



TECNOLÓGICO DE ESTUDIOS SUPERIORES DE COACALCO

Sistema web mediante Visual Builder y
Oracle Cloud para la gestión de
incidencias de la empresa 4CE

TESINA

Para obtener el título de:
INGENIERÍA EN SISTEMAS COMPUTACIONALES

Especialidad
(En Auditoría en seguridad informática)

Presenta:
Beatriz Esperanza García Fragoso
201820231
Asesor: José Roberto Pérez Torres

Coacalco de Berriozábal, Febrero/ 2024

DEDICATORIA

Un proyecto que al inicio parece ser un documento más, pasa a ser un logro que día a día se construye para hacerlo realidad.

El trabajo y esfuerzo diario por culminar un paso a la vida profesional, nos deja satisfacción por conocer las aptitudes con las que contamos.

Este logro es dedicado primeramente a dos personas con más consideración en mi vida, quienes han estado durante todo el proceso de mi carrera profesional, le agradezco infinitamente a mis padres, por guiarme con amor, disciplina y constancia.

En segundo lugar, a todas las personas involucradas que con su apoyo lograron que no desistiera durante el camino, gracias infinitas a mi familia y amigos.

En tercer lugar, le agradezco a la institución por brindarme las herramientas para obtener el conocimiento, ser mi segunda casa durante la carrera y formarme como una profesional.

Le agradezco a la empresa 4CE, por la oportunidad de abrirme un espacio en su equipo, y confiar en que no solo mi formación será siempre lo más importante en el ámbito laboral, si no, que las aptitudes y calidad de persona también cuentan en cualquier ambiente.

ÍNDICE

DEDICATORIA.....	2
ÍNDICE	3
ÍNDICE DE FIGURAS	5
CAPITULO I.....	8
GENERALIDADES DEL PROYECTO.....	8
1.1 INTRODUCCIÓN	9
1.2 JUSTIFICACIÓN.....	10
1.3 PLANTEAMIENTO DEL PROBLEMA.....	11
1.4 ANTECEDENTES.....	12
1.5 OBJETIVOS.....	13
1.5.1 OBJETIVO GENERAL	13
1.5.2 OBJETIVOS ESPECÍFICOS	13
1.6 ALCANCES Y LIMITACIONES.....	14
CAPÍTULO II.....	15
MARCO TEÓRICO.....	15
2.1 Introducción	16
MARCO CONCEPTUAL	17
2.2 Introducción a la Web	17
2.2.1 El primer navegador	17
2.2.2 Separación de funciones.....	18
2.2.3 Modelos de distribución en aplicaciones cliente / servidor	19
2.2.4 Arquitecturas de dos y tres niveles.....	19
2.2.5 Descripción de un sistema cliente/servidor	19
2.2.6 El servidor	21
2.3 Bases de datos relacionales	21
2.3.1 Descripción de base de datos.....	22
2.3.2 Lenguajes de programación.....	22
2.3.3 Lenguaje de definición de datos (DDL, SQL).....	23
2.3.4 Consultas Simples	25
2.3.5 Otras sentencias.....	25
2.3.6 PL/SQL.....	26
MARCO METODOLÓGICO.....	32
2.3.7 Términos y Definiciones	32
2.3.8 Componentes de las metodologías	34
2.3.9 Cierre del marco.....	35

CAPÍTULO III.....	37
DESARROLLO	37
3.1 Propuesta de solución.....	38
3.1.1 Componentes del modelo de datos	39
3.1.1.1 Prefijo de objetos de base de datos.....	39
3.1.1.2 Tablas nuevas	39
3.1.1.3 Columnas adicionales.....	39
3.1.1.4 SQL Scripts	40
3.1.1.5 Vistas nuevas	40
3.1.1.6 Constraints.....	41
3.1.1.7 Índices nuevos	42
3.1.1.8 Secuencias nuevas	42
3.1.1.9 Código sql	43
3.1.1.10 Objetos de la librería	44
3.1.1.11 Responsabilidades (descripción de solo consulta)	44
3.1.1.12 Trigger	46
3.1.1.13 Paquetes.....	46
3.1.1.14 Documentación de procedimientos y funciones en los paquetes	47
3.1.1.15 Código PL/SQL	48
3.1.1.16 Nombre de variables	48
3.1.1.17 Condición especial de variables	49
CAPÍTULO IV	51
RESULTADOS.....	51
4.1 Análisis de resultados	52
4.1.1 Creación de servicios en APEX	73
4.1.1.1 Servicios en Visual Builder	91
4.1.1.2 Variables	98
CONCLUSIONES	112
COMPETENCIAS DESARROLLADAS.....	114
FUENTES DE INFORMACIÓN	119
ANEXOS.....	121
1.1 CARTA DE NO INCOVENIENCIA	122

ÍNDICE DE FIGURAS

Figura 1. Elementos de la metodología (Flores, 2017).....	34
Figura 2. (Modelo de desarrollo incremental iterativo de software, 2023).....	35
Figura 3. Formato de documentación para una función [Elaborado por 4CE].....	47
Figura 4. Formato de documentación para un procedimiento [Elaborada por 4CE].....	48
Figura 5. Diagrama Entidad/Relación [Autoría propia].....	52
Figura 6. Perfil creado en la herramienta APEX.....	53
Figura 7. Tabla XX4CE_HD_ROLES_ALL [Autoría propia].....	53
Figura 8. Tabla XX4CE_HD_USERS_ALL [Autoría propia].....	54
Figura 9. Tabla XX4CE_HD_GROUPS_ASSIGNATIONS_ALL [Autoría propia].....	54
Figura 10. Tabla XX4CE_HD_GROUPS_ALL [Autoría propia].....	55
Figura 11. Tabla XX4CE_HD_BRANDS_ALL [Autoría propia].....	55
Figura 12. Tabla XX4CE_HD_PROJECTS_ALL [Autoría propia].....	56
Figura 13. Tabla XX4CE_HD_TICKETS_ALL [Autoría propia].....	56
Figura 14. Tabla XX4CE_HD_TYPES_ALL [Autoría propia].....	57
Figura 15. Tabla XX4CE_HD_BRANCHES_ALL [Autoría propia].....	57
Figura 16. Tabla XX4CE_HD_ASSIGNED_TICKETS_ALL [Autoría propia].....	58
Figura 17. Tabla XX4CE_HD_RELEASES_ALL [Autoría propia].....	58
Figura 18. Tabla XX4CE_HD_TICKET_COMMENTS_ALL [Autoría propia].....	59
Figura 19. Tabla XX4CE_HD_TICKET_PRIORITY_ALL [Autoría propia].....	59
Figura 20. Tabla XX4CE_HD_STATUS_ALL [Autoría propia].....	60
Figura 21. PackagePKG_XX4CE_HD_ROLES_ALL: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	60
Figura 22. PackageXX4CE_HD_USERS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	61
Figura 23. PackageXX4CE_HD_GROUPS_ASSIGNATIONS_ALL_PKG: a) creación de la estructura del paquete, b) creación de la especificación del paquete [Autoría propia].....	61
Figura 24. PackageXX4CE_HD_GROUPS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	62
Figura 25. PackageXX4CE_HD_BRANDS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	62
Figura 26. Package XX4CE_HD_PROJECTS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	63
Figura 27. PackageXX4CE_HD_TICKETS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	63
Figura 28. PackageXX4CE_HD_TYPES_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	64
Figura 29. PackageXX4CE_HD_ASSIGNED_TICKETS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	64
Figura 30. PackageXX4CE_HD_RELEASES_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	65
Figura 31. PackageXX4CE_HD_TICKET_COMMENTS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	65
Figura 32. PackageXX4CE_HD_USERS_LOGIN_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia].....	66
Figura 33. PROCEDURE_STORE_ROLES [Autoría propia].....	66
Figura 34. DECRYPT_PASS [Autoría propia].....	67
Figura 35. ENCRYPT_PASS [Autoría propia].....	67
Figura 36. PASSWD_ENCRYPT [Autoría propia].....	68
Figura 37. TECHNICIANS_COUNT [Autoría propia].....	68
Figura 38. XX4CE_HD_FN_DELETE_BRANCHES [Autoría propia].....	69
Figura 39. XX4CE_HD_FN_STORE_BRANCHES [Autoría propia].....	69
Figura 40. XX4CE_HD_FN_UPDATE_BRANCHES [Autoría propia].....	70
Figura 41. XX4CE_HD_ROLES_ALL_S [Autoría propia].....	70

Figura 42. XX4CE_HD_USERS_ALL_S [Autoría propia]	71
Figura 43. XX4CE_HD_GROUPS_ALL_S [Autoría propia]	71
Figura 44. XX4CE_HD_BRANDS_ALL_S [Autoría propia]	71
Figura 45. XX4CE_HD_PROJECTS_ALL_S [Autoría propia]	71
Figura 46. XX4CE_HD_TICKETS_ALL_S [Autoría propia]	71
Figura 47. XX4CE_HD_BRANCHES_ALL_S [Autoría propia]	72
Figura 48. XX4CE_HD_RELASES_ALL_S [Autoría propia]	72
Figura 49. XX4CE_HD_TICKET_COMMENTS_ALL_S [Autoría propia]	72
Figura 50. XX4CE_HD_TICKET_PRIORITY_ALL_S [Autoría propia]	72
Figura 51. Método DELET. Modulo_Crud_Roles [Autoría propia]	73
Figura 52. Método GET. Modulo_Crud_Roles [Autoría propia]	73
Figura 53. Método POST. Modulo_Crud_Roles [Autoría propia]	74
Figura 54. Método PUT. Modulo_Crud_Roles [Autoría propia]	74
Figura 55. Método DELETE. Modulo_Crud_Users [Autoría propia]	75
Figura 56. Método GET. Modulo_Crud_Users [Autoría propia]	75
Figura 57. Método POST. Modulo_Crud_Users [Autoría propia]	76
Figura 58. Método PUT. Modulo_Crud_Users [Autoría propia]	76
Figura 59. Método POST. Modulo_Crud_Groups_Assignations [Autoría propia]	77
Figura 60. Método DELETE. Modulo_Crud_Groups_Assignations [Autoría propia]	77
Figura 61. Método GET. Modulo_Crud_Groups_Assignations [Autoría propia]	78
Figura 62. Método INSERT. Modulo_Crud_Groups_Assignations [Autoría propia]	78
Figura 63. Método UPDATE. Modulo_Crud_Groups_Assignations [Autoría propia]	79
Figura 64. Método UPDATE. Modulo_Crud_Groups [Autoría propia]	79
Figura 65. Método INSERT. Modulo_Crud_Groups [Autoría propia]	80
Figura 66. Método DELETE. Modulo_Crud_Groups [Autoría propia]	80
Figura 67. Método DELETE. Modulo_Crud_Brands [Autoría propia]	81
Figura 68. Método INSERT. Modulo_Crud_Brands [Autoría propia]	81
Figura 69. Método UPDATE. Modulo_Crud_Brands [Autoría propia]	82
Figura 70. Método DELETE. Modulo_Crud_Tickets [Autoría propia]	82
Figura 71. Método POST. Modulo_Crud_Tickets [Autoría propia]	83
Figura 72. Método UPDATE. Modulo_Crud_Tickets [Autoría propia]	83
Figura 73. Método GET. Modulo_Crud_Types [Autoría propia]	84
Figura 74. Método POST. Modulo_Crud_Types [Autoría propia]	84
Figura 75. Método PUT. Modulo_Crud_Types [Autoría propia]	85
Figura 76. Método DELETE. Modulo_Crud_Branches [Autoría propia]	85
Figura 77. Método POST. Modulo_Crud_Branches [Autoría propia]	86
Figura 78. Método PUT. Modulo_Crud_Branches [Autoría propia]	86
Figura 79. Método POST. Modulo_Crud_Assigned_Tickets [Autoría propia]	87
Figura 80. Método PUT. Modulo_Crud_Assigned_Tickets [Autoría propia]	87
Figura 81. Método GET. Modulo_Crud_Assigned_Tickets [Autoría propia]	88
Figura 82. Método POST. Modulo_Crud_Relases [Autoría propia]	88
Figura 83. Método PUT. Modulo_Crud_Relases [Autoría propia]	89
Figura 84. Método GET Modulo_Crud_Relases [Autoría propia]	89
Figura 85. Método POST. Modulo_Crud_Ticket_Comments [Autoría propia]	90
Figura 86. Método PUT. Modulo_Crud_Ticket_Comments [Autoría propia]	90
Figura 88. Servicio Insertar [Autoría propia]	91
Figura 89. Servicio Login [Autoría propia]	91
Figura 90. Servicio Registro [Autoría propia]	91
Figura 91. Servicio Users [Autoría propia]	92
Figura 92. Servicio Groups [Autoría propia]	92
Figura 93. Servicio Groups-Assigantions [Autoría propia]	93
Figura 94. Servicio Select_Brands [Autoría propia]	93
Figura 95. Servicio ServConnProjects [Autoría propia]	94
Figura 96. Servicio AssignnetTickets [Autoría propia]	94

Figura 97. Servicio Branches [Autoría propia].....	95
Figura 98. Servicio Brands [Autoría propia]	95
Figura 99. Servicio Comments [Autoría propia]	95
Figura 100. Servicio GetStatus [Autoría propia]	96
Figura 101. Servicio Projects [Autoría propia]	96
Figura 102. Servicio Relases [Autoría propia]	96
Figura 103. Servicio TicketPriority [Autoría propia].....	97
Figura 104. Servicio Tickets [Autoría propia]	97
Figura 105. Servicio Types [Autoría propia]	97
Figura 106. Variables Main-Login [Autpría propia]	98
Figura 107. Pantalla 1 [Autoría propia]	98
Figura 108. Variables Main-Roles [Autoría propia]	99
Figura 109. Pantalla 2 [Autoría propia]	99
Figura 110. Variables Main-Users [Autoría propia].....	100
Figura 111. Pantalla 3 [Autoría propia]	100
Figura 112. Variables Groups_Assignations [Autoría propia].....	101
Figura 113. Pantalla 4 [Autoría propia]	101
Figura 114. Variables AdminDash [Autoría propia].....	102
Figura 115. Pantalla 5 [Autoría propia]	102
Figura 116. Variables Customer-dash [Autoría propia]	103
Figura 117. Pantalla 6 [Autoría propia]	103
Figura 118. Variables Main-Login [Autoría propia]	104
Figura 119. Pantalla 7 [Autoría propia]	104
Figura 120. Variables Main-test [Autoría propia].....	105
Figura 121. Pantalla 8 [Autoría propia]	105
Figura 122. Variables Main-test2 [Autoría propia].....	106
Figura 123. Pantalla 9 [Autoría propia]	106
Figura 124. Variables Support-dash [Autoría propia].....	107
Figura 125. Pantalla 10 [Autoría propia].....	107
Figura 126. Variables ticket-detail-supp [Autoría propia]	108
Figura 127. Pantalla 11 [Autoría propia].....	108
Figura 128. Variables tickets_relases [Autoría propia].....	109
Figura 129. Pantalla 12 [Autoría propia].....	109
Figura 130. Variables Main-Start [Autoría propia].....	110
Figura 131. Pantalla 13 [Autoría propia].....	110
Figura 132. Variables Main-types [Autoría propia]	111
Figura 133. Pantalla 14 [Autoría propia].....	111
Figura 134. Competencia por desarrollar programación orientada a objetos [TESCo].....	115
Figura 135. Competencia por desarrollar bases de datos [TESCo]	117
Figura 136. Matriz foda [Autoría propia].....	121

CAPITULO I

GENERALIDADES DEL PROYECTO

1.1 INTRODUCCIÓN

En la presente documentación, se podrá visualizar el desarrollo de un sistema de asignación de tickets, utilizando un servicio en la nube que te permite crear y alojar rápidamente aplicaciones web y móviles con un trabajo de programación mínimo (Visual Builder) y Oracle Application Express que es una herramienta de Desarrollo Rápido de Aplicaciones que permite diseñar, desarrollar e implantar aplicaciones responsivas sobre la base de datos usando sólo un navegador web, utilizando entre diversos lenguajes, el lenguaje PLS/SQL. Todo esto con el objetivo de mejorar los requerimientos del usuario, de manera que permita a la empresa, la gestión y agilización de los procesos de resolución del problema técnico.

Este documento contiene un apartado que consta de 4 capítulos, en los cuales se describe por sección lo desarrollado, de manera particular en cada uno de ellos, para así identificar de manera más rápida el contenido de este mismo.

Capítulo I. “Generalidades del proyecto”. En este capítulo se encuentran la información sobre el inicio del desarrollo del proyecto, englobando, Introducción, descripción de la empresa u organización, descripción del puesto o área de trabajo, justificación, problemas a resolver y Objetivos (General y Específicos)

Capítulo II. “Marco teórico”. Dentro de este capítulo, se encuentra el marco conceptual y metodológico, que tienen como objetivo el análisis de los conocimientos existentes sobre el proyecto a desarrollar, de esta manera se consigue orientar la investigación.

Capítulo III. “Desarrollo”. En este apartado, se llevan a cabo, los procedimientos requeridos en la realización del sistema de asignación de tickets, mostrando las actividades realizadas con su descripción específica, para la comprensión del total de su realización de inicio a fin.

Capítulo IV. “Resultados”. Se visualizan las consecuencias derivadas de los procesos en las actividades realizadas a lo largo del proyecto.

1.2 JUSTIFICACIÓN

Dada la situación anterior, la empresa 4CE expertos en sistemas ERP, S.C., dentro del departamento de operaciones, no cuenta con el suficiente control de la información recabada de los usuarios y colaboradores, lo que ocasiona desacuerdos con las funciones que lleva a cabo cada empresa, por lo tanto, no se tiene la organización de los incidentes de mayor prioridad, tomando en cuenta que los puntos de más importancia se generan a raíz de los problemas específicos de cada usuario.

Estos problemas se generan debido a la poca capacidad idónea de comunicación y deficiencia de análisis y comprensión de las necesidades y requerimientos de cada usuario.

Como consecuencia la empresa comienza a presentar las pérdidas de clientes como tiempo y dinero

De tal forma de que se requiere de una solución que apoye, a llevar una organización eficiente de la información, y al mismo tiempo se genere una gestión de tickets y automatización de los procesos de trabajo repetitivos, se monitoree el desempeño del Service desk, y se visualice la mejora de servicios con funcionalidades predefinidas para la generación de informes.

1.3 PLANTEAMIENTO DEL PROBLEMA

Las empresas son entes económicos dedicados a ofrecer productos, bienes y servicios para satisfacer las necesidades de la sociedad. Estas entidades requieren de diversos departamentos para coordinar su buen funcionamiento.

Actualmente en la empresa 4CE expertos en sistemas ERP, S.C., la falta de control se hace presente, como consecuencia, la información recabada de los usuarios y colaboradores se ve afectada, las operaciones realizadas suelen ser poco eficientes, esto por el hecho de que los métodos utilizados para realizar dicha gestión no disponen de un proceso competente teniendo en cuenta el nivel corporativo de la empresa a lo largo de los años que lleva operando.

Otra problemática presente en la empresa es la falta de soporte técnico en problemas específicos, ya que se cuenta con poca capacidad idónea de comunicación y deficiencia de análisis y comprensión de las necesidades del usuario.

El problema consiste en que no existe un punto central donde los usuarios puedan recibir ayuda ante los requerimientos que el problema actual les solicite, por lo tanto, no se tiene un seguimiento detallado de la problemática y no se tiene el almacenamiento correcto de los datos, de manera que, al no dar prioridad a los casos con más urgencia, la empresa se verá afectada debido al impacto de incidentes no verificados.

Para mantener la empresa operando con normalidad, correctamente y con eficacia se requiere de un sistema informático que ayude a satisfacer cada una de sus necesidades.

1.4 ANTECEDENTES

Al tener la oportunidad de conocer la situación de la empresa 4CE, se pudo comprender que cuenta con la innovación tecnológica suficiente para mejorar la productividad de sus servicios, de manera que aún no existía un sistema eficaz que se encargara de darle al cliente un soporte por diferentes medios para buscar una solución al problema por lo tanto sino se llevaba a cabo la realización del sistema de tickets, la empresa se vería afectada con serias repercusiones en los procesos empresariales, tomando en cuenta que debido a que si una falla informática se hacía presente, los involucrados también son los clientes y los empleados que dependen de esas operaciones.

Todos los servicios ya sean físicos o lógicos deben mantenerse conectados a sus diferentes áreas para poder obtener la información almacenada.

1.5 OBJETIVOS

1.5.1 OBJETIVO GENERAL

Desarrollar un sistema que permita la mejora en la gestión de datos e información de los clientes de la empresa 4CE

1.5.2 OBJETIVOS ESPECÍFICOS

- Analizar las necesidades de información del cliente para identificar una solución.
- Diseñar las partes estáticas y dinámicas de la solución técnica.
- Construir y validar los requisitos que fueron diseñados para la aplicación que permita la administración de los usuarios de acuerdo con la situación analizada.
- Implementación de la aplicación para la administración y gestión del usuario.

1.6 ALCANCES Y LIMITACIONES

ALCANCES

El presente proyecto estudiará la gestión de tickets de usuarios y colaboradores dentro de la empresa 4CE, en el departamento de operaciones.

Se pretenden realizar distintas actividades, como:

- Identificar el área que necesita mejora e iniciar los cambios necesarios, dentro del departamento de operaciones.
- La recolección y registro de datos de los clientes y colaboradores de la empresa.
- Realizar la gestión de las operaciones totales ejecutadas semanalmente
- Trazar el flujo de información en los procesos y finalmente determinar los requisitos del sistema de información de tickets.
- Reducir los incidentes generados a raíz de la poca organización en el análisis de la información del cliente.

Dando inicio con La recolección y registro de datos de los clientes y colaboradores de la empresa.

LIMITACIONES

- Insuficiente organización de los datos existentes de los clientes, y colaboradores de la empresa 4CE.
- Ingresos suficientes para la realización del proyecto.
- Tiempo insuficiente para la realización de los procesos determinados para la realización del sistema de tickets.
- Insuficientes elementos que son necesarios para las actividades cotidianas, como: equipo de cómputo, dinero y personas.

CAPÍTULO II

MARCO TEÓRICO

2.1 Introducción

El marco teórico se compone a través de dos marcos, siendo estos el conceptual y metodológico. El conceptual aborda aquellos conceptos que son necesarios para la investigación sobre el proyecto y, por último, el marco metodológico especifica los criterios, parámetros y normas que serán recurridas para la elaboración del proyecto. Ante lo anterior, se irán presentando conforme el orden adecuado.

William Daros, investigador argentino, afirma que el marco teórico tiene las siguientes funciones:

- Posibilita describir los problemas en las investigaciones, por ello también es llamando “marco referencial”.
- Es un instrumento fundamental para el análisis de los problemas de investigación.
- Da sentido a los hechos o fenómenos y orienta la organización de los mismos.
- Es el eje integrador de todo el proceso de investigación. Sin el marco teórico no tiene sentido el problema, no se puede elaborar un diseño metodológico con el cual probar las hipótesis.

MARCO CONCEPTUAL

2.2 Introducción a la Web

Los acontecimientos más importantes en el nacimiento de la Web se remontan a los años cuarenta.

Cuando hablamos de las Web, podemos creer que hablamos de un sinónimo de Internet, cuando en realidad son dos interpretaciones distintas, Internet es la red de redes donde reside toda la información, siendo un entorno de aprendizaje abierto, más allá de las instituciones educativas formales. La web es un subconjunto de Internet que contiene información a la que se puede acceder usando un navegador.

Existen tres elementos que fueron clave en el nacimiento de la Web.

HTML como lenguaje para crear los contenidos de la Web, basado en Standard Generalized Markup Language (SGML).

- HTTP como protocolo de comunicación entre los ordenadores de la Web, encargado de la transferencia de las páginas web y demás recursos.
- URL como medio de localización (direccionamiento) de los distintos recursos en Internet. (Berners-Lee, 1989).

2.2.1 El primer navegador

fue programado por “Tim BernersLee” a finales de 1990. Al principio lo llamó WorldWideWeb, pero después cambió el nombre por Nexus, ya que empezaba a usarse World Wide Web para referirse de forma genérica al sistema de comunicación que había ideado.

- El sistema operativo NeXTStep presenta un llamativo entorno gráfico multiventana.
- En la esquina superior izquierda se puede observar el menú del navegador, con su nombre en la primera línea (WorldWideWeb).

- Se pueden ver dos ventanas que muestran imágenes en línea (un libro y el logotipo del CERN). La primera versión del navegador mostraba las imágenes en una ventana aparte.
- El menú Links aparece abierto. Se está creando un enlace sobre la palabra ATLAS que aparece resaltada en la ventana con título CERN Experiments (la ventana que se encuentra en un primer plano).
- La X que tienen las ventanas permite cerrarlas (más tarde lo copiaría Microsoft).
- La ventana que aparece al fondo con título Tim's Home Page tiene la X incompleta porque se ha modificado y no se han guardado los cambios.(Berners-Lee, 1989).

2.2.2 Separación de funciones

Cliente / Servidor, es una arquitectura que nos permite la separación de funciones en tres niveles.

- Lógica de presentación. Se encarga de la entrada y salida de la aplicación con el usuario. Sus principales tareas son: obtener información del usuario, enviar la información del usuario a la lógica de negocio para su procesamiento, recibir los resultados del procesamiento de la lógica de negocio y presentar estos resultados al usuario.
- Lógica de negocio (o aplicación). Se encarga de gestionar los datos a nivel de procesamiento. Actúa de puente entre el usuario y los datos. Sus principales tareas son: recibir la entrada del nivel de presentación, interactuar con la lógica de datos para ejecutar las reglas de negocio (business rules) que tiene que cumplir la aplicación (facturación, cálculo de nóminas, control de inventario, etc.) y enviar el resultado del procesamiento al nivel de presentación.
- Lógica de datos. Se encarga de gestionar los datos a nivel de almacenamiento. Sus principales tareas son: almacenar los datos, recuperar los datos, mantener los datos y asegurar la integridad de los datos.
(Brennan, 2015).

2.2.3 Modelos de distribución en aplicaciones cliente / servidor

Una infraestructura cliente/servidor consta de tres componentes esenciales ellos son:

- Plataforma operativa: Debe soportar todos los modelos de distribución y deberá utilizar componentes estándar de la industria para los servicios de distribución.
- Entorno de desarrollo de aplicaciones: Debe posibilitar la coexistencia de procesos cliente/servidor, desarrollados mediante distintos lenguajes de programación o herramientas.
- gestión de sistemas: aumentan considerablemente el costo de una solución informática y deben acoplarse a las necesidades de la organización. (Carmona, 2010).

2.2.4 Arquitecturas de dos y tres niveles

Aunque todos los modelos de distribución en aplicaciones cliente/servidor que se han comentado antes se basan en arquitecturas de dos capas, normalmente cuando se habla de aplicaciones de dos niveles se está haciendo referencia a una aplicación donde el cliente mantiene la lógica de presentación, de negocio, y de acceso a los datos, y el servidor únicamente gestiona los datos. Suelen ser aplicaciones cerradas que supeditan la lógica de los procesos cliente al gestor de bases de datos que se está usando. En las arquitecturas de tres niveles, la lógica de presentación, la lógica de negocio y la lógica de datos están separadas, de tal forma que mientras la lógica de presentación se ejecutará normalmente en la estación cliente, la lógica de negocio y la de datos pueden estar repartidas entre distintos procesadores.

2.2.5 Descripción de un sistema cliente/servidor

En la arquitectura cliente/servidor, una aplicación se modela como un conjunto de servicios proporcionados por los servidores y un conjunto de clientes que usan este servicio. Los clientes necesitan saber que servidores están disponibles, pero

normalmente no conocen la existencia de otros clientes. Clientes y servidores son procesos diferentes.

Varios procesos servidores pueden ejecutarse sobre un único procesador servidor por lo tanto no hay necesariamente una correspondencia entre procesos y procesadores del sistema. (Sommerville, 2005).

2.2.6 El cliente

El cliente web es un programa con el que interacciona el usuario para solicitar a un servidor web el envío de los recursos que desea obtener mediante HTTP.

La parte cliente de las aplicaciones web suele estar formada por el código HTML que forma la página web más algo de código ejecutable realizado en lenguaje de script del navegador (JavaScript o VBScript) o mediante pequeños programas (applets) realizados en Java.

Por tanto, la misión del cliente web es interpretar las páginas HTML y los diferentes recursos que contienen (imágenes, sonidos, etc.). Las tecnologías que se suelen emplear para programar el cliente web son:

- HTML.
- CSS.
- DHTML.
- Lenguajes de script: JavaScript, VBScript, etc.
- ActiveX.
- Applets programados en Java.
- Distintas tecnologías que necesitan la existencia de un plug-in en el navegador:
- Adobe Acrobat Reader, Autodesk MapGuide, Live Picture PhotoVista, Macromedia Flash, Macromedia Shockwave, Virtual Reality Modeling Language (VRML), etc.

2.2.6 El servidor

El servidor web es un programa que está esperando permanentemente las solicitudes de conexión mediante el protocolo HTTP por parte de los clientes web. En los sistemas Unix suele ser un demonio y en los sistemas Microsoft Windows un servicio.

La parte servidor de las aplicaciones web está formada por:

- Páginas estáticas (documentos HTML) que siempre muestran el mismo contenido.
- Recursos adicionales (multimedia, documentos adicionales, etc.) que se pueden emplear dentro de las páginas o estar disponibles para ser descargados y ejecutados (visualizados) en el cliente.
- Programas o scripts que son ejecutados por el servidor web cuando el navegador del cliente solicita algunas páginas. La salida de este script suele ser una página HTML estándar que se envía al navegador del cliente. Tradicionalmente este programa o script que es ejecutado por el servidor web se basa en la tecnología CGI. En algunos casos pueden acceder a bases de datos.

2.3 Bases de datos relacionales

A consecuencia de los avances de la tecnología a través del tiempo, ha traído la informatización tanto de empresas y negocios, lo que nos ha obligado a crear programas informáticos para realizar las tareas que con anterioridad se realizaban a mano. Estos programas manejaban datos de entrada y salida, debido a que los datos tenían que almacenarse de alguna forma, para que los programas pudiesen acceder a ellos. El proceso de abstracción permite, interpretar, simplificar y reducir los parámetros y las elaciones del mundo real. Al interpretar la realidad, lo que hacemos realmente es indicar las propiedades que caracterizan esa realidad y establecer un modelo inicial que permita representarlo. (Cabello, 2010).

2.3.1 Descripción de base de datos

Llamamos base de datos al conjunto de datos dispuestos con el objetivo de brindar información a los usuarios y permitir transacciones como inserción, eliminación y actualización de datos, mismas que se construyen de diferentes formas como :

- Relacional
- Orientado a objetos
- Jerárquico
- Red
- Dimensional

(Arias, 2014).

2.3.2 Lenguajes de programación

SQL (Structure Query Language), es un lenguaje de consulta estructurado establecido claramente como el lenguaje de alto nivel estándar para sistemas de base de datos relacionales. Los responsables de publicar este lenguaje como estándar fueron precisamente los encargados de publicar estándar, la ANSI (Instituto Americano de Normalización) y la ISO (organismo Internacional de Normalización). Es por lo anterior que este lenguaje lo vas a encontrar en cualquiera de los DBMS relacionales que existen en la actualidad.

A continuación, se describen los tres tipos de sentencias particulares en los siguientes lenguajes:

- Lenguaje de Definición de Datos (DDL, Data Definiton Language) Grupo de sentencias del SQL que soportan la definición y declaración de los objetos de la base de datos. Objetos tales como: la base de datos misma(DATABASE), las tablas(TABLE), las Vistas (VIEW), los índices (INDEX), los procedimientos almacenados (PROCEDURE), los disparadores (TRIGGER).
Reglas (RULE), Dominios (Domain) y Valores por defecto (DEFAULT).
CREATE, ALTER y DROP.

- Lenguaje de Manipulación de Datos (DML, Data Management Language) Grupo de sentencias del SQL para manipular los datos que están almacenados en las bases de datos, a nivel de filas (tuplas) y/o columnas (atributos). Ya sea que se requiera que los datos sean modificados, eliminados, consultados o que se agregaren nuevas filas a las tablas de las bases de datos.

INSERT,

UPDATE,

DELETE y SELECT.

- Lenguaje de Control de Datos (DCL, Data Control Language): Grupo de sentencias del SQL para controlar las funciones de administración que realiza el DBMS, tales como la atomicidad y seguridad.

COMMIT TRANSACTION,

ROLLBACK TRANSACTION,

GRANT

REVOKE

(Pérez, 2010).

2.3.3 Lenguaje de definición de datos (DDL, SQL)

El conjunto de instrucciones SQL, contenidas en su lenguaje de definición de datos (LDD), son las responsables de realizar las modificaciones en la estructura de la base de datos. (Rodríguez, 2015).

En esta oportunidad veremos las sentencias correspondientes a la creación de los esquemas o lo que es lo mismo las tablas que contendrán los datos de la base de datos, La sentencia CREATE TABLE.

- La sentencia CREATE TABLE, define el nombre de la tabla, las columnas con su tipo de datos, las ligaduras de integridad que vigilan el valor que se guarde como dato en las columnas o atributos sean llaves o no.

Sintaxis:

```
CREATE TABLE nombre_tabla  
(  
campo1 tipo dato [NULL/NOT NULL] | CHECK (expresiónLógica) | [  
DEFAULT expresiónConstante],  
campo2 tipo dato [NULL/NOT NULL] | CHECK (expresiónLógica) | [  
DEFAULT expresiónConstante ],  
campo-N,  
PRIMARY KEY(campo_llave),  
FOREIGN KEY (campo_llave) REFERENCES tabla2 (campo_llave-tabla2)).
```

- Eliminación de Tablas:

La sentencia para eliminar una tabla y por ende todos los objetos asociados con esa tabla como: las vistas, disparadores, etc., DROP TABLE, donde r es el nombre de una tabla existente

- DROP TABLE r

Por ejemplo, si deseamos eliminar a la tabla ARTICULO_PANAMA, revisemos a ver si está ya existe, de la siguiente manera SELECT * from ARTICULO_PANAMA

Posteriormente una vez verificada que la tabla existe, se procede a borrar la tabla, escribiendo lo siguiente en la interfase de consultas del SQL.

- Sentencia ALTER: Por lo general, esta sentencia ALTER TABLE se utiliza sobre tablas que ya poseen desde cientos a miles de filas por ser tablas de un sistema de Base de Datos que ya está en producción.

2.3.4 Consultas Simples

El corazón o poder del Lenguaje SQL es el poder hacer consultas de cualquier tipo a la base de datos en forma no procedural. La sentencia SELECT es muy poderosa y ampliamente rica en sus cláusulas y variantes permitiendo la capacidad de atender en poco tiempo a consultas complejas sobre la base de datos.

Gracias a esta sentencia es que se pueden realizar todas las operaciones del Algebra Relacional.

La sentencia SELECT, obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas.

2.3.5 Otras sentencias

- Sentencia SUM:
SELECT SUM (cantidad) as 'Suma de las Cantidades' FROM detalle_ped
- Sentencia AVG:
SELECT AVG (cantidad) as 'Cantidad Promedio Pedida' FROM detalle_ped
- Sentencia MIN:
SELECT MIN (cantidad) as 'Cantidad Mínimo Pedida' FROM detalle_ped
- Sentencia MAX:
SELECT MAX (cantidad) as 'Cantidad Máxima Pedida'
FROM detalle_ped
- Sentencia COUNT
SELECT COUNT (*) as 'Total de Clientes' FROM CLIENTE
- Sentencia DISTINCT
SELECT COUNT(DISTINCT numped) as 'Total de Pedidos Solicitados'
FROM detalle_ped

En esta última consulta hace que los números de pedidos en DETALLE_PED , que son repetidos, la cláusula DISTINCT cuente solo números de uno de los que se repiten.

2.3.6 PL/SQL

el lenguaje de programación PL/SQL (Procedural Language/SQL) que se utiliza para acceder y trabajar con bases de datos Oracle desde distintos entornos. PL/SQL amplía la funcionalidad del lenguaje SQL añadiendo estructuras típicas de los lenguajes procedimentales, como los bloques, las variables y los tipos, las estructuras de control (bucles, órdenes IF-THEN-ELSE, etc.), los cursores, los procedimientos y las funciones, los paquetes, los disparadores, los tipos de objetos y los métodos, las excepciones y otras estructuras clásicas en los lenguajes de programación procedimentales. Se trata de un lenguaje robusto y potente que combina la flexibilidad de SQL con la potencia y la configurabilidad de las construcciones procedimentales y que incorpora las características de los lenguajes de programación orientados al objeto. (López, 2008).

El diseño de las tablas se desarrolló por medió la herramienta APEX que nos habilita el uso de las plantillas, llevándose así el registro de los requerimientos necesarios para capturar los datos y por consiguiente dar inicio al sistema de información web con Visual Builder y Oracle Cloud para la gestión de tickets de la empresa 4CE. creándose como primeros pasos las tablas con los siguientes prefijos seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

Para abordar de manera más específica lo que es APEX, daremos una breve introducción a su significado expresando que es una característica gratuita de la base de datos Oracle que nos permite crear aplicaciones web centralizadas utilizando únicamente nuestro navegador web (Orfali, 2017).

A medida que el tiempo transcurre, la evolución tecnológica, las medidas de seguridad en la nube, y los requerimientos de las empresas avanzan con el objetivo de poder acceder a la información que se encuentra en la nube de una manera más eficaz y menos costosa.

Existen dos servicios que nos proporcionan la utilización de las aplicaciones ejecutándose en una infraestructura que permite la accesibilidad del cliente de una forma ligera.

SaaS (Software as a Service) es la evolución natural del término software bajo demanda, con el que era conocido hace unos años y cuyo representante más genuino es Salesforce.com, empresa que proporciona software de gestión empresarial a petición de las necesidades del cliente. SaaS es un modelo de software basado en la web que proporciona el software totalmente disponible a través de un navegador web.

En un modelo SaaS, el usuario no tiene que preocuparse por conocer donde está alojado el software, que tipo de sistema operativo se utiliza o si está escrito en lenguaje PHP, Java o .Net. Además, el usuario no tiene que instalar ningún programa de software como si se hace en el modelo tradicional.

PaaS es similar a SaaS, aunque el servicio es un entorno completo de desarrollo de aplicaciones y no solo el uso de una aplicación. Las soluciones de PaaS se diferencian de SaaS en que proporcionan una plataforma de desarrollo virtual alojado en la nube y accesibles desde un navegador web.

NIST define PaaS como “La capacidad proporcionada al consumidor de desplegar en la infraestructura de la nube aplicaciones creadas por el consumidor o adquiridas y desarrolladas utilizando herramientas y lenguajes de programación soportados por el proveedor”. El consumidor no gestiona ni controla la infraestructura fundamental de la nube tales como redes, servidores, sistemas operativos o almacenamiento, pero tiene el control sobre las aplicaciones desplegadas; y en su caso configuraciones de entornos de alojamiento de aplicaciones (Aguilar, 2022).

Mediante las necesidades para solucionar problemáticas referentes a la informática, nos hemos visto en la obligación de crear herramientas que a través del tiempo han sido mejoradas debido a las actualizaciones de estas.

Se expresa que la informática es en la actualidad una de las tecnologías más importantes, ya que gracias al uso de las computadoras se han logrado grandes avances en la ciencia y tecnología, incluyendo los cambios en el empleo del internet, avances genéticos o el desarrollo de fármacos (Ramírez, 1999).

Hablando de necesidades informáticas como técnicas y procesos que como finalidad tienen almacenar, procesar, y transmitir datos de forma digital, abordaremos globalmente, una serie de conceptos que nos permitirán concebir los elementos básicos que conforman a un computador, sus operaciones básicas y sus componentes.

El computador tiene como tarea fundamental la ejecución de programas. Para realizar cualquiera de estas instrucciones es necesario realizar un ciclo de instrucción, que está dividido en dos fases.

- Fase de búsqueda de instrucción: Consiste en localizar la siguiente instrucción del programa que se debe realizar
- Fase de ejecución de la instrucción: Consiste en interpretar la instrucción que se debe realizar, mandando las señales adecuadas a todos los componentes del computador

El cálculo en programas normalmente es realizado por la unidad central de procesamiento (CPU), siendo su función principal interpretar las instrucciones de un programa informático mediante la realización de operaciones aritméticas, mediante los dispositivos de entrada/salida a través del tiempo estos fueron reemplazados por microprocesadores (Sabry, 2022).

Los microprocesadores representan un importante avance en la electrónica y se puede definir como un circuito integrado único que consta de miles de compuertas digitales que efectúan las funciones aritméticas, lógicas y de control de una computadora (Ramírez, 1986).

Un sistema operativo es un programa o más bien un conjunto de programas o algoritmos que actúa como intermediario entre el usuario de un computador y el hardware de un computador. (Silva, 2015).

El objetivo de la seguridad informática no solo consiste en las siete características que señala COBIT, que consiste en prevenir los principales ataques internos, externos, físicos y electrónicos sobre la información.

A demás en caso de que llegara a suceder cualquier daño a la información sea alteración o robo, también se debe contar con algún plan para recuperar la información en su totalidad, tomando en cuenta que no todos los planes de recuperación tienen el mismo costo ya que depende del daño que haya sufrido la información, incluyendo pérdida total.

La seguridad depende de todo el equipo del área de informática, ya que todo aquel colaborador debe estar al tanto de los planes a seguir para evitar los futuros daños, y de esta manera trabajando en conjunto se logre obtener una organización eficiente (Urbina 2016).

Red informática: Las redes informáticas también pueden considerarse como una subparte de las telecomunicaciones, la informática, la tecnología de la información y la ingeniería informática, ya que utilizan una tecnología que depende en gran medida de las diversas aplicaciones de estas corrientes científicas y de ingeniería. Basándose en el área de la comunicación, y en la capacidad de atender las necesidades específicas de determinadas multitudes. (Hush, 2020).

Modelo TCP/IP: Este modelo es un protocolo para comunicación en redes que permite que un equipo pueda comunicarse dentro de una red. Está basado en el modelo teórico OSI de capas con la que comparte 4 de ellas; sin embargo, ofrece muchas más opciones y es un modelo práctico. La comprensión de las características principales de la pila de protocolos de Internet TCP/IP posibilita la configuración de redes básicas, por lo que conocer TCP/IP es fundamental en cualquier formación centrada en trabajar con redes e internet. (Alves, 2021).

Las comunicaciones ópticas han significado uno de los mayores avances llevados a cabo en los últimos años para la transmisión de la información. Su tecnología se ha desarrollado a ritmo vertiginoso, consiguiendo sistemas capaces de transmitir información a velocidades cada vez más elevadas, a mayores distancias y con

menores costes, hasta el punto de constituir hoy el medio de transmisión de señales de mejores prestaciones, lo que permite su empleo masivo en todo tipo de redes de telecomunicaciones. La generación de nuevos servicios está produciendo una demanda creciente de capacidad de transmisión. Además, la globalización económica está creando una necesidad creciente de redes mundiales. La liberación de los mercados de telecomunicaciones y el impacto de la competencia también están contribuyendo a la aparición de nuevos operadores. Creemos que únicamente las soluciones basadas en redes ópticas pueden allanar el camino a la "revolución del ancho de banda", que es el resultado inevitable de la evolución en marcha. (Grazzini, 2021).

LAN tiene como objetivo fundamental el estudio de los Sistemas de Comunicaciones, orientado al estudio conceptual de los conceptos básicos de los sistemas, arquitectura y redes de comunicaciones, las Redes de Área Local, incidiendo particularmente en las funciones de administración y gestión, las Redes de Área Extensa, tratando las grandes redes públicas para comunicaciones de datos, profundizando en las de conmutación de paquetes IP y en la Red Digital de Servicios Integrados RDSI, y por último la Interconexión de Redes, con insistencia en los aspectos relacionados con la interconexión de redes locales y las redes de área extensa (Jordancalero, 2006).

Diferencias dentro de las redes en cuanto a su estructura: La red cliente/servidor nos indica que hay un servidor central que proporciona los datos y recursos a los clientes conectados. en este sistema se presupone que todos los procesos se desarrollan en un servidor de gran rendimiento, mientras que los clientes pueden ser estaciones de trabajo con menos prestaciones. La transmisión de datos se realiza siempre y exclusivamente a través del servidor y no es posible acceder directamente de una estación de trabajo a otra. (Frater, 2003).

Red de área personal: Es el intercambio de información entre dispositivos personales y su conexión a equipos periféricos. (Boquera, 2003).

HMTL: Es un lenguaje de marcas que es interpretado por el navegador web para presentarnos el contenido en pantalla. Es el protagonista sobre el que finalmente radica el resultado final del desarrollo web. (Gómez, 2021).

A continuación, se mencionan de forma breve dos protocolos de enlace de datos de amplio uso.

El primero, HDLC, es un protocolo clásico orientado a bits cuyas variantes se han utilizado durante décadas en muchas aplicaciones. El segundo PPP, es un protocolo de enlace utilizado para conectar a internet computadoras domésticas. (Tanenbaum, 2003).

MARCO METODOLÓGICO

Este proyecto se desarrolla en base al modelo de desarrollo incremental

El principio del desarrollo:

En el campo del desarrollo de software, existen dos grupos de metodologías, las denominadas tradicionales (formales) y las ágiles. Las primeras son un tanto rígidas, exigen una documentación exhaustiva y se centran en cumplir con el plan del proyecto definido totalmente en la fase inicial del desarrollo de este; mientras que la segunda enfatiza el esfuerzo en la capacidad de respuesta a los cambios, las habilidades del equipo y mantener una buena relación con el usuario. Ambas propuestas tienen sus propias ventajas y desventajas; de cualquier manera, las metodologías de desarrollo nos dicen el ¿Qué hacer? más no el ¿Cómo hacer?, esto significa que la metodología que elijamos debe ser adaptada al contexto del proyecto, teniendo en cuenta los recursos técnicos y humanos; tiempo de desarrollo y tipo de sistema. (Flores, 2017).

2.3.7 Términos y Definiciones

- Área de desarrollo: Se refiere a una unidad operativa dentro de la DISI en la cual se desarrollan productos de software. Por ejemplo: Coordinación de Transferencia, Tecnológica Interna, Coordinación de Transferencia Tecnológica Externa y coordinación de Tecnología Web son áreas de desarrollo.
- Cliente: Es la persona o empresa receptora de un bien, servicio, producto o idea, a cambio de dinero u otro artículo de valor.
- Coordinador: Es el responsable de gestionar el proyecto.
- Desarrollador: Es un especialista en informática que es capaz de concebir y elaborar sistemas informáticos; así como de implementarlos y ponerlos a punto, utilizando el lenguaje de programación definido.

- **Desarrollador Líder:** Tiene como función realizar el seguimiento y el control del cronograma de actividades.
- **Entregables:** Son los productos intermedios que generan las fases. Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.
- **Método:** Un método se compone de diversos aspectos que nos permitirán conseguir una meta o lograr un objetivo. Se define más claramente como un conjunto de herramientas, las cuales, utilizadas mediante las técnicas correctas, permiten la ejecución de procesos que nos llevarán a cumplir los objetivos que buscamos.
- **Metodología:** Hace referencia al conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige una investigación científica, una exposición doctrinada de tareas que requieran habilidades, conocimientos o cuidados específicos. Con frecuencia puede definirse la metodología como el estudio o elección de un método pertinente o adecuadamente aplicable a determinado objeto.
- **Metodología de Desarrollo de Software:** Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: Quién debe hacer Qué Cuándo y Cómo debe hacerlo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende actividades a seguir. (Flores, 2017).

2.3.8 Componentes de las metodologías

La metodología se define como la disciplina que indicará qué métodos y técnicas hay que usar en cada fase del ciclo de vida del desarrollo del proyecto. Los elementos que componen la metodología son:

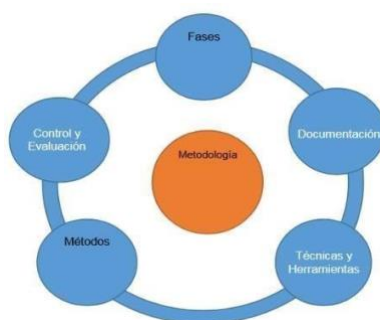


Figura 1. Elementos de la metodología (Flores, 2017).

- **Fases:** Son etapas del proceso de desarrollo de software. En la metodología se identificarán las diferentes actividades que se realizarán en cada fase.
- **Métodos:** Es necesario identificar el modo en el que se realizará el proceso de desarrollo del producto software. Se tendrá que descomponer los procesos en actividades más pequeñas, en estas actividades se definen los valores que recibirá cada fase, así como los que generará y la técnica que se tendrá que usar.
- **Técnicas y Herramientas.:** Indican como se deberá de resolver cada actividad y que herramientas podríamos usar.
- **Documentación:** Es necesario especificar qué documentación se va generar durante cada etapa del proceso; estos documentos deben realizarse de manera completa y usando todos los valores de entrada y salida que se van generando, esto servirá para recoger los resultados y tomar decisiones de las diferentes situaciones planteadas.
- **Control y Evaluación:** Las actividades de control y evaluación se deben de realizar a lo largo de todas las fases para identificar errores y corregirlos a tiempo. (Flores, 2017).

2.3.9 Cierre del marco

Por medio del uso del método incremental se consiguió proyectar cada actividad mediante un ciclo de vida de la metodología. Según el modelo de ciclo de vida que se basa en el modelo incremental, la sucesión de fases puede ampliarse con bucles de retroalimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución, aportaciones a los resultados intermedios que se van produciendo. La implementación de los recursos que esta metodología ofrece se utilizó de la manera más eficiente empleando las siguientes fases para lograr el desarrollo del proyecto.

- Análisis
- Diseño
- Codificación
- Pruebas
- Implementación

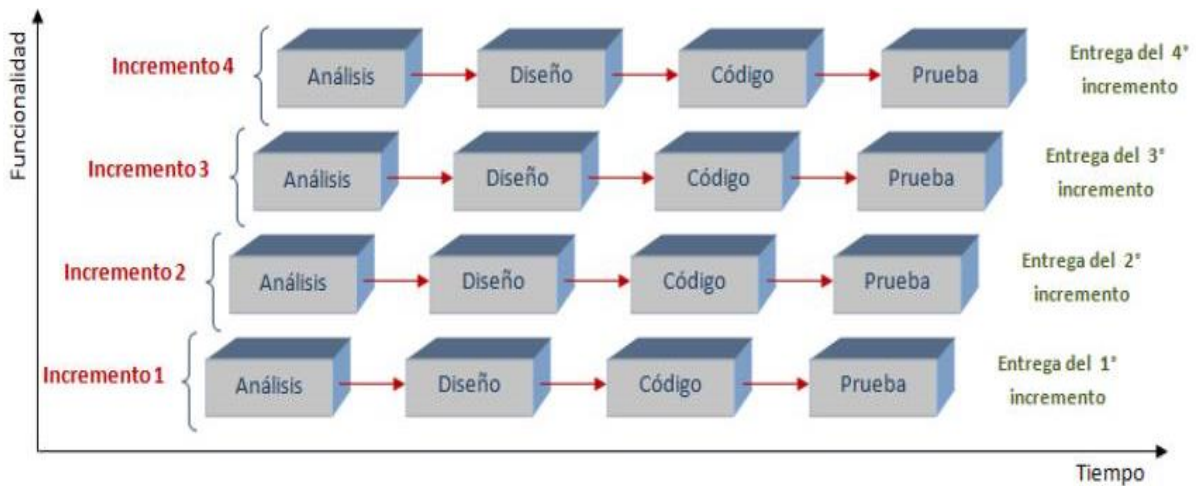


Figura 2. (Modelo de desarrollo incremental iterativo de software, 2023).

El uso del modelo incremental fue empleado para presentar las funciones básicas a partir de los requisitos básicos del proyecto, al ser este un modelo centrado en entrega de producto operativo por incrementos permitió que el proyecto fuera evolucionando conforme pasa el tiempo y nos permitiera analizar las fallas y malos procesos que se iban presentando con el uso del sistema. Con las evaluaciones que se hacían, se iban presentando versiones actualizadas más seguras y adaptadas para que su funcionamiento fuera cada vez mejor.

Usar este modelo nos ayuda a realizar el proyecto de manera más flexible, reduciendo así el coste en los cambios que los alcances proponen. De igual forma es de gran apoyo por la facilidad para realizar pruebas y realizar gestión de riesgos con cada iteración que se realice en un periodo comprendido de tiempo.

CAPÍTULO III

DESARROLLO

3.1 Propuesta de solución

Una solución genera expectativas en el mejoramiento del impacto económico de la empresa a partir de cierto tiempo de que se utilice esta nueva gestión de los datos y los resultados se presenten en el mejor manejo de la información, obteniendo productividad respondiendo a solicitudes similares con respuestas predefinidas que brinden una solución ante los incidentes.

La solución propuesta, se realizará mediante un sistema de información web con un servicio en la nube que te permite crear y alojar rápidamente aplicaciones web(Visual Builder), y una red global de centros de datos administrados por Oracle Cloud para la gestión de tickets de la empresa 4CE. De este modo se reduce el tiempo en la realización y visualización de los incidentes que cada usuario presente y así tengan un mayor alcance de información correcta y agilización de los procesos de resolución del problema técnico.

Tomando en cuenta que la importancia de obtener un sitio web determina los objetivos del proyecto, se puede concluir que generar clientes potenciales nos impacta de manera positiva, ya que se pueden convertir en compradores reales de nuestro servicio, y a su vez se proyecta lo que se hará en un futuro, por consiguiente los ingresos aumentan de forma porcentual, conservando las actualizaciones del sistema para así fidelizar a los clientes y mantener los servicios operando de manera periódica, por tal razón seguir intercambiando información entre los usuarios será de vital estimación para las mejoras continuas del sistema.

En este apartado atacaremos los problemas a los que los usuarios se enfrentan día a día, buscando que las cargas de trabajo estén equilibradas y la asignación de tickets se genere automáticamente, proporcionando a su vez consistencia en los mensajes, sin olvidar la seguridad en las contraseñas con el cifrado para obtener la protección de los datos, o ser implementada como medida preventiva por los administradores del sistema para buscar contraseñas fácilmente.

3.1.1 Componentes del modelo de datos

3.1.1.1 Prefijo de objetos de base de datos

Para unificar los nombres de los objetos creados se utilizará el prefijo XXCMX_ tanto para objetos de base de datos como de aplicación.

Nombres de métodos

La primera letra del nombre de un método debe ser minúscula. Todas las demás palabras deben ser escritas en letra capital. No deben existir espacios.

3.1.1.2 Tablas nuevas

Utilizar el prefijo de la aplicación Custom XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XXCMX_<Módulo>_<Descripción>[_ALL/_TMP]

Por ejemplo: XXCMX_AR_CUST_PROFILES_ALL

USUARIO/DUEÑO: El dueño de la tabla debe ser el usuario XXCMX, se debe crear siempre un sinónimo en el esquema de APPS y otorgar privilegios al usuario APPS.

3.1.1.3 Columnas adicionales

Las tablas deben contener las Who Columns, todas ellas deben ser NOT NULL:

- CREATION_DATE
- CREATED_BY
- LAST_UPDATE_DATE
- LAST_UPDATED_BY

Por su naturaleza, algunas tablas no requieren tener estos campos de auditoría, por ejemplo, las tablas temporales.

3.1.1.4 SQL Scripts

Generar siempre y de forma individual el script SQL para la creación de la tabla (y cualquier otro objeto de la base de datos) para uso futuro.

3.1.1.5 Vistas nuevas

Convención para Nombres

El nombre de la vista debe comenzar con el código de la aplicación Custom seguido del módulo y una descripción, colocando al final el sufijo “_V”.

XXCMX_<Módulo>_<Descripción>_V Hay dos tipos de vistas:

- Las vistas multi-org, son aquellas basadas en tablas _ALL que se encargan de filtrar el ORG_ID que corresponda. El nombre de la vista debe coincidir con el nombre de la tabla multi-org pero sin el _ALL (y agregando el sufijo _V).

Por ejemplo. Si la tabla multi-org se llama XXCMX_AR_PROFILES_ALL la vista entonces debe llamarse:

XXCMX_AR_PROFILES_V

- Las vistas Join son aquellas que involucran a más de una tabla en su definición. El nombre de estas vistas debería ser una combinación de los objetos involucrados agregándole el sufijo _V.

Por ejemplo. Si las tablas son: XXCMX_OM_ORDERS y XXCMX_OM_LINES y se quiere realizar una vista que muestre datos de ambas, el nombre de la misma deberá ser:

XXCMX_OM_ORDER_LINES_V.

USUARIO/DUEÑO: El dueño de las vistas debe ser el usuario APPS.

SQL SCRIPTS: Generar siempre de forma individual el script SQL para la creación de la vista.

3.1.1.6 Constraints

Convención para Nombres

El nombre del constraint debe ser siempre el mismo que el nombre de la tabla añadiendo al final el sufijo PK, Fn o Un según el tipo de constraint (Primary Key, Foreign Key o Unique).

<Nombre_Tabla>_PK/Fn/Un

- PK: Llave primaria
- Fn: Llave Foránea
- Donde: n: Es un número secuencial para identificar a cada uno de los constraints.

Por ejemplo:

XXCMX_AR_CUST_PROFILES_PK XXCMX_AR_CUST_PROFILES_F1
XXCMX_AR_CUST_PROFILES_U1

Los constraints en tablas multi-org y temporales: No deben incluir en su nombre los sufijos [_ALL/_TMP].

En el caso de los constraints tipo Check, la nomenclatura será la siguiente:

XXCMX_<Nombre_Campo>_Cn

Donde: n: Es un número secuencial para identificar a cada uno de los constraints.

USUARIO/DUEÑO: El dueño del constraint debe ser el usuario XXCMX.

SQL SCRIPTS: Las instrucciones para crear constraints deben ser incluidas dentro del script de creación de tablas.

3.1.1.7 Índices nuevos

Convención para Nombres

El nombre del índice debe ser siempre el mismo que el nombre de la tabla añadiendo al final el sufijo Nn.

<Nombre_Tabla>_Nn

- Nn : Es el sufijo para índices no únicos.

Donde:

n: Es un número secuencial para identificar a cada uno de los índices.

Por ejemplo:

XXCMX_AR_CUST_PROFILES > XXCMX_AR_CUST_PROFILES_N1.

Los índices en tablas multi-org y temporales: No deben incluir en su nombre los sufijos [_ALL/_TMP].

Notas:

- Por regla de negocio NO se permiten crear y/o emplear índices de tipo Parallel.
- En caso de requerir crear un índice custom a una tabla estándar, éste deberá de estar sustentado por un SR de Oracle.

USUARIO/DUEÑO: El dueño del índice debe ser el usuario XXCMX.

SQL SCRIPTS: Generar siempre los scripts SQL para la creación de los índices.

3.1.1.8 Secuencias nuevas

Convención para Nombres

El nombre de la secuencia debe ser siempre el mismo que el nombre de la tabla añadiendo al final el sufijo _S.

<Nombre_Tabla>_S

Por ejemplo: XXCMX_AR_CUST_PROFILES_S

Las secuencias en tablas multi-org y temporales: No deben incluir en su nombre los sufijos [_ALL/_TMP].

USUARIO/DUEÑO: El dueño de la secuencia debe ser el usuario XXCMX. Se debe crear siempre un sinónimo y otorgar privilegios al usuario APPS.

SQL SCRIPTS: Generar siempre y de forma individual el script SQL para la creación de la secuencia.

3.1.1.9 Código sql

Para código SQL seguir las siguientes recomendaciones:

Una sentencia SELECT simple que no devuelve registros, dispara una excepción NO_DATA_FOUND. Un INSERT, DELETE, o UPDATE que no afecta registros no dispara una excepción. Es necesario revisar explícitamente el valor de SQL%NOTFOUND si al no haber registros para la transacción, esto se considera un error.

BEGIN

UPDATE parts SET quantity = quantity - 1 WHERE partno = part_id;
check_status(part_id);

IF SQL%NOTFOUND THEN

... END IF;

END;

INSERT INTO rcv_transactions_interface

(interface_transaction_id, group_id, last_update_date,
last_updated_by,

creation_date, created_by) VALUES(v_rcv_transactions_interface,

v_group_id sysdate)

IF SQL%NOTFOUND THEN v_status_out := -501;

```
dbms_output.put_line('WARNING : No se llevo acabo la insercion en  
rcv_transactions_interface');
```

```
END IF;
```

3.1.1.10 Objetos de la librería

Oracle Applications dispone de una librería de objetos comunes para todos los módulos de la aplicación. Los más comunes son:

- Menús
- Funciones
- Flexfields
- Value Sets
- Ejecutables
- Concurrentes
- Grupos de Solicitudes
- Mensajes
- Opciones de Perfil

Todos estos objetos pueden ser definidos o modificados utilizando la responsabilidad de System Administrator y/o Application Developer.

La única norma al definir algunos objetos es anteponer el prefijo XXCMX_ seguido de las siglas del <Módulo>, posteriormente una descripción en el nombre y asociarle la aplicación Custom (XXCMX) para los objetos que lo requieran.

3.1.1.11 Responsabilidades (descripción de solo consulta)

En el caso de las responsabilidades custom la norma a seguir deberá de ser la siguiente:

El campo Nombre Responsabilidad deberá de contener las siglas del módulo a la que estará asociada la responsabilidad, seguido de la descripción; así mismo se dejarán de vincular a la aplicación Custom, y se asociarán a la aplicación del módulo que le corresponda.

Para el campo Clave Responsabilidad se deberá de ingresar el prefijo XXCMX_ seguido de las siglas del <Módulo> y posteriormente una descripción.

Aplicación: Receivables Clave Responsabilidad: XXCMX_AR_KRO_CONSULTA
Descripción: Responsabilidad de solo consulta

- **No usar HAVING si se puede poner la condición en el WHERE**

Es conveniente poner las condiciones en las cláusulas WHERE antes que en las HAVING (cuando se pueda). El uso de HAVING debería restringirse a la comparación con valores obtenidos en el agrupamiento como totales de cada grupo, mínimos, máximos, etc. Ejemplo:

Los clientes pueden pertenecer a 5 categorías distintas (A, B, C, D y E). Queremos obtener los saldos mayores a 500 de los clientes para las categorías A y B, totalizados por categoría.

```
SELECT SUM(SALDO) ... WHERE SALDO > 500 GROUP BY CATEGORIA  
HAVING CATEGORIA = 'A' OR CATEGORIA = 'B'
```

- **Usar los criterios más restrictivos al principio**

En búsquedas con criterios de selección, conviene usar los más restrictivos antes. Ejemplo: Quiero traer los Saldos menores a 500 de Clientes cuyo nombre comienza con la letra 'A'. Si muchos comienzan con 'A' y pocos saldos son menores a 500 será:

```
SELECT ...  
  
WHERE SALDO < 500  
  
AND NOMBRE_CLIENTE LIKE 'A%'
```

- **Evitar usar la sentencia DISTINCT**

La cláusula DISTINCT en una consulta, provoca que se efectúe un proceso de recuperación de los registros que cumplen con las condiciones, y luego filtra y

ordena los registros duplicados. Esto puede tener impacto en el performance, sobre todo en consultas complejas.

3.1.1.12 Trigger

Convención para Nombres

El nombre del trigger debe ser siempre el mismo que el nombre de la tabla añadiendo en seguida las siglas _XYZ y al final el sufijo _TRG.

<Nombre_Tabla>_XYZ_TRG

Donde:

X: Tomará el valor de A(After) o B(Before).

Y: Tomará uno de los siguientes valores I(insert), U(update) o Z(delete).

Z: Tomará el valor de S(statement) o R(Row).

Por ejemplo: XXCMX_AR_CUST_PROFILES_AIS_TRG

USUARIO/DUEÑO: El dueño del trigger debe ser el usuario XXCMX.

SQL SCRIPTS: Generar siempre y de forma individual el script SQL para la creación de los triggers.

3.1.1.13 Paquetes

Siempre se deben crear paquetes para todos los procedimientos de PL/SQL que se definan (no se deben crear funciones o procedimientos individuales).

Convención para Nombres

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

XXCMX_<Módulo>_<Descripción>_PKG

Por ejemplo: XXCMX_AR_CUSTOMERS_PKG.

Para los nombres de los procedimientos y funciones contenidos dentro de cada uno de los paquetes utilizar las normas de escribir todo el nombre en minúscula y separar las palabras con guion bajo “_”.

Por ejemplo: PROCEDURE create_item_value;

USUARIO/DUEÑO: El dueño de los paquetes debe ser el usuario APPS.

Tamaño de los paquetes

El tamaño del paquete está sujeto al tamaño de memoria del compilador.

SQL SCRIPTS: Generar por separado los scripts SQL para la creación del paquete.

Ejemplo:

XXCMX_ONT_ORDER_PKG.pkh: Especificación del paquete el cual contiene la definición de rutinas.

3.1.1.14 Documentación de procedimientos y funciones en los paquetes

Se debe de llevar un control de cada uno de los procedimientos y/o funciones que contienen los paquetes. De acuerdo con el siguiente formato de documentación.

Función:

```
/*=====
FUNCTION
    <Nombre de la función>
DESCRIPTION
    <Descripción de la aplicación de la función>.
ARGUMENTS:
    IN:
        <Nombre_parametro_entrada> <Descripción>
    OUT:
        <Nombre_parametro_salida> <Descripción>
    IN/OUT:
        <Nombre_parametro_entrada_salida> <Descripción>
RETURNS:
    <Descripción del valor que regresa>
NOTES
HISTORY
    Date          Author          Version      Change Reference
-----
    <Fecha de     <Nombre de la   <Número de  <Descripción del
```

Figura 3. Formato de documentación para una función [Elaborado por 4CE]

Procedimiento:

```
| PROCEDURE
|     <Nombre del Procedimiento>
|
| DESCRIPTION
|     <Descripción de la funcionalidad del procedimiento>
|
| ARGUMENTS:
|     IN:
|         <Nombre_parametro_entrada> <Descripción>
|
|     OUT:
|         <Nombre_parametro_salida> <Descripción>
|
|     IN/OUT:
|         <Nombre_parametro_entrada_salida> <Descripción>
|
| RETURNS:
|     <Descripción del valor que regresa>
|
| NOTES
|
| HISTORY
|
| Date           Author           Version      Change Reference
|-----
```

Figura 4. Formato de documentación para un procedimiento [Elaborada por 4CE]

3.1.1.15 Código PL/SQL

Editores de Código

No todos los editores de código tabulan de la misma manera. Un código fuente que se vea bien en un editor, puede no verse tan bien en otro. Esto se debe al distinto manejo del carácter “tab” que cada editor haga.

Para evitar problemas de legibilidad en el código, ningún programa fuente debe incluir tabs en su código. Esto significa que la tabulación deberá hacerse mediante espacios.

3.1.1.16 Nombre de variables

Se recomienda utilizar la siguiente norma para nombrar las variables de los procedimientos:

Variables locales: Anteponer el prefijo l_. Esto permite que los nombres de las variables utilizadas en las sentencias SQL no entren en conflicto con nombres de objetos de la base de datos. Ej: l_header_id.

Variables globales: Anteponer el prefijo g_ para distinguir las variables globales.

Ejemplo: g_set_of_books_id.

Parámetros: Utilizar el prefijo p_ para identificar los parámetros de los procedimientos. Ej.: p_customer_id. Para los parámetros de tipo OUT, utilizar el prefijo x_.

Ejemplo: x_object_version_number.

Registros: Para las variables de tipo registro anteponer siempre el prefijo r_. Por Ejemplo: r_order.

Cursores: Anteponer el prefijo c_.

Ejemplo: c_items.

3.1.1.17 Condición especial de variables

Los programas concurrentes con Stored Procedure, están restringidos a 100 parámetros en adición a los 2 primeros, los cuales son requeridos y deben de ser especificados exactamente como se indica. Se utiliza errbuf para devolver mensajes de error y retcode para devolver el estado de finalización.

Nombres de variables

Los nombres de las variables deben ser sustantivos, frases o abreviaturas de sustantivos. Las variables locales y parámetros deben ser cortos pero significativos, cuando varias palabras componen su nombre, la primera palabra debe de iniciar en minúscula, y el resto con letra capital, sin espacios.

- **Los índices se crean únicamente sobre campos no nulos**

No existen valores nulos en un índice. Esto significa que el uso del NULL en cualquier comparación provocará el no uso del índice. Ejemplo:

```
SELECT ... WHERE CODIGO = IS NULL.
```

Aquí, no se accederá al índice, aunque el mismo este definido sobre la columna CODIGO.

- **Evitar usar el índice si se accede a más del 20% de los registros**

Si más del 20% (aproximadamente) del total de los registros de la tabla cumple con la condición de búsqueda, conviene no usar el índice. Si se usa, el tiempo total de ejecución será mayor y se debe a la sobrecarga en procesamiento que causa la lectura del índice. Para evitar el uso de un índice, sumar 0 a los campos numéricos o concatenar un valor nulo (CAMPO || '') a los alfanuméricos. Ejemplo:

Si en la tabla de Saldos la mayoría de los registros tienen un valor mayor a 2000 y queremos traer todos los registros con valores mayores a 500, entonces la instrucción será:

```
SELECT ... WHERE SALDO > 500
```

CAPÍTULO IV

RESULTADOS

4.1 Análisis de resultados

En este apartado, se podrán observar los resultados que surgieron mediante la propuesta de solución al proyecto, describiendo y analizando los datos y modelados, haciendo uso de las herramientas necesarias para su correcto desarrollo.

Se muestra la representación de entidades de una base de datos. En este diagrama entidad-relación, se ilustra la interconexión de entidades, relaciones y sus atributos, con la finalidad de trazar el flujo de información en los procesos y finalmente determinar los requisitos del proyecto del sistema de información de tickets.

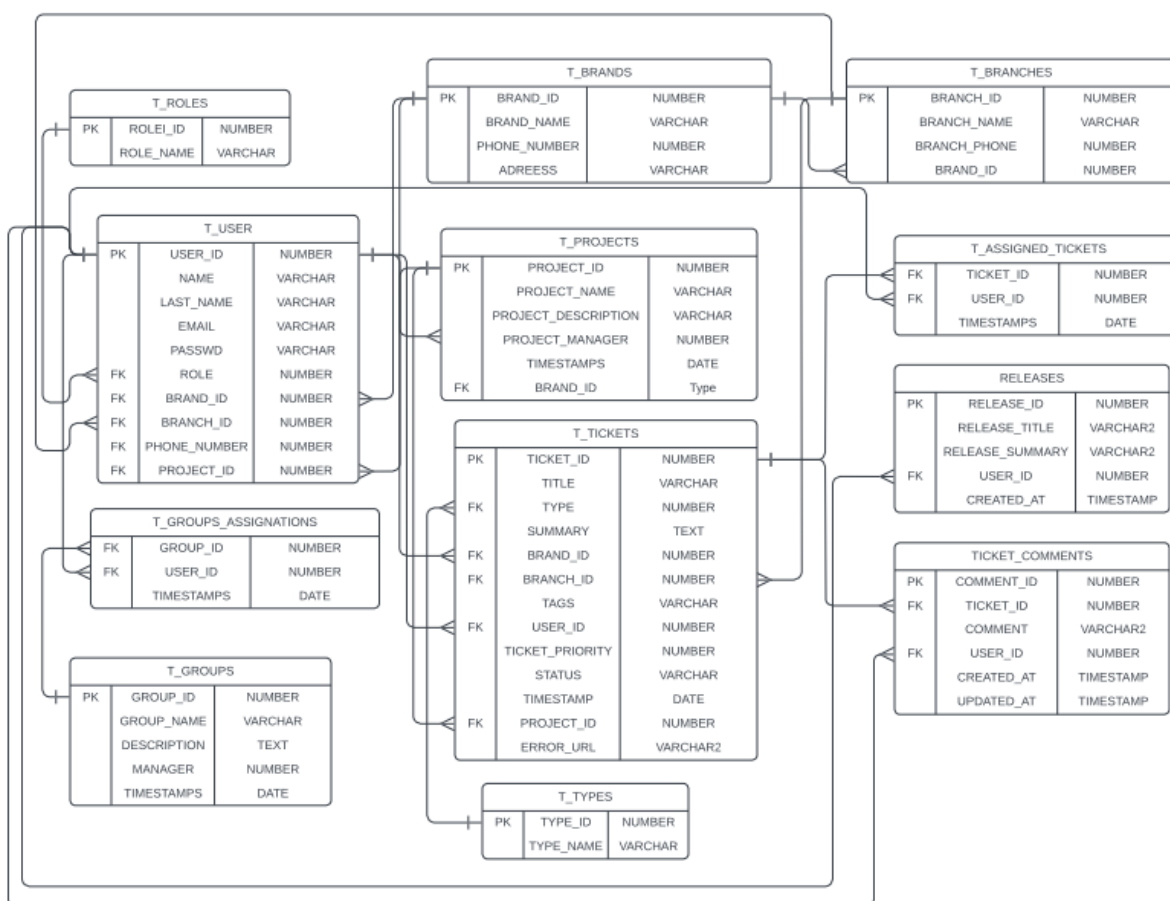


Figura 5. Diagrama Entidad/Relación [Autoría propia]

Se muestra la herramienta APEX, que nos permite desarrollar e implementar aplicaciones web en bases de datos, por lo que fue necesario crear una cuenta de manera personal para trabajar en ella los primeros diseños de las tablas propuestas en el modelo entidad – relación.

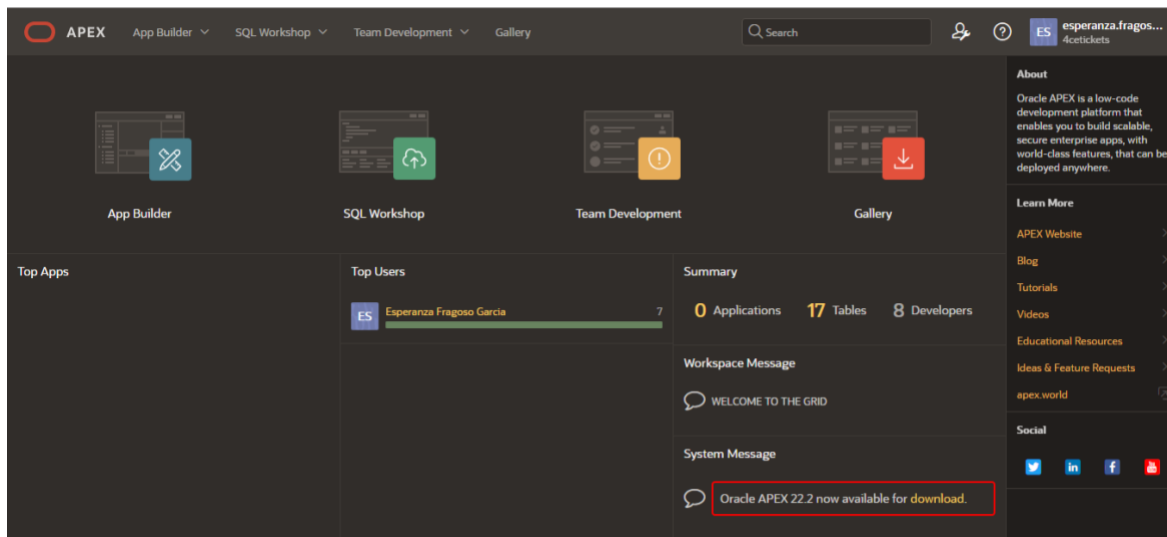


Figura 6. Perfil creado en la herramienta APEX

Se muestra la creación de la tabla “ROLES”, con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

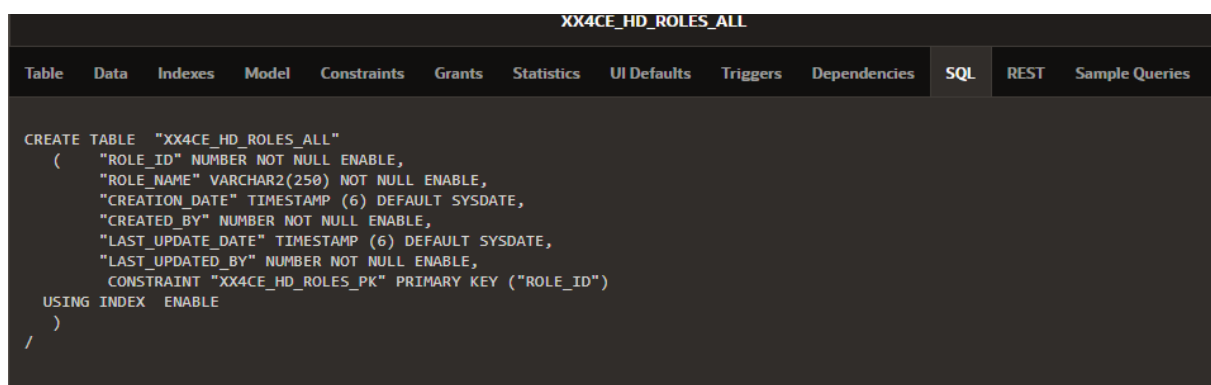


Figura 7. Tabla XX4CE_HD_ROLES_ALL [Autoría propia]

Se muestra la creación de la tabla “USERS”, con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_USERS_ALL										
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL
<pre> CREATE TABLE "XX4CE_HD_USERS_ALL" ("USER_ID" NUMBER NOT NULL ENABLE, "USER_NAME" VARCHAR2(250) NOT NULL ENABLE, "LAST_NAME" VARCHAR2(250) NOT NULL ENABLE, "EMAIL" VARCHAR2(250) NOT NULL ENABLE, "PASSWD" VARCHAR2(250) NOT NULL ENABLE, "ROLE_ID" NUMBER NOT NULL ENABLE, "BRAND_ID" NUMBER NOT NULL ENABLE, "BRANCH_ID" NUMBER NOT NULL ENABLE, "PHONE_NUMBER" NUMBER NOT NULL ENABLE, "PROJECT_ID" NUMBER NOT NULL ENABLE, "CREATION_DATE" TIMESTAMP (6) DEFAULT SYSDATE NOT NULL ENABLE, "CREATED_BY" NUMBER NOT NULL ENABLE, "LAST_UPDATE_DATE" TIMESTAMP (6) DEFAULT SYSDATE NOT NULL ENABLE, "LAST_UPDATED_BY" NUMBER NOT NULL ENABLE, CONSTRAINT "XX4CE_HD_USER_PK" PRIMARY KEY ("USER_ID") USING INDEX ENABLE) / ALTER TABLE "XX4CE_HD_USERS_ALL" ADD CONSTRAINT "XX4CE_HD_USER_F1" FOREIGN KEY ("ROLE_ID") REFERENCES "XX4CE_HD_ROLES_ALL" ("ROLE_ID") ENABLE / </pre>										

Figura 8. Tabla XX4CE_HD_USERS_ALL [Autoría propia]

Se muestra la creación de la tabla “GROUPS ASSIGNATIONS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_GROUPS_ASSIGNATIONS_ALL										
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL
<pre> CREATE TABLE "XX4CE_HD_GROUPS_ASSIGNATIONS_ALL" ("GROUP_ID" NUMBER, "USER_ID" NUMBER, "STATUS" VARCHAR2(255) DEFAULT 'ACTIVO', "CREATION_DATE" TIMESTAMP (6) DEFAULT SYSDATE CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_NN1" NOT NULL ENABLE, "CREATED_BY" NUMBER CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_NN2" NOT NULL ENABLE, "LAST_UPDATE_DATE" TIMESTAMP (6) DEFAULT SYSDATE CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_NN3" NOT NULL ENABLE, "LAST_UPDATED_BY" NUMBER CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_NN4" NOT NULL ENABLE, CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_C1" CHECK (STATUS IN ('ACTIVO','NO ACTIVO')) ENABLE) / ALTER TABLE "XX4CE_HD_GROUPS_ASSIGNATIONS_ALL" ADD CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_F1" FOREIGN KEY ("GROUP_ID") REFERENCES "XX4CE_HD_GROUPS_ALL" ("GROUP_ID") ON DELETE SET NULL ENABLE / ALTER TABLE "XX4CE_HD_GROUPS_ASSIGNATIONS_ALL" ADD CONSTRAINT "XX4CE_HD_GROUPS_ASSIGNATIONS_F2" FOREIGN KEY ("USER_ID") REFERENCES "XX4CE_HD_USERS_ALL" ("USER_ID") ON DELETE SET NULL ENABLE / </pre>										

Figura 9. Tabla XX4CE_HD_GROUPS_ASSIGNATIONS_ALL [Autoría propia]

Se muestra la creación de la tabla “GROUPS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

```

XX4CE_HD_GROUPS_ALL

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

CREATE TABLE "XX4CE_HD_GROUPS_ALL"
(
  "GROUP_ID" NUMBER,
  "GROUP_NAME" VARCHAR2(255),
  "DESCRIPTION" LONG,
  "MANAGER" NUMBER,
  "STATUS" VARCHAR2(255) DEFAULT 'ACTIVO',
  "CREATION_DATE" TIMESTAMP (6) DEFAULT SYSDATE CONSTRAINT "XX4CE_HD_GROUPS_NN1" NOT NULL ENABLE,
  "CREATED_BY" NUMBER CONSTRAINT "XX4CE_HD_GROUPS_NN2" NOT NULL ENABLE,
  "LAST_UPDATE_DATE" TIMESTAMP (6) DEFAULT SYSDATE CONSTRAINT "XX4CE_HD_GROUPS_NN3" NOT NULL ENABLE,
  "LAST_UPDATED_BY" NUMBER CONSTRAINT "XX4CE_HD_GROUPS_NN4" NOT NULL ENABLE,
  CONSTRAINT "XX4CE_HD_GROUPS_C1" CHECK (STATUS IN ('ACTIVO','NO ACTIVO')) ENABLE,
  CONSTRAINT "XX4CE_HD_GROUPS_PK" PRIMARY KEY ("GROUP_ID")
)
USING INDEX ENABLE
/

CREATE OR REPLACE EDITIONABLE TRIGGER "XX4CE_HD_GROUPS_BIR_TRG"
BEFORE INSERT ON XX4CE_HD_GROUPS_ALL
FOR EACH ROW
BEGIN
  SELECT XX4CE_HD_GROUPS_S.NEXTVAL
  INTO
  :NEW.GROUP_ID
  FROM
  DUAL;
END;

```

Figura 10. Tabla XX4CE_HD_GROUPS_ALL [Autoría propia]

Se muestra la creación de la tabla “BRANDS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

```

XX4CE_HD_BRANDS_ALL

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL REST Sample Queries

CREATE TABLE "XX4CE_HD_BRANDS_ALL"
(
  "BRAND_ID" NUMBER,
  "BRAND_NAME" VARCHAR2(255),
  "PHONE_NUMBER" NUMBER,
  "ADDRESS" VARCHAR2(255),
  "CREATION_DATE" TIMESTAMP (6) DEFAULT SYSDATE,
  "CREATED_BY" NUMBER NOT NULL ENABLE,
  "LAST_UPDATE_DATE" TIMESTAMP (6) DEFAULT SYSDATE,
  "LAST_UPDATED_BY" NUMBER NOT NULL ENABLE,
  PRIMARY KEY ("BRAND_ID")
)
USING INDEX ENABLE
/

CREATE OR REPLACE EDITIONABLE TRIGGER "XX4CE_HD_BRANDS_ALL_BIR_TRG"
BEFORE INSERT ON XX4CE_HD_BRANDS_ALL
FOR EACH ROW
BEGIN
  IF :NEW.BRAND_ID IS NULL THEN
    SELECT XX4CE_HD_BRANDS_ALL_S.nextval into :NEW.BRAND_ID FROM dual;
  END IF;
END;
/
ALTER TRIGGER "XX4CE_HD_BRANDS_ALL_BIR_TRG" ENABLE
/

```

Figura 11. Tabla XX4CE_HD_BRANDS_ALL [Autoría propia]

Se muestra la creación de la tabla “PROJECTS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

En esta figura se puede visualizar la modificación de la tabla, donde el campo agregado fue la llave foránea.

```

CREATE TABLE "XX4CE_HD_PROJECTS_ALL"
(
  "PROJECT_ID" NUMBER,
  "PROJECT_NAME" VARCHAR2(255),
  "PROJECT_DESCRIPTION" VARCHAR2(255),
  "PROJECT_MANAGER" NUMBER,
  "BRAND_ID_P" NUMBER,
  "CREATION_DATE" TIMESTAMP (6) DEFAULT SYSDATE,
  "CREATED_BY" NUMBER NOT NULL ENABLE,
  "LAST_UPDATE_DATE" TIMESTAMP (6) DEFAULT SYSDATE,
  "LAST_UPDATED_BY" NUMBER NOT NULL ENABLE,
  PRIMARY KEY ("PROJECT_ID")
  USING INDEX ENABLE
)
/
ALTER TABLE "XX4CE_HD_PROJECTS_ALL" ADD FOREIGN KEY ("BRAND_ID_P")
REFERENCES "XX4CE_HD_BRANDS_ALL" ("BRAND_ID") ENABLE
/

```

Figura 12. Tabla XX4CE_HD_PROJECTS_ALL [Autoría propia]

Se muestra la creación de la tabla “TICKETS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

```

CREATE TABLE "XX4CE_HD_TICKETS_ALL"
(
  "TICKET_ID" NUMBER NOT NULL ENABLE,
  "TICKET_TITLE" VARCHAR2(200) NOT NULL ENABLE,
  "TICKET_TYPE" NUMBER NOT NULL ENABLE,
  "TICKET_SUMMARY" VARCHAR2(800) NOT NULL ENABLE,
  "ERROR_URL" VARCHAR2(600),
  "BRAND_ID" NUMBER NOT NULL ENABLE,
  "BRANCH_ID" NUMBER NOT NULL ENABLE,
  "TAGS" VARCHAR2(500) NOT NULL ENABLE,
  "USER_ID" NUMBER NOT NULL ENABLE,
  "PROJECT_ID" NUMBER NOT NULL ENABLE,
  "TICKET_PRIORITY" NUMBER NOT NULL ENABLE,
  "STATUS" VARCHAR2(100) DEFAULT 'SIN RESOLVER',
  "CREATED_BY" NUMBER NOT NULL ENABLE,
  "CREATION_DATE" TIMESTAMP (6),
  "LAST_UPDATE_BY" NUMBER NOT NULL ENABLE,
  "LAST_UPDATE_DATE" TIMESTAMP (6),
  CONSTRAINT "XX4CE_HD_TICKETS_ALL_PK" PRIMARY KEY ("TICKET_ID")
  USING INDEX ENABLE
)
/

```

Figura 13. Tabla XX4CE_HD_TICKETS_ALL [Autoría propia]

Se muestra la creación de la tabla “TYPES” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_TYPES_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre> CREATE TABLE "XX4CE_HD_TYPES_ALL" ("TYPE_ID" NUMBER, "TYPE_NAME" VARCHAR2(250), "CREATED_BY" NUMBER, "LAST_UPDATED_BY" NUMBER, "CREATED_AT" TIMESTAMP (6), "UPDATED_AT" TIMESTAMP (6), PRIMARY KEY ("TYPE_ID") USING INDEX ENABLE) / </pre>											

Figura 14. Tabla XX4CE_HD_TYPES_ALL [Autoría propia]

Se muestra la creación de la tabla “BRANCHES” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_BRANCHES_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre> CREATE TABLE "XX4CE_HD_BRANCHES_ALL" ("BRANCH_ID" NUMBER NOT NULL ENABLE, "BRANCH_NAME" VARCHAR2(250), "BRANCH_PHONE" NUMBER, "CREATED_BY" NUMBER NOT NULL ENABLE, "CREATION_DATE" TIMESTAMP (6), "LAST_UPDATE_BY" NUMBER NOT NULL ENABLE, "LAST_UPDATE_DATE" TIMESTAMP (6), "BRAND_ID" NUMBER NOT NULL ENABLE, CONSTRAINT "XX4CE_HD_BRANCHES_PK" PRIMARY KEY ("BRANCH_ID") USING INDEX ENABLE) / ALTER TABLE "XX4CE_HD_BRANCHES_ALL" ADD CONSTRAINT "XX4CE_HD_BRANCHES_CREATE_FN" FOREIGN KEY ("CREATED_BY") REFERENCES "XX4CE_HD_USERS_ALL" ("USER_ID") ENABLE / ALTER TABLE "XX4CE_HD_BRANCHES_ALL" ADD CONSTRAINT "XX4CE_HD_BRANCHES_FN" FOREIGN KEY ("BRAND_ID") REFERENCES "XX4CE_HD_BRANDS_ALL" ("BRAND_ID") ENABLE / ALTER TABLE "XX4CE_HD_BRANCHES_ALL" ADD CONSTRAINT "XX4CE_HD_BRANCHES_UPDATE_FN" FOREIGN KEY ("LAST_UPDATE_BY") REFERENCES "XX4CE_HD_USERS_ALL" ("USER_ID") ENABLE / </pre>											

Figura 15. Tabla XX4CE_HD_BRANCHES_ALL [Autoría propia]

Se muestra la creación de la tabla “ASSIGNED_TICKETS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_ASSIGNED_TICKETS_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre>CREATE TABLE "XX4CE_HD_ASSIGNED_TICKETS_ALL" ("TICKET_ID" NUMBER, "USER_ID" NUMBER, "CREATED_BY" NUMBER, "UPDATED_BY" NUMBER, "CREATED_AT" TIMESTAMP (6), "UPDATED_AT" TIMESTAMP (6))</pre>											

Figura 16. Tabla XX4CE_HD_ASSIGNED_TICKETS_ALL [Autoría propia]

Se muestra la creación de la tabla “RELASES” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_RELEASES_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre>CREATE TABLE "XX4CE_HD_RELEASES_ALL" ("RELEASE_ID" NUMBER NOT NULL ENABLE, "RELEASE_TITLE" VARCHAR2(250), "RELEASE_SUMMARY" VARCHAR2(250), "CREATION_DATE" TIMESTAMP (6) NOT NULL ENABLE, "CREATE_BY" NUMBER NOT NULL ENABLE, "LAST_UPDATE_DATE" TIMESTAMP (6) NOT NULL ENABLE, "LAST_UPDATED_BY" NUMBER NOT NULL ENABLE, "COMMENT_ID" NUMBER NOT NULL ENABLE, "DIRECTED_TO" VARCHAR2(200) NOT NULL ENABLE, PRIMARY KEY ("RELEASE_ID")) USING INDEX ENABLE</pre>											

Figura 17. Tabla XX4CE_HD_RELEASES_ALL [Autoría propia]

Se muestra la creación de la tabla “TICKETS COMMENTS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_TICKET_COMMENTS_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre> CREATE TABLE "XX4CE_HD_TICKET_COMMENTS_ALL" ("COMMENT_ID" NUMBER NOT NULL ENABLE, "COMMENTS" VARCHAR2(250), "CREATION_DATE" TIMESTAMP (6) NOT NULL ENABLE, "CREATE_BY" NUMBER NOT NULL ENABLE, "LAST_UPDATE_DATE" TIMESTAMP (6) NOT NULL ENABLE, "LAST_UPDATED_BY" NUMBER NOT NULL ENABLE, "TICKET_ID" NUMBER NOT NULL ENABLE, PRIMARY KEY ("COMMENT_ID") USING INDEX ENABLE) </pre>											

Figura 18. Tabla XX4CE_HD_TICKET_COMMENTS_ALL [Autoría propia]

Se muestra la creación de la tabla “TICKET PRIORITY” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

XX4CE_HD_TICKET_PRIORITY_ALL											
Table	Data	Indexes	Model	Constraints	Grants	Statistics	UI Defaults	Triggers	Dependencies	SQL	REST
<pre> CREATE TABLE "XX4CE_HD_TICKET_PRIORITY_ALL" ("PRIO_ID" NUMBER NOT NULL ENABLE, "PRIO_NAME" VARCHAR2(100) NOT NULL ENABLE) </pre>											

Figura 19. Tabla XX4CE_HD_TICKET_PRIORITY_ALL [Autoría propia]

Se muestra la creación de la tabla “STATUS” , con el prefijo XX, seguido de las siglas del módulo o sistema satelital y posteriormente de la descripción de la tabla, así como también debe de finalizar con el sufijo ALL para las tablas multi-org o TMP para tablas temporales.

```

CREATE TABLE "XX4CE_HD_STATUS_ALL"
(
  "STATUS_ID" NUMBER,
  "STATUS_NAME" VARCHAR2(200),
  "STATUS_VALUE" VARCHAR2(200)
)
/

```

Figura 20. Tabla XX4CE_HD_STATUS_ALL [Autoría propia]

CREACIÓN DE PAQUETES

Siempre se deben crear paquetes para todos los procedimientos de PL/SQL que se definan (no se deben crear funciones o procedimientos individuales).

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

(a)

```

1 create or replace package body PKG_XX4CE_HD_ROLES_ALL
2 AS
3 PROCEDURE insert_value(l_role_id IN NUMBER, l_role_name IN VARCHAR2,
4                       l_created_by IN NUMBER, l_last_updated_by IN NUMBER, l_message OUT VARCHAR2)
5
6 AS
7 BEGIN
8 INSERT INTO XX4CE_HD_ROLES_ALL(ROLE_ID, ROLE_NAME, CREATED_BY, LAST_UPDATED_BY)
9 VALUES(l_role_id, l_role_name, l_created_by, l_last_updated_by);
10 l_message:= 'OK';
11 DBMS_OUTPUT.PUT_LINE(l_message);
12 EXCEPTION
13 WHEN NO_DATA_FOUND
14 THEN
15 l_message:= 'ERROR';
16 DBMS_OUTPUT.PUT_LINE(l_message);
17 END insert_value;
18 END;

```

(b)

```

1 create or replace PACKAGE PKG_XX4CE_HD_ROLES_ALL
2 AS
3 PROCEDURE insert_value(l_role_id IN NUMBER, l_role_name IN VARCHAR2,
4                       l_created_by IN NUMBER, l_last_updated_by IN NUMBER, l_message OUT VARCHAR2);
5 END;
6

```

Figura 21. PackagePKG_XX4CE_HD_ROLES_ALL: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package body XX4CE_HD_USERS_ALL_PKG
2 as
3   PROCEDURE insert_value(i_user_name IN VARCHAR2, i_last_name IN VARCHAR2, i_email IN VARCHAR2,
4     i_password IN VARCHAR2, i_role_id IN NUMBER, i_brand_id IN NUMBER, i_branch_id IN NUMBER,
5     i_phone_number IN NUMBER, i_project_id IN NUMBER, i_created_by IN NUMBER, i_last_updated_by IN NUMBER,
6     i_message OUT VARCHAR2)
7   IS
8   BEGIN
9     INSERT INTO XX4CE_HD_USERS_ALL
10     VALUES (XX4CE_HD_USERS_ALL.S.nextval, i_user_name, i_last_name, i_email, i_password, i_role_id, i_brand_id,
11     i_branch_id, i_phone_number, i_project_id, i_created_by, i_created_by, i_created_by, i_created_by);
12     i_message := 'OK';
13     DBMS_OUTPUT.PUT_LINE(i_message);
14   EXCEPTION
15     WHEN NO_DATA_FOUND
16     THEN
17       i_message := 'ERROR';
18       DBMS_OUTPUT.PUT_LINE(i_message);
19   END insert_value;
20

```

(a)

```

1 create or replace package XX4CE_HD_USERS_ALL_PKG
2 as
3   PROCEDURE insert_value(i_user_name IN VARCHAR2, i_last_name IN VARCHAR2, i_email IN VARCHAR2,
4     i_password IN VARCHAR2, i_role_id IN NUMBER, i_brand_id IN NUMBER, i_branch_id IN NUMBER,
5     i_phone_number IN NUMBER, i_project_id IN NUMBER, i_created_by IN NUMBER, i_last_updated_by IN NUMBER,
6     i_message OUT VARCHAR2);
7
8 end;

```

(b)

Figura 22. PackageXX4CE_HD_USERS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package body XX4CE_HD_GROUPS_ASSIGNATIONS_ALL_PKG
2 as
3   PROCEDURE insert_group_assignment_pd(v_group_id IN NUMBER,
4     v_user_id IN NUMBER,
5     v_created_by IN NUMBER,
6     v_last_updated_by IN NUMBER,
7     v_message OUT VARCHAR2)
8   IS
9   BEGIN
10     INSERT INTO XX4CE_HD_GROUPS_ASSIGNATIONS_ALL (GROUP_ID, USER_ID, CREATED_BY, LAST_UPDATED_BY )
11     VALUES (v_group_id, v_user_id, v_created_by, v_last_updated_by);
12     v_message := 'Registro exitoso';
13   EXCEPTION
14     WHEN NO_DATA_FOUND THEN
15       v_message := 'Error en los datos';
16     WHEN OTHERS THEN
17       v_message := ('Ocurrió un Error '||SUBSTR(SQLERRM, 1, 100));
18   END insert_group_assignment_pd;
19

```

(a)

```

1 create or replace package XX4CE_HD_GROUPS_ASSIGNATIONS_ALL_PKG
2 as
3   PROCEDURE insert_group_assignment_pd(v_group_id IN NUMBER,
4     v_user_id IN NUMBER,
5     v_created_by IN NUMBER,
6     v_last_updated_by IN NUMBER,
7     v_message OUT VARCHAR2);
8
9   PROCEDURE update_group_assignment_pd(v_group_id IN NUMBER,
10     v_user_id IN NUMBER,
11     v_status IN VARCHAR2,
12     v_last_updated_by IN NUMBER,
13     v_message OUT VARCHAR2);
14
15   PROCEDURE delete_group_assignment_pd(v_group_id IN NUMBER,
16     v_user_id IN NUMBER,
17     v_message OUT VARCHAR2);
18
19 end;

```

(b)

Figura 23. PackageXX4CE_HD_GROUPS_ASSIGNATIONS_ALL_PKG: a) creación de la estructura del paquete, b) creación de la especificación del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.



Figura 24. PackageXX4CE_HD_GROUPS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

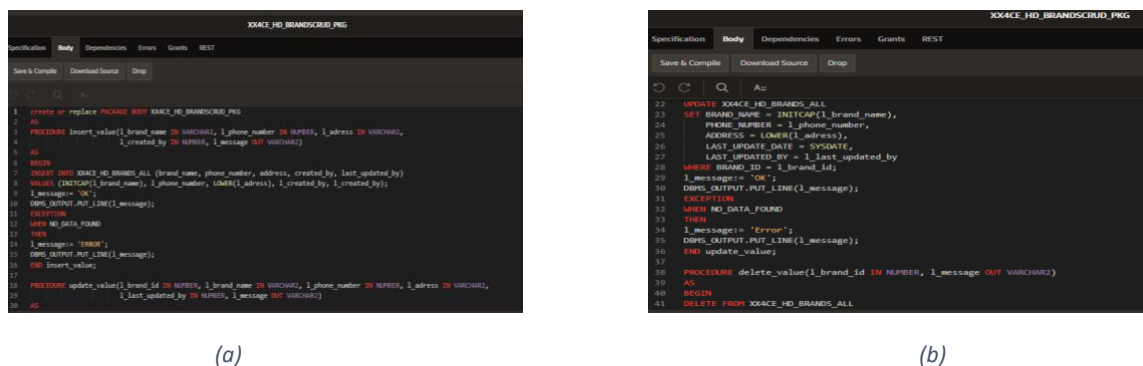


Figura 25. PackageXX4CE_HD_BRANDS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package BODY XX4CE_HD_PROJECTS_ALL_PKG
2 as
3 procedure insert_value(l_project_name in varchar2, l_project_description in varchar2, l_project_manager in number,
4 l_brand_id in number, l_created_by in number, l_message out varchar2)
5 as
6 begin
7 insert into XX4CE_HD_PROJECTS_ALL (PROJECT_NAME, PROJECT_DESCRIPTION, PROJECT_MANAGER,
8 l_brand_id, l_created_by, l_last_updated_by)
9 values (upper(l_project_name), upper(l_project_description), l_project_manager, l_brand_id,
10 l_created_by, l_created_by);
11 l_message := 'OK';
12 dbms_output.put_line(l_message);
13 exception
14 when no_data_found
15 then
16 l_message := 'ERROR';
17 dbms_output.put_line(l_message);
18 end insert_value;
19 procedure update_value(l_project_id in number, l_project_name in varchar2, l_project_description in varchar2,
20 l_project_manager in number, l_brand_id in number, l_last_updated_by in number,
21 l_message out varchar2)
22 as
23 begin

```

(a)

```

1 create or replace package XX4CE_HD_PROJECTS_ALL_PKG
2 as
3 procedure insert_value(l_project_name in varchar2, l_project_description in varchar2, l_project_manager in number,
4 l_brand_id in number, l_created_by in number, l_message out varchar2);
5 procedure update_value(l_project_id in number, l_project_name in varchar2, l_project_description in varchar2,
6 l_project_manager in number, l_brand_id in number, l_last_updated_by in number,
7 l_message out varchar2);
8 procedure delete_value(l_project_id in number, l_message out varchar2);
9 end;

```

(b)

Figura 26. Package XX4CE_HD_PROJECTS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package BODY XX4CE_HD_TICKETS_ALL_PKG
2 as
3 procedure insert_ticket(l_ticket_title in varchar2, l_ticket_type in varchar2, l_ticket_summary in varchar2, l_ticket_brand in number, l_ticket_branch in number, l_ticket_status in number, l_ticket_created_by in number, l_ticket_last_updated_by in number, l_ticket_message out varchar2)
4 as
5 begin
6 insert into XX4CE_HD_TICKETS_ALL (TICKET_TITLE, TICKET_TYPE, TICKET_SUMMARY, TICKET_BRAND, TICKET_BRANCH, TICKET_STATUS, TICKET_CREATED_BY, TICKET_LAST_UPDATED_BY)
7 values (upper(l_ticket_title), upper(l_ticket_type), upper(l_ticket_summary), l_ticket_brand, l_ticket_branch, l_ticket_status, l_ticket_created_by, l_ticket_created_by);
8 l_ticket_message := 'OK';
9 dbms_output.put_line(l_ticket_message);
10 exception
11 when no_data_found
12 then
13 l_ticket_message := 'ERROR';
14 dbms_output.put_line(l_ticket_message);
15 end insert_ticket;
16 procedure update_ticket(l_ticket_id in number, l_ticket_title in varchar2, l_ticket_type in varchar2, l_ticket_summary in varchar2, l_ticket_brand in number, l_ticket_branch in number, l_ticket_status in number, l_ticket_last_updated_by in number, l_ticket_message out varchar2)
17 as
18 begin

```

(a)

```

1 create or replace package XX4CE_HD_TICKETS_ALL_PKG
2 as
3 procedure insert_ticket(l_ticket_title in varchar2, l_ticket_type in varchar2, l_ticket_summary in varchar2, l_ticket_brand in number, l_ticket_branch in number, l_ticket_status in number, l_ticket_created_by in number, l_ticket_last_updated_by in number, l_ticket_message out varchar2);
4 procedure update_ticket(l_ticket_id in number, l_ticket_title in varchar2, l_ticket_type in varchar2, l_ticket_summary in varchar2, l_ticket_brand in number, l_ticket_branch in number, l_ticket_status in number, l_ticket_last_updated_by in number, l_ticket_message out varchar2);
5 procedure delete_ticket(l_ticket_id in number, l_ticket_message out varchar2);
6 end;

```

(b)

Figura 27. Package XX4CE_HD_TICKETS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package BODY_XXCMX_4CE_HD_TYPES_ALL as
2
3   /**insertar valores*/
4   PROCEDURE Insertion_valores(ticket_id_value IN NUMBER, type_name_value IN VARCHAR2,
5     type_created_by_value IN NUMBER, type_last_updated_by_value IN NUMBER, type_created_at IN timestamp,
6     type_last_updated_at IN timestamp,
7     l_message OUT VARCHAR2);
8
9   /**Actualizar*/
10  PROCEDURE Actualization_types(
11    update_id IN NUMBER,
12    update_created_by IN NUMBER,
13    update_last_updated_by IN NUMBER,
14    update_message OUT VARCHAR2);
15
16  /**Eliminar*/
17  PROCEDURE DELETE_TYPES(delete_id IN NUMBER, delete_message OUT VARCHAR2);
18
19 END;

```

(a)

```

1 create or replace package XXCMX_4CE_HD_TYPES
2 as
3   /**insertar*/
4   PROCEDURE Insertion_valores(ticket_id_value IN NUMBER, type_name_value IN VARCHAR2,
5     type_created_by_value IN NUMBER, type_last_updated_by_value IN NUMBER, type_created_at IN timestamp,
6     type_last_updated_at IN timestamp,
7     l_message OUT VARCHAR2);
8
9   /**Actualizar*/
10  PROCEDURE Actualization_types(
11    update_id IN NUMBER,
12    update_created_by IN NUMBER,
13    update_last_updated_by IN NUMBER,
14    update_message OUT VARCHAR2);
15
16  /**Eliminar*/
17  PROCEDURE DELETE_TYPES(delete_id IN NUMBER, delete_message OUT VARCHAR2);
18
19 END;

```

(b)

Figura 28. PackageXX4CE_HD_TYPES_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package BODY_XXCMX_4CE_HD_ASSIGNED_TICKETS as
2
3   /**insertar valores*/
4   PROCEDURE Insertion_valores(ticket_id_value IN NUMBER, ticket_user_id_value IN NUMBER,
5     ticket_created_by_value IN NUMBER, ticket_updated_by_value IN NUMBER,
6     ticket_created_at_value IN timestamp,
7     ticket_updated_at_value IN timestamp, message OUT VARCHAR2 );
8
9   /**Actualizar*/
10  PROCEDURE Actualization_tickets(ticket_id_value IN NUMBER, ticket_user_id_value IN NUMBER,
11    ticket_created_by_value IN NUMBER, ticket_updated_by_value IN NUMBER,
12    ticket_created_at_value IN timestamp,
13    ticket_updated_at_value IN timestamp, message OUT VARCHAR2 );
14
15  /**Eliminar*/
16  PROCEDURE DELETE_ASSIGNED(ticket_id_value IN NUMBER, message OUT VARCHAR2);
17
18 END;

```

(a)

```

1 create or replace package XXCMX_4CE_HD_ASSIGNED_TICKETS
2 as
3   /**insertar*/
4   PROCEDURE Insertion_valores(ticket_id_value IN NUMBER, ticket_user_id_value IN NUMBER,
5     ticket_created_by_value IN NUMBER, ticket_updated_by_value IN NUMBER,
6     ticket_created_at_value IN timestamp,
7     ticket_updated_at_value IN timestamp,
8     message OUT VARCHAR2 );
9
10  /**Actualizar*/
11  PROCEDURE Actualization_tickets(ticket_id_value IN NUMBER, ticket_user_id_value IN NUMBER,
12    ticket_created_by_value IN NUMBER, ticket_updated_by_value IN NUMBER,
13    ticket_created_at_value IN timestamp,
14    ticket_updated_at_value IN timestamp,
15    message OUT VARCHAR2 );
16
17  /**Eliminar*/
18  PROCEDURE DELETE_ASSIGNED(ticket_id_value IN NUMBER, message OUT VARCHAR2);
19
20 END;

```

(b)

Figura 29. PackageXX4CE_HD_ASSIGNED_TICKETS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

(a)

(b)

Figura 30. PackageXX4CE_HD_RELASES_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

(a)

(b)

Figura 31. PackageXX4CE_HD_TICKET_COMMENTS_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

El nombre de los paquetes debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete, así como también debe finalizar con el sufijo _PKG.

```

1 create or replace package users_login
2 is
3     procedure login_user(p_usuario varchar2, p_password varchar2, p_role_id number, p_usuario_id number, p_usuario_nombre varchar2)
4     is
5     begin
6         insert into users_login_all values (users_login_all_s.nextval, 'ROLE_N', creation_date, created_by, last_update_date, updated_by);
7         commit;
8     end;
9 end users_login;

```

(a)

```

1 create or replace package users_login
2 is
3     procedure login_user(p_usuario varchar2, p_password varchar2, p_role_id number, p_usuario_id number, p_usuario_nombre varchar2);
4 end users_login;

```

(b)

Figura 32. PackageXX4CE_HD_USERS_LOGIN_ALL_PKG: a) creación de la estructura del paquete, b) especificación de la estructura del paquete [Autoría propia]

CREACIÓN DE PROCEDIMIENTOS

Se debe de llevar un control de cada uno de los procedimientos y/o funciones que contienen los paquetes.

Para los nombres de los procedimientos y funciones contenidos dentro de cada uno de los paquetes utilizar las normas de escribir todo el nombre en minúscula y separar las palabras con guion bajo “_”.

Por ejemplo: PROCEDURE create_item_value;

```

1 create or replace procedure STORE_ROLES(ROLE_ID NUMBER, ROLE_NAME VARCHAR2, CREATION_DATE TIMESTAMP, CREATED_BY NUMBER, LAST_UPDATE_DATE TIMESTAMP, UPDATED_BY NUMBER, MESSEGE OUT VARCHAR2);
2
3 begin
4     insert into xx4ce_hd_roles_all values (xx4ce_hd_roles_all_s.nextval, 'ROLE_N', creation_date, created_by, last_update_date, updated_by);
5     commit;
6 end;

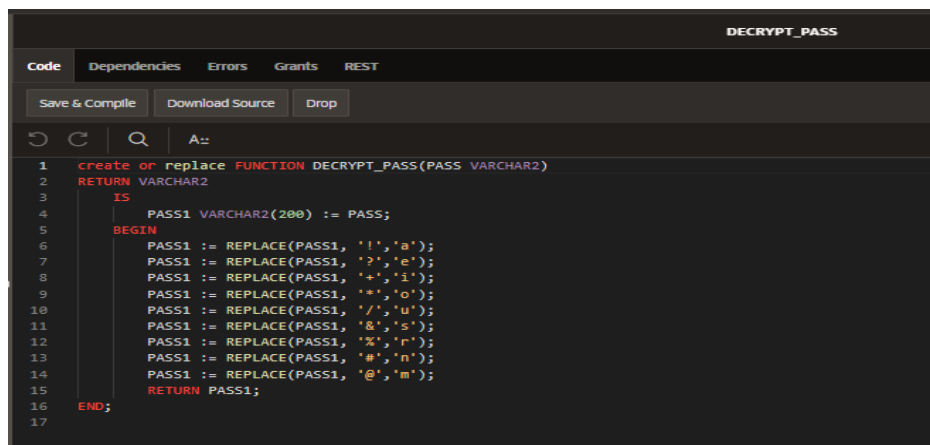
```

Figura 33. PROCEDURE_STORE_ROLES [Autoría propia]

CREACION DE FUNCIONES

Se debe de llevar un control de cada uno de los procedimientos y/o funciones que contienen los paquetes.

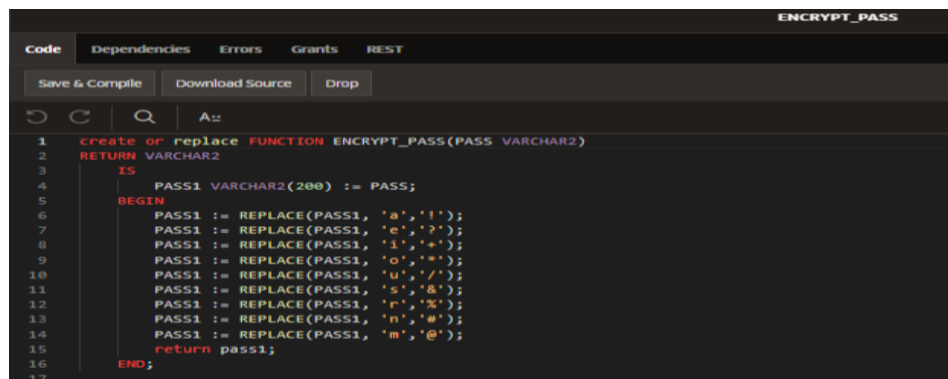
El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete. En la siguiente figura se podrá visualizar la realización, de la función para descryptar las contraseñas para los fines distintos del sistema.



```
1 create or replace FUNCTION DECRYPT_PASS(PASS VARCHAR2)
2 RETURN VARCHAR2
3 IS
4     PASS1 VARCHAR2(200) := PASS;
5 BEGIN
6     PASS1 := REPLACE(PASS1, '!', 'a');
7     PASS1 := REPLACE(PASS1, '?', 'e');
8     PASS1 := REPLACE(PASS1, '+', 'i');
9     PASS1 := REPLACE(PASS1, '*', 'o');
10    PASS1 := REPLACE(PASS1, '/', 'u');
11    PASS1 := REPLACE(PASS1, '&', 's');
12    PASS1 := REPLACE(PASS1, '%', 'r');
13    PASS1 := REPLACE(PASS1, '#', 'n');
14    PASS1 := REPLACE(PASS1, '@', 'm');
15    RETURN PASS1;
16 END;
```

Figura 34. DECRYPT_PASS [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete. En la siguiente figura se podrá visualizar la realización, de la función para encriptar las contraseñas para los fines distintos del sistema.

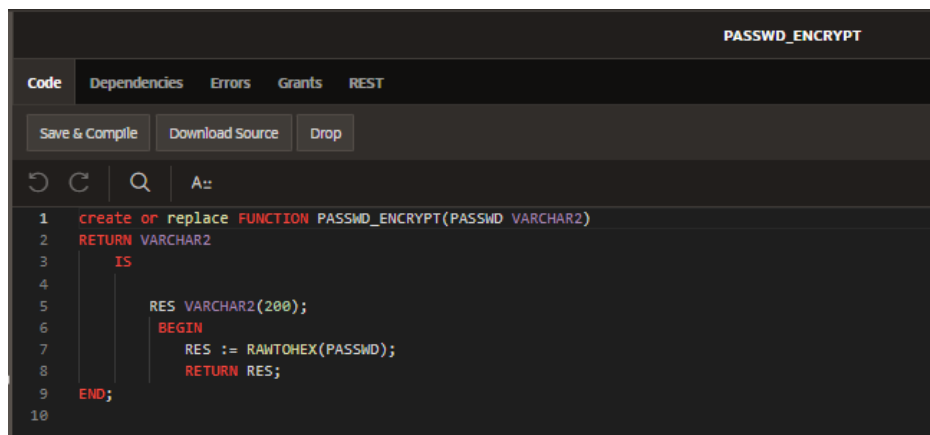


```
1 create or replace FUNCTION ENCRYPT_PASS(PASS VARCHAR2)
2 RETURN VARCHAR2
3 IS
4     PASS1 VARCHAR2(200) := PASS;
5 BEGIN
6     PASS1 := REPLACE(PASS1, 'a', '!');
7     PASS1 := REPLACE(PASS1, 'e', '?');
8     PASS1 := REPLACE(PASS1, 'i', '+');
9     PASS1 := REPLACE(PASS1, 'o', '*');
10    PASS1 := REPLACE(PASS1, 'u', '/');
11    PASS1 := REPLACE(PASS1, 's', '&');
12    PASS1 := REPLACE(PASS1, 'r', '%');
13    PASS1 := REPLACE(PASS1, 'n', '#');
14    PASS1 := REPLACE(PASS1, 'm', '@');
15    return pass1;
16 END;
```

Figura 35. ENCRYPT_PASS [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete.

En la siguiente figura se podrá visualizar la realización, de la variable “PASSWD_ENCRYPT”, seguido de la creación de la función para cifrar la contraseña.

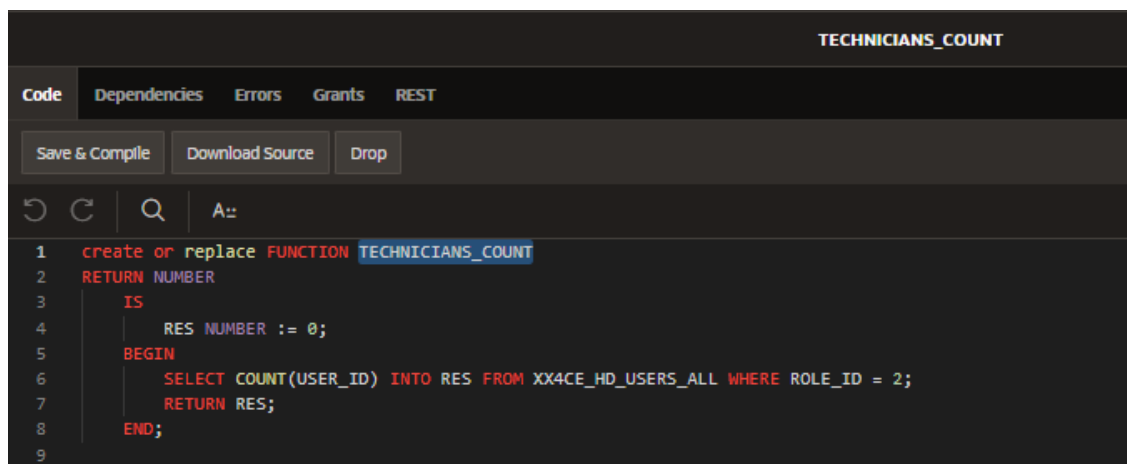


```
1 create or replace FUNCTION PASSWD_ENCRYPT(PASSWD VARCHAR2)
2 RETURN VARCHAR2
3 IS
4
5     RES VARCHAR2(200);
6 BEGIN
7     RES := RAWTOHEX(PASSWD);
8     RETURN RES;
9 END;
```

Figura 36. PASSWD_ENCRYPT [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete.

En la siguiente figura se podrá visualizar la realización, de la función “TECHNICIANS_COUNT”

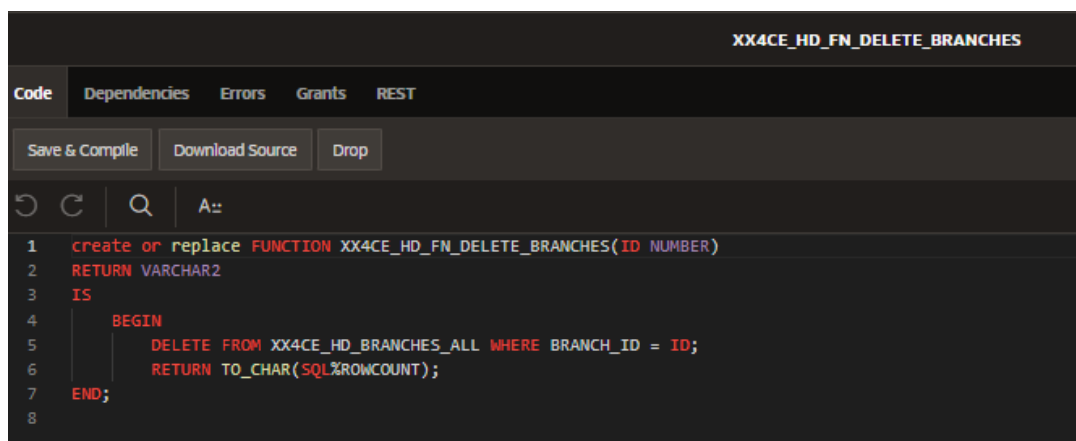


```
1 create or replace FUNCTION TECHNICIANS_COUNT
2 RETURN NUMBER
3 IS
4     RES NUMBER := 0;
5 BEGIN
6     SELECT COUNT(USER_ID) INTO RES FROM XX4CE_HD_USERS_ALL WHERE ROLE_ID = 2;
7     RETURN RES;
8 END;
```

Figura 37. TECHNICIANS_COUNT [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete.

En la siguiente figura se podrá visualizar la realización, de la función “DELETE_BRANCHES”

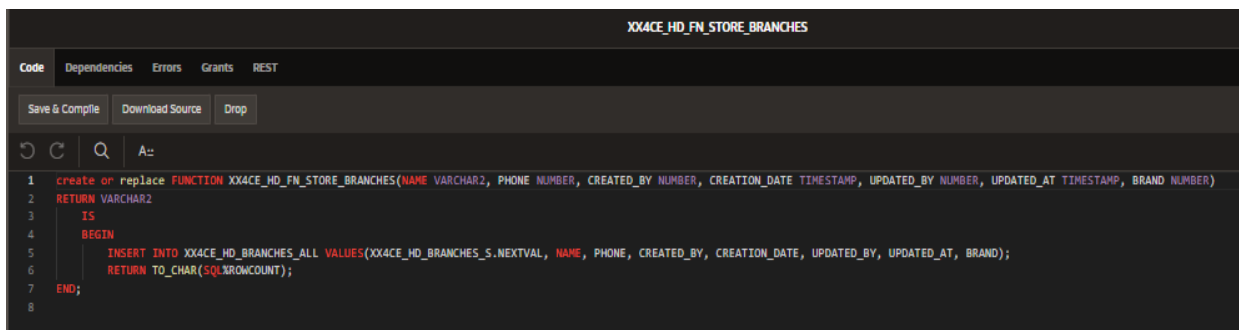


```
1 create or replace FUNCTION XX4CE_HD_FN_DELETE_BRANCHES(ID NUMBER)
2 RETURN VARCHAR2
3 IS
4 BEGIN
5     DELETE FROM XX4CE_HD_BRANCHES_ALL WHERE BRANCH_ID = ID;
6     RETURN TO_CHAR(SQL%ROWCOUNT);
7 END;
8
```

Figura 38. XX4CE_HD_FN_DELETE_BRANCHES [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete.

En la siguiente figura se podrá visualizar la realización, de la función “STORE_BRANCHES”



```
1 create or replace FUNCTION XX4CE_HD_FN_STORE_BRANCHES(NAME VARCHAR2, PHONE NUMBER, CREATED_BY NUMBER, CREATION_DATE TIMESTAMP, UPDATED_BY NUMBER, UPDATED_AT TIMESTAMP, BRAND NUMBER)
2 RETURN VARCHAR2
3 IS
4 BEGIN
5     INSERT INTO XX4CE_HD_BRANCHES_ALL VALUES(XX4CE_HD_BRANCHES_S.NEXTVAL, NAME, PHONE, CREATED_BY, CREATION_DATE, UPDATED_BY, UPDATED_AT, BRAND);
6     RETURN TO_CHAR(SQL%ROWCOUNT);
7 END;
8
```

Figura 39. XX4CE_HD_FN_STORE_BRANCHES [Autoría propia]

El nombre de las funciones debe comenzar siempre con el prefijo XXCMX_ seguido de las siglas del módulo o sistema satelital y posteriormente la descripción del paquete.

En la siguiente figura se podrá visualizar la realización, de la función “STORE BRANCHES”

XX4CE_HD_FN_UPDATE_BRANCHES

Code

Dependencies

Errors

Grants

REST

Save & Compile

Download Source

Drop

A::

```
1 create or replace FUNCTION XX4CE_HD_FN_UPDATE_BRANCHES(ID NUMBER, NAME VARCHAR2, PHONE NUMBER, UPDATED_BY NUMBER, UPDATED_AT TIMESTAMP, BRAND NUMBER)
2 RETURN VARCHAR2
3 IS
4 BEGIN
5     UPDATE XX4CE_HD_BRANCHES_ALL
6     SET BRANCH_NAME = NAME,
7         BRANCH_PHONE = PHONE,
8         LAST_UPDATE_BY = UPDATED_BY,
9         LAST_UPDATE_DATE = UPDATED_AT,
10        BRAND_ID = BRAND
11     WHERE BRANCH_ID = ID;
12     RETURN TO_CHAR(SQL%ROWCOUNT);
13 END;
14
```

Figura 40. XX4CE HD FN UPDATE BRANCHES [Autoría propia]

CREACIÓN DE SECUENCIAS

El nombre de la secuencia debe ser siempre el mismo que el nombre de la tabla añadiendo al final el sufijo S.

<Nombre Tabla> S

Por ejemplo: XXCMX AR CUST PROFILES S

Las secuencias en tablas multi-org y temporales: No deben incluir en su nombre los sufijos [ALL/ TMP].

En la siguiente figura, se puede observar la realización de una secuencia para la tabla “ROLES”

XX4CE_HD_ROLES_ALL_S

Object Details	Grants	Dependencies	SQL
CREATE SEQUENCE "XX4CE_HD_ROLES_ALL_S" MINVALUE 1 MAXVALUE 9999999999999999999 INCREMENT BY 1 START WITH 181 CACHE 20 ORDER NOCYCLE NOKEEP NOSCALE GLOBAL /			

Figura 41. XX4CE HD ROLES ALL S [Autoría propia]

XX4CE_HD_USERS_ALL_S

Object Details	Grants	Dependencies	SQL
CREATE SEQUENCE "XX4CE_HD_USERS_ALL_S" MINVALUE 1 MAXVALUE 9999999999999999999 INCREMENT BY 1 START WITH 421 CACHE 20 ORDER NOCYCLE NOKEEP NOSCALE GLOBAL /			

Figura 42. XX4CE_HD_USERS_ALL_S [Autoría propia]

XX4CE_HD_GROUPS_S					
Object Details	Grants	Dependencies	SQL		
<pre>CREATE SEQUENCE "XX4CE_HD_GROUPS_S" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 6 NOCACHE NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>					

Figura 43. XX4CE_HD_GROUPS_ALL_S [Autoría propia]

XX4CE_HD_BRANDS_ALL_S					
Object Details	Grants	Dependencies	SQL		
<pre>CREATE SEQUENCE "XX4CE_HD_BRANDS_ALL_S" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 441 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>					

Figura 44. XX4CE_HD_BRANDS_ALL_S [Autoría propia]

XX4CE_HD_PROJECTS_ALL_5				
Object Details	Grants	Dependencies	SQL	
<pre>CREATE SEQUENCE "XX4CE_HD_PROJECTS_ALL_5" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 341 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>				

Figura 45. XX4CE_HD_PROJECTS_ALL_S [Autoría propia]

XX4CE_HD_TICKETS_ALL_S					
Object Details	Grants	Dependencies	SQL		
<pre>CREATE SEQUENCE "XX4CE_HD_PROJECTS_ALL_S" MINVALUE 1 MAXVALUE 9999999999999999999 INCREMENT BY 1 START WITH 341 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>					

Figura 46. XX4CE_HD_TICKETS_ALL_S [Autoría propia]

XX4CE_HD_BRANCHES_S				
Object Details	Grants	Dependencies	SQL	
<pre>CREATE SEQUENCE "XX4CE_HD_BRANCHES_S" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 181 CACHE 20 ORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>				

Figura 47. XX4CE_HD_BRANCHES_ALL_S [Autoría propia]

XX4CE_HD_RELEASES_S				
Object Details	Grants	Dependencies	SQL	
<pre>CREATE SEQUENCE "XX4CE_HD_RELEASES_S" MINVALUE 1 MAXVALUE 99999999999 INCREMENT BY 1 START WITH 241 CACHE 20 NOORDER CYCLE NOKEEP NOSCALE GLOBAL /</pre>				

Figura 48. XX4CE_HD_RELEASES_ALL_S [Autoría propia]

XX4CE_HD_TICKET_COMMENTS_S				
Object Details	Grants	Dependencies	SQL	
<pre>CREATE SEQUENCE "XX4CE_HD_TICKET_COMMENTS_S" MINVALUE 1 MAXVALUE 99999999999 INCREMENT BY 1 START WITH 381 CACHE 20 ORDER CYCLE NOKEEP NOSCALE GLOBAL /</pre>				

Figura 49. XX4CE_HD_TICKET_COMMENTS_ALL_S [Autoría propia]

XX4CE_HD_TICKET_PRIORITY_S				
Object Details	Grants	Dependencies	SQL	
<pre>CREATE SEQUENCE "XX4CE_HD_TICKET_PRIORITY_S" MINVALUE 1 MAXVALUE 99999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 ORDER NOCYCLE NOKEEP NOSCALE GLOBAL /</pre>				

Figura 50. XX4CE_HD_TICKET_PRIORITY_ALL_S [Autoría propia]

4.1.1 Creación de servicios en APEX

A continuación, se podrá visualizar la creación de los módulos de cada una de las tablas creadas con anterioridad, utilizando los cruds que se desarrollaron con las acciones de insertar, actualizar y eliminar en cada función, procedimientos o paquetes.

The screenshot shows the 'ORDS Handler Definition' window for the 'DELETE' method. The 'RESTful Service Module' is 'Modulo_crud_roles', the 'Module Base Path' is '/modulo_crud_roles/', and the 'URI Template' is 'roles'. The 'Full URL' is 'https://apex.oracle.com/pls/apex/4cetickets/modulo_crud_roles/roles'. The 'Method' is 'DELETE' and the 'Source Type' is 'PL/SQL'. The 'Source' section contains the following PL/SQL code:

```
1 DECLARE
2     MESSEGE VARCHAR2(200);
3     ROLE_ID NUMBER := :role_id;
4
5 BEGIN
6     CRUD_ROLES.DELETE_ROLES(ROLE_ID, MESSEGE);
7     APEX_JSON.OPEN_OBJECT;
8     APEX_JSON.WRITE('FILAS MOD: ', MESSEGE);
9     APEX_JSON.CLOSE_OBJECT;
10 END;
```

Figura 51. Método DELETE. Modulo_Crud_Roles [Autoría propia]

The screenshot shows the 'ORDS Handler Definition' window for the 'GET' method. The 'RESTful Service Module' is 'Modulo_crud_roles', the 'Module Base Path' is '/modulo_crud_roles/', and the 'URI Template' is 'roles'. The 'Full URL' is 'https://apex.oracle.com/pls/apex/4cetickets/modulo_crud_roles/roles'. The 'Method' is 'GET' and the 'Source Type' is 'Collection Query'. The 'Format' is 'JSON' and the 'Pagination Size' is set to 10. The 'Source' section contains the following SQL query:

```
1 SELECT * FROM XX4CE_HD_ROLES_ALL ORDER BY ROLE_ID DESC
```

Figura 52. Método GET. Modulo_Crud_Roles [Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module: **Modulo_Crud_Roles**

Module Base Path: **/modulo_crud_roles/**

URI Template: **roles**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/modulo_crud_roles/roles

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1 DECLARE
2     MESSEGE VARCHAR2(200);
3     ROLE_ID NUMBER :=21;
4     ROLE_N VARCHAR2 (200) := 'Prueba Rol2';
5     UPDATEDBY NUMBER :=1;
6
7 BEGIN
8     CRUD_ROLES.STORE_ROLES(XX4CE_HD_ROLES_ALL_S.NEXTVAL, :ROLE_NAME, SYSDATE, :CREATED_BY, SYSDATE, :UPDATED_BY, MESSEGE);
9     APEX_JSON.OPEN_OBJECT;
10    APEX_JSON.WRITE('FILAS MOD: ',MESSEGE);
11    APEX_JSON.CLOSE_OBJECT;
12 END;

```

Figura 53. Método POST. Modulo_Crud_Roles[Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module: **Modulo_Crud_Roles**

Module Base Path: **/modulo_crud_roles/**

URI Template: **roles**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/modulo_crud_roles/roles

Method: **PUT**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1 DECLARE
2     MESSEGE VARCHAR2(200);
3     ROLE_ID NUMBER := :role_id;
4     ROLE_N VARCHAR2 (200) := :role_name;
5     UPDATEDBY NUMBER := :last_updated_by;
6
7 BEGIN
8     CRUD_ROLES.UPDATE_ROLES(ROLE_ID, ROLE_N, SYSDATE, UPDATEDBY, MESSEGE);
9     APEX_JSON.OPEN_OBJECT;
10    APEX_JSON.WRITE('FILAS MOD: ',MESSEGE);
11    APEX_JSON.CLOSE_OBJECT;
12 END;
13
14

```

Figura 54. Método PUT. Modulo_Crud_Roles[Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Users**

Module Base Path **/modulo_crud_users/**

URI Template **users**

Full URL https://apex.oracle.com/pls/apex/Acetickets/modulo_crud_users/users

Method **DELETE**

Source Type **PL/SQL**

Comments

Source

```

1 DECLARE
2     MESSEGE VARCHAR2(200);
3     USR_ID NUMBER := :user_id;
4
5 BEGIN
6
7     CRUD_USERS.DELETE_USERS(USR_ID, MESSEGE);
8     APEX_JSON.OPEN_OBJECT;
9     APEX_JSON.WRITE('FILAS MOD: ',MESSEGE);
10    APEX_JSON.CLOSE_OBJECT;
11 END;
```

Figura 55. Método DELETE. Modulo_Crud_Users[Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Users**

Module Base Path **/modulo_crud_users/**

URI Template **users**

Full URL https://apex.oracle.com/pls/apex/Acetickets/modulo_crud_users/users

Method **GET**

Source Type **Collection Query**

Format **JSON**

Pagination Size

Comments

Source

```

1 SELECT * FROM XX4CE_HD_USERS_ALL ORDER BY USER_ID DESC
```

Figura 56. Método GET. Modulo_Crud_Users[Autoría propia]

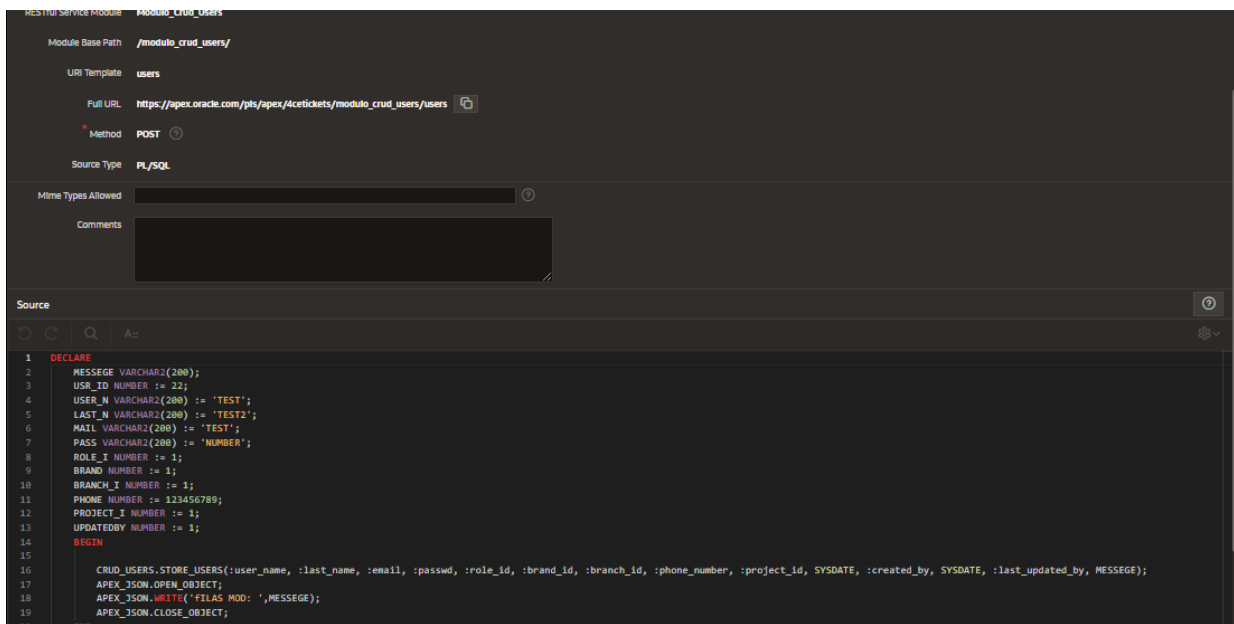


Figura 57. Método POST. Modulo_Crud_Users [Autoría propia]

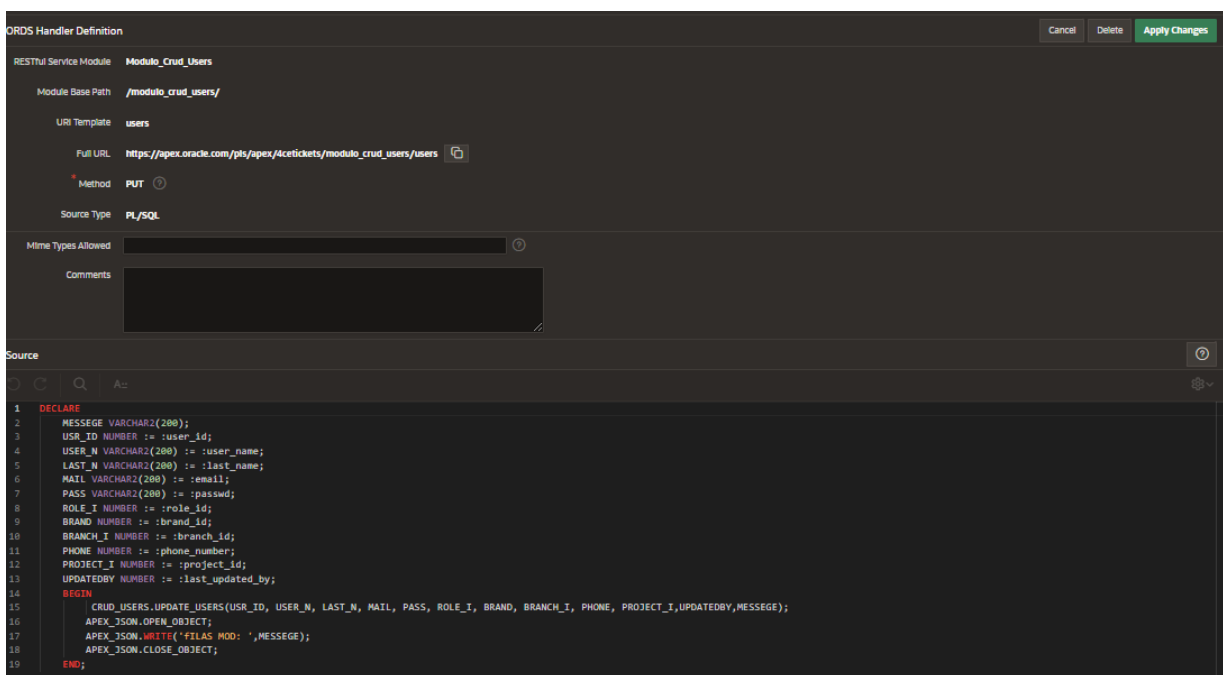


Figura 58. Método PUT. Modulo_Crud_Users [Autoría propia]

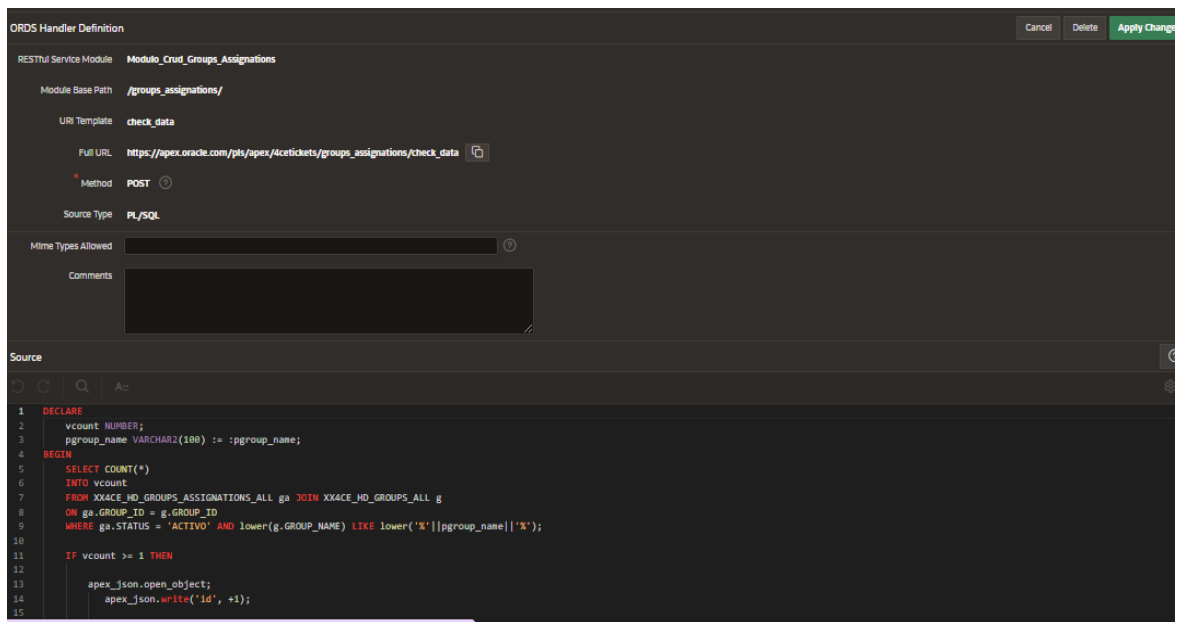


Figura 59. Método POST. Modulo_Crud_Groups_Assignations [Autoría propia]

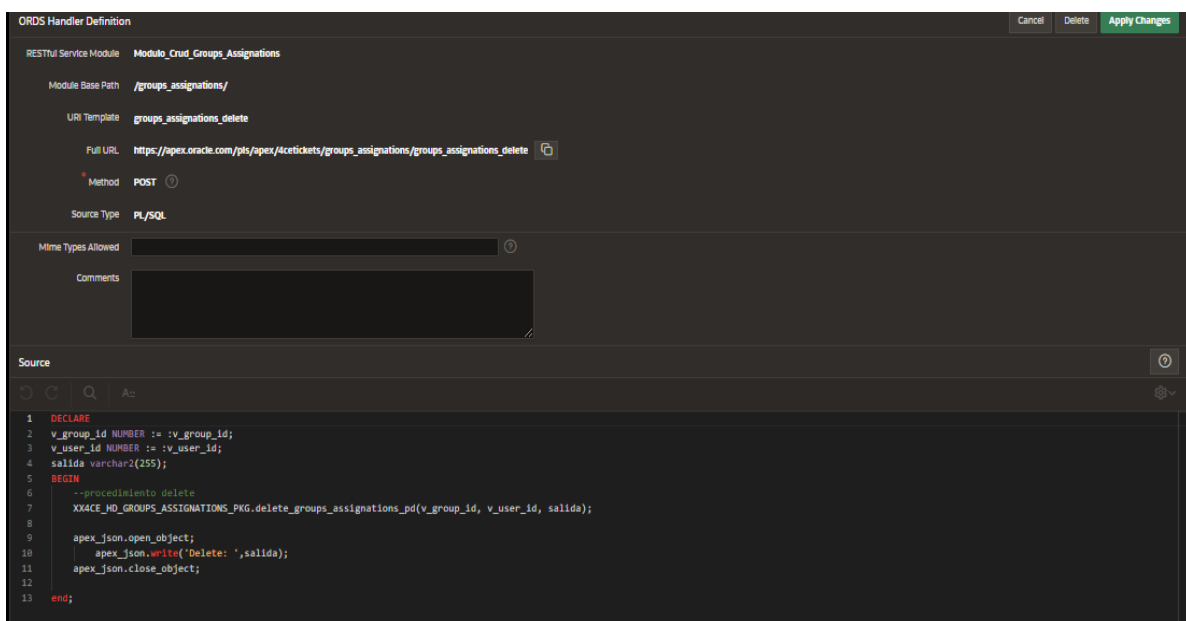


Figura 60. Método DELETE. Modulo_Crud_Groups_Assignations [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Groups_Assignations**

Module Base Path: **/groups_assignations/**

URI Template: **groups_assignations_get_data**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/groups_assignations/groups_assignations_get_data

Method: **GET**

Source Type: **Collection Query**

Format: **JSON**

Pagination Size:

Comments:

Source

```

1 SELECT ga.GROUP_ID, g.GROUP_NAME,
2       ga.USER_ID, u.USER_NAME || ' ' || u.LAST_NAME AS USER_NAME,
3       ga.STATUS,
4       ga.CREATED_DATE,
5       ga.CREATED_BY,
6       ga.LAST_UPDATE_DATE,
7       ga.LAST_UPDATED_BY
8 FROM XXACE_HO_GROUPS_ASSIGNATIONS_ALL ga JOIN XXACE_HO_GROUPS_ALL g
9 ON ga.GROUP_ID = g.GROUP_ID
10 JOIN XXACE_HO_USERS_ALL u
11 ON ga.USER_ID = u.USER_ID
12 WHERE ga.STATUS = 'ACTIVO'
13 ORDER BY ga.GROUP_ID

```

Figura 61. Método GET. Modulo_Crud_Groups_Assignations [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Groups_Assignations**

Module Base Path: **/groups_assignations/**

URI Template: **groups_assignations_insert**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/groups_assignations/groups_assignations_insert

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1 DECLARE
2   v_group_id      NUMBER := :v_group_id;
3   v_user_id       NUMBER := :v_user_id;
4   v_created_by    NUMBER := :v_created_by;
5   v_last_updated_by NUMBER := :v_last_updated_by;
6   salida          varchar2(255);
7 BEGIN
8   --procedimiento Insertar
9   XXACE_HO_GROUPS_ASSIGNATIONS_PKG.insert_groups_assignations_pd(v_group_id,v_user_id,v_created_by,v_last_updated_by,salida);
10
11   apex_json.open_object;
12   apex_json.write('Inserción: ',salida);
13   apex_json.close_object;
14
15 end;

```

Figura 62. Método INSERT. Modulo_Crud_Groups_Assignations [Autoría propia]

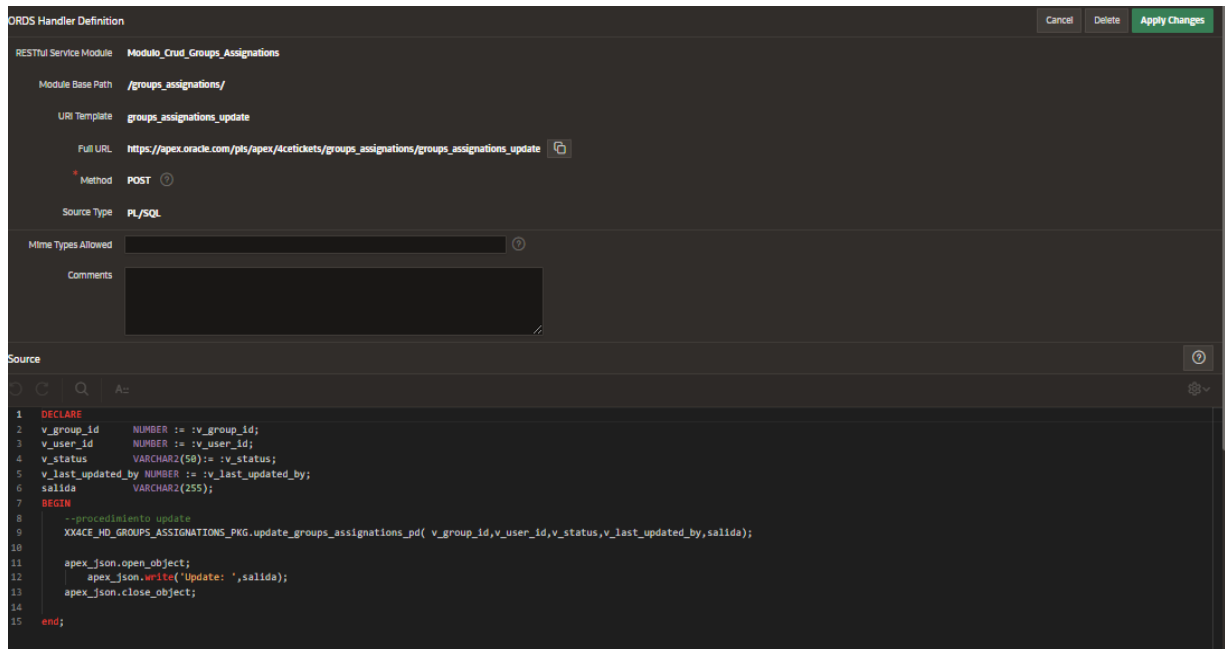


Figura 63. Método UPDATE. Modulo_Crud_Groups_Assignations [Autoria propia]

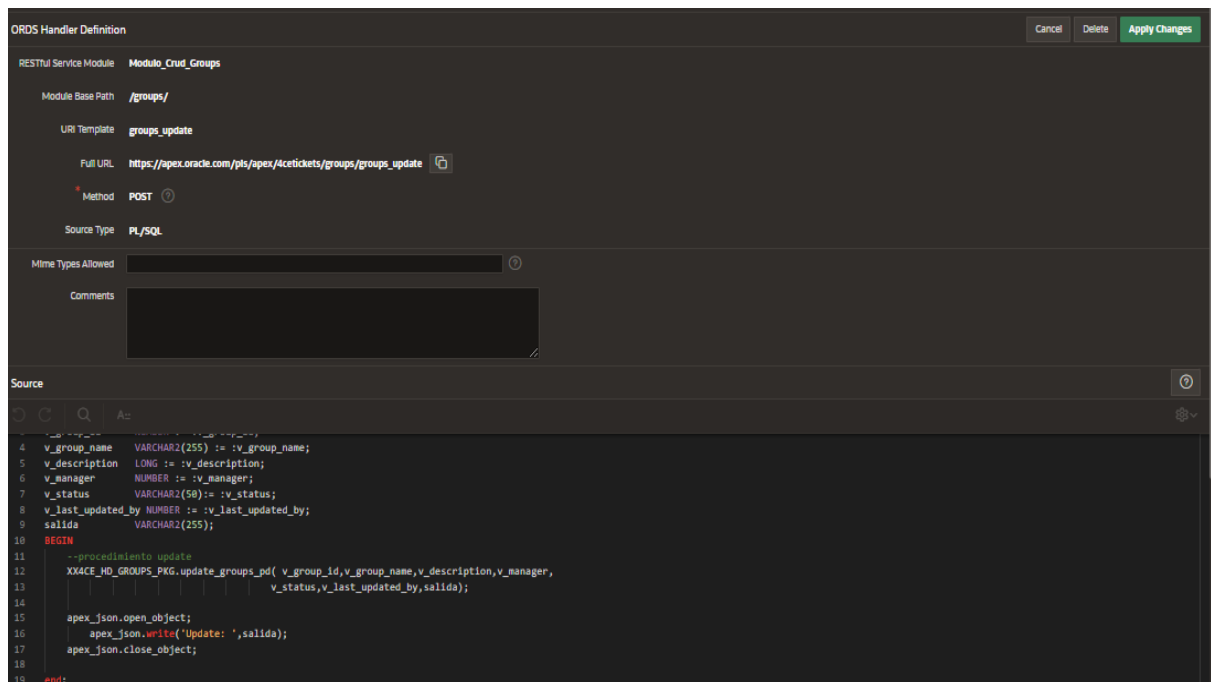


Figura 64. Método UPDATE. Modulo_Crud_Groups [Autoría propia]

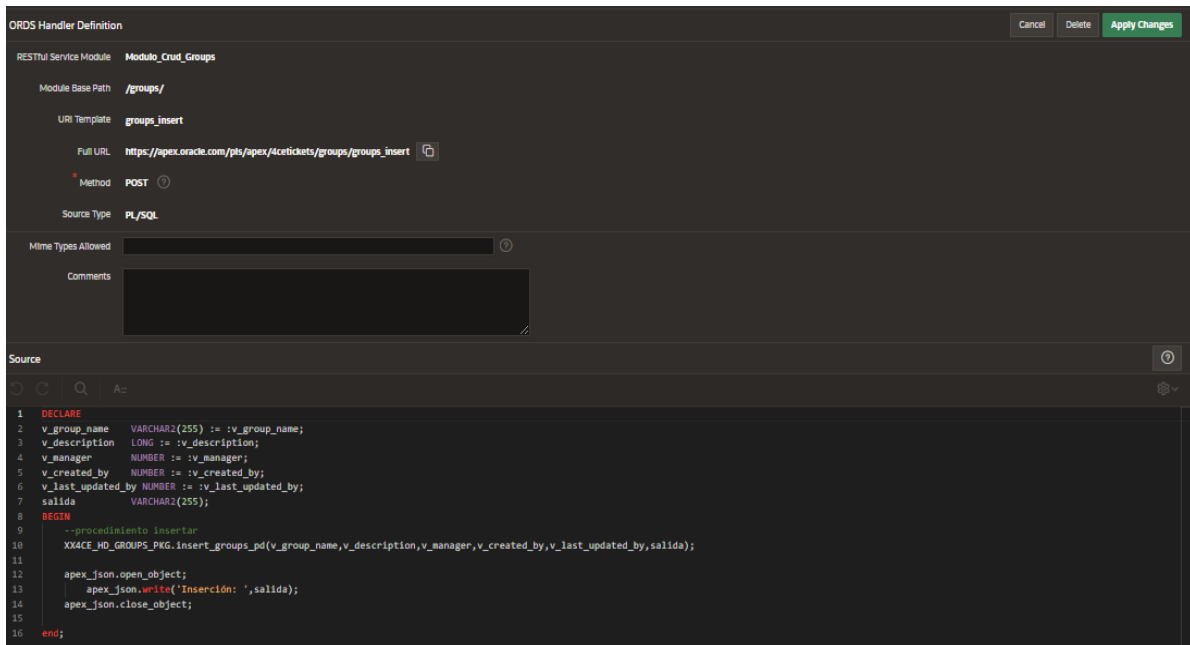


Figura 65. Método INSERT. Modulo_Crud_Groups [Autoría propia]

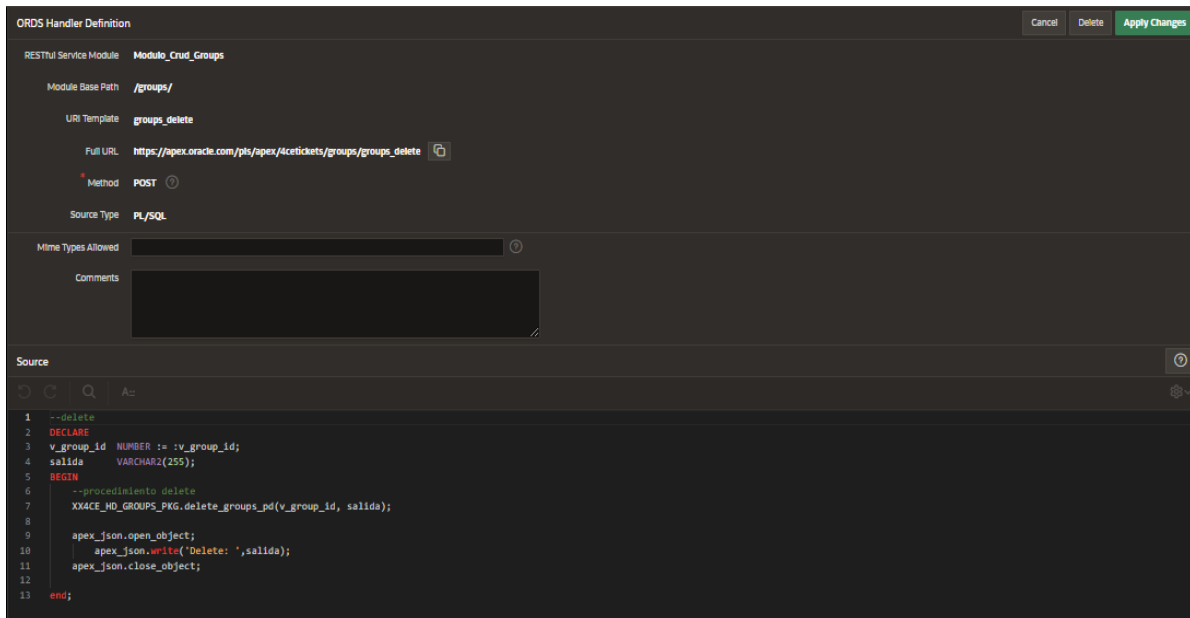


Figura 66. Método DELETE. Modulo_Crud_Groups [Autoría propia]

ORDS Handler Definition

Cancel

Delete

Apply Changes

RESTful Service Module

Module_Crud_Brands_Projects

Module Base Path

/mod_crud_brands_projects/

URI Template

plantilla_brands_crud

Full URL

https://apex.oracle.com/pls/apex/4cetickets/mod_crud_brands_projects/plantilla_brands_crud

Method

DELETE

Source Type

PL/SQL

Comments

Source

1

DECLARE

2

BRAND_ID NUMBER:= :brand_id;

3

MESSAGE VARCHAR2(256);

4

BEGIN

5

XX4ACE_HD_BRANDSCRUD_PKG.delete_value(BRAND_ID, MESSAGE);

6

apex_json.open_object;

7

apex_json.write('message', MESSAGE);

8

apex_json.close_object;

9

END;

ORDS Handler Definition

Cancel

Delete

Apply Changes

RESTful Service Module

Module_Crud_Brands_Projects

Module Base Path

/mod_crud_brands_projects/

URI Template

plantilla_brands_crud

Full URL

https://apex.oracle.com/pls/apex/fcetickets/mod_crud_brands_projects/plantilla_brands_crud

Method

POST

Source Type

PL/SQL

Mime Types Allowed

Comments

Source

As

```
4  descomentalo y pon mis mods. en comentario -- Omar
5
6  DECLARE
7    BRAND_NAME VARCHAR2(250):= :v_1_brand_name;
8    PHONE_NUMBER NUMBER:= :v_1_phone_number;
9    ADDRESS VARCHAR2(250):= :v_1_address;
10   CREATED_BY NUMBER:= :v_1_created_by;
11   MESSAGE VARCHAR2(250);
12
13 BEGIN
14   XXACE_H01_BRANDSCRUD_PKG.insert_value(BRAND_NAME, PHONE_NUMBER, ADDRESS, CREATED_BY, MESSAGE);
15   apex_json.open_object;
16   | apex_json.write('mensaje', MESSAGE);
17   apex_json.close_object;
18
19 */
20
```

DRDS Handler Definition

Cancel

Delete

Apply Changes

RESTful Service Module

Module_Crud_Brands_Projects

Module Base Path

/mod_crud_brands_projects/

URI Template

plantilla_brands_crud

Full URL

https://apex.oracle.com/pls/apex/fcetickets/mod_crud_brands_projects/plantilla_brands_crud

Method

PUT

Source Type

PL/SQL

Mime Types Allowed

Comments

Source

6

7

8

9

10

11

12

13

14

15

16

17

18

19

DECLARE

BRAND_ID NUMBER:= :v_1_brand_id;

BRAND_NAME VARCHAR2(250):= :v_1_brand_name;

PHONE_NUMBER NUMBER:= :v_1_phone_number;

ADDRESS VARCHAR2(250):= :v_1_address;

LAST_UPDATED_BY NUMBER:= :v_1_last_updated_by;

MESSAGE VARCHAR2(250);

BEGIN

XX4CE_HO_BRANDSCRUD_PKG.update_value(BRAND_ID, BRAND_NAME, PHONE_NUMBER, ADDRESS,

LAST_UPDATED_BY, MESSAGE);

apex_json.open_object;

apex_json.write('mensaje', MESSAGE);

apex_json.close_object;

END;

ORDS Handler Definition

Cancel

Delete

Apply Changes

RESTful Service Module

Module_Crud_Tickets

Module Base Path

/tickets/

URI Template

ticket

Full URL

https://apex.oracle.com/pls/apex/4cetickets/tickets/ticket

Method

DELETE

Source Type

PL/SQL

Comments

Source

1

DELETE FROM XX4CE_HD_TICKETS_ALL WHERE TICKET_ID = :ID

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Tickets**

Module Base Path: **/tickets/**

URI Template: **ticket**

Full URL: <https://apex.oracle.com/pls/apex/facetickets/tickets/ticket>

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1 DECLARE
2   MESSAGE VARCHAR2(200);
3   BEGIN
4     XMXACE_HD_TICKETS_ALL_PKG.STORE_TICKETS(:TITLE, :TTYPE, :TSUMMARY, :URL, :BRAND, :BBRANCH, :TTAGS, :USR_ID, :PRJ_ID, :TPRIORITY, :CREATED_BY, :UPDATED_BY, MESSAGE);
5     APEX_JSON.OPEN_OBJECT;
6     APEX_JSON.WRITE('files mod', MESSAGE);
7     APEX_JSON.CLOSE_OBJECT;
8   END;
```

Figura 71. Método POST. Modulo_Crud_Tickets [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Tickets**

Module Base Path: **/tickets/**

URI Template: **ticket**

Full URL: <https://apex.oracle.com/pls/apex/facetickets/tickets/ticket>

Method: **PUT**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1 DECLARE
2   MESSAGE VARCHAR2(200);
3   TICKETID NUMBER := :TICKETID;
4   TITLE VARCHAR2(200) := :TITLE;
5   TTYPE NUMBER := :TTYPE;
6   TSUMMARY VARCHAR2(200) := :TSUMMARY;
7   URL VARCHAR2(200) := :URL;
8   TTAGS VARCHAR2(200) := :TTAGS;
9   UPDATED_BY NUMBER := :UPDATED_BY;
10  BEGIN
11    XMXACE_HD_TICKETS_ALL_PKG.UPDATE_TICKETS(TICKETID, TITLE, TTYPE, TSUMMARY, URL, TTAGS, UPDATED_BY, MESSAGE);
12    APEX_JSON.OPEN_OBJECT;
13    APEX_JSON.WRITE('files mod', MESSAGE);
14    APEX_JSON.CLOSE_OBJECT;
15  END;
```

Figura 72. Método UPDATE. Modulo_Crud_Tickets [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Types**

Module Base Path: **/Types/**

URI Template: **type**

Full URL: <https://apex.oracle.com/pls/apex/facetickets/Types/type>

Method: **GET**

Source Type: **Collection Query**

Format: **JSON**

Pagination Size:

Comments:

Source

```
1 SELECT * FROM XXACE_ID_TYPES_ALL
```

Figura 73. Método GET. Modulo_Crud_Types [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Types**

Module Base Path: **/Types/**

URI Template: **type**

Full URL: <https://apex.oracle.com/pls/apex/facetickets/Types/type>

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```
1 declare
2   TYPE_ID NUMBER := :type_id_value;
3   TYPE_NAME VARCHAR2(250) := :type_name_value;
4   CREATED_BY NUMBER := :type_created_by_value;
5   LAST_UPDATED_BY NUMBER := :type_last_updated_by_value;
6   CREATED_AT timestamp := sysdate;
7   UPDATED_AT timestamp := sysdate;
8   MESSAGE VARCHAR2(200);
9   BEGIN
10  XXACE_T_TYPES.Insertion_valores(TYPE_ID, TYPE_NAME, CREATED_BY, LAST_UPDATED_BY, CREATED_AT, UPDATED_AT, MESSAGE);
11
12
13  apex_json.open_object;
14  apex_json.write('mensaje', MESSAGE);
15  apex_json.close_object;
16 END;
```

Figura 74. Método POST. Modulo_Crud_Types [Autoría propia]

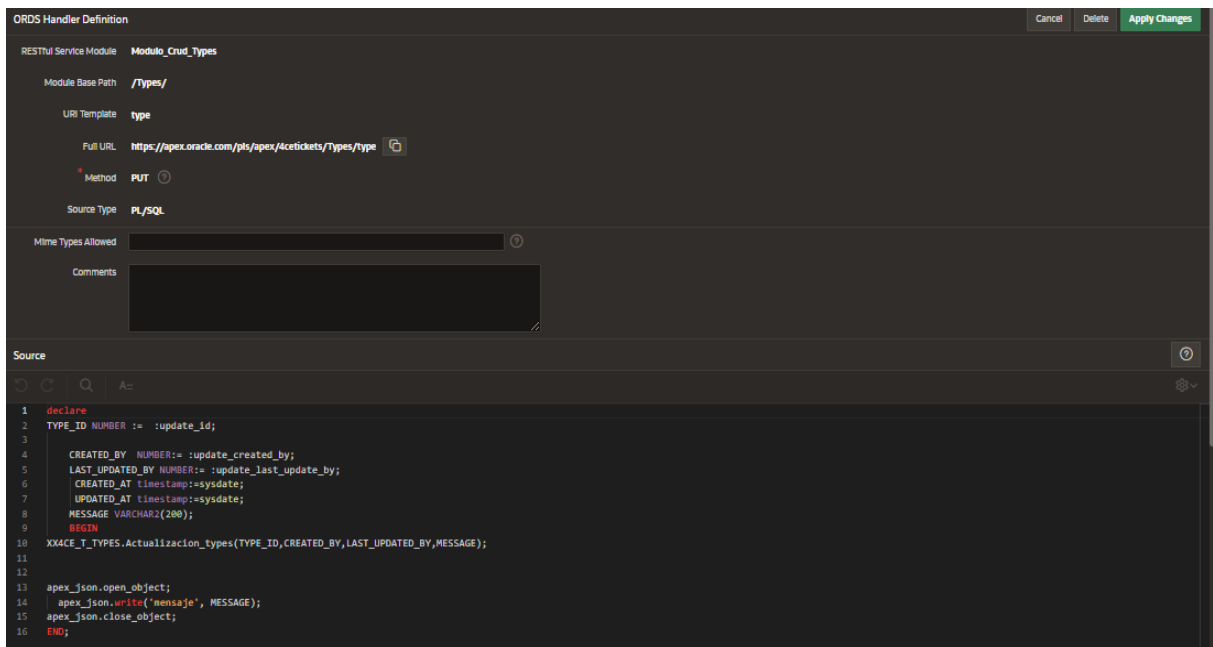


Figura 75. Método PUT. Modulo_Crud_Types [Autoría propia]

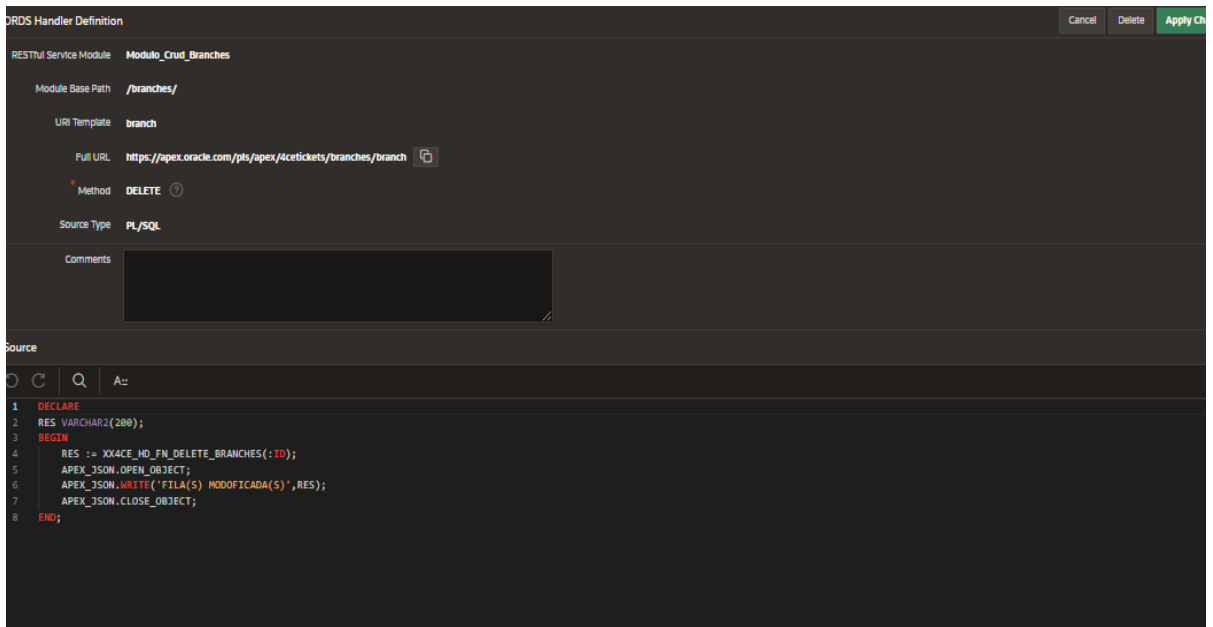


Figura 76. Método DELETE. Modulo_Crud_Branches [Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Branches**

Module Base Path **/branches/**

URI Template **branch**

Full URL <https://apex.oracle.com/pls/apex/4cetickets/branches/branch>

Method **POST**

Source Type **PL/SQL**

Mime Types Allowed

Comments

Source

```

1 DECLARE
2   RES VARCHAR2(200);
3   bname varchar2(200) := :branch_name;
4   bphone varchar2(200) := :branch_phone;
5   created number := :created_by;
6   updated number := :updated_by;
7   bid number := :brand_id;
8 BEGIN
9   RES := XX4CE_HD_FM_STORE_BRANCHES(bname , bphone, created, SYSDATE, updated, SYSDATE, bid);
10  APEX_JSON.OPEN_OBJECT;
11  APEX_JSON.WRITE('FILAS') MODIFICADA(S), RES);
12  APEX_JSON.CLOSE_OBJECT;
13 END;
```

Figura 77. Método POST. Modulo_Crud_Branches [Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Branches**

Module Base Path **/branches/**

URI Template **branch**

Full URL <https://apex.oracle.com/pls/apex/4cetickets/branches/branch>

Method **PUT**

Source Type **PL/SQL**

Mime Types Allowed

Comments

Source

```

1 DECLARE
2   RES VARCHAR2(200);
3   IDs NUMBER := :branch_id;
4   NAMES varchar2(200) := :branch_name;
5   PHONE NUMBER := :branch_phone;
6   UPDATED_BY number := :updated_by;
7   BRAND NUMBER := :brand_id;
8 BEGIN
9   RES := XX4CE_HD_FM_UPDATE_BRANCHES(IDs, NAMES, PHONE, UPDATED_BY, SYSDATE, BRAND);
10  APEX_JSON.OPEN_OBJECT;
11  APEX_JSON.WRITE('FILAS') MODIFICADA(S), RES);
12  APEX_JSON.CLOSE_OBJECT;
13 END;
```

Figura 78. Método PUT. Modulo_Crud_Branches [Autoría propia]

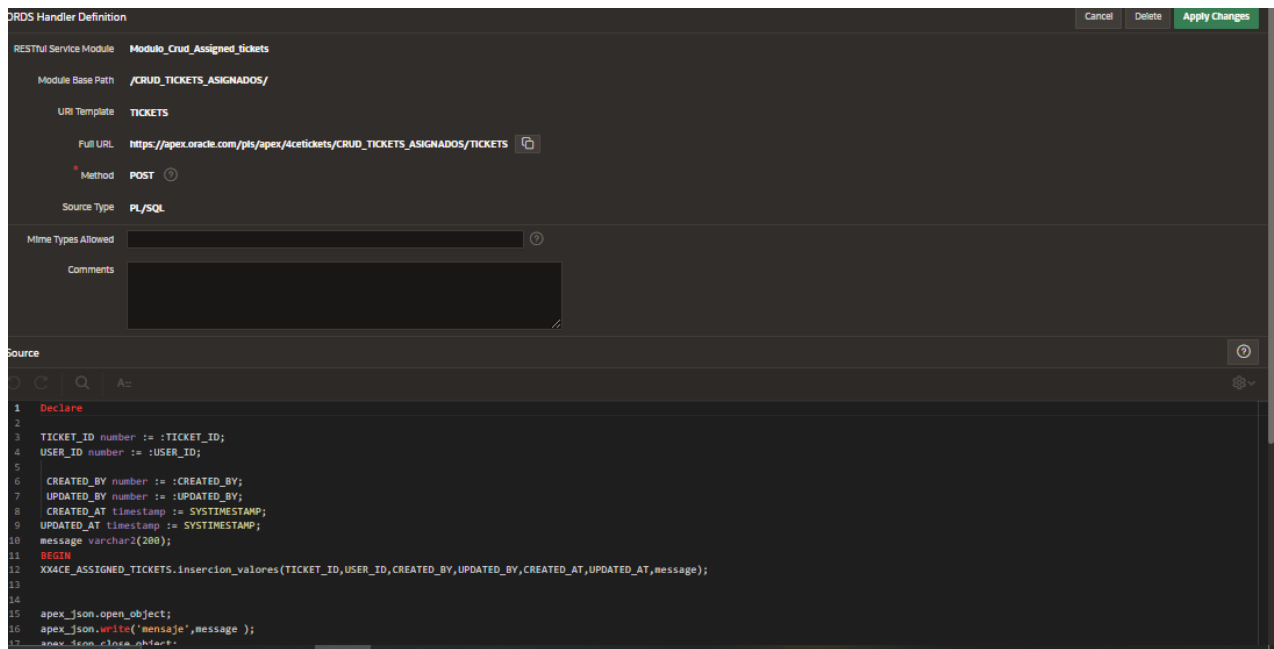


Figura 79. Método POST. Modulo_Crud_Assigned_Tickets [Autoría propia]

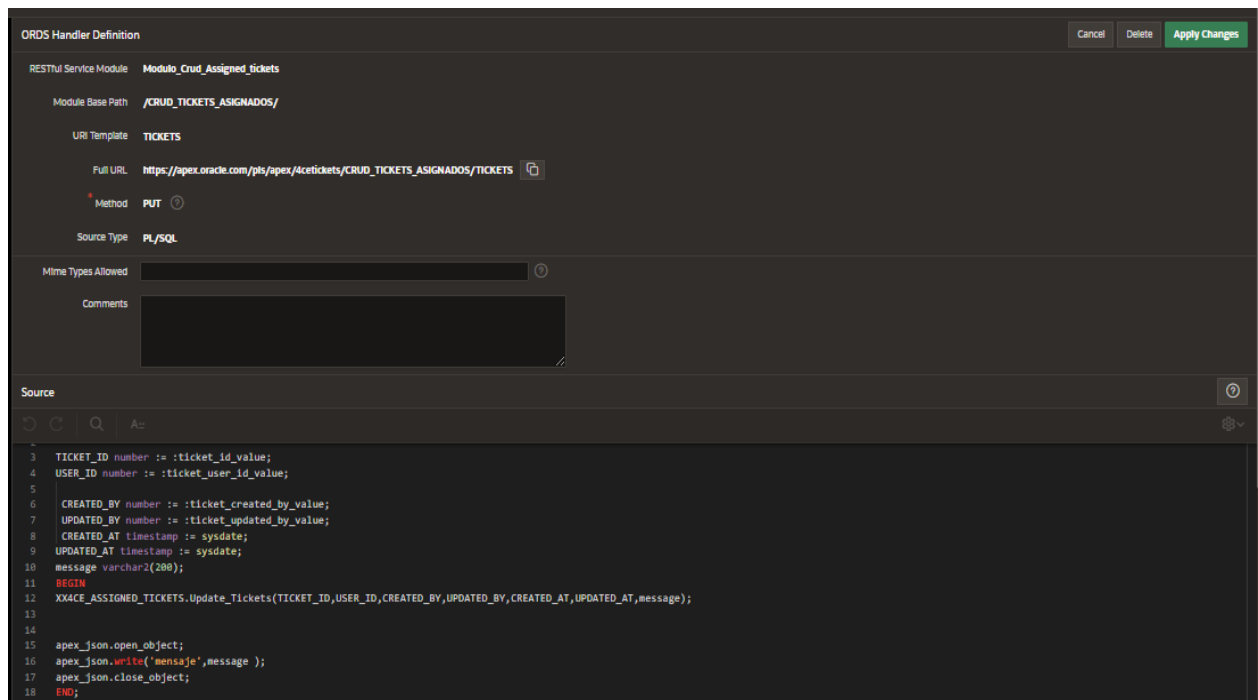


Figura 80. Método PUT. Modulo_Crud_Assigned_Tickets [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Assigned_tickets**

Module Base Path: **/CRUD_TICKETS_ASSIGNADOS/**

URI Template: **TICKETS**

Full URL: https://apex.oracle.com/pls/apex/f4cetickets/CRUD_TICKETS_ASSIGNADOS/TICKETS

Method: **GET**

Source Type: **Collection Query**

Format: **JSON**

Pagination Size:

Comments:

Source

```
1 Select * from XX4CE_HD_ASSIGNED_TICKETS_ALL
```

Figura 81. Método GET. Modulo_Crud_Assigned_Tickets [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Releases**

Module Base Path: **/Modulo_Crud_Releases/**

URI Template: **Plantilla_Releases_UPDATE**

Full URL: https://apex.oracle.com/pls/apex/f4cetickets/Modulo_Crud_Releases/Plantilla_Releases_UPDATE

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```
1 DECLARE
2  RELEASE_IDnuev NUMBER := :E_RELEASE_ID;
3  RELEASE_TITLEnuev VARCHAR2(200) := :E_RELEASE_TITLE;
4  RELEASE_SUMMARYnuev VARCHAR2 (200) := :E_RELEASE_SUMMARY;
5  LAST_UPDATE_DATEnuev TIMESTAMP := SYSDATE;
6  LAST_UPDATED_BYnuev NUMBER := :E_LAST_UPDATED_BY;
7  COMMENT_IDnuev NUMBER := :E_COMMENT_ID;
8  DIRECTED_TOnuev VARCHAR2(200) := :E_DIRECTED_TO;
9  MESSAGE VARCHAR2(200);
10 BEGIN
11
12  UPDATE XX4CE_HD_RELEASES_ALL
13  SET
14  RELEASE_TITLE = RELEASE_TITLEnuev,
15  RELEASE_SUMMARY = RELEASE_SUMMARYnuev,
16  LAST_UPDATE_DATE = LAST_UPDATE_DATEnuev,
17  LAST_UPDATED_BY = LAST_UPDATED_BYnuev;
```

Figura 82. Método POST. Modulo_Crud_Relases [Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Releases**

Module Base Path **/Modulo_Crud_Releases/**

URI Template **Plantilla_Releases_DELETE**

Full URL https://apex.oracle.com/pls/apex/f4cetickets/Modulo_Crud_Releases/Plantilla_Releases_DELETE

Method **PUT**

Source Type **PL/SQL**

Mime Types Allowed

Comments

Source

```

1 DECLARE
2   RELEASE_ID NUMBER := :1_RELEASE_ID;
3   MESSAGE VARCHAR2(200);
4 BEGIN
5   XXACE_HD_CRUDRELEASES_PKG.delete_value(RELEASE_ID,MESSAGE);
6   apex_json.open_object;
7   | apex_json.write('mensaje', MESSAGE);
8   apex_json.close_object;
9 END;
```

Figura 83. Método PUT. Modulo_Crud_Relases [Autoría propia]

ORDS Handler Definition Cancel Delete Apply Changes

RESTful Service Module **Modulo_Crud_Releases**

Module Base Path **/Modulo_Crud_Releases/**

URI Template **Plantilla_Releases_CREATE**

Full URL https://apex.oracle.com/pls/apex/f4cetickets/Modulo_Crud_Releases/Plantilla_Releases_CREATE

Method **GET**

Source Type **Collection Query**

Format **JSON**

Pagination Size

Comments

Source

```

1 SELECT * FROM XXACE_HD_RELEASES_ALL WHERE DIRECTED_TO = :ENTITY ORDER BY RELEASE_ID DESC
```

Figura 84. Método GET Modulo_Crud_Relases [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Ticket_Comments**

Module Base Path: **/Modulo_Crud_Ticket_Comments/**

URI Template: **Plantilla_Ticket_Comments_UPDATE**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/Modulo_Crud_Ticket_Comments/Plantilla_Ticket_Comments_UPDATE

Method: **POST**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1  DECLARE
2  COMMENT_IDnuev NUMBER := :E_COMMENT_ID;
3  COMMENTSnuev VARCHAR2(200) := :E_COMMENTS;
4  LAST_UPDATE_DATEnuev TIMESTAMP := SYSDATE;
5  LAST_UPDATED_BYnuev NUMBER := :E_LAST_UPDATED_BY;
6  TICKET_IDnuev NUMBER := :E_TICKET_ID;
7
8  BEGIN
9
10 UPDATE XX4CE_HD_TICKET_COMMENTS_ALL
11 SET
12 COMMENTS=COMMENTSnuev,
13 LAST_UPDATE_DATE = LAST_UPDATE_DATEnuev,
14 LAST_UPDATED_BY =LAST_UPDATED_BYnuev,
15 TICKET_ID=TICKET_IDnuev
16 WHERE COMMENT_ID = COMMENT_IDnuev;
17

```

Figura 85. Método POST. Modulo_Crud_Ticket_Comments [Autoría propia]

ORDS Handler Definition

RESTful Service Module: **Modulo_Crud_Ticket_Comments**

Module Base Path: **/Modulo_Crud_Ticket_Comments/**

URI Template: **Plantilla_Ticket_Comments_DELETE**

Full URL: https://apex.oracle.com/pls/apex/4cetickets/Modulo_Crud_Ticket_Comments/Plantilla_Ticket_Comments_DELETE

Method: **PUT**

Source Type: **PL/SQL**

Mime Types Allowed:

Comments:

Source

```

1  DECLARE
2  COMMENT_ID VARCHAR2(200) := :I_COMMENT_ID;
3  MESSAGE VARCHAR2(200);
4
5  BEGIN
6  XX4CE_HD_CRUDTICKETCOMMENTS_PKG.delete_value(COMMENT_ID,MESSAGE);
7  apex_json.open_object;
8  | apex_json.write('mensaje', MESSAGE);
9  apex_json.close_object;
10 END;

```

Figura 86. Método PUT. Modulo_Crud_Ticket_Comments [Autoría propia]

4.1.1.1 Servicios en Visual Builder

Se podrá visualizar la creación de los servicios en la plataforma de visual builder, los cuales se recabaron de las tablas creadas en Apex, cargándolos finalmente para su visualización y ejecución.

Welcome		login	main-login	Insertar x
Overview	Servers	Endpoints	Headers	Source
Filter Endpoints		<button>+ Endpoint</button>		
/roles		≡		
GET	Get Many	getRoles	≡	
POST	Create	postRoles	≡	
PUT	Update	putRoles	≡	
DELETE	Delete	deleteRoles	≡	

Figura 87. Servicio Insertar [Autoría propia]

Welcome		login	main-login	Insertar	Service_Login x
Overview	Servers	Endpoints	Headers	Source	
Filter Endpoints		<button>+ Endpoint</button>			
/prueba		≡			
PUT	Get Many	putPrueba	≡		
/registro		≡			
POST	Create	store_Registro	≡		

Figura 88. Servicio Login [Autoría propia]

Welcome		login	main-login	Service_registro x
Overview	Servers	Endpoints	Headers	Source
Filter Endpoints		<button>+ Endpoint</button>		
/user		≡		
PUT	Create	putUser	≡	

Figura 89. Servicio Registro [Autoría propia]

Welcome

login

main-login

Service_registro

USERS x

Overview

Servers

Endpoints

Headers

Source

Filter Endpoints

Endpoint

/users

GET

Get Many

getUsers

POST

Create

postUsers

PUT

Update

putUsers

DELETE

Delete

deleteUsers

Figura 90. Servicio Users [Autoría propia]

Overview	Servers	Endpoints	Headers	Source
GET	Get Many		getGroups_get_data	
/groups_insert				
POST	Get Many		postGroups_insert	
/groups_update				
POST	Get Many		postGroups_update	
/groups_delete				
POST	Get Many		postGroups_delete	
/check_data				
POST	Get Many		postCheck_data	
/groups_get_data_name				
GET	Get Many		getGroups_get_data_name	

Figura 91. Servicio Groups [Autoría propia]

Welcome

service_groups_assignations
x

Overview

Servers

Endpoints

Headers

Source

GET	Get Many	getGroups_assignations_get_data	
		/groups_assignations_insert	
POST	Get Many	postGroups_assignations_insert2	
		/groups_assignations_update	
POST	Get Many	postGroups_assignations_update	
		/groups_assignations_delete	
POST	Get Many	postGroups_assignations_delete	
		/check_data	
POST	Get Many	postCheck_data	
		/groups_assignations_get_data_name	
GET	Get Many	getGroups_assignations_get_data_name	

Figura 92. Servicio Groups-Assignations [Autoría propia]

Welcome

Select_Brands x

Overview

Servers

Endpoints

Headers

Source

Q

Filter Endpoints

+

Endpoint

/plantilla_brands_crud

GET

Get Many

getPlantilla_brands_crud

POST

Get Many

InsertarMarcaServConn

PUT

Get Many

ServConnUpdateBrand

/plantilla_brands_crud_delete

POST

Get Many

ServConnDeleteBrand

PUT

Get Many

ServConnSearchBrandIfExistsID

/plantilla_brands_search_id

GET

Get Many

ServConnSearchBrandBYId

/plantilla_brands_search_nombre

Figura 93. Servicio Select_Brands [Autoría propia]

Welcome	ServConnProjects x	
Overview	Servers	Endpoints Headers Source
Filter Endpoints + Endpoint		
/plantilla_projects_crud		≡
GET	Get Many	ServConnSelectProjects ≡
POST	Get Many	ServConnInsertProject ≡
PUT	Get Many	ServConnUpdateProject ≡
/plantilla_projects_crud_delete		≡
GET	Get Many	ServConnSelectBrandsName ≡
POST	Get Many	ServConnDeleteProject ≡
/plantilla_projects_sel		≡
GET	Get Many	ServConnSelectProjects2 ≡
/plantilla_project_search		≡

Figura 94. Servicio ServConnProjects [Autoría propia]

Welcome	AssignedTickets x	
Overview	Servers	Endpoints Headers Source
Filter Endpoints		
/ByUserID		≡
GET	Get Many	getByUserID ≡
/STATISTICS		≡
GET	Get Many	getSTATISTICS ≡
/AssignedTickets		≡
GET	Get Many	getAssignedTickets ≡
/TICKETS		≡
POST	Get Many	postAssignedTickets ≡

Figura 95. Servicio AssignetTickets [Autoría propia]

Welcome			
Branches x			
Overview Servers Endpoints Headers Source			
Filter Endpoints			
/branch			≡
GET	Get Many	getBranch	≡
POST	Create	postBranches	≡
PUT	Get Many	putBranch	≡
DELETE	Delete	deleteBranch	≡
/ByBrand			≡
GET	Get Many	getByBrand	≡

Figura 96. Servicio Branches [Autoría propia]

Welcome			
Brands x			
Overview Servers Endpoints Headers Source			
Filter Endpoints			
/plantilla_brands_crud			≡
GET	Get Many	getPlantilla_brands_crud	≡
POST	Create	postBrands	≡
PUT	Update	putBrands	≡
DELETE	Delete	deleteBrands	≡

Figura 97. Servicio Brands [Autoría propia]

Welcome			
Comments x			
Overview Servers Endpoints Headers Source			
Filter Endpoints			
/Plantilla_Ticket_Comments_CREATE			≡
GET	Get Many	getPlantilla_Ticket_Comments_CREATE	≡
POST	Get Many	postComments	≡

Figura 98. Servicio Comments [Autoría propia]

Welcome	getStatus x	
Overview	Servers	Endpoints
Filter Endpoints		
/stat		
GET	Get Many	getStat

Figura 99. Servicio GetStatus [Autoría propia]

Welcome	plsApex	Projects x
Overview	Servers	Endpoints
Filter Endpoints		
/plantilla_projects_crud		
GET	Get Many	getPlantilla_projects_crud
POST	Get Many	postPlantilla_projects_crud
PUT	Update	putProjects
DELETE	Delete	deleteProject
/ProjectsTickets		
GET	Get Many	getProjectTickets
/IncidenceProjects		
GET	Get Many	getIncidencePerProject

Figura 100. Servicio Projects [Autoría propia]

Welcome	plsApex	Releases x
Overview	Servers	Endpoints
Filter Endpoints		
/Plantilla_Releases_CREATE		
GET	Get Many	getPlantilla_Releases_CREATE

Figura 101. Servicio Relases [Autoría propia]

Welcome	plsApex	Roles	TicketPriority x	
Overview	Servers	Endpoints	Headers	Source
Filter Endpoints				
/prio				
GET	Get Many		getPrio	

Figura 102. Servicio TicketPriority [Autoría propia]

Welcome

Tickets x

Overview

Servers

Endpoints

Headers

Source

GET

Get ManygetTicket

POST

CreatepostTicket

PUT

Get ManyputTicket

/ticketsbrands

GET

Get ManygetTicketsbrands

/ByBranch

GET

Get ManygetByBranch

/error

GET

Get ManygetError

/ByTitle

GET

Get ManygetTicketByTitle

Figura 103. Servicio Tickets [Autoría propia]

WelcomeTypes x

OverviewServersEndpointsHeadersSource

Filter Endpoints

/type

GETGet Many

getType

Figura 104. Servicio Types [Autoría propia]

4.1.1.2 Variables

Continuamos con la creación de variables que tenemos en cada una de las tablas utilizadas con anterioridad, eso con la finalidad de mapear de una forma correcta cada acción.

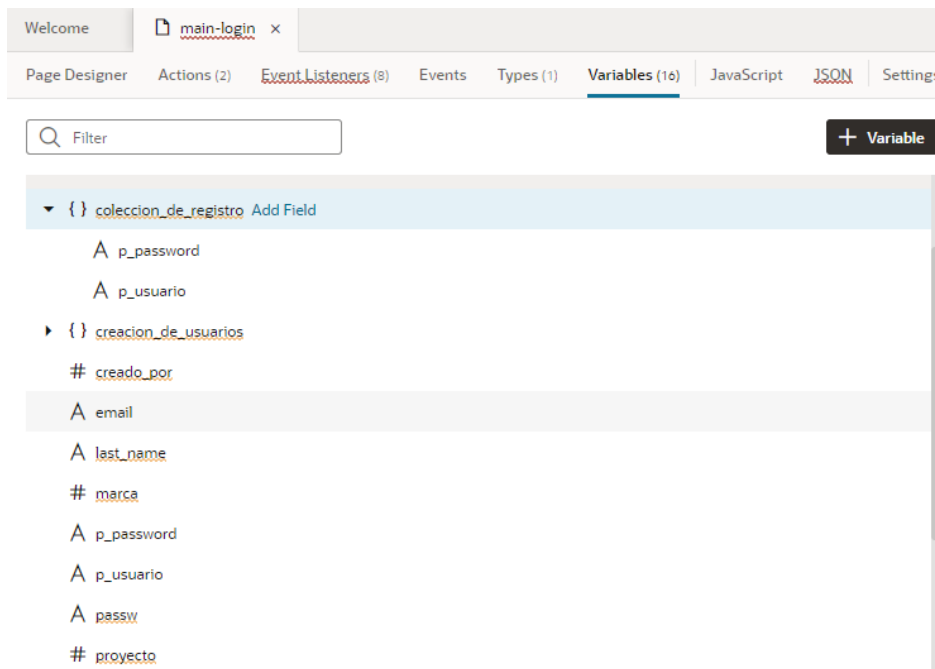


Figura 105. Variables Main-Login [Autpría propia]

The image shows a login form titled 'asignación de tickets'. The form is divided into two main sections. The first section, 'Usuario', contains a text input field with the placeholder text 'Ingresa tu usuario'. The second section, 'Contraseña', contains a text input field with the placeholder text 'Ingresa tu contraseña'. Below these two sections is a button labeled 'Iniciar Sesión'.

Figura 106. Pantalla 1 [Autoría propia]

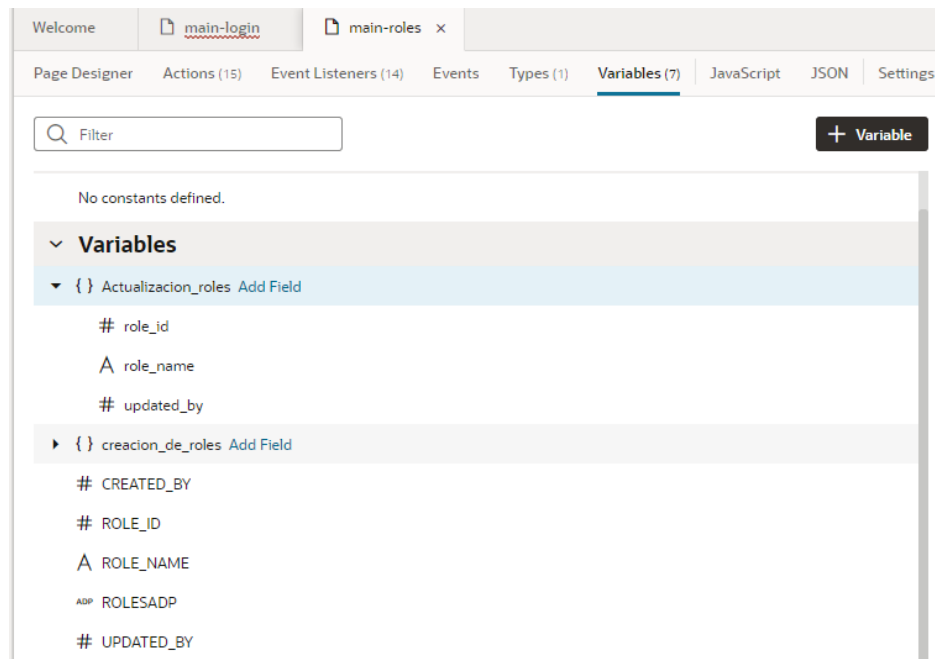


Figura 107. Variables Main-Roles [Autoría propia]

GRG Sistema de información para asignación de tickets

CREAR REGISTRO

main roles

role_id	role_name	created_by	creation_date	last_update_date	last_updated_by	DETALLES
161	PROJECT MANAGER	141	2022-11-29T16:03:01Z	2022-11-29T16:03:01Z	141	  
121	INFRASTRUCTURE	1	2022-11-14T22:57:07Z	2022-11-16T17:40:02Z	1	  
103	DEVELOPMENT	2	2022-11-09T22:28:58Z	2022-11-16T17:42:29Z	2	  
5	HUMAN RESOURCES	1	2022-10-07T22:56:22Z	2022-11-16T17:42:59Z	2	  
4	SUPPORT ASSISTANT	1	2022-10-07T22:55:50Z	2022-10-07T22:55:50Z	1	  

Figura 108. Pantalla 2 [Autoría propia]

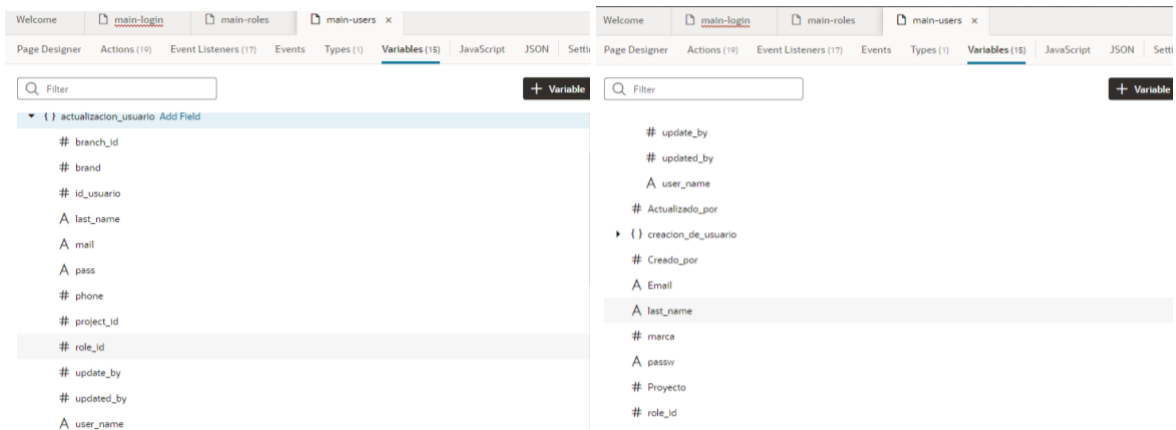


Figura 109. Variables Main-Users [Autoría propia]

crear registro

user_name ↕	passwd ↕	creation_date ↕	ACCIONES ↕
Jet	g?t*#th?g%+d2.0	2022-11-11T15:42:03Z	  
Alexander	@!%v?l&1995	2022-11-11T15:21:00Z	  
hugo	b?!t%+z	2022-11-09T22:16:57Z	  
Edward	@!&t?%/&?%th*%#?	2022-11-07T21:27:37Z	  
Sam	g?t*#th?g%+d	2022-11-07T21:15:53Z	  
Rodrigo	%*d%+g*	2022-11-01T15:05:14Z	  
Gerardo	147	2022-11-01T15:03:10Z	  

Figura 110. Pantalla 3 [Autoría propia]

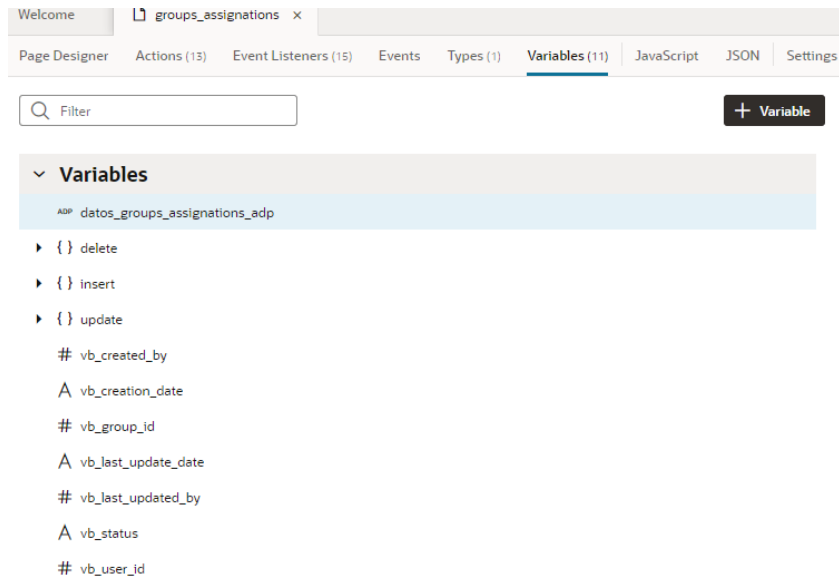


Figura 111. Variables Groups_Assignations [Autoría propia]

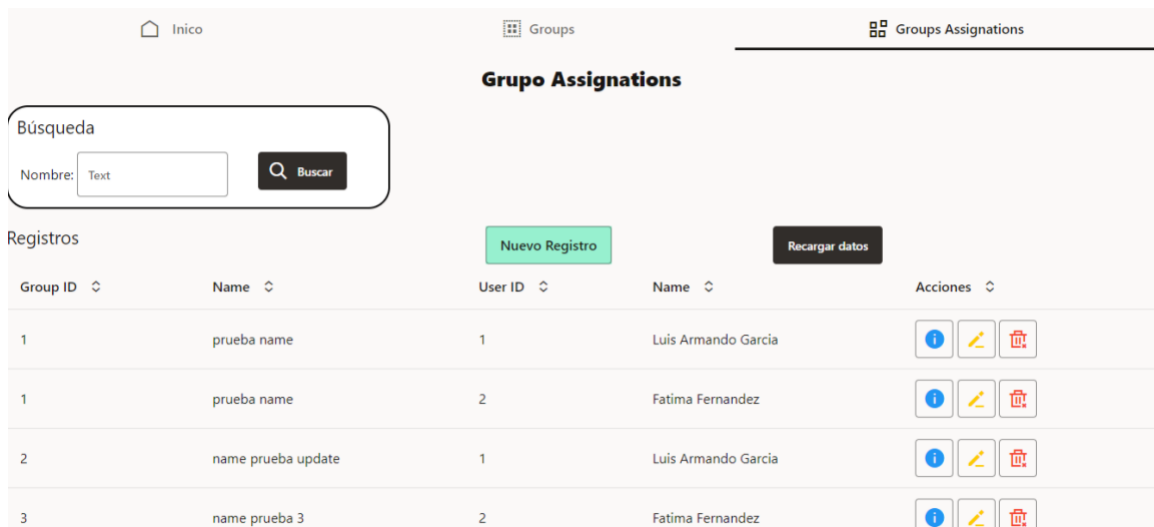


Figura 112. Pantalla 4 [Autoría propia]

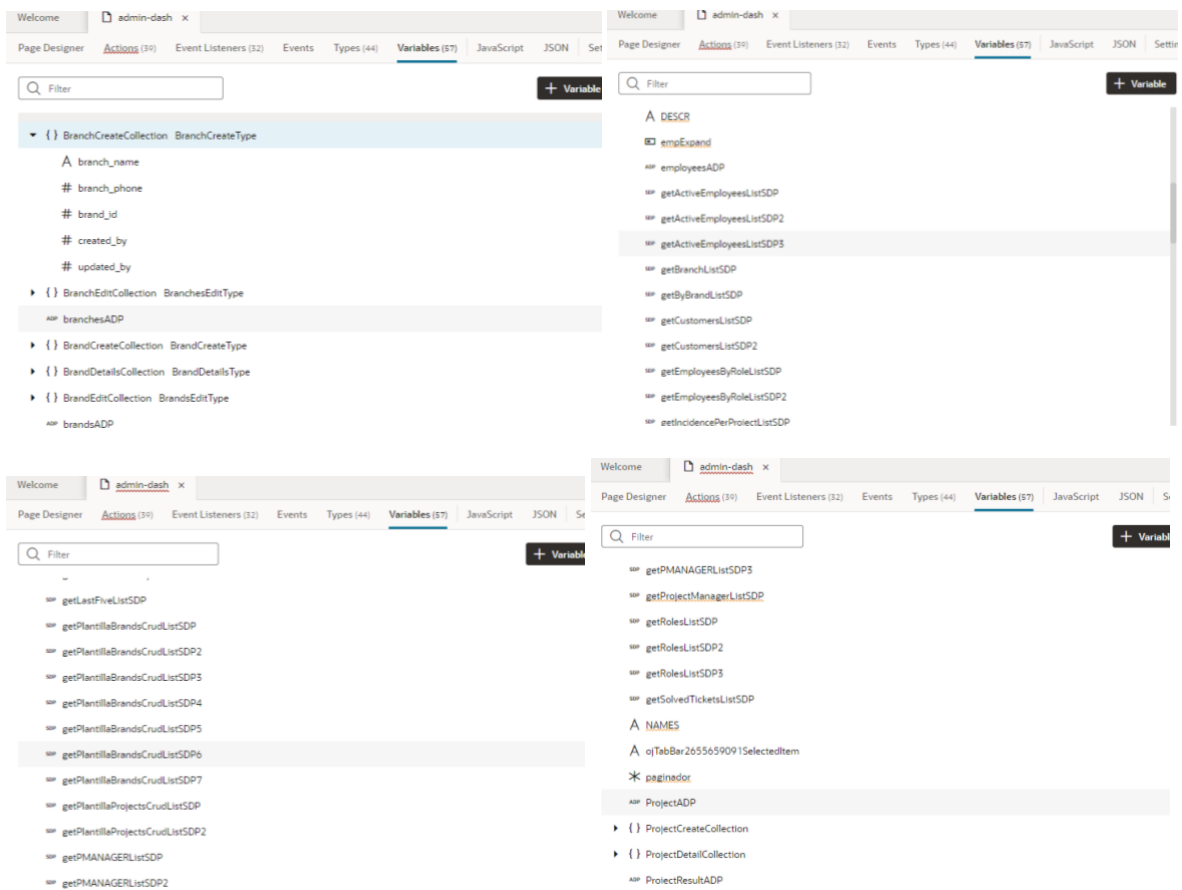


Figura 113. Variables AdminDash [Autoría propia]



Figura 114. Pantalla 5 [Autoría propia]

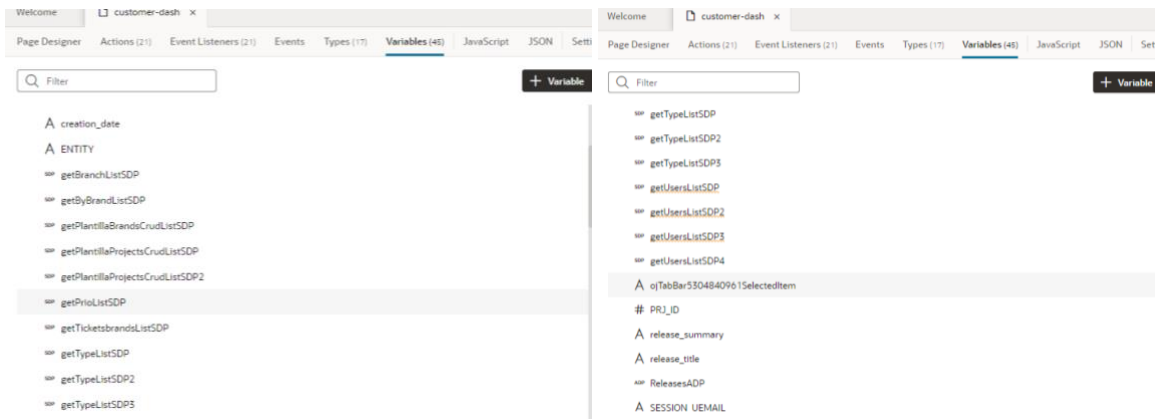


Figura 115. Variables Customer-dash [Autoría propia]



Figura 116. Pantalla 6 [Autoría propia]

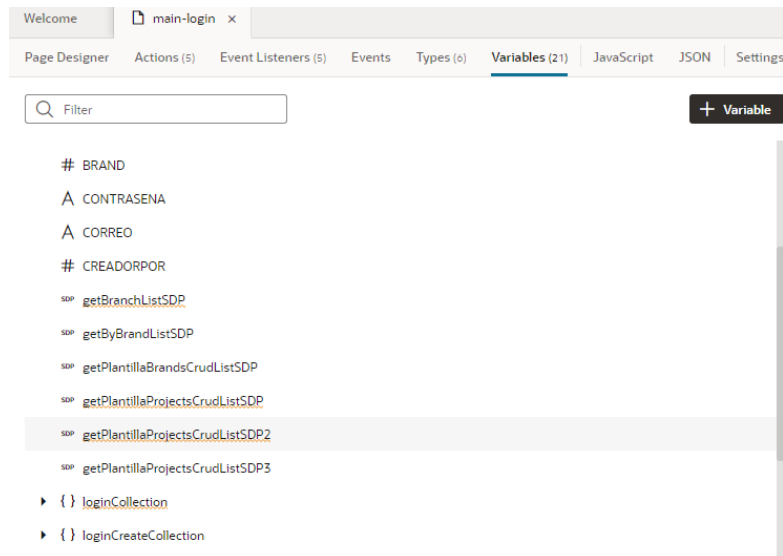


Figura 117. Variables Main-Login [Autoría propia]

Iniciar sesion

Email

Password

→ Iniciar sesion

Aun no esperte del equipo?

Registrese, dando click al boton adjunto.


Restrame

Figura 118. Pantalla 7 [Autoría propia]

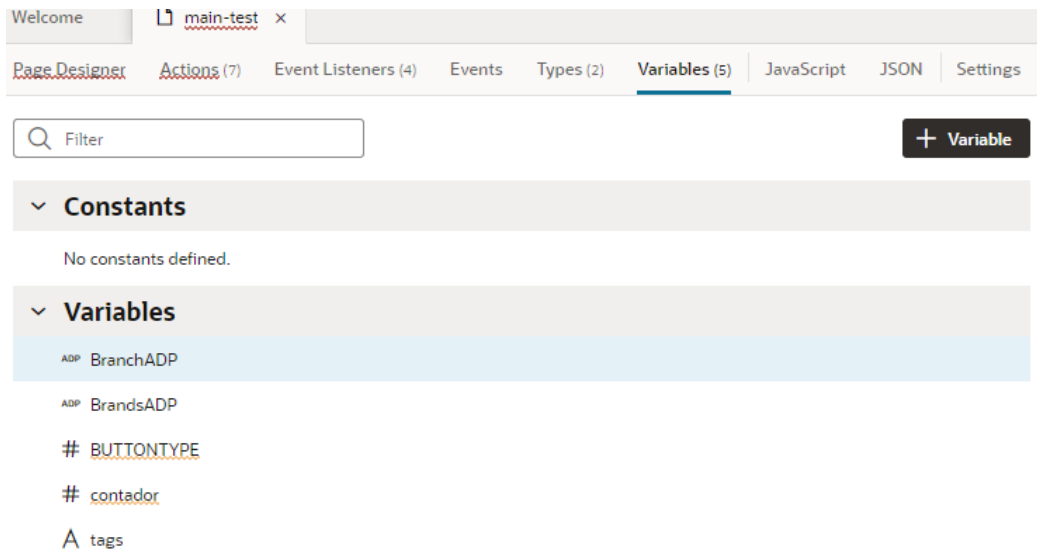


Figura 119. Variables Main-test [Autoría propia]

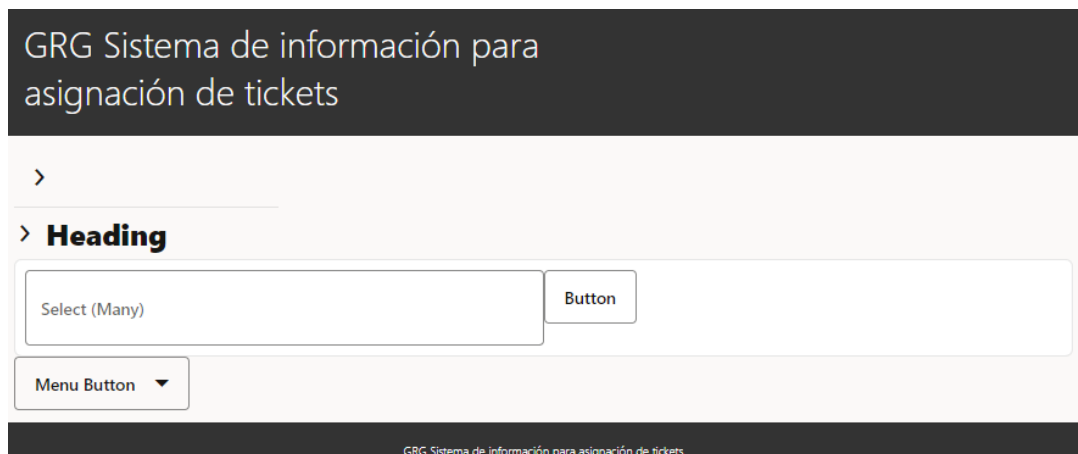


Figura 120. Pantalla 8 [Autoría propia]

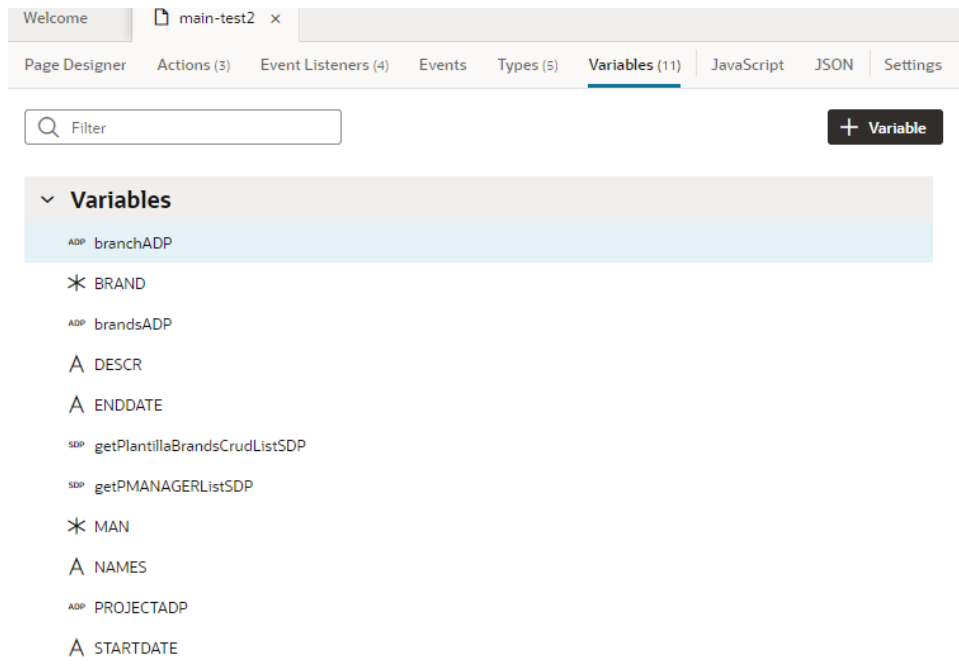


Figura 121. Variables Main-test2 [Autoría propia]

> Registrar Marca

> Registrar Sucursal

> Marcas disponibles

brand_name	brand_id	creation_date
Sub Way	323	2022-11-07T05:07:2
4CE S.A. de C.V.	261	2022-10-25T20:21:0
testingBrand	82	2022-10-08T00:00:0
McDonalds	26	2022-10-07T20:29:4

Heading

Text

Text

Text

Text

Button

Button

Figura 122. Pantalla 9 [Autoría propia]

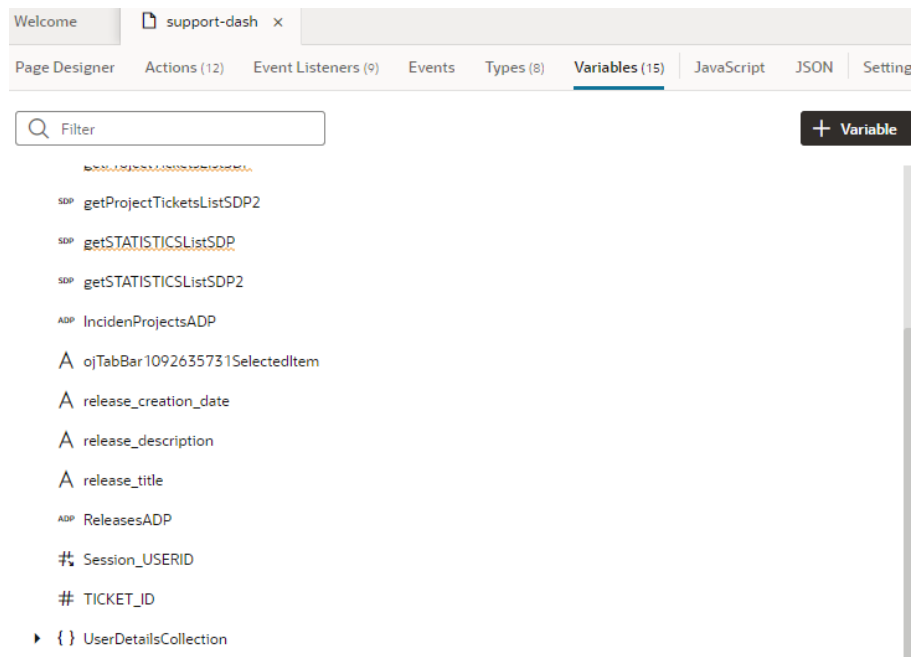


Figura 123. Variables Support-dash [Autoría propia]

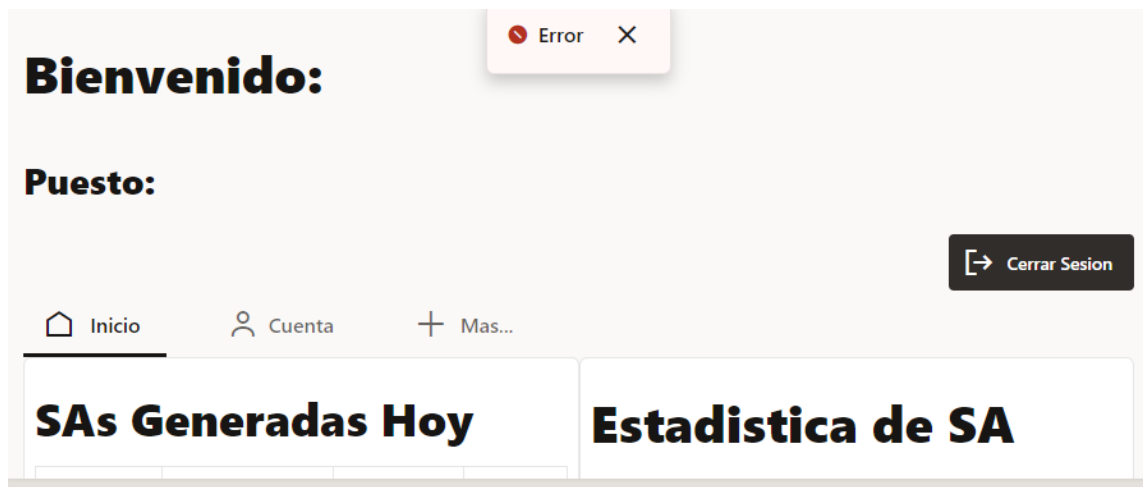


Figura 124. Pantalla 10 [Autoría propia]

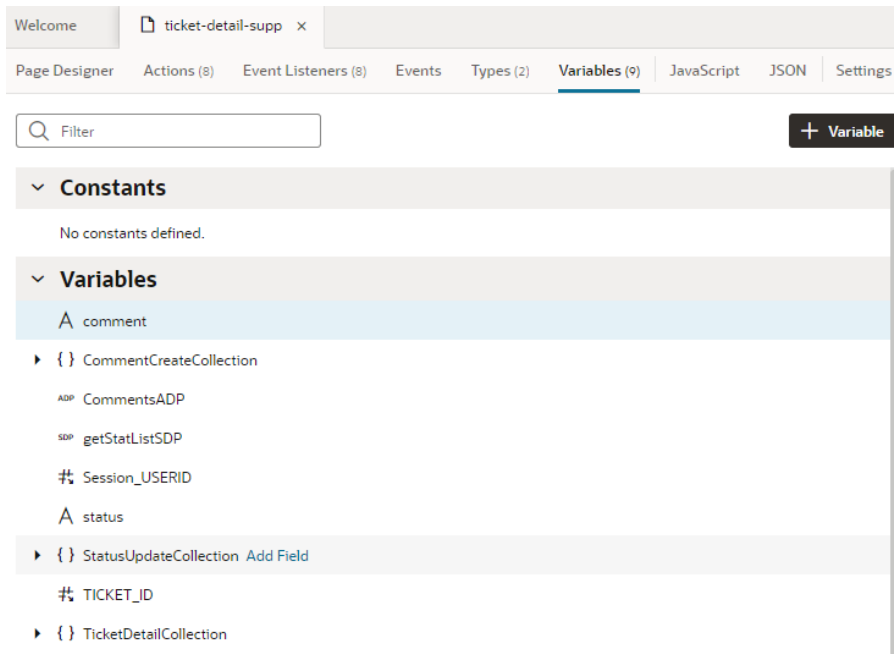


Figura 125. Variables ticket-detail-supp [Autoría propia]

Detalles de SA

Regresar

Error

Error

Título	Sucursal	Creado a las
URL	Proyecto	Prioridad
Marca	Descripción	Status

Figura 126. Pantalla 11 [Autoría propia]

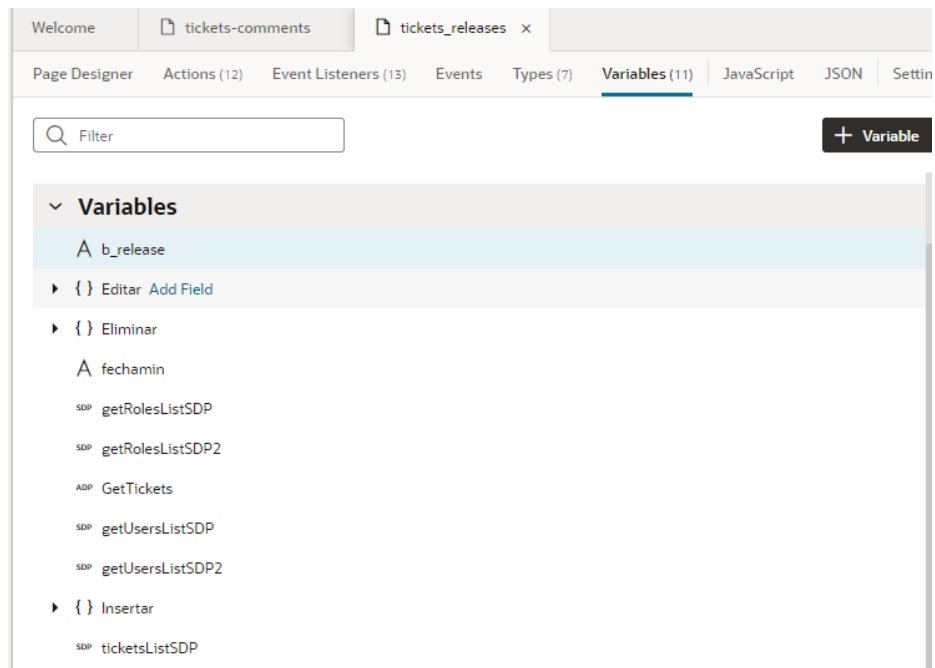


Figura 127. Variables tickets_releases [Autoría propia]

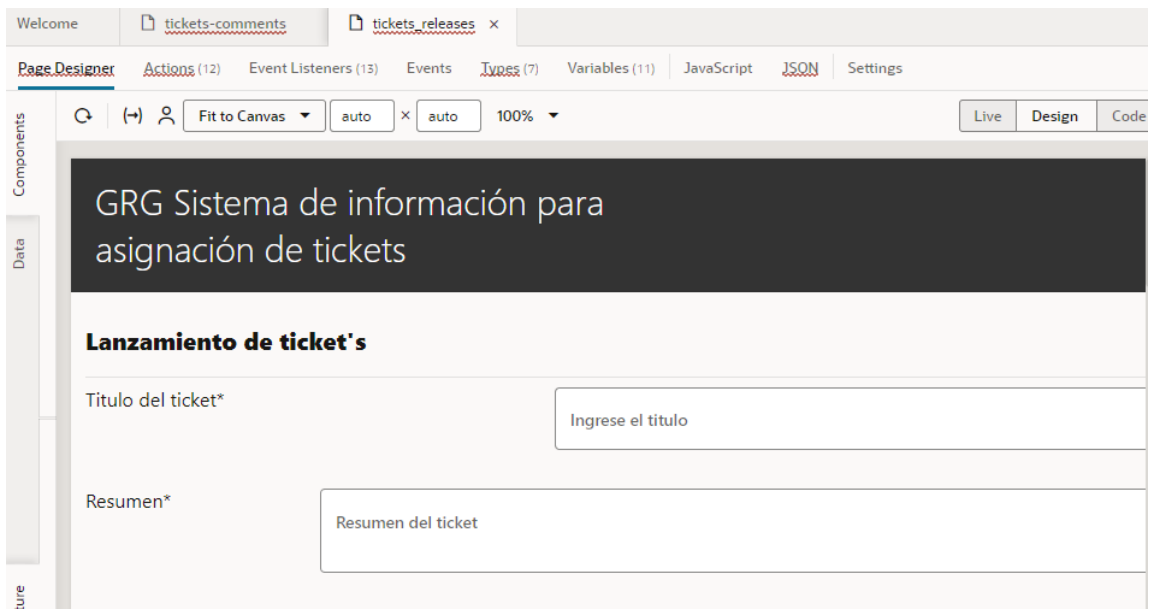


Figura 128. Pantalla 12 [Autoría propia]

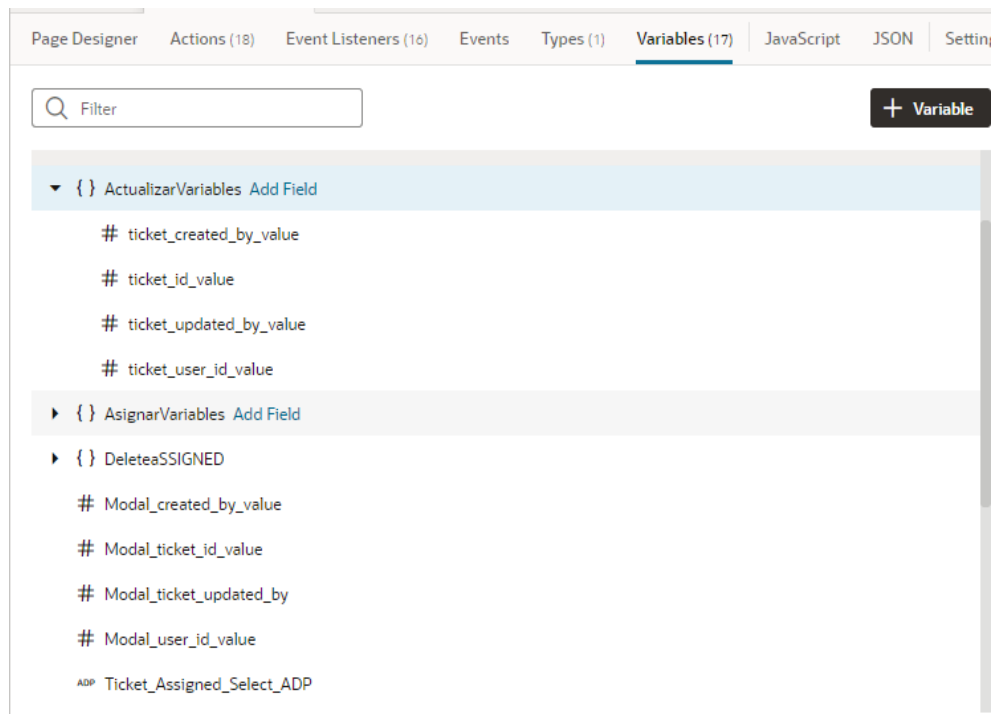


Figura 129. Variables Main-Start [Autoría propia]

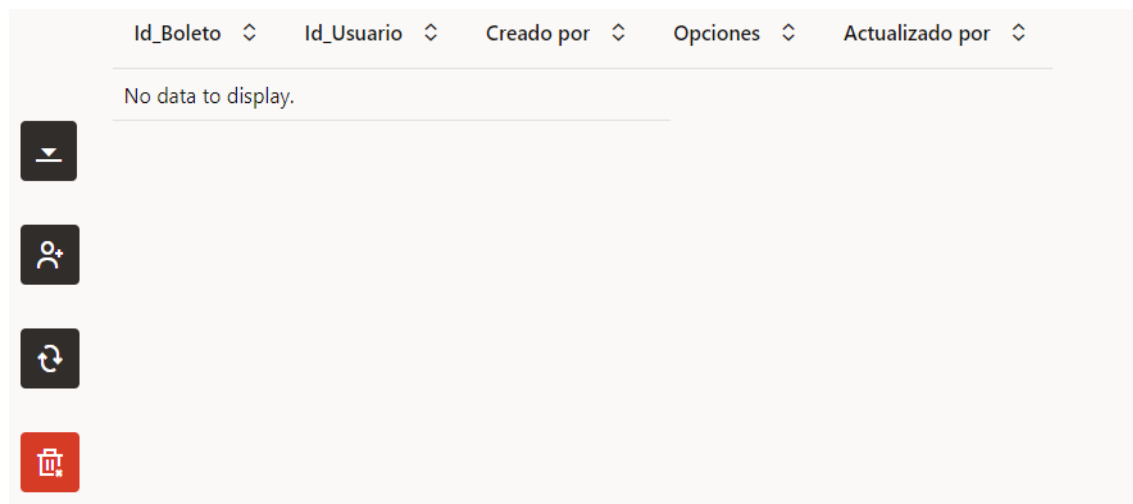


Figura 130. Pantalla 13 [Autoría propia]

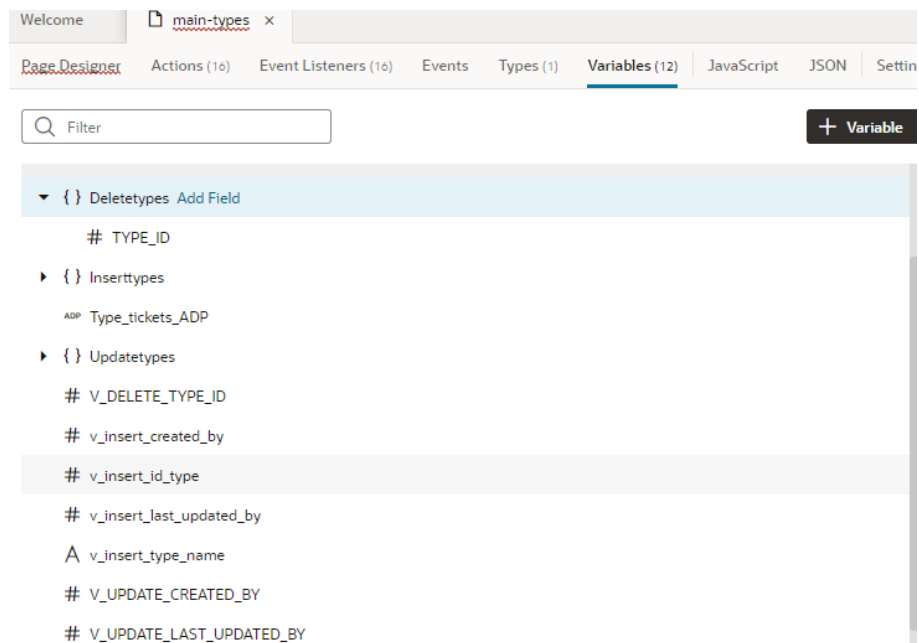


Figura 131. Variables Main-types [Autoría propia]

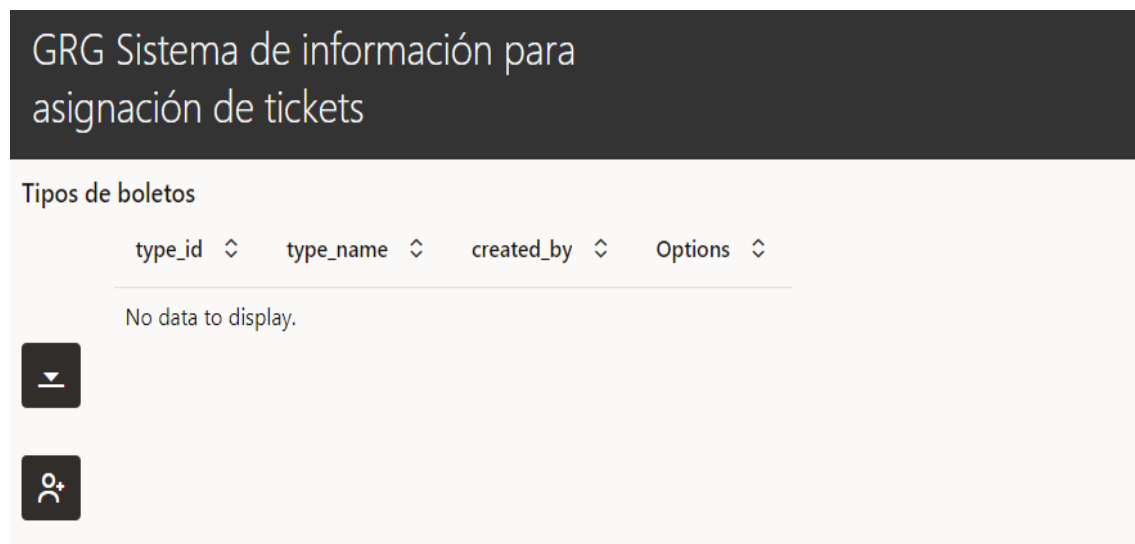


Figura 132. Pantalla 14 [Autoría propia]

CONCLUSIONES

La importancia de las áreas dentro de un organismo, dependen de muchos factores organizacionales, como parte fundamental de la empresa, se considera al personal, ya que se logró observar que cuentan con comunicación a la hora de trabajar en equipo, existe una persona encargada de brindar las nuevas ideas o aportes ante las dificultades que se requieren cubrir, tomando en cuenta las opiniones de todo el equipo involucrado en el proyecto.

Por lo que se puede admitir que si se alcanzó el objetivo de reducir la desorganización en los procesos de actualización de la información de los clientes (estar al corriente con los datos para cumplir en tiempo y forma), debido a que se optimizó el servicio de atención al cliente mediante sus necesidades, construyendo y diseñando el sistema de gestión de tickets, determinando que los canales de comunicación permiten controlar las incidencias de soporte.

La gestión y administración de este sistema se realizó de forma electrónica, por lo tanto, la información se almacena en una base de datos para después consultar su contenido, monitoreando el flujo de consultas y de esta manera poder establecer prioridades para agilizar el proceso de tender los requerimientos del cliente.

Las debilidades con las que puede contar la empresa no son más que áreas de oportunidad, ya que, contemplando los posibles errores, se puede hacer una propuesta de mejora contemplando las fortalezas con las que cuenta.

Los alcances y conocimientos adquiridos a través del proyecto me permitieron el acceso para desarrollar lo comprendido a lo largo de la carrera y logrando establecer que la innovación y el mejoramiento continuo son procesos que definen la calidad y refleja lo que la empresa 4CE necesita para alcanzar un nivel más alto de competitividad a lo largo del tiempo de operación.

La forma de proteger a las empresas de cualquier vulnerabilidad ha evolucionado a través de los avances tecnológicos debido al incremento de amenazas que nos obligan a adoptar nuevas políticas de seguridad.

Las recomendaciones implican un beneficio en un sentido general, de manera que la utilidad de estas radica en la valoración de la sugerencia. Se requiere organizar las medidas necesarias por prioridad, como se muestra a continuación:

- 1.- Diseñar un plan a partir del funcionamiento y necesidades del negocio, con el que se buscará la protección de información.
- 2.- Elaborar un diagrama entidad/relación, para de esta manera obtener la interacción de personas, objetos y conceptos.
- 3.- Ser constante en el aprendizaje y actualización de las nuevas formas de protección.
- 4.- Capacitar a los empleados para incluirlos en las practicas que deben seguir para poder minimizar los riesgos.
- 5.- Tener siempre en cuenta que los datos de una empresa son muy sensibles, por lo tanto, siempre se debe buscar una técnica que nos ayude a la alteración de la información para que no sea legible ante un ataque, siendo una de las maneras más eficientes el cifrado de datos como contraseñas de usuarios, listas de contactos, correo electrónico y conexiones a terminales remotas, por mencionar algunos.
- 6.- Adquirir las herramientas necesarias para llevar a cabo la realización del sistema, ayuda en demasía a reducir tiempo en la realización de los proyectos.

Lo antes mencionado busca poder contribuir a un mejor desarrollo en los entregables, como en la capacidad de absorción de los conocimientos obtenidos.

COMPETENCIAS DESARROLLADAS

En este apartado, se logran plasmar las competencias desarrolladas durante la realización del proyecto.

Se redactará de forma breve la explicación de cómo se trabajó con las competencias, separándolas entre competencia específica y competencia genérica.

Estas competencias aportan al perfil del Ingeniero en Sistemas Computacionales las siguientes habilidades:

- Coordinar y participar en equipos multidisciplinarios para la aplicación de soluciones innovadoras en diferentes contextos.
- Diseñar, implementar y administrar bases de datos optimizando los recursos disponibles, conforme a las normas vigentes de manejo y seguridad de la información.
- Desarrollar y administrar software para apoyar la productividad y competitividad de las organizaciones cumpliendo con estándares de calidad.

Actualmente la información es un activo importante en las organizaciones, ya que si posee las características necesarias apoya a las personas en la eficiente toma de decisiones, por lo que se requiere que inviertan en infraestructura que les permita gestionar y resguardar dicho activo, pero también necesitan de profesionistas especializados que sean capaces de administrarlo

Competencia para desarrollar. programación orientada a objetos (competencia específica)

Desarrolla soluciones de software para resolver problemas en diversos contextos utilizando programación concurrente, acceso a datos, que soporten interfaz gráfica de usuario y consideren dispositivos móviles.

Competencia previa

Diseña y desarrolla programas para la solución de problemas computacionales utilizando el paradigma orientado a objetos.

No.	Temas	Subtemas
1	Interfaz gráfica de usuario.	1.1 Creación de interfaz gráfica para usuarios. 1.2 Tipos de eventos. 1.3 Manejo de eventos. 1.4 Manejo de componentes gráficos de control.
2	Componentes y librerías.	2.1 Definición conceptual de componentes, paquetes / librerías. 2.2 Uso de librerías proporcionadas por el lenguaje. 2.3 Creación de componentes (visuales y no visuales) definidos por el usuario 2.4 Creación y uso de paquetes/librerías definidas por el usuario.
3	Programación concurrente (Multihilos).	3.1 Concepto de hilo. 3.2 Comparación de un programa de flujo único contra uno de flujo múltiple. 3.3 Creación y control de hilos. 3.4 Sincronización de hilos computación.
4	Acceso a datos.	4.1 Introducción. 4.2 Conexión a origen de datos. 4.3 Manipulación de datos. 4.4 Visualización de datos.
5	Programación de dispositivos móviles.	5.1. Introducción a las tecnologías y herramientas móviles. 5.2 Clasificación y aplicaciones de los dispositivos móviles. 5.3 Entorno operativo de las aplicaciones móviles. 5.4 Desarrollo de aplicaciones móviles. 5.5. Aspectos de seguridad.

Figura 133. Competencia por desarrollar programación orientada a objetos [TESCo]

Competencia específica de tópicos avanzados de programación

Desarrolla programas para interactuar con el usuario de una manera amigable, utilizando GUI (Interfaz Gráfica de Usuario) manipuladas a través de eventos.

Competencia genérica

- Capacidad de análisis y síntesis.
- Capacidad de organizar y planificar.
- Comunicación oral y escrita.
- Habilidad para buscar y analizar información proveniente de fuentes diversas.
- Solución de problemas.
- Toma de decisiones.
- Capacidad crítica y autocrítica.
- Capacidad de trabajar en equipo.
- Capacidad de comunicar sus ideas.
- Capacidad de liderazgo.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades de investigación.
- Capacidad de aprender.
- Capacidad de adaptarse a nuevas situaciones.
- Capacidad de generar nuevas ideas (creatividad).
- Habilidad para trabajar en forma autónoma.
- Preocupación por la calidad.
- Búsqueda del logro.

Competencia para desarrollar de bases de datos

Implementa bases de datos para apoyar la toma de decisiones considerando las reglas de negocio.

Competencia previa

Analiza requerimientos definidos por el cliente, por la organización y diseña bases de datos para generar soluciones al tratamiento de información de acuerdo con sus reglas de negocio

No.	Temas	Subtemas
1.	Lenguaje de definición de datos	1.1 Instalación de un SGBD 1.2 Creación del esquema de la base de datos a. Modificación del esquema de la base de datos. b. Aplicación de constraints
2.	Lenguaje de manipulación de datos	2.1 Inserción, eliminación y modificación de registros 2.2 Consultas 2.3 Funciones, conversión, agrupamiento, ordenamiento 2.4 Joins 2.5 Subconsultas 2.6 Operadores set 2.7 Vistas
3.	Control de acceso	3.1 Tipos de usuario 3.2 Creación de usuarios 3.3 Privilegios a usuarios 3.4 Roles
4.	Concurrencia	4.1 Conceptos 4.2 Propiedades de las transacciones 4.3 Grados de consistencia 4.4 Niveles de aislamiento 4.5 Commit y rollback
5.	SQL Procedural	5.1 Stored Procedures 5.2 Functions 5.3 Triggers
6.	Conectividad de Bases de Datos	6.1 ODBC, ADO.NET, JDBC 6.2 Conectividad desde un lenguaje huésped o en dispositivos móviles

Figura 134. Competencia por desarrollar bases de datos [TESCo]

Competencia específica

Implementa bases de datos para apoyar la toma de decisiones considerando las reglas de negocio.

Competencias genéricas

- Capacidad de abstracción, análisis
- Capacidad de aplicar los conocimientos en la práctica
- Capacidad de comunicación escrita
- Habilidades para buscar, procesar y analizar información procedente de fuentes diversas
- Capacidad para actuar en nuevas situaciones.

FUENTES DE INFORMACIÓN

[1]Alves, C. (2021). Protocolo TCP / IP, para principiantes: La guía definitiva para principiantes para aprender el protocolo TCP / IP paso a paso. (n.p.): Amazon Digital Services LLC - KDP Print US.

[2]Arias, Á. (2014). Bases de Datos con MySQL. Estados Unidos: CreateSpace Independent Publishing Platform

[3]Baca Urbina, G. (2016). Introducción a la seguridad informática. México: Grupo Editorial Patria.

[4]Brennan, H. (2015). Full text - Number 11 (2013). The ANZTLA ejournal, 11. <https://doi.org/10.31046/anztla.vi11.276>

[5]BLANCOSOLSONA, A.,HUIDOBROMOYA, J. M.,JORDANCALERO, J. (2006). Redes de área local. España: Ediciones Paraninfo, S.A.

[6]Camuña Rodríguez, J. F. (2015). Lenguajes de definición y modificación de datos sql. IFCT0310. España: IC Editorial.

[7]Carmona, 2010. Teoria General de Sistemas: un enfoque hacia la ingeniería de sistemas. (n.d.). (n.p.): Lulu.com.

[8]Computación De Propósito General En Unidades De Procesamiento De Gráficos: Utilizar la Unidad de procesamiento de gráficos (GPU) para realizar cálculos que normalmente realiza la CPU. (2022). (n.p.): Mil Millones De Conocimientos [Spanish].

[9]España Boquera, M. C. (2003). Servicios avanzados de telecomunicación. España:Díaz de Santos.

[10]Eggeling, T., Frater, H. (2003). Ampliar,RepararyConfigurarsuPC. España: Mar combo

[11]Grazzini, H. O. (2021). Fibras ópticas: Conceptos teóricos y aplicaciones prácticas. Argentina: Jorge Sarmiento Editor - Universitas.

- [12]Hush, J. (2020). Redes informáticas Para Principiantes: La guía Completa de la tecnología inalámbrica, La Seguridad de Redes, Arquitectura de Las Computadoras Y Los Sistemas de comunicación.. (n.p.): Draft2Digital.
- [13]Joyanes Aguilar, L. (2022). Computación en la nube 2ed. España: Marcombo.
- [14]Maman Orfali, C. J. (2017). Oracle APEX 5.1: Una guía práctica para desarrollar aplicaciones web centralizadas usando Oracle Application Express. Estados Unidos: Clarisa Maman Orfali.
- [15]*Modelo de desarrollo incremental iterativo de software, 2023*
- [16]Nevado Cabello, M. V. N. C. (2010). Introducción a las Bases de Datos relacionales:Vol.106páginas. Vision Libros, 2010. https://books.google.com.mx/books?id=0UpB1NUdIC&source=gbp_navlinks_s
- [17]Pérez, M. P. (2010). Microsoft SQL Server 2008 R2. Motor de base de datos y administración: Vol. 592 páginas. RC Libros, 2011.
- [18]Pérez López, C. (2008). Oracle PL/SQL. España: Ra-Ma S.A. Editorial y Publicaciones.
- [18]Ramírez, E. V. (1986). Introducción a los microprocesadores: equipo y sistemas. Puerto Rico: Limusa.
- [19]Rubiales Gómez, M. (2021). Curso de desarrollo Web. HTML, CSS y JavaScript. Edición 2021. España: ANAYA MULTIMEDIA.
- [20]Sommerville, I. (2005). Ingeniería del software. España: Pearson Educación.
- [21]Silva, M. (2015). Sistemas Operativos. Colombia: Alpha Editorial.
- [22]Tanenbaum, A. S. (2003). Redes de computadoras. España: Editorial Alhambra S. A. (SP).
- [23]Urbina Ramírez, S. (1999). Informática y teorías del aprendizaje. Pixel-Bit. Revista de Medios y Educación, 12, 87-100.

ANEXOS

En el siguiente apartado se podrá visualizar la técnica utilizada para identificar las áreas de oportunidad, debilidad, amenazas y fortalezas, con el fin de desarrollar una estrategia para mejorar la organización de la empresa 4CE.

Estos factores internos y externos nos permiten condicionar la situación real de empresa, para así lograr posicionar el proyecto.

FORTALEZAS	DEBILIDADES
1.- Organización específica desde el principio del proyecto hasta su desarrollo.	1.- Complicación en la organización de los datos
2.- Constante monitoreo para actualizaciones en la aplicación.	1.- Estructuración en las implementaciones, para el mejoramiento de la gestión en la empresa
3.- Se cuenta con información necesaria de los colaboradores, clientes y proveedores de la empresa 4CE.	3.- Información errónea ingresada por el usuario
4.- Acceso total al usuario para consultar sus movimientos en la aplicación o en una página web.	4.- Acceso restringido a las consultas de cada usuario
5.- Contacto directo con el usuario a través de correo electrónico.	5.- Contacto deficiente e indirecto con el usuario/cliente.
OPORTUNIDADES	AMENAZAS
1.-mayor amplitud de campo laboral	1.- El sistema no sea totalmente capacitado para realizar lo requerido en la empresa
2.-mayor oportunidad de crecimiento profesional	2.- El personal o usuarios realicen acciones o mal uso de la información almacenada
3.-oportunidad para obtener mayor experiencia en el ámbito	3.- Falta de capacitación para los usuarios
4.-mejor optimización en el manejo de la operatividad financiera de una o mas empresas	4.- La información almacenada por los usuarios no sea la correcta
5.-mejor oportunidad de adquirir conocimientos en la realización de nuevos proyectos	5.- Robo de información

Figura 135. Matriz foda [Autoría propia]

1.1 CARTA DE NO INCOVENIENCIA



Cuautitlán, Edo. Méx., a 24 de Agosto de 2023

ASUNTO: Carta de no inconveniencia

A quien corresponda

PRESENTE

Por medio del presente me es grato informarle que la alumna **Beatriz Esperanza García Fragoso**, con número de control **201820231** de la carrera de **Ingeniería en Sistemas Computacionales**, ha concluido de manera exitosa su proceso de residencias profesionales en la **Dirección de Operaciones** de esta empresa, con el proyecto denominado Sistema de información web con Visual Builder y Oracle Cloud para la gestión de tickets de la empresa 4CE, del 27 de Septiembre de 2022 al 31 de Enero del 2023, cubriendo un horario de 09:00 a 18:00 horas de lunes a viernes, comprendiendo un total de 500 horas.

Adicionalmente, se expide el presente documento para autorizar la no inconveniencia para que la alumna Beatriz Esperanza García Fragoso utilice el proyecto desarrollado con la empresa como parte de su proyecto de titulación.

Sin otro particular por el momento, aprovecho la ocasión para enviarle un cordial saludo.

ATENTAMENTE.

Recursos Humanos

