UNIVERSIDAD DE LA EMPRESA

FACULTAD DE INGENIERÍA

TÉCNICO EN INFORMÁTICA Y ANALISTA EN TECNOLOGÍA INFORMÁTICA

PROYECTO INTEGRADOR

Herramienta de entrenamiento y capacitación para evitar fraudes a través de Ingeniería Social.

Integrantes:

Diego Balbis — C.I. 4.451.623-5
Dayron Muñiz — C.I. 6.295.606-3

Tutor:

Ing. Pablo Martres

Año:

2025

Agradecimientos

A la Universidad de la Empresa por brindarnos las herramientas académicas necesarias para concretar esta etapa.
A nuestro tutor, Ing. Pablo Martres, por su orientación técnica y acompañamiento constante durante el desarrollo del proyecto.
Y a nuestras familias y amigos, por su apoyo incondicional a lo largo de la carrera.

1. Introducción y Descripción del Problema

Los ciberataques a través de la ingeniería social en la actualidad son más comunes de lo que podemos imaginar, a diferencia de los ataques puramente técnicos, estos no buscan vulnerar directamente los sistemas informáticos, sino que se aprovechan del factor humano, haciendo uso de la confianza y de las emociones de las personas para obtener información sensible o acceso no autorizado.

Es por ello que aunque se han destinado importantes recursos a reforzar la infraestructura tecnológica, persiste una problemática en la capacitación de los usuarios en estos temas.

No son pocas las ocasiones en que los atacantes logran su cometido no por fallas técnicas, sino por desconocimiento o falta de entrenamiento frente a técnicas de manipulación.

Con este proyecto nos proponemos realizar una herramienta educativa que posibilite disminuir la vulnerabilidad humana ante los diferentes tipos de ataques basados en ingeniería social.

2. Objetivos

2.1. Objetivo General

Desarrollar una herramienta web interactiva de entrenamiento en ciberseguridad, orientada a la prevención de fraudes mediante ingeniería social. El sistema debe ofrecer contenidos educativos accesibles y actividades didácticas que promuevan el aprendizaje práctico sobre seguridad digital.

2.1.1 Objetivos Específicos

- Desarrollar una herramienta que impacte positivamente en la formación de otros usuarios y cree conciencia de la importancia de la ciberseguridad en la era digital.

- Con el desarrollo de la aplicación lograr culminar satisfactoriamente la Carrera y poder obtener el título de Técnico en Informática y Analista en Tecnología Informática

3. Alcance del proyecto

La herramienta estará destinada a toda persona interesada en aprender sobre fraudes digitales, con especial énfasis en estudiantes, trabajadores y organizaciones que busquen reforzar su cultura de seguridad.

El sistema se concentrará exclusivamente en los ataques de ingeniería social, entendidos como aquellos en los que la manipulación psicológica sustituye a la explotación técnica de vulnerabilidades. Por tanto, no se abordarán ataques puramente informáticos, sino aquellos en los que el atacante obtiene su beneficio mediante engaño o persuasión.

4. Especificación de Requerimientos

4.1. Introducción

En esta sección se establecen los requerimientos del sistema CAPFIS (Capacitación en Fraudes a Través de Ingeniería Social) , con la finalidad de eliminar imprecisiones durante su desarrollo.

El documento describe los requerimientos funcionales y no funcionales necesarios para el desarrollo efectivo del sistema.

4.2. Objetivo

El objetivo es documentar de manera completa y precisa todas las funcionalidades de la herramienta para que así sean comprendidos correctamente por los desarrolladores y partes interesadas.

4.3. Definiciones

Con el objetivo de que todos los involucrados comprendan claramente los conceptos clave del proyecto, se presentan las definiciones de los términos más importantes de CAPFIS.

CAPFIS: Página web destinada a la capacitación de los usuarios en ciberseguridad, espacialmente en la detección y prevención de fraudes a través de la ingeniería social. Incluye contenidos teóricos, vídeos, ejercicios y juegos interactivos.

Usuario: Persona que utiliza la plataforma, con distintos niveles de acceso y permisos:

- **Administrador:** Persona encargada de gestionar la plataforma, con capacidad para modificar usuarios, administrar módulos educativos.

- **Usuario Estándar:** Persona que accede a los módulos de formación, consulta sus resultados y progreso.

Módulo de Capacitación: Unidad educativa dedicada a un tipo de fraude específico.

Registro de Avance: Es el progreso del usuario a través de los módulos.

4.4. Análisis de Requerimientos

4.4.1. Requerimientos Funcionales

RF01 – Login: Permitir a los usuarios iniciar sesión con correo y contraseña.

RF02 – Registro de Usuario: Solicitar al usuario completar datos obligatorios:

- Nombre
- Apellido
- Email
- Teléfono
- Contraseña
- Confirmación de contraseña
- País

- Género
- Fecha de Nacimiento

RF03 – Cambio de contraseña: Permitir a los usuarios cambiar su contraseña proporcionando la actual y la nueva.

RF04 – Ver Perfil: Los usuarios pueden visualizar sus datos registrados.

RF05 – Editar Perfil: Los usuarios pueden modificar su información personal exceptuando el correo electrónico.

RF06 – Cerrar Sesión: Los usuarios pueden cerrar sesión y volver a la pantalla principal de acceso.

RF07 – Acceso a Módulos Educativos: Los usuarios pueden seleccionar y navegar entre los distintos módulos.

RF08 – Seguimiento de Progreso: El sistema registra y muestra el progreso del usuario en cada módulo y etapa.

RF09 – Gestión de Módulos Educativos (Administrador):
El administrador puede crear, editar y eliminar módulos educativos disponibles en la plataforma.

RF10 – Gestión de Etapas de los Módulos (Administrador):
El administrador puede agregar, modificar o eliminar etapas dentro de cada módulo educativo.

RF11 – Gestión de Usuarios (Administrador):
El administrador puede visualizar la lista de usuarios registrados y eliminar usuarios si es necesario.

RF12 – Asignación de Roles (Administrador):
El administrador puede otorgar o revocar permisos de administrador a otros usuarios registrados en el sistema.

4.4.2 Requerimientos No Funcionales

RNF01 – Requerimientos de Usuario para ejecución:

- Hardware: CPU de 1Ghz, 1 GB RAM (32 bits) o 2 GB (64 bits), más de 350 MB de Disco Duro

- Software: Cualquier navegador moderno en Windows, macOS o Linux.

RNF02 – Seguridad: Las contraseñas deben almacenarse cifradas en la base de datos.

RNF03 – Integridad de Datos: Cada usuario tiene un ID único; el correo electrónico es único por usuario.

RNF04 – Escalabilidad: La plataforma debe permitir agregar y editar nuevos módulos sin modificar el código y sin afectar la estabilidad del sistema.

RNF05 – Usabilidad: La interfaz debe ser intuitiva, accesible y responsiva en diferentes dispositivos (PC, tablet, móvil).

4.4.3. Diagrama de Casos de Uso

A continuación, se presenta el diagrama de casos de uso del sistema CAPFIS, que muestra las interacciones principales entre los actores y las funcionalidades definidas.

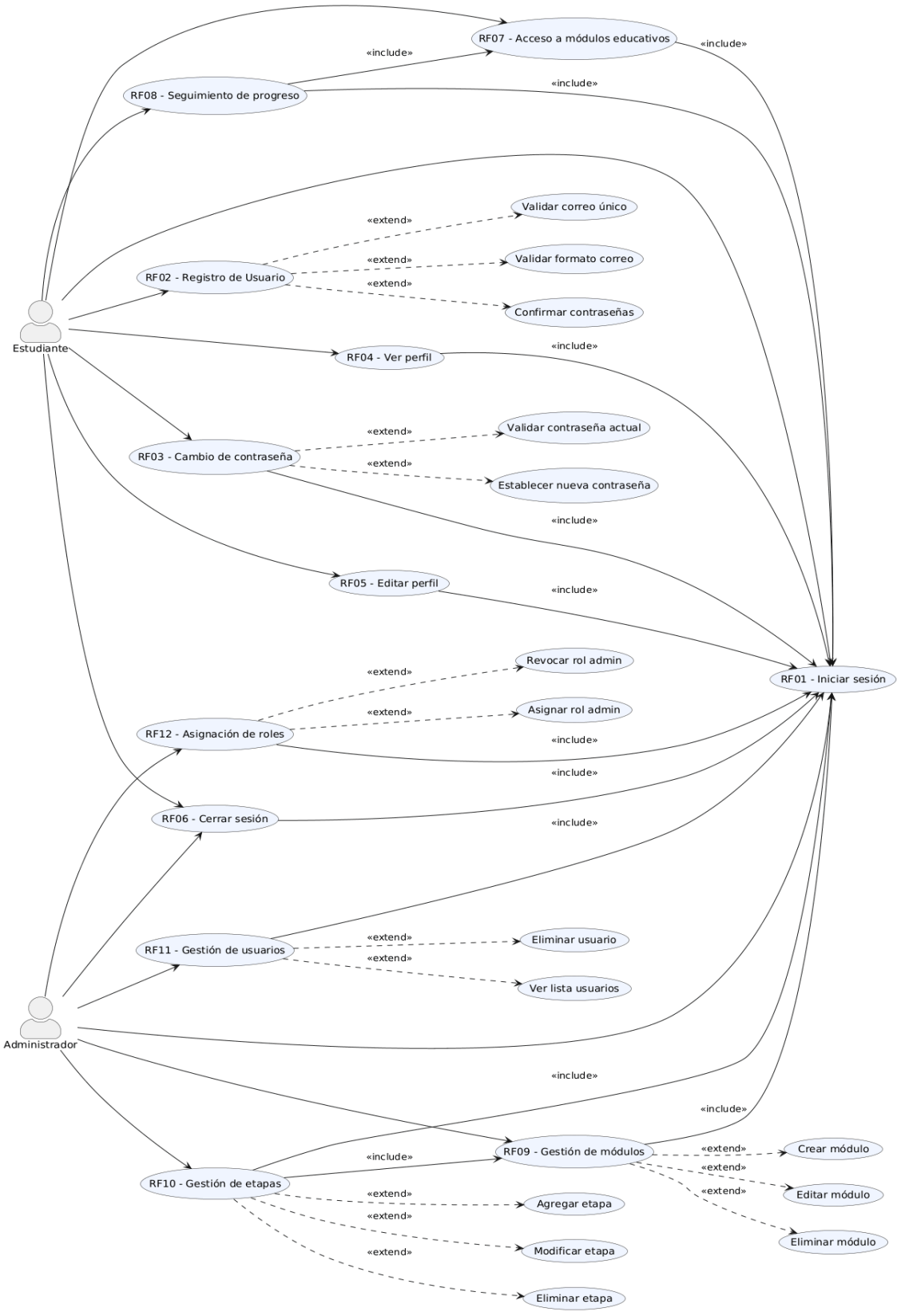


Figura 1 - Diagrama de casos de uso CAPFIS

5. Estado del Arte

Luego de una investigación acerca de trabajos anteriores encontramos varias plataformas dedicadas a la formación en temas de ciberseguridad, destacando PhishLabs y KnowBe4.

PhishLabs ofrece soluciones para detectar y mitigar ataques de phishing y otras amenazas digitales, ayudando a las organizaciones a abordar el abuso de marca y el fraude en línea.[1]

KnowBe4 es la plataforma integrada de capacitación en concientización sobre seguridad y phishing simulado más popular del mundo. Casi 70 000 organizaciones en todo el mundo la utilizan. Ahora existe una forma de gestionar mejor los problemas urgentes de seguridad informática relacionados con la ingeniería social, el spear phishing (suplantación de identidad específica) y los ataques de ransomware.[2]

Se estudiaron y analizaron ambas herramientas y se observó que tanto PhishLabs como KnowBe4 se orientan fundamentalmente al entorno corporativo y requieren licencias de uso, lo que limita su accesibilidad fuera del ámbito empresarial.

Dichas herramientas fueron comparadas con CAPFIS arrojando como resultado que nuestra solución está diseñada para ser utilizada por estudiantes, docentes y cualquier usuario interesado en aprender sobre las técnicas de manipulación empleadas por ciberdelincuentes. Uno de los principales valores de CAPFIS es su flexibilidad para la actualización del contenido. Gracias a su panel de administración, los instructores o responsables del sistema pueden crear, editar o eliminar módulos y etapas de capacitación sin necesidad de modificar el código fuente, lo que facilita su mantenimiento y adaptación a las nuevas formas de ataque. A su vez CAPFIS

[1] Información extraída de https://www.redpoints.com/blog/best-brand-protection-software/ Recuperado el 5 de mayo del 2025.

[2] Información extraída de https://www.knowbe4.com/es/pricing-security-awareness-training#:~:text=KnowBe4%20es%20la%20plataforma%20integrada,todo%20el%20mundo%20la%20utilizan. Recuperado el 6 de mayo del 2025.

integra un sistema de seguimiento del progreso del usuario, que registra automáticamente los avances en cada módulo. De esta forma, el proceso de aprendizaje se convierte en una experiencia dinámica y personalizada, en la que el estudiante puede medir su evolución y reforzar los temas que considere necesarios.

En conclusión, mientras PhishLabs y KnowBe4 se han consolidado como modelos de referencia para el ámbito empresarial, CAPFIS se presenta como una propuesta alternativa, sin licencia paga, totalmente gratuita , orientada al usuario final que quiera tener un primer contacto con el tema y no pretende en ningún momento competir con las mencionadas soluciones.

6. Estudio de Factibilidad

6.1. Introducción

Este análisis permite conocer si el sistema puede implementarse con los recursos disponibles, si resulta sostenible, si cumple con las normas vigentes y si puede ponerse en práctica de manera eficiente. De esta forma, se obtiene una base sólida para tomar decisiones informadas y reducir riesgos durante la ejecución.

6.2. Objetivo

El propósito de este estudio es analizar todos los aspectos del proyecto, identificando y disminuyendo riesgos, y presentando información sobre las distintas dimensiones de factibilidad. El objetivo final es determinar si el desarrollo es viable y conveniente de llevar adelante.

6.3. Análisis de Factibilidad

6.3.1. Factibilidad Técnica

Se eligió la plataforma de desarrollo .NET, ya que facilita la implementación de arquitecturas multicapa además de que se caracteriza por su robustez y seguridad, lo cual la convierte en una alternativa adecuada.

Para la base de datos, se seleccionó SQL Server por su excelente integración con la plataforma .NET, lo que agiliza la comunicación entre los componentes.
Los integrantes del equipo disponen del equipamiento informático necesario para llevar a cabo el desarrollo de la herramienta.

6.3.2. Factibilidad Legal

Teniendo en cuenta la óptica jurídica , no existen impedimentos para el desarrollo ni la aprobación del proyecto.
Se cumple con lo establecido en la Ley N.º 18.331 de Protección de Datos Personales[3] vigente en Uruguay. Durante el registro de usuarios en el sistema, se informa al interesado sobre el uso de sus datos y se solicita la aceptación de los términos.

Se confirma que todos los elementos gráficos utilizados (tipografías, iconos, imágenes, etc.) son de libre uso y no poseen derechos de autor, así como las licencias utilizadas.

6.3.3. Factibilidad Operativa y Humana

El equipo de trabajo posee las competencias necesarias para llevar adelante las tareas. La experiencia académica adquirida a lo largo de la carrera, sumada al acompañamiento del tutor, permitió cumplir con las metas establecidas y obtener un sistema funcional y eficiente.

6.3.4. Factibilidad Económica

El proyecto no requirió de inversiones adicionales, ya que todas las herramientas empleadas son de uso libre y gratuito, y los integrantes disponían del hardware y software necesarios.

6.3.5. Factibilidad Institucional

El proyecto se realiza a solicitud de la Facultad de Ingeniería de la UDE. La universidad comprueba que el equipo de trabajo cumple con los requisitos

[3] Ley N.º 18.331 — *Protección de Datos Personales y Acción de Habeas Data*, Uruguay, 2008. Recuperado el 20 de mayo de 2025, en: https://www.impo.com.uy/bases/leyes/18331-2008

solicitados para la ejecución de este proyecto, a su vez el tutor designado es parte del proyecto, apoyando a la largo de todo el proceso.

6.4. Conclusión

Luego de haber hecho un análisis de las distintas dimensiones enunciadas anteriormente se determinó que la ejecución del proyecto era factible.

7. Plan de Riesgos

7.1. Introducción

El objetivo principal es identificar los riesgos que puedan surgir en las diferentes etapas del proyecto, evaluarlos en función de su impacto y probabilidad, y diseñar medidas de mitigación para los más críticos. Asimismo, se describen aquellos riesgos que representan mayor amenaza, junto con los planes de contingencia y mitigación que permitirán darles respuesta.

7.2. Criterios

Los riesgos se evalúan teniendo en cuenta dos aspectos: Impacto y Probabilidad. Ambos permiten estimar el nivel de riesgo y su efecto en el desarrollo del proyecto. Los mismos fueron identificados a partir de la consulta a los integrantes del equipo conjuntamente con el tutor.

7.3. Cálculo de Riesgos

Para la probabilidad de ocurrencia, se aplica una escala del 1 al 5, donde 1 representa que el evento “casi nunca sucederá” y 5 indica que “es muy probable que suceda”.
Para el impacto utilizamos igual una escala del 1 al 5, siendo que 1 equivale a una afectación de hasta 5 horas (muy bajo) y 5 corresponde a una afectación aproximada de 20 horas (muy alto).
El nivel de riesgo se obtiene multiplicando el impacto por la probabilidad, lo cual nos permite determinar su prioridad de atención dentro del plan de gestión de riesgos.

ID	Tipo	Riesgo	Impacto	Probabilidad	Total (I * P)
R01	Humano	Falta de un integrante por enfermedad o causa de fuerza mayor	4	3	12
R02	Gestión	Requerimientos mal especificados	5	3	15
R03	Gestión	Incumplimiento de los plazos establecidos	4	3	12
R04	Técnico	Limitado dominio de algunas herramientas o técnicas del proyecto	3	3	9
R05	Técnico	Incompatibilidad del software con el desarrollo	3	3	9
R06	Técnico	Pérdida de documentación o del código fuente	3	2	6
R07	Gestión	Incumplimiento de tareas semanales por parte de los miembros	3	1	3
R08	Humano	Abandono definitivo de un integrante del grupo	5	1	5
R9	Técnico	Actualizaciones de las versiones de herramientas utilizadas	3	1	3

7.4. Plan de Mitigación y Contingencia de riesgos

A continuación se detallan las medidas propuestas para los riesgos identificados. Se dará prioridad a aquellos con un puntaje mayor o igual a 6, sin dejar de lado los de menor puntaje, ya que durante el desarrollo del proyecto pueden variar en criticidad.

R01 – Falta de un integrante por enfermedad o fuerza mayor

- Mitigación: el otro integrante deberá asumir temporalmente las tareas pendientes, ajustando así su carga horaria con horas adicionales.
- Contingencia: Si esto no resulta suficiente, se destinarán horas extra en fines de semana o feriados.

R02 – Requerimientos mal especificados

- Mitigación: Los requerimientos deberán documentarse con precisión.
- Contingencia: En caso de inconsistencias, los requerimientos serán revisados junto al equipo y posteriormente validados con el tutor.

R03 – Incumplimiento de los plazos establecidos

- Mitigación: Se debe respetar el cronograma establecido, procurando en lo posible no cambiar fechas y cumplir con el plazo de entrega.
- Contingencia: Se hará un seguimiento semana a semana del cronograma y si es necesario, se redistribuirán tareas y se reorganizará el calendario para garantizar su cumplimiento.

R04 – Limitado dominio de técnicas o herramientas

- Mitigación: Incorporar una fase de investigación y práctica previa que permita dominar las herramientas y las tecnologías usadas.
- Contingencia: En caso de persistir dificultades, se recurrirá al tutor y al material académico disponible en la UDE.

R05 – Incompatibilidad de software

- Mitigación: Evaluar previamente las herramientas necesarias y confirmar su compatibilidad antes de iniciar el desarrollo.
- Contingencia: De surgir inconvenientes, se decidirá en conjunto la herramienta alternativa, con la orientación del tutor.

R06 – Pérdida de documentación o código

- Mitigación:
    - Guardar la documentación en una carpeta compartida (Google Drive).
    - Subir el código semanalmente a GitHub.
    - Respaldar de manera local en los equipos de los integrantes del proyecto
- Contingencia: Ante una pérdida, se restaurará primero desde GitHub y si es necesario, se usará la copia local.

8. Plan de Proyecto

8.1. Introducción

En este documento se definen las fases principales, los roles de los participantes, el modelo de desarrollo adoptado y la planificación a seguir. Cada uno de estos aspectos será monitoreado a lo largo del proceso para mantener una visión clara y ordenada del trabajo.

8.2. Objetivo y Alcance

El propósito de este plan es servir como guía para la correcta ejecución del proyecto CAPFIS. Se establecen los lineamientos que garanticen el cumplimiento de los requisitos y la entrega de un producto que responda a las expectativas de la Universidad, en calidad de cliente.

8.3. Modelo de Desarrollo

Optamos por trabajar con la metodología Scrum debido a que nos permite organizar las tareas en ciclos cortos (Sprints) que facilitan la adaptación, el control del progreso y la incorporación de mejoras de forma iterativa. A su vez, posibilita que el tutor pueda evaluar los avances y sugerir correcciones en cada etapa.

8.4. Etapas del Proyecto

8.4.1. Etapa de Análisis

Se identificaron los objetivos y requerimientos del proyecto. Se establecieron las necesidades funcionales y no funcionales que guiaron el resto del proceso. Esta etapa permitió construir una visión clara del producto a desarrollar.

8.4.2. Etapa de Diseño y Planificación

Se elaboró el modelo arquitectónico del sistema, los diagramas conceptuales y la planificación temporal del proyecto. Se definieron las tareas mediante el diagrama de Gantt, y se seleccionaron las herramientas y tecnologías a usar.

8.4.3. Etapa de Definición

Se formalizaron los planes de gestión del proyecto: riesgos, calidad y configuración.

8.4.4. Etapas de Desarrollo e Implementación

Es el núcleo operativo del proyecto, donde se desarrollaron los módulos funcionales del sistema CAPFIS. Durante esta etapa se ejecutaron los Sprints de acuerdo al cronograma y se implementaron las historias de usuario. La coordinación constante, la revisión de código y la adaptación a los cambios fueron claves para la entrega final del producto.

8.4.5. Etapa de testing y finalización

Se verificó y se validó todas las funcionalidades del sistema . Se documentaron los casos de prueba, se corrigieron incidencias y se realizaron auditorías de código y documentación.

8.5. Políticas del Proyecto

Con el fin de asegurar la coordinación efectiva del equipo y la correcta gestión de los procesos del proyecto CAPFIS, se establecieron políticas internas que guían la comunicación, la toma de decisiones y el control de cambios.
Estas políticas se aplicaron durante todo el ciclo de vida del proyecto y fueron esenciales para mantener la transparencia, la trazabilidad y la alineación entre los integrantes y el tutor académico.

8.5.1. Política de Comunicación y Reuniones

Con el objetivo de garantizar una comunicación constante y asertiva entre los integrantes del equipo y el tutor académico, se estableció la siguiente política de comunicación:

Lineamientos definidos:

- Reuniones semanales del equipo: Se determinó la realización de reuniones semanales, presenciales o virtuales, con el propósito de coordinar tareas, ver avances, o analizar obstáculos y el trabajo en conjunto.

- Reuniones de seguimiento con el tutor: Se estableció la política de efectuar reuniones periódicas con el tutor académico, orientadas a presentar avances, recibir observaciones y validar entregables intermedios. Cada reunión debía ser registrada mediante un acta.
- Comunicación diaria: Se definió el uso de WhatsApp para para la coordinación de tareas, resolución de dudas o emergencias. Asimismo, el uso del tablero Trello como herramienta de gestión visual para el seguimiento del avance de las actividades.

8.5.2. Política de Gestión de Cambios

Cualquier cambio que se hiciere tanto en la documentación, funcionalidades o código del proyecto debía ser consensuado entre ambos integrantes y registrado formalmente en las actas de reunión.

Los pasos definidos fueron los siguientes:

1. Solicitud de cambio: Cuando se detecte una mejora o ajuste necesario por parte del equipo o el tutor.
2. Evaluación de impacto: Se realiza un análisis de si es viable o no teniendo en cuenta sobre todo la no afectación del cronograma.
3. Aprobación y registro: Una vez aprobado se documenta en acta.
4. Implementación y verificación: Se procede a ejecutar el cambio y hacer las validaciones correspondientes.

8.5.2.1. Ejemplo de cambio importante:

Una vez comenzado el desarrollo de la aplicación decidimos cambiar el motor de base de datos de MySQL a SQL Server fundamentalmente por cuestiones de compatibilidad. Este cambio fue revisado y aprobado por el procedimiento anterior, quedando asentado en acta que se adjunta en el Anexo.

Razón del cambio:
En las primeras pruebas que se realizaron con Pomelo.Entity Framework Core.MySql se encontraron incompatibilidades con algunas características del ORM

Entity Framework Core 9.0.0, así como dificultades en la configuración. Además que SQL Server, nos proporcionaba mejor integración nativa con Visual Studio 2022.

8.5.3. Política de Calidad y Documentación

Para asegurar la coherencia y calidad del proyecto, se establecieron las siguientes pautas:

- Todos los documentos técnicos se redactaron bajo el formato **APA 7.ª edición** así como las normas **IEEE** aplicables.
- Cada entregable fue revisado por ambos integrantes antes de su presentación.
- Se aplicaron revisiones iterativas del código fuente .

Esta política garantiza uniformidad en la presentación de la información.

8.5.4. Política de Respaldo y Seguridad

Con el objetivo de proteger el código, la documentación y los recursos multimedia se implementaron las siguientes medidas de respaldo:

- El repositorio principal se mantuvo en GitHub.
- La documentación se almacena en Google Drive, con permisos restringidos al equipo y tutor.
- Se realizaron copias de seguridad semanales para evitar pérdida de información.

8.6. Equipo de Trabajo y Responsabilidades

El equipo del proyecto está integrado por:

- **Diego Balbis**

- **Dayron Muñiz**

Roles

Teniendo en cuenta que se trata de un equipo con sólo dos participantes, ambos asumimos todos los roles de forma simultánea, así de esta forma, cada miembro trabajó en tareas de gestión, análisis, desarrollo y testing, garantizando un equilibrio en cada una de las áreas.

Rol	Responsable	Descripción
Product Owner	Diego Balbis - Dayron Muñiz	Administra el Product Backlog y garantiza alineación con los objetivos del cliente.
Scrum Master	Diego Balbis - Dayron Muñiz	Facilita reuniones, elimina impedimentos y promueve la mejora continua.
Equipo de Desarrollo	Diego Balbis - Dayron Muñiz	Implementa las funcionalidades y entrega los incrementos de producto.
Equipo de Testing	Diego Balbis - Dayron Muñiz	Ejecuta pruebas funcionales y de aceptación en cada Sprint.

Stakeholders

Corresponden a las partes interesadas del proyecto — principalmente la Universidad, el tutor y los usuarios finales.

Participan en la revisión de entregas (Sprint Review) aportando retroalimentación continua.

8.7. Seguimiento y control del proyecto CAPFIS.

Para mantener un control sistemático del progreso del proyecto y asegurar la consecución de los objetivos fijados, se realizaron distintas instancias de seguimiento, revisión y retroalimentación.

Estas actividades se ejecutaron teniendo como referencia a la Política de Comunicación definida con anterioridad.

8.7.1. Reuniones de seguimiento con el Tutor

Durante toda la realización del proyecto se llevaron a cabo reuniones de control con el tutor, con una frecuencia promedio de cada 15 días.
En estas instancias se presentaron los avances de cada fase, el cumplimiento del cronograma, así como de las tarjetas de Trello, recibiendo de su parte las validaciones u observaciones correspondientes, lo cual nos permitió corregir oportunamente cualquier ítem necesario.
Cada reunión se documentó mediante un acta, las mismas se incluyen en el anexo del documento.

8.7.2. Reuniones semanales del Equipo

Sumado a los encuentros con el tutor, el equipo realizó reuniones aproximadamente dos veces por semana.
En las mismas se revisaron las tareas en curso, las completadas, asignar nuevas responsabilidades y para el trabajo en conjunto.

La frecuencia de estas reuniones permitió sostener un ritmo de trabajo constante y ordenado, así como la revisión continua del progreso.

8.7.3. Comunicación Diaria y Herramientas de Control

Durante todo el proyecto, el equipo de trabajo estuvo en comunicación permanente mediante WhatsApp, por la inmediatez que nos proporciona.

Sumado al uso complementario de Trello que nos permitió llevar un control visual de las tareas, registrar avances y mantener actualizado el estado del proyecto.

8.7.4. Reuniones de Entrega Final

En la etapa final del proyecto se llevaron a cabo reuniones específicas para la revisión integral de la documentación, verificación del código, validación de la aplicación y armado de la entrega final.

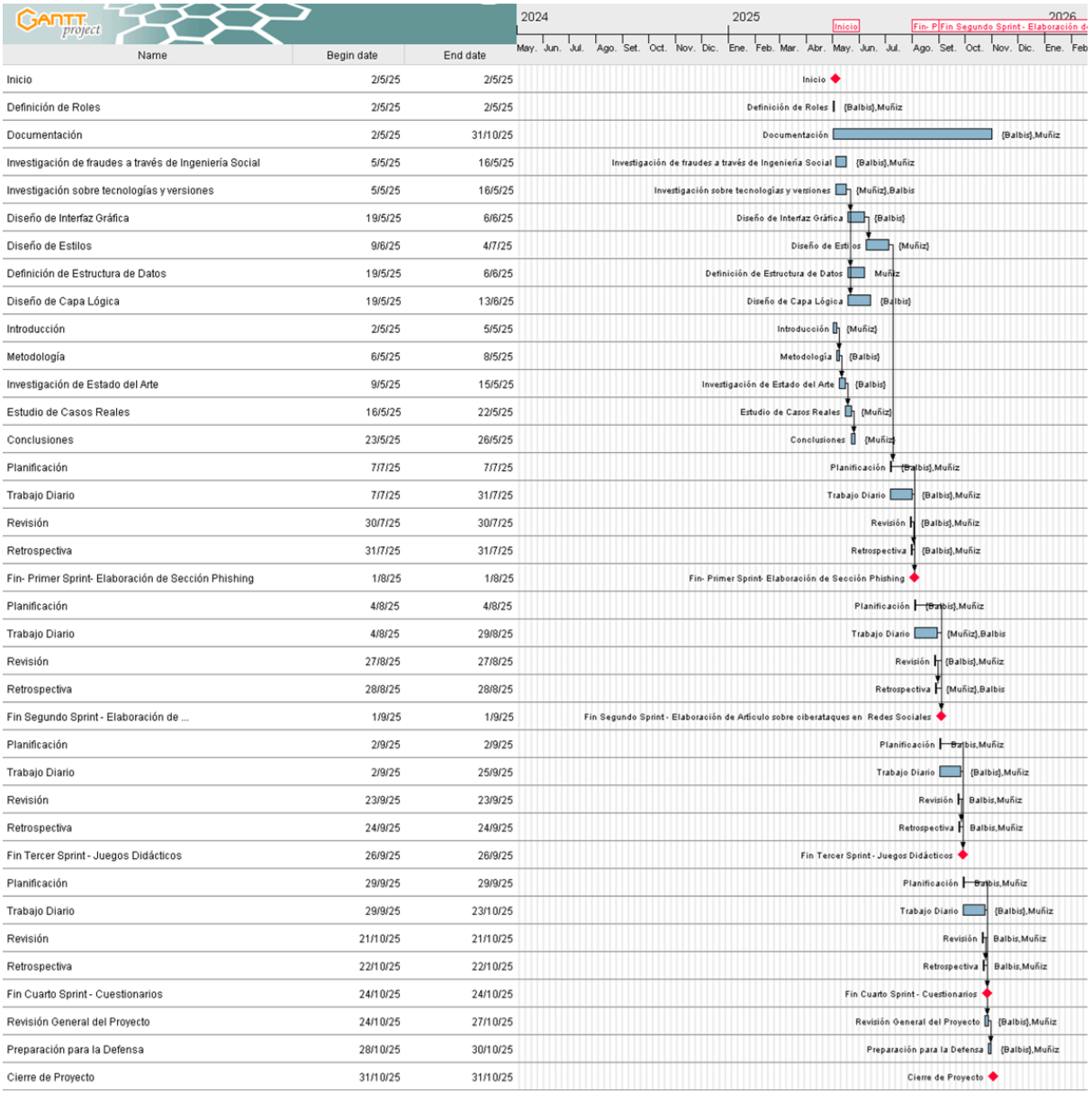
De estas reuniones no se realizaron actas más bien su cometido fue para validar los últimos ajustes y confirmar la conformidad del producto previo a su presentación y entrega a la Universidad.

8.8. Cronograma de Proyecto

Se elaboró un cronograma detallado con el fin de disponer de una visión global de todas tareas necesarias para la ejecución del proyecto así como del tiempo estimado para cada una. Este instrumento resultó fundamental para dar seguimiento a los avances, verificar el cumplimiento de las tareas y anticipar y prever posibles ajustes en los plazos de ejecución.

8.8.1. Gantt

Diagrama de Gantt donde se representan las tareas programadas y su distribución en los distintos sprints.



Name	Begin date	End date
Inicio	2/5/25	2/5/25
Definición de Roles	2/5/25	2/5/25
Documentación	2/5/25	31/10/25
Investigación de fraudes a través de Ingeniería Social	5/5/25	16/5/25
Investigación sobre tecnologías y versiones	5/5/25	16/5/25
Diseño de Interfaz Gráfica	19/5/25	6/6/25
Diseño de Estilos	9/6/25	4/7/25
Definición de Estructura de Datos	19/5/25	6/6/25
Diseño de Capa Lógica	19/5/25	13/6/25
Introducción	2/5/25	5/5/25
Metodología	6/5/25	8/5/25
Investigación de Estado del Arte	9/5/25	15/5/25
Estudio de Casos Reales	16/5/25	22/5/25
Conclusiones	23/5/25	26/5/25
Planificación	7/7/25	7/7/25
Trabajo Diario	7/7/25	31/7/25
Revisión	30/7/25	30/7/25
Retrospectiva	31/7/25	31/7/25
Fin- Primer Sprint- Elaboración de Sección Phishing	1/8/25	1/8/25
Planificación	4/8/25	4/8/25
Trabajo Diario	4/8/25	29/8/25
Revisión	27/8/25	27/8/25
Retrospectiva	28/8/25	28/8/25
Fin Segundo Sprint - Elaboración de ...	1/9/25	1/9/25
Planificación	2/9/25	2/9/25
Trabajo Diario	2/9/25	25/9/25
Revisión	23/9/25	23/9/25
Retrospectiva	24/9/25	24/9/25
Fin Tercer Sprint - Juegos Didácticos	26/9/25	26/9/25
Planificación	29/9/25	29/9/25
Trabajo Diario	29/9/25	23/10/25
Revisión	21/10/25	21/10/25
Retrospectiva	22/10/25	22/10/25
Fin Cuarto Sprint - Cuestionarios	24/10/25	24/10/25
Revisión General del Proyecto	24/10/25	27/10/25
Preparación para la Defensa	28/10/25	30/10/25
Cierre de Proyecto	31/10/25	31/10/25

Figura 2 – Cronograma CAPFIS

8.9. Conclusión

En síntesis, con este Plan de Proyecto establecemos de forma organizada las fases de trabajo, el modelo utilizado para la ejecución y gestión, así como los mecanismos de seguimiento y control. De esta forma tuvimos un panorama claro del proceso, reduciendo lo más posible los errores y riesgos que puedan surgir durante la ejecución.

9. Arquitectura

La arquitectura del sistema CAPFIS se diseñó con el objetivo de garantizar modularidad, mantenibilidad y escalabilidad, siguiendo con las buenas prácticas que establecen los principios de la ingeniería de software moderna.
El modelo adoptado responde a una arquitectura en capas, sustentada en el patrón Modelo–Vista-VistaModelo (MVVM), lo que permite una clara separación entre la interfaz de usuario, la lógica de negocio y la gestión de datos.
De este modo se facilita la evolución del sistema, la reutilización de componentes y la incorporación de nuevas funcionalidades con bajo impacto en el resto de la aplicación.

9.1. Arquitectura lógica

La lógica de la aplicación se estructura bajo una arquitectura en capas siendo así que la separación entre la capa de presentación, la capa de negocio y la capa de datos permite trabajar con distintos niveles de abstracción, logrando una mayor modularidad, escalabilidad y tolerancia a fallos.

Cada capa cumple un rol definido:

- Capa de Presentación: Es la que ve directamente el con el usuario mediante interfaces web construidas con Razor Views y Bootstrap.
- Capa de Negocio: Contiene las reglas lógicas del sistema, implementadas en C# sobre la plataforma .NET.
- Capa de Datos: Gestiona la persistencia de información a través de SQL Server y Entity Framework Core.

9.2. Arquitectura Física

Desde el punto de vista físico, CAPFIS se implementa bajo un esquema cliente-servidor:

- Cliente: Accede a la aplicación mediante un navegador web compatible (Chrome, Edge, Firefox).
- Servidor de Aplicaciones: Aloja la lógica de negocio y coordina la interacción entre la interfaz y la base de datos.
- Servidor de Base de Datos: Gestiona la persistencia y consistencia de la información mediante Microsoft SQL Server 2022.

Esta distribución asegura rendimiento, seguridad y posibilidad de escalar el sistema en entornos productivos.

9.3. Componentes Principales

- Servidor de Aplicaciones: Entorno donde se despliega la aplicación web junto con la lógica de negocio y los servicios. Procesa las reglas del sistema y gestiona la comunicación entre capas.
- Servidor de Base de Datos: Encargado de almacenar y mantener actualizado los datos de la aplicación dígase módulos, usuarios, etc, asegurando la integridad y disponibilidad de la información.

Los dos componentes se integran haciendo uso de Entity Framework Core, que cumple la función de intermediario entre la aplicación y la base de datos.

9.4. Patrones de diseño

El desarrollo del sistema se basa en el patrón de diseño MVVM (Modelo-Vista-VistaModelo). Este modelo permite una separación efectiva entre la capa de presentación y la lógica de negocio, al incorporar una capa intermedia —la VistaModelo— que actúa como enlace entre ambas. Gracias a esta estructura, la aplicación logra una mayor modularidad, facilidad de prueba y mantenimiento, permitiendo actualizar la interfaz o la lógica sin afectar el resto del sistema. Este enfoque promueve una arquitectura más limpia, escalable y flexible, facilitando tanto el desarrollo actual como futuras evoluciones del software.

MVVM consideramos que es más conveniente que MVC porque permite que cada página tenga su propio modelo de vista, reflejando el flujo natural de los módulos educativos, con data binding directo y manejo de estado local, evitando controladores globales y rutas complejas que MVC requeriría.

9.5. Diagrama de Arquitectura

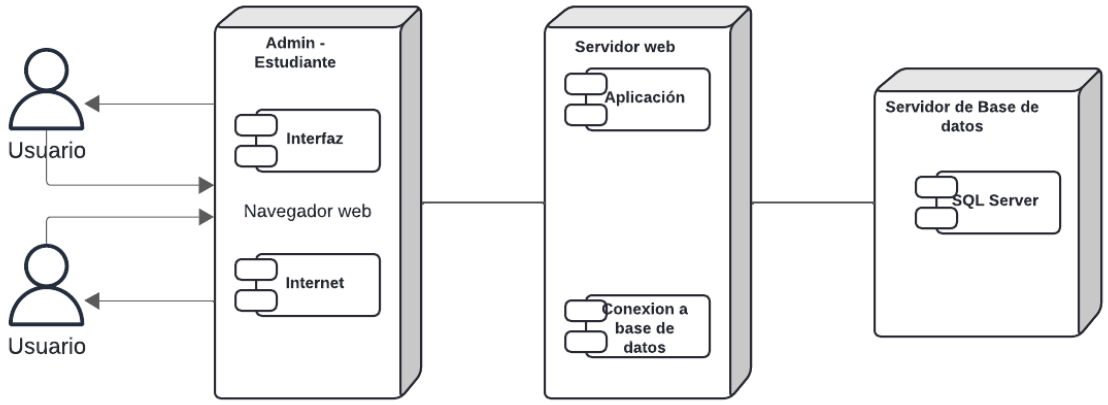


Figura 3 – Diagrama de Arquitectura del Sistema CAPFIS

En el diagrama se representa la arquitectura general del sistema CAPFIS, estructurada bajo un modelo cliente–servidor.

Los usuarios (administradores y estudiantes) acceden a la aplicación mediante un navegador web, que se comunica con el servidor donde se ejecuta la lógica del sistema.

Este, a su vez, se conecta al servidor de base de datos **SQL Server**, responsable del almacenamiento y la gestión de la información.

9.6. Diagrama Conceptual de Clases

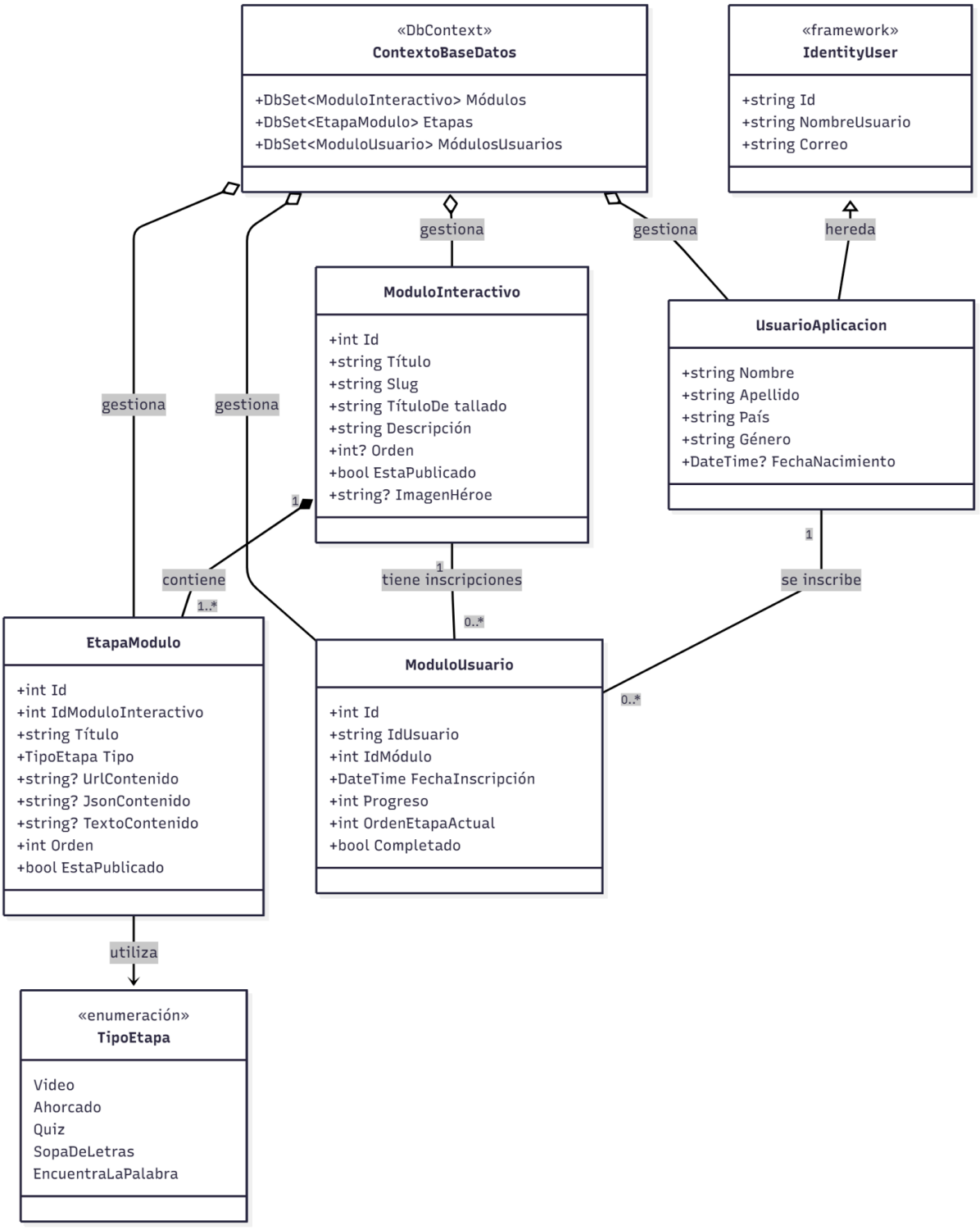


Figura 4 – Diagrama Conceptual de Clases del Sistema CAPFIS

En este diagrama se muestran las principales clases del sistema CAPFIS y sus relaciones.

9.7. Tecnologías Utilizadas

La ejecución de la aplicación se basó en un grupo de tecnologías y herramientas seleccionadas sobre todo por su robustez, compatibilidad con el entorno .NET y facilidad de mantenimiento.

En la siguiente tabla se resumen las principales tecnologías usadas, sus versiones y una breve descripción de sus funciones en la aplicación.

Tecnología	**Versión**	**Descripción y uso en CAPFIS**
.NET	9.0.4	Plataforma de desarrollo moderna que soporta el patrón MVVM. En CAPFIS estructura la aplicación y permite escalabilidad.
C#	13	Lenguaje de programación principal, utilizado para implementar la lógica de negocio.
Razor Views		Motor de renderizado de ASP.NET que combina HTML y C#, usado para generar vistas dinámicas.
Bootstrap	5.3.3	Framework CSS para diseño responsive y moderno de la interfaz web.
Microsoft SQL Server	2022	Sistema de gestión de bases de datos que almacena la información del sistema con alta integridad.
Entity Framework Core	9.0.0	Framework de *Mapeo Objeto-Relacional* (ORM) que conecta las clases del modelo con las tablas de la base de datos. Simplifica las operaciones básicas de datos (crear, leer, actualizar y eliminar – CRUD) y mantiene la coherencia entre la aplicación y la base de datos.
ASP.NET Core Identity		Framework para autenticación y gestión de

		usuarios y roles, garantizando seguridad y cifrado.
Git	2.51.1	Sistema de control de versiones para el código fuente y coordinación del trabajo en equipo.
Google Drive		Plataforma de almacenamiento en la nube utilizada para documentación y respaldos.
GanttProject	3.3.3316	Herramienta de planificación usada para elaborar el diagrama de Gantt y gestionar las tareas del proyecto.

10. Plan de Calidad

10.1. Introducción

En este apartado se definen los lineamientos con el objetivo de asegurar la calidad tanto del producto como del proceso.

A lo largo de la ejecución del proyecto se hizo énfasis en mantener un control constante sobre la documentación, el código fuente y los entregables intermedios, garantizando que cumplieran con los requerimientos definidos.

10.2. Objetivo y Alcance

El objetivo de este plan es establecer las prácticas que permitieron asegurar la calidad del software desde el inicio hasta el cierre del proyecto.

El mismo abarca todos los activos del proyecto, dígase documentación, código, archivos multimedia así como las distintas etapas.

Estas acciones se llevaron a cabo bajo la autoevaluación continua del equipo en conjunto con la revisión quincenal del tutor académico.

10.3. Estándares y Referencias

Se tomaron en cuenta los estándares de las normas IEEE 730 de Aseguramiento de Calidad de Software y IEEE 1058 de Gestión de Proyectos de Software, aplicadas al plano académico.

Por otra parte tomamos en cuenta las prácticas de la metodología Scrum así como el patrón arquitectónico MVVM (Modelo–Vista–VistaModelo), implementado nativamente por el entorno ASP.NET Core 9.

10.3.1 Estándares de lenguaje

El desarrollo de este sistema se realiza en .Net (C#) utilizando tecnologías y frameworks que se complementan y dan como resultado un sistema robusto, seguro y eficiente.

Se siguieron parcialmente los estándares de C# recomendados, incluyendo convenciones y reglas de nomenclatura de identificadores.

También se implementaron los siguientes ejemplos:

Clases y métodos: PascalCase
Variables locales y parámetros de métodos: camelCase
Constantes: PascalCase

La adopción de estas convenciones permite que el código sea más legible, mantenible y consistente, facilitando la colaboración entre los miembros del equipo.

10.4. Enfoque de Calidad

Para el aseguramiento de calidad se tuvo en cuenta un enfoque preventivo y colaborativo, basado en:

- Revisiones cruzadas entre ambos participantes del equipo en cada etapa.
- Validación funcional temprana, probando siempre cada módulo al finalizar su desarrollo.
- Control del avance a través de la herramienta de Trello
- Control de versiones en GitHub, utilizando ramas para desarrollo y commits frecuentes con descripciones claras.

- Respaldo documental en Google Drive, garantizando siempre su disponibilidad.
- Reuniones cada quince días con el tutor, donde se revisaron avances, incidencias así como correcciones propuestas.

10.5. Actividades de Aseguramiento

Durante el proyecto se llevaron a cabo las siguientes actividades concretas:

- Revisión de código: ambos integrantes verificaban la legibilidad, estructura y buenas prácticas de la programación.
- Revisión documental: se controló el formato, ortografía y coherencia técnica y conceptual.
- Pruebas funcionales intermedias: al concluir cada Sprint se ejecutaron pruebas de las funcionalidades implementadas.
- Retroalimentación externa: el tutor verificó de forma continua los avances en cada una de los activos del proyecto.

10.6. Control de Cambios y Trazabilidad

Cada cambio en el proyecto fue documentado y trazado a través de diferentes medios:

- GitHub para código y versiones de software.
- Google Drive para documentos, actas y respaldos.
- Trello para seguimiento de tareas y control de incidencias.
- Actas de reunión donde se registraron los acuerdos y ajustes.

10.7. Criterios de Aceptación

Cada entregable parcial fue considerado aceptado cuando cumplió con:

- Los requerimientos definidos en los RF y HU correspondientes.

- Las validaciones funcionales sin errores críticos.
- La aprobación de ambos integrantes y del tutor en revisión final.
- La correcta integración con el resto del sistema sin romper funcionalidades previas.

10.8. Conclusión

Gracias a los lineamientos trazados en este apartado la aplicación alcanzó una calidad final adecuada para su implementación y presentación ante la Universidad.

11. Plan de Testing

11.1. Introducción

El Plan de Testing se enfoca en la elaboración de pruebas para el cumplimiento de los requerimientos definidos las mismas se integraron dentro de cada Sprint, siguiendo el enfoque iterativo de Scrum, lo que permitió detectar errores a tiempo y asegurar una entrega estable y funcional.

11.2. Objetivo

El objetivo principal del testing fue comprobar el correcto funcionamiento de cada uno de los módulos y etapas del sistema, garantizar la integridad de los datos almacenados y validar que las funcionalidades cumplieran las expectativas estipuladas.
El proceso de pruebas también se enfocó en evaluar el rendimiento, la seguridad y la navegabilidad general de la aplicación web.

11.3. Alcance

Las pruebas abarcan todos los módulos implementados:

- **Módulo de autenticación:** login, registro, cambio de contraseña, cierre de sesión.
- **Gestión de usuario y perfil:** edición de datos, validaciones de campos.

- **Módulos educativos:** navegación entre etapas, actividades interactivas (ahorcado, sopa de letras, encuentra la palabra) y registro del progreso.
- **Gestión administrativa:**creación y edición de módulos y etapas así como la administración de usuarios y asignación de roles.

11.4. Estrategia General

La estrategia de pruebas se basó en cinco etapas complementarias:

1. **Pruebas unitarias:** revisión del código en Visual Studio.
2. **Pruebas de integración:** comprobación del sistema, verificando la correcta comunicación de sus capas.
3. **Pruebas funcionales:** validación de todos los requerimientos definidos en el análisis.
4. **Pruebas de regresión:** validación de la aplicación ante cada actualización ya fuera debido a correcciones o mejoras.
5. **Pruebas de aceptación:** validación final del sistema por parte del equipo y tutor académico.

11.5. Herramientas y Entorno de Pruebas

El testing se realizó en el entorno de desarrollo local, utilizando:

- Visual Studio 2022 como entorno de ejecución.
- SQL Server 2022 para la gestión de la base de datos.
- Navegadores Edge y Chrome para verificar compatibilidad visual.
- Trello para seguimiento del estado de las pruebas.
- GitHub para registrar correcciones y versiones.
- Google Drive para documentar.

11.6. Casos de Prueba

Los casos de prueba se desarrollaron a partir de los requerimientos funcionales y se documentaron detalladamente en el Anexo de Casos de Prueba.

11.7. Conclusión

El proceso de testing validó que CAPFIS cumple los objetivos propuestos y que las funcionalidades desarrolladas son estables, seguras y en concordancia con los requerimientos establecidos.

Las pruebas confirmaron la integridad general del sistema.

12. Plan de Métricas

12.1. Métricas del Producto

A lo largo de toda la ejecución del proyecto, se aplicó un plan de métricas orientado a supervisar la calidad del producto de software.

El objetivo fue cuantificar factores relacionados con mantenibilidad, complejidad, acoplamiento, herencia y tamaño del código, identificando tendencias y posibles focos de mejora.

Las métricas de producto se calcularon mediante un análisis de datos obtenidos a través de Visual Studio 2022 así como los cambios en el tiempo del repositorio en GitHub.

De esta forma se combinó la visión cuantitativa del producto con la evolución temporal del desarrollo.

Lo que se persiguió con el análisis de estas métricas fue tener durante todo el proceso de desarrollo valores que estuvieran en el mejor rango posible, los cuales describimos a continuación:

12.1.1. Índice de Mantenibilidad

El Índice de Mantenibilidad (MI) que cuantifica la facilidad de mantenimiento y comprensión de un sistema de software. Se calcula a partir de métricas como la complejidad ciclomática, las líneas de código y el volumen Halstead, expresándose en una escala de 0 a 100, donde valores mayores indican mejor mantenibilidad.

Rango	Interpretación

0-20	Difícil de mantener. Código con alta complejidad, propenso a errores y refactorizaciones frecuentes.
10 – 20	Mantenibilidad moderada. Aceptable, aunque requiere atención en futuras ampliaciones.
20 – 30	Buena mantenibilidad. Estructura clara y con bajo riesgo de deuda técnica.
30 – 40	Muy buena mantenibilidad. Código bien organizado, con diseño estable y reutilizable.
40 – 100	Excelente capacidad de mantenimiento. Código fácilmente comprensible y adaptable a cambios.

El proyecto obtuvo un valor de 93,67 puntos, lo cual corresponde al rango de excelente mantenibilidad.

12.1.2. Complejidad Ciclomática

Esta métrica de software mide el número de caminos independientes dentro del flujo de control de un método o función, cuando mayor sea el valor significa que más pruebas y esfuerzo de mantenimiento se requieren.

Rango	**Interpretación**
1–10	Baja – fácil de probar y mantener
11–20	Media – necesita mayor revisión
> 20	Alta – riesgo de errores, refactor recomendado

Promedio global : 3,57 → baja complejidad.

Evolución real:

- Sprint 1: 2,03
- Sprint 2: 3,99
- Sprint 3: 4,84
- Sprint 4: 5,64

A partir de los datos obtenidos de Visual Studio 2022 observamos que la métrica aumenta su valor de forma sostenida hasta el Sprint 3, coincidiendo con la

incorporación de los módulos de juegos didácticos ya que demandaron mayor lógica condicional.
Luego en el Sprint 4 el crecimiento se atenúa, reflejando una fase de estabilización y corrección, entonces se evidencia que el proyecto logró un crecimiento controlado de la complejidad, manteniéndose en niveles bajos pese al incremento de funcionalidades.

12.1.3. Profundidad de Herencia

Esta métrica indica cuántos niveles existen desde una clase padre hasta sus clases hijas.

Cuanto mayor es la profundidad, más compleja es la estructura, ya que las clases heredan más atributos, más métodos y funciones por tanto se vuelven más difíciles de mantener.

Valores bajos, por el contrario, reducen el acoplamiento y facilitan la comprensión del diseño.

Rango	**Interpretación**
1–2	Jerarquía simple, fácil de mantener
3–5	Herencia moderada; aceptable en sistemas medianos
> 5	Herencia profunda; riesgo de alto acoplamiento y menor flexibilidad

El resultado arrojado en este análisis fue de un promedio global: 2,91 lo que se traduce en jerarquías cortas y simples. Siendo así que este patrón favorece la mantenibilidad y la claridad del modelo orientado a objetos, manteniendo una estructura comprensible y estable.

12.1.4. Acoplamiento de Clases

El Acoplamiento de Clases mide el número de dependencias directas entre clases, es decir, cuántas clases utilizan o son utilizadas por otra.

Un acoplamiento bajo implica mayor modularidad, independencia y facilidad para el testing unitario, mientras que valores altos pueden generar errores en cascada y menor capacidad de reutilización.

El valor promedio fue obtenido mediante el mismo análisis estático sobre el código fuente de Visual Studio 2022, que contabiliza las referencias y dependencias entre clases, controladores y servicios del sistema, dicho valor fue de 3,42 lo cual es un acoplamiento bajo y saludable.

Rango	**Interpretación**
1–5	Bajo acoplamiento — diseño modular y mantenible
6-10	Acoplamiento medio — requiere control en ampliaciones
> 10	Acoplamiento alto — riesgo de dependencia excesiva

12.1.5. Análisis de métricas

La siguiente tabla muestra la evolución y promedio de métricas por sprint:

Sprint	**Complejidad Ciclomática promedio por archivo**	**Líneas de código (aprox.)**	**Cantidad de commits**	**Interpretación**
1	2.03	2486	Alta	Construcción de la estructura base y primeros módulos.
2	3.99	4325	Moderada	Ampliación funcional y consolidación de arquitectura.
3	4.84	6215	Alta	Incremento de lógica por implementación de juegos didácticos.
4	5.64	6347	Baja	Etapa de cierre, corrección y optimización final.

– *Líneas de código (aprox.)* corresponde a la cantidad de líneas efectivas del código fuente, excluyendo comentarios y espacios en blanco.

– *Cantidad de commits* indica el número de confirmaciones de cambio registradas en el repositorio Git durante cada sprint.

12.1.6. Conclusión de las Métricas de Producto

El análisis de métricas de producto evidencia que el proyecto CAPFIS mantuvo un equilibrio constante entre crecimiento funcional y calidad interna.

El índice de mantenibilidad alcanzó valores sobresalientes (MI ≈ 93,7) a su vez que la complejidad ciclomática se mantuvo dentro de los niveles recomendados .

La combinación de bajo acoplamiento, jerarquías simples y mantenibilidad alta demuestra que el diseño fue consistente con los principios de ingeniería de software moderna.

En resumen:

1. El código evidencia una alta mantenibilidad y estructura limpia.
2. La complejidad aumentó sólo donde hay valor funcional agregado, sin comprometer la estabilidad.
3. El acoplamiento y la herencia se mantuvieron en niveles deseados.
4. La evolución del sistema fue progresiva coherente con la metodología que se aplicó.

12.2. Métricas de Proceso

Las métricas de proceso permiten analizar el tiempo y el esfuerzo invertido durante el desarrollo del proyecto CAPFIS, reflejando cómo se distribuyó la carga de trabajo y el compromiso del equipo en cada fase.
Estas métricas complementan las métricas de producto al aportar una visión de la gestión temporal y organizativa, mostrando cómo la planificación se tradujo en resultados medibles.

Aunque el desarrollo del proyecto se basó en una metodología ágil tipo Scrum, para efectos de medición y análisis de métricas de proceso se agruparon las actividades en seis etapas representativas (Planificación, Definición, Análisis, Diseño, Desarrollo e Implementación, y Testing y Finalización).

Esta clasificación permite cuantificar el esfuerzo global de manera ordenada, manteniendo la filosofía iterativa y colaborativa del enfoque ágil.

El total global alcanzó las 508 horas, manteniéndose dentro del rango previsto y evidenciando una correcta distribución del trabajo entre ambos integrantes.

12.2.1. Métrica de tiempo y esfuerzo

La métrica de tiempo y esfuerzo mide la relación entre las horas destinadas y las tareas completadas, garantizando que el proyecto avance dentro de los plazos previstos.
Para el seguimiento de estas métricas se utilizó un tablero en Trello acompañado por un diagrama de Gantt, lo que permitió visualizar de forma organizada la progresión de actividades y el cumplimiento de los objetivos planificados.

Estas herramientas posibilitaron un control efectivo de la carga horaria, especialmente en las etapas más demandantes del proceso.
Tal como se había previsto, la fase de Desarrollo e Implementación fue la que requirió mayor esfuerzo, tanto en horas como en complejidad de tareas.
Esta etapa resultó fundamental no solo para la construcción del software, sino también para la investigación y aplicación de herramientas tecnológicas, integrando teoría y práctica en un contexto profesional real.

12.2.2. Horas globales por etapa

El siguiente cuadro resume la distribución de horas según la etapa del proyecto.

Vale aclarar que inicialmente habíamos estimado una duración total de 540 horas, teniendo en cuenta el trabajo de en promedio 10 horas semanales por cada integrante del equipo durante 27 semanas.

Ahora bien, acá mostramos los resultados reales y puede observarse que las fases de Desarrollo e Implementación y Testing y Finalización concentran la mayor carga horaria, lo que es consistente con la naturaleza técnica y de validación de estas etapas.

Etapa	Integrante (Dayron Muñiz)	Integrante (Diego Balbis)	Total de horas
Planificación	6	5	11
Definición	30	31	61
Análisis	12	12	24
Diseño	9	8	18
Desarrollo e implementación	108	112	220
Testing y finalización	84	90	174
Total	249	259	508

Tal como se había establecido en la planificación, la fase de Desarrollo e Implementación representó la instancia más demandante en cuanto a tiempo y esfuerzo.

Esta exigencia estuvo vinculada a la necesidad de profundizar en el dominio del lenguaje y de las herramientas de desarrollo, lo cual implicó un trabajo adicional para garantizar que la construcción del sistema se realizara de forma fluida y sin contratiempos.

Este proceso no solo fue clave para materializar el producto final, sino que también se constituyó en una instancia de crecimiento técnico y profesional para los integrantes del equipo.

12.2.3. Distribución porcentual global

La siguiente tabla muestra la proporción de horas invertidas por etapa, lo que permite visualizar el peso relativo de cada fase en el total del proyecto.

Etapa	Horas	Porcentaje sobre el total
Planificación	11	2,2%
Definición	61	12,0%
Análisis	24	4,7%
Diseño	18	3,5%
Desarrollo e implementación	220	43,3%
Testing y finalización	174	34,3%
Total	508	100%

12.2.4. Conclusión de las Métricas de Proceso

El esfuerzo total evidencia un trabajo colaborativo y complementario entre los integrantes del equipo.
Las diferencias entre etapas y participantes son mínimas, lo que refleja una participación balanceada, donde ambos aportaron tanto en las tareas técnicas como en la gestión del proyecto.

La mayor concentración horaria en Desarrollo e Implementación (43 %) es coherente con el núcleo de trabajo en Scrum, donde la mayor parte del tiempo se destina a generar incrementos funcionales del producto.
A su vez, la etapa de Testing y Finalización (34 %) tuvo un peso elevado, asociado a la revisión de código, validaciones funcionales y aseguramiento de la calidad del software.
Las fases de Definición, Análisis y Diseño complementaron este proceso, aportando la base conceptual y estructural necesaria para un desarrollo ordenado.

Además, las tareas de documentación técnica y académica se desarrollaron de forma transversal, acompañando cada fase del proyecto y concentrándose especialmente en la etapa final para la elaboración del informe y la defensa del trabajo.

13. Conclusiones Generales

La realización de este proyecto nos permitió consolidar los conocimientos adquiridos a lo largo de la carrera, integrando aspectos técnicos, metodológicos y humanos en un producto funcional y con propósito social.

Desde su concepción, el sistema se diseñó para abordar una problemática actual: los fraudes digitales basados en ingeniería social. A través de una plataforma web educativa e interactiva, se logró ofrecer una herramienta que promueve la conciencia y el aprendizaje en ciberseguridad, combinando teoría y práctica mediante actividades prácticas como juegos.

En el plano técnico, la adopción del entorno .NET 9 con arquitectura MVVM y base de datos SQL Server 2022 garantizó un diseño modular, escalable y de fácil mantenimiento. Las métricas aplicadas confirmaron un código con alta mantenibilidad, bajo acoplamiento y complejidad controlada, reflejando un desarrollo concordante con los principios de la ingeniería de software moderna.

En el plano metodológico, la aplicación del marco ágil Scrum permitió un trabajo iterativo, con entregas parciales verificables, reuniones de seguimiento constantes y adaptación a los cambios. Esta metodología fomentó la colaboración, la organización del trabajo y la autogestión del equipo, asegurando el cumplimiento de los plazos establecidos.

En cuanto al proceso formativo, la experiencia de planificación, desarrollo, testing y documentación ofreció un aprendizaje integral, fortaleciendo competencias técnicas, comunicacionales y de gestión. CAPFIS se constituye, por tanto, en un producto académico completo que evidencia la capacidad del equipo para analizar, diseñar, implementar y evaluar soluciones tecnológicas aplicadas a un problema real.

Finalmente, el proyecto no solo cumple con los objetivos propuestos, sino que aporta un valor educativo y social, al contribuir a la formación de usuarios más conscientes y preparados frente a las amenazas de la ingeniería social.

13.1 Trabajos a futuro

Aunque la herramienta cumple con los requerimientos relevados y los objetivos propuestos, consideramos que existen diversas líneas de mejora que podrían abordarse en futuras versiones, orientadas a ampliar su alcance educativo, funcionalidad y capacidad de adaptación frente a las nuevas formas de ataque en ingeniería social. Entre ellas se destacan:

- Módulo de estadísticas avanzadas, que brinde reportes automáticos de rendimiento y progreso tanto a los usuarios como al administrador.

- Optimización de accesibilidad, garantizando que la plataforma sea totalmente usable por personas con discapacidades visuales, auditivas o motrices.

- Desarrollo de una aplicación móvil complementaria, que permita acceder a los módulos desde dispositivos Android e iOS de forma nativa, facilitando la continuidad del aprendizaje desde cualquier lugar.

- Integración de inteligencia artificial educativa, con la finalidad de ofrecer tutorías automatizadas, recomendaciones personalizadas y simulaciones adaptativas en función del desempeño del usuario.

Estas mejoras buscan consolidar a CAPFIS como una herramienta educativa viva, flexible y sostenible, capaz de evolucionar.

14. Bibliografía

Arcos Sebastián, S. (2011). *Ingeniería social: Psicología aplicada a la seguridad informática* [Trabajo de fin de grado, Universitat Politècnica de Catalunya]. Repositorio UPCommons. Consultado: 5 de mayo del 2025, de https://upcommons.upc.edu/server/api/core/bitstreams/f31da3e7-5aff-4547-bcc7-e60c0161983c/content

Colli, M. (2020). *Ingeniería social y las herramientas informáticas involucradas en un ataque* [Trabajo final de especialización, Universidad de Buenos Aires]. Biblioteca

Digital de la Facultad de Ciencias Económicas (UBA). Consultado: 7 de mayo del 2025, de http://bibliotecadigital.econ.uba.ar/download/tpos/1502-1646_ColliM.pdf

IBM. (s.f.). *Ingeniería social.* Consultado: 14 de abril del 2025, de https://www.ibm.com/es-es/topics/social-engineering

Instituto Nacional de Ciberseguridad (INCIBE). (s.f.). *Técnicas de ingeniería social* [Infografía]. Consultado: 16 de abril del 2025, de https://www.incibe.es/ciudadania/formacion/infografias/tecnicas-ingenieria-social

Kaspersky. (s.f.). *¿Qué es la ingeniería social?* Consultado: 18 de abril del 2025, de https://latam.kaspersky.com/resource-center/definitions/what-is-social-engineering

Lobato Pacheco, A. G. (2005). *Uso de los estándares de la IEEE para el desarrollo de software* [Tesis de licenciatura, Universidad Nacional Autónoma de México]. Repositorio TesiUNAM. Consultado: 12 de mayo del 2025, de https://tesiunamdocumentos.dgb.unam.mx/pd2005/0601639/0601639.pdf

Microsoft. (2025). *C# - Lenguaje de programación.* Microsoft Learn. Consultado: 24 de abril del 2025, de https://learn.microsoft.com/es-mx/dotnet/csharp/

Microsoft. (2025). *Documentación de .NET.* Microsoft Learn. Consultado: 24 de abril del 2025, de https://learn.microsoft.com/es-mx/dotnet/

Mitnick, K. D., & Simon, W. L. (2011). *El arte de la intrusión: Las historias reales detrás de los exploits de hackers.* McGraw-Hill.

Orbegozo Arana, B. (2019). *Desarrollo de aplicaciones C# con Visual Studio .NET: Curso práctico.* Alfaomega Grupo Editor y Altaria Publicaciones. Consultado: 12 de mayo del 2025, de https://api.pageplace.de/preview/DT0400.9786076225219_A43551109/preview-9786076225219_A43551109.pdf

Palacio Amador, D. (2020). *Propuesta de una metodología de aseguramiento y control de calidad para los proyectos de software de Inclutec* [Trabajo final de graduación, Tecnológico de Costa Rica]. Repositorio Institucional TEC. Consultado: 15 de mayo del 2025, de https://repositoriotec.tec.ac.cr/bitstream/handle/2238/11483/TFG_Dionisio_Palacio.pdf?sequence=1&isAllowed=y

SCRUMstudy™. (2017). *Guía SBOK™: Cuerpo de Conocimiento de Scrum* (3.ª ed.). SCRUMstudy™. Consultado: 10 de mayo del 2025, de https://primeconsultores.com.pe/wp-content/uploads/2021/02/SCRUMstudy-SBOK-Guide-3rd-edition-Spanish.pdf

Sommerville, I. (2011). *Ingeniería del software* (9.ª ed.) [PDF]. Consultado: 24 de abril del 2025, de https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

Solís Fajardo, L. (2023). *Ciberseguridad y amenazas informáticas: el rol de la ingeniería social. Religación. Revista de Ciencias Sociales y Humanidades*, 7(34), 104–117. Consultado: 18 de abril del 2025, de https://revista.religacion.com/index.php/religacion/article/view/1310/1661

Susatama, M. (2022). *La ingeniería social como herramienta de los ciberdelincuentes: análisis de técnicas y contramedidas* [Trabajo de grado, Universidad Piloto de Colombia]. Repositorio Institucional UNIPILOTO. Consultado: 19 de abril del 2025, de https://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/12497/articulo%20Marcela%20susatama.cleaned.pdf