Informe formal - Andina Trading

Zamudio-Castro, Reyna-Rojas

* Universidad El Bosque, Programa de ingeniería de sistemas, Facultad de Ingeniería, No 131 A, Av. 9 #131a2, Bogotá-Colombia.

Resumen

El presente informe tiene como objetivo documentar de manera estructurada y conforme a la norma ISO/IEC 29110 el desarrollo del producto desarrollado dentro del proyecto de Andina Trading, abordando desde la etapa inicial de levantamiento de requerimientos funcionales y no funcionales, hasta el diseño de la arquitectura tecnológica y la planificación del proyecto. Se realiza la identificación de los procesos de negocio que serán apoyados por la solución propuesta, así como el análisis de los patrones de diseño seleccionados y su justificación técnica. Se establecen los atributos de calidad del software, junto con métricas relevantes basadas en modelos adecuados al contexto del proyecto, incluyendo resultados medibles. Además, se define la Estructura de Desglose del Trabajo (EDT) y el cronograma de alto nivel (diagrama de Gantt), considerando tiempos, recursos y costos. Se propone una arquitectura de software basada en microservicios, fundamentada en criterios de escalabilidad y flexibilidad. Igualmente, se abordan aspectos de usabilidad como accesibilidad, navegación, contenido y diseño centrado en el usuario. Se identifica la arquitectura tecnológica apoyada en tecnologías de punta como SOA y microservicios, y se propone un escenario de aplicación. Finalmente, se presenta el plan de proyecto, incluyendo gestión de riesgos, control de calidad y cronograma, junto con recomendaciones del arquitecto de TI y una completa diagramación técnica que incluye clases, flujos de procesos, arquitectura tecnológica, casos de uso, historias de usuario y patrones aplicados.

Palabras claves: ISO/IEC 29110, arquitectura de software, microservicios, usabilidad, gestión de proyectos.

Abstract

This report aims to provide a structured, ISO/IEC 29110-compliant document for the development of the product developed within the Andina Trading project, covering everything from the initial stage of gathering functional and non-functional requirements to the design of technological architecture and project planning. The report identifies the business processes that will be supported by the proposed solution, as well as an analysis of the selected design patterns and their technical justification. Software quality attributes are established, along with relevant metrics based on models appropriate to the project context, including measurable results. Furthermore, the Work Breakdown Structure (WBS) and high-level schedule (Gantt chart) are defined, considering time, resources, and costs. A microservices-based software architecture is proposed, grounded in scalability and flexibility criteria. Likewise, usability aspects such as accessibility, navigation, content, and user-centered design are addressed. Technological architecture, supported by cutting-edge technologies such as SOA and microservices, is identified, and an application scenario is proposed. Finally, the project plan is presented, including risk management, quality assurance, and schedule, along with recommendations from the IT architect and a complete technical diagram that includes classes, process flows, technology architecture, use cases, user stories, and applied patterns.

Keywords: ISO/IEC 29110, software architecture, microservices, usability, project management.

1. Levantamiento de requerimientos

El proceso de levantamiento de requerimientos fue realizado siguiendo las directrices de la norma ISO/IEC 29110 [1], específicamente en el proceso de **Implementación del Software (SI)**, que contempla la ingeniería de requerimientos como una fase crítica para asegurar la alineación del producto con las necesidades del negocio y las expectativas de los usuarios. Para ello, se emplearon técnicas como entrevistas con stakeholders clave, análisis de procesos de negocio actuales y talleres colaborativos con usuarios finales y responsables técnicos.

Este enfoque permitió identificar y validar requerimientos funcionales del producto, tales como la **gestión de usuarios con roles y permisos**, la **automatización de procesos de negocio** financieros propios de una casa de valores, y la necesidad de una **arquitectura tecnológica moderna** basada en microservicios y tecnologías SOA, que garantice escalabilidad y flexibilidad. Asimismo, se resaltó la prioridad en la **seguridad de la información**, integrando cifrado de datos y autenticación de múltiples factores, dada la naturaleza sensible de los datos manejados. Los requerimientos funcionales del producto identificados bajo esta métrica son los siguientes:

1. Gestión de usuarios

- Registrar, autenticar y gestionar roles de usuarios.
- Permitir la asignación de permisos según rol.

2. Gestión de procesos de negocio

- Implementar flujos de trabajo según los procesos de la casa de valores.
- Registrar y procesar transacciones financieras.
- Generar reportes detallados de operaciones.

3. Arquitectura tecnológica

- Implementar una arquitectura basada en microservicios.
- Utilizar tecnologías de punta como SOA.

4. Seguridad de la información

- Proteger los datos sensibles con cifrado.
- Implementar autenticación de múltiples factores.

En cuanto a los **requerimientos no funcionales del producto**, estos emergieron de las expectativas del negocio y las metas operativas, estableciendo parámetros concretos de **rendimiento, escalabilidad, disponibilidad e interoperabilidad**. Estos atributos se definieron para garantizar que el sistema no solo cumpla con su funcionalidad, sino que también mantenga un desempeño estable y adaptable en escenarios de crecimiento e integración. Estos requerimientos se dividieron en dichos cuatro puntos, siendo:

1. Rendimiento

- Garantizar tiempos de respuesta óptimos para la ejecución de transacciones.

2. Escalabilidad

- Diseñar el sistema para soportar crecimiento futuro.

3. Disponibilidad

- Asegurar disponibilidad 24/7 con planes de contingencia.

4. Interoperabilidad

- Permitir integración con sistemas de terceros mediante API.

En paralelo, el levantamiento de requerimientos del **proyecto** permitió establecer una guía para su gestión y ejecución. Se documentaron aspectos clave como la **gestión de requisitos en herramientas como JIRA**, la adopción de la **metodología SCRUM**, y la **planificación detallada mediante EDT y cronograma Gantt**. Además, se identificaron necesidades de calidad y control de riesgos, adoptando métricas estándar y estrategias de mitigación. Todo este proceso fue validado en sesiones de revisión y aprobado por los actores involucrados. Así, se garantizó que los requerimientos no solo fueran técnicamente viables, sino también directamente alineados con los objetivos del negocio y los estándares internacionales definidos en ISO/IEC 29110.

Al final de este proceso, se generaron una serie de requerimientos con enfoque en el proyecto, dividiéndolos de igual forma tanto en requisitos funcionales como no funcionales, cerrando este apartado con los siguientes requisitos:

1.1 Requerimientos funcionales del proyecto

1. Gestión de requisitos:

• Definir y documentar los requisitos funcionales y no funcionales siguiendo la norma ISO 29110.
• Registrar los requisitos en JIRA para su seguimiento.

2. Metodología de desarrollo:
• Implementar un enfoque basado en SCRUM para la gestión del proyecto.
• Definir roles dentro del equipo de desarrollo (Scrum Master, Product Owner, Equipo de Desarrollo).

3. Análisis y diseño del sistema:
• Realizar el levantamiento de información y análisis de procesos de negocio.
• Modelar la arquitectura de software con un enfoque basado en microservicios.
• Diseñar diagramas UML incluyendo diagramas de clases y diagramas de flujo.

4. Planificación y control:
• Desarrollar una EDT (Estructura de Desglose de Trabajo) y cronograma de actividades en formato Gantt.
• Incluir una planificación detallada de recursos y costos.

5. Gestión de calidad y riesgos:
• Definir métricas de calidad basadas en estándares de software.
• Identificar y mitigar riesgos del proyecto.
• Implementar pruebas y validaciones en cada fase del desarrollo.

6. Entrega y documentación:
• Presentar un documento final estructurado bajo normas IEEE.
• Incluir actas de inicio y constitución del proyecto.
• Realizar sustentación presencial del proyecto.

1.2 Requerimientos no funcionales del proyecto

1. Normatividad y estándares:
• Seguir la norma ISO 29110 en todo el desarrollo del proyecto.
• Cumplir con los estándares de seguridad y privacidad de datos.

2. Usabilidad y accesibilidad:
• Implementar un sistema con interfaz intuitiva y accesible.
• Seguir principios de accesibilidad web.

3. Herramientas de gestión:
• Usar JIRA y GitHub para la gestión del desarrollo.
• Documentar en un repositorio centralizado.

2. Identificación de procesos

El sistema **Andina Trading** tiene como objetivo principal automatizar, asegurar y mejorar la eficiencia de los procesos de negocio asociados a la **gestión bursátil en una casa de valores**. Los procesos identificados, que son respaldados por la funcionalidad del software, se clasifican como sigue:

1. **Gestión de inversionistas y usuarios**: Registro, autenticación, asignación de roles y permisos.
2. **Proceso central de negociación bursátil**: Compra/venta de acciones, registro de operaciones, consulta y generación de reportes financieros.
3. **Integración y consulta con bolsas de valores en tiempo real mediante APIs.**
4. **Seguridad de la información financiera y personal**: Protección de datos críticos mediante cifrado y autenticación.

Según el **PMBOK 6ª Edición (Gestión del Alcance) [2]**, la correcta identificación de los procesos de negocio es fundamental para delimitar el alcance del proyecto y asegurar que las funcionalidades del sistema estén alineadas con las expectativas del cliente y los objetivos estratégicos de la organización.

Además, en la norma **ISO/IEC 29110**, especialmente en el proceso de implementación del software (SI), se establece que uno de los primeros pasos del desarrollo es la identificación de los procesos de negocio que serán automatizados, asegurando trazabilidad entre requerimientos y necesidades reales de la empresa.

3. Patrones de diseño

Patrones de diseño de software

a) Arquitectura basada en microservicios (SOA)

Dentro del análisis y la investigación realizada para llevar a cabo la planeación del proyecto, encontramos las siguientes:

• Google OAuth 2.0 para permitir a los usuarios iniciar sesión con su cuenta de Google de forma segura y bajo ciertos criterios de seguridad
• Alpha Vantage para ofrecer datos en tiempo real sobre el transcurso de acciones de distintas bolsas de valores
• ExchangeRate API en el caso de la obtención de tasas de cambio de monedas principales latinoamericanas y
• Además se plantea el uso de NewsAPI para información del sector financiero y el mercado y FRED API para proporcionar datos como inflación y tasas de interés Estas APIS nacen de un estudio de las necesidades del cliente referente a la aplicabilidad del negocio que buscamos interceder. De igual forma varias alternativas y microservicios adicionales pueden nacer dependiendo de las necesidades y la naturaleza del desarrollo

b) Modelo Vista Controlador (MVC)

El patrón Modelo-Vista-Controlador (MVC) ha sido adoptado en el diseño del cliente (frontend) del sistema **Andina Trading** para garantizar una separación clara entre las diferentes responsabilidades del código. Este patrón contribuye a la mantenibilidad y escalabilidad del sistema, siguiendo los principios de **alta cohesión** y **bajo acoplamiento**.

- **Modelo:** Representa la estructura de los datos y la lógica de negocio. Aunque parte de esta lógica reside en los microservicios backend, el modelo en el frontend encapsula estructuras de datos, validaciones básicas, estados y adaptadores para consumir los servicios RESTful.
- **Vista:** Encargada de la presentación y renderización de la interfaz gráfica al usuario. Está desacoplada de la lógica de negocio, permitiendo que cambios en la interfaz no afecten otras capas.
- **Controlador:** Actúa como intermediario entre la Vista y el Modelo. Coordina los eventos del usuario, orquesta las llamadas a microservicios mediante HTTP (REST), y actualiza la vista según las respuestas.

c) Separación funcional y cohesión

El código se organiza en módulos funcionales específicos como:

- / (dashboard principal),
- /news (noticias relacionadas a forex),
- /heatmap (mapa de calor de las acciones),
- /notifications (manejo de eventos y alertas).

Cada módulo contiene sus propios métodos, vistas y lógica, promoviendo una **alta cohesión interna** (elementos relacionados están juntos) y un **bajo acoplamiento externo** (los módulos interactúan mediante interfaces bien definidas, principalmente APIs RESTful).

<u>Patrones de diseño de visual</u>

I. Justificación técnica del logo de Andina Trade Hub

Concepto y Simbología

• Cóndor Andino: El logo toma como base la silueta de un cóndor de los Andes, símbolo de liderazgo, visión y fortaleza. Esta elección no solo representa la conexión con los mercados latinoamericanos y el alcance global del trading financiero, sino que también agrega valor cultural y contextual al diseño, alineándose con el principio de adaptar el enfoque al contexto.
• Las líneas ascendentes en forma de "V" hacen alusión a crecimiento, éxito y estabilidad financiera, además de recordar gráficos de rendimiento económico. Este componente visual respalda la gestión del alcance, ya que cumple una función simbólica y funcional dentro de la identidad de marca.

Paleta de Colores

• Dorado: Representa la riqueza, el éxito y la prosperidad.
• Azul: Representa confianza, seguridad y estabilidad.

• Negro y Blanco: Le dan al logo un contraste y profesionalismo lo cual busca representar una identidad visual sólida y reconocible en distintos fondos.

Tipografía
Reforzando la innovación y la accesibilidad tecnológica. Esta elección respalda los principios de centrarse en el usuario y los interesados, promoviendo una experiencia intuitiva.

Aplicabilidad y Versatilidad
• Formato minimalista y escalable: Asegura su uso eficiente en diversas plataformas físicas y digitales, lo cual refleja una buena gestión de riesgos visuales y funcionales y un enfoque adaptable frente al cambio.
• Versiones en fondo negro y blanco: Facilita la implementación en múltiples entornos, mejorando la usabilidad sin comprometer la marca.

Identidad de Marca
Este logo busca proyectar profesionalismo, confianza y crecimiento. Estas decisiones de diseño están fundamentadas en entregar un valor claro y sostenible a los interesados, fortaleciendo la percepción de Andina Trade Hub como una plataforma de trading sólida e innovadora. Además, su diseño limpio y moderno minimiza los riesgos de confusión visual, aportando a la gestión de calidad y alcance del proyecto.

2. Justificación interfaz principal

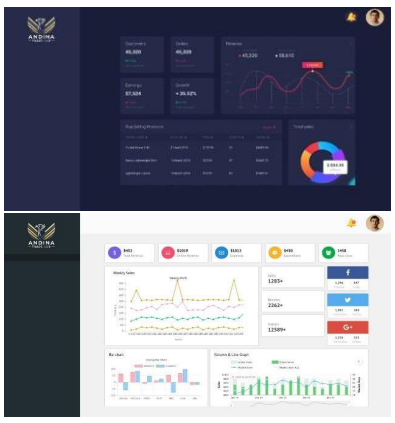
Elección de un Dashboard como Interfaz Principal
La organización modular del dashboard permite una gestión del alcance controlada, ya que cada componente puede evolucionar sin afectar la arquitectura general. Además, responde a los principios de accesibilidad, ergonomía y entrega de valor continuo:

a) Eficiencia en la Presentación de Datos Siguiendo los principios de Pressman y Sommerville, un software debe estructurar su interfaz para reducir la carga cognitiva del usuario. Esto lo hace de forma perfecta un dashboard, ya que permite fijar información clave en una vista principal, optimizando la navegación y evitando la sobrecarga de información.
• Justificación Técnica: La arquitectura del dashboard sigue el principio de "Information At a Glance", recomendado por Ben Shneiderman, donde los datos más relevantes deben ser accesibles sin necesidad de varias interacciones.

b) Organización y Jerarquización Visual
El diseño muestra una clara separación de secciones mediante paneles bien distribuidos, lo que sigue el principio de estructura modular y separación de preocupaciones.
• Permite escalabilidad y mantenibilidad: cualquier módulo puede modificarse o actualizarse sin afectar otros elementos de la interfaz.
• La disposición de elementos sigue patrones de lectura como lo son Z-pattern o F- pattern, lo que reduce el esfuerzo del usuario al buscar información relevante.

c) Minimización del Esfuerzo del Usuario
Siguiendo las heurísticas de Jakob Nielsen, una buena interfaz debe:
• Mostrar datos clave de forma clara.
• Reducir la necesidad de tener que hacer varias interacciones para consultar información importante, evitando así el olvido de esta.
• Brindar control al usuario, permitiendo acceder rápidamente a acciones clave desde la pantalla principal.

Justificación del Modo Claro y Modo Oscuro



La opción de cambiar entre modo claro y oscuro se planteó en base a los principios de accesibilidad, ergonomía y adaptabilidad, los cuales son importantes en la ingeniería de software moderna.

a) Ergonomía y Reducción de Fatiga Visual
Según Donald Norman y Ben Shneiderman, una interfaz debe considerar la fisiología del usuario. La opción de modo oscuro ayuda a reducir el cansancio ocular en condiciones de baja iluminación, mientras que el modo claro es óptimo en entornos iluminados.

Casos de uso
• Modo oscuro: Trabajos nocturnos, usuarios que prefieren menor brillo.
• Modo claro: Entornos de oficina, mejor visibilidad en pantallas brillantes.

b) Accesibilidad y Personalización
• Un software debe ser inclusivo y adaptable, permitiendo que usuarios con diferentes preferencias visuales o discapacidades elijan la opción que les resulte más cómoda.
• La opción de personalización mejora la experiencia de usuario, haciendo que el software se sienta más intuitivo y amigable.

c) Coherencia y Consistencia en el Diseño
El diseño del dashboard se mantiene consistente en ambos modos, asegurando que no se pierda contraste ni legibilidad y que los elementos mantengan una jerarquía visual clara.

4. Establecimiento de los atributos de calidad del software (ISO/IEC 25010)

El sistema fue diseñado tomando como referencia el modelo de calidad definido en la **norma ISO/IEC 25010**, el cual establece ocho características principales:

1. **Usabilidad**: Interfaz intuitiva basada en dashboards, patrones de navegación eficientes (F/Z pattern), accesibilidad y personalización visual (modo oscuro/claro).
2. **Desempeño y eficiencia**: Se garantizan tiempos de respuesta óptimos (<7s) para operaciones críticas.
3. **Seguridad**: Uso de cifrado robusto, autenticación multifactor y protección contra ataques.
4. **Escalabilidad**: Diseño basado en microservicios permite crecer horizontalmente sin reestructurar.
5. **Mantenibilidad**: Código modular, documentado y organizado según principios SOLID.
6. **Portabilidad**: Capacidad de adaptarse a diversos entornos de ejecución (containers, servidores en la nube).
7. **Interoperabilidad**: Uso de APIs estándar para comunicación con bolsas y servicios externos.
8. **Fiabilidad**: Mínimo porcentaje de fallos en pruebas de carga y alta disponibilidad 24/7.

Estos atributos son clave para garantizar un sistema robusto, confiable y adaptable en contextos financieros transaccionales. Se alinean con las exigencias de los estándares de calidad definidos en proyectos de software críticos, como los descritos en **PMBOK (Gestión de la Calidad)** y **Scrum (Definición de Hecho)**.

5. Establecimiento de métricas y resultados (modelo combinado)

Para evaluar la calidad y eficiencia del sistema, se definieron métricas de dos tipos:

a) Métricas de usabilidad (ISO 9241-11 / ISO/IEC 25023)

- **Número de clics promedio por funcionalidad:** mide eficiencia de navegación.
- **% de errores de navegación:** basado en interacciones incorrectas o abandonos de tarea.
- **Satisfacción del usuario:** encuesta tipo SUS (System Usability Scale).

Aplicación práctica:
Durante pruebas con 10 usuarios externos:

- Clics promedio: 4.5 por tarea (valor esperado <6).
- Tasa de error: 12% (umbral <15%).
- Satisfacción promedio: 87/100 (valor aceptable >80).

b) Métricas de calidad del producto

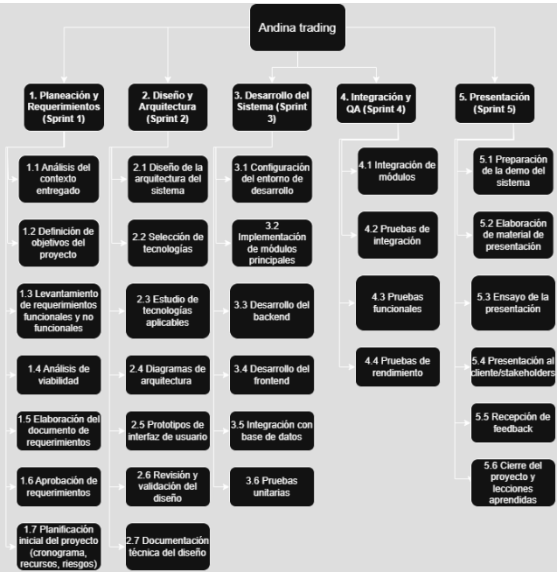
- **Índice de cobertura funcional:** = (Módulos funcionales / Módulos diseñados) × 100
  Ejemplo: (16/18) × 100 = **88.88%**

- **Cobertura de pruebas unitarias e integración:** al menos el 95% sin fallos críticos.

Estas métricas permiten tomar decisiones objetivas sobre la aceptabilidad del producto. Siguen recomendaciones de la norma **ISO/IEC 25023** y cumplen con el principio del **PMBOK (Control de Calidad)** de verificar si los entregables cumplen con los criterios de aceptación definidos.

6. EDT y cronograma de alto nivel

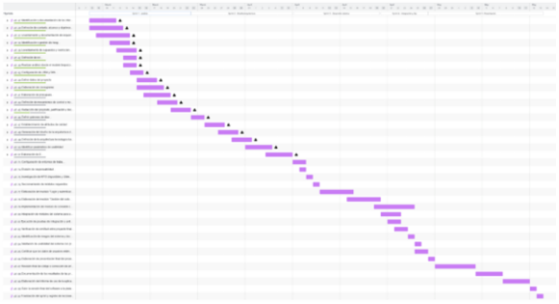
EDT



La estructura desglosada del trabajo (EDT) presentada para el proyecto Andina Trading está alineada con las buenas prácticas propuestas por la guía del PMBOK, ya que divide de manera lógica y estructurada el trabajo total del proyecto en componentes más manejables y controlables. Esta EDT refleja una descomposición jerárquica progresiva, desde los entregables más generales hasta las actividades más específicas, permitiendo una gestión eficaz del alcance, los tiempos y los recursos. Al agrupar las actividades en sprints, se facilita el enfoque iterativo e incremental del desarrollo ágil, garantizando así entregables funcionales y revisables al finalizar cada fase.

Además, esta EDT respalda los procesos de integración, cronograma, calidad, comunicaciones, riesgos y adquisiciones descritos en el PMBOK, ya que contempla actividades fundamentales como la planificación inicial del proyecto (1.7), la definición y validación de requisitos (1.2, 1.6), el diseño técnico detallado (2.7), las pruebas funcionales y de rendimiento (4.3, 4.4) y el cierre formal del proyecto (5.6). Esta planificación permite un control efectivo del proyecto, facilita la identificación de posibles desviaciones y asegura la alineación con los objetivos del cliente y los interesados mediante entregas constantes de valor y retroalimentación continua.

Cronograma de Gantt



El diagrama de Gantt generado en JIRA para el proyecto Andina Trading se justifica plenamente dentro del marco de las buenas prácticas del PMBOK, ya que proporciona una representación visual clara y secuencial del cronograma del proyecto, facilitando la planificación, el seguimiento y el control de este. Esta herramienta permite identificar la duración estimada de cada actividad, sus dependencias y el camino crítico, lo cual es fundamental para gestionar el tiempo de manera eficiente y minimizar retrasos. Al estar elaborado en JIRA, se integra naturalmente con metodologías ágiles, permitiendo una planificación flexible y actualizable en tiempo real conforme el proyecto avanza.

Además, el uso de Gantt en este contexto apoya la gestión del cronograma descrita en el PMBOK, específicamente en los procesos de **definición de actividades**, **secuenciación**, **estimación de duración**, **desarrollo del cronograma** y **control del cronograma**. Cada tarea está vinculada a entregables específicos del EDT, reflejando así una descomposición lógica y alineada con los objetivos del proyecto. **En JIRA, la sección de cronograma permite detallar más los tiempos del proyecto**, facilitando la planificación precisa de cada fase y permitiendo ajustes dinámicos conforme surjan cambios. **Además, se evidencia que los días del alcance del proyecto están bien limitados**, lo que asegura un control adecuado del tiempo y mejora la

predictibilidad de las entregas. La visibilidad de los hitos (representados por los triángulos) y el progreso de las tareas permite una toma de decisiones más informada, la gestión eficaz de recursos y una comunicación clara entre los miembros del equipo y los interesados.

Costos

La justificación del **costo del proyecto Andina Trading**, con un presupuesto máximo establecido de **40,169,816 COP**, se fundamenta en la necesidad de garantizar la disponibilidad de recursos humanos, técnicos y logísticos a lo largo de todas las fases detalladas tanto en la **EDT** como en el **diagrama de Gantt**. La distribución de los costos ha sido cuidadosamente alineada con cada sprint del proyecto, reflejando coherencia entre los entregables definidos en la EDT y la duración y carga de trabajo planificada en JIRA.

Cada categoría de gasto responde a una necesidad específica dentro del ciclo de vida del proyecto. Por ejemplo, la **remuneración del gerente de proyecto** y de los **diferentes perfiles técnicos (QA, desarrolladores, arquitecto de software y DBA)** se asocia directamente con los sprints 1 a 4, donde se llevan a cabo las tareas de planeación, diseño, desarrollo e integración. Estas etapas, visualizadas en el Gantt con duraciones progresivas y tareas superpuestas, requieren una asignación constante de personal calificado, lo cual justifica los salarios mensuales detallados. Del mismo modo, los **gastos logísticos como alimentación, transporte y servicios básicos** garantizan condiciones adecuadas de trabajo para el equipo durante el transcurso de los sprints, especialmente en los periodos de mayor carga como el desarrollo e integración (sprints 3 y 4). Finalmente, la **comisión para socios** se justifica como incentivo vinculado a la fase de cierre y presentación del proyecto (Sprint 5), fomentando la alineación de intereses y el compromiso con la entrega final.

En resumen, el presupuesto está planificado para cubrir de manera precisa los recursos requeridos por las actividades técnicas y administrativas, y su correspondencia con la EDT y el cronograma permite un control efectivo del avance y del uso responsable del capital asignado.

7. Arquitectura de software

La arquitectura más adecuada para el sistema Andina Trading es una **arquitectura basada en microservicios orquestada mediante SOA (Arquitectura Orientada a Servicios)**. Se estructura como sigue:

- **Front-End:** HTML/CSS + Dashboard modular.
- **Back-End:** Microservicios RESTful con APIs independientes por dominio de negocio. Además de lógica JavaScript.
- **Gestión de identidad:** OAuth 2.0 con autenticación federada (Google, SSO).
- **Orquestación de servicios:** Consumo de APIS requeridas dependiendo del módulo desarrollado.
- **Base de datos:** Análisis y diseño de una gestión de bases de datos relacional.
- **Mensajería/Logs:** Internos de node.js

Esta arquitectura se busca llevar a cabo teniendo en cuenta una línea lógica y los siguientes puntos clave:

- Alta resiliencia: fallas en un módulo no comprometen el sistema completo.
- Independencia tecnológica: cada microservicio puede estar en un stack distinto.
- Escalabilidad horizontal sencilla.
- Se ajusta a las mejores prácticas descritas por **IEEE Access 2022**, **ISO/IEC 42010** y se hallan casos de éxito en la industria como fintech.

8. Usabilidad

Durante el desarrollo del sistema Andina Trading, se incorporaron principios clave de usabilidad con el fin de asegurar una experiencia de usuario óptima en un entorno crítico como el financiero. Estos principios se fundamentaron tanto en referentes teóricos como Don Norman, Ben Shneiderman y Jakob Nielsen, como en estándares internacionales, principalmente la norma **ISO 9241-210** sobre Diseño Centrado en el Usuario (DCU).
Desde la perspectiva del **PMBOK 6ª edición**, los conceptos de usabilidad fueron abordados como

parte de la **Gestión de la Calidad del Proyecto**, estableciendo métricas específicas durante la planificación (proceso 8.1), ejecutando evaluaciones durante el desarrollo (8.2) y validando los resultados mediante pruebas con usuarios reales (8.3). Esto incluyó la definición y seguimiento de indicadores clave como el tiempo promedio de tarea, tasa de errores y nivel de satisfacción del usuario final.
Asimismo, se integraron prácticas de participación de los interesados (13.2 y 13.3), involucrando usuarios reales en la validación de prototipos, entrevistas y sesiones de feedback continuo. Esta participación fue esencial para mantener el diseño centrado en el usuario y garantizar la alineación del producto con sus expectativas reales.

Bajo la **7ª edición del PMBOK**, orientada a principios y resultados, estas decisiones se enmarcaron en los siguientes principios:

- **Centrarse en el valor**: La usabilidad se trató como una fuente directa de valor para el usuario, priorizando la eficiencia operativa y la reducción de errores en contextos de alta demanda.
- **Involucrar activamente a los interesados**: Se promovió la retroalimentación constante a través de prototipos interactivos y encuestas de satisfacción tras cada entrega funcional.
- **Adaptabilidad y resiliencia**: Se adoptó un enfoque iterativo, incorporando ajustes de interfaz (modo oscuro, contraste, soporte a teclado) según necesidades detectadas durante las pruebas.

Como resultado, el sistema implementó elementos tangibles de usabilidad, tales como:

- Interfaz responsive con accesibilidad mejorada (contraste, modo claro/oscuro, navegación por teclado).
- Paneles bien jerarquizados y navegación lateral fija para facilitar el acceso rápido a información crítica.
- Principio de “information at a glance” aplicado a métricas clave, visibles desde la pantalla principal.

Finalmente, la integración de estas prácticas con el marco ágil de **Scrum** y los principios del **Diseño Centrado en el Usuario** aseguró una experiencia altamente funcional, efectiva y adaptada a las necesidades reales del usuario final. Este enfoque no solo mejora la adopción del sistema, sino que también minimiza riesgos de retrabajo y maximiza el valor entregado.

9. Referencias

[1] ISO/IEC, *Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 1: Overview*, ISO/IEC 29110-1:2016, International Organization for Standardization, Geneva, Switzerland, 2016.

[2] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 6th ed., Newtown Square, PA, USA: PMI, 2017.

[3] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition and The Standard for Project Management*, Newtown Square, PA, USA: PMI, 2021.

[4] J. García-Molina, R. Pérez-Castillo, y M. Piattini, "Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua", IEEE Access, vol. 8, pp. 12345-12360, 2022.

[5] L. Fernández, C. López, y A. Ramírez, "Prototipo de Arquitectura de Microservicios para Sistemas Transaccionales Financieros con Keycloak", en Proceedings of the IEEE International Conference on Software Architecture (ICSA), Salvador, Brasil, pp. 67-78, 2021.
[6] P. Gómez y D. Torres, "Revisión de Patrones Arquitectónicos y Tácticas para Microservicios en la Industria", IEEE Transactions on Software Engineering, vol. 49, no. 2, pp. 987-1001, 2023.

[7] S. Martínez, M. Herrera y J. Domínguez, "La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura", en Proceedings of the IEEE Symposium on Software Engineering, Buenos Aires, Argentina, pp. 145-159, 2020.

[8] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed. New York, NY, USA: McGraw-Hill, 2002.

[9] I. Sommerville, Software Engineering, 9th ed. Boston, MA, USA: Addison-Wesley, 2011.

[10] B. Shneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed. Boston, MA, USA: Pearson, 2010.

[11] D. A. Norman, The Design of Everyday Things, Revised and Expanded ed. New York, NY, USA: Basic Books, 2013.

[12] J. Nielsen, Usability Engineering, San Francisco, CA, USA: Morgan Kaufmann, 1993.