e impacto. En caso de que no se hallen controles vigentes sobre el riesgo, este restaura sus valores originales (utilizando el patrón Memento mencionado anteriormente).

```
@Transactional
@Scheduled(fixedDelay=1000*3600*24) //cada 24 horas verifica el vencimiento (86400000 ms)
public void verificarVencimientoControl()
{
    List<DefinicionControl> definicionesVencidas = this.controlService.traerDefinicionesVencidas();
    for(DefinicionControl definicion:definicionesVencidas)
    {
        definicion.actualizarEstado();
        this.controlService.actualizarDefinicionControl(definicion);
        Riesgo riesgo = definicion.getRiesgo();
        if(this.controlService.hayControlAplicado(riesgo))
        {
            riesgo.aumentarCriticidad(definicion.getPorcentajeReduccion());
            this.riesgoService.actualizarOcurrenciaImpacto(riesgo);
        }
        else
        {
            this.riesgoService.restaurarRiesgo(riesgo);
        }
    }
}
```

Figura 4.28. Método encargado de la verificación del vencimiento de controles. Elaboración propia.

4.4.5.6. Historial de ejecuciones de controles

El último requerimiento planteado en este Sprint es la posibilidad de que el responsable de seguridad pueda acceder al histórico de las ejecuciones de controles registradas en el sistema. La historia de usuario resumida es la siguiente:

- **ID**: HU-RS008.
- **Nombre**: Historial de ejecución de controles.

- **Prioridad**: alta.
- **Esfuerzo estimado**: 5 puntos de historia.
- **Criterio de aceptación**: se indicó la realización de un control e inmediatamente se guardó la información de su ejecución. Posteriormente, se accedió a la ejecución realizada.

Para cumplimentar este requisito simplemente se definió la consulta de las ejecuciones de control en `ControlController`, y se implementó el componente `HistorialEjecucionControlesComponent` en Angular, que presenta en una tabla la información de estas ejecuciones, tal como lo muestra la Figura 4.29. Adicionalmente, el usuario puede aplicar filtros en la búsqueda y ordenarla por fecha.

Figura 4.29. Tabla con ejecuciones de control. Elaboración propia.

4.4.6. Sprint 3 – Misceláneas

En esta última etapa del desarrollo de la aplicación se consideraron aquellas historias de usuario de menor prioridad y relacionadas con mejoras en las características del sistema. Se definió un tablero en Trello para dar seguimiento a las tareas definidas para esta iteración.

4.4.6.1. Consulta de usuarios registrados

El responsable de seguridad definió el requerimiento de consultar en el sistema los usuarios que se encuentran registrados en el mismo. La historia de usuario que refleja esta demanda es la siguiente:

- **ID**: HU-RS009.
- **Nombre**: Consulta de usuarios registrados.
- **Prioridad**: alta.
- **Esfuerzo estimado**: 3 puntos de historia.

- **Criterio de aceptación**: una vez seleccionada la opción de consultar usuarios, el sistema mostró una tabla con todos los usuarios registrados.

Puesto que ya se definió la administración de usuarios anteriormente, para cumplir con este requerimiento sólo fue necesario otorgar, en el backend, el permiso para consultar usuarios al responsable de seguridad. Desde el frontend, se reutilizó el componente para consultar usuarios, pero con la omisión de las acciones sobre los mismos (modificar y borrar), puesto que el responsable de seguridad sólo tiene el permiso de consultar usuarios y no el de efectuar cualquier otra operación sobre estos. De esta manera, tanto el responsable de sistemas como el de seguridad pueden consultar usuarios, pero sólo el primero puede actualizar el rol o dar de baja a estos, por ello ambos requisitos quedan cumplidos.

4.4.6.2. Cálculo automático de la criticidad del riesgo

El responsable de seguridad definió el requerimiento de que el sistema sea capaz de determinar la criticidad de un riesgo, a fin de simplificar al usuario el cálculo de esta. A continuación, se detalla la historia de usuario correspondiente.

- **ID**: HU-RS003.
- **Nombre**: Cálculo de la criticidad del riesgo.
- **Prioridad**: media
- **Esfuerzo estimado**: 1 puntos de historia.
- **Criterio de aceptación**: al registrar un riesgo o actualizar el mismo, el sistema calculó su criticidad en función del impacto y ocurrencia.

Como ya se mencionó anteriormente, la criticidad de un riesgo resulta del producto entre la probabilidad de que esta eventualidad ocurra y el impacto de la misma sobre el activo informático o la organización. Por consiguiente, sólo fue necesario realizar este cálculo en un método dentro del modelo `Riesgo`, y así cumplir con este requerimiento.

4.4.6.3. ABMC de políticas

Aunque esta aplicación se orienta a la administración de riesgos en activos informáticos y los controles aplicados a estos, es importante que estas medidas de seguridad adoptadas satisfagan las normas y políticas que la organización crea necesarias. De esta manera, se hace efectivo el principio de Cumplimiento de la tríada GRC (Gobernanza, Riesgo y Cumplimiento) descrita en el capítulo 2. En virtud de lo señalado, se definió como requerimiento la posibilidad de cargar políticas y normas organizacionales al sistema, y así

poder relacionarlas con los controles de seguridad implementados. El resumen de esta historia de usuario es el siguiente:

- **ID**: HU-RS006.
- **Nombre**: ABMC de políticas.
- **Prioridad**: media.
- **Esfuerzo estimado**: 3 puntos de historia.
- **Criterio de aceptación**: el sistema registró, modificó, y dio de baja correctamente una política.
- **Observaciones**: la política debe tener un nombre único, una descripción y el año en que se redactó.

El primer paso realizado fue definir la clase `Politica` en el modelo de negocio. La misma tiene una relación de muchos a muchos con la clase `Control`, por ello se agregó a esta última la propiedad `politicasRelacionadas`, que representa tal asociación. En las siguientes capas se definieron las clases `PoliticaDAO`, `PoliticaService` y `PoliticaController` para la persistencia, servicio y API respectivamente. El uso de criterios de búsqueda (Specifications) en la capa de persistencia permite filtrar políticas de acuerdo a lo ingresado por el usuario. Además, se implementó paginación y ordenamiento por el año de redacción. Para habilitar la asociación entre los controles y las políticas, se añadió en el formulario de registro y modificación del control una lista desplegable con las políticas disponibles que el usuario puede seleccionar. En la Figura 4.30 se presenta el formulario de registro de un control con la lista de políticas a relacionar.

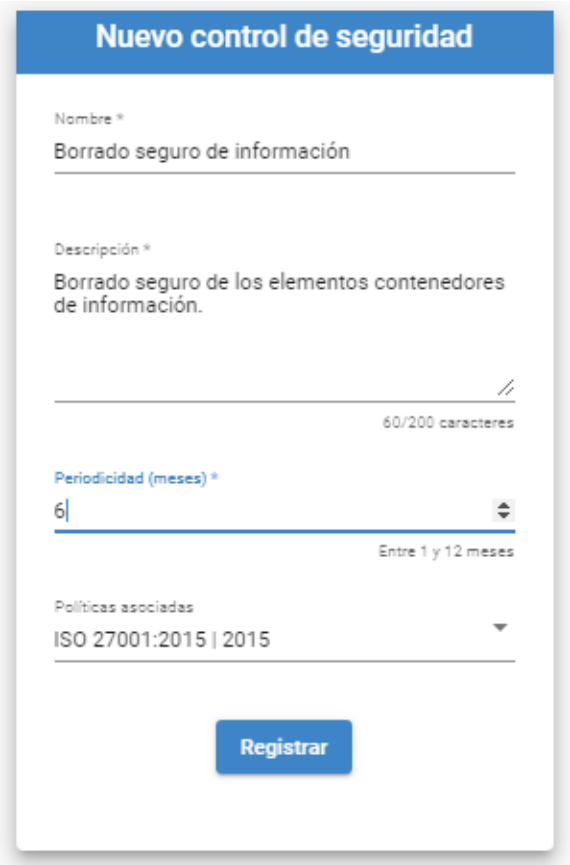

Figura 4.30. Formulario de registro de control. Elaboración propia.

4.4.6.4. Matriz dinámica de riesgos

En esta historia de usuario se solicitó el rearmado automático de la matriz al momento de registrar, actualizar o dar de baja un activo o un riesgo. El resumen de esta historia es el siguiente:

- **ID**: HU-DE003.
- **Nombre**: Matriz dinámica de riesgos.
- **Prioridad**: media.
- **Esfuerzo estimado**: 8 puntos de historia.
- **Criterio de aceptación**: se modificó el valor de un riesgo y la matriz se actualizó automáticamente.

Con el desarrollo de la matriz de riesgos definido en la primera etapa y la ejecución de controles sobre los riesgos en la segunda, este requerimiento ya fue implementado.

4.4.6.5. Diseño responsivo

Este último requerimiento corresponde a la capacidad del sistema de adaptar la visualización de su contenido en función del dispositivo en el que se accede. El resumen de la historia de usuario es el siguiente:

- **ID**: HU-US002.
- **Nombre**: Diseño adaptable a cualquier dispositivo.
- **Prioridad**: baja.
- **Esfuerzo estimado**: 1 punto de historia.
- **Criterio de aceptación**: se redimensionó la pantalla y el contenido se ajustó a esta.

En la actualidad, los usuarios pueden comunicarse con los servicios web desde cualquier medio tecnológico con conexión a internet y por ello, el sistema debe ser capaz de brindar una buena experiencia independientemente del dispositivo con el que se acceda. Para ello es necesario que el contenido sea lo suficientemente flexible como para ser presentado adecuadamente en una computadora, smartphone o tablet. Esta serie de técnicas que permiten a la página web adaptarse al medio utilizado por el usuario para conectarse a la misma es lo que se conoce como Diseño Web Adaptable o Responsive Design (Díaz Bustamante, 2011). La técnica o concepto principal en esta filosofía es el uso de valores fluidos de ancho y alto, es decir, definir el tamaño de un objeto basado en las proporciones (generalmente en porcentajes) que este ocupará dentro del elemento que lo contiene (padre), en lugar de valores fijos. De esta manera, el objeto nunca excederá el tamaño de su elemento padre y si este último cambia, el primero también. En consecuencia, el contenido se ajusta a la pantalla aun cuando esta varíe. Sin embargo, en dispositivos con resoluciones de pantalla pequeñas el contenido puede no ser apreciado correctamente, por lo que en lugar de redimensionar los elementos estos podrían reubicarse de acuerdo al tamaño de la pantalla. Para ello se utiliza el concepto de Consulta de Medios (Media Queries) que inspeccionan el medio utilizado (generalmente la pantalla del dispositivo) y el ancho actual de este, lo que permite reubicar el contenido en función de estas consultas. En conclusión, el diseño responsivo permite mantener la apropiada interacción del usuario con el sistema independientemente del dispositivo utilizado por el primero.

Para definir la estructura adaptable de la vista de la aplicación, se utilizó la propiedad de CSS3 conocida como Caja Flexible o Flexbox. La misma es una tecnología diseñada como

modelo unidimensional que permite distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación (Mozilla, 2019). Una vista está definida por componentes dispuestos lógicamente, y estos pueden contener otros elementos (hijos). En este principio, un elemento contenedor es considerado flexible, y por ello sus elementos hijos pueden tomar cualquier dirección y adaptar sus dimensiones al espacio visible dentro del elemento padre. Por consiguiente, los elementos hijos se denominan ítems flexibles, mientras que el contenedor es considerado como una caja de ítems flexibles (flexbox). La disposición de los elementos flexibles puede ser horizontal o vertical (fila o columna), y es la misma la que define los ejes del contenedor. Si la dirección es por fila, entonces el eje principal es el horizontal y el cruzado el vertical. En la dirección por columna, el eje vertical es el principal mientras que el horizontal es el cruzado. Adicionalmente, los elementos flexibles pueden aumentar o reducir su tamaño en función del espacio libre dentro de su contenedor. De esta manera, la propiedad de cajas flexibles favorece el diseño de aplicaciones adaptables a diferentes dispositivos. La tecnología de Flexbox fue utilizada en la aplicación mediante la librería FlexLayout para Angular, que permite especificar elementos flexibles haciendo uso de directivas en la definición de los mismos (etiquetas HTML). La Figura 4.31 muestra la vista de actualización del perfil del usuario representada en una pantalla de computadora, así como también en la de un Smartphone con una resolución de 480x800 pixeles. Los elementos de esta página fueron divididos en tres contenedores: la imagen de perfil, el formulario con la información del usuario y los botones de confirmar/cancelar la operación. Adicionalmente, los primeros dos bloques fueron incluidos dentro de un contenedor flexible. Estos bloques fueron alineados en fila para pantallas grandes y medianas, y en columna en pantallas de pequeña resolución, intercambiándose esta orientación mediante Media Queries, mencionadas anteriormente. Ambos contenedores fueron definidos con un porcentaje de anchura del 50% del contenedor padre, y ello resulta en una adaptación de los elementos para diferentes pantallas, como se muestra en la figura.

Figura 4.31. Edición de perfil adaptada a pantalla grande y pequeña. Elaboración propia.

Merece una consideración especial la adaptabilidad de las tablas existentes en la aplicación, puesto que deben conservar su significado y no siempre su estructura es compatible con diferentes tamaños de pantalla. Ejemplo de tal situación es una tabla con varias columnas, que al redimensionarse puede presentar dos comportamientos: reduce su tamaño en la misma proporción que la pantalla a fin de permanecer completamente visible, o mantiene sus dimensiones mostrándose incompleta pero permite al usuario desplazarse horizontal y/o verticalmente para ver su contenido. El inconveniente con el primero es que a menor resolución de la pantalla, menor será el tamaño del contenido, perdiéndose la legibilidad del mismo. El segundo comportamiento mantiene la legibilidad del contenido, pero puede perder la referencia de la información analizada en cada fila (principalmente en tablas con numerosas columnas) puesto que los datos más relevantes suelen presentarse en las primeras columnas. En consecuencia, existen alternativas que permiten adaptar o reorganizar el contenido de la tabla sin perder su significado y manteniendo la experiencia de usuario en diferentes dispositivos. La solución implementada en la aplicación consistió en redimensionar levemente el contenido mientras este sea legible (mediante Media Queries) y, en pantallas medianas/pequeñas, permitir el desplazamiento pero manteniendo la primera columna y la cabecera estáticas. De esta manera, al desplazarse horizontalmente, el usuario

mantiene la referencia de lo que está leyendo en esa fila (la primera columna muestra la información más relevante) e incluso facilita la lectura de la misma. La cabecera estática permite al usuario desplazarse verticalmente sin perder el significado de las columnas.

Adicionalmente, se implementó la característica de modo oscuro, que permite al usuario hacer uso del sistema sin forzar la vista. La Figura 4.32 presenta un ejemplo de esta alternativa.

Figura 4.32. Ejemplo de modo oscuro. Elaboración propia.

Con la implementación de esta historia de usuario, se completó la etapa de desarrollo del sistema.

4.4.7. Contenerización del sistema

Una vez desarrollada una aplicación, esta puede ser empaquetada (aislada) en un contenedor de software, lo que permite una mejor portabilidad y despliegue de la misma. El concepto de contenedor brinda una mejora a la virtualización, que tiene como objetivo abstraerse del hardware del equipo. Mientras que una máquina virtual recrea un sistema operativo completo consumiendo varios recursos del equipo anfitrión, un contenedor hace uso del sistema operativo existente. En cambio, cada contenedor dispone únicamente de las dependencias necesarias para la aplicación que va a ejecutarse en él, sin compartirlas con

otros contenedores. Por ello, estas unidades autocontenidas e independientes son más rápidas y eficientes que las máquinas virtuales. Al permitir la modularización del sistema (cada servicio del mismo puede ser aislado en un contenedor propio), este enfoque favorece a su escalabilidad y mantenimiento (se puede modificar un servicio sin interrumpir el funcionamiento de los demás). En resumidas palabras, los contenedores aíslan el software de su entorno y aseguran su funcionamiento uniforme sin importar diferencias como, por ejemplo, el desarrollo y producción (Muñoz y Linares, 2020, p.3).

La presente aplicación fue encapsulada utilizando la herramienta Docker. Esta es una plataforma de código abierto destinada a la gestión de contenedores de software y que permite abstraer los mismos de varios sistemas operativos (Docker, 2020). Los elementos de Docker necesarios para su utilización son:

- **Imagen**: es una captura del estado del contenedor, con el servicio y sus dependencias. Actúa como una plantilla sobre la que se crean los contenedores. Una imagen nunca cambia, sino que puede ser extendida a través de archivos "Dockerfile". Uno de los repositorios más utilizados es Docker Hub, que contiene imágenes para diferentes servicios (lenguajes de programación, bases de datos, etc.).
- **Contenedor**: es una instancia de la imagen en ejecución y, como tal, puede ser pausado, restaurado o borrado.
- **Volumen**: puesto que un contenedor es dinámico, la información persistente que este utiliza puede perderse cuando es destruido. Por ello, es mejor utilizar un volumen, que es un área del sistema destinada a almacenar información. Este es independiente del ciclo de vida del contenedor, por lo que la información persiste aún cuando este último es destruido.
- **Red**: permite comunicar los servicios contenidos. Para ello, cada contenedor expone un puerto específico, que se traduce a un puerto en el equipo anfitrión (host). Este último puerto es utilizado por los demás contenedores para acceder al servicio expuesto. Para favorecer la independencia del equipo host, Docker utiliza el nombre del servicio como dominio para acceder al mismo (ej.: tesis-backend:8080).

En primer lugar, se definieron contenedores para la aplicación backend, frontend, la base de datos MySQL y el servicio de mensajería RabbitMQ. De esta manera, y tal como lo muestra

la Figura 4.33, se organizó el sistema para su contenerización y posterior despliegue. El servicio de backend expone el puerto 8080 en su contenedor, que se traduce al 9010 en el host. Este servicio accede a la base de datos MySQL, que gestiona la información utilizando el volumen `riesgos-data`. El backend consume además el servicio de RabbitMQ, al que se accede por el puerto 5672. Por último, se embebió el servicio de frontend en un servidor web Nginx, que expone el puerto 80 y que se corresponde con el 9001 del equipo host.

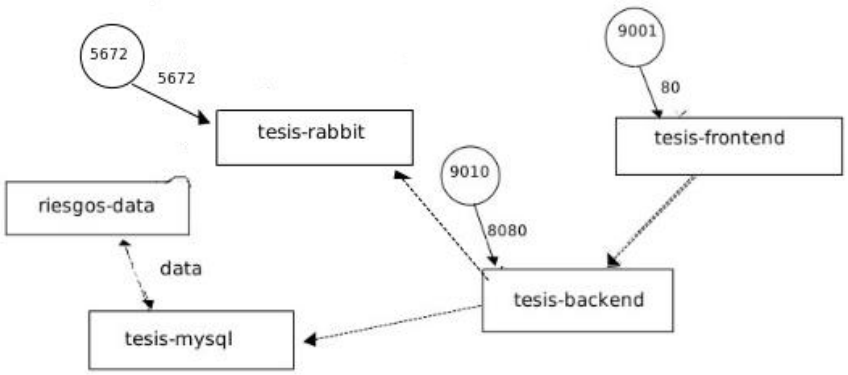

Figura 4.33. Estructura del sistema mediante contenedores. Elaboración propia.

4.4.7.1. Contenerización del backend

Inicialmente, la aplicación del backend requiere del JDK[17] para su funcionamiento. Por consiguiente, se utilizó como base la imagen de OpenJDK 11, la versión libre del JDK de Oracle. Se seleccionó la versión ligera `openjdk:11-slim`, que provee sólo la funcionalidad necesaria para ejecutar la aplicación. Puesto que esta última corresponde a un proyecto Maven, es necesario empaquetarla en este contexto y generar así el respectivo ejecutable. Finalmente, la imagen resultante se refleja en el siguiente Dockerfile, tal como se muestra en la Figura 4.34:

[17] JDK Java Development Kit o Kit de Desarrollo de Java

```
FROM openjdk:11-slim AS builder
ENV APP_HOME=/usr/app/
WORKDIR $APP_HOME
COPY . .
RUN chmod +x mvnw && ./mvnw package -Dmaven.test.skip=true

FROM openjdk:11-slim AS docker
LABEL maintainer="rlpena@unrn.edu.ar"
LABEL "unrn.edu.ar"="UNRN"
LABEL "service"="API Rest"
RUN apt update && apt install -y curl jq
COPY --from=builder /usr/app/target/*.jar app.jar
ENTRYPOINT ["java","-jar","app.jar"]
EXPOSE 8080
```

Figura 4.34. Dockerfile de la imagen del backend. Elaboración propia.

En este archivo se definen dos etapas, una para el empaquetado de la aplicación y otra para su ejecución. En la primera etapa, se declara la variable de entorno `APP_HOME` como `/usr/app` y se establece esta ruta como directorio de trabajo. Se copia entonces el contexto al contenedor, y luego se genera el `.jar` de la aplicación Maven. Para esto último, se otorga el permiso de ejecución del comando, y se excluyen las pruebas puesto que no son necesarias para producción. En la segunda etapa, se copia el ejecutable generado al contenedor como `app.jar`. Por último, se expone el puerto 8080 para el backend. A partir de este archivo, se generó la imagen usando el comando "`docker build -t riesgos-backend:1.0 .`" (sin comillas). En consecuencia, la imagen generada se denomina `riesgos-backend` y su versión es 1.0. Es una buena práctica denominar la imagen utilizando el formato `{nombre}:{versión}`. Este comando crea una imagen utilizando por defecto un archivo con el nombre `Dockerfile` en la ruta actual, y puesto que el archivo definido se denomina así, no fue necesario especificarlo. En caso de tener otro nombre de archivo, este debe indicarse mediante el parámetro `-f` del comando.

Para el contenedor de persistencia, se seleccionó la imagen de MySQL 8, puesto que es la versión utilizada por la base de datos de la aplicación. Utilizando esta imagen como base, no resultó necesario extender su funcionalidad. No obstante, se requirió copiar un archivo de configuración personalizado. En la primera ejecución del contenedor (denominado `tesis-mysql`), mediante el comando `docker exec -i tesis-mysql mysql`

`-uroot -p bd_riesgos < bd_tesis_creacion.sql` se ejecutó el script de creación de la base de datos. Este comando de Docker permite ejecutar una acción dentro de un contenedor existente, en este caso MySQL. La base de datos se persistió en el volumen, por lo tanto, tal información no se perderá cuando el contenedor termine su ejecución.

Para el servicio de RabbitMQ se utilizó la imagen de la versión 3 “alpina” del mismo, es decir, una versión ligera con la funcionalidad indispensable para el funcionamiento del servicio. A este contenedor hubo que agregarle un archivo de configuración personalizado, así como también uno de carga de datos puesto que durante el desarrollo del sistema se utilizó un servidor Rabbit existente, siendo necesaria una migración al nuevo servicio. La imagen utilizada también brinda una interfaz gráfica de administrador, facilitando el mantenimiento del servicio.

Una forma sencilla de iniciar múltiples contenedores es mediante la herramienta Docker Compose, que permite definir e iniciar automáticamente varios servicios y sus dependencias, definidas en un único archivo en el lenguaje YAML. La Figura 4.35 muestra el Compose para la contenerización de los servicios del backend descritos.

Figura 4.35. Compose de los servicios de backend. Elaboración propia.

En primer lugar, debe indicarse la versión de Compose que se está utilizando, puesto que las configuraciones pueden variar entre versiones diferentes. A continuación, bajo la propiedad `services`, se declaran los diferentes contenedores a crear. Cada servicio comienza con la definición de su nombre, y luego sus propiedades, tales como:

- Image: la imagen sobre la que se creará el contenedor.
- Ports: los puertos para acceder al servicio. Se indica primero el puerto del host, y luego el equivalente en el contenedor.
- Healthcheck: la prueba a aplicar sobre el servicio para evaluar su correcto inicio.
- Environment: variables de entorno utilizadas por el servicio (ej.: URLs, claves, etc.)
- Volumes: el o los volúmenes que utilizará el servicio. En caso de definirse fuera del la declaración de los contenedores, ese volumen podrá ser accedido por todos los servicios que lo requieran.
- Network: la red que comunicará el servicio con los demás. Esta red debe declararse luego fuera del alcance de la definición de contenedores, tal como sucede en este ejemplo con la red `riesgos_net`.

Por último, con el comando `docker-compose -p lpena -f docker-compose-prod.yml up -d` se ejecutó el archivo indicado (`docker-compose-prod.yml`), iniciando los servicios allí declarados.

4.4.7.2. Contenerización del frontend

Para el despliegue de la aplicación del frontend desarrollada en Angular, se requiere de NodeJS, por lo que se eligió como base la imagen de la versión 12.8 del mismo. Además, es recomendable implementar la aplicación en un servidor web, en este caso se utilizó Nginx, puesto que es ligero y de alto rendimiento. Para generar la imagen del frontend, se creó un Dockerfile con una etapa para la instalación de Node y las dependencias requeridas por el proyecto, mientras que la siguiente etapa se encarga del despliegue de la aplicación en el servidor de Nginx. La Figura 4.36 muestra el archivo resultante:

```
FROM node:12.8.1 as build-stage
WORKDIR /app
COPY *.json ./
RUN npm install
COPY . .
RUN node_modules/.bin/ng build --prod

FROM nginx:1.17.9-alpine as production-stage
LABEL maintainer="rlpena@unrn.edu.ar"
LABEL "unrn.edu.ar"="UNRN"
LABEL "service"="Angular Frontend Tesis Riesgos"
ENV API_HOST localhost
ENV API_PORT 8080
RUN rm -rf /usr/share/nginx/html/*
COPY frontend.conf /etc/nginx/nginx.conf
COPY --from=build-stage /app/dist/riesgosF /usr/share/nginx/html
VOLUME /home/app
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Figura 4.36. Dockerfile de la imagen del frontend. Elaboración propia.

En la primera etapa, se crea un directorio de trabajo en `/app`, y se corre la instalación del Node Package Manager (NPM) junto con las dependencias del proyecto. Luego de copiar los archivos al contenedor, se ejecuta el comando `ng build` con el entorno de producción. Este comando genera una carpeta denominada `dist` que es más liviana para desplegar que utilizar la carpeta de `node_modules`. En la siguiente etapa, se descarga la imagen ligera de

Nginx, y se declaran variables de entorno. Luego, se elimina el archivo `index.html` que viene por defecto en Nginx, y se copia un archivo de configuración de despliegue llamado `frontend.conf` al contenedor. A continuación, copia la carpeta generada con la aplicación lista para desplegar en Nginx. Por último, se crea un volumen en `/home/app` y se expone el puerto 80 y se define el comando para mantener la ejecución del servidor en primer plano. Al igual que con el backend, se generó la imagen mediante `docker build`, denominándose esta `riesgos-frontend` en su versión 1.0. En la Figura 4.37 se detalla el Docker Compose del servicio de frontend.

```yaml
version: '3.3'
services:
    tesis-frontend:
        image: riesgos-frontend:1.0
        container_name: tesis-frontend
        ports:
            - '9001:80'
        networks:
            - riesgos_net
        healthcheck:
            test: wget -O /dev/null http://curso-frontend || exit 1
            timeout: 10s
            interval: 15s
            retries: 5
networks:
    riesgos_net:
        external:
            name: riesgos_net
```

Figura 4.37. Docker Compose del frontend. Elaboración propia.

De igual manera que con el servicio de backend, se inició el servicio mediante el comando `docker-compose up`, iniciando la aplicación frontend y, en consecuencia, permitiendo el acceso al sistema. Por consiguiente, el software fue empaquetado en contenedores, facilitando su portabilidad y posterior despliegue en un servidor.

5. Verificación y validación

En el capítulo anterior se describió el desarrollo del sistema y con ello las pruebas unitarias e integración pertinentes, que permitieron cumplir con los requerimientos funcionales señalados en las historias de usuario, así como la calidad del código fuente. En

este apartado se detallan las evaluaciones del sistema en su conjunto, completando así las actividades de verificación y validación señaladas en el plan de pruebas.

5.1. Entorno de pruebas

El sistema empaquetado en contenedores Docker fue desplegado en un servidor de la UNRN. Este empaquetado o contenerización de la aplicación con sus dependencias permite la portabilidad de la misma, a la vez que facilita su mantenimiento, debido a su enfoque modular. El entorno de pruebas consta de las siguientes características:

- Servidor con dominio http://pruebasudocu.unrn.edu.ar
- Acceso al frontend en Angular a través del puerto 9001.
- Acceso al backend de Spring Boot por el puerto 9010
- Servicio de RabbitMQ por el puerto 5672
- Especificaciones del servidor:
    - Sistema operativo Linux CentOS 7 – 64 bits
    - CPU Intel Xeon E-31235 - 4 núcleos, 3.20 GHz
    - 16 GB de memoria RAM
    - Python v. 2.7.5

La ejecución de los casos de prueba fue realizada en una notebook con sistema operativo Linux Mint 19.3 Tricia, utilizando el navegador Mozilla Firefox 76.

5.2. Casos de prueba

En esta primera versión, el sistema es capaz de administrar activos informáticos, sus riesgos inherentes, políticas organizacionales y controles de seguridad. Además, estos últimos pueden ser ejecutados manual o automáticamente. Como herramienta de visualización, se utiliza la matriz de probabilidad e impacto. Partiendo de estas características, y considerando las actividades que se realizan con mayor frecuencia, se implementaron los casos de prueba que a continuación se describen.

5.2.1. Registro de activos

Una de las primeras tareas que el usuario debe realizar es agregar un nuevo activo al sistema. En este caso, se registró un servidor llamado Proxmox 03. La Figura 5.1 muestra el formulario de ingreso del nuevo activo (izquierda) y su consecuente registro en el sistema (derecha).

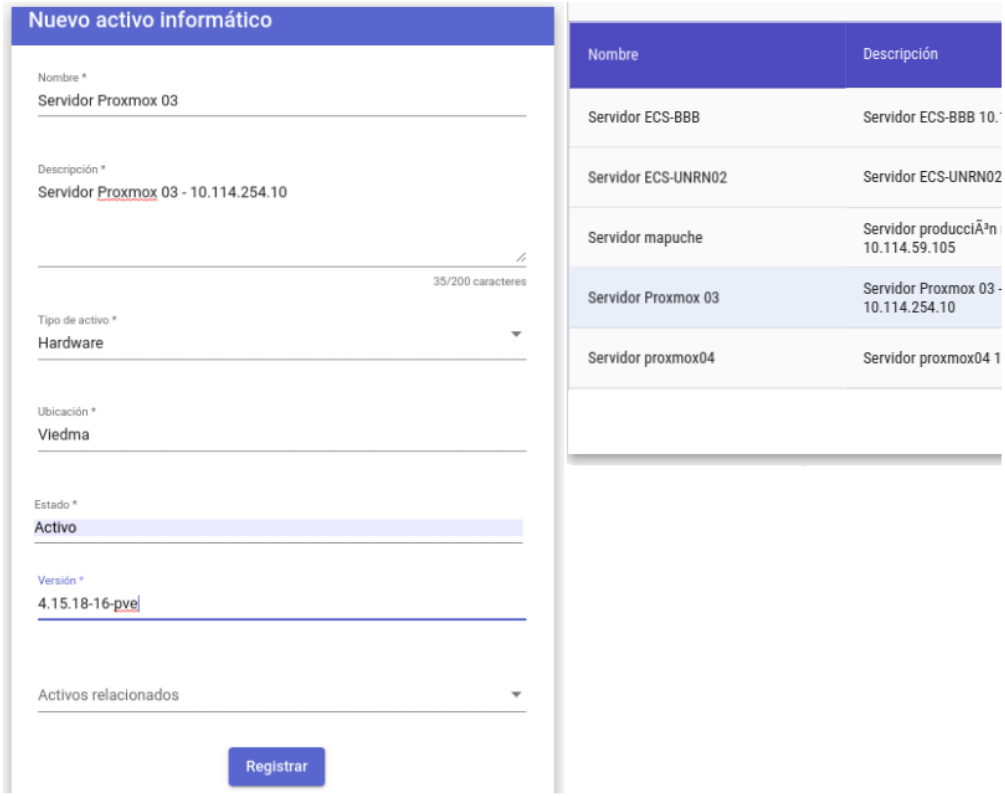

Figura 5.1. Registro exitoso de un activo de prueba en el sistema. Elaboración propia.

5.2.2. Definición de riesgos

Otra actividad frecuente es la de registrar un nuevo riesgo en un activo. En este caso se añadió al sistema un riesgo de interrupción en el servidor que aloja un Proxmox, por corte de luz. En la Figura 5.2 se puede apreciar el formulario de registro, así como la tabla que muestra el nuevo riesgo persistido en el sistema.

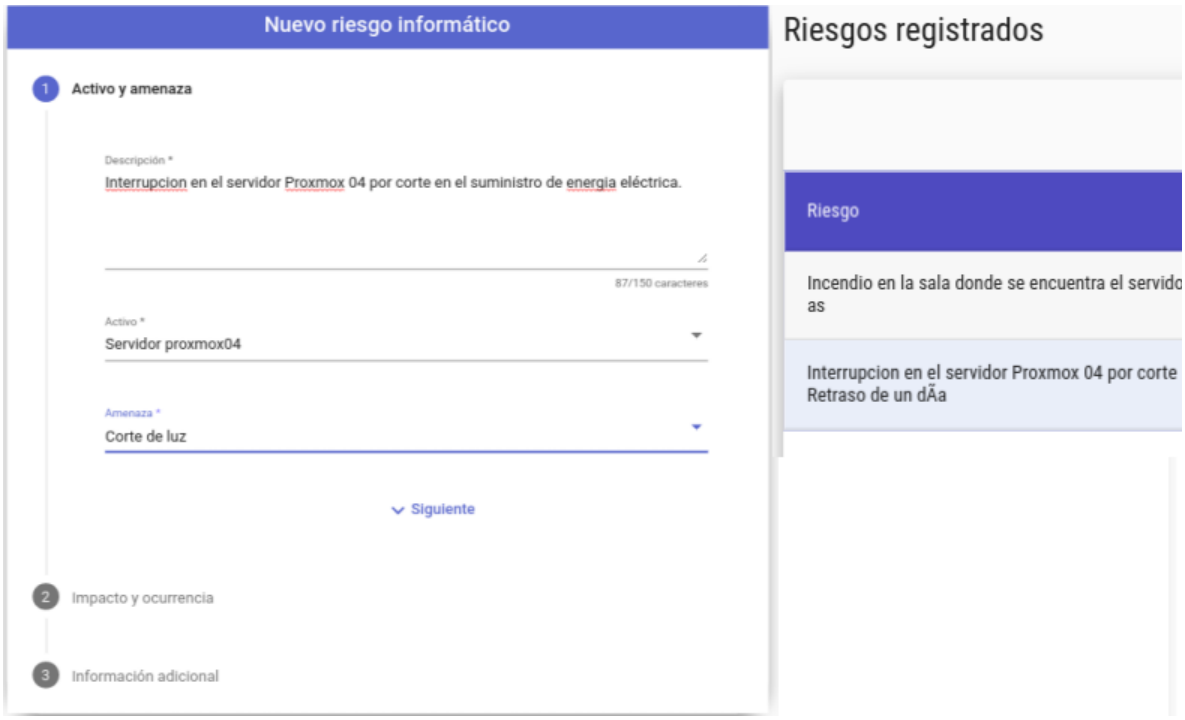

Figura 5.2. Formulario y registro exitoso de un riesgo. Elaboración propia.

5.2.3. Aplicación de controles

El objetivo de este caso de prueba es demostrar la capacidad del sistema de asociar controles y ejecutarlos incluso de forma automatizada. Para ello, se definió un riesgo de pérdida de información de la base de datos de esta aplicación, con un 45% de criticidad. A esta amenaza se le asignó un control de resguardo o backup, que reduce en un 80% la criticidad de la eventualidad. En la Figura 5.3 se puede observar esta relación control-riesgo y la opción de programar una tarea de ejecución automática.

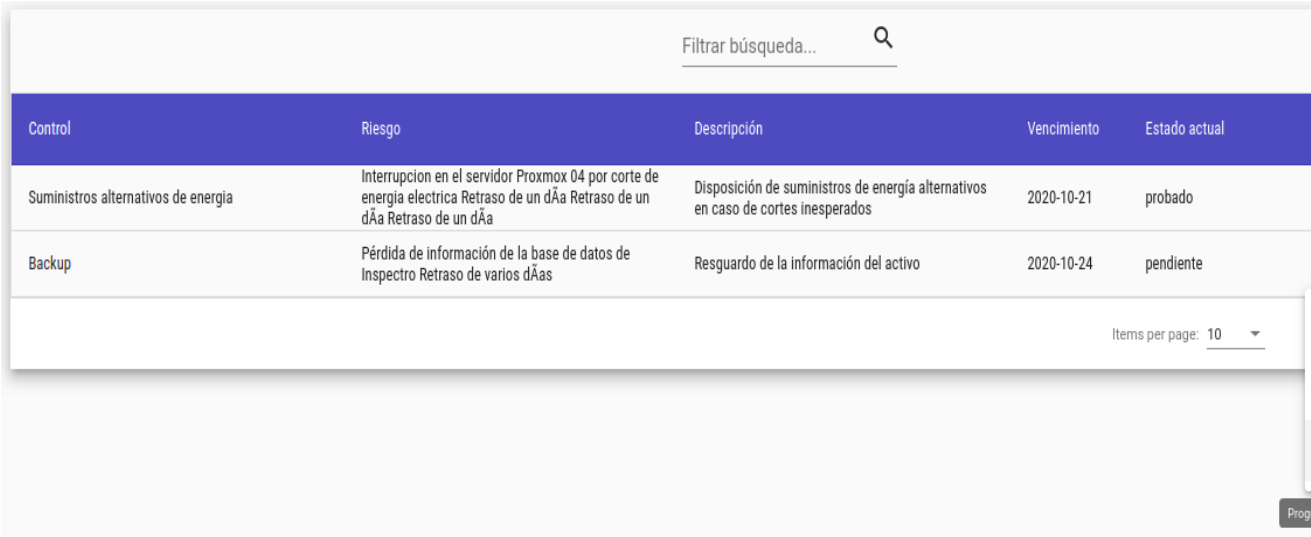

Figura 5.3. Tabla con controles asociados a riesgos. Elaboración propia.

Tal como detalla la Figura 5.4, se programó una tarea de ejecución del control de resguardo mencionado cada dos minutos, a fin de comprobar esta funcionalidad inmediatamente. No obstante, y como ya se ha aclarado, es posible definir tareas para diferentes períodos de tiempo (diario, semanal, mensual, anual, etc.). Debido a que la base de datos está incluida en un contenedor, se utilizó el comando `exec` de Docker. De esta manera, se ejecuta el servicio `mysqldump` dentro del contenedor de MySQL, aplicando la copia de resguardo de la base de datos y preservando esta en un archivo llamado `riesgos_db_back.sql`. Cabe mencionar que además de modificar y borrar la tarea, el sistema brinda la posibilidad de pausar/reanudar la misma.

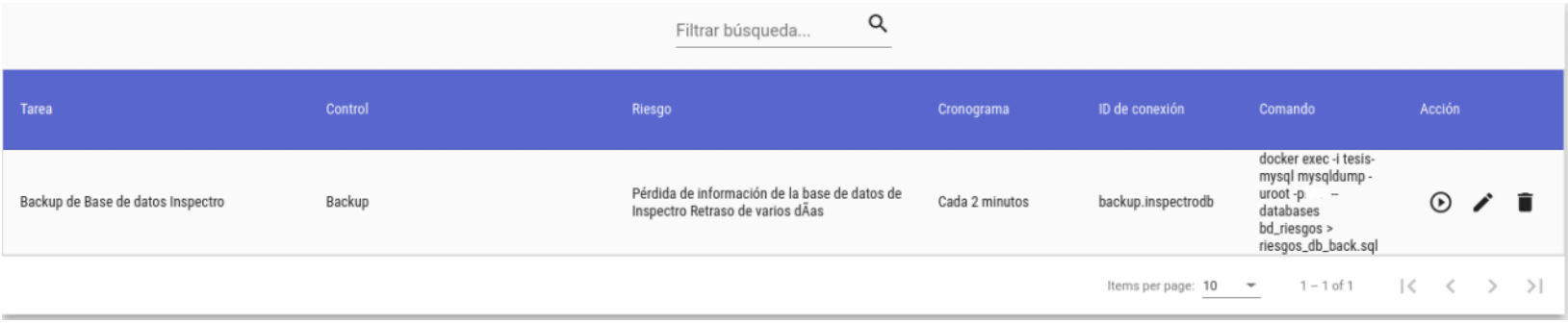

Figura 5.4. Tarea de ejecución de resguardo de base de datos. Elaboración propia.

Como se mencionó en el capítulo anterior, la ejecución automatizada y remota de controles es posible gracias al servicio de mensajería RabbitMQ, que permite la comunicación entre diferentes clientes y/o servicios mediante el uso de colas. Para este caso, el equipo destino (donde se ejecuta el control) hace uso de un script Python que aguarda de forma asíncrona la llegada de mensajes a una determinada cola. Al recibir un mensaje del sistema con el control, ejecuta el comando indicado y luego genera una respuesta al sistema. En caso de éxito, el sistema genera una nueva ejecución de control y la persiste. La Figura 5.5 muestra la ejecución exitosa en la consola de comandos, así como la generación del archivo de resguardo mencionado.

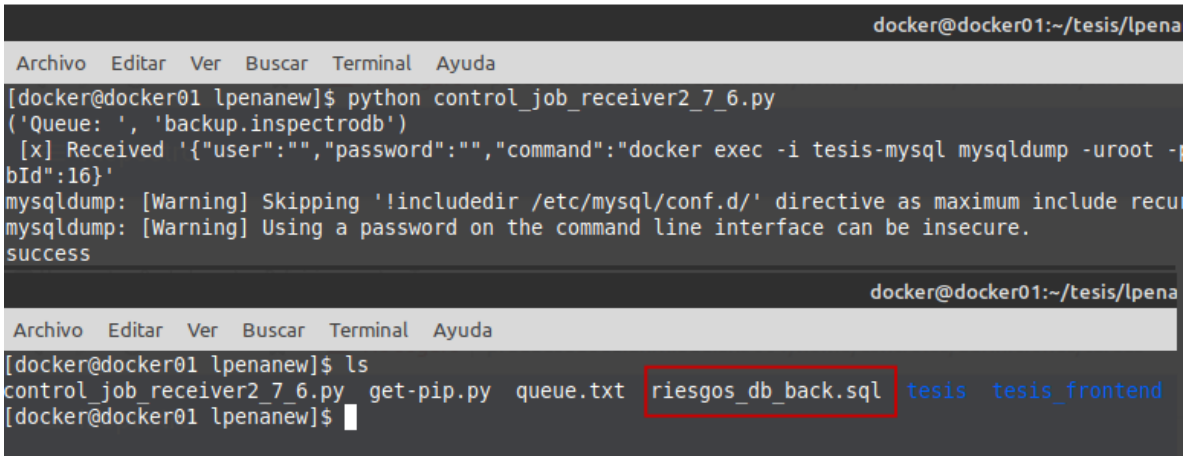

Figura 5.6. Consola con ejecución exitosa de control automático. Elaboración propia.

Una vez que un control es ejecutado sobre un determinado riesgo, este último reduce su criticidad de acuerdo al porcentaje especificado en el primero, y la matriz de probabilidad e impacto es actualizada automáticamente. En este caso, la criticidad del riesgo de pérdida de información (inicialmente un 45%) es reducida en un 80%, es decir, su criticidad actual es de 9%. En la Figura 5.7 se puede observar la matriz de probabilidad e impacto actualizada con los riesgos de acuerdo a su criticidad.

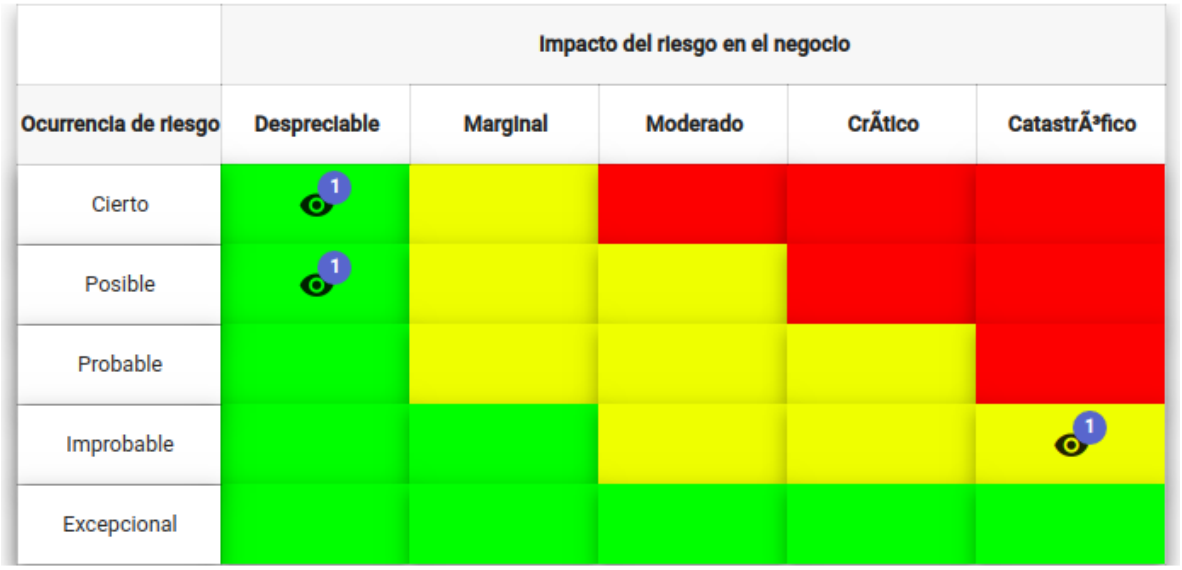

Figura 5.7. Matriz de probabilidad e impacto actualizada. Elaboración propia.

De esta manera, el sistema superó la prueba, demostrando ser capaz de ejecutar controles de seguridad para reducir riesgos y hacerlo además de forma automatizada.

5.3. Conclusiones de las pruebas

Mediante la ejecución de las pruebas descritas, se pudo verificar el correcto funcionamiento de los diferentes componentes del sistema, en un posible escenario de producción. El servicio del frontend ha logrado comunicarse con la API REST del backend, y este ha respondido correctamente a las peticiones del primero. De ello se deduce la comunicación exitosa con el servicio de persistencia. Además, la ejecución automática del control ha demostrado el funcionamiento del servicio de RabbitMQ, en el servidor como en el cliente (el agente de control Python). En consecuencia, se cumplió con la contenerización del sistema, encapsulando los servicios del mismo y logrando su funcionamiento en el servidor.

El sistema demostró ser capaz de afrontar las tareas implicadas en el flujo cotidiano de trabajo, haciéndolo de manera fácil y comprensible para el usuario. Como se verificó y validó, la aplicación cumple con las tareas más frecuentes en el proceso de gestión de riesgos, siendo capaz de registrar activos, asociar riesgos a los mismos, determinar y programar

controles de seguridad, y mantener una matriz de probabilidad e impacto para visualizar el estado actual de la organización. El menú dividido en módulos (activos, riesgos, controles, etc.), así como el breadcrumb (barra de navegación) facilitan al usuario su orientación por el sistema. La disposición de las acciones de alta, baja y actualización en las tablas, mediante íconos bien definidos, hace más cómoda la administración de la información. De esta manera, el sistema permite al usuario realizar las tareas de gestión de riesgos y hacerlo fácilmente.

La ejecución programada y automatizada de controles resultó una novedad para el usuario, puesto que le permite delegar responsabilidad en el sistema, flexibilizando así su trabajo. Pese a ser una característica experimental, demostró poder cumplir con esta flexibilización y tener potencial para mejorar aún más el flujo de trabajo. En esta primera versión, el cliente debe instalar el script Python y establecer el nombre de la cola de mensajes manualmente en un archivo llamado `queue.txt`, coincidiendo este con la clave de conexión definido en la tarea. Como futura mejora, el sistema permitirá la descarga automática del script al crearse la tarea, con el nombre de la cola ya incluido. No obstante, el sistema ya demostró su capacidad de automatizar controles, facilitando la labor al usuario.

En conclusión, la ejecución de las pruebas permitió demostrar el funcionamiento del sistema empaquetado en contenedores, facilitando así la portabilidad del mismo. Se evidenció también el cumplimiento del flujo de trabajo cotidiano de gestión de riesgos, y haciéndolo de manera fácil. Esta facilidad se vio incrementada por la posibilidad de definir controles automatizados y remotos, flexibilizando la labor. Por consiguiente, es posible afirmar que el software se encuentra preparado para ser utilizado por cualquier organización que lo requiera.

6. Conclusiones y líneas futuras

En este apartado se exponen las conclusiones que se desprenden del presente trabajo y el sistema desarrollado, así como también las líneas que quedan abiertas para futuras investigaciones que hagan a la mejora del producto.

6.1. Conclusiones

La presencia cada vez más significativa y ubicua de herramientas de TI dentro de la organización incrementa la relevancia de las mismas y, en consecuencia, tales activos deben ser protegidos a fin de garantizar la seguridad de la información que emplean. Es una certeza que la información es el recurso esencial de toda organización, puesto que de ella depende la toma de decisiones con respecto a la evaluación del desempeño, la optimización de los recursos y el cumplimiento de los objetivos definidos. Actualmente, la información fluye, se procesa y genera a través de herramientas tecnológicas, por ello es importante garantizar la seguridad de los activos informáticos gestionando los riesgos a los que estos están expuestos.

Las amenazas de ciberseguridad están a la orden del día y en constante crecimiento y sofisticación, comprometiendo la disponibilidad, confidencialidad e integridad de la información. No obstante, no todas las organizaciones han comprendido aún la magnitud de tal situación por lo que consideran que la gestión de riesgos no es suficiente para invertir en materia de seguridad. Ello suele deberse al costo económico y de tiempo que representa adoptar soluciones complejas, así como a la falta de capacitación sobre seguridad informática.

En virtud de lo señalado, se desarrolló el presente sistema con el objetivo de acercar el proceso de gestión de riesgos informáticos a las organizaciones de una forma sencilla de comprender, a la vez de dinámica y efectiva. La aplicación presenta una visión integral de la gestión de riesgos compuesta por la administración de amenazas, activos informáticos, riesgos, controles y políticas organizacionales. Adicionalmente, brinda la posibilidad de programar ejecuciones automatizadas de controles, lo que permite al usuario delegar la tarea a la herramienta y facilitar así esta labor. El diseño responsivo favorece a que el contenido de la aplicación pueda visualizarse correctamente en dispositivos de diferentes resoluciones de pantalla, incluyendo los móviles. Debido a su concepción mediante una metodología ágil como Scrum, el presente sistema fue desarrollado de manera flexible y de fácil adaptación a las necesidades cada vez más cambiantes del mercado. La utilización de esta herramienta orienta a la organización a lograr el principio de Gobernanza, Riesgo y Cumplimiento, puesto que los controles implementados pueden enmarcarse bajo normas y políticas, verificando así

el cumplimiento de las mismas. Tanto la gestión de riesgos como el cumplimiento de normas, permiten obtener conclusiones y tomar decisiones que faciliten la gobernanza. Esta visión integral permite entonces la generación de valor para la organización que la implementa.

6.2. Líneas futuras de investigación

Actualmente, el sistema presta sus servicios a través de la web y por ello sólo se ejecuta en navegadores. No obstante, el incremento en el uso de los dispositivos móviles brinda la posibilidad de extender el sistema a través del desarrollo de una app para estas plataformas. La arquitectura definida en este proyecto (separación de backend y frontend, y comunicación mediante la API REST) favorece la escalabilidad de la aplicación, por lo que no se vería afectada al añadir una extensión móvil. Este desarrollo mejoraría la experiencia de usuario. Las principales herramientas utilizadas en la actualidad para esta labor son el Android SDK y React Native. Ambas proporcionan un entorno de desarrollo para aplicaciones móviles, aunque el primero está orientado específicamente al sistema operativo Android, mientras que React Native permite desarrollar aplicaciones multiplataformas, y por ello su popularidad va en aumento.

Aún cuando la aplicación sea desplegada bajo demanda en los servidores (físicos o remotos), no se descarta su prestación como un servicio en la nube (un SaaS) próximamente. Ello se debe a la revolución que ha causado esta modalidad y su rápida adopción en las organizaciones debido a las ventajas que representa el tener la información almacenada remotamente, como la escalabilidad del negocio y el ahorro en mantenimiento de los servidores. Pese a que se han expuesto en este proyecto los problemas de la nube relacionados con la (in)seguridad de la información almacenada allí, resulta difícil obviar esta tendencia y es conveniente adoptarla más adelante. Cabe aclarar que actualmente la organización no necesita contar con un servidor en sus instalaciones para desplegar el sistema, sino que puede hacerlo utilizando una plataforma en la nube (Plataforma como Servicio o PaaS), asumiendo el riesgo de seguridad expuesto en este trabajo.

Se añadirá al sistema la implementación y uso de indicadores de desempeño de la gestión de riesgos, lo que proporcionará una visión aún más objetiva que justifique la decisión que se toma con respecto a la clasificación de los riesgos según su impacto y ocurrencia así como los controles aplicados. Los indicadores retroalimentarán al proceso con información relevante para el posterior análisis del esquema de gestión de riesgos. Mediante tales indicadores, el sistema será capaz de determinar el nivel del proceso de gestión de riesgos implementado y el nivel de riesgo actual en que se encuentra la organización. Esta información será presentada al área ejecutiva mediante un tablero de control, una herramienta que le permitirá visualizar la información relevante en conjunto y tomar decisiones en consecuencia.

7. Referencias

[1]. Alaimo, M. (2013). *Proyectos ágiles con Scrum*. Buenos Aires, Argentina: Ediciones Kleer.

[2]. Alemán, H., y Rodríguez Barrera, C. (2015). Metodologías Para el Análisis de Riesgos en los SGSI. *Publicaciones e Investigación, 9(*1*),* 73-86. Recuperado de https://doaj.org/article/6b3560fd044b48f79b382a3b00391aff

[3]. Aloka UK.(2017). *Assurance Risk Compliance*. Recuperado de https://arcrisk.com/

[4]. Álvarez, M. A. (11 de Abril de 2018). Autenticación por token. *Desarrollo Web*. Recuperado de https://desarrolloweb.com/articulos/autenticacion-token.html [Consultado 12 Ago., 2019].

[5]. Cervantes, H. (2010). Arquitectura de Software. *Software Gurú #27*. Recuperado de https://sg.com.mx/revista/27/arquitectura-software [Consultado 21 May., 2019]

[6]. Díaz Bustamante, J. (16 de Agosto de 2011). Introducción al Diseño Web Adaptable o Responsive Web Design. *Emenia*. Recuperado de https://www.emenia.es/diseno-web-adaptable-o-responsive-web-design/ [Consultado 22 Nov., 2019]

[7]. Docker Inc. (2020). *Get started with Docker*. Recuperado de https://www.docker.com/

[8]. Europa Press. Los ciberataques serán más inteligentes en 2019, según predice Check Point. (28 de Noviembre de 2018). *PortalTic*, *Europa Press*. Recuperado de

https://www.europapress.es/portaltic/ciberseguridad/noticia-ciberataques-seran-mas-inteligentes-2019-predice-check-point-20181128135227.html

[9]. Google. (2019). *Google Firebase*. Recuperado de https://www.docker.com/

[10]. Halterman, J. (2019). *Model Mapper*. Recuperado de http://modelmapper.org/

[11]. International Organization for Standardization. (2009). *ISO/IEC 31000:2009- Gestión del Riesgo: Procesos y guías, 1era ed.*. Suiza: ISO/IEC.

[12]. Isla Visual. (13 de Noviembre de 2012). Diferencias entre Scrum y XP [Gráfico]. Recuperado de http://www.islavisual.com/articulos/desarrollo_web/diferencias-entre-scrum-y-xp.php [Consultado 12 Mar., 2019]

[13]. Jaramillo, E. (2017). *Análisis de la seguridad de datos en la arquitectura de Software como Servicio (SaaS)* (Tesis de grado). Universidad Nacional de Loja, Loja, Ecuador. Recuperado de http://dspace.unl.edu.ec/jspui/handle/123456789/18941

[14]. Lindros, K. ¿Qué es GRC y por qué lo necesita?. (20 de Julio de 2017). *CIO España*. Recuperado de https://www.ciospain.es/gobierno-ti/que-es-grc-y-por-que-lo-necesita

[15]. Lugani, C.F. y Peña, R.L. (2018). Desarrollo de un esquema de Gestión de Riesgos para la Universidad Nacional de Río Negro. *Anales de SIE 2018, Simposio de Informática en el Estado, 47º Jornadas Argentinas de Informática e Investigación Operativa (JAIIO)*, 170-182. ISSN: 2451-7534. Recuperado de http://47jaiio.sadio.org.ar/sites/default/files/SIE-14.PDF [Consultado 05 Mar., 2019]

[16]. Mozilla. (2019). Conceptos básicos de flexbox. Recuperado de https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Conceptos_Basicos_de_Flexbox [Consultado 20 Nov., 2019]

[17]. Muñoz, H., Linares, O. (2020). *Introducción a Docker, Unidad I* [apuntes de curso].

[18]. Oficina Nacional de Tecnologías de Información. (2015). *Disposición 1/2015-Política Modelo de Seguridad de la Información* (Publicación nº 33.077). Recuperado de http://servicios.infoleg.gob.ar/infolegInternet/anexos/240000-244999/242859/norma.htm

[19].Omega. (2018). *Pims Risk Management*. Recuperado de https://www.omega.no/omega-services/software/risk-management

[20]. Padinger, G. Ciberataques a bancos latinoamericanos y el fantasma norcoreano: los afectados en 2018 y las amenazas para 2019. (22 de Diciembre de 2018). *Infobae*. Recuperado de https://www.infobae.com/america/tecno/2018/12/22/ciberataques-a-bancos-latinoamericanos-y-el-fantasma-norcoreano-los-afectados-en-2018-y-las-amenazas-para-2019/ [Consultado 01 Mar., 2019]

[21]. Papazafeiropoulou, A. y Spanaki, K. (2016). Understanding governance, risk and compliance information systems (GRC IS): The experts view. *Information Systems Frontiers*, *18*(6), 1251-1263. doi: https://doi.org/10.1007/s10796-015-9572-3

[22]. Pivotal Software. (2020). *RabbitMQ*. Recuperado de https://www.rabbitmq.com/

[23]. Pricewaterhouse Coopers. (2018). *Fortaleciendo la sociedad digital contra los ataques cibernéticos. Resultados de la Encuesta Global de Seguridad de la Información 2018*. Recuperado de https://www.pwc.com.ar/es/publicaciones/assets/encuesta-global-seguridad-informacion-primer-reporte-2018.pdf

[24]. Schwaber, K. y Sutherland, J. (2017). The Scrum Guide, The Definitive Guide to Scrum: The Rules of the Game. Recuperado de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100 [Consultado 06 Mar., 2019]

[25]. Software AG. (2019). *Quartz, job scheduler*. Recuperado de http://www.quartz-scheduler.org/

[26]. Solarte, F.,N., Enriquez Rosero, E.,R., y Benavides Ruano, M.,C.(2015). Metodología de análisis y evaluación de riesgos aplicados a la seguridad informática y de información bajo la norma ISO/IEC 27001. *Revista Tecnológica ESPOL – RTE*, *28*(5), 492-507 Recuperado de http://www.rte.espol.edu.ec/index.php/tecnologica/article/view/456/321 [Consultado 01 Mar., 2019]

[27]. Stecky Efantis, M. (16 de Mayo de 2016). 5 Easy Steps to Understanding JSON Web Tokens (JWT). Medium. Recuperado de https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec [Consultado 10 Ago., 2019]

[28]. Symantec Corporation. (2018). *The Symantec Approach to Software Asset Management*. Recuperado de https://www.symantec.com/content/dam/symantec/docs/solution-briefs/the-symantec-approach-to-software-asset-management-en.pdf [Consultado 05 Mar., 2019]

[29]. TIOBE Software BV. (2019). *TIOBE Index for April 2019*. Recuperado de https://www.tiobe.com/tiobe-index/ [Consultado 02 May., 2019]

Anexos

Anexo A. Historias de usuario

En este apartado se describen los requerimientos del sistema preservados en las historias de usuario, de acuerdo a los diferentes roles.

Usuario:

<table>
<tr><th>Fecha</th><th>Autor</th><th>Versión</th><th>Referencia del cambio</th></tr>
<tr><td>22/04/2019</td><td>Ricardo Luis Peña</td><td>1.0</td><td>-</td></tr>
<tr><td colspan="2">Número: HU-US001</td><td colspan="2">Usuario: Responsable de Sistemas, Responsable de Seguridad, Director Ejecutivo.</td></tr>
<tr><td colspan="4">Nombre historia: Acceso seguro al sistema</td></tr>
<tr><td colspan="4">Prioridad en negocio: Muy alta</td></tr>
<tr><td colspan="2">Puntos estimados: 8</td><td colspan="2">Iteración asignada: 1</td></tr>
<tr><td colspan="4">Descripción:

Como Responsable de Sistemas, Responsable de Seguridad y Director Ejecutivo queremos acceder al sistema a mediante un nombre de usuario y contraseña, con el objetivo de garantizar un uso más restringido y protegido de intrusos.</td></tr>
<tr><td colspan="4">Observaciones:

• El nombre de usuario debe ser único en el sistema.
• La contraseña debe ser encriptada (con algoritmo sha-1 o md5).
Información del usuario:

• Nombre
• Apellido
• DNI
• CUIL
• Email</td></tr>
<tr><td colspan="4">Criterio de Aceptación: El sistema aceptó la combinación usuario/contraseña correcta, y rechazó el par incorrecto, informando en este último caso del error.</td></tr>
</table>

Tabla A.1. Historia de usuario para el requerimiento de "Acceso seguro al sistema". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-HU002	**Usuario:** Responsable de Sistemas, Responsable de Seguridad, Director Ejecutivo.
Nombre historia: Diseño adaptable a cualquier dispositivo	
Prioridad en negocio: Baja	
Puntos estimados: 1	**Iteración asignada:** 3

Descripción:

Como Responsable de Sistemas, Responsable de Seguridad y Director Ejecutivo, queremos que el sistema presente una interfaz adaptable a cualquier dispositivo, con el objetivo de tener mayor accesibilidad a las funcionalidades del sistema.

Observaciones: -

Criterio de Aceptación: Se redimensionó la pantalla y la vista se ajustó.

Tabla A.2. Historia de usuario para el requerimiento de "Diseño adaptable". Elaboración propia.

Responsable de Sistemas:

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-SI001	**Usuario:** Responsable de Sistemas
Nombre historia: ABMC de activos	
Prioridad en negocio: Muy alta	
Puntos estimados: 5	**Iteración asignada:** 1

Descripción:

Como Responsable de Sistemas, quiero registrar, actualizar, dar de baja y consultar los activos informáticos que recaen sobre mi responsabilidad; con el objetivo de organizar la información de tales activos y verificar su estado.

Observaciones: Formulario de carga de un nuevo activo: • Tipo (software, hardware, datos, comunicación, proveedores/terceros, proyecto TI, información no contenida en software, tecnología IoT) • Nombre (único) • Descripción • Ubicación • Estado • Versión • Propietario • Responsable (en caso de no especificarse, el responsable es el mismo usuario)
Criterio de Aceptación: se creó, consultó, modificó y eliminó un activo correctamente.

Tabla A.3. Historia de usuario para el requerimiento de "ABMC de activos". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-SI002	**Usuario:** Responsable de Sistemas
Nombre historia: ABMC de usuarios	
Prioridad en negocio: Alta	
Puntos estimados: 5	**Iteración asignada:** 2

Descripción:

Como Responsable de Sistemas, quiero registrar, actualizar, borrar y consultar los usuarios del sistema, con el objetivo de administrar los usuarios y sus permisos.

Observaciones:

Información del usuario:

- Nombre de usuario
- Contraseña
- Rol
- Nombre
- Apellido

• CUIL • Email La modificación del usuario sólo será del rol, puesto que la información personal (ej.: nombre) será modificada por el propio usuario dueño de la misma.
Criterio de Aceptación: se creó, actualizó y borró un usuario del sistema.

Tabla A.4. Historia de usuario para el requerimiento de "ABMC de usuarios". Elaboración propia.

Responsable de Seguridad:

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS001	**Usuario:** Responsable de Seguridad
Nombre historia: ABMC de amenazas	
Prioridad en negocio: Muy alta	
Puntos estimados: 3	**Iteración asignada:** 1
Descripción: Como Responsable de Seguridad, quiero registrar, actualizar, dar de baja y consultar amenazas de seguridad, con el objetivo de mantener organizada su información y reconocer con mayor facilidad a qué activos afectan tales eventualidades.	
Observaciones: Información de la amenaza: • Nombre • Descripción	
Criterio de Aceptación: se creó, actualizó y eliminó una amenaza correctamente.	

Tabla A.5. Historia de usuario para el requerimiento de "ABMC de amenazas". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS002	**Usuario:** Responsable de Seguridad

Nombre historia: ABMC de riesgos	
Prioridad en negocio: Muy alta	
Puntos estimados: 5	**Iteración asignada:** 1
Descripción: Como Responsable de Seguridad, quiero registrar, actualizar, dar de baja y consultar riesgos de seguridad sobre los activos informáticos de la organización, con el objetivo de conocer qué amenazas afectan a cada activo y en qué medida lo hacen, y decidir las medidas a llevar a cabo.	
Observaciones: Información del riesgo: • Valor de criticidad • Probabilidad de ocurrencia • Impacto cuantitativo en el negocio • Impacto cualitativo en el negocio • Valor monetario (opcional) • Descripción • Amenaza • Activo	
Criterio de Aceptación: se registró, actualizó, consultó y dio de baja un riesgo satisfactoriamente.	

Tabla A.6. Historia de usuario para el requerimiento de "ABMC de riesgos". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS003	**Usuario:** Responsable de Seguridad
Nombre historia: Cálculo de la criticidad del riesgo	
Prioridad en negocio: Media	
Puntos estimados: 1	**Iteración asignada:** 3

Descripción:

Como Responsable de Seguridad, quiero que el sistema realice el cálculo automático del valor de criticidad del riesgo en función de su probabilidad e impacto, con el objetivo de facilitar el análisis cuantitativo de tal riesgo.

Observaciones:

Niveles de probabilidad de ocurrencia y su valor:

- Muy difícil que suceda: 0.1
- Difícil que suceda: 0.3
- Posiblemente suceda: 0.5
- Muy probable que suceda: 0.7
- Casi seguro que sucede: 0.9

Valores de impacto sobre desempeño del activo:

- No inhabilita sus funciones: 0.1
- Lo inhabilita en menos de 10%: 0.3
- Lo inhabilita entre 10 y 25%: 0.5
- Lo inhabilita entre 25 y 50%: 0.7
- Lo inhabilita en más del 50%: 0.9

Valores de impacto sobre el tiempo:

- No tiene un efecto considerable: 0.1
- Retrasa hasta un 5%: 0.3
- Retrasa entre 5 y 10%: 0.5
- Retrasa entre 10 y 15%: 0.7
- Retrasa más del 15%: 0.9

Impacto sobre el costo económico:

- No afecta demasiado el costo: 0.1
- Tiene un costo adicional de menos del 5%: 0.3
- Tiene un costo adicional entre 5 y 10%: 0.5
- Tiene un costo adicional entre 10 y 15%: 0.7
- Tiene un costo adicional de más del 15%: 0.9

Criterio de Aceptación: al consultar un riesgo se observó su valor de criticidad ya calculado.

Tabla A.7. Historia de usuario para el requerimiento de "Criticidad de riesgo". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS004	**Usuario:** Responsable de Seguridad, Responsable de Sistemas
Nombre historia: ABMC de controles	
Prioridad en negocio: Alta	
Puntos estimados: 3	**Iteración asignada:** 2

Descripción:

Como Responsable de Seguridad y Responsable de Sistemas, queremos registrar, actualizar, dar de baja y consultar controles de seguridad, con el objetivo de evaluar el desempeño de tales medidas en relación a los objetivos deseados.

Observaciones:

Información del control:

- Nombre
- Descripción
- Identificador unívoco
- Periodicidad (en meses)
- Norma/política asociada (opcional)
- Responsable.

Criterio de Aceptación: se registró, consultó, modificó y borró un control satisfactoriamente.

Tabla A.8. Historia de usuario para el requerimiento de "ABMC de controles". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS005	**Usuario:** Responsable de Seguridad
Nombre historia: Asignación de control a riesgo	
Prioridad en negocio: Alta	

Puntos estimados: 3	**Iteración asignada:** 2
Descripción: Como Responsable de Seguridad, quiero asignar los controles de seguridad a los riesgos identificados según [HU-RS002], con el objetivo de verificar la reducción o mitigación de los mismos.	
Observaciones: -	
Criterio de Aceptación: se asoció un control a un riesgo, y se puede observar la definición creada.	

Tabla A.9. Historia de usuario para el requerimiento de "Asignar un control a riesgo". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS006	**Usuario:** Responsable de Seguridad
Nombre historia: ABMC de políticas	
Prioridad en negocio: Media	
Puntos estimados: 3	**Iteración asignada:** 3
Descripción: Como Responsable de Seguridad, quiero registrar, actualizar, dar de baja y consultar políticas y normas organizacionales, con el objetivo de asociar las mismas a los controles de seguridad.	
Observaciones: Información de la política: • Nombre • Descripción • Año de redacción	
Criterio de Aceptación: El sistema registró, modificó, y dio de baja correctamente una política.	

Tabla A.10. Historia de usuario para el requerimiento de "ABMC de políticas". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS007	**Usuario:** Responsable de Seguridad
Nombre historia: Prueba de controles de seguridad	
Prioridad en negocio: Alta	
Puntos estimados: 3	**Iteración asignada:** 2
Descripción: Como Responsable de Seguridad, quiero indicar los controles que fueron probados, con el objetivo de justificar su cumplimiento.	
Observaciones: -	
Criterio de Aceptación: Al indicar un control como probado, el sistema registró el historial de ejecución, y por ello figura en las ejecuciones de los controles.	

Tabla A.11. Historia de usuario para el requerimiento de "Probar controles". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-RS008	**Usuario:** Responsable de Seguridad
Nombre historia: Historial de ejecución de controles	
Prioridad en negocio: Alta	
Puntos estimados: 5	**Iteración asignada:** 2
Descripción: Como Responsable de Seguridad, quiero registrar la información de la ejecución de un control en el momento en que indico su realización (según HU-RS007), con el objetivo de preservar estos eventos, consultarlos y facilitar su auditoría.	
Observaciones:	
Criterio de Aceptación: Se indicó la realización de un control e inmediatamente se guardó la información de su ejecución.	

Tabla A.12. Historia de usuario para el requerimiento de "Historial de controles". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio

22/04/2019	Ricardo Luis Peña	1.0	-
Número: HU-RS009		**Usuario:** Responsable de Seguridad	
Nombre historia: Consulta de usuarios registrados			
Prioridad en negocio: Media			
Puntos estimados: 3		**Iteración asignada:** 3	
Descripción: Como Responsable de Seguridad, quiero consultar la información de los usuarios registrados en el sistema, con el objetivo de verificar la existencia de los mismos y la correcta asignación de sus roles.			
Observaciones:			
Criterio de Aceptación: Una vez seleccionada la opción de consultar usuarios, el sistema muestra una tabla con todos los usuarios registrados.			

Tabla A.13. Historia de usuario para el requerimiento de "Usuarios registrados". Elaboración propia.

Director Ejecutivo:

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-
Número: HU-DE001		**Usuario:** Director Ejecutivo	
Nombre historia: Visualización de matriz de riesgos			
Prioridad en negocio: Alta			
Puntos estimados: 13		**Iteración asignada:** 1	
Descripción: Como Director Ejecutivo, quiero visualizar los riesgos sobre los activos informáticos en una matriz de probabilidad e impacto, con el objetivo de reconocer con facilidad aquellos riesgos con mayor criticidad y tomar medidas en consecuencia.			
Observaciones: Escala de valores de criticidad y color asociado: • Mínimo: <10% (Verde)			

- Moderado: >10% y <40% (Amarillo)
- Crítico: >40% (Rojo)

Eje Horizontal:

- Etiqueta: Impacto
- Valores posibles: despreciable, marginal, moderado, crítico, catastrófico

Eje Vertical:

- Etiqueta: Probabilidad
- Valores posibles: excepcional, improbable, probable, posible, cierto

Criterio de Aceptación: El sistema presentó la matriz de riesgos con colores establecidos.

Tabla A.14. Historia de usuario para el requerimiento de "Visualizar matriz de riesgos". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio
22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-DE002	**Usuario:** Director Ejecutivo
Nombre historia: Vencimiento de control sobre riesgo	
Prioridad en negocio: Alta	
Puntos estimados: 5	**Iteración asignada:** 2

Descripción:

Como Director Ejecutivo, quiero que el sistema verifique la caducidad de un determinado control sobre un riesgo y actualice la criticidad de este último en función a ese vencimiento, con el objetivo de conocer el estado de los controles implementados y saber cuándo se deben aplicar.

Observaciones: -

Criterio de Aceptación: El sistema actualizó el valor del riesgo al vencerse el control aplicado.

Tabla A.15. Historia de usuario para el requerimiento de "Vencimiento de un control". Elaboración propia.

Fecha	Autor	Versión	Referencia del cambio

22/04/2019	Ricardo Luis Peña	1.0	-

Número: HU-DE003	**Usuario:** Director Ejecutivo
Nombre historia: Matriz dinámica de riesgos	
Prioridad en negocio: Media	
Puntos estimados: 8	**Iteración asignada:** 3
Descripción: Como Director Ejecutivo, quiero que la matriz de probabilidad e impacto se actualice automáticamente al agregar, modificar o quitar un activo o riesgo determinado, con el objetivo de lograr una mayor flexibilidad al gestionar los riesgos.	
Observaciones: -	
Criterio de Aceptación: Se modificó el valor de un riesgo y la matriz se actualizó automáticamente.	

Tabla A.16. Historia de usuario para el requerimiento de "Matriz dinámica de riesgos". Elaboración propia.

Anexo B. Escenarios de casos de uso

A continuación, se detallan los diferentes escenarios de interacción entre el sistema y sus usuarios, de acuerdo al rol definido para estos últimos.

Usuario:

Nombre : Acceso seguro [CU-US001]
Referencia a Requerimiento: [HU-US001: Acceso seguro al sistema]
Descripción: Proceso de acceso al sistema por parte de un usuario a través de un nombre único y contraseña.
Actores: Usuario
Precondiciones: El usuario se encuentra registrado en el sistema. El usuario seleccionó la opción de ingresar.
Flujo Normal: Paso 1: El sistema despliega el formulario de ingreso con nombre y contraseña Paso 2: El usuario ingresa el nombre único Paso 3: El sistema verifica la existencia del nombre ingresado Paso 4: El usuario ingresa la contraseña. Paso 5: El sistema verifica la correlación entre nombre y contraseña Paso 6: El sistema presenta al usuario la vista principal (home)

Flujo Anormal: Paso 2: El sistema detectó que el nombre de usuario no existe a) Informa al usuario del error b) Cancela la operación. c) Solicita el reingreso del nombre de usuario
Postcondiciones: El usuario ingresó correctamente al sistema

Tabla B.1. Escenario del caso de uso "Acceso seguro". Elaboración propia.

Responsable de Sistemas:

Nombre: Registro de activo [CU-SI001]
Referencia a Requerimiento: [HU-SI001:ABMC de Activos]
Descripción: Registro en el sistema de un nuevo activo informático
Actores: Responsable de Sistemas
Precondiciones: El responsable de sistemas ingresó al sistema correctamente autenticado.
Flujo Normal: Paso 1: El responsable de sistemas selecciona la opción de cargar un activo. Paso 2: El sistema presenta un formulario a completar. Paso 3: El responsable de sistemas completa el formulario con la información del nuevo activo. Paso 4: El responsable de sistemas selecciona la opción "Registrar". Paso 5: El sistema verifica la validez de los datos ingresados. Paso 6: El sistema registra el nuevo activo.
Flujo Anormal: Paso 5: El sistema detectó que la información ingresada es errónea o vacía. a) Informa del error. b) Solicita el ingreso de datos válidos. c) Vuelve al paso 2.
Postcondiciones: El responsable de sistemas registró un nuevo activo informático en el sistema.

Tabla B.2. Escenario del caso de uso para registrar activos. Elaboración propia.

Nombre: Baja de activo [CU-SI002]
Referencia a Requerimiento: [HU-SI001:ABMC de Activos]
Descripción: Eliminación de un activo informático del sistema.
Actores: Responsable de Sistemas
Precondiciones: El responsable de sistemas ingresó correctamente al sistema.

Flujo Normal: Paso 1: Se ejecuta el caso de uso “Consulta de activo” [CU-SI004]. Paso 2: El responsable de sistemas selecciona un activo. Paso 3: El responsable de sistemas selección la opción de “Eliminar”. Paso 4: El sistema solicita confirmación de la operación. a) Si el responsable de sistemas decide cancelar, entonces el sistema cancela la operación. b) Si el responsable de sistemas decide confirmar, entonces continua el paso 5. Paso 5: El sistema elimina el activo. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró activos registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de sistemas dio de baja un activo del sistema.

Tabla B.3. Escenario del caso de uso para eliminar activos. Elaboración propia.

Nombre: Actualización de activo [CU-SI003]
Referencia a Requerimiento: [HU-SI001-ABMC de Activos]
Descripción: Modificación de la información de un activo informático registrado en el sistema.
Actores: Responsable de Sistemas
Precondiciones: El responsable de sistemas ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso “Consulta de activo” [CU-SI004]. Paso 2: El responsable de sistemas selecciona un activo. Paso 3: El responsable de sistemas selecciona la opción de “Modificar”. Paso 4: El sistema presenta una vista con la información del activo seleccionado. Paso 5: El responsable de sistemas modifica la información del activo. Paso 6: El responsable de sistemas selecciona la opción de “Actualizar”. Paso 7: El sistema verifica la validez de los datos modificados. Paso 8: El sistema registra la modificación del activo. Paso 9: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró activos registrados. a) Informa de la situación. b) Cancela la operación. Paso 8: El sistema detectó que la información ingresada es errónea o vacía. a) Informa del error. b) Solicita el ingreso de datos válidos. c) Vuelve al paso 4.

Postcondiciones: El responsable de sistemas actualizó la información de un activo en el sistema.

Tabla B.4. Escenario del caso de uso para actualizar activos. Elaboración propia.

Nombre: Consulta de activo [CU-SI004]
Referencia a Requerimiento: [HU-SI001: ABMC de Activos]
Descripción: Consulta de activos informáticos registrados en el sistema
Actores: Responsable de Sistemas
Precondiciones: El responsable de sistemas ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de sistemas selecciona la opción “Mis activos”. Paso 2: El sistema busca los activos registrados por el operativo. a) Si no encontró activos, informa de la situación. b) Si encontró activos registrados, continúa al paso 3. Paso 3: El sistema presenta los activos informáticos registrados por el operativo.
Flujo anormal:
Postcondiciones: El responsable de sistemas obtuvo los activos registrados en el sistema.

Tabla B.5. Escenario del caso de uso para consultar activos. Elaboración propia.

Nombre: Registro de usuario [CU-SI005]
Referencia a Requerimiento: [HU-SI002:ABMC de usuarios]
Descripción: Registro de un nuevo usuario en el sistema.
Actores: Responsable de Sistemas
Precondiciones: El responsable de sistemas ingresó al sistema correctamente.
Flujo Normal: Paso 1: El responsable de sistemas selecciona la opción “Registrar usuario”. Paso 2: El sistema presenta un formulario a completar con la información del usuario. Paso 3: El responsable de sistemas ingresa la información del usuario. Paso 4: El responsable de sistemas selecciona la opción “Registrar”. Paso 5: El sistema verifica que la información ingresada es correcta. Paso 6: El sistema registra el nuevo usuario. Paso 7: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 5: El sistema encontró información incorrecta o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 2.

Postcondiciones: El responsable de sistemas registró un nuevo usuario en el sistema.

Tabla B.6. Escenario del caso de uso para registrar usuarios. Elaboración propia.

Nombre: Baja de usuario [CU-SI006]
Referencia a Requerimiento: [HU-SI002: ABMC de usuarios]
Descripción: Eliminación de un usuario del sistema.
Actores: Responsable de sistemas
Precondiciones: El responsable de sistemas ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de usuario" [CU-SI008]. Paso 2: El responsable de sistemas selecciona un usuario. Paso 3: El responsable de sistemas selecciona la opción "Dar de baja". Paso 4: El sistema solicita confirmación de la operación. a) Si el responsable de sistemas no confirma, el sistema cancela la operación. b) Si el responsable de sistemas confirma la operación, sigue al paso 5. Paso 5: El sistema da de baja al usuario seleccionado. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró usuarios registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de sistemas dio de baja un usuario del sistema.

Tabla B.7. Escenario del caso de uso para eliminar usuarios. Elaboración propia.

Nombre: Actualización del rol de usuario [CU-SI007]
Referencia a Requerimiento: [HU-SI002: ABMC de usuarios]
Descripción: Modificación del rol de un usuario del sistema.
Actores: Responsable de sistemas
Precondiciones: El responsable de sistemas ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de usuarios" [CU-SI008]. Paso 2: El responsable de sistemas selecciona un usuario. Paso 3: El responsable de sistemas selecciona la opción "Modificar rol usuario". Paso 4: El sistema muestra una vista con la información del rol del usuario. Paso 5: El responsable de sistemas modifica el rol del usuario. Paso 6: El responsable de sistemas selecciona la opción "Actualizar". Paso 7: El sistema solicita la confirmación de la operación. a) Si el responsable de sistemas no confirma, entonces el sistema cancela la operación. b) Si el responsable de sistemas confirma, entonces continúa al paso 8.

Paso 8: El sistema actualiza el rol del usuario seleccionado. Paso 9: El sistema informa el éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró usuarios registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de sistemas actualizó el rol de un usuario del sistema.

Tabla B.8. Escenario del caso de uso para actualizar rol de usuario. Elaboración propia.

Nombre: Consulta de usuarios [CU-SI008]
Referencia a Requerimiento: [HU-SI002: ABMC de usuarios], [HU-RS009: Consulta de usuarios registrados]
Descripción: Consulta de la información de los usuarios registrados en el sistema.
Actores: Responsable de Sistemas, Responsable de Seguridad [responsable]
Precondiciones: El responsable ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable selecciona la opción “Consultar usuarios”. Paso 2: El sistema busca los usuarios registrados. Paso 3: El sistema muestra la información de los usuarios registrados.
Flujo Anormal: Paso 2: El sistema no encontró usuarios registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable accedió a la información de los usuarios registrados.

Tabla B.9. Escenario del caso de uso para consultar usuarios. Elaboración propia.

Responsable de Seguridad:

Nombre: Registro de amenaza [CU-RS001]
Referencia a Requerimiento: [HU-RS001: ABMC de amenazas]
Descripción: Registro en el sistema de una nueva amenaza.
Actores: Responsable de seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.

Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Registrar amenaza”. Paso 2: El sistema presenta un formulario con la información requerida. Paso 3: El responsable de seguridad completa el formulario con los datos de la amenaza. Paso 4: El responsable de seguridad selecciona la opción “Registrar”. Paso 5: El sistema verifica que la información ingresada es correcta. Paso 6: El sistema registra la nueva amenaza. Paso 7: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 5: El sistema encontró información errónea o vacía. a) Informa del error. b) Cancela la operación. c) Vuelve al paso 2.
Postcondiciones: El responsable de seguridad dio de alta una nueva amenaza.

Tabla B.10. Escenario del caso de uso para registrar amenazas. Elaboración propia.

Nombre: Baja de amenaza [CU-RS002]
Referencia a Requerimiento: [HU-RS001: ABMC de amenazas]
Descripción: Eliminación de una amenaza en el sistema.
Actores: Responsable de seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso “Consulta de amenaza” [CU-RS004] Paso 2: El responsable de seguridad selecciona una amenaza. Paso 3: El responsable de seguridad selecciona la opción “Eliminar”. Paso 4: El sistema solicita la confirmación de la operación. a) Si el responsable de seguridad decide cancelar, entonces el sistema cancela la operación. b) Si el responsable de seguridad decide confirmar, entonces continúa al paso 5. Paso 5: El sistema da de baja la amenaza. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró amenazas registradas. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad dio de baja la amenaza del sistema.

Tabla B.11. Escenario del caso de uso para eliminar amenazas. Elaboración propia.

Nombre: Actualización de amenaza [CU-RS003]
Referencia a Requerimiento: [HU-RS001:ABMC de amenazas]

Descripción: Modificación de la información de una amenaza.
Actores: Responsable de seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de amenaza" [CU-RS004]. Paso 2: El responsable de seguridad selecciona una amenaza. Paso 3: El responsable de seguridad selecciona la opción de "Editar". Paso 4: El sistema presenta una vista con la información de la amenaza seleccionada. Paso 5: El responsable de seguridad modifica la información de la amenaza. Paso 6: El responsable de seguridad selecciona la opción de "Actualizar". Paso 7: El sistema verifica que los datos ingresados sean correctos. Paso 8: El sistema actualiza la información de la amenaza. Paso 9: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró amenazas registradas. a) Informa de la situación. b) Cancela la operación. Paso 7: El sistema encontró información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información de la amenaza. c) Vuelve al paso 4.
Postcondiciones: El responsable de seguridad actualizó una amenaza.

Tabla B.12. Escenario del caso de uso para actualizar amenazas. Elaboración propia.

Nombre: Consulta de amenaza [CU-RS004]
Referencia a Requerimiento: [HU-RS001: ABMC de amenazas]
Descripción: Consulta de las amenazas registradas en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción de "Amenazas". Paso 2: El sistema busca las amenazas registradas. Paso 3: El sistema presenta las amenazas registradas y su información.
Flujo Anormal: Paso 2: El sistema no encontró amenazas registradas. a) Informa de la situación. b) Cancela la operación.

Postcondiciones: El responsable de seguridad obtuvo información de las amenazas registradas.

Tabla B.13. Escenario del caso de uso para consultar amenazas. Elaboración propia.

Nombre: Registro de riesgo [CU-RS005]
Referencia a Requerimiento: [HU-RS002: ABMC de riesgos], [HU-RS003: Cálculo de la criticidad del riesgo]
Descripción: Registro de un nuevo riesgo al sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Registrar activo”. Paso 2: El sistema presenta un formulario con la información a completar. Paso 3: El responsable de seguridad completa los datos del formulario. Paso 4: El responsable de seguridad selecciona “Registrar”. Paso 5: El sistema verifica que la información ingresada sea correcta. Paso 6: El sistema calcula la criticidad del riesgo. Paso 7: El sistema registra el nuevo riesgo. Paso 8: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 5: El sistema encontró información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 2.
Postcondiciones: El responsable de seguridad registró un nuevo riesgo en el sistema.

Tabla B.14. Escenario del caso de uso para registrar riesgos. Elaboración propia.

Nombre: Baja de riesgo [CU-RS006]
Referencia a Requerimiento: [HU-RS002: ABMC de riesgos]
Descripción: Eliminación de un riesgo del sistema.
Actores: Responsable de seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso “Consulta de riesgo” [CU-RS008]. Paso 2: El responsable de seguridad selecciona un riesgo. Paso 3: El responsable de seguridad selecciona la opción “Eliminar”. Paso 4: El sistema solicita la confirmación de la operación. a) Si el responsable de seguridad no confirma, cancela la operación. b) Si el responsable de seguridad confirma, entonces continúa al paso 5. Paso 5: El sistema elimina el riesgo seleccionado.

Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad dio de baja un riesgo del sistema.

Tabla B.15. Escenario del caso de uso para eliminar riesgos. Elaboración propia.

Nombre: Actualización de riesgo [CU-RS007]
Referencia a Requerimiento: [HU-RS002: ABMC de riesgos]
Descripción: Modificación de la información de un riesgo determinado
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de riesgo" [CU-RS008]. Paso 2: El responsable de seguridad selecciona un riesgo. Paso 3: El responsable de seguridad selecciona la opción "Modificar". Paso 4: El sistema presenta una vista con la información del riesgo seleccionado. Paso 5: El responsable de seguridad modifica la información del riesgo. Paso 6: El responsable de seguridad selecciona la opción "Actualizar". Paso 7: El sistema verifica que la información ingresada sea correcta. Paso 8: El sistema actualiza la información del riesgo. Paso 9: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación. Paso 7: El sistema encontró información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 4.
Postcondiciones: El responsable de seguridad actualizó la información de un riesgo.

Tabla B.16. Escenario del caso de uso para actualizar riesgos. Elaboración propia.

Nombre: Consulta de riesgo [CU-RS008]

Referencia a Requerimiento: [HU-RS002: ABMC de riesgos]
Descripción: Consulta de los riesgos registrados en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona “Riesgos”. Paso 2: El sistema busca riesgos registrados. Paso 3: El sistema muestra los riesgos registrados y su información detallada.
Flujo Anormal: Paso 2: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad obtuvo los riesgos registrados en el sistema.

Tabla B.17. Escenario del caso de uso para consultar riesgos. Elaboración propia.

Nombre: Registro de control [CU-RS009]
Referencia a Requerimiento: [HU-RS004: ABMC de controles]
Descripción: Registro de un nuevo control de seguridad en el sistema.
Actores: Responsable de seguridad, Responsable de Sistemas [responsable]
Precondiciones: El responsable ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable selecciona “Nuevo Control”. Paso 2: El sistema presenta un formulario con la información requerida del control. Paso 3: El responsable completa el formulario. Paso 4: El responsable selecciona “Registrar”. Paso 5: El sistema verifica que la información ingresada sea correcta. Paso 6: El sistema registra el nuevo control. Paso 7: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 5: El sistema identificó información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 2.
Postcondiciones: El responsable registró un nuevo control en el sistema..

Tabla B.18. Escenario del caso de uso para registrar controles. Elaboración propia.

Nombre: Baja de control [CU-RS010]
Referencia a Requerimiento: [HU-RS004: ABMC de controles]
Descripción: Eliminación de un control del sistema.
Actores: Responsable de Seguridad, Responsable de Sistemas [responsable]
Precondiciones: El responsable ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de control" [CU-RS012]. Paso 2: El responsable selecciona un control determinado. Paso 3: El responsable selecciona la opción "Eliminar". Paso 4: El sistema solicita la confirmación de la operación. a) Si el responsable no confirma, se cancela la operación. b) Si el responsable confirma, entonces continúa al paso 5. Paso 5: El sistema elimina el control seleccionado. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró controles registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable dio de baja un control del sistema.

Tabla B.19. Escenario del caso de uso para eliminar controles. Elaboración propia.

Nombre: Actualización de control [CU-RS011]
Referencia a Requerimiento: [HU-RS004: ABMC de controles]
Descripción: Modificación de la información de un control en el sistema.
Actores: Responsable de seguridad, Responsable de Sistemas [responsable]
Precondiciones: El responsable ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de control" [CU-RS012]. Paso 2: El responsable selecciona un control determinado. Paso 3: El responsable selecciona la opción "Modificar". Paso 4: El sistema muestra una vista con la información del control seleccionado. Paso 5: El responsable modifica la información del control. Paso 6: El responsable selecciona la opción "Actualizar". Paso 7: El sistema verifica que la información ingresada sea correcta. Paso 8: El sistema actualiza la información del control. Paso 9: El sistema informa del éxito de la operación.

Flujo Anormal: Paso 1: El sistema no encontró controles registrados. a) Informa de la situación. b) Cancela la operación. Paso 7: El sistema identificó información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 4.
Postcondiciones: El responsable actualizó la información de un control.

Tabla B.20. Escenario del caso de uso para actualizar controles. Elaboración propia.

Nombre: Consulta de control [CU-RS012]
Referencia a Requerimiento: [HU-RS004: ABMC de controles]
Descripción: Consulta de los controles registrados en el sistema.
Actores: Responsable de Seguridad, Responsable de Sistemas [responsable]
Precondiciones: El responsable ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable selecciona “Controles”. Paso 2: El sistema busca los controles registrados. Paso 3: El sistema presenta los controles registrados con su información detallada.
Flujo Anormal: Paso 2: El sistema no encontró controles registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable obtuvo información de los controles registrados.

Tabla B.21. Escenario del caso de uso para consultar controles. Elaboración propia.

Nombre: Asignación de control a un riesgo [CU-RS013]
Referencia a Requerimiento: [HU-RS005: Asignación de control a riesgo]
Descripción: Asignación de un control de seguridad a un riesgo determinado.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.

Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Asignar un control”. Paso 2: El sistema presenta los riesgos registrados. Paso 3: El responsable de seguridad selecciona un riesgo. Paso 4: El sistema presenta los controles registrados. Paso 5: El responsable de seguridad selecciona un control. Paso 6: El responsable de seguridad selecciona la opción “Asignar”. Paso 7: El sistema asigna el control al riesgo. Paso 8: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 2: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación. Paso 4: El sistema no encontró controles registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad asignó un control a un riesgo determinado.

Tabla B.22. Escenario del caso de uso para asignar controles a riesgos. Elaboración propia.

Nombre: Registro de política [CU-RS014]
Referencia a Requerimiento: [HU-RS006: ABMC de políticas]
Descripción: Registro de una nueva política en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad seleccionó la opción “Registrar política”. Paso 2: El sistema presenta un formulario a completar con la información de la política. Paso 3: El responsable de seguridad completa el formulario. Paso 4: El responsable de seguridad selecciona la opción “Registrar”. Paso 5: El sistema verifica que la información ingresada sea correcta. Paso 6: El sistema registra la nueva política. Paso 7: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 5: El sistema encontró información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 2.
Postcondiciones: El responsable de seguridad registró una nueva política en el sistema.

Tabla B.23. Escenario del caso de uso para registrar políticas. Elaboración propia.

Nombre: Baja de política [CU-RS015]
Referencia a Requerimiento: [HU-RS006: ABMC de políticas]
Descripción: Eliminación de una política en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de política" [CU-RS017]. Paso 2: El responsable de seguridad selecciona una política. Paso 3: El responsable de seguridad selecciona la opción "Eliminar". Paso 4: El sistema solicita confirmación de la operación. a) Si el responsable de seguridad decide no confirmar, cancela la operación. b) Si el responsable de seguridad decide confirmar, continúa al paso 5. Paso 5: El sistema da de baja la política. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 1: El sistema no encontró políticas registradas. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad dio de baja una política del sistema.

Tabla B.24. Escenario del caso de uso para eliminar políticas. Elaboración propia.

Nombre: Actualización de política [CU-RS016]
Referencia a Requerimiento: [HU-RS006: ABMC de políticas]
Descripción: Modificación de la información de una política en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: Se ejecuta el caso de uso "Consulta de política" [CU-RS017]. Paso 2: El responsable de seguridad selecciona una política. Paso 3: El responsable de seguridad selecciona la opción "Modificar". Paso 4: El sistema presenta una vista con la información de la política seleccionada. Paso 5: El responsable de seguridad modifica la información de la política. Paso 4: El responsable de seguridad selecciona la opción "Actualizar". Paso 5: El sistema verifica que la información ingresada sea correcta. Paso 6: El sistema actualiza la información de la política. Paso 7: El sistema informa del éxito de la operación.

Flujo Anormal: Paso 1: El sistema no encontró políticas registradas. a) Informa de la situación. b) Cancela la operación. Paso 5: El sistema encontró información errónea o vacía. a) Informa del error. b) Solicita el reingreso de la información. c) Vuelve al paso 4.
Postcondiciones: El responsable de seguridad actualizó una política en el sistema.

Tabla B.25. Escenario del caso de uso para actualizar políticas. Elaboración propia.

Nombre: Consulta de política [CU-RS017]
Referencia a Requerimiento: [HU-RS006: ABMC de políticas]
Descripción: Consulta de las políticas registradas en el sistema.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Políticas”. Paso 2: El sistema busca las políticas registradas. Paso 3: El sistema muestra las políticas registradas y su información detallada.
Flujo Anormal: Paso1: El sistema no encontró políticas registradas. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad accedió a las políticas registradas en el sistema.

Tabla B.26. Escenario del caso de uso para consultar políticas. Elaboración propia.

Nombre: Prueba de control [CU-RS018]
Referencia a Requerimiento: [HU-RS007: Prueba de controles de seguridad], [HU-RS008: Historial de ejecución de controles]
Descripción: Comprobación de la aplicación de los controles definidos.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.

Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Probar controles”. Paso 2: El sistema presenta una vista con los controles definidos y aún no probados. Paso 3: El responsable de seguridad selecciona un control. Paso 4: El responsable de seguridad selecciona la opción “Probar”. Paso 5: El sistema registra la ejecución del control. Paso 6: El sistema informa del éxito de la operación.
Flujo Anormal: Paso 2: El sistema no encontró controles por probar. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad probó un control en el sistema.

Tabla B.27. Escenario del caso de uso para probar controles. Elaboración propia.

Nombre: Consulta de la ejecución de un control [CU-RS019]
Referencia a Requerimiento: [HU-RS008: Historial de ejecución de controles]
Descripción: Consulta de los registros de ejecución de los controles de seguridad.
Actores: Responsable de Seguridad
Precondiciones: El responsable de seguridad ingresó correctamente al sistema.
Flujo Normal: Paso 1: El responsable de seguridad selecciona la opción “Ejecuciones de controles”. Paso 2: El sistema busca las transacciones de ejecución de controles registradas. Paso 3: El sistema presenta las ejecuciones de los controles y su información.
Flujo Anormal: Paso 2: El sistema no encontró ejecuciones registradas. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El responsable de seguridad consultó los controles ejecutados.

Tabla B.28. Escenario del caso de uso para consultar controles ejecutados. Elaboración propia.

Director Ejecutivo:

Nombre: Visualización de matriz de riesgos [CU-DE001]
Referencia a Requerimiento: [HU-DE001: Visualización de matriz de riesgos]
Descripción: Visualización de riesgos en una matriz de probabilidad e impacto.
Actores: Director Ejecutivo

Precondiciones: El director ejecutivo ingresó correctamente al sistema.
Flujo Normal: Paso 1: El director ejecutivo selecciona la opción “Riesgos de seguridad”. Paso 2: El sistema busca los riesgos registrados y su información. Paso 3: El sistema presenta la matriz de probabilidad e impacto con los riesgos agrupados según su criticidad.
Flujo Anormal: Paso 2: El sistema no encontró riesgos registrados. a) Informa de la situación. b) Cancela la operación.
Postcondiciones: El director ejecutivo accedió a la matriz de riesgos.

Tabla B.29. Escenario del caso de uso para visualizar la matriz de riesgos. Elaboración propia.