



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Sistemas

**Implementación de un backend en Java Gradle basado
en flujos XML para la gestión de sesiones USSD
orientadas a servicios de autogestión en
telecomunicaciones para una Empresa de Telefonía**

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Sistemas

AUTOR

Kevin Daniel SANCHEZ CERVANTES

ASESOR

Mg. César Alberto MOLINA NEYRA

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Sanchez, K. (2026). *Implementación de un backend en Java Gradle basado en flujos XML para la gestión de sesiones USSD orientadas a servicios de autogestión en telecomunicaciones para una Empresa de Telefonía*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Kevin Daniel Sanchez Cervantes
Tipo de documento de identidad	DNI
Número de documento de identidad	73940791
URL de ORCID	https://orcid.org/0009-0000-4006-3178
Datos de asesor	
Nombres y apellidos	Molina Neyra, César Alberto
Tipo de documento de identidad	DNI
Número de documento de identidad	40553679
URL de ORCID	https://orcid.org/0000-0002-8018-7973
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Trujillo Trejo John Ledgard
Tipo de documento	DNI
Número de documento de identidad	06187585
Miembro del jurado 1	
Nombres y apellidos	Bustamante Olivera Víctor Hugo
Tipo de documento	DNI
Número de documento de identidad	25655590
Datos de investigación	
Línea de investigación	No aplica

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima
	Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2025
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Sistemas

Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional

Siendo las 16:00 horas del día 31 de marzo del año 2026, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Mg. Trujillo Trejo John Ledgard (Presidente), Mg. Bustamante Olivera Víctor Hugo (Miembro) y el Mg. Molina Neyra Cesar Alberto (Miembro Asesor), usando la plataforma Meet (<http://meet.google.com/wcc-jfha-tms>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“IMPLEMENTACIÓN DE UN BACKEND EN JAVA GRADLE BASADO EN FLUJOS XML PARA LA GESTIÓN DE SESIONES USSD ORIENTADAS A SERVICIOS DE AUTOGESTIÓN EN TELECOMUNICACIONES PARA UNA EMPRESA DE TELEFONIA”**, por el Bachiller **Sanchez Cervantes Kevin Daniel**; para obtener el Título Profesional de Ingeniero de Sistemas.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de 19 (**DIECINUEVE**).

A continuación el Presidente de Jurados el Lic. Trujillo Trejo John Ledgard, declara al Bachiller **Ingeniero de Sistemas**.

Siendo las 16:50 horas, se levantó la sesión.

Presidente

Lic. Trujillo Trejo John Ledgard

Miembro

Lic. Bustamante Olivera Víctor Hugo

Miembro Asesor

Mg. Molina Neyra Cesar Alberto

CERTIFICADO DE SIMILITUD

Yo, Cesar Alberto Molina Neyra en mi condición de asesor acreditado mediante Resolución Decanal N° 001268-2025-D-FISI/UNMSM y OFICIO N° 0023-PTTSP-FISI-2025 del Trabajo de Suficiencia Profesional, cuyo título es Implementación de un backend en Java Gradle basado en flujos XML para la gestión de sesiones USSD orientadas a servicios de autogestión en telecomunicaciones para una Empresa de Telefonía, presentado por el bachiller Sanchez Cervantes Kevin Daniel para optar el Título Profesional de Ingeniero de Sistemas, CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 3% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del **TITULO PROFESIONAL DE INGENIERO DE SISTEMAS.**

Lima, 4 de junio de 2026

Firma del Asesor:



Nombres y Apellidos del asesor: Cesar Alberto Molina Neyra

DNI: 40553679

FICHA CATALOGRÁFICA

IMPLEMENTACIÓN DE UN BACKEND EN JAVA GRADLE BASADO EN FLUJOS XML PARA LA GESTIÓN DE SESIONES USSD ORIENTADAS A SERVICIOS DE AUTOGESTIÓN EN TELECOMUNICACIONES PARA UNA EMPRESA DE TELEFONIA

AUTOR: SANCHEZ CERVANTES, KEVIN DANIEL

ASESOR: MOLINA NEYRA, CESAR ALBERTO

LIMA – PERÚ, 2026

TÍTULO PROFESIONAL: Título Profesional de Ingeniero de Sistemas

ÁREA / PROGRAMA / LÍNEA DE INVESTIGACIÓN:

Ingenierías / Tecnología de la Información y Comunicación / Ingeniería de Software

PREGRADO: Universidad Nacional Mayor de San Marcos - Facultad de Ingeniería de Sistemas e Informática - Escuela Profesional de Ingeniería de Sistemas e Informática

FORMATO:

Formato 21 × 29.7 cm

Páginas: 114

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

**Desarrollo e implementación de un backend en Java Gradle basado en flujos XML
para la gestión de sesiones USSD orientadas a servicios de autogestión en
telecomunicaciones para una empresa de Telefonía**

Autor: Sanchez Cervantes, Kevin Daniel

Asesor: Molina Neyra, Cesar Alberto

**Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de
Ingeniero de Sistemas**

Fecha: Enero del 2026

RESUMEN

Este trabajo de suficiencia profesional relata la experiencia vivida durante el diseño y la puesta en marcha de una solución tecnológica basada en servicios USSD, desarrollada para Telefónica Movistar Argentina. A lo largo del proyecto, el propósito principal fue construir una arquitectura backend que pudiera crecer con el tiempo y adaptarse a los sistemas que la empresa ya utilizaba lo que también impulsó a buscar crear un motor capaz de interpretar los flujos XML que interpretara los árboles de las marcaciones que trabajaba la empresa y también manejar sesiones USSD en tiempo real.

El desarrollo consta de la creación de un backend en Java usando Gradle, junto con el diseño de una base de datos donde junto con esto se implementaron medidas de seguridad y autenticación, también se implementaron herramientas adicionales para poder procesar plugins dinámicos, webservices, servicios de mensajería SMS y archivos personalizados para extraer campos específicos, que ampliaron las funciones del sistema y facilitaron su mantenimiento.

Desde el punto de vista económico, el resultado fue muy favorable: la solución implementada representó una inversión menor y redujo notablemente los costos de soporte frente a lo que ofrecía el proveedor anterior.

Como resultado el proyecto realizado no solo demostró su eficiencia en la solución de las necesidades que la motivaron a desarrollarse, sino que también evidenció la importancia de la comunicación constante con el área usuaria, el trabajo conjunto con el equipo de desarrollo y la capacidad de adaptarse a los plazos exigentes sin perder calidad en los entregables.

Palabras clave: Java, XML, USSD, Backend

MAJOR NATIONAL UNIVERSITY OF SAN MARCOS
FACULTY OF SYSTEMS ENGINEERING AND COMPUTER SCIENCE
PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

**Development and implementation of a Java Gradle backend based on XML flows
for the management of USSD sessions oriented to self-management services in
telecommunications at Telefónica Argentina**

Author: Sanchez Cervantes, Kevin Daniel

Advisor: Molina Neyra, Cesar Alberto

**Title: Professional Proficiency Project to qualify for the Professional Title of
Systems Engineer**

Date: January 2026

ABSTRACT

This professional proficiency project recounts the experience gained during the design and implementation of a technological solution based on USSD services, developed for Telefónica Movistar Argentina. Throughout the project, the main purpose was to build a backend architecture that could grow over time and adapt to the systems already used by the company, which also prompted the search for an engine capable of interpreting XML flows that would interpret the dialing trees used by the company and also handle USSD sessions in real time.

The development consists of the creation of a backend in Java using Gradle, along with the design of a database where security and authentication measures were implemented. Additional tools were also implemented to process dynamic plugins, web services, SMS messaging services, and custom files to extract specific fields, which expanded the system's functions and facilitated its maintenance.

From an economic standpoint, the result was very favorable: the implemented solution represented a lower investment and significantly reduced support costs compared to what the previous provider offered.

As a result, the project not only demonstrated its efficiency in meeting the needs that motivated its development, but also highlighted the importance of constant communication with the user area, joint work with the development team, and the ability to adapt to demanding deadlines without compromising the quality of deliverables.

Keywords: Java, XML, USSD, Backend

INDICE GENERAL

INTRODUCCIÓN.....	1
CAPÍTULO I – TRAYECTORIA PROFESIONAL.....	3
CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA..	10
VISIÓN.....	12
MISION	12
ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS	13
EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN	15
CAPÍTULO III - ACTIVIDADES DESARROLLADAS.....	15
3.1. SITUACIÓN PROBLEMÁTICA	17
3.2 Factores críticos de éxito identificados	19
3.3. SOLUCIÓN AL PROBLEMA	20
3.4 OBJETIVOS	22
3.4.1 Objetivo general.....	22
3.4.2 Objetivos específicos	23
3.5 ALCANCE.....	23
3.6 ETAPAS Y METODOLOGÍA	24
3.6.1 Identificación y selección de proyectos	26
3.6.2 Iniciación y planificación del proyecto.....	26
3.6.3 Análisis de requerimientos	26
3.6.4 Diseño lógico y físico del sistema	27

3.6.5 Implementación	27
3.6.6 Pruebas del sistema.....	27
3.6.7 Implementación en producción y soporte.....	28
3.7 FUNDAMENTOS UTILIZADOS.....	28
3.7.1 Fundamentos Teóricos.....	28
3.7.2 Variables causa-efecto	33
3.8 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS	34
3.8.1 Iniciación y planificación del proyecto.....	35
3.9 DISEÑO LÓGICO Y FÍSICO DEL SISTEMA	52
3.9.2 Implementación	67
3.10 EVALUACIÓN	97
3.10.1 EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-BENEFICIO..	97
3.10.2 Flujo de Caja Proyectado.....	98
3.10.3 Interpretación del VAN y TIR.....	98
CAPITULO IV - REFLEXIÓN CRITICA DE LA EXPERIENCIA.....	99
CAPITULO V – CONCLUSIONES Y RECOMENDACIONES	101
Conclusiones	101
Recomendaciones.....	102
BIBLIOGRAFÍA.....	103

Índice de Tablas

Tabla 1 Experiencias Laborales	3
Tabla 2 Educación Académica	7
Tabla 3 Cursos realizados / Certificaciones.....	8
Tabla 4 Idiomas estudiados por el autor	9
Tabla 5 Desempeño vs tipo de condición USSD.....	40
Tabla 6 Principales paquetes de la solución USSD primera parte.....	57
Tabla 7 Principales paquetes de la solución USSD segunda parte.....	58
Tabla 8 Paquetes de la segunda parte del proyecto USSD	59
Tabla 9 Principales librerías y dependencias del proyecto	65
Tabla 10 Descripción de las distintas tablas en el proyecto USSD	93
Tabla 11 Análisis comparativo de costos y ventajas competitivas de la solución propuesta frente al proveedor anterior.....	97
Tabla 12 Tabla de flujo de caja proyectado.....	98

Índice de Figuras

Figura 1 Cronograma de proyecto para el proyecto USSD	36
Figura 2 Características a incluir en la solución USSD.....	39
Figura 3 Servicios y Marcaciones.....	41

Figura 4 Relevación de información sobre WebServices	42
Figura 5 Vista grafica de una marcación con sus opciones	43
Figura 6 Vista de todos los árboles de las marcaciones a nivel grafico	45
Figura 7 Flujo de saltos en una marcación	46
Figura 8 Estructura de los archivos MAPPING Y ERROR	46
Figura 9 Vista de un XML y su estructura	47
Figura 10 Vista de la tabla de contenido de los Plugin a implementar.....	49
Figura 11 Correo de la empresa	50
Figura 12 Mapeos de los parametros de un plugin	51
Figura 13 Diagrama ER de la solucion USSD.....	52
Figura 14 Vistas complementaria del diagrama ER de la solución USSD.....	53
Figura 15 Diseño integral de la solución telefónica	54
Figura 16 Vista de los paquetes de la solución USSD.....	56
Figura 17 Principales paquetes de la solución USSD tercera parte.....	61
Figura 18 Principales paquetes de la solución USSD cuarta parte	62
Figura 19 Librerías del proyecto.....	67
Figura 20 Vista del service principal	69
Figura 21 Vista del controlador del proyecto USSD	70
Figura 22 Clase Flow del proyecto USSD.....	73
Figura 23 Service de presentación de menús USSD	75

Figura 24 Lógica de navegación entre menús USSD	76
Figura 25 Lógica de envío de SMS vía HTTP	78
Figura 26 Clase encargada de gestionar consultas a la BD	81
Figura 27 Modelo para la clase Interface USSD	82
Figura 28 Clase Service que contiene el modelo de los servicios contemplados para el proyecto USSD	84
Figura 29 Modelo de la Clase Transaction del proyecto USSD	86
Figura 30 Modelo de la clase orientada a los WebServices	87
Figura 31 Clase orientado a la gestión de tokens	89
Figura 32 Service que contiene la lógica de cada uno de los plugins.....	91
Figura 33 Modelado de las tablas correspondientes a la solución USSD.....	93
Figura 34 Estimaciones de volúmenes de información	95
Figura 35 Pruebas de Carga	96

INTRODUCCIÓN

El presente trabajo de suficiencia profesional se orienta a exponer la experiencia adquirida en el desarrollo de soluciones tecnológicas en el sector de telecomunicaciones, tomando como eje central el proyecto “Implementación de un backend en Java Gradle basado en flujos XML para la gestión de sesiones USSD orientadas a servicios de autogestión en telecomunicaciones para una empresa de Telefonía”. Este proyecto se realizó en el marco de las funciones realizadas en la empresa donde se desarrolló la experiencia profesional, empresa privada del sector tecnológico, orientada al diseño e implementación de soluciones para los sectores de telecomunicaciones, transporte y sistemas de información, la cual por razones de confidencialidad será referida de manera genérica a lo largo del documento, donde se aplicó manera práctica conocimientos en programación, arquitectura de software y gestión en base de datos.

La solución desarrollada por el presente autor consistió en la solución de un backend que actúa como motor lógico encargado de interpretar arboles XML donde dichos flujos se llevan a cabo con la interacción entre el usuario y el sistema mediante menús USSD permitiendo que, a partir de una marcación en el celular, se despliegue una serie de opciones dinámicas que son procesadas en tiempo real y que internamente tienen muchas casuísticas. Para lograr esto el sistema implementado interpreta arboles completos XML donde hay diversos tipos de tags de diferentes marcaciones donde se gestiona la sesión activa de cada usuario y se determina la ruta que debe seguir según las opciones que se van marcando y los nodos que le continúan en cada interacción, enlazándose además con una capa de datos que tiene información de la totalidad de árboles de ser necesario(en caso de redirecciones o de inclusión de fragmentos de árboles en los menús), así como plugins o webservices garantizando de esta forma la correcta continuidad de los procesos.

Este trabajo además no se limita únicamente al aspecto técnico del proyecto, sino que también busca reflejar cómo la trayectoria profesional y la experiencia previa del autor fueron fundamentales para enfrentar los retos que demandó la solución de la empresa. La implementación de este backend no solo optimizó la gestión integral del sistema USSD, también contribuyó a mejorar los procesos de consultas a webservices, lectura de archivos. MAPPING y. ERROR reduciendo la dependencia de atención directa y ofreciendo a los clientes de Telefónica Argentina una herramienta más ágil y eficiente.

De esta manera con la finalidad de abordar este tema de manera integral y completa el presente documento se encuentra organizado en varios capítulos. En el Capítulo I se presenta la trayectoria profesional, resaltando la experiencia acumulada en diferentes proyectos y empresas tanto públicas como privadas, luego el Capítulo II describe el contexto en el que se desarrolla la experiencia, detallando la actividad de la empresa donde se desarrolló la experiencia profesional, su misión, visión, organización y el cargo desempeñado, siguiendo el Capítulo III donde se expone las actividades desarrolladas, dividiéndose en la situación problemática que motivó el proyecto, la definición del problema, los objetivos, la solución implementada y su alcance, el Capítulo IV plantea una reflexión crítica sobre la experiencia resaltando los principales aprendizajes y desafíos afrontados, por último, el Capítulo V reúne las conclusiones y recomendaciones que se derivan del trabajo realizado.

Como se puede apreciar en el presente documento constituye no solo la memoria técnica de un proyecto específico, sino también la evidencia de un recorrido profesional que ha permitido poner en práctica las competencias necesarias para afrontar con solvencia retos del sector tecnológico y de telecomunicaciones.

CAPÍTULO I – TRAYECTORIA PROFESIONAL

El autor se desempeña como senior en Análisis de Datos (Data Analyst) y Data Engineer con amplia experiencia en la construcción de soluciones de software empresariales, estadísticos y de telecomunicaciones desde el diseño hasta la puesta en producción.

Con enfoque en la mejora continua, optimización de procesos y metodologías ágiles (Scrum, Kanban). Alto dominio en diversas tecnologías y habilidades de liderazgo, resolución de conflictos y toma de decisiones.

Actualmente desarrolla sus labores en el INEI liderando el área de infraestructura y dando continuo soporte a las necesidades con información en la nube (AWS) así como On-Premise.

Anteriormente el autor trabajó para empresas de diferentes tamaños e instituciones. Los detalles se describen en la Tabla 1.

Experiencia como Bachiller:

Tabla 1
Experiencias Laborales

Empresa/Periodo	Descripción de actividades realizadas
INEI	Cargo: Sr. Cloud Data Analyst
(Julio 2025-Actualidad)	Ubicación: Lima, Presencial
	Actividades:

- Extracción y transformación de datos utilizando herramientas y plataformas en la nube de AWS.
- Trabajar en estrecha colaboración con otras áreas de TI y equipos de negocio, para asegurar que la información final se alinee con las necesidades de la organización.
- Configuración de instancias de Amazon RDS para varios motores de bases de datos.
- Administración, monitoreo y optimización del rendimiento de bases de datos en la nube (AWS RDS) y on-premise (SQL Server, Oracle)
- Supervisión del desempeño de instancias RDS con CloudWatch, Performance Insights y herramientas on-premise (SQL Server Profiler).
- Ajuste de índices, consultas, planes de ejecución y configuración de parámetros para maximizar la eficiencia de las cargas de trabajo.
- Administración de backups, restores y planes de contingencia en entornos productivos críticos.
- Aplicación de parches y upgrades tanto en RDS como en instancias locales para mantener la seguridad y compatibilidad.

Tecnologías y Herramientas/Lenguajes:

SSMS, Oracle SQL Developer, AWS RDS, CloudWatch, CloudTrail, Backup, IAM y KMS para la administración, monitoreo y seguridad, Grafana, Prometheus, ETL, AWS Glue, AWS Lambda y Git, garantizar seguridad y compliance con Secrets Manager.

**Citi / Manpower
Group**

Cargo: Coordinador de Proyectos

Ubicación: Colombia, Remoto

**(Febrero
2023-Abril
2025)**

Actividades:

- Lideré iniciativas de proyectos desde el análisis de requerimientos hasta la implementación.
- Interactué con stakeholders en LATAM para alinear requerimientos del negocio.

- Ejecuté análisis de riesgos y propuse planes de mitigación efectivos.
- Mejora e implementación de flujos internos en plataformas del banco con la plataforma CITIDOCS(CITI).
- Responsable de ejecutar los proyectos para los productos transaccionales del Banco para los clientes corporativos de todo LATAM.
- Priorización y organización de requerimientos en Jira, asegurando la correcta definición de épicas, historias de usuario y tareas.
- Coordinación con el Product Owner para mantener el backlog actualizado y alineado con los objetivos del proyecto.
- Conducción de daily stand-ups, sprint planning, sprint review y retrospectivas, promoviendo la colaboración entre desarrolladores, QA y stakeholders.
- Resolución de impedimentos y gestión de dependencias para asegurar el cumplimiento de los compromisos asumidos en cada sprint.

Tecnologías y Herramientas/Lenguajes: Zephyr, Jyra, Citidocs, MSProject, Excel, Word, BasicCosmos, Scrum

HACOM S.A.C.

**(Agosto 2020-
Febrero 2023)**

Cargo: Analista de Sistemas Senior/ Data Engineer

Ubicación: Lima, Remoto

Actividades:

- Desarrollo de sistemas de control interno para tramas GPS y tráfico masivo de datos.
- Creación de portales web para auditorías y monitoreo de indicadores en Telco.
- Implementación de soluciones para Telefónica de Argentina (logs, aprovisionamiento, mensajería).
- Desarrollo de sistemas de ventas y soluciones optimizadas para el sector telecomunicaciones,etc.
- Desarrollo y mantenimiento de un backend en Java para gestión de sesiones USSD, implementando un motor dinámico basado en flujos XML que definían menús interactivos, con integración a sistemas core de telecomunicaciones para

consultas de saldo, activación de servicios y promociones en tiempo real.

- Sistemas de depuración de Logs de telefonía para obtener los números a provisionar o ser provisionados.

Tecnologías y Herramientas/Lenguajes: Java, Python, Angular, JavaScript, C++, PHP, SQL, MongoDB, PostgreSQL, Ionic, Springboot, Linux, Springboot, Grafana, Prometheus, Postman, GIT

Experiencia como egresado/estudiante:

Empresa/Periodo	Descripción de actividades realizadas
INEI (Octubre 2016- Abril 2020)	Cargo: Analista de Sistemas- Asistente Informático- Programador de Sistemas Ubicación: Lima, Presencial Actividades: • Desarrollo de aplicativos móviles para encuestas y censos nacionales. • Propuesta e implementación pionera de capacitaciones digitales. • Modelado de procesos censales y seguimiento de proyectos de captación de datos. • Liderazgo en desarrollo y puesta en marcha de APKs y sistemas de monitoreo gerencial. • Desarrollo de diagrama de flujos en Bizagi, modelado de procesos para distintas encuestas. • Desarrollo y optimización de stored procedures, vistas, lógica de procesamiento de información y triggers para soportar sistemas de encuestas a nivel nacional. • Implementación de soluciones eficientes en SQL para la gestión de grandes volúmenes de datos y la automatización de procesos críticos. • Diseño y creación de diagramas de Entidad-Relación (ER) para modelar bases de datos de acuerdo con los requerimientos de las encuestas.

Tecnologías y Herramientas/Lenguajes: Java, Python, Angular, JavaScript, C++, PHP, SQL, Android, MySQL, Springboot, Maven

Leadercom S.A.

Cargo: Programador de Sistemas

**(Diciembre 2014-
Diciembre 2015)**

Ubicación: Lima, Presencial

Actividades:

- Desarrollo de sistemas web comerciales y herramientas de monitoreo para clientes.
- Modelado de procesos y diseño de interfaces UX/UI.
- Desarrollo de aplicaciones móviles comerciales
- Desarrollo de store procedures, vistas y diseño de diagrama de base de datos p

Tecnologías y Herramientas/Lenguajes: Java, JavaScript, C++, SQL, Android, MySQL

Nota. Fuente: Elaboración Propia

La educación académica del autor se detalla en la Tabla 2:

Tabla 2
Educación Académica

Período	Educación Académica
---------	---------------------

2013-2020	Grado Académico de Bachiller en Ingeniería de Sistemas
	Escuela Académico Profesional de Ingeniería de Sistemas
	Facultad de Ingeniería de Sistemas e Informática
	Universidad Nacional Mayor de San Marcos

Nota. Fuente: Elaboración Propia

Los cursos realizados por el autor se muestran en la Tabla 3:

Tabla 3
Cursos realizados / Certificaciones

Periodo	Certificación	Institución
2025	Zoho Nivel Admin.	Udemy
2013	Universidad Nacional Mayor de San Marcos	Ofimática: Herramientas Web – Excel- Word
2020	MitoCode	Java Full Stack conformado por Spring Boot 2 y Angular 11
2023	InfoPUCP	Curso de Capacitación en Excel Avanzado Macros
2016	Java Web	CETPRO Comunidad Europea
2016	SQL Server	CETPRO Comunidad Europea

2025	Ultimate AWS Certified Solutions Architect Associate 2025	Udemy
2025	Ultimate AWS Certified Cloud Practitioner CLF-C02 2025	Udemy

Nota. Fuente: Elaboración Propia

Los estudios de idiomas realizados por el autor se listan en la Tabla 4:

Tabla 4
Idiomas estudiados por el autor

Periodo	Idioma	Nivel	Institución
2019	Inglés	Avanzado	Focus Your Mind
2020	Portugués	Avanzado	Centro de Idiomas San Marcos

Nota. Fuente: Elaboración Propia

CAPÍTULO II - CONTEXTO EN EL QUE SE DESARROLLA LA EXPERIENCIA

EMPRESA – ACTIVIDAD QUE REALIZA

La empresa donde se desarrolló la experiencia profesional es una empresa peruana fundada en el año 2008, integrada por capitales peruanos y uruguayos. Desde sus inicios se ha orientado al sector de las telecomunicaciones, desarrollando e implementando plataformas tecnológicas de última generación que cumplen con estándares internacionales como 3GPP, ETSI y OMA. Gracias a esta base técnica y a la experiencia acumulada que la empresa ha ido ganando en cada proyecto la compañía ha logrado expandir sus servicios a más de diez mercados en Sudamérica y Centroamérica, actualmente se encuentra apuntando a mercados de Europa y Estados Unidos, consolidándose como un socio estratégico para distintos operadores móviles.

Entre las especializaciones de la empresa se puede decir que abarca soluciones de geolocalización, mensajería avanzada, servicios de difusión (Cell Broadcast) e interceptación legal de comunicaciones, entre muchas más. No obstante, el alcance de la empresa ha ido más allá del campo estrictamente telecomunicacional, diversificando su portafolio hacia el sector transporte y gestión empresarial.

En este marco, la empresa ha desarrollado productos como una plataforma de monitoreo vehicular en tiempo real con funciones de seguridad y control operativo; sistemas móviles y web orientados a empresas que desean realizar ventas digitales; un

sistema que combina localización GPS con herramientas de gestión móvil para el personal en campo; y, un software de optimización dirigido al transporte urbano que busca aportar orden, eficiencia y sostenibilidad a este sector. Todos estos desarrollos reflejan la capacidad de la empresa para adaptarse a las necesidades locales, integrando innovación tecnológica con la realidad del mercado peruano y regional.

La empresa no se ha limitado únicamente a ofrecer soluciones tecnológicas para telecomunicaciones, sino que ha sabido diversificar sus servicios para atender necesidades más amplias dentro del ecosistema digital. En la actualidad, la empresa combina servicios de valor agregado (VAS) con herramientas de business intelligence, permitiendo a sus clientes acceder a campañas basadas en localización, marketing móvil, análisis de big data en tiempo real y control de servicios críticos como mensajería, USSD y plataformas de localización segura. Este enfoque integral le ha permitido consolidarse como un socio estratégico para operadores y empresas que buscan transformar digitalmente sus procesos sin incurrir en costos elevados.

Un rasgo importante de la compañía es su capacidad de integración end-to-end, es decir, abarcar desde el desarrollo de plataformas hasta la implementación y soporte de los sistemas en campo. Gracias a ello, puede atender proyectos que van desde el diseño de arquitecturas complejas para operadoras móviles hasta la provisión de hardware especializado, como servidores de alto rendimiento o equipos GPS para servicios de geolocalización. De este modo, combina la experiencia en software con un conocimiento técnico sólido en infraestructura (Veritrade, 2022).

En los últimos años, la empresa ha mostrado también un crecimiento en su presencia comercial y en la manera de acercarse al mercado. A través de plataformas como LinkedIn, la empresa comunica de forma activa sus proyectos y mantiene contacto con clientes potenciales, reforzando su posicionamiento en el sector corporativo (LinkedIn, 2024). Esta dinámica ha favorecido que logre alianzas estratégicas en la región y que participe en concursos y licitaciones en los que ha resaltado no solo por la calidad de sus soluciones, sino también por ofrecer propuestas más rentables frente a competidores internacionales.

Finalmente, la organización complementa su oferta con un enfoque sólido en la gestión del talento humano, asegurando que sus colaboradores cuenten con capacitación continua y con las competencias necesarias para adaptarse a las demandas del mercado. Este balance entre innovación tecnológica, solidez técnica, eficiencia en costos y capital humano preparado constituye la base sobre la cual la empresa ha construido su posición en el mercado regional (Datahub, 2023).

VISIÓN

La visión de la empresa donde el presente autor labora es consolidarse como tal una empresa de referencia en la región reconocida por ofrecer soluciones tecnológicas que acompañen la evolución de las telecomunicaciones y que al mismo tiempo fortalezcan sectores estratégicos como el transporte y la gestión empresarial. La compañía busca ser percibida no solo como un proveedor de servicios de amplios y variados sectores, sino también como un socio confiable que contribuye a la seguridad, la productividad y la modernización de sus clientes en cada mercado donde participa con sus proyectos.

MISION

La misión de la empresa donde el presente autor labora consiste en diseñar e implementar soluciones innovadoras en telecomunicaciones y gestión tecnológica que respondan a las necesidades particulares de cada organización y clientes a la que presta servicios, para lograr esto combina plataformas sólidas y robustas, servicios de valor agregado y sistemas de control orientados a la eficiencia operativa. También mantiene el compromiso de ofrecer soporte técnico especializado, capacitación constante y un acompañamiento cercano que garantice la sostenibilidad de cada proyecto desde la planificación hasta la etapa de post producción, mencionar de igual manera que uno de sus propósitos es también generar un impacto positivo en la productividad de sus

clientes, integrando seguridad, innovación y calidad en cada uno de sus productos y servicios.

ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

Durante mi etapa en la empresa donde realice actividades, el presente autor formo parte del equipo de desarrollo de Telco, Infraestructura y Tecnologías de la Información donde desempeñé el cargo de Analista de Sistemas Senior / Data Engineer, mi rol estuvo enfocado en el diseño, arquitectura, desarrollo y puesta en producción de proyectos tecnológicos vinculados tanto a telecomunicaciones como a soluciones de control y monitoreo en tiempo real, mencionar que dentro del equipo, el autor tuvo diversas responsabilidades como participar activamente en el ciclo de vida completo de las aplicaciones: desde la concepción de la solución, la construcción de los componentes, la integración con diferentes plataformas y bases de datos, hasta su despliegue en entornos de producción incluyendo soporte y acompañamientos post producción. Esta posición me permitió involucrarme de manera directa en proyectos de gran impacto, tanto internos como para clientes nacionales e internacionales, lo que exigió un alto nivel de adaptabilidad, dominio técnico y capacidad de trabajo en equipo.

Entre las funciones principales que el presente autor desempeño destacan las siguientes:

- Diseñar y desarrollar de sistemas de control interno para el procesamiento de tramas GPS y el manejo de tráfico masivo de datos, estos desarrollos requerían el uso de técnicas avanzadas de concurrencia en Java (manejo de hilos y mapas concurrentes) así como la integración de protocolos como MQTT, mencionar tambien que a nivel de bases de datos, el presente autor implemento diversos procedimientos almacenados, triggers y esquemas que aseguraban la integridad y eficiencia en el manejo de la información.
- Construcción de portales web orientados a la auditoría y al monitoreo de indicadores clave en el sector telecomunicaciones, empleando tecnologías como Django y MongoDB, lo que permitió a los clientes contar con herramientas visuales y de consulta en tiempo real.

- Implementación de soluciones para clientes internacionales, como Telefónica Argentina, donde desarrollé módulos relacionados con la gestión de logs, el aprovisionamiento de líneas y servicios, así como la mensajería. En estos proyectos utilicé frameworks modernos como Spring Boot y lenguajes como Python, lo que permitió entregar soluciones flexibles y escalables.

- Desarrollo de sistemas de ventas y aplicaciones optimizadas para el sector telecomunicaciones, en los cuales se buscaba integrar procesos de negocio con plataformas tecnológicas robustas que garantizaran continuidad y confiabilidad.

- Despliegue de aplicaciones en entornos Linux, donde participé en la puesta en producción de sistemas desarrollados en Angular, Python y Java, asegurando la correcta configuración de servidores, dependencias y monitoreo de servicios.

- Desarrollo de módulos de trackeo GPS en tiempo real, en los cuales apliqué técnicas de procesamiento eficiente de tramas en Java, asegurando que los sistemas pudieran manejar un gran volumen de información de manera concurrente y sin pérdida de datos.

- Creación de un sistema en Java para análisis de distancias entre antenas móviles, cuyo objetivo era calcular la separación entre ellas a partir de archivos CSV y, en base a esos cálculos, asignar la tecnología de comunicación más adecuada. Este proyecto contribuyó a optimizar la gestión de la red y a mejorar la asignación de recursos tecnológicos.

- Diseño de sistemas complementarios, como el desarrollo de un sistema de ventas web para una discoteca local llamada *Aqua*, que incluía la venta de entradas y delivery de snacks y bebidas. Este proyecto me permitió aplicar mis conocimientos en el desarrollo de plataformas de comercio electrónico adaptadas a necesidades específicas del cliente.

- Desarrollo de una API basada en XML sobre HTTP, diseñada para gestionar sesiones de usuario mediante login y logout. La solución incluía un esquema de control de sesiones con tiempo máximo de inactividad de cinco minutos. Una vez autenticado, el sistema permitía realizar operaciones como altas y bajas de aprovisionamiento de números telefónicos, consultas y modificaciones, para el manejo

de datos como motor de base de datos se usó MongoDB para el almacenamiento y gestión de los datos se utilizó MongoDB, logrando un desempeño eficiente y confiable.

- Gestionar y desplegar en entornos Linux donde además de configurar servidores para la puesta en producción, tuve a mi cargo la ejecución de comandos y scripts en sistemas operativos Linux, todo esto incluyó el deploy de proyectos contruidos en Angular y Django así como también la ejecución y automatización de procesos escritos en Python y Java, todas estas actividades permitieron al autor afianzar competencias en administración de servidores, control de versiones y ejecución de tareas críticas para la continuidad operativa de los sistemas logrando así que se refleje un buen perfil orientado tanto al diseño de soluciones a medida como al fortalecimiento de la infraestructura tecnológica, contribuyendo directamente a los objetivos de innovación y calidad que caracterizan a la empresa donde se desarrollaron actividades..

EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

La experiencia adquirida por el autor en la empresa donde se desarrolló actividades, significaron una etapa clave en el desarrollo profesional del sustentante, ya que le permitieron aplicar conocimientos técnicos en proyectos reales de alto impacto y a la vez, fortalecer habilidades de gestión, análisis de conflictos y resolución de problemas, mencionar que uno de los campos más significativos en los que participé fue el desarrollo de sistemas de geolocalización y control vehicular en tiempo real, los cuales se vinculan directamente con los productos insignia de la empresa como por ejemplo la plataforma LiveTrack, donde trabajé en la implementación de módulos para el procesamiento de tramas GPS, optimizando la concurrencia mediante el uso de hilos y estructuras de datos como mapas concurrentes, todos estos proyectos contribuyeron a la enseñanza de la importancia sobre eficiencia en el manejo de grandes volúmenes de datos y la necesidad de diseñar soluciones escalables con el tiempo que respondan de manera inmediata a la demanda de los usuarios.

También participé en proyectos orientados a la gestión de telecomunicaciones, como la construcción de portales web para auditoría de indicadores y el desarrollo de sistemas de aprovisionamiento de líneas telefónicas, en estos casos la integración de

tecnologías como Python, Spring Boot y bases de datos no relacionales como MongoDB resultó fundamental para entregar soluciones modernas y alineadas con las exigencias del sector. Una experiencia destacada fue el desarrollo de un sistema de cálculo de distancias entre antenas móviles, el cual permitió asignar tecnologías de comunicación en función de la proximidad entre estaciones, este proyecto específicamente me brindó la oportunidad de profundizar en el análisis de datos, aplicando lógica de negocio y programación orientada a objetos para resolver un problema muy concreto de la industria, de la misma forma, tuve la oportunidad de participar en iniciativas más orientadas al comercio electrónico y la experiencia del usuario, como el sistema de ventas web para la discoteca Aqua. Aunque este proyecto se apartaba del núcleo principal de las telecomunicaciones, me permitió explorar otras áreas de desarrollo, como la integración de pasarelas de venta y delivery, ampliando mi visión sobre la aplicación de la tecnología en distintos sectores económicos. Otra de las responsabilidades que enriqueció mi experiencia fue la relacionada con el manejo de comandos en entornos Linux para despliegue de proyectos y ejecución de procesos automatizados, mencionar también que participé en la puesta en marcha de aplicaciones desarrolladas en Angular y Django, al igual que en la ejecución de scripts escritos en Python y Java, los cuales permitieron integrar nuevas funcionalidades y mantener la estabilidad de los sistemas en producción, todas estas actividades me ayudaron a fortalecer mi capacidad de trabajar con servidores en entornos reales y de responder con eficiencia ante los retos que surgen en la etapa de operación de los proyectos tecnológicos, para finalizar también comentar mi experiencia con el desarrollo de una API en XML sobre HTTP, pude aplicar conceptos de seguridad y manejo de sesiones, asegurando que los procesos de login/logout y aprovisionamiento de números telefónicos se gestionaran de manera eficiente, en donde el diseño de esta solución representó un reto importante, ya que exigía mantener un balance entre simplicidad en el uso e integridad en el manejo de la información. Todas estas experiencias en conjunto contribuyeron a consolidar mis capacidades como desarrollador y arquitecto de sistemas, reafirmando mi interés por la optimización de procesos, la gestión eficiente de datos y el diseño de soluciones tecnológicas escalables. Más allá de los conocimientos técnicos adquiridos, esta etapa también me permitió comprender la relevancia del trabajo colaborativo, la adaptación a entornos cambiantes y la necesidad de mantener una actitud de aprendizaje constante frente a los retos que impone el sector tecnológico.

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1. SITUACIÓN PROBLEMÁTICA

En diferentes lugares del mundo los servicios USSD cuyas siglas significan Unstructured Supplementary Service Data se han convertido en una herramienta clave para facilitar autogestión, inclusión financiera y acceso a servicios básicos, y es que es importante resaltar que en regiones con conectividad limitada o donde los teléfonos inteligentes aún no están al alcance de todos, USSD permite a los usuarios revisar saldos, contratar servicios, realizar pagos o recibir notificaciones sin necesidad de internet ni interfaces gráficas complejas, ya que además este tipo de soluciones ha sido usado ampliamente en África, Asia y América Latina para atender poblaciones de bajos recursos o con bajo acceso tecnológico (TeleOSS,2023).

Mencionar también que el uso de USSD no solo ha marcado un antes y un después en el sector telecomunicaciones, sino que también ha tenido un papel decisivo en el ámbito financiero, sobre todo en regiones donde la banca tradicional no llega con facilidad y es que, a través de simples códigos marcados en el celular, millones de personas pueden hacer transacciones básicas sin necesidad de internet ni de teléfonos inteligentes de última generación. Sin embargo, lo que parecía ser una solución ideal también generó fricciones entre dos actores principales: los bancos y las empresas de telecomunicaciones.

Las compañías de telecomunicaciones sostienen que, al ser propietarias de la infraestructura de red, tienen derecho a cobrar por cada transacción realizada vía USSD. En cambio, los bancos aseguran que esos costos deberían regularse o, en algunos casos, ser absorbidos por las telcos, ya que los servicios financieros móviles se consideran un

motor de inclusión social. Esta diferencia de posiciones ha provocado tensiones en distintos países.

Un ejemplo claro se dio en Nigeria, donde en 2021 los operadores amenazaron con suspender los servicios USSD a los bancos por deudas acumuladas que superaban los 40.000 millones de nairas (que equivale a 26 millones de dólares). Durante semanas se temió que millones de usuarios quedaran sin la posibilidad de realizar operaciones bancarias básicas, como transferencias o consultas de saldo. Ante el riesgo de paralizar el sistema financiero digital, el Banco Central de Nigeria tuvo que intervenir y negociar nuevas tarifas (Prime Business Africa, 2021).

En Kenia ocurrió algo parecido: las autoridades de comunicaciones y el Tesoro Nacional tuvieron que establecer reglas para los cobros de USSD en servicios bancarios. El motivo principal fue la preocupación por los costos que enfrentaban los usuarios de zonas rurales, quienes dependen en gran medida de este canal para acceder a sus cuentas. El gobierno intervino limitando los cargos y exigiendo mayor transparencia en la facturación (Competition Authority of Kenya, 2016).

También en India hubo tensiones similares, sobre todo con el programa de inclusión financiera *Jan Dhan Yojana*. Este plan buscaba que millones de personas en comunidades rurales accedieran a cuentas bancarias básicas mediante USSD. Sin embargo, las tarifas fijadas por las operadoras hicieron que el servicio fuera poco viable para los bancos, lo que obligó al gobierno a establecer topes regulatorios en los precios (PwC, 2023).

Estos casos muestran que, aunque el USSD es una tecnología relativamente sencilla, su papel como soporte de servicios críticos lo coloca en el centro de conflictos económicos y regulatorios. Si no existe un equilibrio en el modelo de costos ni una regulación clara, el servicio corre el riesgo de volverse insostenible, lo que afecta directamente a los usuarios finales.

Esta tensión obliga a buscar formas más eficientes de operar, como automatización, mejoras en el backend, uso de datos para monitoreo en tiempo real y optimización de sesiones de usuario para reducir redundancias y latencias.

En el caso específico de la empresa para la que desarrollé el proyecto, la empresa participó en un concurso para proporcionar el servicio USSD para Telefonica Argentina. El proveedor previo presentaba limitaciones importantes: su sistema tenía altos costos operativos, poca flexibilidad frente a cambios, retrasos en respuestas y dependencias manuales que afectaban la experiencia del usuario. Estas deficiencias llevaron a que la operadora buscara una alternativa que ofreciera tanto reducción de costos como mejor rendimiento técnico.

En síntesis, el problema central que dio origen a este proyecto estuvo vinculado a la necesidad de contar con una solución capaz de responder a los requerimientos de la empresa en relación con sus servicios de marcación y, en particular, con la gestión de interacciones a través de la tecnología USSD. El proveedor anterior ofrecía un servicio que muy aparte de ser costoso presentaba limitaciones en cuanto a flexibilidad y eficiencia operativa de los requerimientos que el usuario final presentaba, todo esto abrió la oportunidad de diseñar una alternativa que no solo asegurara el cumplimiento de los estándares funcionales exigidos sino que también incorporara mejoras en la administración de sesiones en tiempo real, con una alta flexibilidad en las adiciones de casos lógicos tanto para los plugins como para los diferentes webservices y un procesamiento adecuado de los flujos XML siendo así que de esta forma el proyecto se orientó hacia el desarrollo de un backend en Java Gradle, capaz de optimizar la autogestión del usuario final y al mismo tiempo reducir de manera significativa los costos operativos en comparación con el servicio ofrecido anteriormente.

3.2 Factores críticos de éxito identificados

- Que el sistema interprete todos los tags y atributos definidos en los XML (como page, redirect, ismenu, volatile) de forma precisa.
- Que la gestión de sesiones sea estable, controlando tiempos de inactividad y manteniendo continuidad entre interacciones.

- Que la capa de datos respalde adecuadamente la persistencia de las interacciones, garantizando integridad y disponibilidad.
- Que la solución reduzca el costo operacional frente al proveedor anterior, sin sacrificar calidad ni rapidez.
- Que el sistema sea escalable y tolerante: que pueda manejar varias sesiones simultáneas sin perder rendimiento.

3.3. SOLUCIÓN AL PROBLEMA

El desafío de modernizar los servicios de autogestión en telecomunicaciones no es exclusivo de Argentina, sino un fenómeno que ha cobrado relevancia global en los últimos años. La creciente demanda por soluciones rápidas, accesibles y económicas ha impulsado a diversas compañías a implementar plataformas basadas en USSD y motores de interpretación XML para gestionar interacciones masivas de clientes.

A nivel internacional, se han documentado experiencias exitosas que sirven como referencia para validar la viabilidad del presente proyecto. En India, por ejemplo, el sistema Unified Payments Interface (UPI), que combina banca móvil y USSD, ha permitido la inclusión financiera de millones de personas sin acceso a internet, marcando un precedente sobre la importancia de estas tecnologías para países con grandes poblaciones y realidades de conectividad dispares (Reserve Bank of India, 2021). En África, regiones como Kenia y Nigeria se han consolidado como líderes en el uso de plataformas USSD para banca móvil (mobile money), alcanzando tasas de adopción superiores al 70% en algunos segmentos de la población, lo cual evidencia su capacidad para sostener transacciones críticas en tiempo real (GSMA, 2021). De manera similar, en Sudáfrica se han desplegado plataformas híbridas que combinan USSD con pasarelas

SMS para mejorar la experiencia de usuario en servicios públicos, educativos y de salud (World Bank, 2020).

En Europa, las telecomunicaciones han orientado sus esfuerzos hacia la eficiencia en la operación y el cumplimiento de estándares internacionales. Vodafone, por ejemplo, ha implementado mecanismos de autogestión basados en XML, apoyados en recomendaciones del European Telecommunications Standards Institute (ETSI) y de la Open Mobile Alliance (OMA). Esto ha permitido ofrecer soluciones de autogestión más seguras y escalables, reduciendo costos asociados al soporte tradicional (ETSI, 2020). Dichas experiencias internacionales validan la pertinencia del modelo adoptado en este proyecto, ya que demuestran que la gestión de sesiones USSD con motores XML no solo es técnicamente viable, sino que responde a una necesidad estratégica del sector.

En el plano regional y nacional, Latinoamérica también ha experimentado la necesidad de modernizar sus plataformas de autogestión. En países como Perú y Colombia, la implementación de backends orientados a flujos XML ha sido clave para acercar servicios de telecomunicaciones a comunidades con conectividad limitada. Organismos como OSIPTEL en Perú han resaltado que la adopción de estos mecanismos no solo garantiza mayor cobertura, sino también la reducción de costos operativos, generando un impacto positivo tanto para las empresas como para los usuarios (OSIPTEL, 2022). De igual modo, en México y Brasil, operadores móviles han reportado que el uso de motores de interpretación XML en servicios USSD contribuye a descongestionar canales tradicionales de atención, lo que mejora la satisfacción del cliente y fortalece la fidelización (GSMA, 2020).

En este contexto, el caso de Telefónica Movistar Argentina (TMA) representa un ejemplo concreto de cómo estas tendencias internacionales y regionales pueden aterrizar a un entorno local. La compañía enfrentaba la necesidad de reemplazar a su proveedor anterior, el cual operaba bajo un modelo más rígido, costoso y con

limitaciones técnicas. La propuesta presentada ofreció no solo una solución original y adaptada a los requerimientos de la empresa, sino que también demostró eficiencia al superar múltiples escenarios de prueba con altos volúmenes de información y picos de concurrencia en tiempo real.

El factor decisivo para ganar el concurso fue doble: por un lado, la solidez técnica de la solución, que garantizaba continuidad, escalabilidad y alineamiento con estándares internacionales; por otro, los costos significativamente reducidos frente al esquema anterior. Esto último resultó especialmente relevante en un contexto de optimización de recursos, ya que permitió a la empresa mantener calidad de servicio mientras disminuía su inversión operativa.

Por tanto, confiar en este trabajo significa apostar por una solución probada y validada en condiciones reales de operación, inspirada en experiencias globales pero ajustada a la realidad de Argentina. Además, responde de forma integral a las necesidades de la empresa: mejorar la eficiencia, ampliar la autogestión del cliente y reducir los costos, lo cual sienta las bases para una estrategia sostenible en el tiempo.

3.4 OBJETIVOS

3.4.1 Objetivo general

Desarrollar e implementar un backend en Java Gradle basado en flujos XML para la gestión de sesiones USSD orientadas a servicios de autogestión en el sector de telecomunicaciones de Telefónica Movistar Argentina, asegurando eficiencia, escalabilidad y reducción de costos operativos.

3.4.2 Objetivos específicos

- Diseñar la arquitectura del backend con capacidad de integrarse de manera eficiente a los sistemas existentes de Telefónica Movistar Argentina.
- Implementar un motor de interpretación de flujos XML capaz de gestionar sesiones USSD en tiempo real, manteniendo un control riguroso del estado de cada interacción.
- Reducir los costos operativos de la empresa, ofreciendo una alternativa más económica frente al proveedor anterior.

3.5 ALCANCE

El proyecto fue desarrollado específicamente para Telefónica Movistar Argentina, dentro del área de telecomunicaciones encargada de la autogestión de clientes. El alcance incluyó:

- El diseño y desarrollo de un backend en Java Gradle.
- La integración del motor de interpretación XML con las plataformas y servicios preexistentes de la empresa.
- La puesta en marcha en producción, con soporte inicial para garantizar la estabilidad del sistema.

- La delimitación temporal corresponde al periodo comprendido entre la fase de análisis y la implementación en producción, mientras que espacialmente se orientó a los entornos técnicos de Telefónica Movistar Argentina. No obstante, el diseño modular de la solución la hace replicable en otros contextos empresariales y territoriales, lo cual constituye un valor añadido.

3.6 ETAPAS Y METODOLOGÍA

Para el desarrollo del proyecto se optó por la metodología de Kendall & Kendall (2021), reconocida en el ámbito de la Ingeniería de Software por su enfoque estructurado y progresivo en la construcción de sistemas de información. Esta metodología divide el trabajo en etapas claramente definidas, lo que facilita el orden, el control y la correcta documentación del proyecto.

De acuerdo con Kendall y Kendall (2021) las principales fases o etapas son:

- Identificación y selección de proyectos: en esta fase inicial se detectan las necesidades de la organización y se eligen los proyectos con mayor relevancia. La clave está en definir cuáles aportarán más valor estratégico y operativo.
- Iniciación y planificación del proyecto: aquí se determinan los recursos, el cronograma y los responsables. Esta etapa asegura que el equipo conozca los objetivos, limitaciones y riesgos del proyecto antes de iniciar el desarrollo.
- Análisis de requerimientos: consiste en recopilar y detallar la información de los procesos actuales, identificando qué debe resolver el sistema. Se definen las funcionalidades, el manejo de datos y las expectativas de los usuarios.
- Diseño lógico y físico del sistema: en este punto se transforma la información del análisis en modelos concretos. El diseño lógico describe el flujo de información y las reglas del negocio, mientras que el diseño físico define aspectos técnicos como la arquitectura, bases de datos, interfaces y herramientas a utilizar.

- Implementación: corresponde a la construcción del sistema, desarrollando el código, configurando las bases de datos y preparando los entornos. Se materializa todo lo definido en el diseño.
- Pruebas del sistema: se valida que el software cumpla con los requerimientos, funcione correctamente bajo diferentes escenarios y pueda manejar grandes volúmenes de datos sin errores críticos.
- Implementación en producción y Soporte: es el momento en que el sistema se despliega para su uso real dentro de la organización. Incluye la capacitación a los usuarios y la transición desde sistemas anteriores, asegurando continuidad operativa. Una vez en producción, el sistema requiere mantenimiento y ajustes para adaptarse a cambios del entorno o nuevas necesidades. El soporte garantiza la continuidad del servicio y la corrección de problemas que puedan surgir con el tiempo.

Como señalan Kendall y Kendall (2021), seguir una metodología ordenada como esta permite a las organizaciones disminuir riesgos, optimizar tiempos de desarrollo y asegurar que las soluciones implementadas realmente generen valor.

El uso de esta metodología resultó fundamental porque el proyecto debía responder a una problemática concreta: reemplazar un servicio anterior con altos costos operativos por una solución más eficiente, capaz de interpretar flujos XML y gestionar sesiones USSD en tiempo real con control de estado. Dado que se trataba de un sistema crítico para los servicios de autogestión de Telefónica Argentina, el proceso debía garantizar no solo el cumplimiento de los requerimientos técnicos, sino también la estabilidad y confiabilidad necesarias para sostener grandes volúmenes de interacción.

De este modo, las fases de Kendall y Kendall fueron aplicadas paso a paso, desde la identificación y selección del proyecto hasta el soporte posterior a la implementación en producción. Cada una de ellas permitió ordenar el trabajo, reducir riesgos y asegurar que el sistema respondiera de manera efectiva a las necesidades tanto de la empresa como de los usuarios finales según lo siguiente:

3.6.1 Identificación y selección de proyectos

En esta fase se analizan las oportunidades de mejora y se determina qué proyectos son viables. En el caso de Telefónica de Argentina, la necesidad de un nuevo backend capaz de gestionar sesiones USSD de forma eficiente, flexible y a menor costo, frente a las limitaciones del proveedor anterior, fue el detonante para priorizar este proyecto.

3.6.2 Iniciación y planificación del proyecto

Aquí se establecen los recursos, tiempos, responsables y el alcance inicial. En esta fase se definieron los lineamientos estratégicos y operativos que permitirían llevar a cabo la implementación del sistema USSD. La planificación se estructuró en un cronograma de actividades que contemplaba hitos clave, responsables y tiempos de ejecución, garantizando así un control adecuado del avance del proyecto.

3.6.3 Análisis de requerimientos

Se recopila información sobre lo que el sistema debe lograr. En este caso, se documentaron los requisitos relacionados con la lectura de flujos XML, la administración de sesiones USSD en tiempo real, la interoperabilidad con APIs y bases de datos, y la necesidad de manejar altos niveles de concurrencia.

3.6.4 Diseño lógico y físico del sistema

El diseño lógico se centró en la estructura del backend: manejo de tags XML, control de estados de sesión, conexión a base de datos y uso de hilos y mapas concurrentes en Java.

El diseño físico se enfocó en la arquitectura en producción: despliegue sobre servidores Linux, integración con protocolos (USSD, SMS, APIs internas) y pruebas de rendimiento.

3.6.5 Implementación

En la fase de implementación, se trabajó con una base de datos que almacenaba los flujos XML correspondientes a cada marcación, además de cierta información necesaria para la consulta. La base cumplía un rol de soporte para la interpretación dinámica de los flujos y el seguimiento de la sesión activa.

El backend en Java Gradle se encargaba de interpretar en tiempo real los XML, gestionar el estado de la sesión USSD y garantizar que la experiencia del usuario fuera consistente, aun bajo condiciones de alto volumen de solicitudes.

3.6.6 Pruebas del sistema

Incluyeron pruebas unitarias, de integración y de estrés. Se verificó el correcto manejo de sesiones, la interpretación de los distintos tipos de flujos XML, así como la

respuesta en escenarios de concurrencia con múltiples usuarios interactuando de manera simultánea.

3.6.7 Implementación en producción y soporte

El sistema fue desplegado en el entorno de Telefónica Argentina, donde se brindó soporte inicial, capacitación y ajustes menores para adaptarse a los cambios operativos.

Esta metodología resultó apropiada porque permitió avanzar de forma ordenada y minimizando riesgos, garantizando que cada etapa tuviera resultados claros antes de pasar a la siguiente.

3.7 FUNDAMENTOS UTILIZADOS

3.7.1 Fundamentos Teóricos

La red GSM como soporte de los servicios USSD

La red GSM (Global System for Mobile Communications) constituye uno de los fundamentos más importantes en la evolución de las telecomunicaciones móviles y ha servido históricamente como plataforma para la prestación de servicios de señalización y mensajería interactiva. A pesar de la aparición de tecnologías más avanzadas como LTE y 5G, GSM continúa siendo relevante en diversos contextos operativos, principalmente por su estabilidad, amplia cobertura y capacidad para soportar servicios que no dependen del acceso a datos móviles, como es el caso de USSD (Stallings, 2020).

Desde el punto de vista arquitectónico, la red GSM se compone de distintos subsistemas que permiten la comunicación entre el usuario final y los sistemas centrales del operador. Entre estos se encuentran la estación móvil, la red de acceso radioeléctrico y

el núcleo de red, donde operan componentes clave como el Home Location Register (HLR), el Visitor Location Register (VLR) y los centros de conmutación. Estos elementos no solo permiten la gestión de llamadas de voz, sino que también facilitan el intercambio de información de señalización a través de canales de control, los cuales son esenciales para el funcionamiento de servicios suplementarios como USSD (3GPP, 2021).

USSD (Unstructured Supplementary Service Data) es un servicio que opera directamente sobre la red GSM y permite establecer sesiones interactivas en tiempo real entre el abonado y las aplicaciones del operador o de terceros. A diferencia del servicio de mensajes cortos (SMS), USSD no almacena información ni utiliza mecanismos de entrega diferida, sino que mantiene una sesión activa mientras dura la interacción con el usuario. Esta característica le otorga una mayor velocidad de respuesta y una experiencia más cercana a una comunicación en línea, aun cuando se ejecuta sobre redes de segunda generación (Ericsson, 2022).

En el escenario actual, USSD continúa siendo ampliamente utilizado en sectores como banca móvil, telecomunicaciones, servicios públicos y plataformas de autoservicio, especialmente en regiones donde la cobertura GSM sigue siendo predominante. Su vigencia se explica por su bajo costo operativo, su eficiencia en el uso de recursos de red y su compatibilidad con una amplia gama de dispositivos móviles, incluidos aquellos de gama básica (GSMA, 2021). Estas características lo convierten en una alternativa confiable para la provisión de servicios críticos que requieren alta disponibilidad y respuesta inmediata.

Desde la perspectiva de los sistemas de información, la red GSM actúa como el medio de transporte que permite que las solicitudes USSD generadas por el usuario lleguen a los servidores de aplicación. En estos servidores se ejecutan los componentes lógicos encargados de interpretar los flujos de navegación, gestionar el estado de las sesiones y generar las respuestas correspondientes en tiempo real. En este contexto, GSM no solo cumple una función de infraestructura, sino que se posiciona como un habilitador tecnológico clave para soluciones orientadas a la interacción masiva y eficiente con el usuario final (Dahlman, Parkvall & Sköld, 2020).

Uso de USSD como vía alterna de comunicación y sus ventajas

En la actualidad, gran parte de la interacción digital entre usuarios y servicios se realiza a través de aplicaciones móviles o interfaces basadas en Internet, como los chatbots. Estas soluciones han ganado popularidad debido a su capacidad de ofrecer experiencias ricas y visualmente atractivas; sin embargo, dependen de la disponibilidad de datos móviles o conexiones Wi-Fi, así como de la respuesta de servidores centrales para funcionar correctamente (Alotaibi & Alotaibi, 2022). Este modelo plantea una limitación importante en contextos donde la conectividad es limitada, inestable o costosa.

En contraste, los servicios basados en USSD (Unstructured Supplementary Service Data) operan directamente sobre la infraestructura de la red GSM, sin necesidad de acceso

a Internet ni de aplicaciones instaladas en el dispositivo del usuario (3GPP, 2021). Esto implica que, incluso en áreas con cobertura básica de señal GSM y sin servicios de datos, los usuarios pueden interactuar con sistemas de autoservicio, realizar consultas de saldo, activar promociones o acceder a servicios de terceros de manera inmediata. Por ejemplo, servicios de banca móvil en regiones de África y Asia utilizan USSD como canal principal de transacciones debido a su fiabilidad y bajo costo operativo (GSMA, 2021).

Una de las ventajas más destacadas de USSD radica en su bajo consumo de recursos y su rapidez de respuesta. A diferencia de las aplicaciones que requieren múltiples capas de procesamiento (interfaz gráfica, consumo de APIs REST, manejo de sesiones web y renderizado de contenido), una sesión USSD se mantiene activa por períodos breves y está optimizada para intercambios de texto de bajo volumen, lo cual representa un uso eficiente de los canales de señalización de la red (Ericsson, 2022). Esto es especialmente útil en servicios donde la inmediatez y la simplicidad son prioritarias, como consultas de saldo, recargas de tiempo aire o activaciones de servicio.

Otra ventaja es la universalidad del acceso: como USSD no requiere instalación de software adicional, cualquier teléfono móvil que opere sobre una red GSM puede utilizarlo, lo que elimina barreras asociadas a la fragmentación de sistemas operativos o a la necesidad de contar con dispositivos de gama alta (Stallings, 2020). Esto lo convierte en un recurso valioso para operadores y empresas que buscan ofrecer servicios accesibles a toda su base de clientes, sin importar el tipo de dispositivo o nivel de conectividad.

Por todas estas razones, USSD se mantiene como una vía alterna robusta de comunicación, complementaria a las aplicaciones móviles y a las soluciones basadas en Internet. Su funcionamiento independiente de canales de datos, su menor coste y su capacidad de operar incluso en condiciones de conectividad mínima son factores que explican su uso continuado, particularmente en sectores donde la disponibilidad y la confiabilidad del servicio son críticas.

La seguridad constituye un aspecto crítico en cualquier sistema de comunicación que intercambie información sensible entre el usuario y un proveedor de servicios. En el caso de USSD, la seguridad se gestiona principalmente a nivel de la infraestructura de la red GSM, lo que diferencia de manera sustancial este mecanismo frente a las aplicaciones móviles que operan sobre redes IP públicas (3GPP, 2021).

Los mensajes USSD no viajan por Internet, sino que se transmiten a través de los canales de señalización de la red móvil, los cuales se encuentran bajo el control directo del operador de telecomunicaciones. Este modelo reduce significativamente la superficie de ataque asociada a amenazas comunes en entornos IP, como ataques de intermediario (Man-in-the-Middle), interceptación de tráfico en redes Wi-Fi públicas o suplantación de servidores remotos (GSMA, 2020). Además, la autenticación del usuario se basa en la SIM y el MSISDN, lo que introduce un factor de seguridad inherente al propio acceso a la red celular.

Otro elemento relevante es que las sesiones USSD son temporales y no persistentes. Cada interacción se mantiene activa solo durante un período corto y se cierra automáticamente una vez completada la transacción o alcanzado el tiempo de espera configurado. Esta característica limita la posibilidad de reutilización de sesiones o de exposición prolongada de información sensible, a diferencia de algunas aplicaciones móviles que mantienen tokens de sesión activos por tiempos extendidos (Ericsson, 2022).

En contraste, las aplicaciones móviles modernas suelen implementar mecanismos de seguridad avanzados como cifrado TLS, autenticación multifactor, certificados digitales y almacenamiento seguro de credenciales. No obstante, su dependencia de Internet introduce riesgos adicionales, ya que la comunicación atraviesa múltiples capas externas al control del operador móvil, incluyendo redes públicas, servidores intermedios y servicios de terceros (ENISA, 2021). Asimismo, estas aplicaciones pueden verse comprometidas por vulnerabilidades del sistema operativo, malware, aplicaciones maliciosas o errores en la gestión de permisos del dispositivo.

Desde una perspectiva comparativa, USSD ofrece un enfoque de seguridad más controlado y centralizado, adecuado para servicios transaccionales simples y de autogestión, donde la prioridad es la disponibilidad, la rapidez y la reducción de riesgos asociados a entornos abiertos. Por esta razón, sectores como la banca móvil básica, los servicios gubernamentales y las plataformas de autoservicio de operadores continúan utilizando USSD como canal confiable, especialmente en escenarios donde la conectividad a Internet no puede garantizarse de forma continua (GSMA, 2021).

En síntesis, aunque USSD no fue diseñado originalmente como un protocolo de alta seguridad criptográfica, su integración directa con la red GSM, su carácter efímero y su aislamiento del entorno IP lo convierten en una alternativa segura y robusta frente a aplicaciones móviles, particularmente cuando se requiere un canal confiable, universal y con bajo nivel de exposición a amenazas externas.

El desarrollo del proyecto se basó en fundamentos teóricos relacionados con los sistemas de información, la ingeniería de software y la gestión de telecomunicaciones. Entre los más relevantes destacan:

- Uso de XML como repositorio de lógica: en lugar de ser un simple lenguaje de marcado, los XML se convirtieron en la fuente central de las reglas de interacción. Cada flujo o menú estaba representado en ellos, lo que hacía más flexible la actualización del servicio, evitando cambios directos en el código del backend.

- Base de datos como repositorio de configuraciones y flujos: su papel fue más estático, enfocado en servir los XML, tener las URLs de los webservice a consultar si es que alguna marcación las invocara durante el flujo y datos necesarios para consultas puntuales durante la sesión.
- Enfoque en eficiencia y costo: la arquitectura liviana, sin capas innecesarias de persistencia, redujo costos de infraestructura y mantenimiento, lo que fue clave para competir con proveedores previos como Telefónica de Argentina.
- La persistencia y consulta de datos se apoyó en PostgreSQL, un sistema gestor de bases de datos relacional de código abierto reconocido por su robustez y soporte a transacciones complejas. En este proyecto, la base de datos no funcionó como un repositorio de información transaccional extensa, sino como un espacio de soporte donde se almacenaban los XML correspondientes a cada marcación, webservices, plugins, así como datos asociados al control de sesiones activas.
- En cuanto al manejo de concurrencia y procesamiento en tiempo real, se implementaron librerías como Akka (específicamente akka.actor.ActorRef), que facilitaron la construcción de componentes concurrentes y distribuidos, clave para soportar múltiples solicitudes simultáneas de usuarios. Este enfoque permitió garantizar que cada sesión USSD pudiera manejarse de manera independiente sin comprometer el rendimiento global.
- Por otro lado, se incorporaron características modernas de Java como Streams (java.util.stream.Collectors), lo cual simplificó el procesamiento de colecciones de datos en memoria, reduciendo la complejidad en operaciones de filtrado y agrupamiento. La implementación se apoyó además en Lombok, una librería que permitió optimizar la definición de clases eliminando código repetitivo (constructores, getters y setters), favoreciendo la legibilidad y mantenibilidad del sistema.

- El registro y monitoreo del sistema se gestionó con Log4j (org.apache.log4j.Logger), herramienta que resultó fundamental para generar trazas de ejecución, identificar errores en tiempo real y llevar un control histórico de eventos. Esta capa de logging permitió diagnosticar de forma rápida cualquier anomalía en los procesos y optimizar la fase de pruebas.
- Para la capa de datos, se trabajó con Hibernate y JPA Repository, los cuales simplificaron las operaciones con la base de datos mediante el uso de un modelo de persistencia orientado a objetos. Esto permitió que las consultas y operaciones sobre PostgreSQL se integraran de manera fluida al código del backend, reduciendo la dependencia de consultas SQL explícitas y mejorando la abstracción.
- En términos de seguridad, se utilizó Spring Security, garantizando mecanismos de autenticación y autorización necesarios para resguardar el acceso a la aplicación. Si bien el proyecto no se centraba en un portal web expuesto al público, la seguridad fue un aspecto crítico en el manejo de sesiones e interacciones con servicios internos.

3.7.2 Variables causa-efecto

- Variable causa: Limitaciones del sistema anterior para manejar sesiones USSD de forma eficiente, altos costos de mantenimiento y escasa flexibilidad en la integración.
- Variable efecto: Desarrollo e implementación de un backend en Java Gradle capaz de interpretar flujos XML, gestionar sesiones USSD en tiempo real, mejorar la eficiencia operativa y reducir significativamente los costos para Telefónica de Argentina.

3.8 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS

En esta sección se detallan las tareas realizadas dentro del marco de la metodología Kendall y Kendall, la cual comprende fases que van desde la identificación del proyecto hasta su puesta en marcha y posterior soporte. Cada una de estas etapas fue adaptada al contexto específico de la empresa de telefonía que contrató la solución y a las características técnicas del sistema diseñado.

El área encargada de la implementación del sistema USSD estuvo conformada por un equipo multidisciplinario, diseñado para cubrir tanto los aspectos técnicos del desarrollo como la gestión del proyecto. La composición fue la siguiente:

- 2 Software Developers Senior: responsables de liderar el desarrollo del backend en Java, la definición de la arquitectura del sistema y la integración de las librerías y frameworks empleados (Hibernate, Spring Security, JPA, entre otros).
- 2 Desarrolladores Semi Senior: encargados de apoyar en la codificación de los flujos XML, la gestión de sesiones USSD y la interacción con la base de datos. Su labor permitió agilizar la implementación de nuevas funcionalidades y realizar pruebas unitarias de manera continua.
- 1 Ingeniero en Telecomunicaciones: encargado de asegurar la correcta integración del sistema con la infraestructura de red de la empresa cliente, verificando la estabilidad de las sesiones USSD y validando la compatibilidad con los estándares del sector.

- 1 Project Manager: responsable de la coordinación general, la planificación de tareas según la metodología Kendall y Kendall, y el seguimiento de hitos y entregables. Además, fue el enlace directo con el cliente, garantizando que los requerimientos se mantuvieran alineados con los objetivos del proyecto.

A continuación, se presentan las fases desarrolladas, destacando cómo la metodología guió la toma de decisiones, la definición de objetivos y la aplicación de buenas prácticas durante todo el proceso de implementación

3.8.1 Iniciación y planificación del proyecto

En esta fase se definieron los lineamientos estratégicos y operativos que permitirían llevar a cabo la implementación del sistema USSD. La planificación se estructuró en un cronograma de actividades que contemplaba hitos clave, responsables y tiempos de ejecución, garantizando así un control adecuado del avance del proyecto.

Las primeras tareas estuvieron orientadas a aspectos contractuales y administrativos, como la carta de adjudicación y la recepción de la orden de compra, lo que formalizó el inicio del trabajo. Posteriormente, se dio paso a actividades técnicas preliminares, como el site survey, las visitas a locales y la programación de permisos para el equipo técnico, asegurando que las condiciones físicas y logísticas estuvieran alineadas con los requerimientos de la solución.

Una vez cubierta esta etapa inicial, la planificación contempló la implantación del hardware, la configuración y el envío de equipos, así como los procesos de desaduanaje, recepción, instalación y energización, fundamentales para garantizar la operatividad de la infraestructura. A ello le siguió el comisionamiento, que permitió validar la configuración inicial antes de proceder con fases más críticas.

El cronograma también incluyó hitos de aceptación parcial, seguidos de la integración a la red y pruebas con los distintos elementos que formarían parte del entorno

de telecomunicaciones. Finalmente, se programaron actividades de migración de servicios, donde se trasladaron progresivamente los flujos USSD existentes al nuevo sistema, cerrando con la aceptación final y las migraciones sin observaciones, asegurando que el proyecto quedara completamente operativo y validado.

De esta manera, la planificación no solo organizó las tareas técnicas y administrativas, sino que también permitió coordinar esfuerzos entre los distintos actores involucrados, asegurando una transición ordenada hacia el nuevo sistema.

Figura 1
Cronograma de proyecto para el proyecto USSD



Nota. Fuente: Documentación de la empresa

3.8.1.1 Análisis de requerimientos

Dentro de todo proceso de desarrollo de software, una de las etapas más relevantes es la identificación y definición de los requerimientos, ya que constituyen la

base sobre la cual se construye el sistema. Estos pueden dividirse en dos grandes categorías: requerimientos funcionales, que describen las acciones que el sistema debe realizar, y requerimientos no funcionales, que establecen las condiciones de calidad bajo las cuales debe operar.

En el marco del proyecto de implementación de un backend en Java Gradle para la gestión de sesiones USSD en Telefónica Argentina, se identificaron los siguientes aspectos principales:

3.8.1.1.1 Requerimientos funcionales

- Estos están relacionados directamente con las tareas y servicios que el sistema debe ejecutar para cumplir con los objetivos del negocio:
- Procesar y gestionar los flujos XML que representan los menús USSD enviados desde la marcación del usuario.
- Administrar sesiones USSD en tiempo real, contemplando la apertura, mantenimiento y cierre de las mismas.
- Integrarse con APIs internas de Telefónica Argentina para operaciones de aprovisionamiento, consultas y actualizaciones.
- Registrar los eventos y trazas de ejecución en archivos de log, permitiendo un control y monitoreo eficiente de las transacciones.

3.8.1.1.2 Requerimientos no funcionales

Por otro lado, existen condiciones de calidad que no definen funciones específicas, pero resultan esenciales para garantizar el correcto desempeño del sistema:

- Disponibilidad: el sistema debe estar activo de manera continua, dado que el canal USSD es crítico y de uso permanente.
- Escalabilidad: capacidad de soportar un alto volumen de sesiones concurrentes sin afectar la experiencia del usuario.
- Seguridad: implementación de controles que protejan datos sensibles y manejo seguro de sesiones temporales con tiempo de expiración (ejemplo: 5 minutos de inactividad).
- Rendimiento: asegurar tiempos de respuesta eficientes en interacción con el usuario.
- Mantenibilidad: facilitar la incorporación de nuevos flujos XML o la adaptación a futuras necesidades del negocio.
- Compatibilidad: asegurar que el sistema pueda integrarse con las plataformas y aplicaciones ya desplegadas en la infraestructura de la empresa.

Figura 2
Características a incluir en la solución USSD

CARACTERÍSTICAS				
5.1	La solución deberá ser capaz de soportar simultáneamente múltiples servicios USSD (pull y push) bajo múltiples códigos de servicio, siempre por medio de aplicaciones que usen métodos propietarios de conexión o API.	MANDATORIO	CUANTIFICABLE	
5.2	La solución deberá soportar servicios USSD simples y avanzados, basados en menú.	MANDATORIO	CUANTIFICABLE	
5.3	La plataforma debe soportar mensajes tipo pull.	MANDATORIO	CUANTIFICABLE	
5.4	La plataforma debe soportar mensajes tipo push.	MANDATORIO	CUANTIFICABLE	
5.5	La plataforma debe soportar todos los tipos USS, con sus respectivas particularidades en cada tipo de red.	MANDATORIO	CUANTIFICABLE	
5.6	La plataforma USSD debe soportar los siguientes métodos de mensajería entre el equipo del usuario y la aplicación: PSSD Request/Response PSSR Request/Response USSR Request/Confirm USSN Request/Confirm	MANDATORIO	CUANTIFICABLE	
5.7	La plataforma USSD debe soportar los siguientes escenarios de interacción: Single Network Initiated USSD Message/Notification Multiple Network Initiated USSD Message/Notification Solución con soporte USSD (desarrollada en el caso de que exista) para servicios propuestos deberá admitir servicios USSD utilizando los siguientes conjuntos de caracteres: Estándar de 7 bits (hasta 128 caracteres) Binario de 8 bits USSD de 16 bits El proveedor deberá indicar cuáles soporta.	MANDATORIO	CUANTIFICABLE	
5.8	La solución deberá soportar la optimización de la versión MAP.	MANDATORIO	CUANTIFICABLE	
5.9	La solución deberá soportar la segmentación de MAP.	MANDATORIO	CUANTIFICABLE	
5.10	La solución deberá soportar la configuración de tiempos de espera para evitar que las sesiones no concluidas o inactivas ocupen canales por un tiempo excesivo.	MANDATORIO	CUANTIFICABLE	
5.11	La solución propuesta debe proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.12	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.13	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.14	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.15	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.16	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.17	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.18	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.19	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.20	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	
5.21	La solución propuesta deberá proveer los criterios de empujamiento de mensajes para evitar los MO USSD request hacia la aplicación que no corresponden basados en cualquier combinación de los siguientes: USSD string MISDN MSISDN Time & Time (Day of week, Morning/Afternoon)	MANDATORIO	CUANTIFICABLE	

Nota. Fuente: Documentación de la empresa

A partir de la información presentada en la figura, se desprende un conjunto de características consideradas mandatorias para la implementación de la plataforma USSD. Dichos aspectos evidencian la necesidad de garantizar la interoperabilidad, la flexibilidad y la eficiencia en el manejo de los servicios.

En primer lugar, se observa que la solución debía soportar múltiples servicios USSD en modalidad pull y push, gestionando de manera simultánea distintos códigos de servicio y aplicaciones que operan bajo protocolos de conexión diversos.

Asimismo, se destaca la gestión avanzada de sesiones, que contempla tanto sesiones simples como aquellas basadas en menús, incluyendo la configuración de tiempos de espera para liberar recursos en caso de inactividad.

Otro elemento relevante identificado en la figura corresponde a la compatibilidad con diferentes métodos de mensajería USSD (PSSD, PSSR, USSR y USSN), además de escenarios variados de interacción, tanto iniciados por la red como por el usuario.

La información también refleja la importancia de la flexibilidad en la codificación de mensajes, asegurando soporte para caracteres en 7 bits, 8 bits y UCS2 de 16 bits. A ello se suman los requerimientos de optimización y segmentación MAP, los cuales fortalecen la eficiencia en la señalización y el rendimiento de la red.

Finalmente, se evidencia la necesidad de contar con criterios de enrutamiento flexibles (basados en parámetros como USSD string, MSISDN, IMSI, hora/fecha o ubicación) y la capacidad de alojar múltiples servicios, accesibles ya sea mediante menús estructurados o códigos de servicio diferenciados.

Tabla 5
Desempeño vs tipo de condición USSD

Desempeño USSD	Condición
- Total de aciertos por código de servicio	MANDATORIO
	CALIFICABLE
- Sesión completada con éxito (total y por servicio)	MANDATORIO
	CALIFICABLE
- Sesiones fallidas o incompletas (total y por servicio)	MANDATORIO
	CALIFICABLE
- Tasa de transacción total por segundo	MANDATORIO
	CALIFICABLE

Nota. Fuente: Elaboración Propia

De la información presentada en la figura, se desprende un conjunto de indicadores asociados al desempeño operativo de la solución USSD, los cuales han sido clasificados bajo la categoría mandatorio calificable. Esta categorización refleja que, además de ser condiciones obligatorias, su cumplimiento debía ser evaluado cuantitativamente durante las pruebas del sistema.

De la información presentada en la figura, se observa que el sistema USSD debía garantizar un control estricto sobre su desempeño mediante indicadores clave definidos como mandatorios calificables. Entre ellos se incluyen el total de aciertos por código de servicio, la medición de sesiones completadas con éxito, la identificación de sesiones fallidas o incompletas y la tasa de transacciones por segundo. Estos parámetros no solo permiten evaluar de manera objetiva la estabilidad y confiabilidad de la plataforma, sino

que además ofrecen un marco claro para verificar que el sistema responda de forma eficiente a grandes volúmenes de tráfico en escenarios de alta demanda.

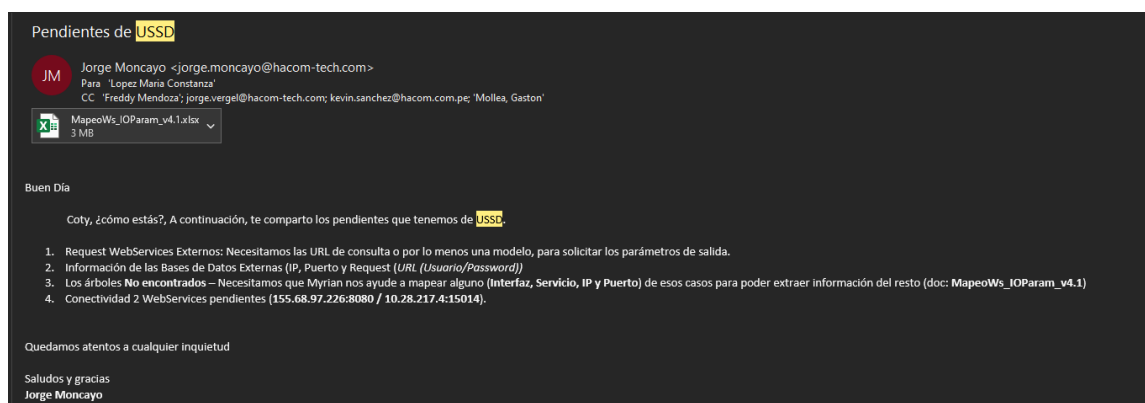
Figura 3
Servicios y Marcaciones

A	B	C	D	E
	Código	Descripción	Código de servicio	
1	APPOFFLINE	venta de bolsas a travez de APP OFFLINE	*701	
2	Amazon	AmazonPrime Video	*269	
3	BANCO1TEST	BANCO 1 TEST	*420	
4	BANCO2TEST	banco 2 test	*421	
6	BISA	BISA TEST	*422	
7	BNB	BNB mwallet	*262	
8	BilleteraM	BilleteraMovil	*222	
9	Blackberry	Planes blackberry	*111	
10	BolsaPromo	Bolas Promocionales	*513	
11	BolsasMovi	Bolsas movil direccional al *007	*512	
12	EIR	Registracion EIR	*108	
13	EstoyCero	Nuevo servicio Estoy Cero	*025	
14	ExtCredito	Extension de Credito *440	*440	
15	Extiendete	Servicio Extiendete	*277	
16	FacturaBD	Sistema de Actualización de Datos de Factura	*746	
17	FidePuntos	Fidepuntos	*426	
18	Flytxt	Flytxt WS con IT	*777	
19	GIS	Cuestionario GIS	*447	
20	HSS_0	TEST HSS0	*340	
21	HSS_1	HSS_1	*341	
22	HSS_2	HSS_2	*342	
23	HSS_3	HSS_3	*343	
24	HSS_4	HSS_4	*344	
25	HSS_5	TEST HSS	*345	
26	MEGABOLSAS	MEGABOLSAS	*040	
27	MenuPrinc	Menu Principal	*123	
28	PCRF_Regla	Consultas de reglas PCRF	*515	
29	PLANES3G	Planes 3G	*511	
30	PasameCreN	pasame credito navidad	*015	
31	PasameCred	Pásame Crédito	*009	
32	PlanNoctur	Plan nocturno	*499	
33	Prestamito	flujo de prestamo de 1 BS por defecto	*026	
34	R_B_T	RingBack Tone	*404	
35	RecarCred	Recarga de Crédito	*109	
36	Replan	Servicio de replan	*456	
Cover Instrucciones Check List A) General B) Cloud C) USSD D) Sistema de Gestión E) CONTADORES & KPI F) Soport				

Nota. Fuente: Cloud de la empresa

A partir de la figura se puede apreciar la relación de flujos USSD implementados en la plataforma, donde cada código responde a un servicio específico y diferenciado. Por ejemplo, el flujo denominado APP Offline (*701) no corresponde a una aplicación móvil en sí, sino que se utiliza como diferenciador interno para identificar el servicio de venta de bolsas de datos mediante USSD, sin requerir conexión a internet. De manera similar, se observa el flujo de Amazon Prime Video (*269), que habilita la gestión de suscripciones asociadas a este servicio; mientras que BANCO1TEST (*420) y BANCO2TEST (*421) representan entornos de prueba orientados a la validación de integraciones con entidades financieras, también se ven las diferentes ofertas y bolsas que ofrece como tales distintas marcaciones. Entonces se puede ver como en la figura se refleja la diversidad de escenarios que abarca la solución, desde productos de telecomunicaciones hasta servicios de entretenimiento y operaciones bancarias.

Figura 4
Relevación de información sobre WebServices



Nota. Fuente: Correo empresarial de la empresa

Dentro del levantamiento de requerimientos, uno de los puntos críticos identificados fue la necesidad de integrar el sistema USSD con diversos servicios externos, de los cuales dependían ciertos flujos XML vinculados a marcaciones específicas. Esta integración no podía asumirse de manera genérica, sino que requería información precisa sobre la infraestructura y las credenciales de acceso proporcionadas por la empresa cliente.

Tal como se refleja en la comunicación interna sostenida con el área técnica, se solicitó de manera formal la entrega de las URLs de consulta de los Web Services, así

Otro aspecto importante fue el mapeo de árboles de servicios no encontrados, que consistía en identificar aquellas interfaces, servicios, IPs y puertos que no habían sido documentados de manera inicial. Esta labor resultó clave para poder replicar el patrón de integración en el resto de flujos, evitando inconsistencias en la operación del sistema. Finalmente, se incluyó la verificación de conectividad hacia direcciones específicas de Web Services pendientes, lo cual formaba parte de la validación técnica antes de la puesta en marcha de los escenarios de prueba.

Figura 5
Vista grafica de una marcación con sus opciones



De la figura mostrada se observa que parte del estudio y recopilación de información a implementar como en este caso los webservices inmersos en los XML muchos de ellos no tenían información completa, como puertos, IP y otras cosas necesarias para hacer las peticiones satisfactorias a las APIs, entonces había que hacer un barrido general de todos los arboles de XML para hacer esta detección, nodo por nodo, servicio por servicio.

Esto es importante ya que al momento de desarrollar/codificar la solución era necesario tener claro la naturaleza de todos los servicios y ver cuales estaban faltando para ver que marcaciones junto con sus XMLs no iban a poder ser testeados por falta de información.

La captura mostrada pertenece a la representación gráfica de cada marcación a desarrollar con sus flujos completos, y en ellos las distintas acciones que realizar, esto esta detallado en archivos HTML como veremos a continuación.

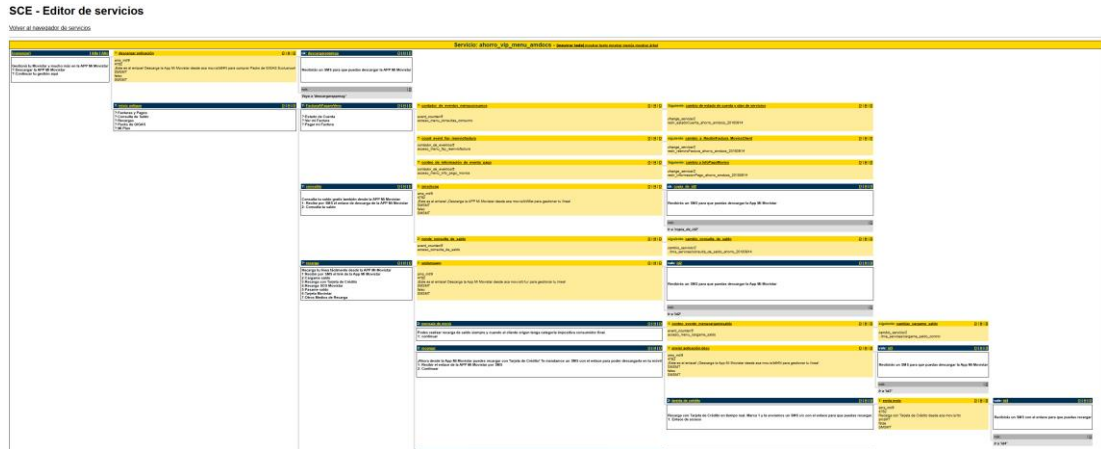
Figura 6
Vista de todos los árboles de las marcaciones a nivel grafico

svc_recharge_new_files	03/11/2021 17:46	Carpeta de archivos	
Terabox_313_files	03/11/2021 17:46	Carpeta de archivos	
Terabox_files	03/11/2021 17:46	Carpeta de archivos	
TestMT_files	03/11/2021 17:46	Carpeta de archivos	
vip_menu_files	03/11/2021 17:46	Carpeta de archivos	
vip_menu_new_files	03/11/2021 17:46	Carpeta de archivos	
voz_a_texto_313_files	03/11/2021 17:46	Carpeta de archivos	
voz_a_texto_files	03/11/2021 17:46	Carpeta de archivos	
voz_a_texto_original_files	03/11/2021 17:46	Carpeta de archivos	
Wanda_files	03/11/2021 17:46	Carpeta de archivos	
ahorro_consulta_consumos	17/08/2021 23:17	Opera GX Web Do...	1.036 KB
ahorro_empresas_menu	17/08/2021 23:17	Opera GX Web Do...	83 KB
ahorro_empresas_menu_313	17/08/2021 23:17	Opera GX Web Do...	79 KB
ahorro_empresas_menu_consumos	17/08/2021 23:17	Opera GX Web Do...	41 KB
ahorro_menu	17/08/2021 23:17	Opera GX Web Do...	190 KB
ahorro_menu_313	02/12/2021 17:06	Microsoft Edge H...	3 KB
ahorro_menu_amdocs	17/08/2021 23:17	Opera GX Web Do...	185 KB
ahorro_menu_amdocs_313	24/09/2021 18:00	Opera GX Web Do...	361 KB
ahorro_menu_amdocs_consumo	17/08/2021 23:17	Opera GX Web Do...	355 KB
ahorro_menu_consumo	17/08/2021 23:17	Opera GX Web Do...	90 KB
ahorro_vip_menu	17/08/2021 23:17	Opera GX Web Do...	79 KB
ahorro_vip_menu_313	17/08/2021 23:17	Opera GX Web Do...	173 KB
ahorro_vip_menu_amdocs	17/08/2021 23:17	Opera GX Web Do...	173 KB
ahorro_vip_menu_amdocs_313	24/09/2021 18:01	Opera GX Web Do...	348 KB
ahorro_vip_menu_consumos	17/08/2021 23:17	Opera GX Web Do...	388 KB
alta_numero_free	17/08/2021 23:17	Opera GX Web Do...	79 KB
alta_numero_free_313	18/08/2021 19:04	Opera GX Web Do...	3.375 KB
busca_amigos	18/08/2021 19:05	Opera GX Web Do...	3.419 KB
cambio_de_oferta	19/08/2021 19:02	Opera GX Web Do...	21 KB
cargame_nro_extra	17/08/2021 23:17	Opera GX Web Do...	344 KB
cargame_saldo_autoesponsoreo	19/08/2021 19:03	Opera GX Web Do...	189 KB
cargame_saldo_control	19/08/2021 19:05	Opera GX Web Do...	198 KB
cargame_saldo_full	19/08/2021 19:06	Opera GX Web Do...	226 KB
check_perfil_num_free	19/08/2021 19:08	Opera GX Web Do...	226 KB
color_sms	17/08/2021 23:17	Opera GX Web Do...	191 KB
compra_bono	19/08/2021 19:09	Opera GX Web Do...	25 KB
compra_de_packs	19/08/2021 19:10	Opera GX Web Do...	263 KB
compra_de_paquetes	19/08/2021 19:11	Opera GX Web Do...	88 KB
compra_de_paquetes_20180815	19/08/2021 19:12	Opera GX Web Do...	182 KB
compra_de_paquetes_20180903	19/08/2021 19:13	Opera GX Web Do...	182 KB
compra_de_paquetes_20180903_313	17/08/2021 23:17	Opera GX Web Do...	271 KB
compra_de_paquetes_20180903_armatup...	19/08/2021 19:15	Opera GX Web Do...	271 KB
compra_de_paquetes_20180903_ws_sinar...	19/08/2021 19:16	Opera GX Web Do...	171 KB
compra_de_paquetes_20191202	19/08/2021 19:17	Opera GX Web Do...	171 KB
compra_de_paquetes_20200217	19/08/2021 19:18	Opera GX Web Do...	190 KB
compra_de_paquetes_20200217_prueba	19/08/2021 19:19	Opera GX Web Do...	246 KB
compra_de_paquetes_20210722	19/08/2021 19:20	Opera GX Web Do...	288 KB
compra_de_paquetes_20210722_313_packs	19/08/2021 19:21	Opera GX Web Do...	298 KB
compra_de_paquetes_ahorro	19/08/2021 19:22	Opera GX Web Do...	298 KB
compra_de_paquetes_ahorro_313	19/08/2021 19:23	Opera GX Web Do...	284 KB
	19/08/2021 19:25	Opera GX Web Do...	193 KB
	19/08/2021 19:29	Opera GX Web Do...	179 KB

Nota. Fuente: Cloud de la empresa

Veamos a continuación partes de estos XML “trees” o árboles que en conjunto servían como indicadores de flujo o mapas de rutas, los cuales de forma grafica estaban representados en los HTML como se visualiza (con todos los ítems que contenían los menús por los que recorría, los recuadros amarillos implicaban saltos automáticos entre un nodo y otro según la opción que se iba seleccionando en el flujo USSD).

Figura 7
Flujo de saltos en una marcación

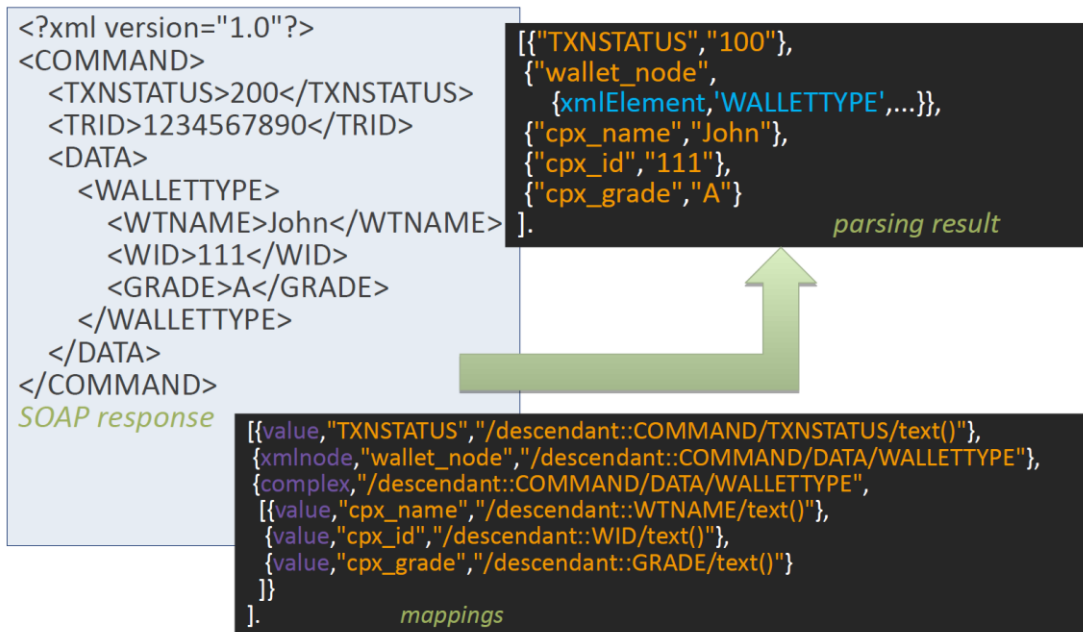


Nota. Fuente: Cloud de la empresa de telefonía

Otro punto muy importante durante el recojo de información es el formato de JSONs que devuelve un servicio al consultarlo, que puede ser en algunos casos un nodo simple o en otros casos un nodo complejo como en los siguientes ejemplos:

Figura 8
Estructura de los archivos MAPPING Y ERROR

svc_xml_generic:request/8
[Mapping File example: single node](#)



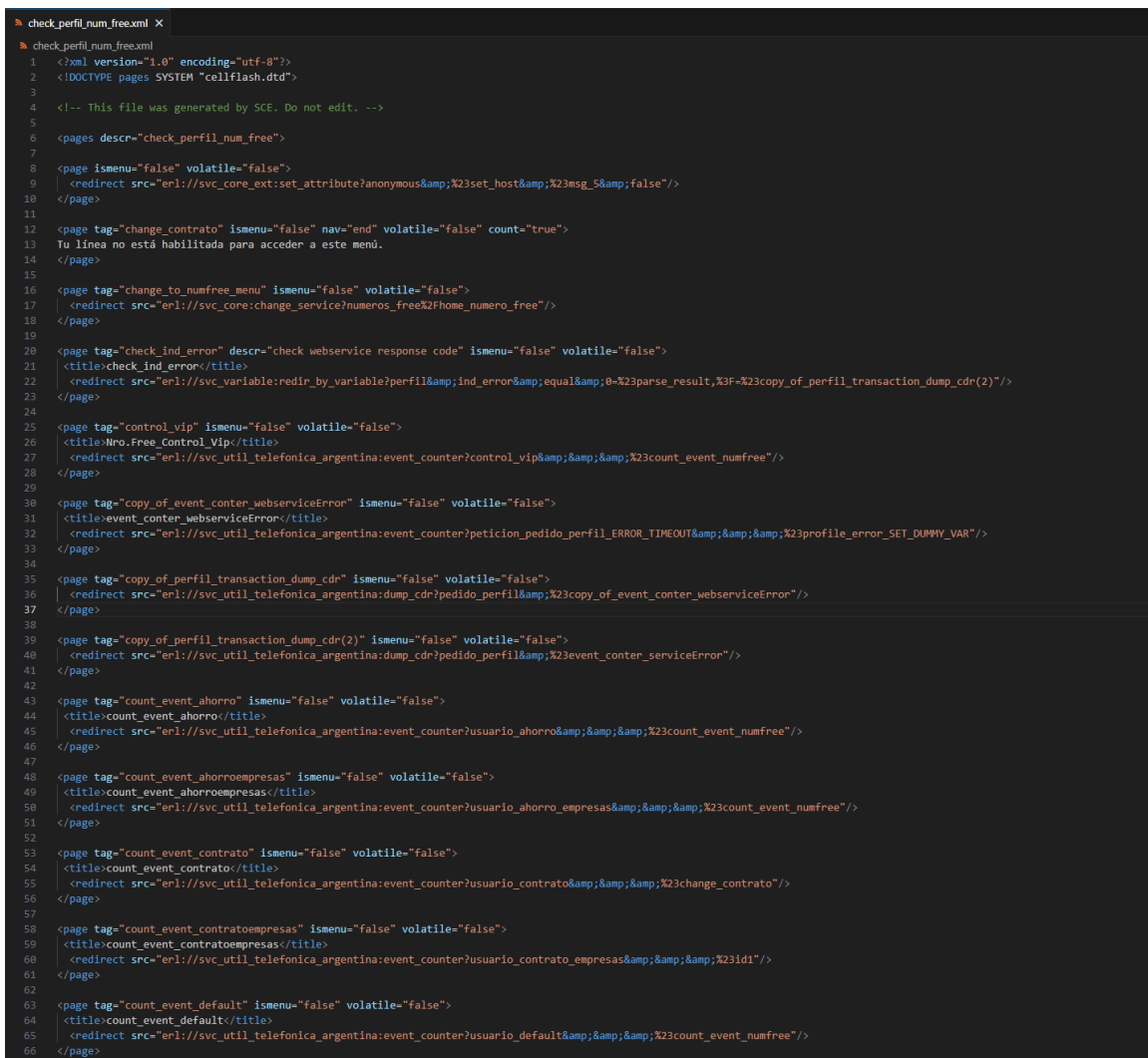
Nota. Fuente: Documentación de la empresa de telefonía

Como se ve en la figura hay 2 tipos de nodos, además del valor como tal representado por VALUE, que viene a ser el valor a extraer del XML devuelto por el WS(web service).

Cuando se refiere a un nodo complejo se hace la búsqueda hasta dar con el parámetro que esta solicitando el text(), para extraerle su valor representado en el XML(en ese ejemplo en específico para dar una demostración para el Tag de WTNAME el valor a extraer seria de Jhon).

Los valores obtenidos a partir de los XML generados en las consultas a los servicios web son transformados y almacenados en estructuras de tipo JSON, las cuales

Figura 9
Vista de un XML y su estructura



```
check_perfil_num_free.xml
check_perfil_num_free.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE pages SYSTEM "cellflash.dtd">
3
4 <!-- This file was generated by SCE. Do not edit. -->
5
6 <pages descr="check_perfil_num_free">
7
8 <page ismenu="false" volatile="false">
9 | <redirect src="erl://svc_core_ext:set_attribute?anonymous&X23set_host&X23msg_5&X23count_event_numfree"/>
10 </page>
11
12 <page tag="change_contrato" ismenu="false" nav="end" volatile="false" count="true">
13 | Tu línea no está habilitada para acceder a este menú.
14 </page>
15
16 <page tag="change_to_numfree_menu" ismenu="false" volatile="false">
17 | <redirect src="erl://svc_core:change_service?numeros_free%2Fhome_numero_free"/>
18 </page>
19
20 <page tag="check_ind_error" descr="check webservice response code" ismenu="false" volatile="false">
21 | <title>check_ind_error</title>
22 | <redirect src="erl://svc_variable:redir_by_variable?perfil&ind_error&equal&0~X23parse_result,X3F~X23copy_of_perfil_transaction_dump_cdr(2)"/>
23 </page>
24
25 <page tag="control_vip" ismenu="false" volatile="false">
26 | <title>Nro.Free_Control_Vip</title>
27 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?control_vip&&&X23count_event_numfree"/>
28 </page>
29
30 <page tag="copy_of_event_conter_webserviceError" ismenu="false" volatile="false">
31 | <title>event_conter_webserviceError</title>
32 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?peticion_pedido_perfil_ERROR_TIMEOUT&&&X23profile_error_SET_DUMMY_VAR"/>
33 </page>
34
35 <page tag="copy_of_perfil_transaction_dump_cdr" ismenu="false" volatile="false">
36 | <redirect src="erl://svc_util_telefonica_argentina:dump_cdr?pedido_perfil&X23copy_of_event_conter_webserviceError"/>
37 </page>
38
39 <page tag="copy_of_perfil_transaction_dump_cdr(2)" ismenu="false" volatile="false">
40 | <redirect src="erl://svc_util_telefonica_argentina:dump_cdr?pedido_perfil&X23event_conter_serviceError"/>
41 </page>
42
43 <page tag="count_event_ahorro" ismenu="false" volatile="false">
44 | <title>count_event_ahorro</title>
45 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?usuario_ahorro&&&X23count_event_numfree"/>
46 </page>
47
48 <page tag="count_event_ahorroempresas" ismenu="false" volatile="false">
49 | <title>count_event_ahorroempresas</title>
50 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?usuario_ahorro_empresas&&&X23count_event_numfree"/>
51 </page>
52
53 <page tag="count_event_contrato" ismenu="false" volatile="false">
54 | <title>count_event_contrato</title>
55 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?usuario_contrato&&&X23change_contrato"/>
56 </page>
57
58 <page tag="count_event_contratoempresas" ismenu="false" volatile="false">
59 | <title>count_event_contratoempresas</title>
60 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?usuario_contrato_empresas&&&X23id1"/>
61 </page>
62
63 <page tag="count_event_default" ismenu="false" volatile="false">
64 | <title>count_event_default</title>
65 | <redirect src="erl://svc_util_telefonica_argentina:event_counter?usuario_default&&&X23count_event_numfree"/>
66 </page>
67
```

Nota. Fuente: XML de la empresa telefónica

son posteriormente devueltas como respuesta dentro de la sesión USSD. Estos JSON se organizan en dos categorías principales: los denominados. MAPPING, asociados a respuestas exitosas, y los. ERROR, vinculados a escenarios en los que se produce alguna falla. Si bien difieren en la naturaleza de la información que contienen, ambos mantienen una **estructura homogénea**, lo que resulta especialmente relevante, ya que permite aplicar una misma lógica de desarrollo para su tratamiento y procesamiento.

En la Figura se observa un ejemplo de archivo XML que ilustra la lógica subyacente en la navegación de los servicios USSD. Cada sección del documento está compuesta por nodos denominados *pages*, los cuales representan estados o pasos específicos dentro del flujo. Dichos nodos contienen atributos relevantes, como *ismenu*, que permite distinguir si la página corresponde a un menú interactivo presentado al usuario o si, por el contrario, constituye un paso interno del proceso. Asimismo, los elementos `<redirect>` señalan la ruta que seguirá la interacción una vez alcanzada ese punto, haciendo referencia a servicios internos de la plataforma que se encuentran codificados en la propia URL.

Lo interesante de este esquema es que cada nodo cumple funciones diferenciadas: algunos despliegan mensajes informativos al usuario, otros redirigen hacia servicios web o módulos de negocio, mientras que ciertos nodos ejecutan operaciones internas como la transformación de parámetros, por ejemplo, la conversión del número MSISDN a formato nacional, o el conteo de eventos para fines de control. De este modo, el XML no es un documento estático, sino más bien un **mapa dinámico de navegación**, que define de manera precisa las transiciones posibles en la interacción.

La implementación desarrollada en Java Gradle interpreta este tipo de estructuras en tiempo real, resolviendo correctamente qué acción ejecutar en cada nodo. Esto implicaba procesar condiciones, invocar servicios externos y gestionar la continuidad del flujo de forma eficiente, garantizando una experiencia transparente para el usuario

final. En consecuencia, los XML se constituyen como un insumo fundamental dentro del análisis de requerimientos, dado que expresan con claridad la secuencia lógica que el sistema debe soportar y que, posteriormente, se traduce en reglas de negocio y funciones implementadas en el backend.

El punto final necesario para completar la recolección de datos a tener en cuenta en el desarrollo de la solución USSD era implementar todos los plugins en base a la documentación que manejaban a nivel de conceptos, para mejor entendimiento, a continuación, mostraremos un resumen de algunos plugin utilizados:

Figura 10
Vista de la tabla de contenido de los Plugin a implementar

Table of Contents

1. Introduction	1
1.1. Purpose	1
1.2. Environment	1
1.3. Scope	1
2. Core plugins	2
2.1. svc_core:incl_attribute/2	2
2.1.1. Overview	2
2.1.2. Parameters	2
2.1.2.1. "attr"	2
2.1.3. Fixed redirections	2
2.2. svc_core:redir_by_attribute/4	3
2.2.1. Overview	3
2.2.2. Parameters	3
2.2.2.1. "attr"	3
2.2.2.2. "match"	3
2.2.2.3. "patterns"	4
2.2.3. Fixed redirections	4
2.3. svc_core:store_pre_page/2	4
2.3.1. Overview	4
2.3.2. Parameters	5
2.3.2.1. "ok"	5
2.3.3. Fixed redirections	5
2.4. svc_core:goto_stored_page/2	5
2.4.1. Overview	5
2.4.2. Parameters	5
2.4.2.1. "nok"	5
2.4.3. Fixed redirections	5
2.5. svc_core:incl_by_attribute/4	5
2.5.1. Overview	5
2.5.2. Parameters	5
2.5.2.1. "attr"	5
2.5.2.2. "match"	6
2.5.2.3. "patterns"	6
2.5.3. Fixed redirections	7
2.6. svc_core:generic_form/5	7
2.6.1. Overview	7
2.6.2. Parameters	7
2.6.2.1. "filter"	7

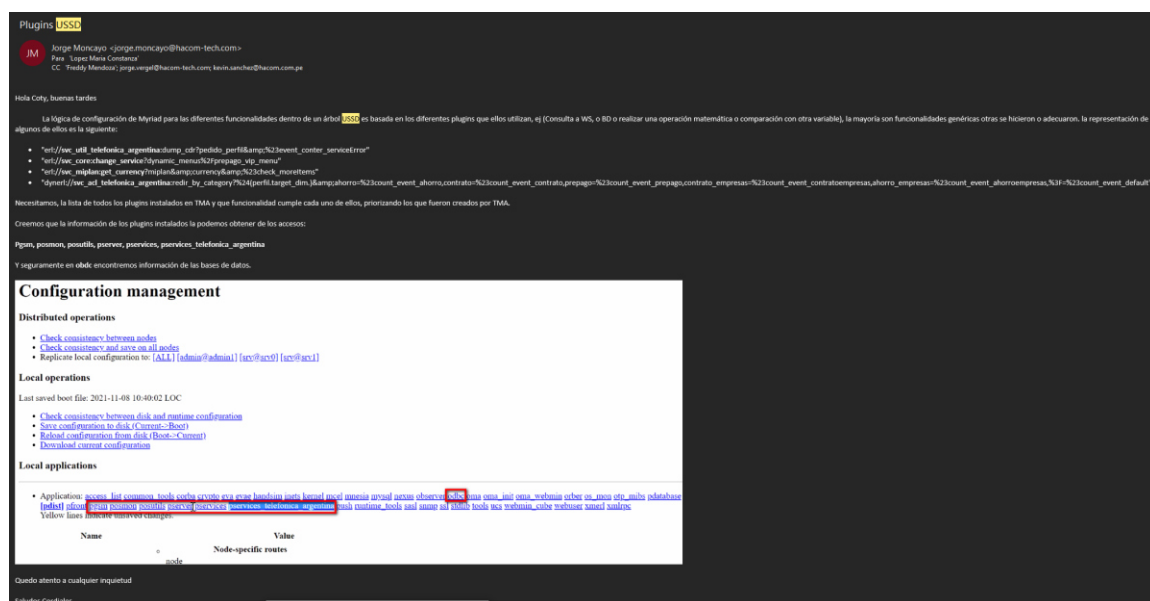
Nota. Fuente: Documentación de la empresa

Dichos plugins están creados cada uno para fines específicos, hay plugins para redirecciones especiales, redirecciones por fecha, para recargas, para detalles de la cuenta telefónica, para reembolsos, para pagos, para consultar a base de datos, etc.

Ahora si bien maneja una documentación general que especifica los atributos y para qué sirve cada uno de ellos junto con una breve descripción del plugin también hay plugins específicos que maneja la propia entidad de telefonía a la cual se le presto el servicio.

Para esta lista de plugins propios de la empresa tuvieron que hacerse reuniones adicionales para solicitar y aclarar cualquier duda que hubiera respecto a estos plugins.

Figura 11
Correo de la empresa



Nota. Fuente: Correo empresarial de la empresa.

Muchos de estos plugins propios o como la misma empresa telefónica le llama son los plugins customizados, que han tenido que ser descifrados ya que estaban hechos en Earlang, pero para nuestro nuevo backend serán incorporados junto con los plugin genéricos en la lógica desarrollada. Aquí se tuvo que analizar cada archivo y aplicar una

ingeniería inversa para poder descifrar en si que es lo que, hacia el plugin, una vez descifrada se hacia la descripción en documentos de texto para su recopilación y posterior codificación y desarrollo.

Figura 12
Mapeos de los parametros de un plugin

```
svc_monederos
=====

svc_monederos:add_with_exponent/5
-----

### URLs

`erl://svc_monederos:add_with_exponent?monederos&result&#monederos_menu&#msg_technical_error" `
`erl://svc_monederos:add_with_exponent?monederos&result&#opcion_menu&#generic_error" `
`erl://svc_monederos:add_with_exponent?monederos&result&#opcion_menu&#msg_technical_error" `
`erl://svc_monederos:add_with_exponent?quema&result&#muestra_resultOk_AA&#saldo_mensaje_error" `

```json
1> svc_monederos:add_with_exponent(plugin_info,[],[],[],[]).
{plugin_info,undefined,
 "This plugin calculate the cost with exponent when status is AC, and save the result in the new variable,\n",
 undefined,command,
 [{svc,{oma_type,string},
 "This parameter specify the service that used to store and obtain variables"},
 {result,{oma_type,string},
 "This parameter specify the result"},
 {ok,{link,[],"calculate successfully."},
 {nok,{link,[],"calculate failed."}}],
 []},undefined}
...

svc_monederos:decode_xml_attribute/5

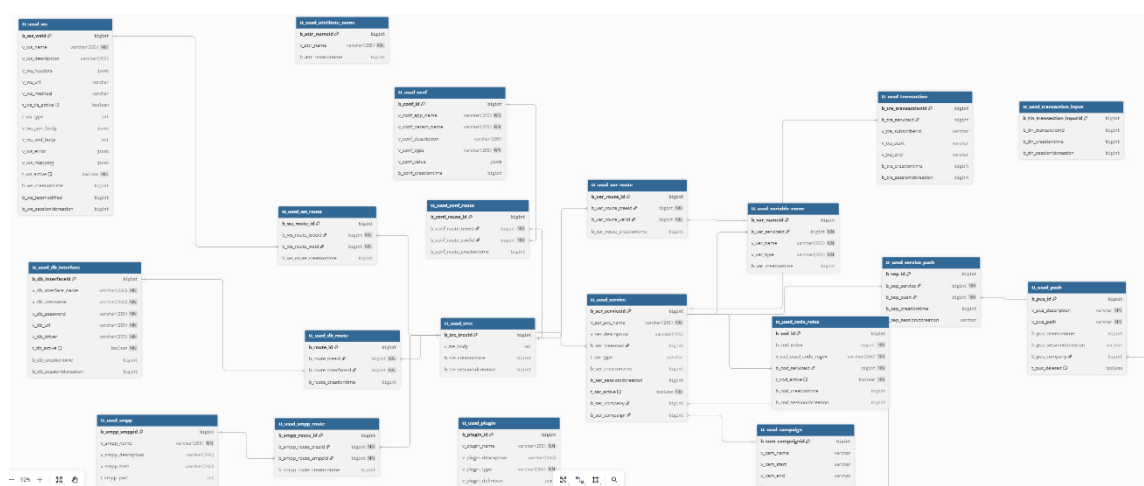
URLs

`erl://svc_monederos:decode_xml_attribute?monederos&boltonExponent&boltonExponent&#bono_menu_app" `
`erl://svc_monederos:decode_xml_attribute?monederos&boltonExponent&boltonExponent&#bonos_menu" `
`erl://svc_monederos:decode_xml_attribute?monederos&exponent&exponent&#calculate_with_exponent" `
`erl://svc_monederos:decode_xml_attribute?quema&exponent&exponent&#calculate_with_exponent" `
```

### 3.9 DISEÑO LÓGICO Y FÍSICO DEL SISTEMA

En la etapa de diseño, la elaboración de un Diagrama Entidad–Relación (DER) constituye un paso fundamental para garantizar la correcta estructuración de la base de datos, ya que permite representar de manera visual las entidades que intervienen en el sistema, sus atributos y las relaciones existentes entre ellas. Un buen planteamiento de este diagrama no solo facilita la comprensión integral del modelo de datos, sino que también asegura la coherencia, consistencia y escalabilidad de la solución a lo largo de su ciclo de vida. En el caso del sistema **USSD**, el diagrama mostrado concentra las tablas que administran tanto la lógica de los servicios como la interacción con componentes externos (Web Services, interfaces de base de datos y SMPP), además de la trazabilidad

**Figura 13**  
**Diagrama ER de la solución USSD**



*Nota.* Fuente: Elaboración Propia

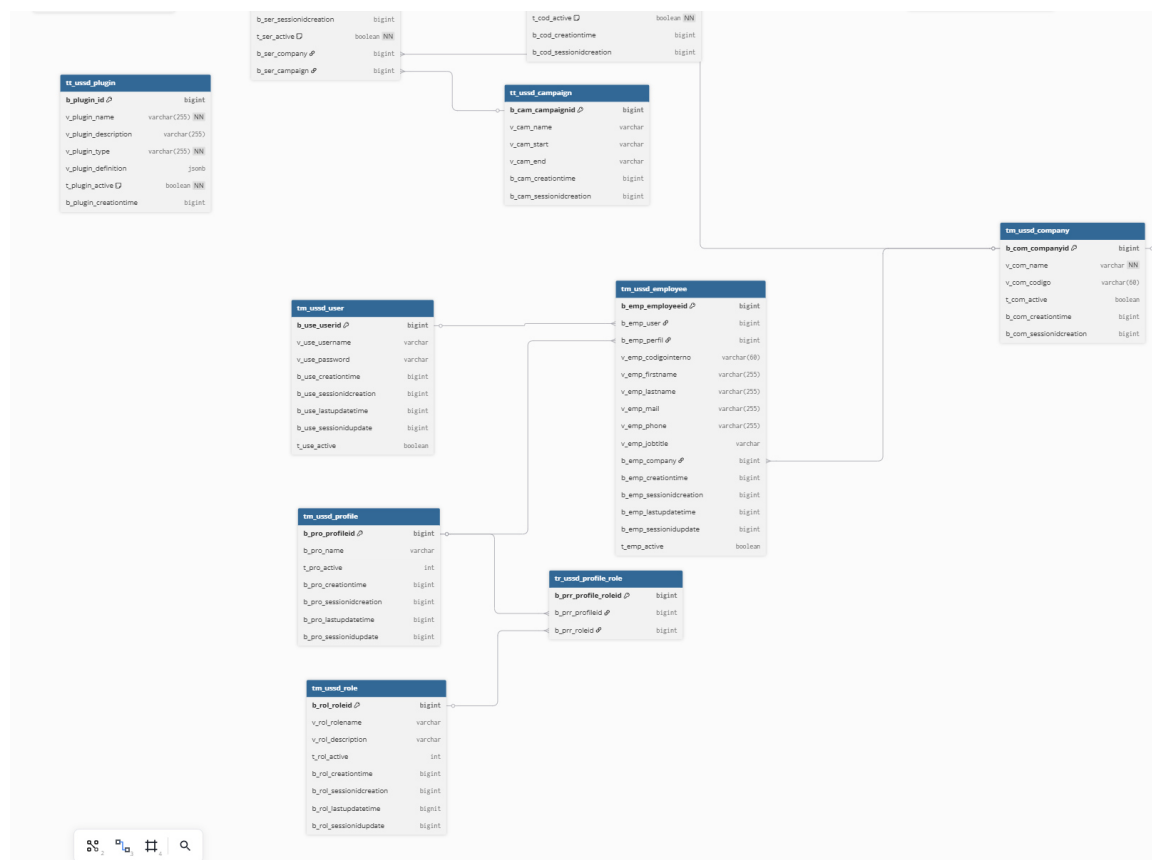
de las transacciones y configuraciones asociadas.

Dentro de esta primera sección del modelo se distinguen entidades clave como **tt\_usdd\_ws**, encargada de registrar la información de los servicios web (nombre, descripción, URL, método, cabeceras, cuerpos de petición/respuesta en JSON o XML, indicadores de estado, entre otros), o **tt\_usdd\_db\_interface**, que almacena los parámetros de conexión hacia bases de datos externas (nombre de la interfaz, usuario, contraseña, URL, driver). Asimismo, se identifican tablas como **tt\_usdd\_tree**, que representa la estructura del flujo de navegación en formato XML, **tt\_usdd\_service**,

destinada a detallar los servicios USSD habilitados (nombre, descripción, tipo, estado), y **tt\_ussd\_transaction**, que conserva la trazabilidad de las sesiones iniciadas por los usuarios (ID de transacción, servicio invocado, suscriptor, tiempos de inicio y fin). De igual modo, se incluyen tablas de soporte como **tt\_ussd\_variable\_name**, **tt\_ussd\_conf**, o **tt\_ussd\_plugin**, que permiten parametrizar y extender la lógica del sistema de forma flexible.

En conjunto, este diseño refleja cómo el sistema organiza los componentes que hacen posible la ejecución de los flujos USSD: desde la definición de rutas y configuraciones, hasta la gestión de variables, códigos de servicio y conexiones hacia terceros. Gracias a esta estructuración, la solución garantiza la interoperabilidad entre los diferentes módulos, así como un control preciso de cada interacción que ocurre durante la sesión del usuario.

**Figura 14**  
**Vistas complementaria del diagrama ER de la solución USSD**

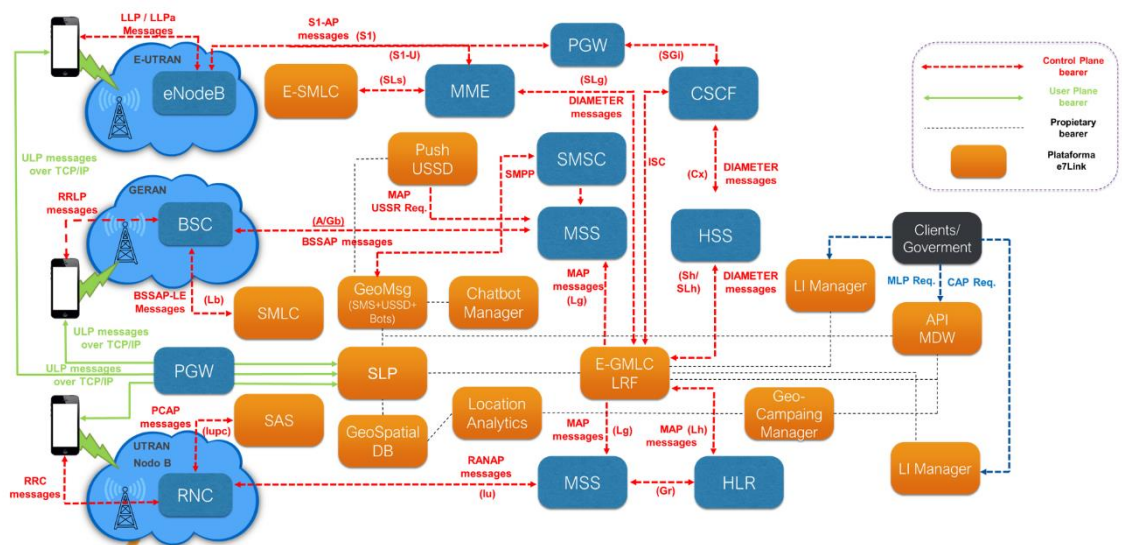


*Nota.* Fuente: Elaboración Propia

En esta segunda sección del diagrama se observa el módulo de gestión de usuarios, perfiles, roles y compañías, el cual resulta esencial para garantizar la seguridad, trazabilidad y administración de los accesos dentro del sistema USSD. El planteamiento de estas tablas permite establecer un control robusto de autenticación y autorización, asegurando que cada usuario disponga únicamente de las funciones necesarias de acuerdo con su perfil asignado.

Entre las tablas principales se encuentra **tm\_ussd\_user**, donde se registran las credenciales y el estado de los usuarios (nombre de usuario, contraseña, fechas de creación y última actualización, entre otros). Relacionada con ella, la tabla **tm\_ussd\_employee** amplía la información con datos personales y laborales (nombre, apellido, correo, teléfono, puesto y compañía a la que pertenece), vinculando así la identidad técnica con la información organizacional. Por otro lado, la entidad **tm\_ussd\_profile** define los distintos perfiles disponibles en el sistema, mientras que **tm\_ussd\_role** especifica los roles con sus descripciones y atributos de activación. La relación entre ambos se gestiona a través de la tabla intermedia **tr\_ussd\_profile\_role**, que habilita la flexibilidad para asignar múltiples roles a un mismo perfil según las necesidades operativas.

**Figura 15**  
**Diseño integral de la solución telefónica**



*Nota.* Fuente: Cloud de la empresa

Finalmente, la tabla **tm\_ussd\_company** centraliza la información de las compañías que utilizan o administran el servicio, lo cual refuerza la capacidad multiempresa del sistema. En conjunto, esta parte del diseño asegura que el sistema USSD no solo atienda la lógica transaccional y de servicios, sino que también disponga de un esquema de control organizacional alineado con las mejores prácticas de **gobernanza de datos y seguridad de la información**.

El diagrama ilustra la interacción entre distintos elementos de la red de telecomunicaciones, mostrando cómo los servicios móviles dependen de una infraestructura interconectada que integra tanto la capa de señalización como la de datos. En este contexto, el servicio USSD (Unstructured Supplementary Service Data) se posiciona como un componente clave, al permitir la comunicación directa entre el usuario y las aplicaciones del operador sin necesidad de acceso a Internet. A través de la pasarela de *Push USSD* y su vínculo con módulos como el *Chatbot Manager* o la mensajería (*GeoMsg*), se gestionan solicitudes y respuestas en tiempo real, lo que habilita al usuario final a interactuar con menús dinámicos para consultas, activación de servicios o acceso a funcionalidades avanzadas.

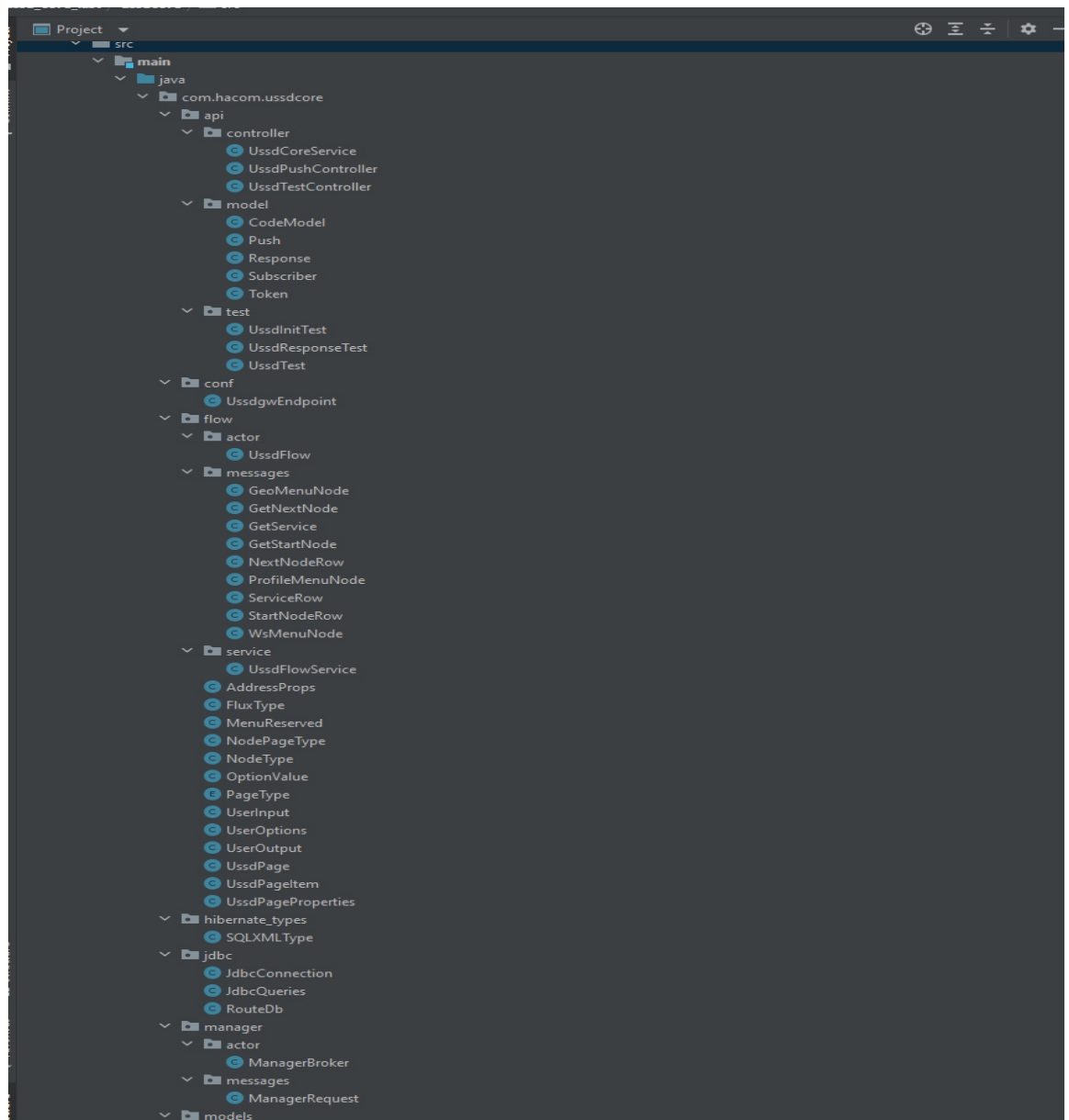
Su importancia radica en la capacidad de integrarse con diferentes nodos y plataformas, como el *MSS*, *HSS* o *E-GMLC*, garantizando que las solicitudes USSD se procesen de manera rápida y confiable. Este nivel de interconexión no solo asegura continuidad en la experiencia del usuario, sino que también permite a la empresa operadora implementar soluciones innovadoras como chatbots, campañas geolocalizadas o análisis de comportamiento. En suma, el USSD actúa como un puente entre la red central y las aplicaciones de valor agregado, representando un recurso estratégico para mantener la accesibilidad y simplicidad de los servicios en entornos donde la conectividad de datos puede ser limitada.

Modelo de la solución en Java Gradle

El sistema desarrollado bajo el framework Java Gradle fue organizado en distintos paquetes que responden a principios de modularidad, separación de responsabilidades y escalabilidad. Cada módulo agrupa clases con funciones específicas que, en conjunto, permiten la interpretación de flujos XML y la gestión de sesiones USSD en tiempo real.

En esta primera parte se ve la primera parte de los paquetes tal como fueron estructurados según su función.

**Figura 16**  
**Vista de los paquetes de la solución USSD**



*Nota.* Fuente: Código fuente del proyecto desarrollado

Como se observa en el gráfico, para un mayor entendimiento sintetizado se elaboro una tabla que señala las clases principales, los paquetes y subpaquetes así como la descripción académica de cada una de ellas.

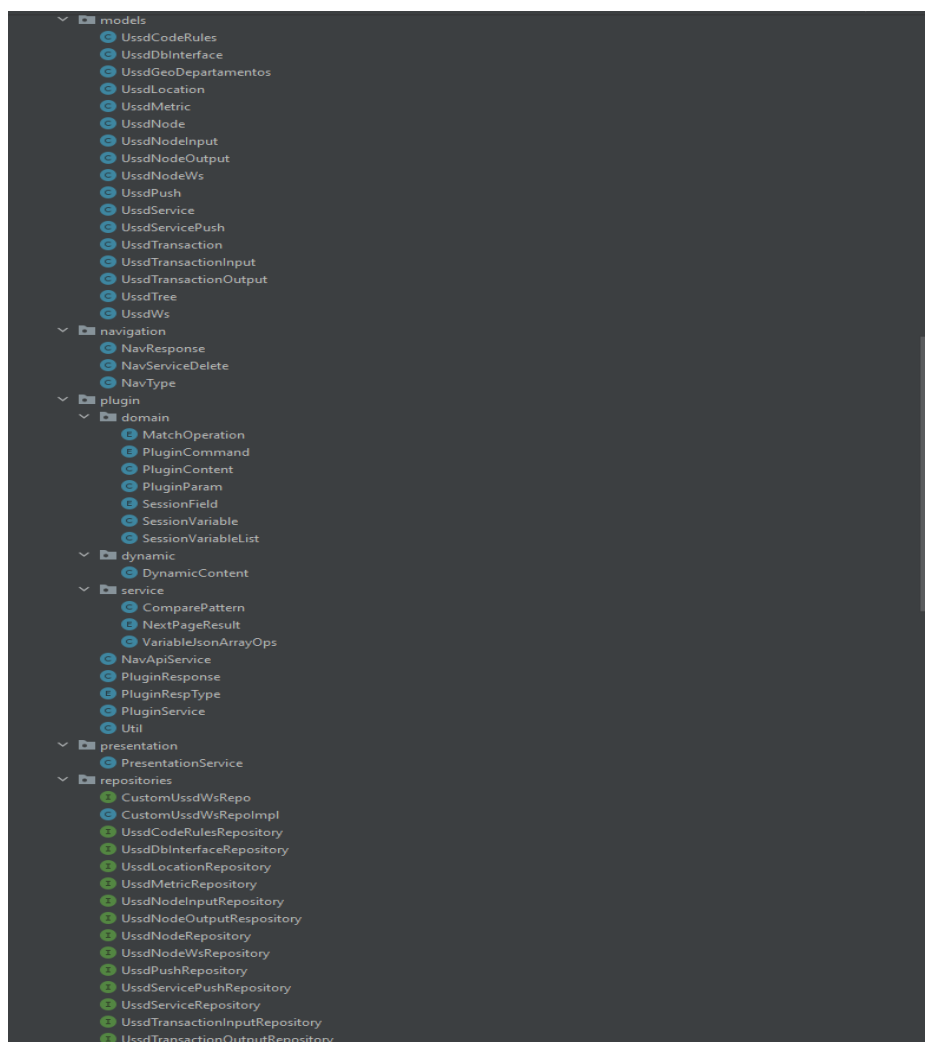
**Tabla 6**  
**Principales paquetes de la solución USSD primera parte**

<b>Paquete / Subpaquete</b>	<b>Clases principales</b>	<b>Descripción académica</b>
<b>api.controller</b>	UssdCoreService, UssdPushController, UssdTestController	Controladores que exponen los servicios centrales del backend. Permiten gestionar solicitudes entrantes en producción y entornos de prueba.
<b>api.model</b>	CodeModel, Push, Response, Subscriber, Token	Modelos de datos que representan las estructuras transmitidas en las interacciones USSD.
<b>api.test</b>	UssdInitTest, UssdResponseTest, UssdTest	Pruebas unitarias e integrales que validan la correcta ejecución de los flujos definidos.
<b>conf</b>	UssdgWEndpoint	Configuración crítica del gateway de USSD, que actúa como punto de entrada a la red de telecomunicaciones.
<b>flow</b>	UssdFlow	Clase que implementa el actor principal, encargado de orquestar la interacción entre nodos del flujo USSD.
<b>flow.messages</b>	GeoMenuNode, GetNextNode, ServiceRow, WsMenuNode	Estructuras de navegación que modelan pasos intermedios de los menús dinámicos USSD.
<b>service</b>	UssdFlowService, NodeType, PageType, UserInput, UserOutput, UssdPageProperties	Núcleo de interpretación de flujos XML. Gestiona sesiones y parametriza los distintos elementos de los menús USSD.
<b>hibernate_types</b>	SQLXMLType	Extiende Hibernate para el manejo de datos XML almacenados en bases de datos.
<b>jdbc</b>	JdbcConnection, JdbcQueries, RouteDb	Implementa la capa de acceso a datos mediante JDBC: conexión, ejecución de consultas y enrutamiento de peticiones hacia BD externas.

<b>manager</b>	ManagerBroker, ManagerRequest	Define actores y mensajes que coordinan la comunicación entre servicios internos y externos.
<b>models</b>	UssdCodeRules, UssdDbInterface, UssdGeoDepartamentos, UssdLocation, UssdMetric	Entidades del sistema USSD: reglas de codificación, conexiones a BD externas, datos geográficos y métricas de uso.

**Nota.** Fuente: Elaboración Propia

**Tabla 7**  
Principales paquetes de la solución USSD segunda parte



**Nota.** Fuente: Código fuente del proyecto desarrollado

También se definen entidades críticas para la operación, como `UssdTransaction` (entrada y salida), `UssdService`, `UssdPush`, `UssdWs` y `UssdTree`, que modelan respectivamente las transacciones, los servicios expuestos y los árboles XML de navegación.

**Tabla 8**  
**Paquetes de la segunda parte del proyecto USSD**

<b>Paquete / Subpaquete</b>	<b>Clases principales</b>	<b>Descripción académica</b>
<b>navigation</b>	<code>NavResponse</code> , <code>NavServiceDelete</code> , <code>NavType</code>	Gestiona el flujo de navegación dentro de sesiones USSD. Estandariza respuestas, controla finalización de rutas y clasifica tipos de navegación.
<b>plugin.domain</b>	<code>PluginCommand</code> , <code>PluginContent</code> , <code>PluginParam</code> , <code>SessionField</code> , <code>SessionVariable</code>	Define los elementos centrales de los plugins y estructuras de sesión que permiten personalización dinámica en los flujos.
<b>plugin.dynamic</b>	<code>DynamicContent</code>	Reemplaza valores dinámicos en XML, normalmente obtenidos desde consultas a Web Services.
<b>plugin.service</b>	<code>PluginService</code> , <code>PluginResponse</code> , <code>ComparePattern</code> , <code>NextPageResult</code> , <code>VariableJsonArrayOps</code>	Gestiona la interpretación de parámetros y la navegación condicional entre nodos a través de servicios reutilizables.
<b>presentation</b>	<code>PresentationService</code>	Orquesta la lógica de presentación al usuario, asegurando coherencia entre flujos XML, reglas de negocio e interfaz USSD.
<b>repositories</b>	<code>UssdNodeRepository</code> , <code>UssdTransactionInputRepository</code> , <code>UssdServiceRepository</code> ,	Implementa la capa de persistencia con Spring Data y

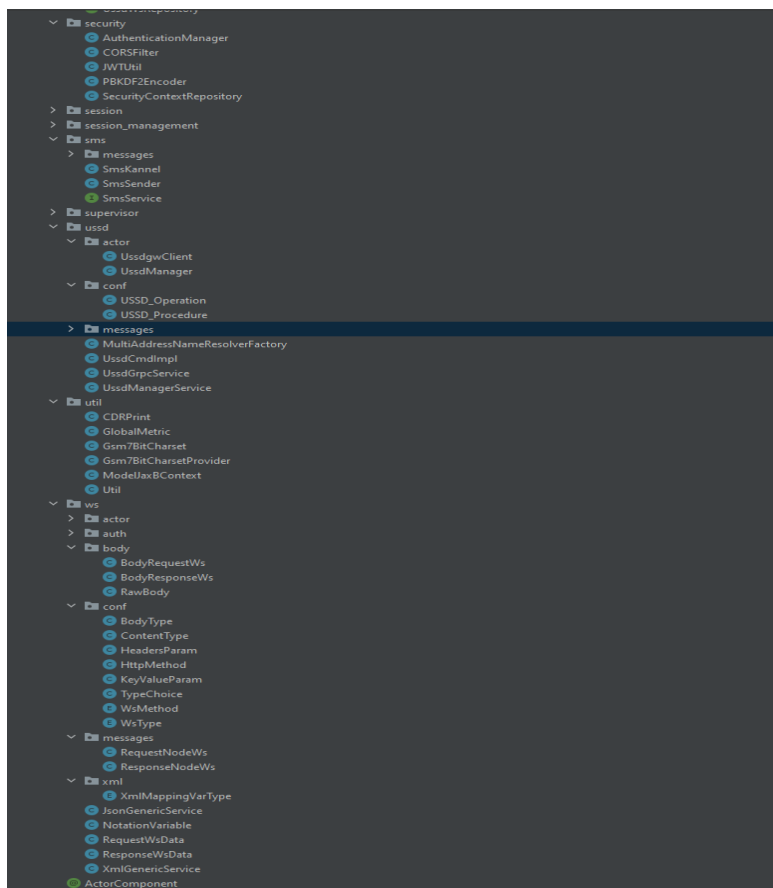
	CustomUssdWsRepo, CustomUssdWsRepoImpl	repositorios personalizados para consultas avanzadas.
<b>security</b>	AuthenticationManager, CORsFilter, JWTUtil, PBKDF2Encoder, SecurityContextRepository	Proporciona autenticación, validación de tokens y codificación de contraseñas, manteniendo la seguridad en la gestión de sesiones USSD.
<b>session.session_management</b>		Aplica políticas de inicio, mantenimiento y cierre de sesiones, garantizando control y límites de tiempo en cada transacción.
<b>sms</b>	SmsKannel, SmsSender, SmsService	Integra servicios SMS como complemento a los flujos USSD, habilitando notificaciones y comunicación híbrida.
<b>supervisor</b>	UssdgWClient, UssdManager	Monitorea procesos internos y gestiona conexiones con el gateway USSD. En <i>conf</i> se estructuran operaciones estandarizadas (USSD_Operation, USSD_Procedure).
<b>messages</b>	UssdCmdImpl, UssdGrpcService, UssdManagerService	Administra el intercambio de mensajes internos del backend, asegurando consistencia entre módulos.
<b>util</b>	CDRPrint, GlobalMetric, ModelJaxbContext	Herramientas de soporte para registrar métricas, manejar codificación y procesar mensajes en sistemas de alta concurrencia.
<b>ws.body</b>	BodyRequestWs, BodyResponseWs, RawBody	Modela el cuerpo de peticiones y respuestas en la integración con

*Nota.* Fuente: Elaboración Propia

En conf, clases como `BodyType`, `HeadersParam` o `WsType` permiten configurar los distintos tipos de mensajes y métodos de comunicación.

El subpaquete `xml` incluye `XmlMappingVarType`, `RequestWsData` y `ResponseWsData`, que manejan la transformación de estructuras XML/JSON en objetos internos, garantizando interoperabilidad con APIs externas.

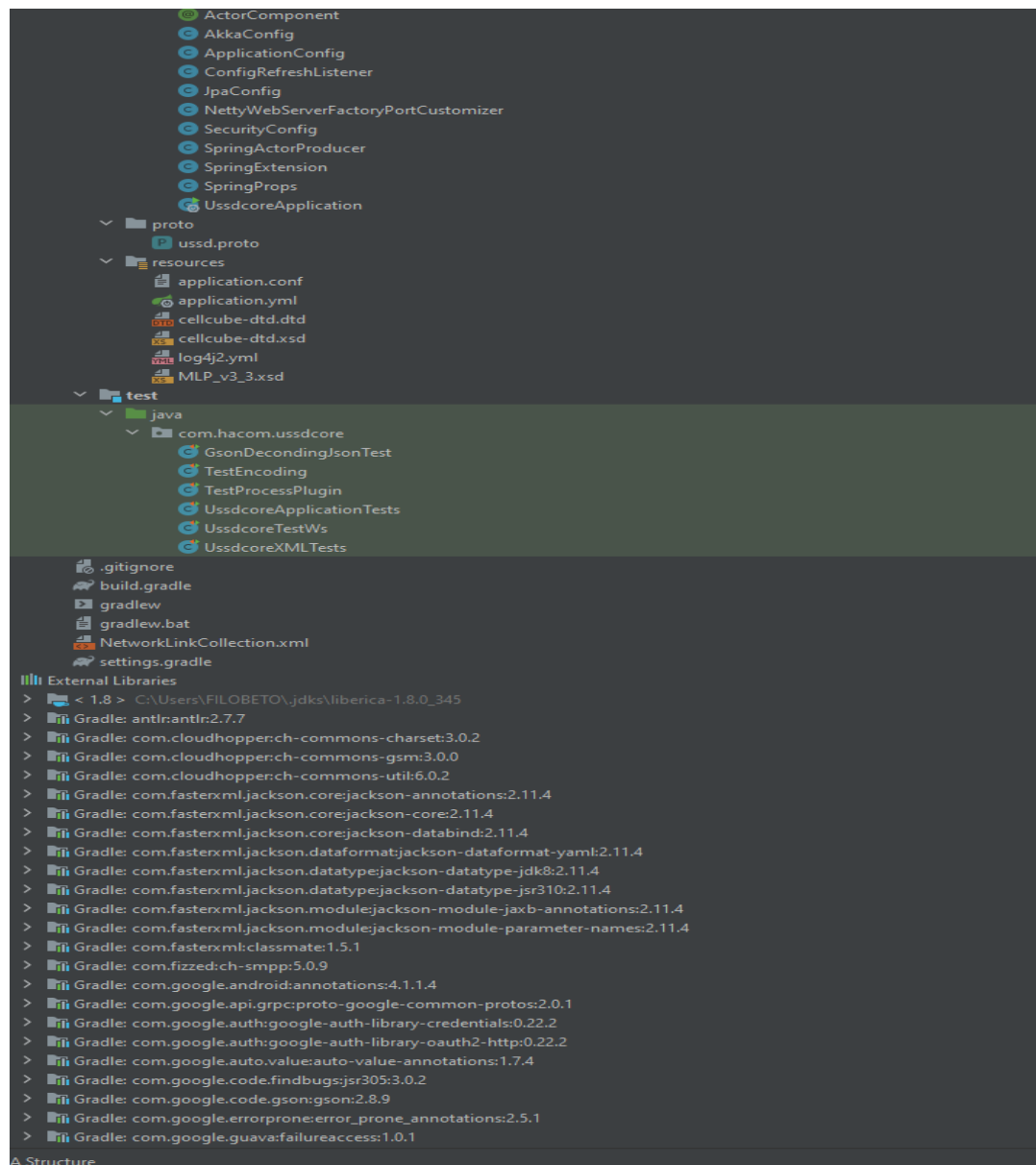
**Figura 17**  
Principales paquetes de la solución USSD tercera parte



*Nota.* Fuente: Código fuente del proyecto desarrollado

Finalmente, se destacan servicios como XmlGenericService y JsonGenericService, que generalizan el consumo de servicios, reduciendo la necesidad de código repetitivo.

**Figura 18**  
**Principales paquetes de la solución USSD cuarta parte**



*Nota.* Fuente: Código fuente del proyecto desarrollado

### **3.9.1 Configuración principal y arranque de la aplicación**

- Este bloque contiene clases que definen la configuración global del sistema y los parámetros de inicio.
- `ApplicationConfig`, `JpaConfig`, `SpringProps` y `SecurityConfig` se encargan de establecer la base de la aplicación, configurando conexiones a base de datos, seguridad y propiedades generales.
- Componentes como `AkkaConfig`, `SpringActorProducer` y `SpringExtension` evidencian la integración entre Spring y Akka, lo que permite manejar concurrencia a través de actores distribuidos.
- Finalmente, la clase `UssdcoreApplication` actúa como el punto de entrada, inicializando la ejecución del backend y orquestando los módulos necesarios para gestionar las sesiones USSD.

#### **3.9.1.1 Proto**

- El archivo `ussd.proto` define los contratos de comunicación utilizando gRPC, lo que facilita la interoperabilidad con otros sistemas de telecomunicaciones.

- Este enfoque garantiza que las solicitudes y respuestas se intercambien bajo un esquema binario eficiente, mejorando el rendimiento en escenarios de alta concurrencia.

- **3.9.1.2 Recursos**

- Aquí se incluyen archivos de configuración y esquemas que permiten parametrizar la solución sin modificar el código.
- `application.conf` y `application.yml` concentran las variables de entorno y propiedades de despliegue.
- Archivos como `log4j2.yml` gestionan la traza de ejecución y auditoría, mientras que los esquemas DTD/XSD (por ejemplo, `cellcube.dtd` y `MLP_v3_3.xsd`) garantizan la validación de los XML utilizados en los flujos.

### **3.9.1.3 Pruebas (test)**

- Este paquete confirma el compromiso con la calidad y validación del software.

- Se incluyen clases como `UssdcoreApplicationTests`, `UssdcoreXMLTests` y `TestProcessPlugin`, que permiten comprobar desde la inicialización del sistema hasta la correcta ejecución de plugins y procesamiento de flujos.
- Otras pruebas, como `GsonDecodingJsonTest` y `TestEncoding`, validan la correcta serialización y deserialización de datos en formatos JSON/XML, fundamentales para la interoperabilidad.

### 3.9.1.4 Dependencias externas y librerías principales del proyecto

**Tabla 9**  
Principales librerías y dependencias del proyecto

Librería / Dependencia	Descripción funcional	Relevancia en el proyecto
<b>Jackson</b> ( <code>fasterxml.jackson</code> )	Serialización y deserialización de datos en JSON y XML.	Permite procesar flujos USSD representados en XML y transformarlos en objetos internos.
<b>Hibernate / Hibernate Types</b>	Framework ORM para el mapeo objeto-relacional.	Facilita la persistencia de transacciones y sesiones en la base de datos.
<b>HikariCP</b>	Gestor de <i>connection pooling</i> de alto rendimiento.	Optimiza las conexiones a base de datos en entornos de alta concurrencia.
<b>Akka</b>	Implementación del modelo de actores para concurrencia.	Gestiona múltiples sesiones USSD simultáneas de forma eficiente.
<b>gRPC (grpc-core, grpc-stub, grpc-protobuf)</b>	Framework de comunicación basado en contratos binarios.	Garantiza interoperabilidad con servicios internos y externos.
<b>Netty</b>	Framework de comunicación asíncrona NIO.	Maneja transmisión de mensajes en tiempo real dentro del backend.

<b>OkHttp / Okio</b>	Cliente HTTP/HTTPS liviano y eficiente.	Permite consumir APIs externas durante las transacciones USSD.
<b>JSON Web Token (JWT)</b>	Gestión de autenticación y autorización.	Asegura el acceso controlado a los servicios internos.
<b>Micrometer / Prometheus Client</b>	Recolección de métricas de desempeño.	Permite monitorear tiempos de respuesta y carga del sistema.
<b>Swagger</b>	Documentación automática de APIs REST.	Facilita la integración con terceros y el mantenimiento del sistema.
<b>Zipkin (Brave, Zipkin-reporter)</b>	<i>Distributed tracing</i> para observabilidad.	Permite rastrear las transacciones a través de diferentes módulos.
<b>Apache Commons (lang, text, csv, httpclient)</b>	Utilitarios para manipulación de cadenas, archivos y peticiones HTTP.	Reduce complejidad en operaciones comunes y aumenta la productividad.
<b>Log4j2</b>	Framework de logging y auditoría.	Garantiza trazabilidad y monitoreo de eventos del sistema.
<b>BouncyCastle / Conscrypt</b>	Librerías criptográficas avanzadas.	Refuerzan la seguridad en la transmisión y almacenamiento de datos.

**Figura 19**  
**Librerías del proyecto**

```
> Gradle: com.jayway.jsonpath:json-path:2.4.0
> Gradle: com.oracle.database.jdbc:ojdbc8:19.8.0.0
> Gradle: com.squareup.okhttp:okhttp:2.7.4
> Gradle: com.squareup.okio:okio:1.17.5
> Gradle: com.sun.activation:jakarta.activation:1.2.2
> Gradle: com.sun.istack:istack-commons-runtime:3.0.12
> Gradle: com.thoughtworks.xstream:xstream:1.4.2
> Gradle: com.typesafe.akka:akka-actor-typed_2.13:2.6.0
> Gradle: com.typesafe.akka:akka-actor_2.13:2.6.0
> Gradle: com.typesafe.akka:akka-slf4j_2.13:2.6.0
> Gradle: com.typesafe.akka:akka-testkit_2.13:2.6.0
> Gradle: com.typesafe.config:1.4.0
> Gradle: com.vaadin.external.google:android-json:0.0.20131108.vaadin1
> Gradle: com.vividsolutions:jts:1.13
> Gradle: com.vladmihalcea:hibernate-types-52:2.9.13
> Gradle: com.zaxxer:HikariCP:3.4.5
> Gradle: commons-codec:commons-codec:1.15
> Gradle: io.github.classgraph:classgraph:4.8.116
> Gradle: io.grpc:grpc-ali:1.37.0
> Gradle: io.grpc:grpc-alts:1.37.0
> Gradle: io.grpc:grpc-api:1.37.0
> Gradle: io.grpc:grpc-auth:1.37.0
> Gradle: io.grpc:grpc-context:1.37.0
> Gradle: io.grpc:grpc-core:1.37.0
> Gradle: io.grpc:grpc-grpclb:1.37.0
> Gradle: io.grpc:grpc-netty:1.37.0
> Gradle: io.grpc:grpc-netty-shaded:1.37.0
> Gradle: io.grpc:grpc-okhttp:1.37.0
> Gradle: io.grpc:grpc-protobuf:1.37.0
> Gradle: io.grpc:grpc-protobuf-lite:1.37.0
> Gradle: io.grpc:grpc-rfs:1.37.0
> Gradle: io.grpc:grpc-services:1.37.0
> Gradle: io.grpc:grpc-stub:1.37.0
> Gradle: io.grpc:grpc-testing:1.37.0
> Gradle: io.grpc:grpc-xds:1.37.0
> Gradle: io.jsonwebtoken:jjwt-api:0.11.2
> Gradle: io.jsonwebtoken:jjwt-impl:0.11.2
> Gradle: io.jsonwebtoken:jjwt-jackson:0.11.2
> Gradle: io.micrometer:micrometer-core:1.6.13
> Gradle: io.micrometer:micrometer-registry-prometheus:1.6.13
> Gradle: io.netty.incubator:netty-incubator-codec-quicklinux-x86_64:0.0.20.Final
> Gradle: io.netty:netty:3.9.6.Final
> Gradle: io.netty:netty-buffer:4.1.70.Final
> Gradle: io.netty:netty-codec:4.1.70.Final
> Gradle: io.netty:netty-codec-dns:4.1.70.Final
> Gradle: io.netty:netty-codec-http:4.1.70.Final
> Gradle: io.netty:netty-codec-http2:4.1.70.Final
> Gradle: io.netty:netty-codec-socks:4.1.70.Final
> Gradle: io.netty:netty-common:4.1.70.Final
> Gradle: io.netty:netty-handler:4.1.70.Final
> Gradle: io.netty:netty-handler-proxy:4.1.70.Final
> Gradle: io.netty:netty-resolver:4.1.70.Final
> Gradle: io.netty:netty-resolver-dns:4.1.70.Final
> Gradle: io.netty:netty-resolver-dns-classes-macos:4.1.70.Final
> Gradle: io.netty:netty-resolver-dns-native-macos:4.1.70.Final
> Gradle: io.netty:netty-transport:4.1.70.Final
> Gradle: io.netty:netty-transport-classes-epoll:4.1.70.Final
> Gradle: io.netty:netty-transport-native-epoll:4.1.70.Final
> Gradle: io.netty:netty-transport-native-unix-common:4.1.70.Final
> Gradle: io.opencensus:opencensus-api:0.24.0
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

### 3.9.2 Implementación

Principales clases y componentes de la solución

La clase `UssdCoreService`, ubicada en el paquete `ussdcore.api.controller`, constituye uno de los puntos de entrada más relevantes del backend. Mediante la

anotación `@RestController`, expone servicios web accesibles a través de peticiones HTTP, lo que permite a clientes externos interactuar directamente con la lógica del sistema. Esta capa funciona como interfaz de comunicación entre las solicitudes externas y el núcleo de procesamiento USSD.

En su implementación, la clase define tres operaciones principales:

- `getStatus()`: un método de tipo GET que retorna un objeto `Response`, útil para verificar la disponibilidad del servicio.
- `addService()`: método de tipo POST que recibe un objeto `CodeModel` en formato JSON, lo registra en el sistema y actualiza las métricas globales mediante la clase `GlobalMetric`.
- `deleteService()`: método de tipo PUT que elimina un código USSD previamente registrado, asegurando que la aplicación pueda actualizar su catálogo de servicios de forma dinámica.

**Figura 20**  
**Vista del service principal**

```
package com.hacom.ussdcore.api.controller;

import com.hacom.ussdcore.api.model.CodeModel;
import com.hacom.ussdcore.api.model.Response;
import com.hacom.ussdcore.util.GlobalMetric;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Mono;

import javax.validation.Valid;

@RestController
@RequestMapping("/USSDCore")
public class UssdCoreService {
 private static final Logger logger = LogManager.getLogger(UssdCoreService.class);

 @GetMapping(path = "/getStatus", produces = "application/json")
 private Mono<Response> getStatus () { return Mono.just(new Response(status: "Okkey")); }

 @PostMapping(path = "/addCode", consumes = "application/json", produces = "application/json")
 private Mono<Response> addService(@Valid @RequestBody CodeModel ussdCode) {
 logger.info("Add Code Service : " + ussdCode);
 GlobalMetric.addUssd_code(ussdCode.getUssdCode());
 return Mono.just(new Response(status: "accepted"));
 }

 @PutMapping (path = "/deleteCode", consumes = "application/json", produces = "application/json")
 private Mono<Response> deleteService(@Valid @RequestBody CodeModel ussdCode) {
 logger.info("Delete Code Service : " + ussdCode);
 GlobalMetric.deleteUssd_code(ussdCode.getUssdCode());
 return Mono.just(new Response(status: "accepted"));
 }
}
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

El uso de `Mono<Response>` revela que se ha optado por un enfoque reactivo y no bloqueante, lo cual es esencial en escenarios donde se manejan múltiples solicitudes concurrentes. Además, la integración con Log4j2 (logger) permite registrar cada transacción, garantizando trazabilidad y facilitando la auditoría de las operaciones realizadas.

Este fragmento evidencia lo siguiente: validación de entradas con `@Valid`, registro de métricas, y retorno reactivo de respuestas. En conjunto, `UssdCoreService` constituye la puerta de acceso principal del backend USSD, garantizando disponibilidad, trazabilidad y dinamismo en la gestión de servicios.

**Figura 21**  
**Vista del controlador del proyecto USSD**

```
@RequestMapping("/USSD")
public class UssdPushController {
 private static final Logger logger = LogManager.getLogger(UssdPushController.class);

 private final ApplicationConfig applicationConfig;
 private final ApplicationContext applicationContext;

 private ActorSystem actorSystem;

 @Autowired
 private JWTUtil jwtUtil;

 @Autowired
 public UssdServiceRepository ussdServiceRepository;

 @Autowired
 public UssdServicePushRepository ussdServicePushRepository;

 @Autowired
 public UssdPushRepository ussdPushRepository;

 @Autowired
 public UssdPushController(ApplicationContext applicationContext, ApplicationConfig applicationConfig) {
 this.applicationConfig = applicationConfig;
 this.applicationContext = applicationContext;
 this.actorSystem = applicationContext.getBean(ActorSystem.class);
 }

 // @CrossOrigin
 @PostMapping(path = "/push", consumes = "application/json", produces = "application/json")
 private Mono<Response> pushService(@Valid @RequestBody Push push, ServerHttpRequest request) {
 logger.info("UssdPushService received from " + request.getRemoteAddress().getAddress().getHostAddress() + ":" + request.getRemoteAddress().getPort());
 logger.debug("body : " + push);

 push.getSubscribers().forEach(Logger::info);

 ActorPath actorPath = actorSystem.child(ManagerBroker.NAME);
 actorSystem.actorSelection(actorPath).tell(new ManagerRequest(push), ActorRef.noSender());

 return Mono.just(new Response("accepted"));
 }
}
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

De la imagen se observa que la clase `UssdPushController`, definida en el paquete de controladores, se encarga de gestionar las solicitudes de tipo push dentro de la arquitectura USSD. Mediante la anotación `@RequestMapping("/USSD")`, expone un endpoint específico que procesa interacciones provenientes del backend hacia el abonado, lo que permite notificar al usuario o iniciar sesiones sin que este haya marcado un código en su dispositivo.

El controlador se apoya en diversos componentes inyectados con `@Autowired`:

- `ApplicationConfig` y `ApplicationContext`: proveen configuraciones globales y el contexto de ejecución.
- `ActorSystem`: integra el modelo de actores Akka, asegurando que los mensajes USSD puedan gestionarse de forma concurrente y escalable.
- Repositorios (`ussdServiceRepository`, `ussdServicePushRepository`, `ussdPushRepository`): facilitan el acceso y persistencia de la información asociada a los servicios y transacciones USSD.
- `JwtUtil`: aporta mecanismos de seguridad basados en tokens, garantizando que solo solicitudes autorizadas puedan ser procesadas.

El método principal, `pushService()`, expuesto con la anotación `@PostMapping`, recibe un objeto `Push` en formato JSON y la información de la petición HTTP (`ServerHttpRequest`). Su flujo de ejecución contempla:

- Registro de la solicitud entrante, incluyendo la dirección IP y puerto de origen, mediante el sistema de logging.
- Iteración sobre la lista de suscriptores definidos en el objeto `Push`.

- Envío del mensaje al sistema de actores (ManagerBroker) a través de una petición asíncrona, donde se construye un objeto ManagerRequest y se envía mediante actorSystem.actorSelection(...).
- Devolución de una respuesta reactiva (Mono<Response>), confirmando la aceptación de la operación.

Este fragmento refleja la **integración entre Spring WebFlux y Akka**: mientras el primero gestiona la exposición del servicio y el procesamiento de peticiones HTTP en formato JSON, el segundo se encarga de distribuir las solicitudes hacia los actores correspondientes que ejecutan la lógica de negocio.

Además de gestionar solicitudes *push* con mensajes JSON, la clase **UssdPushController** incorpora dos endpoints adicionales que fortalecen la comunicación y la seguridad del sistema:

#### **/v1/push**

Este método, expuesto con `@GetMapping`, recibe como parámetros el **identificador de servicio (id)** y el **número del abonado (msisdn)**. Su función principal es validar la procedencia de la solicitud:

- Registra en el log la dirección IP y puerto de origen, así como los parámetros recibidos.
- Verifica el encabezado HTTP Origin para confirmar que la petición proviene de un remitente válido.
- Si no se detecta un origen, retorna un estado **400 (BAD\_REQUEST)** con la respuesta "NOACK".
- En caso exitoso, devuelve un estado **200 (OK)** acompañado de "OK".

Esta lógica asegura un primer filtro de validación, reduciendo la posibilidad de solicitudes malformadas o no autorizadas.

Este endpoint genera un **token de autenticación basado en JWT (JSON Web Token)**. Recibe un parámetro id de tipo Long, lo registra en el log y construye un objeto Token que encapsula la cadena generada por el componente jwtUtil.

Este fragmento refleja cómo el controlador combina mecanismos de validación de solicitudes con procesos de autenticación basados en **tokens JWT**. La integración de estos métodos robustece la solución USSD, proporcionando un balance entre accesibilidad y seguridad en los servicios expuestos.

**Figura 22**  
**Clase Flow del proyecto USSD**

```

@NotNullConfidenceJms @NotNullConfidenceJms @SubscribeJms @UnsubscribeJms @SecurityConfigJms @SQSURLTypeJms @ManagedEntityJms @UnlinkConfidenceRepositoryJms
[EditorComponent]
public class UsdFlow extends AbstractActorWithTimers {
 public static final String NAME = "usdflow";
 private static final Logger logger = LogManager.getLogger(UsdFlow.class);

 private static final Object TICK_KEY = "PeriodicReadDB";
 private static final class PeriodicReadDB {}

 @Autowired
 private ApplicationConfig applicationConfig;

 @Autowired
 private UsdBInterfaceRepository usdBInterfaceRepository;

 @Override
 private RouteBD routeBD() {
 // Override
 public void prestart() throws Exception {
 super.prestart();
 logger.info("Actor[" + self().path() + "] started.");

 try {
 readHandler();
 } catch (Exception e) {
 logger.error("PeriodicReadDB " + e.getMessage());
 }

 getTimers().startTimerAtFixedRate(TICK_KEY, new PeriodicReadDB(), Duration.ofMinutes(applicationConfig.getPeriodicReadDBMin()));
 }

 // Override
 public void poststop() throws Exception {
 super.poststop();
 logger.info("Actor[" + self().path() + "] stopped.");
 }

 // Override
 public Receive createReceive() {
 return receiveBuilder()
 .match(GetService.class, msg -> {
 // ...
 })
 .match(GetStartNode.class, msg -> {
 logger.debug("GetStartNode received " + msg);
 })
 .match(GetTextNode.class, msg -> {
 logger.debug("GetTextNode received " + msg);
 })
 .match(PeriodicReadDB.class, msg -> {
 logger.debug("PeriodicReadDB received ");
 try {
 readHandler();
 } catch (Exception e) {
 logger.error("PeriodicReadDB " + e.getMessage());
 }
 })
 };
 }
 }
}

```

*Nota.* Fuente: Código fuente del proyecto desarrollado

De la imagen se observa que la clase `UssdFlow`, ubicada en el paquete de actores del sistema, representa el componente central en la orquestación del flujo USSD.

Anotada con `@ActorComponent`, hereda de `AbstractActorWithTimers`, lo que le permite ejecutar tareas concurrentes y programar operaciones periódicas de forma automática. Esta estructura se ajusta al paradigma de programación reactiva y distribuida, esencial en sistemas de telecomunicaciones que requieren procesar miles de interacciones simultáneamente.

El método `preStart()` se ejecuta al inicio del ciclo de vida del actor, registrando la creación del componente y desencadenando la carga inicial de configuraciones desde la base de datos. Para mantener actualizada esta información, el actor utiliza un temporizador interno (`getTimers()`) que ejecuta periódicamente la operación `readDBandSave()`, encargada de consultar la tabla de interfaces activas (`UssdDbInterfaceRepository`) y actualizar dinámicamente las conexiones en `RouteDb`. Esto asegura que el sistema pueda adaptarse de manera continua a cambios en las configuraciones sin necesidad de reinicios.

El método `createReceive()` define los mensajes que el actor puede procesar, entre los cuales destacan:

- `GetService`: orientado a la recuperación de un servicio específico.
- `GetStartNode`: inicializa el nodo raíz del flujo USSD.
- `GetNextNode`: gestiona la transición hacia el siguiente nodo de la sesión.

- **PeriodicReadDB:** ejecuta la lectura periódica de configuraciones desde la base de datos.

La implementación de este patrón garantiza que el backend pueda responder a múltiples mensajes de manera concurrente, manteniendo la robustez y escalabilidad del sistema.

**Figura 23**  
**Service de presentación de menú USSD**

```

public String getMessageFromPage(UssdPage ussdPage){
 StringJoiner ussdMsg = new StringJoiner("\n");
 //header
 if (ussdPage.getHeader() != null) {
 ussdMsg.add(ussdPage.getHeader());
 }
 logger.debug(ussdMsg.toString());
 //enumeration list
 if (ussdPage.getItems() != null) {
 ussdPage.getItems().isEmpty() {
 for (i = 0; i < ussdPage.getItems().size(); i++) {
 ussdMsg.add((i + 1) + " " + ussdPage.getItems().get(i).getValue().trim());
 }
 }
 logger.debug("Añadiendo navegación -> end/home : " +
 (ussdPage.getPageProp().getNav().compareTo("end")==0 ? " / " : " ") +
 ussdPage.getPageProp().isHome());
 if(!((ussdPage.getPageProp().getNav().compareTo("end")==0 || ussdPage.getPageProp().isHome())) { //es el final
 logger.debug("Volatil: " + ussdPage.getPageProp().isVolatil());
 if (ussdPage.getPageProp().isVolatil()) {
 String backText = "Atras";
 if (ussdPage.getPageProp().getBackText().isEmpty()) {
 backText = ussdPage.getPageProp().getBackText();
 }
 ussdMsg.add("0. " + backText);
 String menuText = "Menu";
 if (ussdPage.getPageProp().getMenuText().isEmpty()) {
 menuText = ussdPage.getPageProp().getMenuText();
 }
 ussdMsg.add("00. " + menuText);
 String inicioText = "Home";
 ussdMsg.add("000. " + inicioText);
 }
 }
 logger.debug(ussdMsg.toString());
 return ussdMsg.toString();
 }
}

//**
// For USSD pull
// @param request
// @param request
// @return
//**
public UssdCommand getUssdCommandFromPage(UssdPage ussdPage, UssdCommand request, boolean isPull) {
 String msg = getMessageFromPage(ussdPage);
 if (ussdPage.getPageType() == PageType.MENU || ussdPage.getPageType() == PageType.FORM ||
 ussdPage.getPageType() == PageType.FORM_PASSWORD) {
 return UssdCommand.newBuilder(request)
 .setUssdString(msg)
 .setUssdServiceOp(UssdCommand.Ussd_service_op.USSD_SERVICE_UNSTRUCTURED_SS_REQUEST)
 .build();
 }
 else if (ussdPage.getPageType() == PageType.END) {
 if (!isPull) {

```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase **PresentationService**, anotada con **@Service**, constituye la puerta de salida de la información procesada hacia el usuario final. Su objetivo principal es transformar los objetos lógicos del sistema (por ejemplo, **UssdPage**) en cadenas de texto que respeten la estructura de menú y submenú característica de los servicios USSD.

Esta capa resulta fundamental, ya que define la experiencia del usuario en términos de claridad, orden y facilidad de navegación.

El método `getMessageFromPage(UssdPage)` organiza dinámicamente el contenido de una página USSD. A través de un `StringJoiner`, construye el mensaje línea por línea: primero incorpora el encabezado, luego enumera los ítems disponibles y finalmente añade opciones de navegación como Atrás (0), Menú (00) y Inicio (000).

**Figura 24**  
**Lógica de navegación entre menús USSD**

```
public String isForm(Document actdoc) throws XPathExpressionException, IOException, SAXException, ParserConfigurationException {
 XPathFactory xpathFactory = XPathFactory.newInstance();
 XPath xpath = xpathFactory.newXPath();
 XPathExpression expr_13 =
 xpath.compile(expression: "//page/text()");
 String title= (String) expr_13.evaluate(actdoc,XPathConstants.STRING);
 XPathExpression expr_14 =
 xpath.compile(expression: "//page/form/entry/prompt/text()");
 String title2= (String) expr_14.evaluate(actdoc,XPathConstants.STRING);

 String getForm=null;
 if (title=="") {
 System.out.println(title2);
 getForm=title2;
 }else {
 if(title2==""){
 System.out.println(title);
 getForm=title;
 }else{
 System.out.println(title + "\n"+title2);
 getForm=title + "\n"+title2;
 }
 }

 return getForm;
}

public boolean isInclude(Document doc) throws XPathExpressionException, IOException, SAXException, ParserConfigurationException {
 boolean isInclude;
 XPathFactory xpathFactory = XPathFactory.newInstance();
 // Create XPath object
 XPath xpath = xpathFactory.newXPath();
 XPathExpression expr_1 =
 xpath.compile(expression: "//pages/page/include/@src");
 String include= (String) expr_1.evaluate(doc,XPathConstants.STRING);

 Node nodeFound = null;
 Node parentNode = null;
 Document actdoc;
 List<String> lista= new ArrayList<>();
 if (include != ""){
 isInclude=true;
 XPathExpression expr_12 =
 xpath.compile(expression: "//pages/page/include[@src='"+include+"']");
 nodeFound = (Node) expr_12.evaluate(doc, XPathConstants.NODE);

 parentNode = nodeFound.getParentNode();

 actdoc=StringtoDoc(nodeToString(parentNode));
 XPathExpression expr_13 =
 xpath.compile(expression: "//page/title/text()");
 String title= (String) expr_13.evaluate(actdoc,XPathConstants.STRING);

 XPathExpression expr_14 =
 xpath.compile(expression: "//page/tag");
 }
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

Además, la clase integra verificaciones condicionales para determinar cuándo mostrar dichas opciones, evitando duplicidad en nodos finales o páginas de tipo home.

Por otro lado, el método `getUssdCommandFromPage(UssdPage, UssdCommand, boolean)` encapsula el mensaje generado en un objeto `UssdCommand`, definiendo también la operación correspondiente según el tipo de página (MENU, FORM, END). Este mecanismo asegura que la lógica de negocio se traduzca en comandos compatibles con la red de telecomunicaciones, ya sea en modo pull (cuando el usuario inicia la interacción) o push (cuando el sistema envía notificaciones).

Dentro de la solución USSD, uno de los aspectos más relevantes es la capacidad de **leer, interpretar y manipular documentos XML** que contienen la definición de los menús interactivos. El fragmento de código presentado concentra funciones orientadas a esta tarea, utilizando **XPath** como herramienta principal para navegar entre nodos y extraer la información necesaria.

El método `isForm(Document actdoc)` ejemplifica cómo se identifica si una página contiene un formulario, evaluando etiquetas como `//page/text()` y `//page/form/entry/prompt/text()`. Según la disponibilidad de estos campos, se construye dinámicamente un título que luego se proyectará al usuario. Este enfoque flexible asegura que los formularios puedan procesarse aun cuando varíen en su estructura interna.

Por su parte, el método `isInclude(Document doc)` gestiona la detección y el procesamiento de etiquetas `<include>`. Esto permite reutilizar fragmentos de XML previamente definidos, logrando así que los flujos sean **modulares y escalables**. Cuando se detecta un `include`, el sistema busca en la base de datos (a través de `ussdTreeRepository`) la porción de flujo correspondiente, la transforma en documento y la incorpora dentro de la jerarquía original. De esta manera, el backend puede **reconstruir dinámicamente** un menú USSD sin necesidad de modificar el archivo base.

Otros métodos complementarios como `getFirstChild(Document doc)`, `findByTag(Document doc, String nameMenu)` y `unwrap(Document doc, String tagName)`

refuerzan esta lógica, permitiendo localizar nodos específicos, convertirlos en documentos independientes y manipularlos como estructuras reutilizables. Asimismo, utilitarios como `nodeToString(Node)` y `convertStringToDocument(String)` garantizan la conversión entre objetos Java y representaciones XML, facilitando la persistencia y transmisión de los flujos.

**Figura 25**  
**Lógica de envío de SMS vía HTTP**



```
1 package com.hacom.ussdcore.sms;
2
3 import ...
4
5
6
7
8
9
10
11
12
13 public class SmsKannel implements SmsService {
14
15 @Override
16 public void sendSms(SmsInit smsInit, String url, HttpClient httpClient, ActorRef actor) {
17
18 StringJoiner param = new StringJoiner("&");
19 param.add(url);
20 param.add("&to=").add(smsInit.getMsisdn());
21 param.add("&text=");
22 param.add(byteArrayToURLString(smsInit.getText().getBytes(StandardCharsets.US_ASCII))); //it could contain 256 char
23 param.add("&validity=").add("30"); //Its the DCS field and setting it to be 8bit binary message
24
25 httpClient
26 .get()
27 .uri(param.toString())
28 .responseContent()
29 .aggregate()
30 .asString()
31 .doOnError(e -> {
32 actor.tell(new SmsError(e, smsInit.getMsisdn()), ActorRef.noSender());
33 })
34 .onErrorReturn("error")
35 .timeout(Duration.ofSeconds(5)) // timeout
36 .doOnError(e -> {
37 actor.tell(new SmsError(e, smsInit.getMsisdn()), ActorRef.noSender());
38 })
39 .doOnSuccess(r -> {
40 actor.tell(new SmsSuccess(r, smsInit.getMsisdn()), ActorRef.noSender());
41 })
42 .subscribe();
43 }
44
45 private String byteArrayToURLString(byte in[]) {
46 byte ch = 0x00;
47 int i = 0;
48 if (in == null || in.length <= 0)
49 return null;
50
51 String[] pseudo = { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
52 "A", "B", "C", "D", "E", "F" };
53 StringBuffer out = new StringBuffer(capacity: in.length * 2);
54
55 while (i < in.length) {
56 // First check to see if we need ASCII or HEX
57 if ((in[i] >= '0' && in[i] <= '9')
58 || (in[i] >= 'a' && in[i] <= 'z')
59 || (in[i] >= 'A' && in[i] <= 'Z') || in[i] == '$'
60 || in[i] == '-' || in[i] == '_' || in[i] == '.'
61 || in[i] == '!') {
62 out.append((char) in[i]);
63 i++;
64 } else {
65 out.append(ch);
66 ch = (byte) (in[i] << 4 | in[i] >> 4);
67 i++;
68 }
69 }
70 return out.toString();
71 }
72 }
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

El método `sendSms(SmsInit smsInit, String url, HttpClient httpClient, ActorRef actor)` implementa la lógica para enviar mensajes de texto a través de un cliente HTTP. Este proceso resulta crítico en un entorno de telecomunicaciones, ya que muchas transacciones USSD requieren confirmar operaciones o enviar información adicional al usuario vía SMS.

El procedimiento inicia construyendo dinámicamente la URL de solicitud mediante la clase `StringJoiner`. En ella se concatenan parámetros esenciales como:

- Número destino (`msisdn`), obtenido del objeto `SmsInit`.
- Texto del mensaje (`text`), que es codificado a formato seguro para URL mediante el método auxiliar `byteArrayToURLString`.
- Validez del mensaje, establecida en 30 unidades de tiempo, la cual corresponde al campo DCS utilizado para mensajes de tipo binario de 8 bits.

Una vez formada la URL, el cliente HTTP (`httpClient`) ejecuta una llamada no bloqueante, utilizando un enfoque reactivo. Este patrón permite procesar la respuesta o los errores sin interrumpir el flujo principal del sistema. En caso de error, se notifica al actor correspondiente mediante el envío de un objeto `SmsError`. Si la operación es exitosa, se envía un objeto `SmsSuccess` junto con el resultado de la operación. De este modo, se garantiza la trazabilidad y control sobre cada mensaje emitido.

### Método auxiliar de codificación

El método privado `byteArrayToURLString(byte[] in)` transforma un arreglo de bytes en una cadena apta para ser incluida en una URL. Para ello, evalúa cada carácter:

- Si es un carácter alfanumérico o permitido (a-z, A-Z, 0-9, \$, -, \_, ., !), se conserva tal cual.
- En caso contrario, se convierte a formato hexadecimal con prefijo %, siguiendo la convención de percent-encoding en URLs.

Este procedimiento garantiza que el contenido del SMS, que puede incluir caracteres especiales, sea interpretado correctamente por el servidor de mensajería sin corromper la transmisión.

**Figura 26**  
**Clase encargada de gestionar consultas a la BD**

```
package com.hacom.ussdcore.jdbc;

import ...
@Service
public class JdbcQueries {
 public String selectQuery(JdbcTemplate jdbcTemplate, String sqlQuery, String[] fieldName, int timeoutMs){
 return Mono.just("query")
 .flatMap(v -> {
 Mono.just(jdbcTemplate.query(sqlQuery,
 (rs, nrow) -> {
 JSONObject jsonObject = new JSONObject();
 for (String s : fieldName) {
 JsonElement jsonElement = new JsonPrimitive(rs.getString(s));
 jsonObject.add(s, jsonElement);
 }
 return jsonObject;
 })
)
 })
 .onErrorReturn(new ArrayList<>())
 .timeout(Duration.ofMillis(timeoutMs))
 .map(l -> {
 JSONArray jsonArray = new JSONArray();
 for (JsonObject jsonObject : l) {
 jsonArray.add(jsonObject);
 }
 return jsonArray.toString();
 })
 .block();
 }

 public Integer updateInsertQuery(JdbcTemplate jdbcTemplate, String sqlQuery, int timeoutMs){
 return Mono.just("query")
 .flatMap(v -> {
 Mono.just(jdbcTemplate.update(sqlQuery))
 })
 .onErrorReturn(0)
 .timeout(Duration.ofMillis(timeoutMs))
 .onErrorReturn(0)
 .block();
 }
}
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

- JdbcQueries: Acceso y manipulación de datos en la base de datos

La clase JdbcQueries, anotada con @Service, se encarga de gestionar las operaciones de consulta y actualización hacia la base de datos PostgreSQL mediante el uso de JdbcTemplate. Este componente de Spring Framework simplifica la ejecución de sentencias SQL, evitando la gestión manual de conexiones y result sets, lo que contribuye a un acceso más seguro y eficiente a los datos.

El método selectQuery(...) permite ejecutar consultas de lectura (tipo SELECT) y retornar los resultados en formato JSON. Para ello, recibe como parámetros el JdbcTemplate, la sentencia SQL a ejecutar, los nombres de los campos que deben

mapearse y un tiempo máximo de espera (timeoutMs). El resultado es procesado de manera reactiva con la librería Project Reactor (Mono), que transforma cada fila en un objeto JsonObject y, finalmente, construye un arreglo (JsonArray) en formato de cadena. Este enfoque asegura compatibilidad con el ecosistema del backend, donde las respuestas en formato JSON son estándar para integraciones internas y externas.

El método updateInsertQuery(...), por su parte, ejecuta operaciones de escritura (INSERT, UPDATE, DELETE) y devuelve un valor entero que indica la cantidad de filas afectadas. También implementa manejo de errores y tiempo de espera, devolviendo el valor cero en caso de fallo. Este diseño robusto garantiza que las transacciones de la aplicación no se vean interrumpidas ante fallos puntuales en la base de datos.

**Figura 27**  
**Modelo para la clase Interface USSD**

```
package com.hacom.ussdcore.models;

import javax.persistence.*;

@Entity
@Table(name = "tt_ussd_db_interface")
public class UssdDbInterface {

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;

 @NotNull
 @Column(name = "v_db_interface_name", unique = true)
 private String name;

 @NotNull
 @Column(name = "v_db_username")
 private String username;

 @NotNull
 @Column(name = "v_db_password")
 private String password;

 @NotNull
 @Column(name = "v_db_url")
 private String url;

 @NotNull
 @Column(name = "v_db_driver")
 private String driverClass;

 @NotNull
 @Column(name = "t_db_active")
 private Boolean active;

 @Column(name = "b_db_creation_time")
 private Long creationTime;
}
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase **UssdDbInterface**, definida como una entidad JPA mediante la anotación `@Entity`, modela la tabla `tt_ussd_db_interface`, encargada de almacenar las credenciales y parámetros de conexión hacia las distintas bases de datos utilizadas por la solución USSD. Su diseño permite que las configuraciones de conexión se gestionen de manera dinámica y persistente, sin necesidad de modificar el código fuente.

Entre sus campos más relevantes se encuentran:

- **name**: identificador único de la interfaz de conexión.
- **username y password**: credenciales de acceso a la base de datos.
- **url**: dirección de conexión JDBC que especifica el servidor y la base de datos destino.
- **driverClass**: clase del controlador JDBC a utilizar, garantizando compatibilidad con distintos motores (PostgreSQL, Oracle, MySQL, etc.).
- **active**: indicador lógico que define si la interfaz está habilitada para su uso.

Asimismo, los campos **creationTime** y **sessionIdCreation** permiten registrar metadatos asociados a la creación de la configuración, reforzando la trazabilidad de los cambios en la plataforma.

Este modelo cobra especial importancia en el contexto de la clase **UssdFlow**, donde las interfaces activas son consultadas periódicamente para mantener actualizada la lista de conexiones válidas. De esta manera, el sistema puede adaptarse a entornos heterogéneos de bases de datos y garantizar una **alta disponibilidad** en el acceso a la información.

**Figura 28**

**Clase Service que contiene el modelo de los servicios contemplados para el proyecto USSD**

```
11 @ToString
12 @NoArgsConstructor
13 @Entity
14 @Table(name = "tt_ussd_service")
15 public class UssdService implements Cloneable {
16
17 @Id
18 @Column(name = "b_ser_serviceid")
19 @GeneratedValue(strategy = GenerationType.IDENTITY)
20 private Long id;
21 @NotNull
22 @Column(name = "v_ser_svc_name", unique = true)
23 private String name;
24 @Column(name = "v_ser_description")
25 private String description;
26 @ManyToOne
27 @JoinColumn(name = "b_ser_treemain", nullable = false)
28 private UssdTree tree;
29 @Column(name = "t_ser_type")
30 private String type;
31 @Column(name = "b_ser_creationtime")
32 private Long creationTime;
33 @Column(name = "b_ser_sessionidcreation")
34 private Long sessionIdCreation;
35 @NotNull
36 @Column(name = "t_ser_active")
37 private Boolean active;
38
39 @Column(name = "b_ser_company")
40 private Long companyId;
41
42 @Column(name = "b_ser_campaign")
43 private Long campaignId;
44
45 @Override
46 public Object clone() throws CloneNotSupportedException {
47 return super.clone();
48 }
49
50 }
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase `UssdService`, definida como entidad JPA y mapeada a la tabla `tt_ussd_service`, representa el **núcleo de la configuración de servicios disponibles en la plataforma USSD**. A través de esta entidad, el sistema administra cuáles servicios están activos, su estructura lógica asociada y las campañas o compañías a las que pertenecen.

Entre sus campos más relevantes destacan:

- **id**: identificador único del servicio, generado automáticamente.
- **name** y **description**: definen el nombre y la descripción del servicio, asegurando su identificación dentro de la plataforma.

- **tree**: relación con la entidad **UssdTree**, que contiene la estructura de menús y nodos que el usuario recorrerá al utilizar el servicio. Esta relación de tipo *ManyToOne* permite reutilizar árboles de navegación entre distintos servicios.
- **type**: clasifica el servicio según su naturaleza (por ejemplo, informativo, transaccional o promocional).
- **active**: indica si el servicio se encuentra disponible para ser ejecutado por los abonados.

Además, los campos **companyId** y **campaignId** permiten asociar cada servicio a una empresa o a una campaña de marketing específica, lo que aporta flexibilidad en la gestión comercial. Finalmente, atributos como **creationTime** y **sessionIdCreation** registran metadatos relacionados con la creación y la gestión de sesiones, reforzando la trazabilidad operativa.

El hecho de que la clase implemente la interfaz **Cloneable** resalta la necesidad de generar **copias exactas de los servicios** en memoria, una práctica común cuando se requiere instanciar múltiples sesiones concurrentes sin alterar el objeto original.

Este fragmento refleja cómo el modelo abstrae un servicio USSD real, vinculándolo con su estructura de menús y estableciendo los parámetros que regulan su disponibilidad.

**Figura 29**  
**Modelo de la Clase Transaction del proyecto USSD**

```
> import ...

@Getter
@Setter
@ToString
@NoArgsConstructor
@Entity
@Table(name = "tt_ussd_transaction")
public class UssdTransaction {

 @Id
 @Column(name = "b_tra_transactionid")
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;

 @Column(name = "b_tra_serviceid")
 private Long serviceId;

 @Column(name = "v_tra_subscriberid")
 private String subscriberId;

 @Column(name = "v_tra_start")
 private String startTime;

 @Column(name = "v_tra_end")
 private String endTime;

 @Column(name = "b_tra_creationtime")
 private Long creationTime;

 @Column(name = "b_tra_sessionidcreation")
 private Long sessionIdCreation;

}
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase `UssdTransaction`, mapeada a la tabla `tt_ussd_transaction`, constituye la entidad encargada de **registrar y almacenar las transacciones ejecutadas dentro del sistema USSD**. A diferencia de los modelos orientados a la configuración (`UssdService`, `UssdDbInterface`), esta clase refleja la **operación en tiempo de ejecución**, es decir, el uso real de los servicios por parte de los abonados.

Entre sus atributos principales se encuentran:

- **id**: identificador único de la transacción.

- **serviceId**: referencia al servicio USSD utilizado, lo que permite vincular la transacción con su definición en la tabla de servicios.
- **subscriberId**: número de abonado que originó la interacción, constituyendo el vínculo entre la plataforma y el usuario final.
- **startTime** y **endTime**: marcan el inicio y fin de la sesión USSD, esenciales para medir la duración de la interacción.
- **creationTime** y **sessionIdCreation**: proporcionan metadatos asociados a la creación y seguimiento de la transacción en el sistema.

Este modelo no solo tiene valor operativo, sino también estratégico: permite generar **métricas de uso, reportes estadísticos y auditorías**, lo que resulta fundamental en contextos donde la trazabilidad y la calidad del servicio son criterios críticos.

Este fragmento refleja cómo el **XML almacenado en base de datos** es transformado en un objeto interpretable en tiempo de ejecución, asegurando que la lógica declarada pueda ejecutarse de manera transparente.

**Figura 30**  
**Modelo de la clase orientada a los WebServices**

```

24 @Table(name = "t_usssd_ws")
25 @TypeDef(name = "jsonb", typeClass = JsonBinaryType.class)
26 public class UsdWs {
27 @Id
28 @Column(name = "b_ws_wsId")
29 @GeneratedValue(strategy = GenerationType.IDENTITY)
30 private Long id;
31 @Column(name = "v_ws_name")
32 private String name;
33 @Column(name = "v_ws_description")
34 private String description;
35 @Type(type = "jsonb")
36 @Column(name = "v_ws_headers", columnDefinition = "jsonb")
37 private List<HeadersParam> headers;
38 @Column(name = "v_ws_url")
39 private String url;
40 @Column(name = "v_ws_method")
41 @Enumerated(EnumType.STRING)
42 private WsMethod method;
43 @Column(name = "t_ws_tls_active")
44 private Boolean tlsActive;
45 @Column(name = "t_ws_type")
46 @Enumerated(EnumType.ORDINAL)
47 private WsType type;
48 @Type(type = "jsonb")
49 @Column(name = "v_ws_json_body", columnDefinition = "jsonb")
50 private String jsonBody;
51 @Type(type = "com.hacoms.usssdcore.hibernate.types.SQLXMLType")
52 @Column(name = "v_ws_xml_body", columnDefinition = "xml")
53 private String xmlBody;
54 @Type(type = "jsonb")
55 @Column(name = "v_ws_error", columnDefinition = "jsonb")
56 private String error;
57 @Type(type = "jsonb")
58 @Column(name = "v_ws_mapping", columnDefinition = "jsonb")
59 private String mapping;
60 @Column(name = "b_ws_creationTime")
61 private Long creationTime;
62 @Column(name = "b_ws_sessionIdCreation")
63 private Long sessionIdCreation;

```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase **UssdWs**, mapeada a la tabla **tt\_ussd\_ws**, constituye el modelo encargado de gestionar la **configuración y consumo de Web Services** desde el backend USSD. Este componente permite que los flujos interactivos definidos en XML puedan extenderse hacia sistemas externos, consumiendo información en tiempo real y devolviendo respuestas al abonado de manera dinámica.

Entre sus atributos más relevantes se encuentran:

- **url y method:** definen el endpoint del servicio externo y el método de invocación (GET, POST, etc.).
- **headers:** almacenados como estructura JSONB, permiten parametrizar encabezados de autenticación, formatos de contenido u otros metadatos necesarios en la comunicación.
- **jsonBody y xmlBody:** representan el cuerpo de la solicitud, pudiendo almacenarse en formato JSON o XML según el tipo de servicio que se consuma.
- **tlsActive:** indica si la comunicación debe realizarse bajo protocolos seguros (TLS/SSL).
- **type:** clasifica el servicio según su naturaleza, facilitando su uso dentro de la plataforma.
- Particular relevancia tienen los campos **error** y **mapping**, ambos almacenados en formato JSONB:
- **error:** define los mensajes o estructuras de error que pueden ser devueltos por el servicio externo. Esto permite al backend reconocer fallas y dar una respuesta controlada al usuario.
- **mapping:** describe la correspondencia entre los datos contenidos en la respuesta del servicio (por ejemplo, en formato XML) y los valores que deben ser extraídos y presentados en el flujo USSD. Gracias a este mapeo, el sistema puede transformar respuestas complejas en información concreta, legible y útil para el abonado.

Este diseño dota al backend de una **gran flexibilidad**, pues los servicios pueden actualizarse, ampliarse o modificarse directamente desde la base de datos, sin necesidad de alterar el código fuente. Además, la compatibilidad con JSONB y XML asegura que la plataforma pueda adaptarse a distintos tipos de servicios utilizados en el ecosistema de telecomunicaciones.

Dentro del proyecto, el paquete **security** constituye un conjunto de componentes diseñados para garantizar la **autenticación, autorización y protección de las comunicaciones**. En un entorno como el de los servicios USSD, donde se procesan transacciones y datos sensibles, contar con una arquitectura de seguridad robusta es esencial.

**Figura 31**  
Clase orientado a la gestión de tokens

```
15 @Component
16 public class JWTUtil {
17
18 private String secret = "m1$RE0e8IvgR@Zkx0C1DWfhUY&5nEwgSnS4K2M0";
19
20 private String expirationTime = "28800";
21
22 private Key key;
23
24 @PostConstruct
25 > public void init() { this.key = Keys.hmacShaKeyFor(secret.getBytes()); }
26
27 public Claims getAllClaimsFromToken(String token) {
28 return Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(token).getBody();
29 }
30
31 > public String getUsernameFromToken(String token) { return getAllClaimsFromToken(token).getSubject(); }
32
33 > public Date getExpirationDateFromToken(String token) { return getAllClaimsFromToken(token).getExpiration(); }
34
35 private Boolean isTokenExpired(String token) {
36 final Date expiration = getExpirationDateFromToken(token);
37 return expiration.before(new Date());
38 }
39
40 public String generateToken() {
41 Map<String, Object> claims = new HashMap<>();
42 claims.put("role", Collections.singletonList("user"));
43 return doGenerateToken(claims);
44 }
45
46 public String generateToken(Long serviceId) {
47 Map<String, Object> claims = new HashMap<>();
48 claims.put("role", Collections.singletonList("user"));
49 claims.put("serviceId", Collections.singletonList(serviceId));
50 return doGenerateToken(claims);
51 }
52
53 private String doGenerateToken(Map<String, Object> claims) {
54 long expirationTimeLong = Long.parseLong(expirationTime); //in second
55 return Jwts.builder().setClaims(claims).setExpiration(new Date(System.currentTimeMillis() + expirationTimeLong)).signWith(key).compact();
56 }
57
58 > isTokenExpired
59
60 > security > JWTUtil > isTokenExpired
```

*Nota.* Fuente: Código fuente del proyecto desarrollado

La clase más destacada es **JWTUtil**, que implementa la lógica de generación y validación de **JSON Web Tokens (JWT)**. Estos tokens son ampliamente utilizados en

sistemas distribuidos para autenticar a los usuarios y servicios sin necesidad de mantener sesiones persistentes en el servidor. El componente inicializa una clave secreta (`secret`) con la cual firma los tokens y define un tiempo de expiración (`expirationTime`) para controlar su vigencia. A través de sus métodos, es posible:

- **Generar tokens** con información adicional (*claims*), como el rol del usuario o el identificador de un servicio.
- **Extraer información del token**, por ejemplo el usuario (`getUsernameFromToken`) o la fecha de expiración (`getExpirationDateFromToken`).
- **Validar tokens** para determinar si siguen siendo válidos o si ya expiraron.

Este diseño asegura que cada petición pueda ser autenticada mediante un token firmado y con tiempo limitado, lo que refuerza la integridad del sistema.

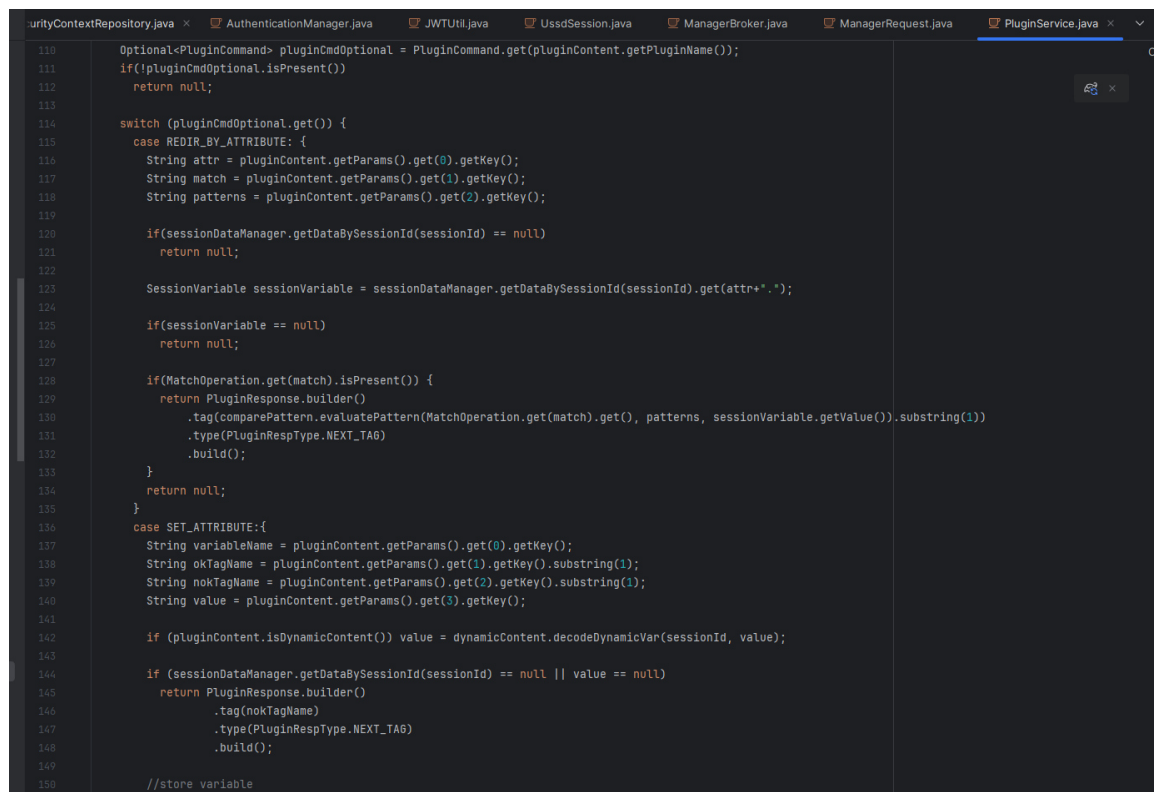
De forma complementaria, el paquete cuenta con otras clases que refuerzan la seguridad:

- **AuthenticationManager**: actúa como el orquestador de la autenticación, delegando en los distintos componentes la validación de credenciales y la emisión de tokens.
- **PBKDF2Encoder**: implementa un algoritmo de hash robusto (PBKDF2) para proteger contraseñas, evitando que se almacenen en texto plano.
- **CORSFilter**: define las reglas de Cross-Origin Resource Sharing, controlando desde qué dominios externos puede consumirse la API.

- SecurityContextRepository: mantiene el estado de seguridad y el contexto asociado a cada petición, integrándose con el resto de la arquitectura Spring Security.

Entonces como se explica, este paquete establece una infraestructura de seguridad integral, donde los tokens JWT permiten autenticar de manera ligera y escalable, mientras que los demás componentes aseguran el cumplimiento de buenas prácticas en cifrado, control de orígenes y persistencia del contexto de seguridad.

**Figura 32**  
Service que contiene la lógica de cada uno de los plugins



```

110 Optional<PluginCommand> pluginCmdOptional = PluginCommand.get(pluginContent.getPluginName());
111 if(!pluginCmdOptional.isPresent())
112 return null;
113
114 switch (pluginCmdOptional.get()) {
115 case REDIR_BY_ATTRIBUTE: {
116 String attr = pluginContent.getParams().get(0).getKey();
117 String match = pluginContent.getParams().get(1).getKey();
118 String patterns = pluginContent.getParams().get(2).getKey();
119
120 if(sessionDataManager.getDataBySessionId(sessionId) == null)
121 return null;
122
123 SessionVariable sessionVariable = sessionDataManager.getDataBySessionId(sessionId).get(attr+".");
124
125 if(sessionVariable == null)
126 return null;
127
128 if(MatchOperation.get(match).isPresent()) {
129 return PluginResponse.builder()
130 .tag(ComparePattern.evaluatePattern(MatchOperation.get(match).get(), patterns, sessionVariable.getValue()).substring(1))
131 .type(PluginRespType.NEXT_TAG)
132 .build();
133 }
134 return null;
135 }
136 case SET_ATTRIBUTE:{
137 String variableName = pluginContent.getParams().get(0).getKey();
138 String okTagName = pluginContent.getParams().get(1).getKey().substring(1);
139 String nokTagName = pluginContent.getParams().get(2).getKey().substring(1);
140 String value = pluginContent.getParams().get(3).getKey();
141
142 if (pluginContent.isDynamicContent()) value = dynamicContent.decodeDynamicVar(sessionId, value);
143
144 if (sessionDataManager.getDataBySessionId(sessionId) == null || value == null)
145 return PluginResponse.builder()
146 .tag(nokTagName)
147 .type(PluginRespType.NEXT_TAG)
148 .build();
149
150 //store variable

```

*Nota.* Fuente: Código fuente del proyecto desarrollado

El paquete de **plugins** constituye un elemento diferencial dentro de la solución, pues permite extender la lógica de los flujos definidos en XML a través de instrucciones especiales que se interpretan en tiempo de ejecución. A través de **URLs con esquemas personalizados** (`erl://` o `dynertl://`), los nodos del árbol USSD pueden invocar funciones adicionales que alteran la navegación o incorporan datos dinámicos, ampliando la capacidad de respuesta del sistema sin necesidad de modificar la lógica central.

La clase **PluginService** concentra la lógica de interpretación y ejecución de estas instrucciones. En primer lugar, **decodePluginUrl** analiza y transforma la cadena de invocación en un objeto **PluginContent**, que encapsula el nombre del plugin, sus parámetros y si corresponde a un contenido dinámico. Posteriormente, el método **processPlugin** evalúa la instrucción recibida y ejecuta distintas operaciones, entre ellas:

- **Redireccionar por atributos (REDIR\_BY\_ATTRIBUTE):** determina el siguiente nodo de navegación según el valor de variables de sesión y patrones de coincidencia definidos.
- **Asignación de atributos (SET\_ATTRIBUTE y variantes):** permite almacenar valores en la sesión activa, ya sea fijos o dinámicamente obtenidos, para condicionar la navegación posterior.
- **Ejecución de lógica dinámica:** mediante integración con clases como `DynamicContent`, `SessionDataManager` o `ComparePattern`, el sistema puede interpretar valores en tiempo real y dirigir al abonado a la página o servicio correspondiente.

Este enfoque convierte al XML en algo más que un simple descriptor estático: los **plugins actúan como puntos de extensión**, otorgando la posibilidad de conectar variables de sesión, consultas a bases de datos (`JdbcQueries`) o servicios externos (`NavApiService`, `XmlGenericService`) con el flujo del usuario. En consecuencia, el abonado percibe un menú altamente dinámico, donde las opciones mostradas y la lógica de navegación se adaptan según las condiciones actuales de la sesión o la información recuperada de otros sistemas.

**Figura 33**  
Modelado de las tablas correspondientes a la solución USSD

```

66
67
68 Table tr_ussd_profile_role (
69 b_prr_profileid bigint (pk, increment)
70 b_prr_roleid bigint
71)
72
73 Table tr_ussd_service_push (
74 b_ssp_id bigint (pk, increment)
75 b_ssp_service varchar (not null)
76 b_ssp_push bigint (not null)
77 b_ssp_creationtime bigint
78 b_ssp_sessionduration varchar
79)
80
81 Table tt_ussd_push (
82 b_pos_id bigint (pk, increment)
83 v_pos_description varchar (not null)
84 v_pos_path varchar (not null)
85 b_pos_creationtime bigint
86 b_pos_sessionduration varchar
87 t_pos_company bigint
88 t_pos_deleted boolean (default: false)
89)
90
91 Table tt_ussd_db_interface (
92 b_db_interfaceid bigint (pk, increment) // autoincrement
93 v_db_interface_name varchar(255) (not null, unique)
94 v_db_username varchar(255) (not null)
95 v_db_password varchar(255) (not null)
96 v_db_url varchar(255) (not null)
97 v_db_driver varchar(255) (not null)
98 v_db_active boolean (not null, default: false)
99 b_db_creationtime bigint
100 b_db_sessionduration bigint
101)
102
103 Table tt_ussd_plugin (
104 b_plugin_id bigint (pk, increment)
105 v_plugin_name varchar(255) (not null, unique)
106 v_plugin_description varchar(255)
107 v_plugin_type varchar(255) (not null) // function(return) or command
108 v_plugin_definition json
109 t_plugin_active boolean (not null, default: false)
110 b_plugin_creationtime bigint
111)
112
113 Table tt_ussd_conf (
114 b_conf_id bigint (pk, increment)
115 v_conf_app_name varchar(255) (not null)
116 v_conf_param_name varchar(255) (not null)
117 v_conf_description varchar(255)
118 v_conf_type varchar(255) (not null) // value or list
119 v_conf_value json
120 b_conf_creationtime bigint
121)

```

*Nota.* Fuente: Código fuente del proyecto desarrollado

De la imagen que contiene los scripts de creación para las tablas de la solución USSD se puede resumir en la tabla siguiente:

**Tabla 10**  
Descripción de las distintas tablas en el proyecto USSD

Nombre de la tabla	Campos clave representativos	Función principal
<b>tm_ussd_company</b>	b_com_companyid, v_com_name, t_com_active	Registra las empresas que operan en la plataforma, asociando servicios y campañas a un identificador único.
<b>tt_ussd_campaign</b>	b_cam_campaignid, v_cam_name, v_cam_start, v_cam_end	Define campañas USSD activas, vinculadas a servicios específicos y con fechas de inicio y fin.
<b>tm_ussd_user</b>	b_use_userid, v_use_username, v_use_password, t_use_active	Gestiona usuarios del sistema con credenciales de acceso y estado de activación.
<b>tm_ussd_profile</b>	b_pro_profileid, b_pro_name, t_pro_active	Establece perfiles de usuario, que se relacionan con roles para controlar privilegios.
<b>tm_ussd_role</b>	b_rol_roleid, v_rol_rolename, v_rol_description	Define roles de acceso que, vinculados a perfiles, determinan permisos específicos.
<b>tr_ussd_profile_role</b>	b_prr_profileid, b_prr_roleid	Tabla intermedia que relaciona perfiles y roles para asignar permisos de forma flexible.

<b>tt_ussd_service</b>	b_ser_serviceid, v_ser_svc_name, b_ser_treemain, t_ser_active	Centraliza la definición de servicios USSD, vinculándolos con un árbol de navegación (tt_ussd_tree).
<b>tt_ussd_tree</b>	b_tre_treeid, v_tre_body (XML)	Contiene la lógica de los menús USSD en formato XML, base de la navegación de los abonados.
<b>tt_ussd_transaction</b>	b_tra_transactionid, b_tra_serviceid, v_tra_subscriberid	Registra las transacciones generadas por los usuarios, con sus tiempos de inicio y fin.
<b>tt_ussd_push</b>	b_pus_id, v_pus_description, v_pus_path, t_pus_deleted	Define procesos de envío <i>push</i> hacia abonados, asociados a compañías y servicios.
<b>tr_ussd_service_push</b>	b_sep_service, b_sep_push	Relaciona servicios con mensajes <i>push</i> , permitiendo auditoría y trazabilidad.
<b>tt_ussd_db_interface</b>	b_db_interfaceid, v_db_interface_name, v_db_url, t_db_active	Configura conexiones a bases de datos externas, con credenciales y estado de activación.
<b>tt_ussd_db_route</b>	b_route_treeid, b_route_interfaceid	Relaciona árboles de navegación con interfaces de base de datos, habilitando consultas dinámicas.
<b>tt_ussd_ws</b>	b_ws_wsid, v_ws_name, v_ws_url, v_ws_method, v_ws_mapping	Gestiona servicios web externos consumidos por el sistema, con mapeo de respuestas JSON/XML.
<b>tt_ussd_ws_route</b>	b_ws_route_treeid, b_ws_route_wsid	Asocia árboles USSD con Web Services específicos.
<b>tt_ussd_plugin</b>	b_plugin_id, v_plugin_name, v_plugin_type, v_plugin_definition	Define plugins para extender la lógica USSD, con funciones dinámicas configuradas vía JSONB.
<b>tt_ussd_conf</b>	b_conf_id, v_conf_app_name, v_conf_param_name, v_conf_value	Centraliza parámetros de configuración de la aplicación en formato JSONB.
<b>tt_ussd_conf_route</b>	b_conf_route_treeid, b_conf_route_confid	Relaciona configuraciones con árboles de navegación para modularidad en la ejecución.
<b>tt_ussd_code_rules</b>	b_cod_id, v_cod_ussd_code_regex, b_cod_serviceid	Define reglas y expresiones regulares para validar códigos USSD e invocar servicios correspondientes.
<b>tt_ussd_variable_name</b>	b_var_nameid, b_var_serviceid, v_var_name, v_var_type	Gestiona variables asociadas a un servicio, que pueden almacenarse como valores o listas.
<b>tt_ussd_var_route</b>	b_var_route_treeid, b_var_route_varid	Relaciona variables con árboles USSD para que influyan en la navegación dinámica.
<b>tt_ussd_attribute_name</b>	b_attr_nameid, v_attr_name	Define atributos adicionales que pueden asociarse a variables de servicio.

<b>tt_ussd_smpp</b>	b_smpp_smppid, v_smpp_name, v_smpp_host, t_smpp_port	Configura interfaces SMPP para el envío/recepción de mensajes SMS asociados a la plataforma.
<b>tt_ussd_smpp_route</b>	b_smpp_route_treid, b_smpp_route_smppid	Relaciona árboles de navegación con interfaces SMPP configuradas.

## Pruebas del sistema, implementación en producción y soporte

**Figura 34**  
**Estimaciones de volúmenes de información**

	log/sesiones USSD AO(KB)	sesiones USSD MO/día	sesiones USSD AO/día	logs/día (MB)	logs/día (GB)	logs/15 días (GB)	logs/mes (GB)	holgura disk (30%) - (GB)	RAM requerida (GB)	holgura RAM (30%) - (GB)
1	0,00625	127000000	3620000	3122,680664	3,049492836	45,74239254	91,48478508	130,6925501	21,16666667	30,23809524
2	0,00625	127000000	3620000	3122,680664	3,049492836	45,74239254	91,48478508	130,6925501	21,16666667	30,23809524
3	0,00625	63500000	905000	1555,81665	1,519352198	22,79028296	45,58056593	65,11509418	10,58333333	15,11904762
4	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
5	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
6	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
7	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
8	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
9	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
10	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
11	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
12	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
13	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
14	0,004125	127000000	3620000	2060,969238	2,012665272	30,18997908	60,37995815	86,25708308	10,58333333	15,11904762
15	0,004125	63500000	1206666,667	1028,054199	1,003959179	15,05938768	30,11877537	43,02682195	5,291666667	7,55952381

### Pruebas de rendimiento y dimensionamiento de recursos

*Nota.* Fuente: Documentación obtenida luego de realizar las pruebas correspondientes.

Con el objetivo de garantizar la estabilidad operativa y la capacidad de respuesta del sistema USSD en escenarios de alta concurrencia, se llevaron a cabo pruebas orientadas al análisis de logs generados y al consumo de recursos de infraestructura. Estas pruebas permitieron proyectar de manera cuantitativa la demanda que el sistema puede generar durante su funcionamiento en producción, contemplando tanto las sesiones móviles originadas (MO) como las de terminación (AO).

La evaluación realizada se centró en estimar el volumen de información generado diariamente a partir de los registros de sesiones, expresado en megabytes y gigabytes, así como en determinar su impacto acumulado en periodos de quince días y un mes. A partir de estas proyecciones se estableció también la holgura de almacenamiento necesaria, considerando un margen de seguridad del 30% para prevenir riesgos asociados a la saturación de disco.

De forma complementaria, se calcularon los requerimientos de memoria RAM necesarios para el procesamiento de dichas cargas, incluyendo igualmente un margen del 30% que asegura un rendimiento sostenido y una mayor resiliencia frente a picos de uso inesperados.

La siguiente tabla resume los resultados obtenidos, evidenciando cómo las pruebas permitieron anticipar el crecimiento del consumo de recursos y, en consecuencia, definir una infraestructura que garantice la continuidad del servicio y la adecuada gestión del volumen de transacciones procesadas:

#### Proyecciones de demanda y pruebas de capacidad

**Figura 35**  
**Pruebas de Carga**

INPUTS				
	Dic-20	Dic-21	Dic-22	Dic-23
Usuarios activos	1.500.000	1.900.000	2.200.000	2.500.000
Total Mobile Originated (MO) USSD requests required	93.000.000	117.800.000	136.400.000	155.000.000
Total Application Originated (AO) USSD requests required	2.000.000	2.533.333	2.933.333	3.333.333
Total Mobile Originated (MO) USSD requests that the platform must physically support	100.000.000	127.000.000	148.000.000	169.000.000
Total Application Originated (AO) USSD requests that the platform must physically support	2.860.000	3.620.000	4.200.000	4.770.000
	Dic-20	Dic-21	Dic-22	Dic-23
MAX TPS front end	7.500	9.500	11.000	12.500
MIN TPS BH	6.000	7.600	8.800	10.000
Duración promedio de sesiones	80s			
Promedio de número de niveles de menú	8	10	12	14

*Nota.* Fuente: Documentación obtenida luego de realizar las pruebas correspondientes.

Además de la estimación de recursos físicos, se realizaron pruebas de carga que permitieron determinar la capacidad de la plataforma para soportar un número creciente de usuarios y transacciones en distintos escenarios anuales. Estas pruebas contemplaron tanto las solicitudes generadas desde el dispositivo móvil (Mobile Originated – MO) como aquellas iniciadas desde la aplicación (Application Originated – AO), así como los valores máximos y mínimos de transacciones por segundo (TPS) en condiciones de operación normal y en horas de mayor tráfico.

Los resultados obtenidos muestran un crecimiento sostenido en el número de usuarios y sesiones, lo cual se traduce en una mayor exigencia sobre la infraestructura. La duración promedio de las sesiones (80 segundos) y el aumento progresivo en los

niveles de menú (de 8 a 14) refuerzan la necesidad de contar con una plataforma flexible y escalable, capaz de mantener la continuidad del servicio frente a volúmenes cada vez más altos de demanda.

### 3.10 EVALUACIÓN

#### 3.10.1 EVALUACIÓN ECONÓMICA / EVALUACIÓN COSTO-BENEFICIO

La evaluación económica de la solución implementada por la empresa se realizó considerando tanto la inversión inicial como los beneficios anuales obtenidos en comparación con la propuesta del proveedor anterior. La solución presentada por la empresa en cuestión tuvo un costo de 200,000 USD, lo que representó una reducción del 40% respecto a la alternativa evaluada, cuyo costo ascendía a 280,000 USD.

De igual manera, los costos de soporte anual fueron significativamente menores, alcanzando los 10,000 USD anuales, frente a los 20,000 USD requeridos por la solución competidora. Adicionalmente, la propuesta de la empresa destacó por su mayor flexibilidad técnica, permitiendo implementar plugins y servicios adicionales en plazos más cortos, lo que facilitó la puesta en operación de nuevos servicios y la generación de beneficios económicos adicionales.

**Tabla 11 Análisis comparativo de costos y ventajas competitivas de la solución propuesta frente al proveedor anterior**

Concepto	La Empresa Competencia (USD)		Diferencia (%)
	(USD)	(USD)	
<b>Inversión inicial</b>	200,000	280,000	+40%
<b>Soporte anual</b>	10,000	20,000	+100%
<b>Flexibilidad técnica</b>	Alta	Limitada	—
<b>Tiempo de adaptación</b>	Rápido	Lento	—

*Nota.* Fuente: Elaboración propia

Con base en estas consideraciones, se estimó un beneficio neto de 60,000 USD anuales durante 5 años, que incluyen ahorros por soporte y beneficios derivados de la flexibilidad técnica. Para evaluar la rentabilidad de esta inversión se utilizaron los indicadores financieros: Valor Actual Neto (VAN) y Tasa Interna de Retorno (TIR).

### 3.10.2 Flujo de Caja Proyectado

Tabla 12 Tabla de flujo de caja proyectado

Año	Flujo Neto (USD)
0	-200,000
1	60,000
2	60,000
3	60,000
4	60,000
5	60,000

*Nota.* Fuente: Elaboración propia

#### Cálculo del VAN y TIR

- Tasa de descuento (COK): 10%
- Horizonte de evaluación: 5 años
- VAN:  $\approx 27,447$  USD
- TIR:  $\approx 15.2\%$

### 3.10.3 Interpretación del VAN y TIR

El análisis financiero evidencia que la implementación de la solución propuesta por la empresa resulta favorable:

El VAN positivo (27,447 USD) indica que, una vez descontados los flujos de efectivo futuros, el proyecto no solo recupera la inversión inicial, sino que además genera valor adicional para la organización.

La TIR (15.2%) se ubica por encima del COK asumido (10%), lo que confirma la viabilidad económica de la solución y demuestra que cada dólar invertido genera una rentabilidad superior al costo de oportunidad del capital.

De esta forma podemos ver que, la evaluación económico-financiera demuestra que la solución de la empresa no solo fue más competitiva en términos de costo inicial y soporte, sino que además asegura una rentabilidad sostenible a lo largo del tiempo, validada a través de los indicadores financieros empleados.

## **CAPITULO IV - REFLEXIÓN CRITICA DE LA EXPERIENCIA**

Como autor del presente trabajo, en mi rol de desarrollador backend y diseñador de la base de datos, considero fundamental destacar los aprendizajes, retos y experiencias vividas durante el desarrollo del proyecto.

La elaboración del modelo de datos y la construcción de la lógica backend representaron un desafío significativo, especialmente por la necesidad de adaptar los procesos a los requerimientos específicos de la empresa de telefonía. Para lograrlo, estuve en constante coordinación con el área usuaria, recabando información clave que me permitió comprender el funcionamiento del negocio y traducirlo en soluciones técnicas concretas.

A lo largo del desarrollo, enfrenté situaciones donde el factor tiempo fue determinante. En varios entregables los plazos resultaron bastante ajustados, lo que me exigió optimizar esfuerzos y priorizar tareas críticas para cumplir con los compromisos asumidos sin comprometer la calidad del sistema.

Uno de los aspectos más valiosos de esta experiencia fue el trabajo en equipo. La comunicación fluida y la colaboración con mis compañeros de desarrollo fueron determinantes para resolver problemas técnicos de manera más eficiente, compartir buenas prácticas y asegurar que cada módulo del sistema se integrara de forma adecuada.

En el ámbito técnico, el proyecto me permitió profundizar mis conocimientos en el diseño de bases de datos relacionales, en la programación de desarrollo de servicios backend en Java y con una arquitectura modulada orientadas a eventos y flujos USSD.

Además, las constantes pruebas realizadas me brindaron la oportunidad de fortalecer mis competencias en la validación de sistemas en entornos reales de telecomunicaciones.

Finalmente, este proyecto no solo significó un crecimiento técnico, sino también profesional y personal. Pude comprobar la importancia de la planificación, la disciplina frente a los plazos exigentes y, sobre todo, el valor del trabajo colaborativo como base para alcanzar los objetivos comunes en un contexto de alta exigencia tecnológica.

## **CAPITULO V – CONCLUSIONES Y RECOMENDACIONES**

### **Conclusiones**

- La solución desarrollada permitió diseñar una arquitectura backend sólida y escalable, capaz de integrarse de manera eficiente con los sistemas preexistentes de Telefónica Movistar Argentina. Este aspecto garantizó la interoperabilidad con entornos ya desplegados y aseguró la continuidad operativa sin generar fricciones técnicas.
- La implementación del motor de interpretación de flujos XML demostró un alto nivel de efectividad en la gestión de sesiones USSD en tiempo real, manteniendo el control del estado de cada interacción y respondiendo con la estabilidad y rapidez requeridas por el negocio de telecomunicaciones.
- Desde una perspectiva económica, el proyecto cumplió con el objetivo de reducir los costos operativos de la empresa de telecomunicaciones, representando una alternativa más competitiva frente al proveedor anterior. Los indicadores financieros calculados (VAN positivo y TIR superior al costo de oportunidad) confirmaron que la solución no solo recupera la inversión, sino que también genera un beneficio económico sostenible a lo largo del tiempo.
- La experiencia obtenida en el desarrollo y puesta en marcha de esta solución evidenció la relevancia del trabajo en equipo y de la coordinación con el área usuaria, factores que resultaron determinantes para traducir los requerimientos de negocio en una herramienta técnica ajustada a las necesidades reales de la organización.

## Recomendaciones

- Considerando el éxito de la solución en Telefónica Movistar Argentina, se recomienda replicar este modelo de arquitectura y gestión de flujos USSD en los demás países donde la empresa de telecomunicaciones desarrolla operaciones, garantizando así la estandarización de procesos y la optimización de recursos tecnológicos a nivel regional.
- La empresa debería aprovechar la escalabilidad de la solución para centralizar y homogeneizar la gestión de flujos USSD en distintos mercados, lo que permitirá mantener consistencia en la experiencia del cliente final y facilitar el control de indicadores clave.
- Se sugiere que la estrategia de expansión internacional incluya no solo la implementación técnica, sino también la capacitación de los equipos locales en cada país, asegurando que los beneficios obtenidos en Argentina puedan reproducirse con el mismo nivel de eficiencia y confiabilidad.
- Finalmente, es recomendable que la empresa mantenga una visión de largo plazo en torno a esta solución, utilizándola como base para futuros proyectos de innovación tecnológica, fortaleciendo con ello su competitividad en el sector de telecomunicaciones a nivel global.

## BIBLIOGRAFÍA

- ArkEsel. (2023). Exploring USSD Technology: Opportunities for Software Developers. Recuperado de <https://arkesel.com/exploring-ussd-technology-opportunities-for-software-developers-2/>
- TeleOSS. (2023). Why Aggregators in Africa and Asia Still Rely on USSD Gateway for Growth. Recuperado de <https://teleoss.co/why-aggregators-in-africa-and-asia-still-rely-on-ussd-gateway-for-growth>
- Competition Authority of Kenya (CAK). (2016). Directrices de tarifas y transparencia para servicios USSD. Recuperado de artículos relacionados en medios sobre la reducción de precios para USSD en Kenia.
- Prime Business Africa. (2025, 4 de junio). Telcos Threaten To Stop USSD Services Over ‘Misinformation’ On Billing By Banks. Recuperado de <https://www.primebusiness.africa/telcos-threaten-to-stop-ussd-services-over-misinformation-on-billing-by-banks>
- Observers BD. (2024). USSD price hike by MNOs to hit millions of poor!. Recuperado de <https://www.observerbd.com/news/69623>
- PwC. (2023). Global Telecoms Outlook Perspectives: State of Competition in Telecommunications. Recuperado de informes públicos de PwC
- Datahub. (2023). Perfil empresarial de Hacom S.A.C. Recuperado de <https://www.datahub.io>

- HACOM. (2023). Servicios y soluciones en telecomunicaciones. Recuperado de <http://www.hacom.com.pe>
- LinkedIn. (2024). Hacom S.A.C. – Página oficial. Recuperado de <https://www.linkedin.com>
- Veritrade. (2022). Informe de importaciones y exportaciones de Hacom S.A.C. Recuperado de <https://www.veritradecorp.com>
- ETSI. (2020). European Telecommunications Standards Institute Reports. ETSI.
- GSMA. (2020). The Mobile Economy Latin America. GSM Association.
- GSMA. (2021). State of the Industry Report on Mobile Money. GSM Association.
- OSIPTEL. (2022). Informe sobre servicios de telecomunicaciones y autogestión en Perú. Lima: OSIPTEL.
- Reserve Bank of India. (2021). Report on Trend and Progress of Banking in India. RBI.
- World Bank. (2020). Digital Solutions for Inclusion in Sub-Saharan Africa. Washington: World Bank.

- Kendall, K. E., & Kendall, J. E. (2021). *Systems Analysis and Design* (10<sup>a</sup> ed.). Pearson.
- 3GPP. (2021). Technical Specification Group Core Network and Terminals; Unstructured Supplementary Service Data (USSD); Stage 2. 3rd Generation Partnership Project.
- Dahlman, E., Parkvall, S., & Sköld, J. (2020). *4G, LTE-Advanced Pro and The Road to 5G* (3rd ed.). Academic Press.
- Ericsson. (2022). USSD: A reliable channel for interactive mobile services. Ericsson Technology Review.
- GSMA. (2021). Mobile network infrastructure and legacy service support. GSM Association.
- Stallings, W. (2020). *Wireless Communications & Networks* (2nd ed.). Pearson Education.
- Alotaibi, M., & Alotaibi, F. (2022). Mobile application adoption and connectivity limitations: A review. *Journal of Telecommunications and Digital Economy*.