

**DESARROLLO DE LOS MÓDULOS DE RADICACIÓN Y ADMINISTRACIÓN
DE USUARIOS Y ROLES DE LA APLICACIÓN WEB DE GESTIÓN DOCUMENTAL
UTILIZANDO EL MARCO DE INTERFAZ DE USUARIO VAADIN PARA LA
EMPRESA OPE COLOMBIA**

Autor

Deimer Joel Cegarra Cordero

Director

PhD. Diana Ruth Martínez Suárez



**INGENIERÍA DE SISTEMAS
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y
TELECOMUNICACIONES – EEST.
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
UNIVERSIDAD DE PAMPLONA
VILLA DEL ROSARIO, 2022**

Dedicatoria

A mis padres y hermanos que siempre me han apoyado y motivado a cumplir mis sueños, a Dios por darme salud y no permitir que desista de mis objetivos y a mis familiares que siempre me han deseado lo mejor y apoyado para llegar a cumplir esta meta.

Agradecimientos

A Dios que siempre me ha guiado por el mejor camino para lograr lo que me propongo y llegar hasta este momento tan especial.

A la empresa OPE Colombia por abrirme sus puertas para realizar mi práctica profesional y ayudarme a desarrollar mis habilidades, y a todas las personas que de alguna manera me colaboraron.

A los docentes que me brindaron el conocimiento para comenzar a formarme como profesional y a la PhD Diana Ruth Martínez Suárez por brindarme su apoyo y tiempo en mi proceso de práctica profesional y en la elaboración del presente trabajo.

Resumen

El presente trabajo trata sobre la creación de los módulos de Radicación y Administración de Usuarios y Roles de la aplicación web de gestión documental de la empresa OPE Colombia. Estos módulos tienen como propósito permitir la creación de usuarios y perfiles de acceso para las funcionalidades de gestión de la producción, trámite, almacenamiento digital y recuperación de documentos. El proceso de desarrollo de este software se llevó a cabo aplicando la metodología Scrum en conjunto con la norma ISO 29110, con el fin de mejorar la calidad del producto a través de sus guías y estándares, e implementando tecnologías tales como Vaadin en el FrontEnd, Spring Boot en el Backend, y PostgreSQL como motor de base de datos, en sus versiones más recientes. Entre los resultados esperados está la mejora en los procesos de gestión documental de las entidades públicas del departamento y del país que contraten este servicio con la empresa OPE Colombia, mediante la optimización de los mismos, evitando el manejo de papel.

Palabras clave:

Radicación y Administración de Usuarios y Roles, gestión documental, ISO 29110.

Abstract

The present work deals with the creation of the modules of Filing and Administration of Users and Roles of the document management web application of the company OPE Colombia. The purpose of these modules is to allow the creation of users and access profiles for the production management, processing, digital storage and document retrieval functionalities. The development process of this software was carried out applying the Scrum methodology in conjunction with the ISO 29110 standard, in order to improve the quality of the product through its guides and standards, and implementing technologies such as Vaadin in the FrontEnd, Spring Boot in the BackEnd, and PostgreSQL as the database engine, in its most recent versions. Among the expected results is the improvement in the document management processes of the public entities of the department and the country that contract this service with the company OPE Colombia by optimizing them, avoiding the handling of paper.

Keywords:

Filing and Administration of Users and Roles, document management, ISO 29110.

Contenido

1	Generalidades	20
1.1.	Planteamiento del Problema	20
1.2.	Justificación de Problema	20
1.3.	Delimitaciones	21
1.3.1.	Objetivo general	21
1.3.2.	Objetivos específicos.....	21
1.3.3.	Acotaciones	21
2	Marco Teórico	23
2.1.	Antecedentes de la Investigación.....	23
2.1.1.	Antecedentes Internacionales	23
2.1.2.	Antecedentes Nacionales.....	24
2.1.3.	Antecedentes Regionales.....	25
2.2.	Base Teórica	26
2.2.1.	ISO/IEC 29110.....	26
2.2.2.	Sistema de Gestión Documental	27
2.2.3.	Radicación de documentos.....	28
2.2.4.	Microservicios.....	29
2.2.5.	API REST.....	31
2.2.6.	Arquitectura hexagonal	32
2.2.7.	Arquitectura MVP	33
2.3.	Marco Legal	33
2.3.1.	Ley estatutaria 1581 de 2012, Por la cual se dictan disposiciones generales para la protección de datos personales.....	33
2.3.2.	Ley 594 de 2000, Por medio de la cual se dicta la Ley General de Archivos y se dictan otras disposiciones.....	34

2.3.3.	Ley 23 de 1982, sobre derechos de autor	34
2.3.4.	Decreto 2106 de 2019, Por el cual se dictan normas para simplificar, suprimir y reformar trámites, procesos y procedimientos innecesarios existentes en la administración pública”	34
2.3.5.	Circular externa 005 de 2012	35
2.4.	Ope Colombia	35
2.4.1.	Productos de software	35
3	Desarrollo del Sistema	37
3.1.	Herramientas de desarrollo	37
3.1.1.	Java.....	37
3.1.2.	Vaadin Flow	37
3.1.3.	Spring Boot	38
3.1.4.	Eclipse IDE	38
3.1.5.	PostgreSQL	39
3.1.6.	MyBatis	40
3.1.7.	MinIO	40
3.1.8.	RabbitMQ.....	40
3.1.9.	Docker	41
3.1.10.	Docker Compose	41
3.1.11.	Jenkins	41
3.2.	Metodología.....	42
3.2.1.	SCRUM.....	42
3.2.1.1.	Roles.....	42
3.2.1.2.	Documentos SCRUM o Artefactos	43
3.2.1.3.	Eventos	44
3.2.1.4.	Historias de usuario	45

3.2.1.5.Hitos	46
3.3. Análisis de Requerimientos Funcionales y no Funcionales.....	46
3.3.1. Diagramas de casos de uso.....	46
3.3.2. Historias de usuario y criterios de aceptación	51
3.4. Diseño	64
3.4.1. Diseño Arquitectónico.....	64
3.4.2. Arquitectura Hexagonal	68
3.4.3. Arquitectura MVP	71
3.4.4. Diseño de bases de datos	71
3.5. Desarrollo FrontEnd	74
3.5.1. Siepdoc_micro_vaadin	74
3.6. Desarrollo BackEnd	110
3.6.1. Siepdoc_commons_permissions	111
3.6.2. Siepdoc_commons_vo	114
3.6.3. Siepdoc_commons_services.....	114
3.6.4. Siepdoc_micro_eureka	116
3.6.5. Siepdoc_micro_config	117
3.6.6. Siepdoc_micro_scgu	117
3.6.7. Siepdoc_micro_scc.....	124
3.6.8. Siepdoc_micro_scgr	125
3.6.9. Siepdoc_micro_envio_correo.....	128
3.6.10. Siepdoc_micro_reports.....	129
3.6.11. Siepdoc_micro_oauth.....	132
3.6.12. Siepdoc_micro_gateway	136
3.7. Fase de pruebas	138

Conclusiones	143
Recomendaciones	144
Referencias.....	145
ANEXO 1. PRODUCT BACKLOG.....	149
ANEXO 2. DAILY SCRUM	151
ANEXO 3. HITOS.....	152
ANEXO 4. MANUAL DE OPERACIONES Y MANTENIMIENTO	153

Lista de Ilustraciones

Ilustración 1 Radicación de documentos	29
Ilustración 2 Patrón básico de Arquitectura de Microservicios.	30
Ilustración 3 Beneficios de los microservicios.	31
Ilustración 4 Diagrama de Caso de uso Iniciar Sesión	47
Ilustración 5 Diagrama de Caso de uso Generar radicado de salida.....	48
Ilustración 6 Diagrama de Caso de uso Generar radicado de entrada	49
Ilustración 7 Diagrama de Casos de uso para CRUD de tablas básicas	50
Ilustración 8 Diagrama de Caso de uso Generar reportes.....	51
Ilustración 9 Arquitectura SIEPDOC, tomada de OPE Colombia.....	66
Ilustración 10 Arquitectura de herramientas para el funcionamiento de SIEPDOC, tomada de OPE Colombia.	67
Ilustración 11 Arquitectura para despliegue e integración continua, tomada de OPE Colombia.	67
Ilustración 12 Arquitectura Hexagonal.	69
Ilustración 13 Regla de dependencia de capas de la Arquitectura Hexagonal.	69
Ilustración 14 Arquitectura Hexagonal aplicada a SIEPDOC	70
Ilustración 15 Diagrama en bloques que representa el funcionamiento de la arquitectura MVP.	71
Ilustración 16 Modelo ER del esquema scgu.....	72
Ilustración 17 Modelo ER del esquema scc.....	73
Ilustración 18 Modelo ER del esquema scgd.....	74

Ilustración 19 Menú de opciones expandido	75
Ilustración 20 Menú de opciones minimizado	76
Ilustración 21 Menú de Radicación	76
Ilustración 22 Menú de configuracion parte 1	77
Ilustración 23 Menú de configuracion parte 2	77
Ilustración 24 Menú de Estadísticas	78
Ilustración 25 Formulario Radicación de salida parte 1	79
Ilustración 26 Formulario Radicación de salida parte 2	79
Ilustración 27 Formulario Radicación de salida parte 3	80
Ilustración 28 Opciones luego de generar un radicado de salida.....	81
Ilustración 29 Generar Sticker	81
Ilustración 30 Cargar documento principal.....	82
Ilustración 31 Carga exitosa de documento principal.....	83
Ilustración 32 Lista de adjuntos	83
Ilustración 33 Cargar adjuntos	84
Ilustración 34 Lista de informados	85
Ilustración 35 Informar	86
Ilustración 36 Lista de otros antecedentes	87
Ilustración 37 Seleccionar antecedente	87
Ilustración 38 Lista de otros destinatarios	88
Ilustración 39 Editar email del destinatario	88
Ilustración 40 Seleccionar destinatario	89
Ilustración 41 Formulario Radicación de entrada parte 1	90

Ilustración 42 Formulario Radicación de entrada parte 2	90
Ilustración 43 Opciones luego de generar un radicado de entrada	91
Ilustración 44 Listado usuarios	91
Ilustración 45 Formulario usuarios	92
Ilustración 46 Seleccionar dependencia.....	93
Ilustración 47 Seleccionar cargo	93
Ilustración 48 Seleccionar funcionario	94
Ilustración 49 Seleccionar rol	95
Ilustración 50 Detalle usuarios.....	95
Ilustración 51 Actualizar usuarios	96
Ilustración 52 Link para crear contraseña	97
Ilustración 53 Formulario crear contraseña	97
Ilustración 54 Contraseña creada	98
Ilustración 55 Recuperar contraseña.....	98
Ilustración 56 Formulario para enviar email.....	99
Ilustración 57 Listado roles.....	100
Ilustración 58 Formulario roles.....	101
Ilustración 59 Permisos tipos de radicados.....	102
Ilustración 60 Permisos CRUD gestión de usuarios	103
Ilustración 61 Permisos CRUD de configuración.....	104
Ilustración 62 Permisos CRUD gestión de documentos	105
Ilustración 63 Permisos estadísticas.....	106
Ilustración 64 Detalle roles	107

Ilustración 65 Actualizar roles	108
Ilustración 66 Estructura de carpetas Siepdoc_micro_vaadin parte 1	109
Ilustración 67 Estructura de carpetas Siepdoc_micro_vaadin parte 2	109
Ilustración 68 Presenter de la clase ListUsuariosViewImpl	110
Ilustración 69 Visión general de los proyectos en Eclipse IDE.....	111
Ilustración 70 Estructura de paquetes siepdoc_commos_permissions	112
Ilustración 71 Definición de niveles de permisos para CRUD de tablas	112
Ilustración 72 Definición de permisos CRUD de usuarios.....	113
Ilustración 73 Definición de niveles de permisos para Radicación y Estadísticas	113
Ilustración 74 Permisos para Radicación	113
Ilustración 75 Estructura de paquetes siepdoc_commos_vo	114
Ilustración 76 Estructura de paquetes siepdoc_commons_services.....	115
Ilustración 77 Clase común para los controller.....	116
Ilustración 78 Configuración Netflix Eureka Server	116
Ilustración 79 Configuración Config Server.....	117
Ilustración 80 Conexión al repositorio de archivos de propiedades	117
Ilustración 81 Estructura de paquetes Siepdoc_micro_scgu.....	118
Ilustración 82 Controller CRUD tabla usuarios.....	119
Ilustración 83 Interface del converter para la tabla usuarios	119
Ilustración 84 Implementación de converter de la tabla usuarios.....	120
Ilustración 85 Interface del service para la tabla usuarios	120
Ilustración 86 Implementación del service para la tabla usuarios	121
Ilustración 87 Caso de uso create para tabla usuarios.....	121

Ilustración 88 Mapper básico de la tabla usuarios	122
Ilustración 89 Mapper extendido de la tabla usuarios	122
Ilustración 90 MapperExt xml de la tabla usuarios parte 1	123
Ilustración 91 MapperExt xml de la tabla usuarios parte 2	123
Ilustración 92 Archivos MyBatis	124
Ilustración 93 Estructura de paquetes Siepdoc_micro_scc	124
Ilustración 94 Estructura de paquetes Siepdoc_micro_scgr	125
Ilustración 95 Caso de uso Crear Radicación	126
Ilustración 96 Caso de uso para para almacenar documento principal del radicado	126
Ilustración 97 Repositorio para almacenar archivos en minio	127
Ilustración 98 Repositorio para obtener URL de ubicación de un archivo de minio	127
Ilustración 99 Repositorio para obtener los bytes de los archivos de minio	128
Ilustración 100 Repositorio para eliminar un archivo de minio	128
Ilustración 101 Estructura de paquetes Siepdoc_micro_envio_correo	129
Ilustración 102 Caso de uso para envío de emails	129
Ilustración 103 Estructura de paquetes Siepdoc_micro_reports	130
Ilustración 104 Caso de uso para generar reportes	131
Ilustración 105 Plantilla reporte Correspondencia de entrada en Jaspersoft Studio	131
Ilustración 106 Estructura de paquetes Siepdoc_micro_oauth	132
Ilustración 107 Clase de configuración de Spring Security	133
Ilustración 108 Clase CustomAuthenticationFilter parte 1	133
Ilustración 109 Clase CustomAuthenticationFilter parte 2	134
Ilustración 110 Clase CustomAuthenticationFilter parte 3	134

Ilustración 111 Clase CustomAuthenticationFilter parte 4.....	135
Ilustración 112 Clase CustomAutorizationFilter	135
Ilustración 113 Creación del UserDetails	136
Ilustración 114 Estructura de paquetes Siepdoc_micro_gateway.....	136
Ilustración 115 Clase de configuracion de Spring Security	137
Ilustración 116 Clase JwtAuthenticationFilter.....	137
Ilustración 117 Clase AuthenticationManagerJwt.....	138
Ilustración 118 Configuracion de rutas de los microservicios.....	138
Ilustración 119 Descripción de criterios para estimación del tamaño parte 1	149
Ilustración 120 Descripción de criterios para estimación del tamaño parte 1	149
Ilustración 121 Estimación de tamaño parte 1	150
Ilustración 122 Estimación de tamaño parte 2	150

Lista de Tablas

Tabla 1 Login.....	51
Tabla 2 Cerrar sesión	52
Tabla 3 Reestablecer contraseña de usuario	52
Tabla 4 CRUD de usuarios	53
Tabla 5 CRUD de Roles y permisos	53
Tabla 6 CRUD de funcionarios.....	54
Tabla 7 CRUD de los tipos de identificación	54
Tabla 8 CRUD de los continentes.....	54
Tabla 9 CRUD de los países	55
Tabla 10 CRUD de los departamentos.....	55
Tabla 11 CRUD de los municipios	55
Tabla 12 CRUD de los medios de recepción	56
Tabla 13 CRUD de los días no hábiles	56
Tabla 14 CRUD de los tipos de radicados	56
Tabla 15 CRUD de los cargos	57
Tabla 16 CRUD de los tipos de transacción	57
Tabla 17 CRUD de las dependencias.....	58
Tabla 18 CRUD de los estados de los radicados	58
Tabla 19 CRUD de los ciudadanos	58
Tabla 20 CRUD de los tipos de documentos	59
Tabla 21 Radicaciones de entrada.....	59

Tabla 22 Radicaciones de salida	61
Tabla 23 Otros tipos de radicaciones	63
Tabla 24 Generar reportes.....	63
Tabla 25 Registrar datos obligatorios del formulario de radicación de entrada.	139
Tabla 26 Registrar datos obligatorios incompletos del formulario de radicación de entrada.	139
Tabla 27 Radicar sin registrar datos del formulario de radicación de entrada.....	140
Tabla 28 Registrar datos obligatorios del formulario de radicación de salida.....	140
Tabla 29 Registrar datos obligatorios incompletos del formulario de radicación de salida.	141
Tabla 30 Radicar sin registrar datos del formulario de radicación de salida.	141
Tabla 31 Daily Scrum	151
Tabla 32 Hitos	152

1 Generalidades

1.1. Planteamiento del Problema

Actualmente, existen empresas que manejan sus procesos de gestión documental de forma física y tienen almacenados decenas y hasta cientos de documentos, lo cual hace que la gestión correspondiente de cada documento sea un proceso lento y poco productivo. En entidades públicas como las alcaldías, por ejemplo, cuando llega una solicitud o una queja, entre otras, se verifica que sea competencia de la entidad, de ser así se crea el radicado con sus formatos correspondientes, posteriormente se envía a la dependencia correspondiente para darle el trámite necesario al documento. Una vez hecho el trámite se da respuesta según el caso y se hace la conservación del documento radicado. Es por esto, que existe la necesidad de mejorar los procesos de gestión documental en las entidades públicas con el fin de optimizarlos y que sean mucho más rápidos, eficientes y seguros. De esta necesidad nace la pregunta de investigación:

¿Cómo una aplicación web puede solucionar los problemas relacionados con la radicación de documentos de las entidades públicas del departamento Norte de Santander y del país?

1.2. Justificación de Problema

Hoy en día la tecnología se ha vuelto una solución de calidad y mejora en el ámbito empresarial gracias a la automatización de procesos aportando seguridad y eficacia y reduciendo costos y tiempo a la hora de llevar a cabo su trabajo. Con el constante crecimiento y avance de la tecnología es inevitable que las empresas hagan uso de ésta, tal es el caso de las aplicaciones web de gestión documental, que ayudan a las empresas a agilizar sus procesos y les garantiza la seguridad de su información mediante la gestión electrónica de la producción, trámite,

almacenamiento digital y recuperación de documentos, evitando el manejo en papel y contribuyendo así al medio ambiente. En consecuencia, la creación de una aplicación web para gestionar los procesos documentales es de vital importancia para las entidades públicas del departamento Norte de Santander y del país.

1.3. Delimitaciones

1.3.1. Objetivo general

Desarrollar los módulos de Radicación y Administración de Usuarios y Roles de la aplicación web de gestión documental utilizando el marco de interfaz de usuario Vaadin para la empresa OPE Colombia.

1.3.2. Objetivos específicos

- Realizar el análisis de los requerimientos funcionales y no funcionales.
- Aplicar las recomendaciones del estándar ISO 29110.
- Construir los módulos Radicación y Administración de Usuarios y Roles de la aplicación de gestión documental de OPE de Colombia.
- Validar la aplicación mediante las pruebas de aceptación.

1.3.3. Acotaciones

- El módulo de Administración de Usuarios y Roles incluye la gestión de usuarios y roles, así como la asignación de permisos a las distintas funcionalidades del proyecto de gestión documental en general. Además, incluye también el control del acceso de usuarios por medio del Login.
- El módulo de Radicación incluye CRUD de tablas básicas de radicación y proceso de radicación. Además, incluye reportes como plantilla de entrada, plantilla de salida, y listado de radicaciones por usuario a nivel general y a nivel de detalle.

- En el proceso se realizarán las etapas de planeación, investigación, análisis y diseño, y desarrollo basadas en la ISO 29110.
- En el proyecto se utilizará la metodología Scrum.
- Se aplicará el estándar ISO 29110 para mejorar la calidad del producto, y para mejorar el desempeño del grupo de trabajo del proyecto.
- El desarrollo de software será realizado en el lenguaje de programación Java en su versión 17 utilizando el marco de interfaz de usuario Vaadin en su versión 23, Java Spring Boot en su versión 2.6.0 y el motor de base de datos PostgreSQL en su versión 14.

2 Marco Teórico

2.1. Antecedentes de la Investigación

2.1.1. *Antecedentes Internacionales*

En el trabajo “Estudio de Factibilidad para la Implementación del Sistema de Gestión Documental “Quipux” en la Universidad Regional Autónoma de los Andes”, Freire dice que para lograr una eficiente gestión documental en una organización en la actualidad, es indispensable contar con un sistema informático que permita el manejo, almacenamiento y conservación de todos aquellos documentos institucionales que son vitales para la gestión. Las instituciones de educación superior en su afán de mejoramiento y en especial la Universidad Regional Autónoma de los Andes han considerado la evolución de sus procesos de comunicación interna mediante las TIC con el propósito de incrementar la eficiencia, eficacia y desempeño institucional, incluyéndolo en el Plan de Mejora Institucional. Este estudio pretende determinar la posibilidad de implementar el sistema QUIPUX en el contexto de la institución UNIANDES, con el propósito de apoyar el cumplimiento de los objetivos de esta relacionados con el Plan de Mejora. Para ello, se establecen las características de este sistema, así como sus requerimientos y experiencia de uso en diferentes organismos e instituciones ecuatorianas. Como resultado, se obtiene un análisis desde el punto de vista estructural y financiero. (Freire, 2014)

En la investigación titulada “Gestión Documental en la Empresa Servicios Integrados de Limpieza, 2018”, Cabrera y Marlon, se basan en dar solución a la mejora de la gestión documental de los contratos de trabajadores con base en la creación de un software Web en Java que permite firmar digitalmente los documentos, empleando un sintagma holístico. La investigación presenta un enfoque mixto ya que combina la técnica cuantitativa con la técnica cualitativa, con una población de 43 empleados de la empresa servicios integrados de limpieza.

Entre los resultados esperados con la ayuda del software propuesto está la mejora de la gestión documental, ya que gracias a este software se podrá automatizar y mejorar el tiempo de respuesta de los documentos de los contratos de trabajadores, también se podrá obtener los documentos en línea con información verídica que los usuarios pueden tomar y utilizar a su beneficio. Así mismo, gracias al software SistWebDigit se podrá firmar los documentos digitalmente, esto conllevará a que se pueda firmar desde cualquier lugar donde se encuentre la persona (Gerente). (Cabrera & Marlon, 2018)

2.1.2. Antecedentes Nacionales

En el trabajo titulado “Software para Gestión Documental, un Componente Modular del Sistema de Gestión de Seguridad de la Información (SGSI)” por Martelo, Madera y Betín, el objetivo principal consiste en desarrollar un software para contribuir al control de los documentos generados a partir del proceso de implantación de un Sistema de Gestión de Seguridad de la Información (SGSI). Dicho software permite recepcionar, administrar y organizar la documentación generada en el proceso de implantación del SGSI. Para soportar dicho software, se diseña e implementa un modelo que define acciones de gestión necesarias para la aprobación, revisión, actualización, estados y legibilidad en documentos durante el ciclo de vida del SGSI. Lo anterior, produjo como resultado un módulo para gestión documental que permite el control de documentos durante el proceso de implantación de un SGSI, trabajando bajo procedimientos del estándar ISO 27001. (Martelo et al., 2015)

El siguiente trabajo “Diseño de un modelo de gestión documental para industrias Haceb S.A para su posterior automatización a través de un software de gestión” por Montaña Posada, consiste en el diseño de un modelo de gestión documental para INDUSTRIAS HACEB S.A que permita su automatización a través de un software de gestión. El proyecto fue no experimental,

transaccional, descriptivo, y tiene como objetivo que la organización pueda implementar un modelo de gestión documental, ágil y simple que le permita obtener mejoras a nivel de procesos, prevenir posibles no conformidades en las acreditaciones de alta calidad y asegurar la preservación del conocimiento de la organización materializado en información documentada de alta calidad. Así pues, se realizó diagnóstico del estado de la gestión documental de la compañía, donde se lograron detectar las principales debilidades en cuanto a la información documentada, siendo la principal debilidad la falta de una cabeza visible responsable de gestionar y controlar la gestión documental y se lograron establecer los principales requerimientos a cumplir con el modelo a proponer, siendo el más importante el cumplimiento de las normas ISO 9001, ISO 14001, ISO 18001 y BASC en cuanto a información documentada. Así, con la información anterior se diseñó el modelo de gestión documental de industrias HACEB S.A, con su respectivo plan de implementación y su mejora mediante la implementación de una herramienta de automatización. (Montaña Posada, 2018)

2.1.3. Antecedentes Regionales

El presente proyecto titulado “Sistematización del Proceso de Prestación del Servicio de Laboratorios de la Facultad de Ingenierías y Arquitectura de la Universidad de Pamplona” por Carvajal Suarez, Clemen y Torres Chávez, consiste en realizar la sistematización de la prestación del servicio de los laboratorios de la Facultad de Ingeniería y Arquitectura de la Universidad de Pamplona. Se pretende mediante la implementación de tecnologías de la información: reducir la utilización del papel, mejorar la prestación de servicios por parte de los laboratorios de la facultad de ingeniería y arquitectura, mantener un control de asistencia por parte de profesores y estudiantes y optimizar la administración de los recursos físicos, materiales, equipos e insumos de laboratorio. Para sistematizar la prestación del servicio de laboratorios primero se analizó la

documentación (procedimientos, formatos e indicadores) de los laboratorios, información obtenida de la descripción del Sistema Integrado de Gestión de Calidad; Segundo se estableció un mecanismo de sistematización de la información a través del diseño y elaboración del software anexo a la plataforma universitaria, mediante el lenguaje de programación JAVA y base de datos MySQL (phpMyAdmin), tercero se seleccionó el sistema de control de usuario del servicio prestado de laboratorio utilizando Lectores Biométricos de Huella y por último se Validó para colocar en funcionamiento del sistema en las diferentes unidades de laboratorio. (Carvajal Suarez et al., 2017)

En el trabajo titulado “Aplicativo Web de Gestión Documental de Procesos Judiciales para la Organización, Accesibilidad y Seguridad de los Expedientes y Anexos de Casos Jurídicos” Reyes, Buitrago, Arévalo y Salazar nos mencionan que, conocer los aspectos primordiales que se llevan a cabo en la defensa de un caso penal por parte de un abogado, los hizo comprender las dificultades más comunes que se le pueden presentar a un profesional de esta área. Su aplicación web y móvil permitirá generar un impacto tecnológico y social en la ciudad de Cúcuta a los defensores públicos y clientes de la región permitiendo la organización, accesibilidad y seguridad de sus documentos, expedientes y gestión de clientes a los cuales responde. (Reyes Vargas et al., 2019)

2.2. Base Teórica

2.2.1. ISO/IEC 29110

La ISO / IEC 29110 es una nueva serie de normas e informes técnicos que llevan como título “Ingeniería de Software – Perfiles de Ciclo de Vida para Pequeñas Organizaciones y ha sido desarrollada para entidades muy pequeñas. Una PE’s (VSE por sus siglas en inglés – Very Small Entities) se define como una entidad (empresa, organización, departamento o proyecto)

que tiene menos de 25 personas. La mayoría de las PYMES de software pertenecen a la categoría VSE.

La ISO/IEC 29110 ha sido desarrollada para mejorar la calidad del producto y/o servicio de software, y para mejorar el desempeño de la organización, sin pretender excluir el uso de diferentes metodologías de Ciclo de Vida tales como: Cascada, Iterativo, Incremental, Evolutivo o Ágil.

Características de la Norma ISO/IEC 29110

- Modelo Internacional basado en una combinación de estándares (ISO/IEC 12207, ISO/IEC 15289, ISO/IEC 15504, entre otras)
- Es reconocido Internacionalmente
- Es específico para el desarrollo y mantenimiento de software
- Dirigido a Pequeñas Organizaciones de entre 1 a 25 personas
- Orientado a Proyectos tanto “Internos” como “Externos»
- Se obtiene un Certificado de Conformidad con vigencia de 3 años y vigilancias anuales. (nycecolombia.co, 2015)

2.2.2. Sistema de Gestión Documental

Un Sistema o programa de gestión documental, se puede definir como el conjunto de instrucciones en las que se detallan las operaciones para el desarrollo de los procesos de la gestión documental al interior de cada entidad, tales como producción, recepción, distribución, trámite, organización, consulta, conservación y disposición final de los documentos. Con la implantación de un sistema o programa de gestión documental se pretende alcanzar los siguientes objetivos:

- Normalizar la utilización de materiales, soportes y equipos de calidad y que a la vez preserven el cuidado del medio ambiente.
- Lograr una acertada normalización en los procedimientos para el recibo, radicación y distribución de la correspondencia mediante la utilización de sistemas eficientes de correo y mensajería.
- Regular el manejo y organización del sistema de administración de documentos y archivos a partir de la noción de Archivo Total y los enunciados de finalidad, responsabilidad, confidencialidad, seguridad y accesibilidad.
- Implementar el desarrollo de procesos básicos de aplicación de la tabla de Retención Documental, organización, transferencias primarias, recuperación, preservación, conservación de la información y disposición final de los documentos.
- Facilitar la recuperación de la información en forma rápida y oportuna.
- Encaminar los archivos para que sean verdaderos centros de información, útiles para la administración e importantes para la cultura. (Berdugo Baez, 2016)

2.2.3. Radicación de documentos

Según el Acuerdo 060 de 2001, artículo 2, las comunicaciones Oficiales “Son todas aquellas recibidas o producidas en desarrollo de las funciones asignadas legalmente a una entidad, independientemente del medio utilizado”, mediante las cuales se da inicio, continuidad y/o conclusión a un trámite como se muestra en la ilustración 1, y establece que la radicación de comunicaciones oficiales “es el procedimiento por medio del cual, las entidades asignan un número consecutivo, a las comunicaciones recibidas o producidas, dejando constancia de la fecha y hora de recibo o de envío, con el propósito de oficializar su trámite y cumplir con los

términos de vencimiento que establezca la Ley. Estos términos, se empiezan a contar a partir del día siguiente de radicado el documento”. (Tovar Gonzales, 2021)

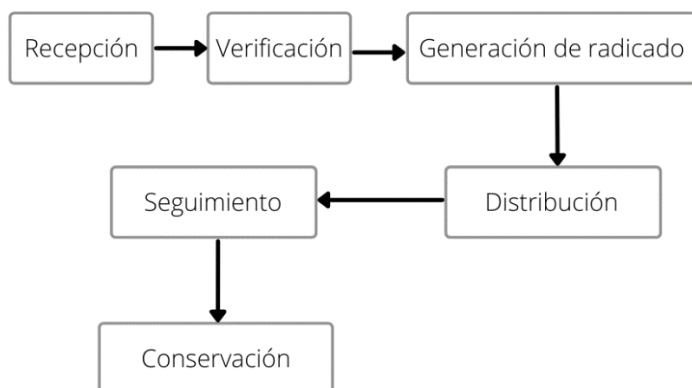
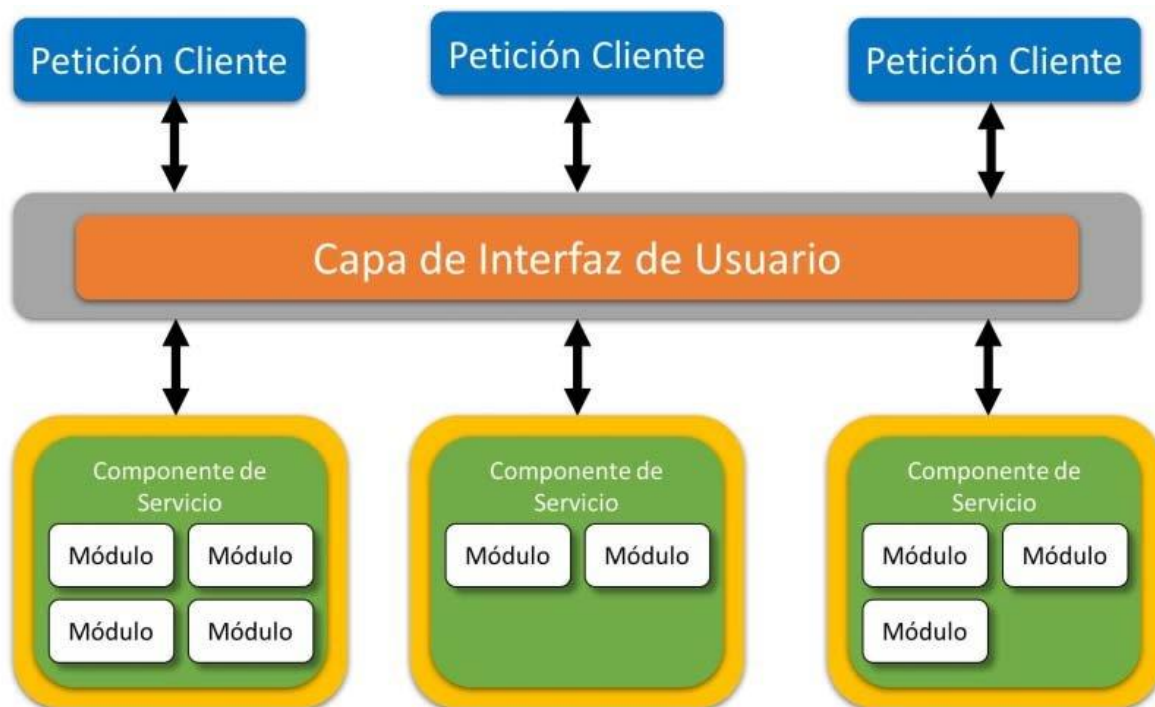


Ilustración 1 Radicación de documentos

2.2.4. Microservicios

Es una arquitectura orientada a la infraestructura y aporta un enfoque para el desarrollo de aplicaciones únicas como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto (HTTP) como se puede observar en la ilustración 2. Estos servicios están contruidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos. Una arquitectura de microservicios promueve el desarrollo y despliegue de aplicaciones compuestas por unidades independientes, autónomas, modulares y auto- contenidas, lo cual difiere de la forma tradicional o monolítico. (López & Maya, 2017)



*Ilustración 2 Patrón básico de Arquitectura de Microservicios.
Tomada de https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FFigura-3-Patron-basico-de-arquitectura-de-Microservicios-Tomado-de-Arquitectura-de_fig2_351366178&psig=AOvVaw2e_*

Una de las ventajas de utilizar microservicios es la capacidad de publicar una aplicación grande como un conjunto de pequeñas aplicaciones (microservicios) que se pueden desarrollar, desplegar, escalar, manejar y visualizar de forma independiente como se muestra en la ilustración 3. Los microservicios permiten a las empresas gestionar las aplicaciones de código base grande usando una metodología más práctica donde las mejoras incrementales son ejecutadas por pequeños equipos en bases de código y despliegues independientes. La agilidad, reducción de costes y la escalabilidad granular, traen algunos retos de los sistemas distribuidos y las prácticas de gestión de los equipos de desarrollo que deben ser considerados. (López & Maya, 2017)

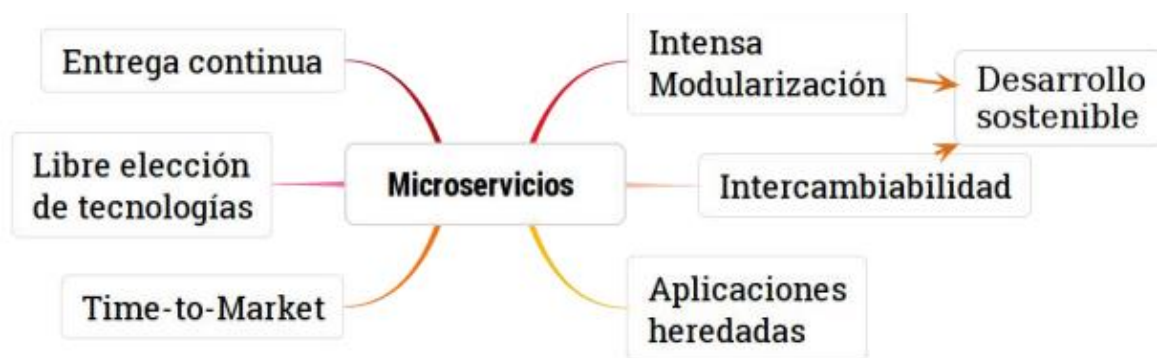


Ilustración 3 Beneficios de los microservicios.

Tomada de

<https://dspace.redclara.net/bitstream/10786/1277/1/93%20Arquitectura%20de%20Software%20basada%20en%20Microservicios%20para%20Desarrollo%20de%20Aplicaciones%20Web.pdf>

2.2.5. API REST

API es un acrónimo del término inglés Application Programming Interface (en español interfaz de programación de aplicaciones). Su principal característica es que permiten que sus productos se comuniquen con otros sin que estos conozcan su implementación. De esta manera se simplifican los costes y programación y se reducen los tiempos. Su funcionamiento se considera que es parecido a un contrato, ya que, si el emisor envía una solicitud con una estructura conocida, el receptor podrá realizar la operación pertinente de forma adecuada y devolver una respuesta con también un cierto formato establecido. REST es un acrónimo de Representational State Transfer (transferencia de representación de estado en español). Su característica principal es que no tiene estado (stateless) lo que significa que entre dos llamadas el servicio pierde todos los datos ya que el estado lo tiene que mantener el cliente, de aquí su nombre. Si se necesita que un servicio REST recuerde cierta información, es necesario identificarse en cada llamada, ya sea por un usuario y contraseña, un token u otro tipo de credenciales. (Maza Valles, 2020)

Un servicio REST se define como una agregación de diferentes recursos que pueden ser alcanzados desde un identificador universal de recurso (URI) base. Un recurso representa a una entidad del mundo real cuyo estado está expuesto y puede cambiarse accediendo a un URI. Una Representación es la descripción de los mensajes enviados o recibidos de un Recurso en términos de un lenguaje tecnológico. Actualmente XML y JSON son los idiomas más populares para describir estos mensajes. (López & Maya, 2017)

2.2.6. Arquitectura hexagonal

Es un patrón estructural para diseñar software. La idea detrás de esta es establecer entradas y salidas en los bordes del diseño. La arquitectura hexagonal fue un cambio con respecto a la arquitectura típica en capas, donde es posible usar la inyección de dependencia y otras técnicas para posibilitar pruebas sobre esta. Según Cockburn, la arquitectura hexagonal consiste en el modelo de dominio, los servicios de aplicación y los puertos con los adaptadores. Cada lado del hexágono representa un puerto concreto, aunque en la práctica podría haber más puertos distintos junto con su correspondiente adaptador. Basándose en el principio de inversión de dependencia, el modelo de dominio como núcleo y por lo tanto la lógica de negocio es independiente de los servicios de aplicación y adaptadores que lo rodean, esto simplifica el traspaso o cambio de decisiones tecnológicas. Alrededor de la lógica de negocio (el hexágono) debe estar libre de cuestiones de tecnología, solo el exterior del hexágono habla con el interior mediante interfaces, llamadas puertos. (Cambarieri et al., 2020)

2.2.7. Arquitectura MVP

La arquitectura de software Modelo-Vista-Presentador (MVP de ahora en adelante fue ideada para realizar software con interfaces gráficas, ya que el procesamiento interno de la aplicación se encuentra desacoplada de la interfaz de usuario. La Vista y el Modelo están separados, mientras que el Presentador transmite la información entre ellos (ilustración 15). El Modelo es el componente encargado de almacenar la información y realizar el procesamiento de datos requerido por el usuario a partir de la Vista. La Vista es la interfaz de usuario, la cual se encarga de exhibir datos y enviar señales al Presentador (a partir de la interacción del usuario) para que se las transmita al Modelo y realice las tareas correspondientes. Luego el Modelo definirá las tareas que se ejecutarán en el programa, se las enviará al Presentador y serán exhibidas por la Vista. (A. G. Bonelli Toro et al., 2019)

2.3. Marco Legal

2.3.1. Ley estatutaria 1581 de 2012, Por la cual se dictan disposiciones generales para la protección de datos personales

La presente ley tiene por objeto desarrollar el derecho constitucional que tienen todas las personas a conocer, actualizar y rectificar las informaciones que se hayan recogido sobre ellas en bases de datos o archivos, y los demás derechos, libertades y garantías constitucionales a que se refiere el artículo 15 de la Constitución Política; así como el derecho a la información consagrado en el artículo 20 de la misma. (funcionpublica.gov.co, 2021)

2.3.2. Ley 594 de 2000, Por medio de la cual se dicta la Ley General de Archivos y se dictan otras disposiciones

La presente ley tiene por objeto establecer las reglas y principios generales que regulan la función archivística del Estado. La presente ley comprende a la administración pública en sus diferentes niveles, las entidades privadas que cumplen funciones públicas y los demás organismos regulados por la presente ley. (normativa.archivogeneral.gov.co, 2018)

2.3.3. Ley 23 de 1982, sobre derechos de autor

Los autores de obras literarias, científicas y artísticas gozarán de protección para sus obras en la forma prescrita por la presente Ley y, en cuanto fuere compatible con ella, por el derecho común. También protege esta Ley a los intérpretes o ejecutantes, a los productores de programas y a los organismos de radiodifusión, en sus derechos conexos a los del autor. (defensoria.gov.co, 2021)

2.3.4. Decreto 2106 de 2019, Por el cual se dictan normas para simplificar, suprimir y reformar trámites, procesos y procedimientos innecesarios existentes en la administración pública”

El presente decreto tiene por objeto simplificar, suprimir y reformar trámites, procesos y procedimientos innecesarios existentes en la Administración Pública, bajo los principios constitucionales y legales que rigen la función pública, con el propósito de garantizar la efectividad de los principios, derechos y deberes de las personas consagrados en la Constitución mediante trámites, procesos y procedimientos administrativos sencillos, ágiles, coordinados, modernos y digitales. (funcionpublica.gov.co, 2019)

2.3.5. Circular externa 005 de 2012

El propósito de la Iniciativa de Cero Papel es mejorar la eficiencia del Estado, generando economía en los procesos administrativos y fortaleciendo la transparencia de la administración pública, a través de la transición hacia una gestión electrónica de documentos, es decir la gestión de documentos nacidos electrónicamente. (normativa.archivogeneral.gov.co, 2012)

2.4. Ope Colombia

OPE Colombia, Outsourcing de Procesos Empresariales SAS, es una compañía cucuteña que inició sus actividades en el año 2008, constituida especialmente como un proveedor tercerizado para prestarle a las entidades, públicas o privadas, el servicio de Business Process Outsourcing (BPO) y cuya misión es ayudar a reducir los costos asociados con los procesos de soportes y la optimización de sus recursos. (opecolombia.com, 2022)

2.4.1. Productos de software

SIEPDOC es un software de Gestión Documental y de Procesos, que permite gestionar electrónicamente la producción, el trámite, el almacenamiento digital y la recuperación de documentos, evitando su manejo en papel, garantizando la seguridad de la información y la trazabilidad de cualquier proceso que se implemente mediante su funcionalidad. Entre los beneficios de SIEPDOC se encuentra la consulta fácil y rápida de documentos, se minimiza el riesgo de pérdida de documentos, control de tiempo de documentos en poder del responsable, centralización de documentos, confidencialidad de los documentos, disminución del impacto ambiental al reducir el consumo de papel, entre otros. (opecolombia.com, 2022)

El software SIEP está enfocado en procesos, subprocesos, actividades y tareas, los cuales pueden ser adquiridos en su totalidad o de forma separada. Le permite a la entidad gestionar y administrar los procesos de Planeación, Contratación, suministros, Presupuesto, Recurso

Humano, Tesorería y Contabilidad. Entre los beneficios de SIEP se encuentra la agilidad para la captura de datos, planeación de actividades de mantenimiento, reducción de inventarios, identificación de activos fijos, detección de pérdida de activos, fácil administración de bienes, entre otros. (opecolombia.com, 2022)

3 Desarrollo del Sistema

3.1. Herramientas de desarrollo

3.1.1. *Java*

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. (java.com, 2021)

Java es uno de los lenguajes de programación más utilizados a nivel mundial, de hecho, uno de los principales beneficios de desarrollar en Java es que grandes empresas como Google, Amazon, Uber, empresas financieras o de banca lo prefieren por lo seguro, portable y mantenible. (platzi.com, 2021)

3.1.2. *Vaadin Flow*

Vaadin Flow es un marco de interfaz de usuario de Java con componentes web de JavaScript, con el cual se pueden crear aplicaciones web en Java controladas por servidor, o en JavaScript utilizando los componentes web de Vaadin con cualquier BackEnd. Este marco de trabajo viene con un gran conjunto de componentes de interfaz de usuario prediseñados, que se pueden utilizar en componentes de JavaScript, tanto a través de la API de Java como en JavaScript. Además, se pueden combinar para crear interfaces de usuario complejas y agregar nuevas funcionalidades. Adicionalmente, Vaadin proporciona acceso completo al DOM, en Java incluso desde el lado del servidor y la flexibilidad se extiende a la pila de programación, optando por escribir la interfaz de usuario en Java, TypeScript, JavaScript o cualquier combinación de

ellos. Las mismas funciones están disponibles independientemente del idioma que se utilice.
(vaadin.com, 2021)

3.1.3. Spring Boot

Spring hace que la creación de aplicaciones web sea rápida y sin complicaciones. Al eliminar gran parte del código repetitivo y la configuración asociados con el desarrollo web, obteniendo un modelo de programación web moderno que agiliza el desarrollo de aplicaciones HTML del lado del servidor, API REST y sistemas bidireccionales basados en eventos.
(spring.io, 2021)

Spring Boot es una extensión de Spring Framework que sigue el enfoque de "Convención sobre configuración", que ayuda a construir aplicaciones basadas en Spring de manera rápida y fácil. El objetivo principal de Spring Boot es crear rápidamente aplicaciones basadas en Spring sin requerir que los desarrolladores escriban la misma configuración repetitiva una y otra vez. Spring Boot está construido sobre Spring Framework y forma parte integral de éste. Actualmente, es el más utilizado dentro del conjunto de Frameworks, al igual que dentro del mundo Java. El objetivo principal de Spring Boot es simplificar el desarrollo de aplicaciones por medio de la autogestión de un gran número de configuraciones, tareas y componentes que son necesarios para la ejecución de una aplicación. De esta manera, se logra que los desarrolladores se enfoquen en el desarrollo de la lógica de negocio del sistema. (Ramírez Pérez, 2020)

3.1.4. Eclipse IDE

Es un entorno de desarrollo de software de código abierto, multiplataforma y compuesto por un amplio conjunto de herramientas Open Source para la construcción de aplicaciones de cliente enriquecido [...] y de entornos de desarrollo integrado (IDE). Eclipse soporta la combinación de diferentes lenguajes de programación de manera sencilla, proporcionando un

IDE para cada uno de ellos. Es habitual que los desarrolladores y usuarios que hacen uso de esta plataforma lo hagan utilizando el entorno de desarrollo Java. (Abad Gómez-Pastrana, 2018)

3.1.5. PostgreSQL

PostgreSQL, o también conocido como Postgres, es un sistema de gestión de bases de datos relacionales (RDBMS) libre y de código abierto (Open Source) que hace énfasis en la extensibilidad y el cumplimiento de SQL. PgAdmin es la herramienta oficial para administrar bases de datos en PostgreSQL. Permite desde hacer búsquedas SQL hasta desarrollar toda nuestra base de datos de forma muy fácil e intuitiva; directamente desde la interfaz gráfica. (Cesar, 2017)

PostgreSQL destaca por su amplísima lista de prestaciones que lo hacen capaz de competir con cualquier SGBD comercial:

- Está desarrollado en C, con herramientas como Yacc y Lex
- La API de acceso al SGBD se encuentra disponible en C, C++, Java, Perl, PHP, Python y TCL, entre otros.
- Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario.
- Su administración se basa en usuarios y privilegios.
- Sus opciones de conectividad abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC.
- Los mensajes de error pueden estar en español y hacer ordenaciones correctas con palabras acentuadas o con la letra ‘ñ’.
- Es altamente confiable en cuanto a estabilidad se refiere. (Ginestà Gibert & Pérez Mora, 2012)

3.1.6. MyBatis

MyBatis es un framework de persistencia que soporta SQL, procedimientos almacenados y mapeos avanzados. MyBatis elimina casi todo el código JDBC, el establecimiento manual de parámetros y la obtención de resultados. MyBatis puede configurarse con XML o anotaciones y permite mapear tipos de datos primitivos, objetos de tipo Map y POJOs (Plain Old Java Objects) a registros de base de datos. (*MyBatis – MyBatis 3 | Introducción*, 2022)

3.1.7. MinIO

MinIO es una solución de almacenamiento de objetos (como fotos, videos, archivos o imágenes) de alto rendimiento que proporciona una API compatible con Amazon Web Services S3 y es compatible con todas las funciones principales de S3. MinIO está diseñado para implementarse en cualquier lugar: nube pública o privada, infraestructura baremetal, entornos orquestados e infraestructura perimetral. (min.io, 2022)

3.1.8. RabbitMQ

RabbitMQ es un corredor de mensajería, un intermediario para la mensajería. Brinda a sus aplicaciones una plataforma común para enviar y recibir mensajes, y a sus mensajes un lugar seguro para vivir hasta que se reciban. Las aplicaciones pueden conectarse entre sí, como componentes de una aplicación más grande, o con los dispositivos y datos de los usuarios. La mensajería es asíncrona y desacopla las aplicaciones al separar el envío y la recepción de datos. (rabbitmq.com, 2022)

En RabbitMQ se crean colas que almacenan mensajes, los cuales son enviados por componentes de software llamados productores a otros llamados consumidores quienes reciben y procesan estos mensajes.

3.1.9. Docker

Docker es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Docker permite separar las aplicaciones de la infraestructura para que se pueda entregar software rápidamente. Con Docker, se puede administrar la infraestructura de la misma manera que se administra las aplicaciones. Al aprovechar las metodologías de Docker para enviar, probar e implementar código rápidamente, se puede reducir significativamente la demora entre escribir el código y ejecutarlo en producción.

Docker brinda la capacidad de empaquetar y ejecutar una aplicación en un entorno vagamente aislado llamado contenedor. El aislamiento y la seguridad permiten ejecutar muchos contenedores simultáneamente en un host determinado. Los contenedores son livianos y contienen todo lo necesario para ejecutar la aplicación, por lo que no se necesita depender de lo que está instalado actualmente en el host. (docs.docker.com, 2022a)

3.1.10. Docker Compose

Docker Compose es una herramienta para ayudar a definir y compartir aplicaciones de varios contenedores. Con Compose, se puede crear un archivo YAML para definir los servicios y, con un solo comando, podemos activar o desactivar todo. (docs.docker.com, 2022b)

3.1.11. Jenkins

Jenkins es un servidor de automatización autónomo de código abierto que se puede utilizar para automatizar todo tipo de tareas relacionadas con la creación, prueba y entrega o

implementación de software. Nos ayuda a acelerar el proceso de desarrollo y entrega de software a través de la automatización. (jenkins.io, 2022)

3.2. Metodología

3.2.1. SCRUM

Scrum es un marco ligero que ayuda a las personas, los equipos y las organizaciones a generar valor a través de soluciones adaptables para problemas complejos. Los co-creadores de Scrum, Ken Schwaber y Jeff Sutherland, han escrito The Scrum Guide para explicar Scrum de manera clara y sucinta. Esta guía contiene la definición de Scrum. Esta definición consta de las responsabilidades, los eventos, los artefactos y las reglas que los unen de Scrum. En pocas palabras, Scrum requiere un Scrum Master para fomentar un entorno donde:

1. Un Product Owner ordena el trabajo de un problema complejo en un Product Backlog.
2. El Equipo Scrum convierte una selección del trabajo en un Incremento de valor durante un Sprint.
3. El equipo Scrum y sus partes interesadas inspeccionan los resultados y se ajustan para el próximo Sprint. (Scrum.org, 2021)

3.2.1.1. Roles

En los roles principales de Scrum mencionados anteriormente encontramos que el Product Owner se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner ayuda al usuario a escribir las historias de usuario, las prioriza, y las coloca en el Product Backlog. Representa a los stakeholders (interesados externos o internos).

El Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se auto organizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan.

El equipo de desarrollo tiene la responsabilidad de entregar el producto. Es recomendable un pequeño equipo de 5 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc.).

Los roles designados para este proyecto son Product Owner, Líder de Proyecto o Scrum Master, Líder Técnico, Tester, Equipo de Desarrollo.

3.2.1.2. Documentos SCRUM o Artefactos

El **Product Backlog** o pila del producto se trata como un documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos del proyecto priorizados que definen el trabajo a realizar (PBI, Product Backlog Item), el cual contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno sobre la inversión (ROI). Representa el *qué* va a ser construido en su totalidad. Es abierto y solo puede ser modificado por el *product owner*. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido. Esta estimación ayuda al *product owner* a ajustar la línea temporal (KEV) y, de manera limitada, la prioridad de las diferentes tareas. Por ejemplo, si dos características tienen el mismo valor de negocio la que requiera menor tiempo de desarrollo tendrá probablemente más prioridad, debido a que su ROI será más alto. En el anexo 1 de este documento se puede observar el product backlog.

El **Sprint Backlog** o pila de sprint es el subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe el *cómo* el equipo va a implementar los requisitos durante el sprint. Por lo general los requisitos se subdividen en tareas, a las cuales se asignan ciertas *horas de trabajo*, pero ninguna tarea con una duración superior a 16 horas. Si una tarea es mayor de 16 horas, deberá ser dividida en otras menores. Las tareas en el *sprint backlog* nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca adecuado. Para OPE Colombia el sprint backlog son las historias de usuario, cada ítem de backlog es una historia de usuario y la asignación se encuentra en SIEP Helpdesk, que es un software de la empresa que se utiliza para este fin.

El **Incremento** es una parte del producto desarrollada en un sprint como, por ejemplo, pruebas o codificación, o cualquier artefacto generado. Son los avances del sprint backlog.

3.2.1.3. Eventos

El **Sprint Planning** o planificación del sprint es una jornada de trabajo dedicada a explicarse los objetivos y prioridades del producto por el product owner (dueño del producto). Aquí se define el sprint backlog.

El **Sprint** es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los Sprint sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo; la magnitud es definida por el equipo y debe ser lo más corta posible. Al final de cada sprint, el equipo deberá presentar los avances logrados (ítems de sprint backlog), y el resultado obtenido es un producto que, potencialmente, se puede entregar al cliente (generar un incremento del sistema). Así mismo, se recomienda no agregar objetivos al sprint o *sprint backlog* a menos que su falta amenace al éxito del proyecto. La constancia permite la

concentración y mejora la productividad del equipo de trabajo. El tiempo mínimo de un Sprint es de dos (2) semanas y el máximo es de cuatro (4) semanas. En OPE Colombia los Sprint tienen una duración de 4 semanas.

El **Daily Scrum** o Scrum diario es una reunión de 15 minutos máximo realizada todos los días al inicio de la jornada laboral. En esta reunión cada miembro del equipo de desarrollo expone las actividades que realizó el día anterior, las actividades que va a realizar durante el día y los problemas presentes en dichas actividades. En el anexo 2 de este documento se puede observar la estructura que se maneja en OPE Colombia para el documento del Daily Scrum y su contenido durante el presente trabajo.

El **Sprint Review** o revisión del sprint es una reunión realizada al finalizar el sprint donde se presentan los incrementos. OPE Colombia realiza esta reunión el último sábado de cada mes luego del Scrum diario.

El **Sprint Retrospective** o retrospectiva es una reunión donde se identifican las fortalezas y debilidades durante el sprint, se habla de lo que se hizo bien y lo que se hizo mal, y de las cosas que se deben mejorar en el siguiente sprint. También se evalúa el cumplimiento de los objetivos del sprint anterior y se plantean los objetivos del siguiente sprint. OPE Colombia realiza la retrospectiva el último sábado de cada mes luego del Scrum diario y el Sprint Review. La retrospectiva de este proyecto se puede observar en el documento anexo “Anexo 2. Scrum diario”, en la hoja ‘Retrospectiva’.

3.2.1.4. Historias de usuario

Una **historia de usuario** es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario acompañadas de las discusiones con los usuarios y las pruebas de validación son utilizadas en las metodologías ágiles para la

especificación de requisitos y son una forma rápida de administrarlos sin tener que elaborar gran cantidad de documentos formales. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes. Las historias de usuario del presente proyecto se evidencian en el subcapítulo Análisis de Requerimientos funcionales y no Funcionales.

3.2.1.5. Hitos

Un hito es una tarea que simboliza el haber conseguido un logro importante en el proyecto. Los hitos son una forma de conocer el avance del proyecto sin estar familiarizado con el proyecto y constituyen un trabajo de duración cero porque simbolizan un logro, un punto, un momento en el proyecto. En el cronograma de un proyecto deben existir varios hitos que informen la fecha estimada en que se piensa cumplirlos, y que luego en la ejecución se compara con la fecha real. En resumen, los hitos son fechas estimadas para determinadas actividades, estos forman parte del Sprint Review. La reunión y revisión de los hitos en OPE Colombia se realiza cada sábado luego de la reunión del Scrum diario, los hitos del presente proyecto se evidencian en el anexo 3 de este documento.

3.3. Análisis de Requerimientos Funcionales y no Funcionales

En esta sección se presenta la identificación y detalle de los requerimientos funcionales (historias de usuario) y no funcionales, así como el diagrama de casos de uso que nos permite ver las relaciones entre las historias de usuario.

3.3.1. Diagramas de casos de uso

Un diagrama de casos de uso es una herramienta muy útil que permite expresar las funcionalidades de una aplicación y el comportamiento de estas, y un caso de uso es un requisito que puede ser expresado en estos diagramas. A continuación, se presentan los diagramas de casos de uso para los módulos desarrollados en el presente trabajo.

En la ilustración 4 se puede observar el diagrama de caso de uso Iniciar Sesión, allí se puede apreciar que el funcionario tiene asociación directa con el caso de uso Iniciar Sesión, y este tiene relación de extensión con el caso de uso Reestablecer contraseña, es decir, es opcional.

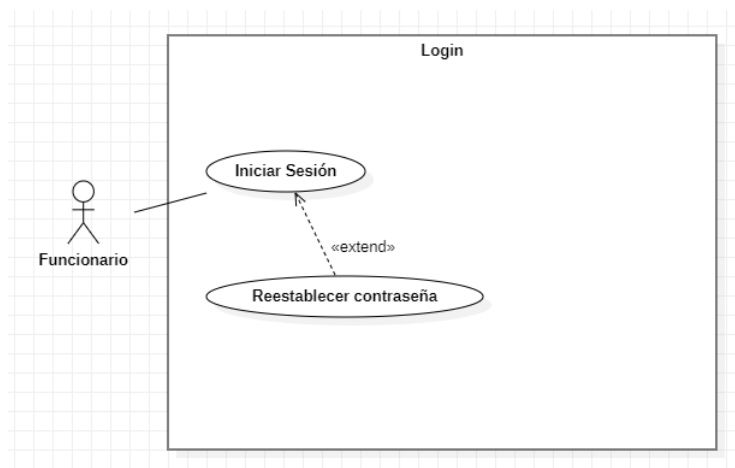


Ilustración 4 Diagrama de Caso de uso Iniciar Sesión

En la ilustración 5 se puede observar el diagrama de caso de uso Generar radicado de salida, allí se puede apreciar que el funcionario tiene asociación directa con el caso de uso Generar radicado de salida, y este tiene relación de extensión con las diferentes funcionalidades que se habilitan luego de generar el radicado, es decir, son opcionales. También tiene relación de inclusión con el caso de uso Iniciar Sesión, ya que el funcionario debe estar logueado para radicar.

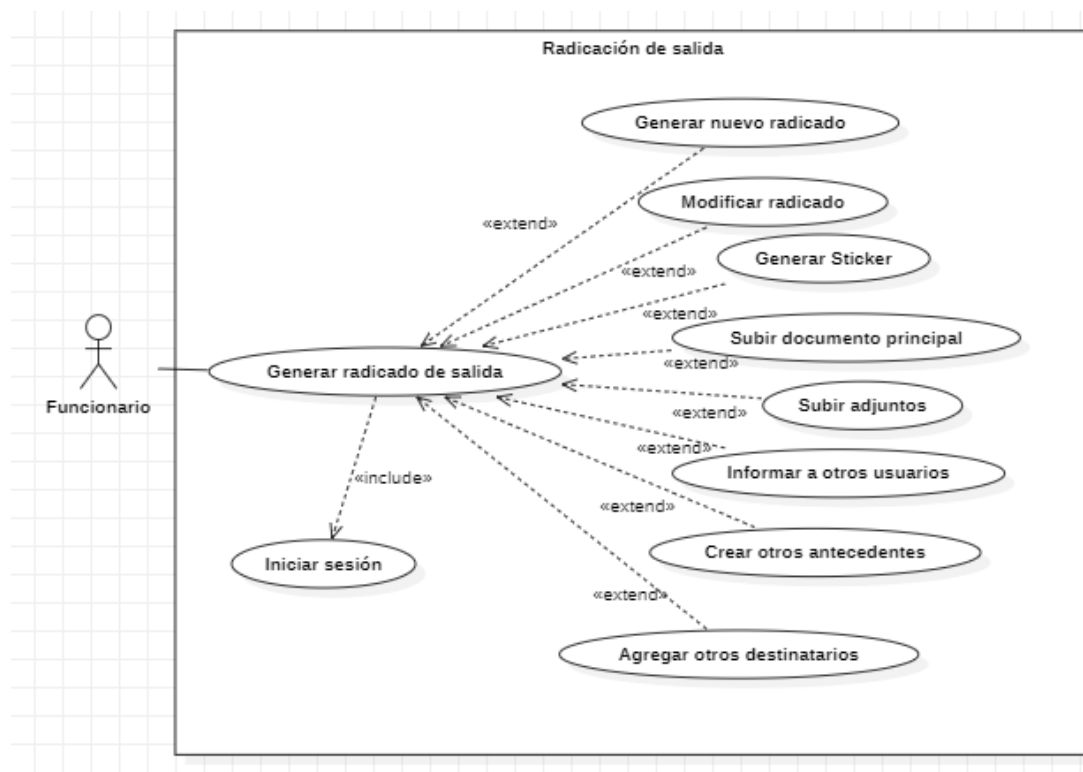


Ilustración 5 Diagrama de Caso de uso Generar radicado de salida

En la ilustración 6 se puede observar el diagrama de caso de uso Generar radicado de entrada, como se puede ver el funcionario tiene asociación directa con el caso de uso Generar radicado de entrada, y este tiene relación de extensión con las diferentes funcionalidades que se habilitan luego de generar el radicado, es decir, son opcionales. También tiene relación de inclusión con el caso de uso Iniciar Sesión, ya que el funcionario debe estar logueado para radicar.

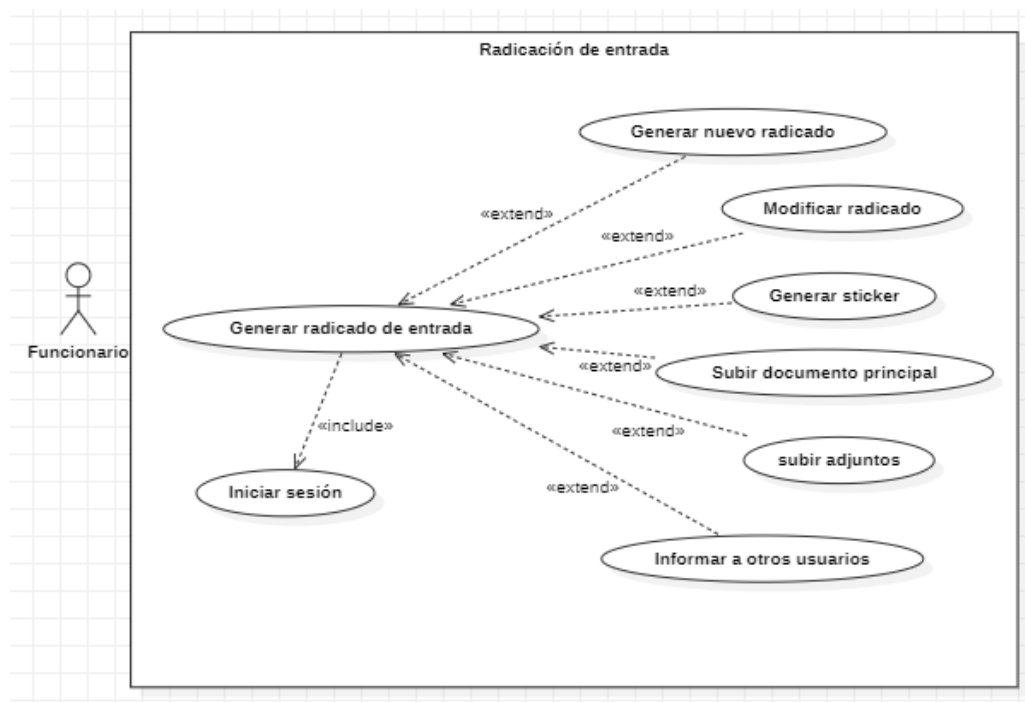


Ilustración 6 Diagrama de Caso de uso Generar radicado de entrada

En la ilustración 7 se puede apreciar el diagrama de caso de uso para los CRUD de tablas básicas, puede podemos observar el funcionario tiene asociación directa con los casos de uso CRUD, y estos tienen relación de inclusión con el caso de uso Iniciar Sesión, ya que el funcionario debe estar logueado para acceder a estas funcionalidades.

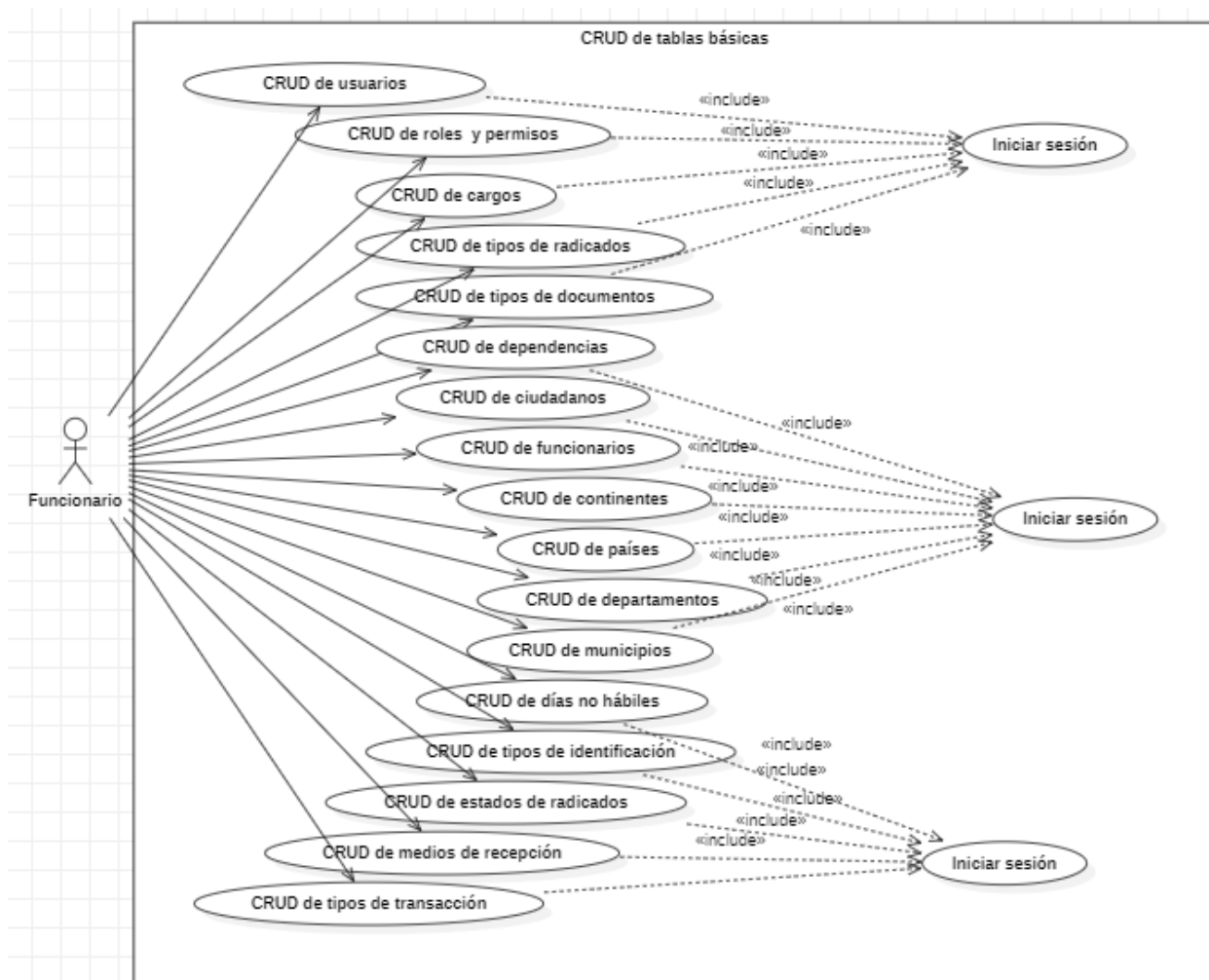


Ilustración 7 Diagrama de Casos de uso para CRUD de tablas básicas

En la ilustración 8 se puede observar el diagrama de caso de uso Generar reportes, como se puede apreciar el funcionario tiene asociación directa con el caso de uso Generar reportes que se expresa de forma generalizada ya que existen tres tipos de reportes, y estos tienen relación de inclusión con el caso de uso Iniciar Sesión, ya que el funcionario debe estar logueado para acceder a estas funcionalidades.

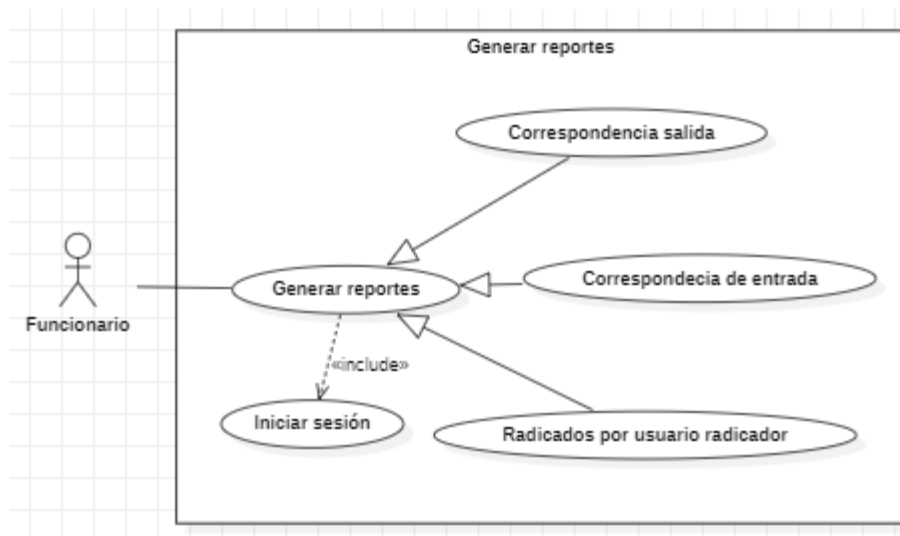


Ilustración 8 Diagrama de Caso de uso Generar reportes

3.3.2. Historias de usuario y criterios de aceptación

En esta sección se describen las historias de usuario junto a sus respectivos criterios de aceptación, este formato de tabla está basado en el documento de Historias de Usuario de la ISO 29110 que trabaja la empresa OPE Colombia.

Tabla 1 Login

Identificador (ID) de la historia	1
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito controlar el ingreso a la aplicación
Descripción razón o resultado	Con la finalidad de permitir el uso de la aplicación a usuarios autorizados
Número (#) de escenario	1
Criterio de aceptación (Título)	Iniciar sesión
Contexto	En caso de querer ingresar a la aplicación
Evento	Se ingresa usuario y contraseña, luego inicia sesión dando clic en el botón 'Iniciar Sesión'
Resultado / Comportamiento esperado	En caso de ser validos los datos se permite el ingreso a la aplicación con los permisos correspondientes del rol asignado al usuario en la aplicación
Número (#) de escenario	2
Criterio de aceptación (Título)	Validar usuario activo
Contexto	Al intentar ingresar en la aplicación, se valida si el usuario cumple con las condiciones para iniciar sesión

Evento	
Resultado / Comportamiento esperado	Si el usuario esta inactivo no se le permite el ingreso, si está activo se permite el ingreso. si la fecha de vencimiento del contrato ha vencido se pasa a estado 'Suspendido'.
Número (#) de escenario	3
Criterio de aceptación (Título)	Límite de intentos de inicio de sesión
Contexto	En caso de ingresar 3 veces con datos no válidos
Evento	Al dar clic 3 veces en el botón 'Iniciar Sesión' se bloquea al usuario
Resultado / Comportamiento esperado	La aplicación debe bloquear temporalmente el inicio de sesión al usuario y pasarlo a estado 'Suspendido', el usuario debe contactarse con el usuario administrador

Tabla 2 Cerrar sesión

Identificador (ID) de la historia	2
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito cerrar sesión en la aplicación
Descripción razón o resultado	Con la finalidad de controlar las sesiones en la aplicación
Número (#) de escenario	1
Criterio de aceptación (Título)	Cerrar sesión
Contexto	En caso de querer cerrar la sesión en la aplicación
Evento	Se da clic en la opción 'Cerrar Sesión' de la parte superior derecha de la pantalla, en las opciones del usuario
Resultado / Comportamiento esperado	La aplicación debe enviar al usuario a la página de inicio de sesión

Tabla 3 Reestablecer contraseña de usuario

Identificador (ID) de la historia	3
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito recuperar mi contraseña cuando la pierda
Descripción razón o resultado	Con la finalidad de recuperar la contraseña para el acceso a la aplicación
Número (#) de escenario	1
Criterio de aceptación (Título)	Recuperar contraseña vía correo electrónico
Contexto	En caso de olvidar o perder la contraseña
Evento	Se debe seleccionar la opción '¿Olvidaste tu contraseña?', ingresar su nombre de usuario y dar en el botón 'generar nueva clave'
Resultado / Comportamiento esperado	La aplicación debe cambiar el estado del usuario a 'verificando email' y enviar un correo con un link que envíe al usuario a registrar la nueva contraseña
Número (#) de escenario	2
Criterio de aceptación (Título)	Recuperar contraseña vía administrador del sistema
Contexto	En caso de no tener correo electrónico o haberlo perdido
Evento	El administrador del sistema puede ingresar al módulo configuración, opción administración, opción funcionarios, y

	editar el correo electrónico del funcionario para poder usar la función de recuperar contraseña.
Resultado / Comportamiento esperado	La aplicación debe actualizar el correo electrónico del usuario.

Tabla 4 CRUD de usuarios

Identificador (ID) de la historia	4
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito administrar los usuarios del sistema
Descripción razón o resultado	Con la finalidad de gestionar los usuarios
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de usuarios
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un usuario
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción usuarios y roles, pestaña usuarios
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar un usuario. los campos del CRUD de usuarios son: nombre de usuario, contraseña, dependencia, cargo, perfil (estándar o jefe), funcionario, Rol, estado, fecha vencimiento contrato, envío SMS, permitir SMS, nivel de seguridad, teléfono empresa, extensión teléfono, ubicación en la empresa, Radicación directa idioma.
Número (#) de escenario	2
Criterio de aceptación (Título)	Envío de correo para crear contraseña
Contexto	
Evento	El usuario administrador debe crear un usuario
Resultado / Comportamiento esperado	Una vez creado un usuario se debe enviar correo electrónico en el cual se envíe una URL donde el usuario pueda acceder a crear su nueva contraseña

Tabla 5 CRUD de Roles y permisos

Identificador (ID) de la historia	5
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito administrar los roles y permisos del sistema
Descripción razón o resultado	Con la finalidad de gestionar los roles y permisos
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de roles y permisos
Contexto	en caso de necesitar crear, consultar, actualizar o eliminar un rol y permisos
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción usuarios y roles, pestaña roles
Resultado / Comportamiento esperado	la aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar roles y permisos. los campos del CRUD de roles son: nombre, descripción. los permisos deben estar en categorías: permisos generales, permisos editor documentos

	internos, permisos editor respuesta salida, permisos de seguridad, permisos tipos de radicados, otros permisos especiales, permisos CRUD de tablas básicas de usuarios, permisos CRUD de tablas básicas de configuración, permisos CRUD de tablas básicas de gestión de documentos, permisos Estadísticas
--	---

Tabla 6 CRUD de funcionarios

Identificador (ID) de la historia	6
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito administrar los funcionarios del sistema
Descripción razón o resultado	Con la finalidad de gestionar los funcionarios
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de funcionarios
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un funcionario
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción funcionarios
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar funcionarios. Los campos del CRUD son: identificación, tipo de identificación, nombre completo, fecha de nacimiento, email, celular, estado

Tabla 7 CRUD de los tipos de identificación

Identificador (ID) de la historia	7
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito administrar los tipos de identificación del sistema
Descripción razón o resultado	Con la finalidad de gestionar los tipos de identificación
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los tipos de identificación
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un tipo de identificación
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción tipo de identificación
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de identificación. Los campos del CRUD son: nombre, estado

Tabla 8 CRUD de los continentes

Identificador (ID) de la historia	8
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los continentes
Descripción razón o resultado	Con la finalidad de gestionar los continentes
Número (#) de escenario	1

Criterio de aceptación (Título)	CRUD de los continentes
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un continente
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'continentes'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar continentes. Los campos del CRUD son: nombre, estado

Tabla 9 CRUD de los países

Identificador (ID) de la historia	9
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los países
Descripción razón o resultado	Con la finalidad de gestionar los países
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los países
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un país
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'países'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar países. Los campos del CRUD son: nombre, estado, continente

Tabla 10 CRUD de los departamentos

Identificador (ID) de la historia	10
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los departamentos
Descripción razón o resultado	Con la finalidad de gestionar los departamentos
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los departamentos
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un departamento
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'departamentos'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar departamentos. Los campos del CRUD son: código, nombre, estado, país

Tabla 11 CRUD de los municipios

Identificador (ID) de la historia	11
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los municipios

Descripción razón o resultado	Con la finalidad de gestionar los municipios
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los municipios
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un municipio
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'municipios'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar municipios. Los campos del CRUD son: código, nombre, estado, departamento

Tabla 12 CRUD de los medios de recepción

Identificador (ID) de la historia	12
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los medios de recepción
Descripción razón o resultado	Con la finalidad de gestionar los medios de recepción
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los medios de recepción
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un medio de recepción
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'medios de recepción'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar medios de recepción. Los campos del CRUD son: descripción, estado

Tabla 13 CRUD de los días no hábiles

Identificador (ID) de la historia	13
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los días no hábiles
Descripción razón o resultado	Con la finalidad de gestionar los días no hábiles
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los días no hábiles
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un día no hábil
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'días no hábiles'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar días o hábiles. Los campos del CRUD son: fecha no hábil

Tabla 14 CRUD de los tipos de radicados

Identificador (ID) de la historia	14
Rol	Usuario administrador

Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los tipos de radicados
Descripción razón o resultado	Con la finalidad de gestionar los tipos de radicados
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los tipos de radicados
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un tipo de radicado
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'tipos de radicados'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de radicados. Los campos del CRUD son: código, nombre, estado. se debe poder asignar, leer o eliminar tipos de documentos a los tipos de radicados.

Tabla 15 CRUD de los cargos

Identificador (ID) de la historia	15
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los cargos
Descripción razón o resultado	Con la finalidad de gestionar los cargos
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los cargos
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un continente
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'cargos'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar cargos. Los campos del CRUD son: código, nombre, tipo de vinculación, estado

Tabla 16 CRUD de los tipos de transacción

Identificador (ID) de la historia	16
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los tipos de transacción
Descripción razón o resultado	Con la finalidad de gestionar los continentes
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los tipos de transacción
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un tipo de transacción
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'tipos de transacción'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de transacción. Los campos del CRUD son: código, nombre, permite SMS, permite email, estado

Tabla 17 CRUD de las dependencias

Identificador (ID) de la historia	17
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de las dependencias
Descripción razón o resultado	Con la finalidad de gestionar las dependencias
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de las dependencias
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar una dependencia
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'dependencias'
Resultado / Comportamiento esperado	La aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar dependencias. Los campos del CRUD son: código, nombre, estado. Se debe poder leer y asignar consecutivo único a un tipo de radicado. Se debe poder asignar o eliminar cargos de las dependencias.

Tabla 18 CRUD de los estados de los radicados

Identificador (ID) de la historia	18
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los estados de los radicados
Descripción razón o resultado	Con la finalidad de gestionar los estados de los radicados
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los estados de los radicados
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un estado de radicado
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'Estados radicados'.
Resultado / Comportamiento esperado	la aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de documentos. Los campos del CRUD son: nombre, estado.

Tabla 19 CRUD de los ciudadanos

Identificador (ID) de la historia	19
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los ciudadanos
Descripción razón o resultado	Con la finalidad de gestionar los ciudadanos
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los ciudadanos
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un ciudadano
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'Ciudadanos'.

Resultado / Comportamiento esperado	la aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de documentos. Los campos del CRUD son: tipo de identificación, identificación, nombre completo, dirección, teléfono, email, estado, continente, país, departamento, municipio.
-------------------------------------	--

Tabla 20 CRUD de los tipos de documentos

Identificador (ID) de la historia	20
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito tener el CRUD de los tipos de documentos
Descripción razón o resultado	Con la finalidad de gestionar los tipos de documentos
Número (#) de escenario	1
Criterio de aceptación (Título)	CRUD de los tipos de documentos
Contexto	En caso de necesitar crear, consultar, actualizar o eliminar un tipo de documento
Evento	El usuario administrador debe ingresar al módulo configuración en la barra de opciones lateral izquierda, opción administración, opción 'Tipos de documentos'
Resultado / Comportamiento esperado	la aplicación debe mostrar una UI donde se pueda consultar, crear, eliminar o actualizar tipos de documentos. Los campos del CRUD son: nombre, termino de respuesta en días, estado.

Tabla 21 Radicaciones de entrada

Identificador (ID) de la historia	21
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito generar un radicado de entrada
Descripción razón o resultado	Con la finalidad de crear radicados de entrada
Número (#) de escenario	1
Criterio de aceptación (Título)	Crear radicado nuevo
Contexto	En caso de que necesite crear un radicado
Evento	El usuario debe ingresar al módulo radicación en la barra de opciones lateral izquierda, opción entrada
Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz con el formulario de radicación, en la cabecera de la interfaz debe decir el tipo de radicación. En el formulario debe estar la opción para buscar antecedente, la fecha del documento (cuando se recibió) y un campo donde se especifique si tiene alguna referencia, cuenta interna u oficio. Deber haber una opción para buscar al ciudadano remitente, registrarlo o modificarlo. Opción para buscar un medio de recepción, un tipo de documento, continente, país, departamento, municipio, dependencia destino y funcionario destino. Campo para dignatario, folios, asunto, descripción de anexos, tiempo para respuesta, si requiere respuesta o no el radicado y fecha fija para respuesta en caso que se necesite una fecha exacta con hora. Una vez lleno el formulario se da clic en el botón 'radicar' y la aplicación debe mostrar un mensaje donde diga que se generó el radicado y el número de radicado, debajo de este

	mensaje debe aparecer los botones: 'nuevo', 'modificar', 'sticker', 'documento principal', 'adjuntar documentos', 'informar'.
Número (#) de escenario	2
Criterio de aceptación (Título)	Crear un numero de radicado único luego de radicar
Contexto	
Evento	Cuando el usuario da clic en el botón 'radicar'
Resultado / Comportamiento esperado	La aplicación genera el radicado y le asigna un numero único. las primeras 4 cifras corresponden al año. las siguientes 3 cifras corresponden a la dependencia que radica. hay entidades con dependencias que pueden tener más de 3 cifras, en ese caso se asignan ceros a la izquierda dinámicamente. las siguientes 6 cifras corresponden al consecutivo general o el de la dependencia, si esta es independiente. La última cifra corresponde al tipo de radicado (entrada)
Número (#) de escenario	3
Criterio de aceptación (Título)	Crear sticker luego de radicar
Contexto	
Evento	Cuando el usuario da clic en el botón 'sticker'
Resultado / Comportamiento esperado	La aplicación debe crear dinámicamente el sticker dependiendo de la entidad y mostrar una previsualización para la impresión en una ventana nueva
Número (#) de escenario	4
Criterio de aceptación (Título)	Subir documento principal del radicado luego de radicar
Contexto	
Evento	El usuario debe dar clic en el botón 'documento principal'
Resultado / Comportamiento esperado	La aplicación debe mostrar una ventana donde se observe el usuario, la dependencia, el evento a realizar, campo para un comentario, numero de radicado, asunto, fecha del radicado y la opción de subir el documento principal. el nombre del documento debe tener la siguiente estructura: las primeras cifras son el consecutivo del radicado, la siguiente cifra es el tipo del radicado, luego la abreviatura del mes (ENE, de enero, por ejemplo), por ultimo las 4 cifras del año, la aplicación debe validar el consecutivo, el tipo y el año, que coincidan con el radicado. finalmente se da clic en el botón 'realizar'.
Número (#) de escenario	5
Criterio de aceptación (Título)	Subir adjuntos luego de radicar
Contexto	
Evento	El usuario debe dar clic en el botón 'adjuntar documentos'
Resultado / Comportamiento esperado	La aplicación de mostrar una interfaz donde muestre la lista de adjuntos y la opción de añadir un adjunto, esta opción debe abrir una ventana donde se pueda seleccionar el adjunto y escribir una descripción, luego de guardarlo debe volver a listar los adjuntos.
Número (#) de escenario	6
Criterio de aceptación (Título)	Modificar datos
Contexto	Cuando el usuario modifica algún dato del formulario de radicación
Evento	El usuario debe dar clic en el botón 'modificar'
Resultado / Comportamiento esperado	la aplicación debe modificar el campo editado del formulario (no se puede modificar la dependencia y el usuario destino) y mostrar un mensaje sobre el estado de la modificación realizada.
Número (#) de escenario	7
Criterio de aceptación (Título)	Informar a otro usuario
Contexto	Cuando el usuario ha radicado y necesita informar a otro usuario
Evento	El usuario debe dar clic en el botón 'informar'

Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz donde liste los usuarios informados y la opción de informar a un usuario, esta opción debe abrir una ventana donde se pueda seleccionar los usuarios y escribir una observación, luego de informar debe listar de nuevo los usuarios informados.
-------------------------------------	--

Tabla 22 Radicaciones de salida

Identificador (ID) de la historia	22
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito generar un radicado de salida
Descripción razón o resultado	Con la finalidad de crear radicados de salida como respuesta a otros radicados
Número (#) de escenario	1
Criterio de aceptación (Título)	Crear radicado nuevo
Contexto	En caso de que necesite crear un radicado
Evento	El usuario debe ingresar al módulo radicación en la barra de opciones lateral izquierda, opción entrada
Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz con el formulario de radicación, en la cabecera de la interfaz debe decir el tipo de radicación. En el formulario debe estar la opción para buscar antecedente, la fecha del documento (cuando se recibió) y un campo donde se especifique si tiene alguna referencia, cuenta interna u oficio. Deber haber una opción para buscar al ciudadano remitente, registrarlo o modificarlo. Opción para buscar un medio de recepción, un tipo de documento, continente, país, departamento, municipio, dependencia destino y funcionario destino. Campo para dignatario, folios, asunto, descripción de anexos, si es respuesta o no el radicado. Una vez lleno el formulario se da clic en el botón 'radicar' y la aplicación debe mostrar un mensaje donde diga que se generó el radicado y el número de radicado, debajo de este mensaje debe aparecer los botones: 'nuevo', 'modificar', 'sticker', 'documento principal', 'adjuntar documentos', 'informar', 'otros antecedentes', 'otros destinatarios'.
Número (#) de escenario	2
Criterio de aceptación (Título)	Crear un numero de radicado único luego de radicar
Contexto	
Evento	Cuando el usuario da clic en el botón 'radicar'
Resultado / Comportamiento esperado	La aplicación genera el radicado y le asigna un numero único. las primeras 4 cifras corresponden al año. las siguientes 3 cifras corresponden a la dependencia que radica. hay entidades con dependencias que pueden tener más de 3 cifras, en ese caso se asignan ceros a la izquierda dinámicamente. las siguientes 6 cifras corresponden al consecutivo general o el de la dependencia, si esta es independiente. La última cifra corresponde al tipo de radicado (entrada)
Número (#) de escenario	3
Criterio de aceptación (Título)	Crear sticker luego de radicar
Contexto	
Evento	Cuando el usuario da clic en el botón 'sticker'

Resultado / Comportamiento esperado	La aplicación debe crear dinámicamente el sticker dependiendo de la entidad y mostrar una previsualización para la impresión en una ventana nueva
Número (#) de escenario	4
Criterio de aceptación (Título)	Subir documento principal del radicado luego de radicar
Contexto	
Evento	El usuario debe dar clic en el botón 'documento principal'
Resultado / Comportamiento esperado	La aplicación debe mostrar una ventana donde se observe el usuario, la dependencia, el evento a realizar, campo para un comentario, numero de radicado, asunto, fecha del radicado y la opción de subir el documento principal. el nombre del documento debe tener la siguiente estructura: las primeras cifras son el consecutivo del radicado, la siguiente cifra es el tipo del radicado, luego la abreviatura del mes (ENE, de enero, por ejemplo), por ultimo las 4 cifras del año, la aplicación debe validar el consecutivo, el tipo y el año, que coincidan con el radicado. finalmente se da clic en el botón 'realizar'.
Número (#) de escenario	5
Criterio de aceptación (Título)	Subir adjuntos luego de radicar
Contexto	
Evento	El usuario debe dar clic en el botón 'adjuntar documentos'
Resultado / Comportamiento esperado	La aplicación de mostrar una interfaz donde muestre la lista de adjuntos y la opción de añadir un adjunto, esta opción debe abrir una ventana donde se pueda seleccionar el adjunto y escribir una descripción, luego de guardarlo debe volver a listar los adjuntos.
Número (#) de escenario	6
Criterio de aceptación (Título)	Modificar datos
Contexto	Cuando el usuario modifica algún dato del formulario de radicación
Evento	El usuario debe dar clic en el botón 'modificar'
Resultado / Comportamiento esperado	La aplicación debe modificar el campo editado del formulario (no se puede modificar la dependencia y el usuario destino) y mostrar un mensaje sobre el estado de la modificación realizada.
Número (#) de escenario	7
Criterio de aceptación (Título)	Informar a otro usuario
Contexto	Cuando el usuario ha radicado y necesita informar a otro usuario
Evento	El usuario debe dar clic en el botón 'informar'
Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz donde liste los usuarios informados y la opción de informar a un usuario, esta opción debe abrir una ventana donde se pueda seleccionar los usuarios y escribir una observación, luego de informar debe listar de nuevo los usuarios informados.
Número (#) de escenario	8
Criterio de aceptación (Título)	Agregar otros antecedentes
Contexto	
Evento	El usuario debe dar clic en el botón 'otros antecedentes'
Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz donde liste los antecedentes y la opción de añadir otros antecedentes, esta opción debe abrir una ventana donde se pueda seleccionar el antecedente, luego de seleccionarlo debe volver a listar.
Número (#) de escenario	9
Criterio de aceptación (Título)	Agregar otros destinatarios
Contexto	
Evento	El usuario debe dar clic en el botón 'otros destinatarios'

Resultado / Comportamiento esperado	La aplicación debe mostrar una interfaz donde liste los destinatarios y la opción de añadir otros destinatarios, esta opción debe abrir una ventana donde se pueda seleccionar el destinatario y editar su email si se desea, luego de seleccionarlo debe volver a listar.
-------------------------------------	--

Tabla 23 Otros tipos de radicaciones

Identificador (ID) de la historia	23
Rol	Usuario administrador
Tipo de requerimiento	Proceso
Característica / Funcionalidad/Requerimiento	Necesito generar radicados con los tipos de radicados que se creen
Descripción razón o resultado	Con la finalidad de generar radicados de cualquier tipo
Número (#) de escenario	1
Criterio de aceptación (Título)	Generar radicados
Contexto	En caso que el usuario cree tipos de radicados y requiera generar radicados de esos tipos
Evento	El usuario debe ingresar al módulo radicación en la barra de opciones lateral izquierda, y buscar el tipo de radicado que desea generar
Resultado / Comportamiento esperado	La aplicación debe mostrar un formulario con los mismos campos del tipo de radicado de entrada y las mismas opciones luego de radicar.

Tabla 24 Generar reportes

Identificador (ID) de la historia	24
Rol	Usuario administrador
Tipo de requerimiento	Reporte
Característica / Funcionalidad/Requerimiento	Necesito generar reportes de la aplicación
Descripción razón o resultado	Con la finalidad de generar reportes
Número (#) de escenario	1
Criterio de aceptación (Título)	Generar reportes
Contexto	En caso de necesitar generar un tipo de reporte
Evento	El usuario administrador debe ingresar al módulo 'Estadísticas' en la barra de opciones lateral izquierda
Resultado / Comportamiento esperado	Al abrir el menú debe aparecer los tipos de reporte, reporte de correspondencia de entrada, reporte de correspondencia de salida y reporte de radicados por usuario radicador a nivel general y a nivel de detalle. La aplicación debe mostrar una interfaz para cada opción de reporte donde se pueda llenar los datos correspondientes para el reporte. al dar clic en 'generar' debe agrupar la lista por usuarios. se debe visualizar en pdf y Excel, en general y en detalle por usuario agrupado.

3.4. Diseño

El diseño de software es un proceso fundamental en el desarrollo de las aplicaciones, ya que permite dar una visión de cómo se va a elaborar el proyecto y así disminuir la mayor cantidad de errores y manejar su complejidad. En esta sección se presenta la arquitectura del sistema y el diseño de base de datos del presente trabajo.

3.4.1. *Diseño Arquitectónico*

La arquitectura de software nos ayuda a generar la estructura de nuestro sistema y a tener una visión general, lo que nos permite tener un control detallado de los componentes y capas de nuestro sistema. En la ilustración 9 se puede observar la arquitectura de la aplicación SIEPDO, la cual vamos a describir en esta sección.

La arquitectura de microservicios nos habla de que, en lugar de crear una aplicación monolítica, desarrollamos una aplicación como un conjunto de servicios pequeños. Un monolito se puede dividir en servicios pequeños, los cuales pueden tener funciones específicas dentro de toda la aplicación. Cada microservicio corre su propio proceso de forma independiente de los demás microservicios. Los microservicios se deben comunicar, si lo necesitan, con mecanismos muy livianos para realizar el intercambio de información de la manera más rápida y óptima posible. Normalmente, se desarrollan alrededor de las necesidades del negocio, hay una fuerte conexión entre cómo operan la lógica del negocio y como se desarrollan los servicios. Debe haber un proceso de despliegue automatizado de la aplicación y toda su infraestructura.

Los microservicios pueden estar organizados de distintas formas, a esto se le conoce como topologías. En la topología basada en API REST, los servicios son consumidos, pero no se accede a ellos directamente, sino que se utiliza una capa de API, la cual expone las URL públicas a las que los distintos clientes pueden acceder, esta API se comunica con los servicios quienes

procesan las solicitudes y le envían una respuesta para que la API la envíe al cliente. En la topología basada en aplicación, la aplicación que accede a los microservicios es una aplicación del mismo ecosistema de los microservicios y lo que hace es conectarse directamente a ellos, los clientes o usuarios interactúan con esta aplicación y ella internamente encapsula o esconde estos llamados a los microservicios. Las topologías basadas en mensajería no interactúan con los servicios a través de REST sino a través de un bróker de mensajería o de una cola, los usuarios interactúan con la aplicación y esta envía los mensajes en la cola y esa cola se encarga de dirigir el mensaje al servicio correspondiente, es utilizada en monitoreo y comunicación asíncrona.

En el desarrollo del presente trabajo se utiliza la topología basada en API REST por medio de la cual se realiza la comunicación entre microservicios. Se utiliza los Starters de Spring Cloud Starter de Netflix Eureka Server para la localización entre microservicios y para su comunicación con Spring Cloud Starter OpenFeign. Spring Cloud Starter Gateway para el enrutamiento de solicitudes, autenticación y control de acceso para las llamadas de las API. Spring Cloud Config Server para externalizar la configuración de nuestras aplicaciones, y un microservicio de autenticación que trabaja con Java JWT y Spring Security para autenticar los usuarios y controlar el acceso en la aplicación, esto lo podemos observar en la ilustración 10.

Para la persistencia de los datos se utiliza el marco de persistencia MyBatis Velocity, que se implementara en cada uno de los microservicios que poseen la lógica de negocio. El motor de bases de datos que se utiliza es PostgreSQL, en este proyecto se maneja una sola base de datos con tres esquemas, un esquema para cada microservicio de la lógica del negocio, estos esquemas serán tratados como bases de datos independientes, cuando se desee tener información entre esquemas se consumen entre microservicios como si fueran clientes de ellos mismos, esta comunicación no se hace por medio del Gateway.

Vaadin Flow, el marco de desarrollo (de la plataforma Vaadin) que utilizamos en el presente proyecto, trabaja con una arquitectura del lado del servidor que nunca expone su interior al navegador, lo que reduce la exposición a los ataques de seguridad.

En los microservicios que manejan la lógica del negocio se implementa la arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores. En el microservicio de autenticación y microservicio de enrutamiento se implementa manera similar la misma estructura de paquetes y arquitectura. El microservicio de Vaadin, donde se encuentra la lógica del usuario, se utiliza la arquitectura MVP y una estructura de carpetas basada también en un proyecto de Vaadin llamado ‘Business App Starter’ el cual proporciona una aplicación como base con componentes y plantillas de vista gratuitos para crear aplicaciones web comerciales.

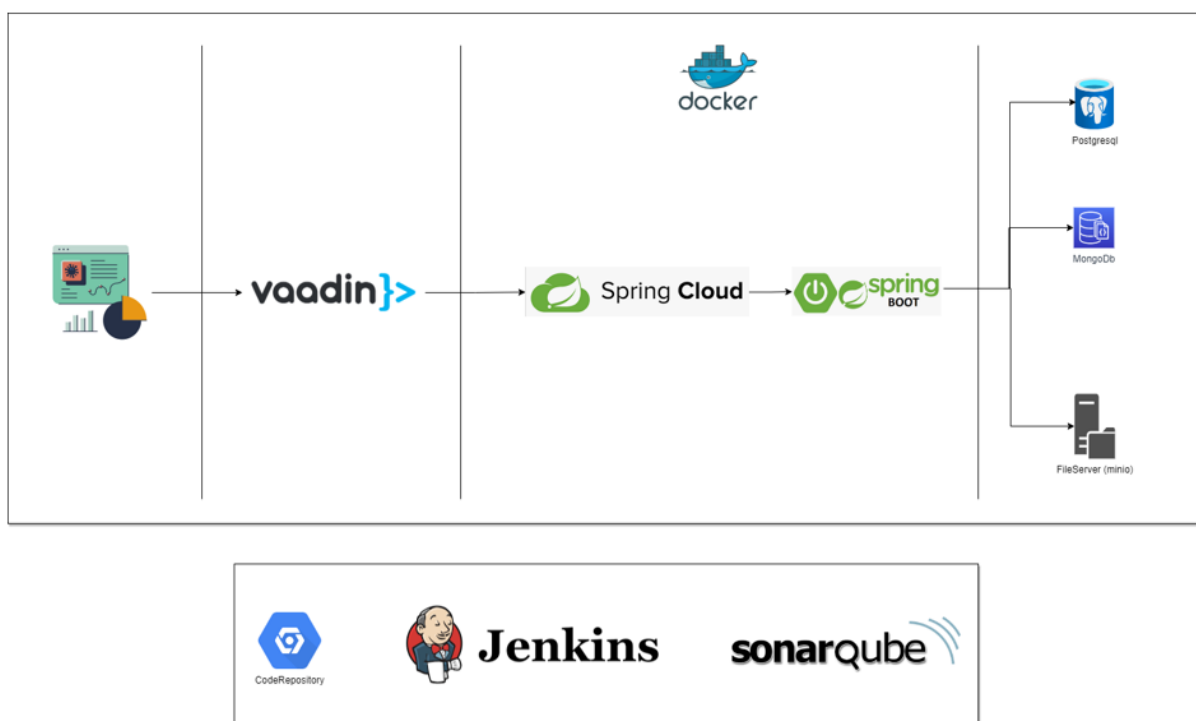


Ilustración 9 Arquitectura SIEPDOC, tomada de OPE Colombia.

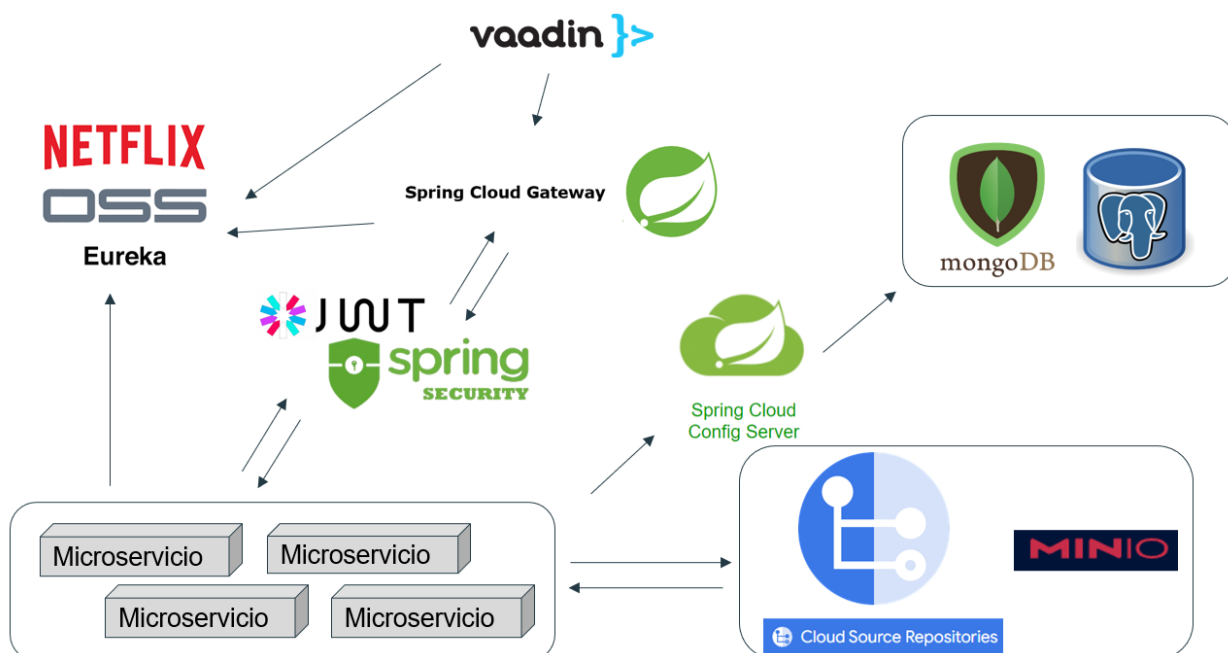


Ilustración 10 Arquitectura de herramientas para el funcionamiento de SIEPDOC, tomada de OPE Colombia.

En la ilustración 11 se puede observar la arquitectura de las herramientas que se utilizan para el despliegue continuo y la integración continua de la aplicación SIEPDOC, además se puede apreciar la herramienta utilizada para el monitoreo de la calidad del código.

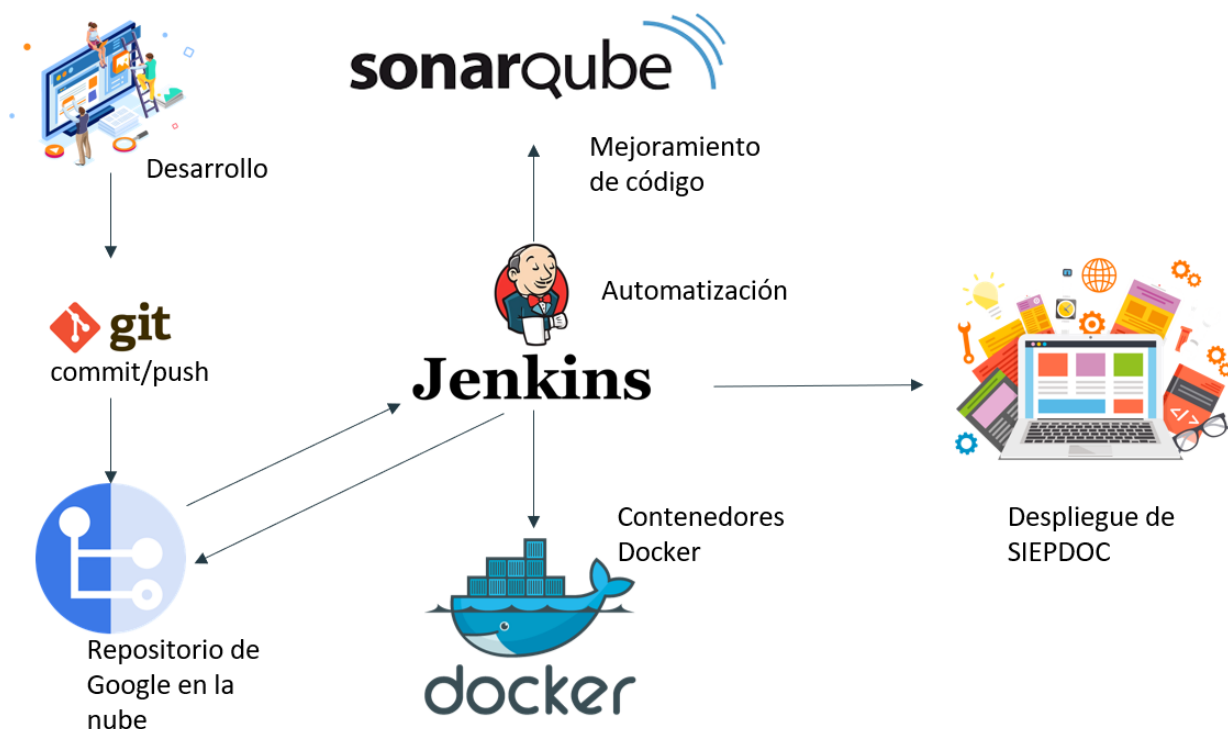


Ilustración 11 Arquitectura para despliegue e integración continua, tomada de OPE Colombia.

3.4.2. Arquitectura Hexagonal

La ilustración 12 muestra una representación gráfica de lo que es la Arquitectura Hexagonal, en el Core o núcleo está el dominio de la aplicación que son las entidades o servicios de dominio que contienen las reglas de negocio que se han establecido en la aplicación. En la capa de aplicación están los servicios de aplicación que representan los casos de uso del sistema. En la capa de Framework se encuentra la infraestructura donde se puede encontrar, por ejemplo, el repositorio o sistema de persistencia de datos y los controladores. De esta manera se aísla toda la lógica de negocio y casos de uso de la infraestructura, ya que, dado el caso en que se desee cambiar el sistema de persistencia de datos, por ejemplo, no se va a ver afectado el dominio.

La regla de dependencias de las capas la podemos observar en la ilustración 13, esta regla nos dice que algo que está en una capa más interior no puede conocer lo que hay en una capa más exterior.

Como la lógica de negocio es independiente de otros elementos de la aplicación tales como la base de datos o la interfaz de usuario, la comunicación con estos elementos se realiza por medio de los puertos que son las interfaces que realizan la comunicación externa y los adaptadores, que son las implementaciones de los puertos, que realizan la comunicación interna con nuestra lógica de negocio. Los puertos y adaptadores se clasifican en actores primarios y actores secundarios, los actores primarios son los que se utilizan para invocar la lógica de negocio, estos se utilizan por ejemplo en las API REST, y los actores secundarios son quienes son utilizados por la lógica de negocio, estos se utilizan por ejemplo en los repositorios para el intercambio de datos con la base de datos.



Ilustración 12 Arquitectura Hexagonal.
Tomada de <https://fideloper.com/hexagonal-architecture>

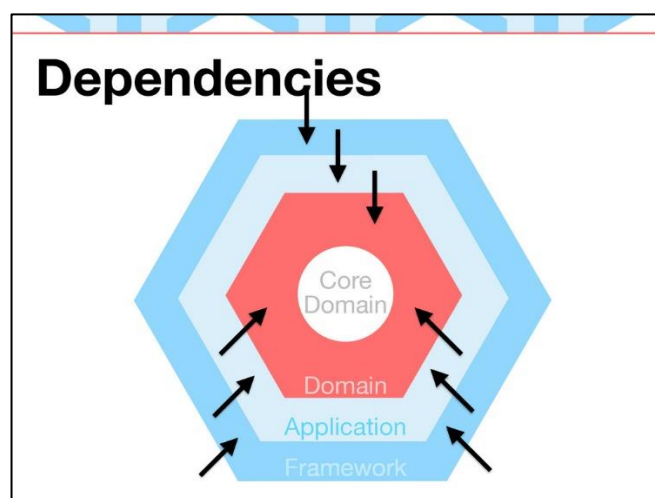


Ilustración 13 Regla de dependencia de capas de la Arquitectura Hexagonal.
Tomada de <https://fideloper.com/hexagonal-architecture>

En la ilustración 14 se puede observar la estructura de directorios aplicada en SIEPDOG basada en la arquitectura hexagonal. En la carpeta infraestructura se puede observar el dominio acompañado de los puertos y adaptadores.

En la carpeta de dominio se encuentra los casos de uso y los modelos, en la carpeta de adaptadores se encuentra la implementación de los puertos, y en la carpeta puertos se encuentran

las interfaces que se comunican con el exterior de la lógica de negocio, como lo son las carpetas repositorios, mappers y clientes.

En la carpeta puertos primarios se encuentran los servicios, estos se utilizan para la comunicación entre los controladores y la lógica del negocio. En la carpeta puertos secundarios se encuentran los repositorios para la conexión a la base de datos y las conexiones feign a otros microservicios, con los cuales es necesario interactuar e intercambiar información; también se encuentran los mappers a los cuales se conectan los repositorios, esto permite una fácil adaptación en caso de cambiar el marco de persistencia de datos.

Finalmente, hay un paquete ‘api’ donde se encuentra los controladores y converters que transforman la información de los VO, en términos de la lógica del negocio.

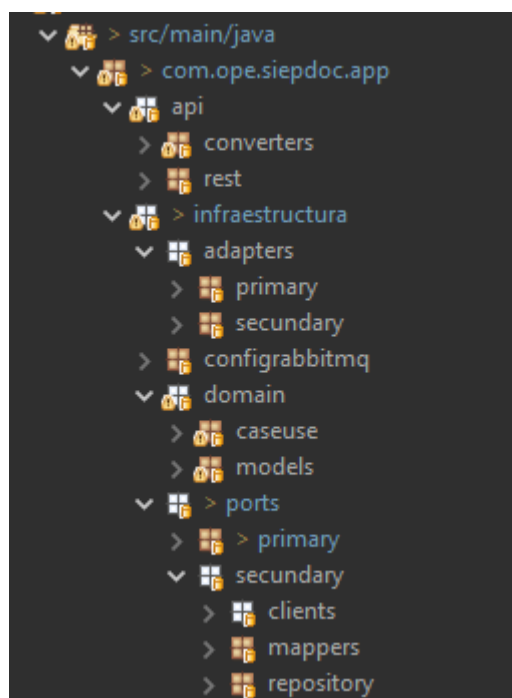


Ilustración 14 Arquitectura Hexagonal aplicada a SIEPDOC

La implementación de esta arquitectura se explica detalladamente en la sección 3.6 dedicada al BackEnd, donde se describe como se ha diseñado cada microservicio teniendo en cuenta estos parámetros.

3.4.3. Arquitectura MVP

Una de las principales ventajas de la arquitectura MVP es que la interfaz del usuario queda desacoplada del código de construcción de la interfaz de usuario, por lo tanto, es posible reemplazar a Vaadin con otro marco de interfaz de usuario sin cambiar el código construido. En la ilustración 15 se puede observar que la vista está separada del modelo de datos y en medio de estos se encuentra el presentador que los comunica. En SIEPDOG la vista es la interfaz de usuario de Vaadin Flow, el presentador son clases llamadas presenter, y el modelo son services que consumen las API del BackEnd para devolver la información al presenter.

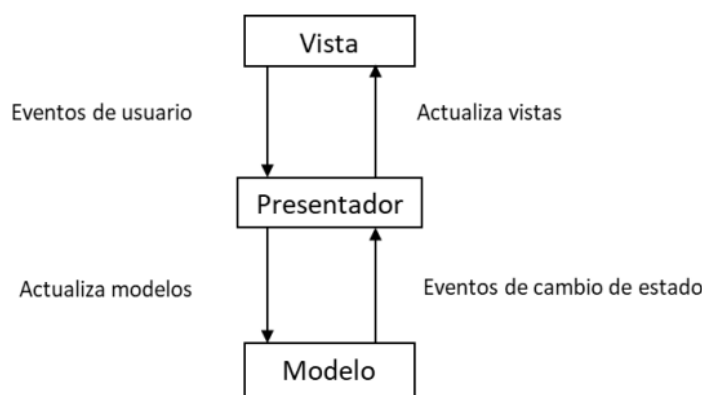


Ilustración 15 Diagrama en bloques que representa el funcionamiento de la arquitectura MVP.
Tomada de <https://dialnet.unirioja.es/servlet/articulo?codigo=7940724>

La implementación de esta arquitectura se explica detalladamente en la sección 3.5 dedicada al FrontEnd, donde se describe como se ha diseñado el microservicio teniendo en cuenta estos parámetros.

3.4.4. Diseño de bases de datos

Cabe resaltar que en el presente proyecto se implementa una base de datos en PostgreSQL con tres esquemas diferentes, los cuales se manejan como base de datos independientes en la lógica del negocio. Entre los esquemas se encuentra el esquema “scc” que contiene las tablas básicas de configuraciones y tablas de ubicación, el esquema “scgu” contiene

las tablas relacionadas al usuario y permisos, y el esquema “scgd” que contiene la tabla de radicación y las tablas de gestión de documentos y radicados.

En la ilustración 16 se puede observar el modelo entidad relación del esquema de base de datos “scgu”, este esquema contiene las tablas para la creación de usuarios, y roles y permisos.

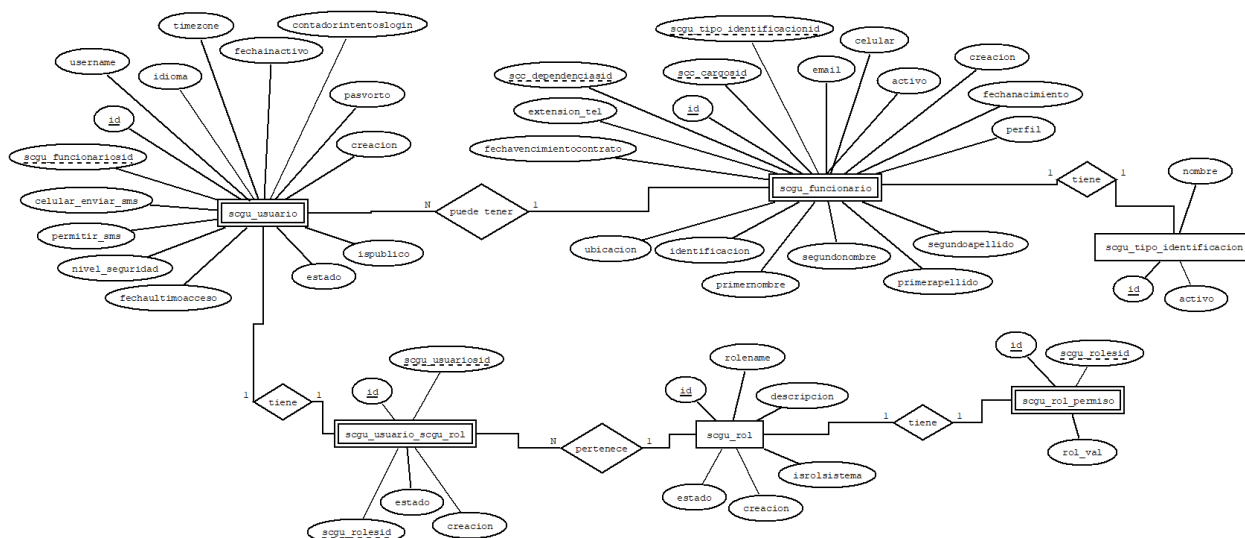


Ilustración 16 Modelo ER del esquema scgu

En la ilustración 17 se muestra el modelo entidad relación del esquema de base de datos “scc”, este esquema contiene las tablas básicas, como tablas de ubicación y tipos de radicados, que se utilizan tanto en el esquema “scgu” como el esquema “scgd”.

En la ilustración 18 se observa el modelo entidad relación del esquema de base de datos “scgd”, este esquema contiene las tablas para la creación de radicados.

En la ilustración 18 se observa el modelo entidad relación del esquema de base de datos “scgd”, este esquema contiene las tablas para la creación de radicados.

3.5. Desarrollo FrontEnd

3.5.1. Siepdoc_micro_vaadin

El microservicio que posee la lógica e interfaz de usuario tiene como nombre Siepdoc_micro_vaadin. Esta sección presenta las principales pantallas de la aplicación como el Login y la navegación del usuario, los principales módulos de la aplicación como lo son Radicación, y Administración de Usuarios y Roles. Las demás pantallas básicas del usuario y de estadísticas se pueden consultar en el documento anexo “Anexo3. Manual de usuario”.

3.5.1.1. *Cómo navegar entre las transacciones del sistema*

Al ingresar a la aplicación se podrá observar el menú de la ilustración 19, el cual siempre estará en la parte izquierda acompañando fielmente el uso del programa, además permitirá cambiar fácilmente de función o de pantalla según se requiera. En el recuadro negro vacío se ubica el logo de las entidades. Este menú de opciones permite minimizarse como se muestra en la ilustración 20, lo cual, a su vez permite visualizar mejor algunas pantallas de la aplicación, esta opción se encuentra en la parte inferior del menú de navegación.



Ilustración 19 Menú de opciones expandido

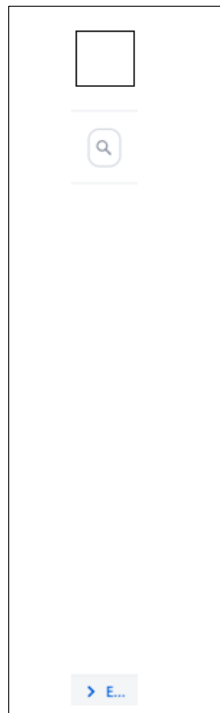


Ilustración 20 Menú de opciones minimizado

Al hacer clic en los diferentes menús, se puede observar cómo se despliega los submenús y las diferentes opciones de estos, como el menú de Radicación y sus opciones en la ilustración 21.

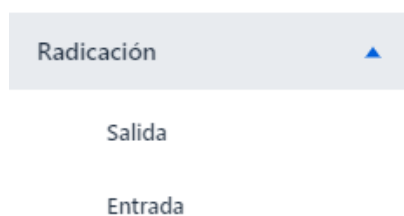


Ilustración 21 Menú de Radicación

También se puede observar el menú de Configuración y sus opciones en la ilustración 22 y 23.

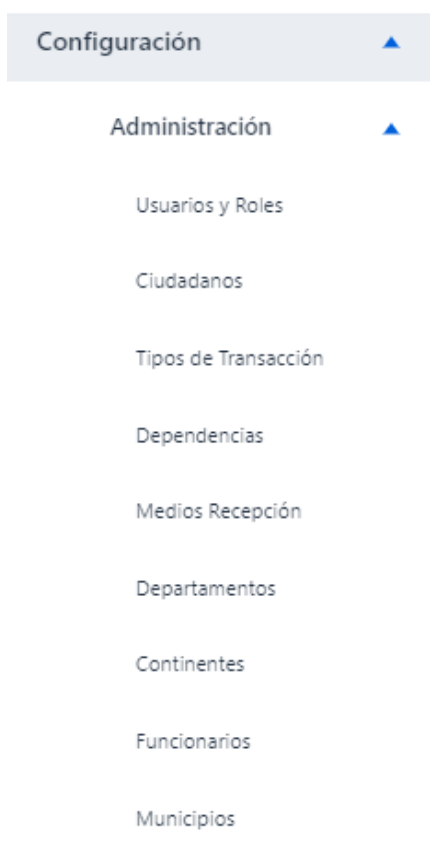


Ilustración 22 Menú de configuracion parte 1

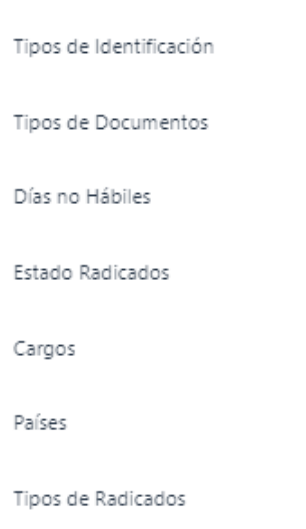


Ilustración 23 Menú de configuracion parte 2

También tenemos el menú de Estadísticas en la ilustración 24, en donde podemos generar diferentes reportes como se observa en las opciones del menú.

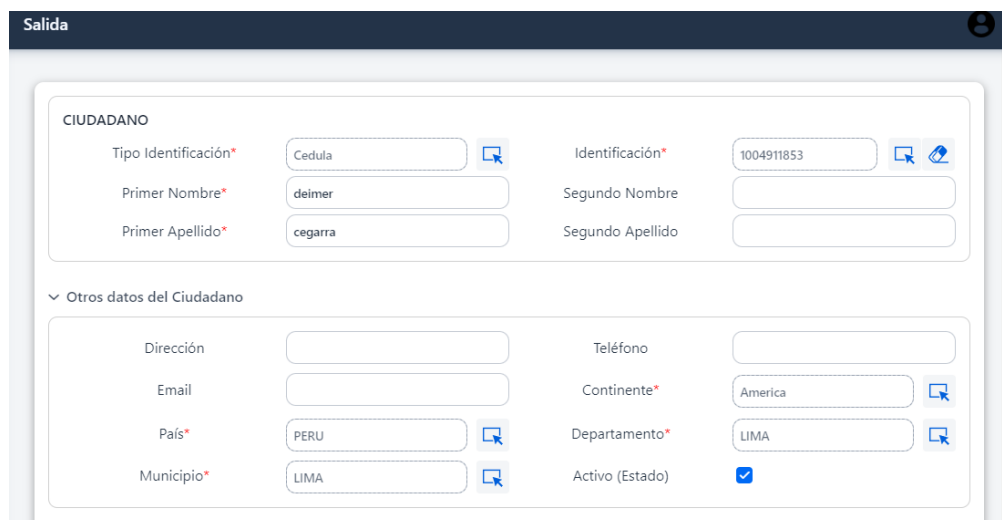


Ilustración 24 Menú de Estadísticas

3.5.1.2. Radicación de salida

En el menú de Radicación, como se pudo apreciar anteriormente la navegación del usuario, existen dos tipos de radicados fijos que son del sistema y no pueden ser eliminados. Las ilustraciones 25, 26, y 27 presentan el formulario de radicación de salida y sus diferentes opciones. Los campos obligatorios son aquellos que tienen un asterisco rojo.

En la ilustración 25 se puede observar la sección ‘Otros datos del Ciudadano’ acompañado de una flecha hacia abajo a su izquierda, esto significa que estos campos se pueden minimizar o esconder al hacer clic en esta flecha o en el título. En caso de que el ciudadano no esté registrado anteriormente en la aplicación, se puede crear desde esta pantalla, para esto se deben llenar los campos que son obligatorios.



Salida

CIUDADANO

Tipo Identificación* Cedula Identificación* 1004911853

Primer Nombre* deimer Segundo Nombre

Primer Apellido* cegarra Segundo Apellido

▼ Otros datos del Ciudadano

Dirección Teléfono

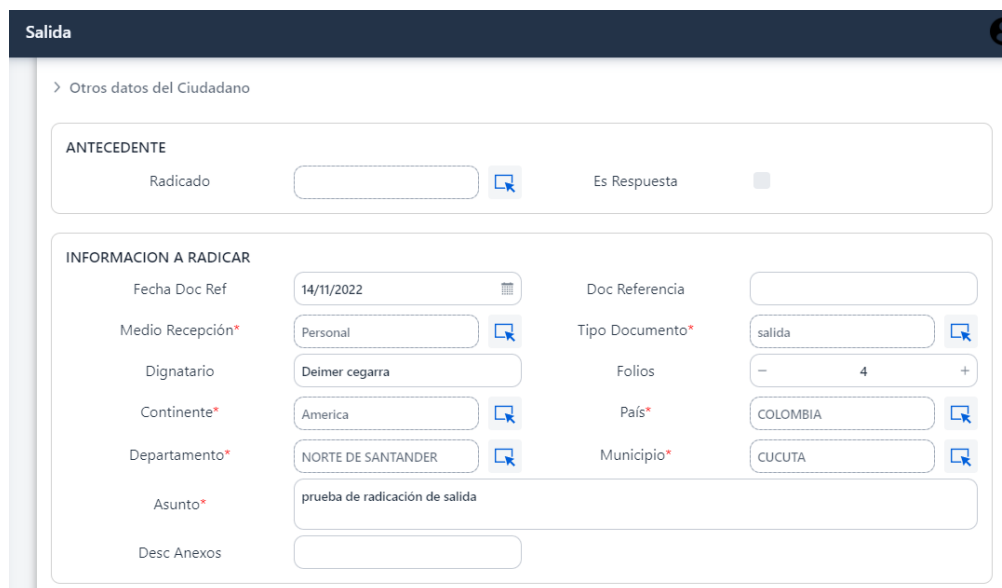
Email Continente* America

País* PERU Departamento* LIMA

Municipio* LIMA Activo (Estado) ☒

Ilustración 25 Formulario Radicación de salida parte 1

En la ilustración 26 se puede observar más campos del formulario, además se puede apreciar cómo está minimizado o escondido los campos de la sección ‘Otros datos del Ciudadano’.



Salida

> Otros datos del Ciudadano

ANTECEDENTE

Radicado Es Respuesta

INFORMACION A RADICAR

Fecha Doc Ref 14/11/2022 Doc Referencia

Medio Recepción* Personal Tipo Documento* salida

Dignatario Deimer cegarra Folios - 4 +

Continente* America País* COLOMBIA

Departamento* NORTE DE SANTANDER Municipio* CUCUTA

Asunto* prueba de radicación de salida

Desc Anexos

Ilustración 26 Formulario Radicación de salida parte 2

Se puede observar más campos en la ilustración 27, y los botones ‘Limpiar’ que vacía el formulario completamente y ‘Radicar’ que genera el radicado.

Salida

INFORMACIÓN A RADICAR

Fecha Doc Ref	14/11/2022	Doc Referencia	
Medio Recepción*	Personal	Tipo Documento*	salida
Dignatario	Deimer cegarra	Folios	4
Continente*	America	País*	COLOMBIA
Departamento*	NORTE DE SANTANDER	Municipio*	CUCUTA
Asunto*	prueba de radicación de salida		
Desc Anexos			

ASIGNACION

Dependencia*	000-Administradores	Funcionario*	prueba correo
--------------	---------------------	--------------	---------------

Limpiar **Radicar**

Ilustración 27 Formulario Radicación de salida parte 3

Luego de radicar, se puede apreciar en la ilustración 28, que se habilitan diferentes botones para realizar acciones sobre el radicado generado. El botón ‘Nuevo’ sirve para limpiar el formulario para poder generar un nuevo radicado. El botón ‘Modificar’ sirve para actualizar el radicado si se editan los campos. La dependencia y el funcionario (usuario) no se pueden modificar luego de ser generado el radicado.

Ilustración 28 Opciones luego de generar un radicado de salida

3.5.1.3. Sticker

Al dar clic en el botón ‘Sticker’ se va a generar el sticker correspondiente al radicado como el ejemplo de la ilustración 29. En el recuadro negro vacío se ubica el logo de la entidad.

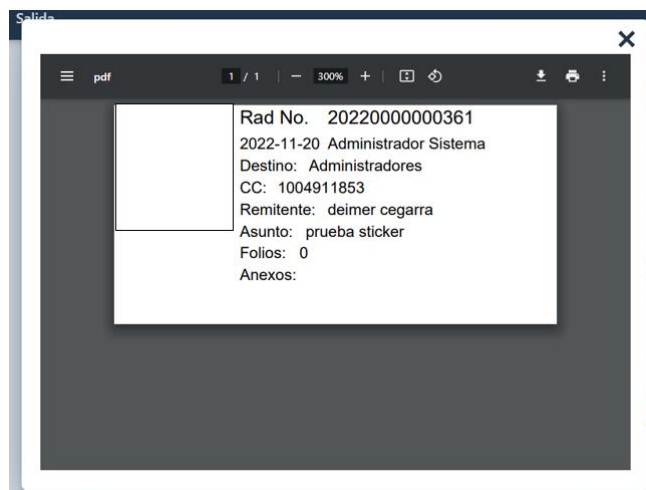
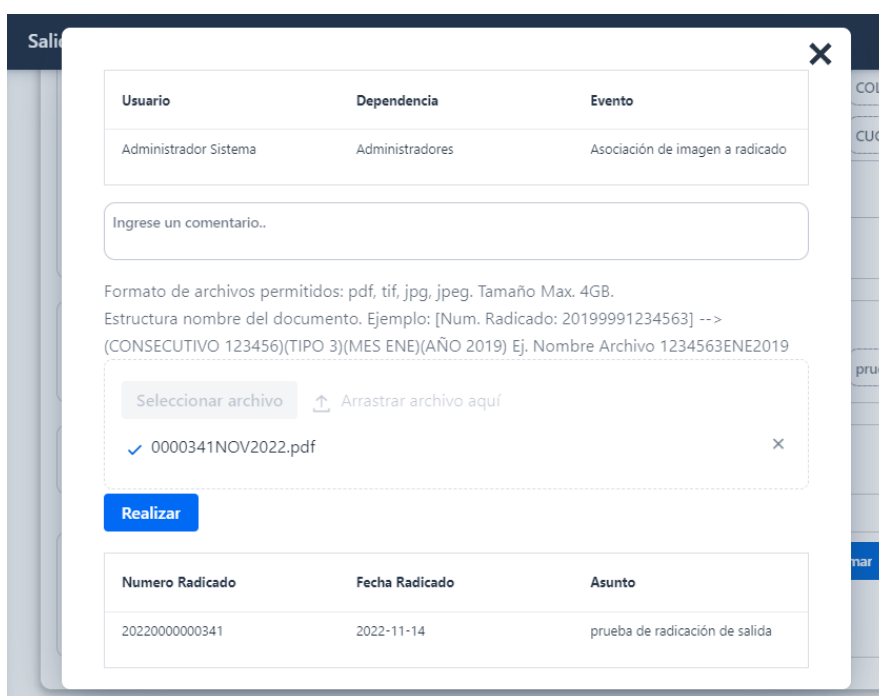


Ilustración 29 Generar Sticker

Luego de tener impreso el sticker se adjunta al documento en físico y se procede escanear el documento en el programa del pc para cargarlo como Documento Principal.

3.5.1.3.1. Documento principal

Al dar clic en el botón “Documento Principal” aparece la pantalla de la ilustración 30 donde se puede cargar el documento principal del radicado, en esta podemos ver información del usuario y del radicado, también especifica el formato que debe tener el nombre del documento, el tamaño y tipo de archivo permitidos.



Usuario	Dependencia	Evento
Administrador Sistema	Administradores	Asociación de imagen a radicado

Ingrese un comentario..

Formato de archivos permitidos: pdf, tif, jpg, jpeg. Tamaño Max. 4GB.
 Estructura nombre del documento. Ejemplo: [Num. Radicado: 20199991234563] --> (CONSECUTIVO 123456)(TIPO 3)(MES ENE)(AÑO 2019) Ej. Nombre Archivo 1234563ENE2019

Selecionar archivo Arrastrar archivo aquí

✓ 0000341NOV2022.pdf

Realizar

Numero Radicado	Fecha Radicado	Asunto
20220000000341	2022-11-14	prueba de radicación de salida

Ilustración 30 Cargar documento principal

Al dar clic en el botón ‘Realizar’ y cargar exitosamente el documento principal, aparece la pantalla de la ilustración 31, la cual indica que la acción fue completada con éxito.

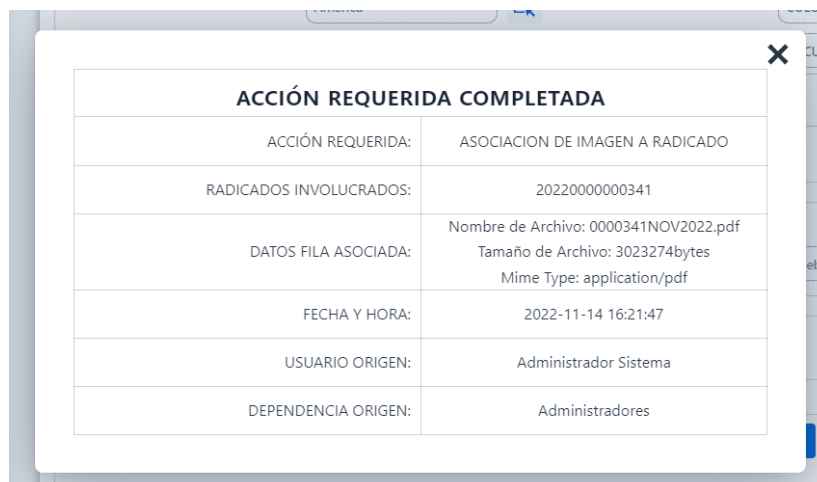


Ilustración 31 Carga exitosa de documento principal

3.5.1.3.2. Adjuntar documentos

Al dar clic en el botón ‘Adjuntar Documentos’ aparece la pantalla de la ilustración 32 en la cual podemos observar la lista de adjuntos del radicado. También se pueden eliminar los adjuntos desde esta pantalla.

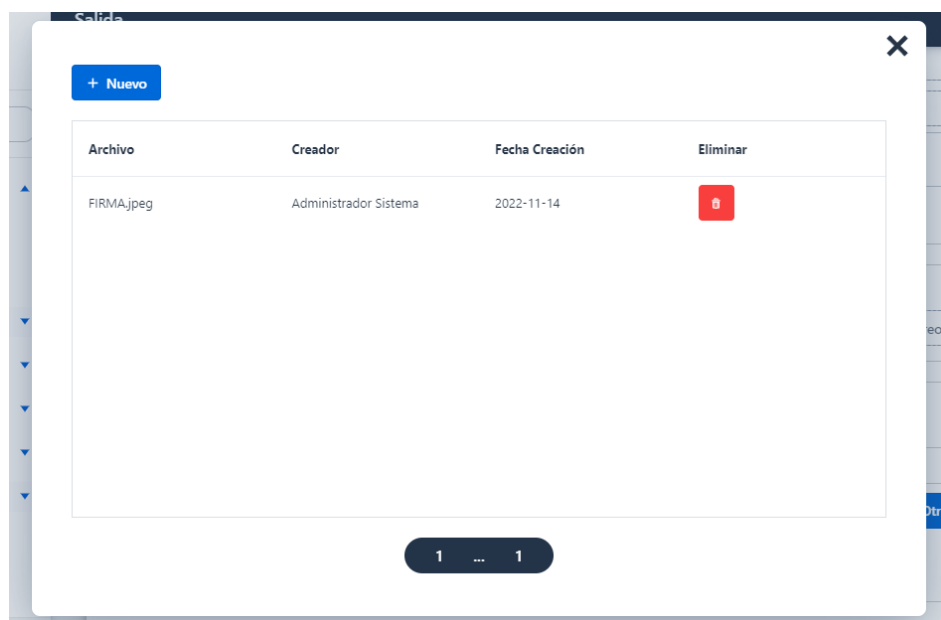


Ilustración 32 Lista de adjuntos

Al dar clic en el boton 'Nuevo' aparece la pantalla de la ilustracion 33, la cual nos permite cargar los adjuntos del radicado. Tambien nos indica que el campo 'descripcion' es obligatorio. Al dar clic en el boton 'Realizar' y cargar exitosamente el adjunto, nos enviará de vuelta a la pantalla de la ilustracion 32 donde aparecerá en la lista nuestro adjunto cargado.

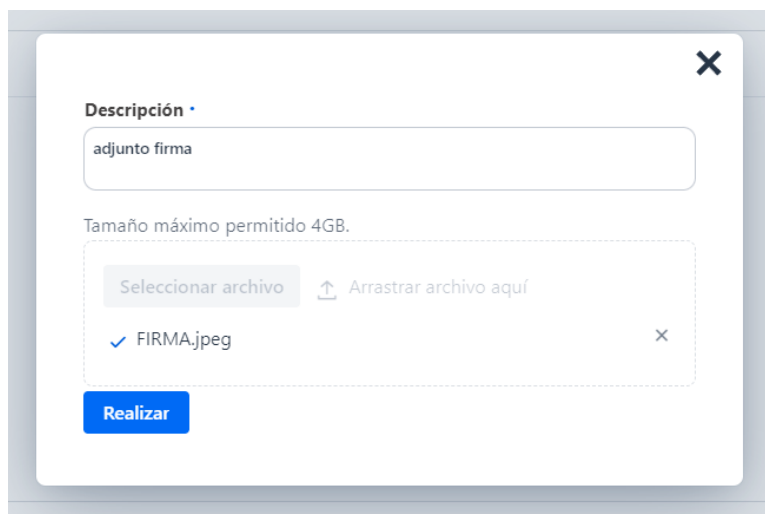
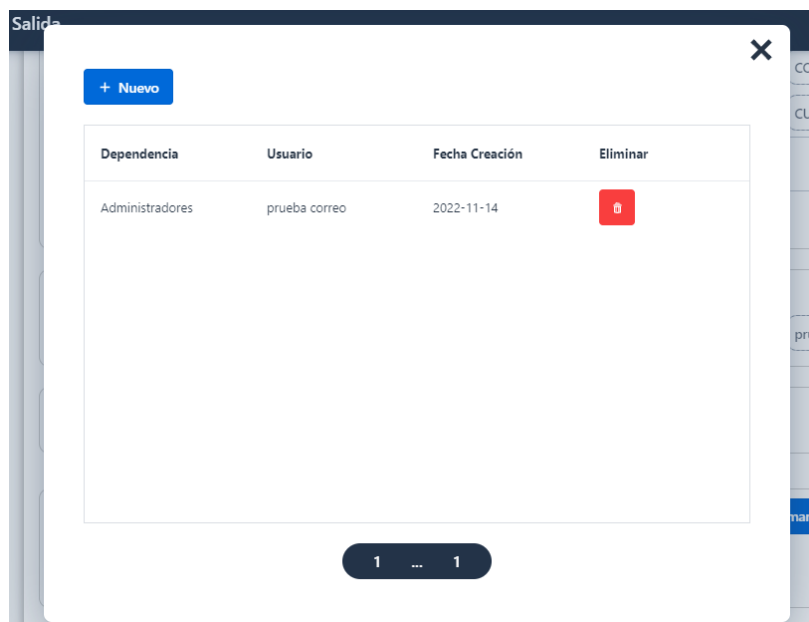


Ilustración 33 Cargar adjuntos

3.5.1.3.3. Informar

Al dar clic en el boton 'Informar' aparece la pantalla de la ilustracion 34 en la cual podemos observar la lista de informados del radicado. También se pueden eliminar los informados desde esta pantalla.



Dependencia	Usuario	Fecha Creación	Eliminar
Administradores	prueba correo	2022-11-14	

Ilustración 34 Lista de informados

Al dar clic en el boton ‘Nuevo’ aparece la pantalla de la ilustracion 35, la cual nos permite informar a otros funcionarios (usuarios) sobre el radicado. Tambien nos indica que el campo ‘observacion’ es obligatorio. Al dar clic en el boton ‘Realizar’ nos enviará de vuelta a la pantalla de la ilustracion 34 donde aparecerá en la lista los informados.

Dependencias

☐	Codigo	Nombre
☒	000	Administradores
☐	001	nn
☐	002	inactiva

Usuarios

☐	Primer Nombre	Primer Apellido
☒	prueba	correo
☐	Administrador	Sistema

Observacion

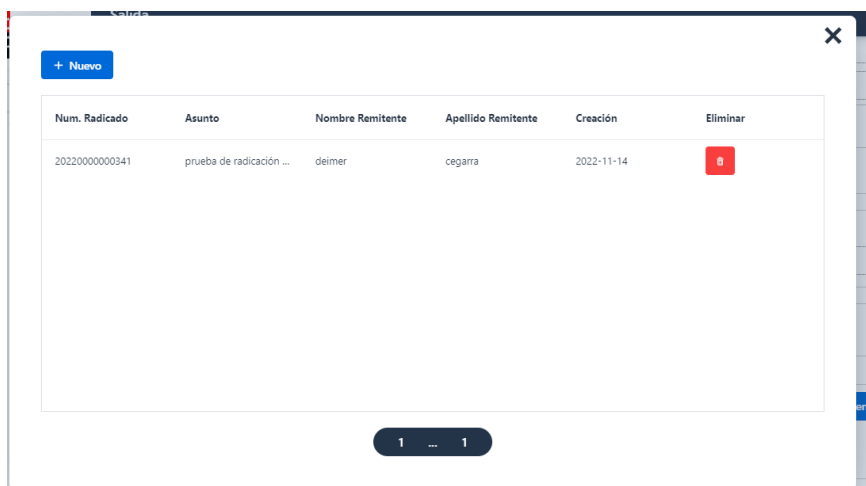
prueba informar a otro usuario

Realizar

Ilustración 35 Informar

3.5.1.3.4. Otros antecedentes

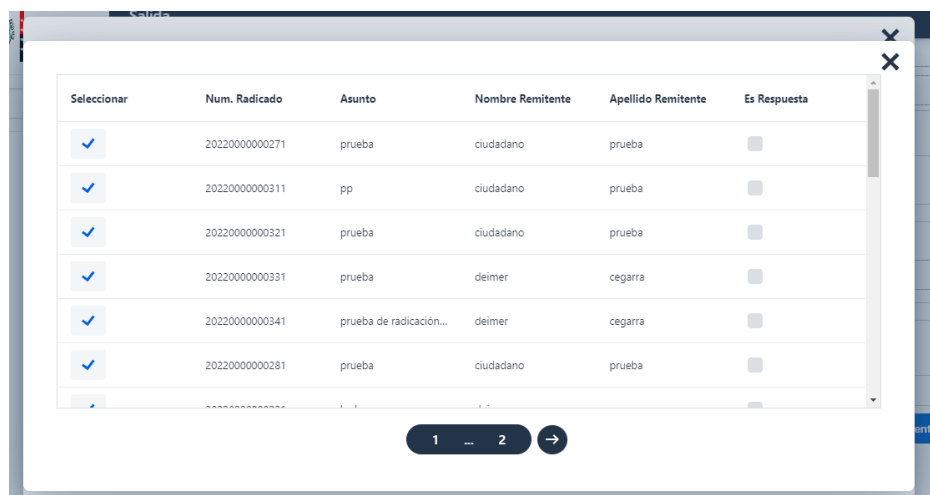
Al dar clic en el boton ‘Otros Antecedentes’ aparece la pantalla de la ilustracion 36 en la cual podemos observar la lista de antecedentes del radicado. También se pueden eliminar los antecedentes desde esta pantalla.



Num. Radicado	Asunto	Nombre Remitente	Apellido Remitente	Creación	Eliminar
202200000000341	prueba de radicación ...	deimer	cegarra	2022-11-14	

Ilustración 36 Lista de otros antecedentes

Al dar clic en el boton ‘Nuevo’ aparece la pantalla de la ilustracion 37, la cual nos permite agregar otros antecedentes al radicado. Tambien podemos marcar el checkbox en la columna ‘Es Respuesta’ para indicar que el radicado actual es respuesta de ese atecedente que se va a seleccionar. Al dar clic en el boton de la columna ‘Seleccionar’ nos enviará de vuelta a la pantalla de la ilustracion 36 donde aparecerá en la lista los antecedentes.



Seleccionar	Num. Radicado	Asunto	Nombre Remitente	Apellido Remitente	Es Respuesta
☒	202200000000271	prueba	ciudadano	prueba	☐
☒	202200000000311	pp	ciudadano	prueba	☐
☒	202200000000321	prueba	ciudadano	prueba	☐
☒	202200000000331	prueba	deimer	cegarra	☐
☒	202200000000341	prueba de radicación...	deimer	cegarra	☐
☒	202200000000281	prueba	ciudadano	prueba	☐
☒	202200000000291	prueba	ciudadano	prueba	☐

Ilustración 37 Seleccionar antecedente

3.5.1.3.5. Otros destinatarios

Al dar clic en el boton ‘Otros Destinatarios’ aparece la pantalla de la ilustracion 38 en la cual podemos observar la lista de destinatarios del radicado. También se pueden eliminar los destinatarios desde esta pantalla.

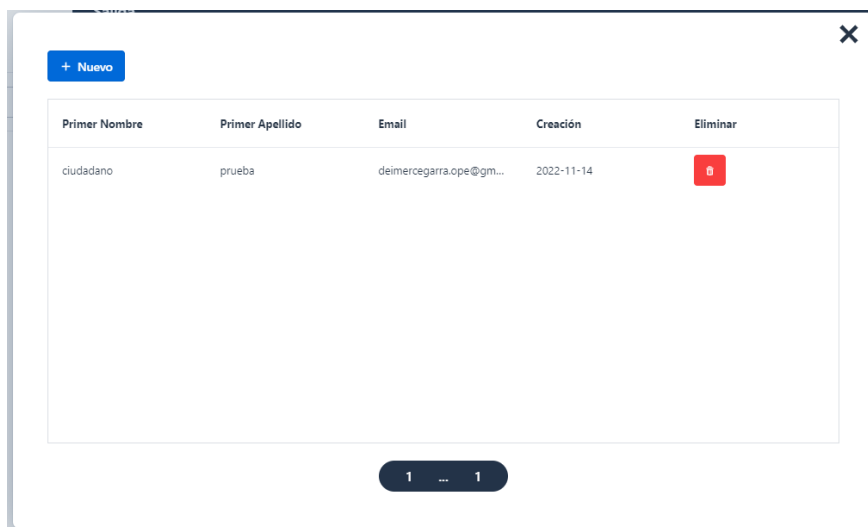


Ilustración 38 Lista de otros destinatarios

Al dar clic en el boton ‘Nuevo’ aparece la pantalla de la ilustracion 39, la cual nos permite editar el email del destinatario que seleccionemos. Al dar clic en el boton ‘Registrar’ nos envia de vuelta a la pantalla de la ilustracion 38 donde aparecera en la lista de destinatarios.

Primer Apellido Email Creación Elimini

Destinatario

Email

Ilustración 39 Editar email del destinatario

Al dar clic en el boton ‘Buscar’ aparece la pantalla de la ilustracion 40, la cual nos permite seleccionar el destinatario. Al dar clic en el boton de la columna ‘Seleccionar’ nos enviará de vuelta a la pantalla de la ilustracion 39 donde aparecerá el nombre del destinatario y el email (si tiene uno asignado).

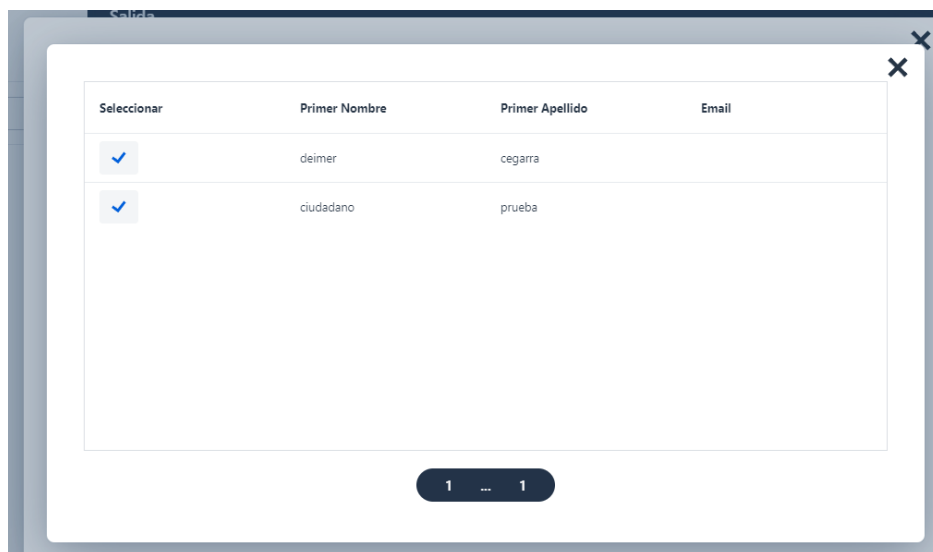


Ilustración 40 Seleccionar destinatario

3.5.1.4. Radicación de entrada

En las ilustraciones 41, 42 y 43 se puede observar el formulario de radicación de entrada y sus diferentes opciones. Los campos obligatorios son aquellos que tienen un asterisco rojo al igual que el formulario de radicación de salida. Hay pocas diferencias entre este formulario y el de salida como se puede observar, igualmente con las opciones luego de radicar, hay menos botones.

Entrada

CIUDADANO

Tipo Identificación* Cedula Identificación* 1004911853

Primer Nombre* deimer Segundo Nombre

Primer Apellido* cegarra Segundo Apellido

> Otros datos del Ciudadano

ANTECEDENTE

Radicado

INFORMACION A RADICAR

Fecha Doc Ref 14/11/2022 Doc Referencia

Medio Recepción* Personal Tipo Documento* tutela

Dignatario deimer joel cegarra Folios 0

Ilustración 41 Formulario Radicación de entrada parte 1

Entrada

Medio Recepción* Personal Tipo Documento* tutela

Dignatario deimer joel cegarra Folios 1

Continente* America País* COLOMBIA

Departamento* ATLANTICO Municipio* SOLEDAD

Asunto* prueba de radicación de entrada

Desc Anexos

Requiere Respuesta ☒ Tiempo para Respuesta 5

Fecha Fija Respuesta

ASIGNACION

Dependencia* 000-Administradores Funcionario* prueba correo

Limpiar Radicar

Ilustración 42 Formulario Radicación de entrada parte 2

Ilustración 43 Opciones luego de generar un radicado de entrada

NOTA: los demás tipos de radicados que se crean van a tener las mismas opciones y mismo formulario que los radicados de entrada.

3.5.1.5. *CRUD de usuarios*

En la ilustración 44 podemos observar la pantalla principal de los usuarios. En esta pantalla se listan todos los usuarios y se pagan de a 25 registros.

Username	Nombre	Dependencia	Cargo	Estado	Email
dev	prueba correo	nn	Administrador	Suspendido	deimercegarra.ope@gm...
ADMON	Administrador Sistema	Administradores	Administrador	Activo	admin@siepdcc.com
prueba	prueba correo	Administradores	Administrador	verificando email	deimercegarra.ope@gm...
joel	prueba correo	Administradores	Administrador	Activo	deimercegarra.ope@gm...

Ilustración 44 Listado usuarios

Para crear un nuevo registro debemos dar clic sobre el boton ‘Nuevo’ y nos aparecera la pantalla de la ilustracion 45. En este formulario podemos observar que existen campos obligatorios y campos no obligatorios. La dependencia, el cargo, el funcionario y el rol tambien son obligatorios.

The screenshot displays a web form titled 'Formulario usuarios' with a close button (X) in the top right corner. The form is organized into a grid of input fields:

- Username**: Text input field.
- Dependencia**: Text input field with a selection icon (blue square with a magnifying glass).
- Cargo**: Text input field with a selection icon.
- Funcionario**: Text input field with a selection icon.
- Perfil**: Dropdown menu with 'Estandar' selected.
- Identificación**: Text input field.
- Primer Nombre**: Text input field.
- Segundo Nombre**: Text input field.
- Primer Apellido**: Text input field.
- Segundo Apellido**: Text input field.
- Fecha de Nacimiento**: Text input field with a calendar icon.
- Email**: Text input field.
- Rol**: Text input field with a selection icon.
- Fecha Vencimiento Contrato**: Text input field with a calendar icon.
- Celular Enviar SMS**: Text input field.
- Permitir Sms**: Dropdown menu with 'No' selected.
- Nivel de Seguridad**: Dropdown menu with '1' selected.
- Teléfono Empresa**: Text input field.
- Extensión Telefónica**: Text input field.
- Ubicación de la persona en la empresa**: Text input field.
- Radicación Directa**: Dropdown menu with 'No' selected.
- Idioma**: Dropdown menu with 'Español' selected.

A blue 'Guardar' (Save) button is located at the bottom left of the form.

Ilustración 45 Formulario usuarios

Para seleccionar una dependencia debemos dar clic sobre el botón que se encuentra al lado del campo, luego saldrá la pantalla de la ilustración 46, para seleccionar una dependencia solo debemos dar clic en el botón de la columna ‘Seleccionar’ que se observa en la ilustración 46.

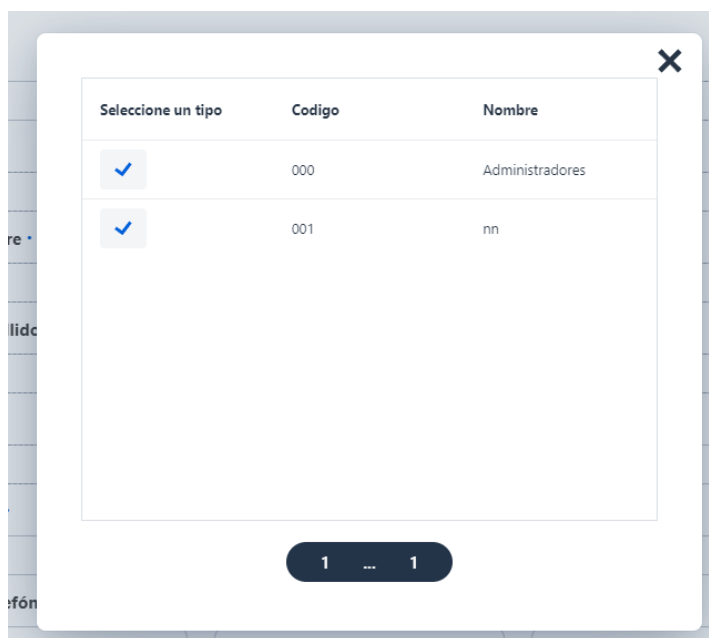


Ilustración 46 Seleccionar dependencia

Para seleccionar un cargo debemos dar clic sobre el botón que se encuentra al lado del campo, luego saldrá la pantalla de la ilustración 47, para seleccionar un cargo solo debemos dar clic en el botón de la columna ‘Seleccionar’ que se observa en la ilustración 47.

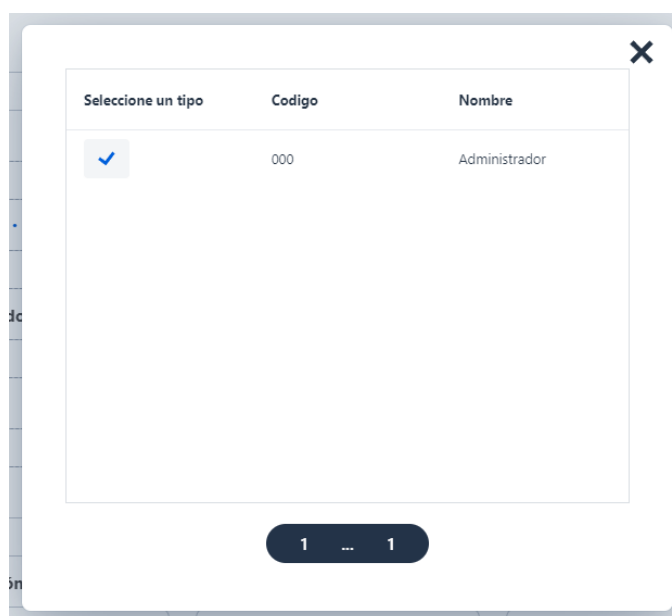


Ilustración 47 Seleccionar cargo

Para seleccionar un funcionario debemos dar clic sobre el botón que se encuentra al lado del campo, luego saldrá la pantalla de la ilustración 48, para seleccionar un funcionario solo debemos dar clic en el botón de la columna ‘Seleccionar’ que se observa en la ilustración 48.

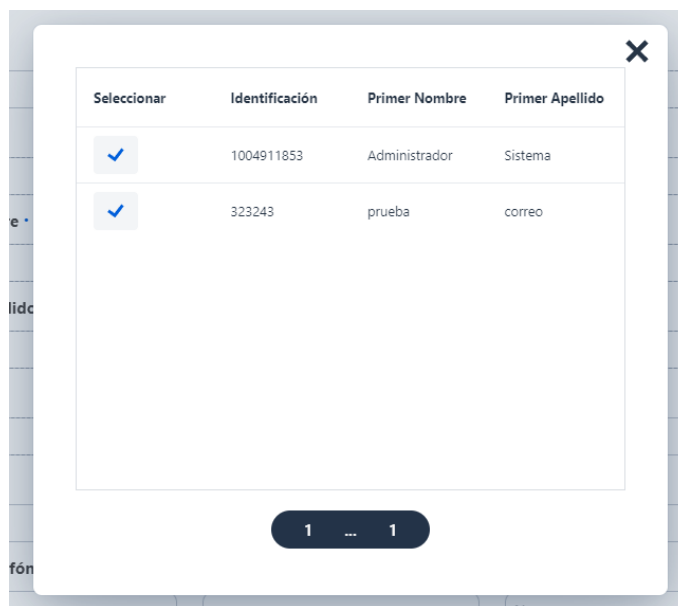


Ilustración 48 Seleccionar funcionario

Para seleccionar un rol debemos dar clic sobre el botón que se encuentra al lado del campo, luego saldrá la pantalla de la ilustración 49, para seleccionar un rol solo debemos dar clic en el botón de la columna ‘Seleccionar’ que se observa en la ilustración 49.

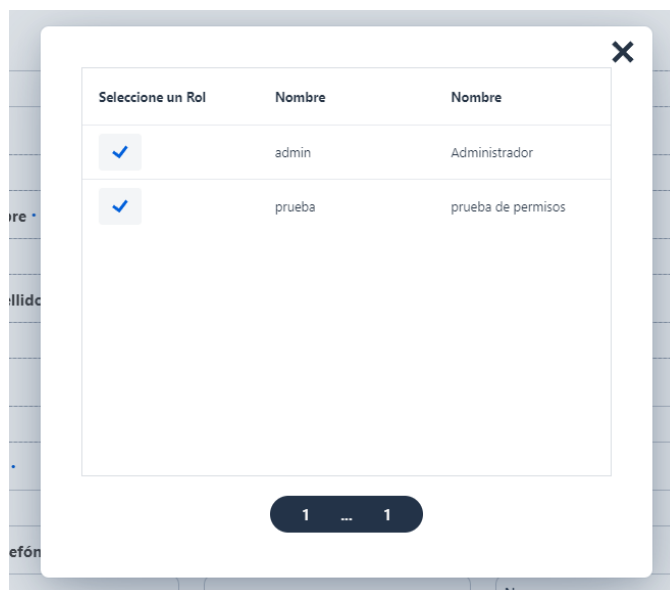


Ilustración 49 Seleccionar rol

Para ver el detalle de un registro solo debemos dar clic sobre la fila del registro que queramos ver a detalle en la ilustracion 44, luego saldrá la pantalla de la ilustracion 50.

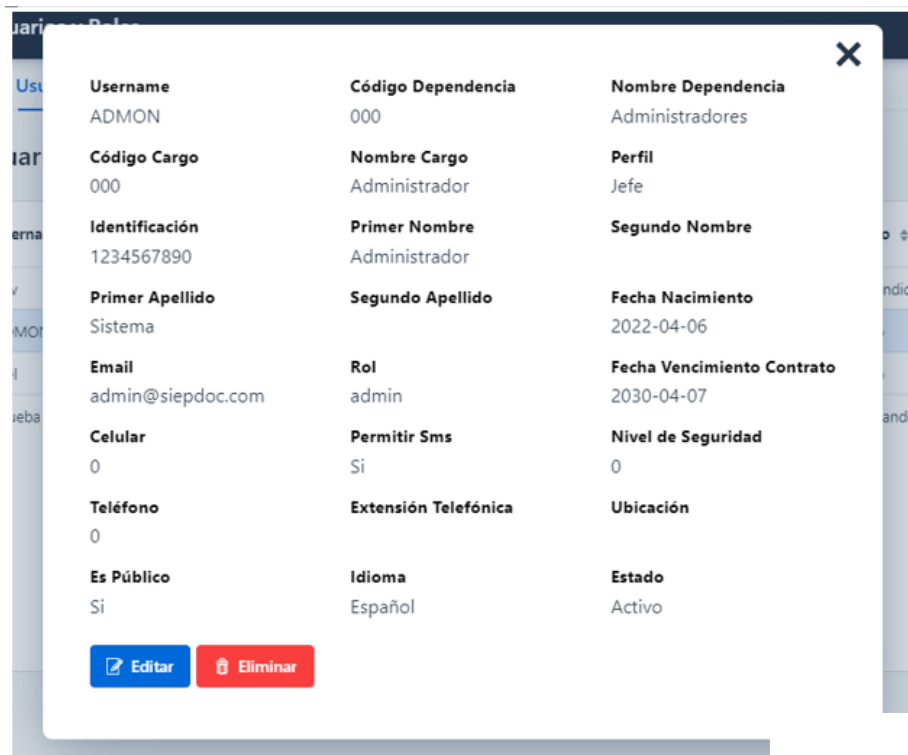


Ilustración 50 Detalle usuarios

Para actualizar un registro debemos dar clic en el boton ‘Editar’ de la ilustracion 50, luego saldrá la pantalla de la ilustracion 51. Para eliminar un usuario debemos dar clic en el boton ‘Eliminar’, esto nos mostrará una pantalla donde debemos confirmar que deseamos eliminar el usuario.

Username	Dependencia	Cargo
prueba2	000-Administradores	000-Administrador
Funcionario	Perfil	Identificación
Administrador Sistema	Estandar	1004911853
Primer Nombre	Segundo Nombre	Primer Apellido
Administrador		Sistema
Segundo Apellido	Fecha de Nacimiento	Email
	5/4/2022	admin@siepdoc.com
Rol	Fecha Vencimiento Contrato	Celular Enviar SMS
admin	20/1/2023	
Permitir Sms	Nivel de Seguridad	Teléfono Empresa
No	1	
Extensión Telefónica	Ubicación de la persona en la empresa	Radicación Directa
		No
Idioma		
Español		

[Guardar](#)

Ilustración 51 Actualizar usuarios

3.5.1.5.1. Creación de contraseña luego de crear el usuario

Para la creación de la contraseña del usuario debemos crear el usuario, luego de crear exitosamente el usuario, se envía un email como se muestra en la ilustración 52. En este email se adjunta el link con el cual, el usuario puede acceder a la aplicación para crear su contraseña. Este link tiene una duración de 2 horas.

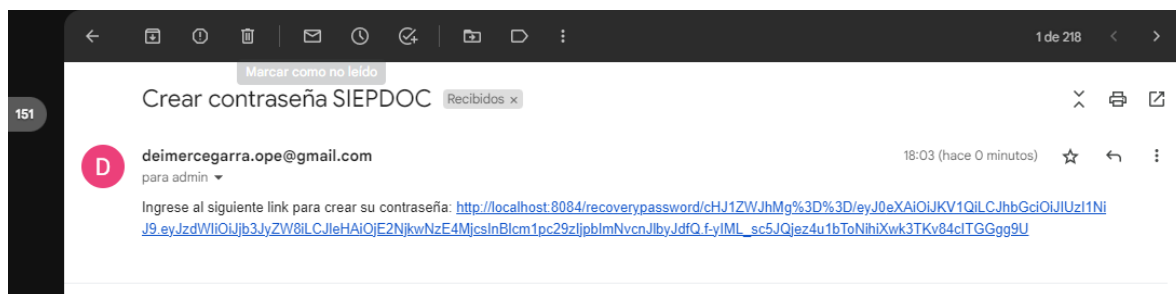


Ilustración 52 Link para crear contraseña

Luego de acceder al link mencionado en la ilustración 52, saldrá la pantalla de la ilustración 53, esta pantalla consta de un formulario donde debemos escribir nuestra contraseña. En el recuadro negro se ubica el logo de la entidad.

Ilustración 53 Formulario crear contraseña

Luego de dar clic en el botón ‘Crear Contraseña’ de la ilustración 53, aparece la pantalla de la ilustración 54, la cual nos indica que nuestra contraseña ha sido creada. También tiene la

opción de ‘Iniciar Sesión’ la cual nos dirige al login de la aplicación. En el recuadro negro se ubica el logo de la entidad.



Su nueva contraseña ha sido creada.

Iniciar Sesión 

Ilustración 54 Contraseña creada

3.5.1.5.2. Recuperación de contraseña de usuario

Para recuperar la de usuario, debemos ubicarnos en el login de la aplicación y dar clic en la opción ‘¿Olvidaste tu contraseña?’ como se muestra en la ilustración 55.

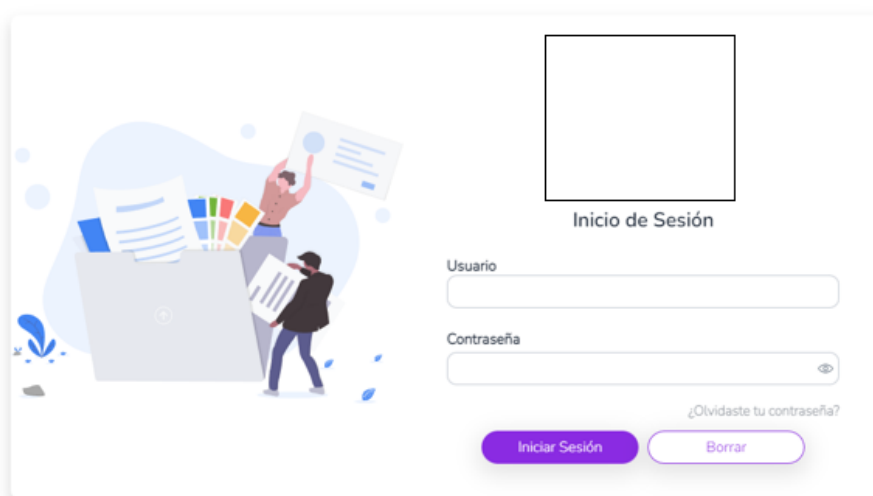
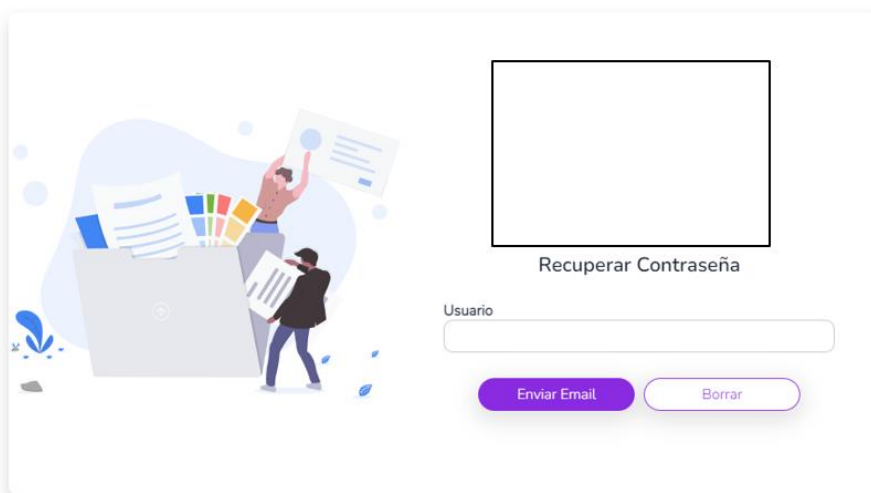


Ilustración 55 Recuperar contraseña

Para recuperar la contraseña solo debemos ingresar nuestro usuario y dar clic en el botón ‘Enviar Email’ como se muestra en la ilustración 56, esto hará que se envíe un correo similar al de la ilustración 52. En el recuadro negro se ubica el logo de la entidad.



Recuperar Contraseña

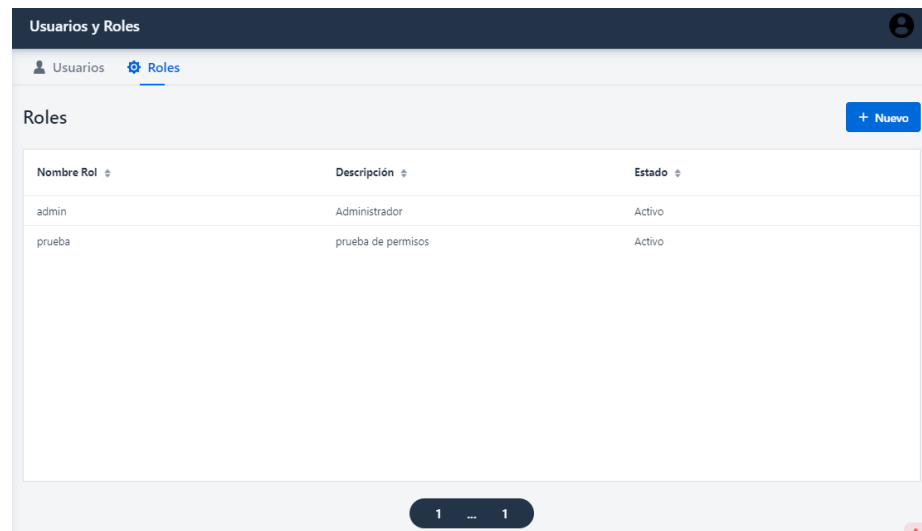
Usuario

Enviar Email Borrar

Ilustración 56 Formulario para enviar email

3.5.1.6. CRUD de Roles y Permisos

En la ilustración 57 podemos observar la pantalla principal de los roles. En esta pantalla se listan todos los roles y se paganan de a 25 registros.



Nombre Rol	Descripción	Estado
admin	Administrador	Activo
prueba	prueba de permisos	Activo

Ilustración 57 Listado roles

Para crear un nuevo registro debemos dar clic sobre el boton ‘Nuevo’ y nos aparecera la pantalla de la ilustracion 58. En este formulario podemos observar el campo ‘Nombre’ que es obligatorio, el campo ‘descripcion’ y el campo activo que indica el estado del registro. Tambien se puede observar la seccion de permisos.

Información Rol

Nombre • Descripción

☒ Activo

Permisos

- > Permisos Generales
- > Permisos Editor Documento Interno
- > Permisos Editor Respuesta Salida
- > Permisos de Seguridad
- > Permisos Tipos de Radicados
- > Otros Permisos Especiales
- > Permisos Tablas SCGU
- > Permisos Tablas SCC
- > Permisos Tablas SCGD
- > Permisos Estadísticas

Guardar

Ilustración 58 Formulario roles

En la ilustración 59 podemos observar los permisos sobre los tipos de radicaciones que tiene la entidad, acompañados de su código y sus niveles de permiso. Los tipos de radicación de salida y entrada son del sistema y no pueden eliminarse.

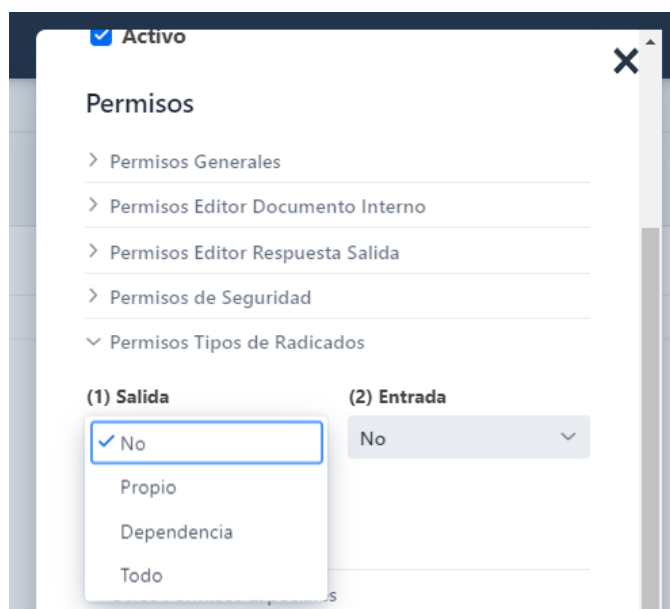


Ilustración 59 Permisos tipos de radicados

En la ilustración 60 podemos observar los permisos sobre los CRUD de gestión de usuarios y sus niveles de permiso.

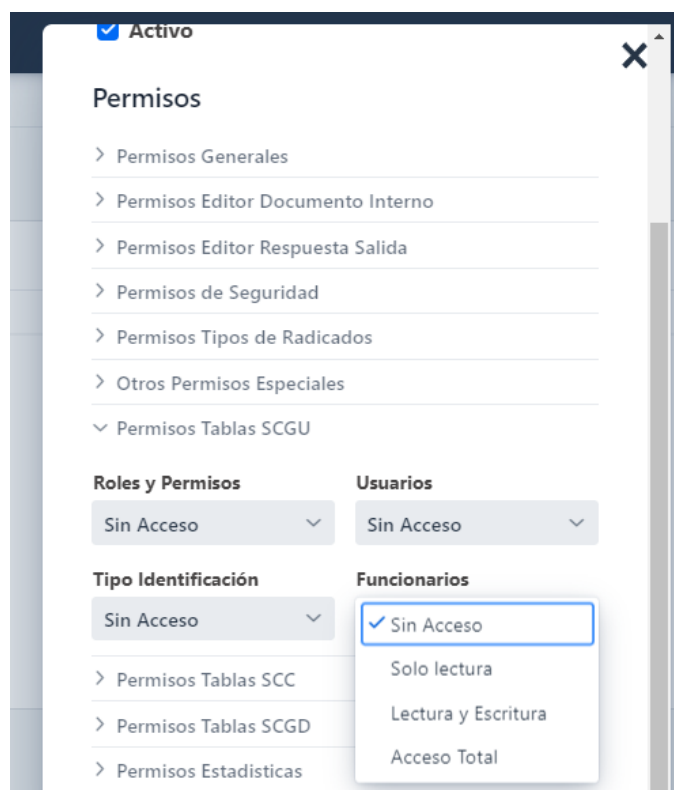


Ilustración 60 Permisos CRUD gestión de usuarios

En la ilustración 61 podemos observar los permisos sobre los CRUD de configuración y sus niveles de permiso.

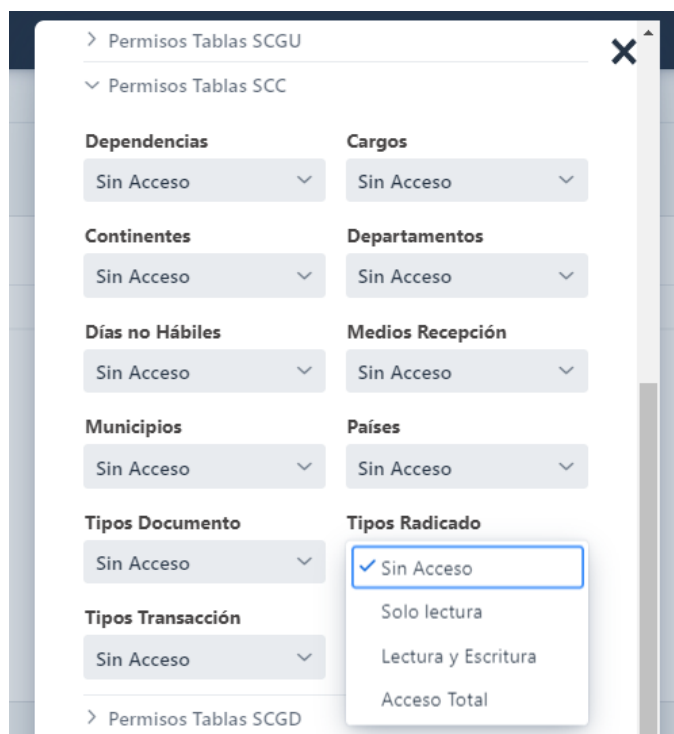


Ilustración 61 Permisos CRUD de configuración

En la ilustración 62 podemos observar los permisos sobre los CRUD de gestión de documentos y sus niveles de permiso.

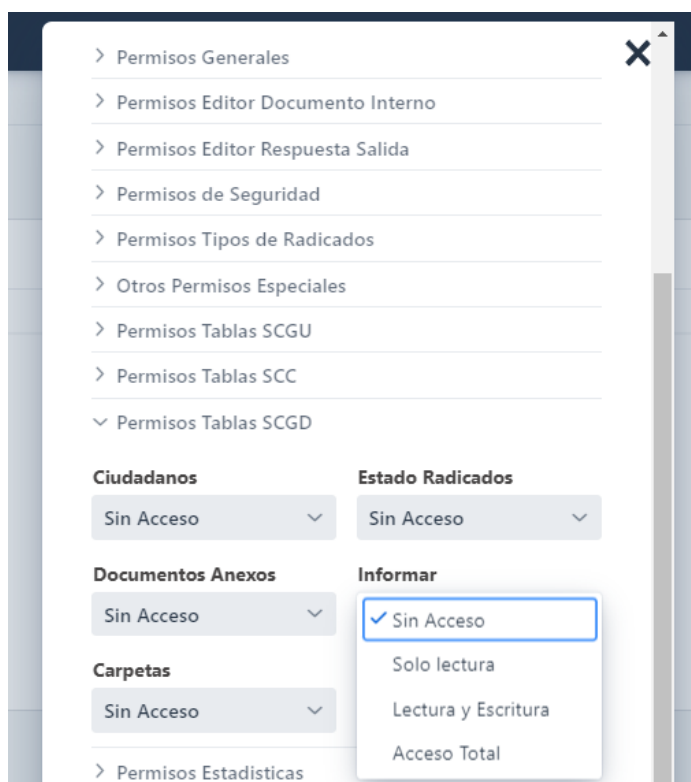


Ilustración 62 Permisos CRUD gestión de documentos

En la ilustración 63 podemos observar los permisos sobre los reportes y sus niveles de permiso.

✓ Activo

Permisos

- > Permisos Generales
- > Permisos Editor Documento Interno
- > Permisos Editor Respuesta Salida
- > Permisos de Seguridad
- > Permisos Tipos de Radicados
- > Otros Permisos Especiales
- > Permisos Tablas SCGU
- > Permisos Tablas SCC
- > Permisos Tablas SCGD
- ✓ Permisos Estadísticas

Correspondencia Entrada

No

Correspondencia Salida

✓ No

Propio

Dependencia

Todo

Radicados por Usuario ...

No

Guardar

Ilustración 63 Permisos estadísticas

Para ver el detalle de un rol solo debemos dar clic sobre la fila del registro que queramos ver a detalle en la ilustracion 57, luego saldrá la pantalla de la ilustracion 64.

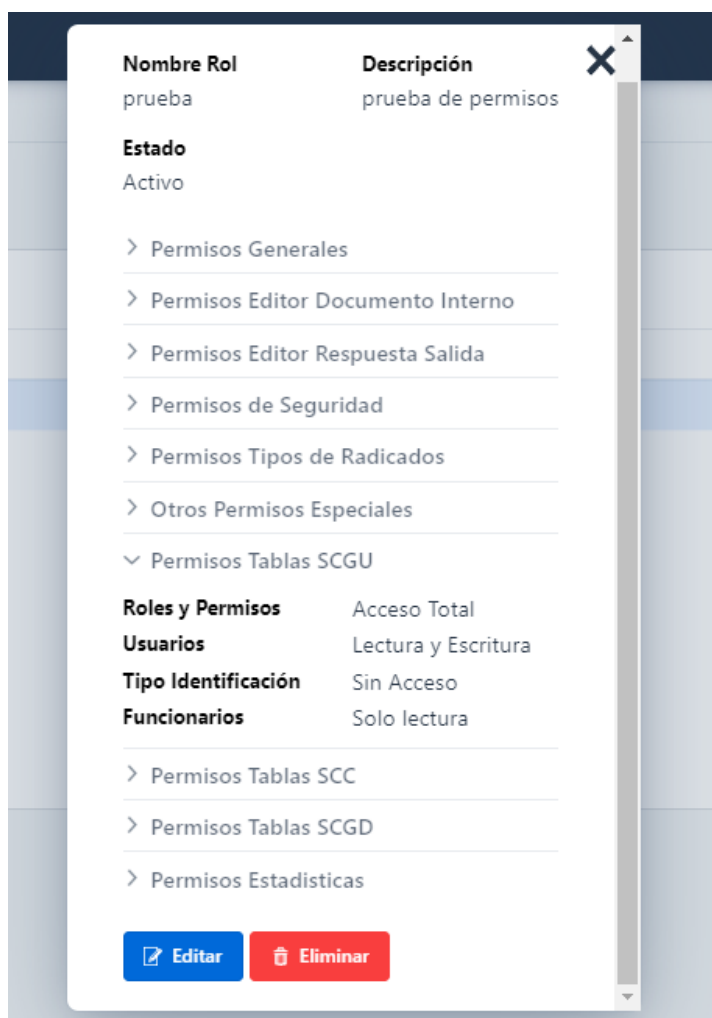


Ilustración 64 Detalle roles

Para actualizar un rol debemos dar clic en el boton ‘Editar’ de la ilustracion 64, luego saldrá la pantalla de la ilustracion 65. Para eliminar un rol debemos dar clic en el boton ‘Eliminar’, esto nos mostrará una pantalla donde debemos confirmar que deseamos eliminar el rol.

Información Rol

Nombre • prueba

Descripción prueba de permisos

☒ **Activo**

Permisos

- > Permisos Generales
- > Permisos Editor Documento Interno
- > Permisos Editor Respuesta Salida
- > Permisos de Seguridad
- > Permisos Tipos de Radicados
- > Otros Permisos Especiales
- ▼ Permisos Tablas SCGU

Roles y Permisos Acceso Total ▼

Usuarios Lectura y Escritura ▼

Tipo Identificación Sin Acceso ▼

Funcionarios Solo lectura ▼

Ilustración 65 Actualizar roles

3.5.1.7. Estructura de carpetas

Como se menciona anteriormente en el capítulo de arquitectura, para este microservicio del FrontEnd se utiliza la arquitectura MVP. En las ilustraciones 66 y 67 podemos observar la estructura de carpetas del microservicio, basada en la arquitectura y en proyectos de vaadin Starters de Vaadin.

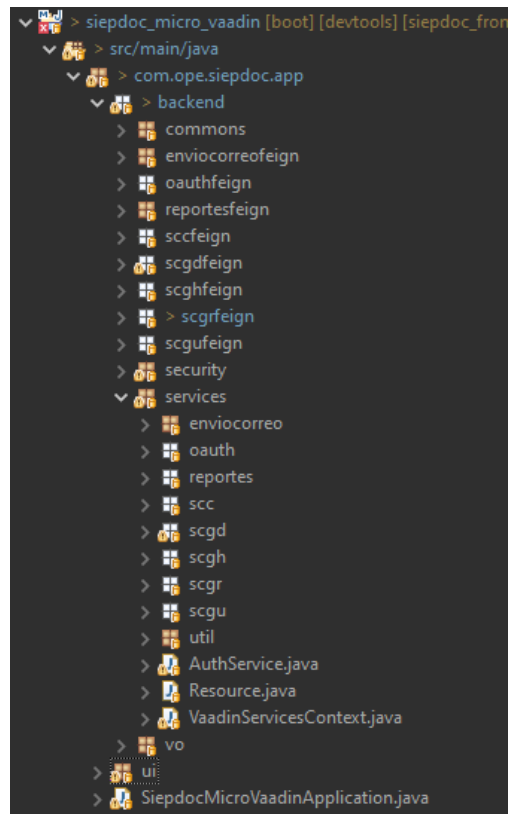


Ilustración 66 Estructura de carpetas Siepdoc_micro_vaadin parte 1

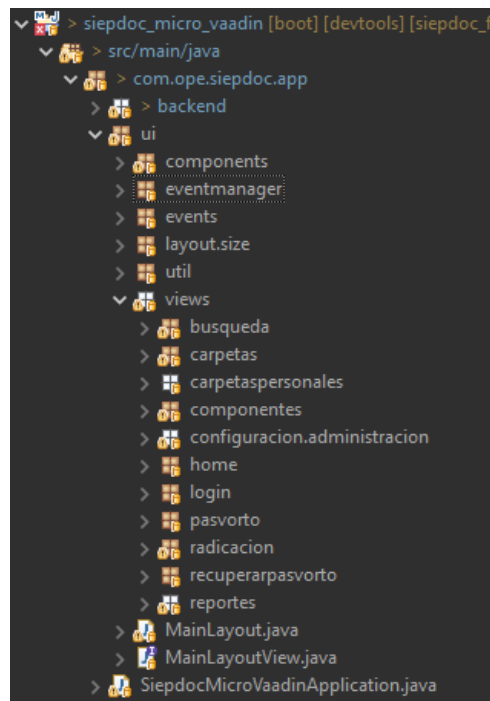


Ilustración 67 Estructura de carpetas Siepdoc_micro_vaadin parte 2

3.5.1.8. Eventos y comunicación entre clases

Para la comunicación entre clases se utilizan eventos, la librería utilizada es EventBus de Guava. En la ilustración 68 se puede observar un ejemplo de un presenter, allí se puede ver que hay un evento escuchando un post que se lanza en la vista, luego el presenter consume los services que traen la información del BackEnd. Finalmente, el presenter envía la información con un post hacia la vista donde está el evento escuchando para mostrar la información al usuario. De esta manera se implementa la arquitectura MVP.

```
@Component
@UIScope
public class ListUsuariosPresenter extends ViewPresenter {

    UsuariosService servicioUsuarios = VaadinServicesContext.getApplicationContext().getBean(UsuariosService.class);

    @Subscribe
    public void findIdUsuarios(ListUsuariosPresenterEvent.GotoPresenterFindIdUsuarios event) {
        ScguUsuariosVo usuarioVo = (ScguUsuariosVo)event.getData();
        usuarioVo= servicioUsuarios.read(usuarioVo.getId(),
            Utils.PRE_TOKEN + VaadinSession.getCurrent().getAttribute(ScguUserLoginVo.class).getToken()).getVo();

        eventManager.getEventBus().post(new ReadUsuariosPresenterEvent.GotoPresenterFindIdUsuarioRoles(this,usuarioVo.getId(),
            eventManager.getEventBus().post(new ReadUsuariosPresenterEvent.GotoPresenterCapturarUsuario(this,usuarioVo));
    }
}
```

Ilustración 68 Presenter de la clase ListUsuariosViewImpl

3.6. Desarrollo BackEnd

En la lógica del negocio tenemos diferentes microservicios los cuales tienen distribuidas sus funcionalidades de acuerdo a los esquemas de bases de datos, cada esquema se trabaja con un microservicio. Además, tenemos microservicios encargados de las configuraciones y comunicaciones entre ellos y la conexión a las bases de datos. También tenemos un microservicio que se encarga de autenticar los usuarios de nuestra aplicación FrontEnd y un microservicio encargado de darnos seguridad y enrutar las peticiones que se realicen a nuestros EndPoints. Por último, también tenemos librerías comunes utilizadas para almacenar la definición de los permisos de usuario y nuestros objetos utilizados para comunicar el FrontEnd

con nuestro BackEnd. En la ilustración 69 podemos observar la lista de los microservicios en el IDE utilizado para el desarrollo, Eclipse IDE.

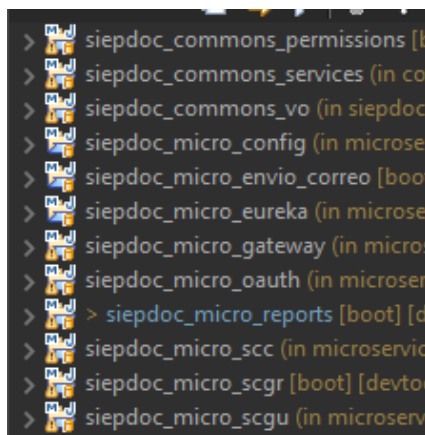


Ilustración 69 Visión general de los proyectos en Eclipse IDE

3.6.1. Siepdoc_commons_permissions

Esta es una librería propia de la aplicación SIEPDOCC, la cual se encarga de almacenar los permisos de la aplicación en clases por esquema de base de datos, y se define un permiso por cada tabla y por funcionalidades. Además, se manejan por niveles estos permisos dependiendo de cada funcionalidad. A continuación, veremos la definición de estos permisos. En la ilustración 70 podemos observar la estructura de paquetes de la librería.

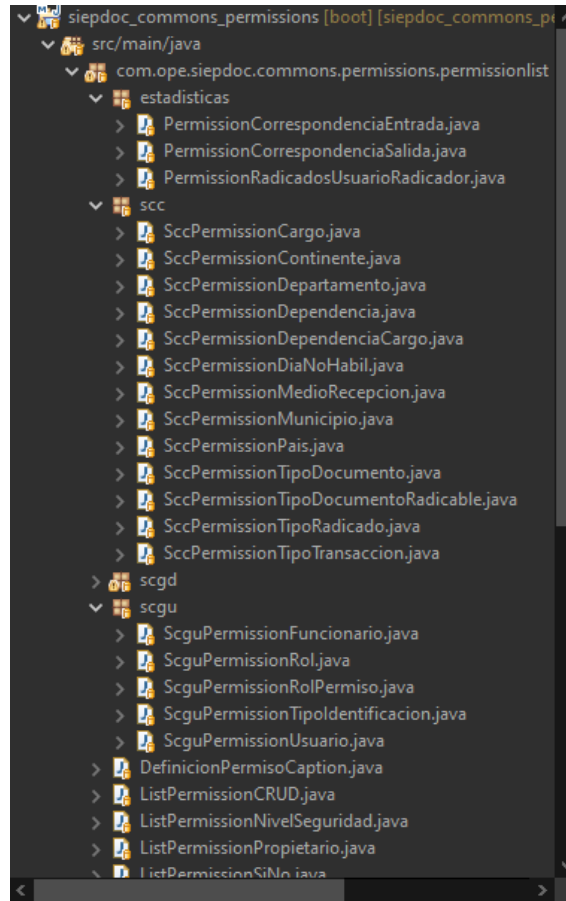


Ilustración 70 Estructura de paquetes siepdoc_commons_permissions

En la ilustración 71 podemos observar la definición de los niveles de permisos para CRUD de tablas, estos se almacenan como variables estáticas.

```
public class ListPermissionCRUD {

    public static String ACCESSTOTAL="3";
    public static String ONLYREAD="1";
    public static String READWRITE="2";
    public static String NOACCESS="0";

    public static String KEYACCESSTOTAL="Acceso Total";
    public static String KEYONLYREAD="Solo lectura";
    public static String KEYREADWRITE="Lectura y Escritura";
    public static String KEYNOACCESS="Sin Acceso";
}
```

Ilustración 71 Definición de niveles de permisos para CRUD de tablas

En la ilustración 72 podemos observar la definición de los permisos para el CRUD de usuarios, estos se almacenan como variables estáticas. Los permisos se almacenan como un

String, están compuestos por el nombre seguido de dos puntos (:) y el nivel del permiso. De esta manera se maneja los permisos para todos los CRUD en el FrontEnd.

```
public class ScguPermissionUsuario {

    public static String SCGU_PERMISSION_NAME_USUARIO="scgu_permission_usuario";

    public static String SCGU_PERMISSION_ACCESSTOTAL_USUARIO=SCGU_PERMISSION_NAME_USUARIO+" : "+ListPermissionCRUD.ACCESSTOTAL;
    public static String SCGU_PERMISSION_ONLYREAD_USUARIO=SCGU_PERMISSION_NAME_USUARIO+" : "+ListPermissionCRUD.ONLYREAD;
    public static String SCGU_PERMISSION_READWRITE_USUARIO=SCGU_PERMISSION_NAME_USUARIO+" : "+ListPermissionCRUD.READWRITE;
    public static String SCGU_PERMISSION_NOACCESS_USUARIO=SCGU_PERMISSION_NAME_USUARIO+" : "+ListPermissionCRUD.NOACCESS;
```

Ilustración 72 Definición de permisos CRUD de usuarios

En la ilustración 73 podemos observar la definición de los niveles de permisos para Radicación y Estadísticas, estos se almacenan como variables estáticas.

```
public class ListPermissionPropietario {

    public static String PROPIO="Propio";
    public static String DEPENDENCIA="Dependencia";
    public static String TODO="Todo";
    public static String NOACCESS="No";
```

Ilustración 73 Definición de niveles de permisos para Radicación y Estadísticas

En la ilustración 74 podemos observar la definición de los permisos para el formulario de Radicación, estos se almacenan como variables estáticas. Los permisos se almacenan como un String, están compuestos por el nombre seguido de dos puntos (:) y el nivel del permiso. De esta manera se maneja los permisos para los formularios de reportes (modulo Estadísticas) en el FrontEnd.

```
public class PermisosTiposRadicados {

    public static String SCGD_PERMISSION_SALIDA="scgd_permission_salida";
    public static String SCGD_PERMISSION_ENTRADA="scgd_permission_entrada";

    public static String SCGD_PERMISSION_SALIDA_PROPIO=SCGD_PERMISSION_SALIDA+" : "+ListPermissionPropietario.PROPIO;
    public static String SCGD_PERMISSION_SALIDA_DEPENDENCIA=SCGD_PERMISSION_SALIDA+" : "+ListPermissionPropietario.DEPENDENCIA;
    public static String SCGD_PERMISSION_SALIDA_TODO=SCGD_PERMISSION_SALIDA+" : "+ListPermissionPropietario.TODO;
    public static String SCGD_PERMISSION_SALIDA_NOACCESS=SCGD_PERMISSION_SALIDA+" : "+ListPermissionPropietario.NOACCESS;
```

Ilustración 74 Permisos para Radicación

3.6.2. *Siepdoc_commons_vo*

Esta es una librería propia de la aplicación SIEPDOC, la cual se encarga de almacenar clases y objetos, para la transferencia de datos que se utilizan en los diferentes microservicios.

En la ilustración 75 podemos observar la estructura de paquetes de la librería.

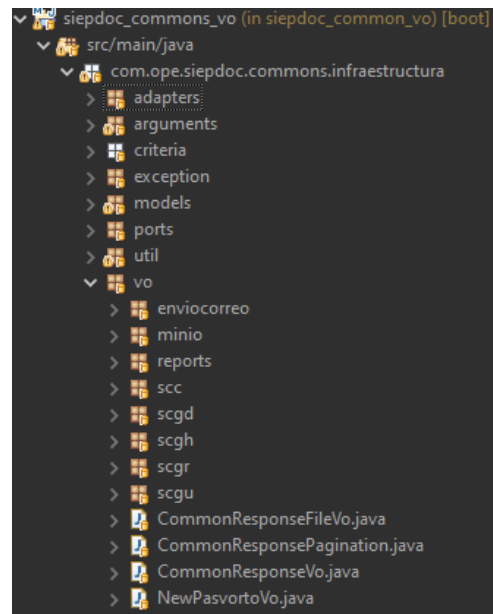


Ilustración 75 Estructura de paquetes siepdoc_commos_vo

3.6.3. *Siepdoc_commons_services*

Esta es una librería propia de la aplicación SIEPDOC, la cual se encarga de almacenar clases comunes que se utilizan principalmente en los CRUD de los diferentes microservicios, esto ayuda a minimizar el código repetitivo en la aplicación. En la ilustración 76 podemos observar la estructura de paquetes de la librería.

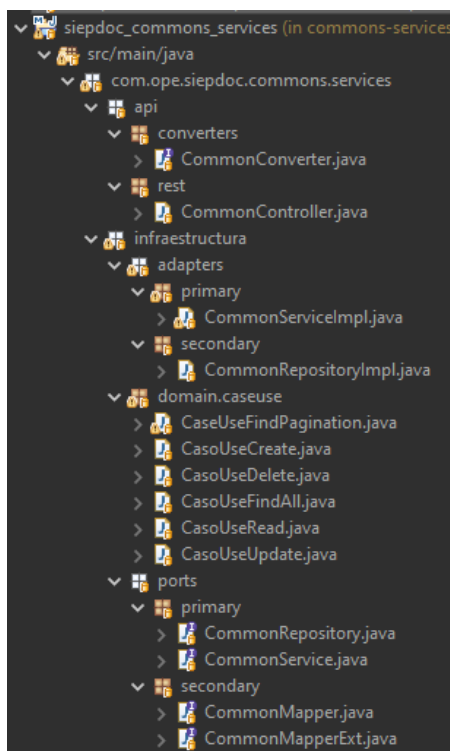


Ilustración 76 Estructura de paquetes siepdoc_commons_services

En la ilustración 77 podemos observar la clase común que se utiliza para los controller, esta contiene el CRUD que se va a utilizar en los controller de todas las tablas básicas. De esta manera se crearon las demás clases comunes de las API para los CRUD de tablas básicas de SIEPDOG como se muestra en la ilustración 76.

```

public class CommonController<E, V, C extends CommonConverter<E, V>, S extends CommonService<E,SC>,SC extends SearchCriteria> {
    @Autowired
    protected S commonService;

    @Autowired
    protected C commonConverter;

    @Operation(summary = "Guardar")
    @ApiResponses(value={
        @ApiResponse(responseCode = "200", description = "Registro exitoso"),
        @ApiResponse(responseCode = "400", description = "Petición no válida",
            content = @Content)})
    @PostMapping
    public ResponseEntity<?> create (@RequestBody V vo) {
        CommonResponseVo<V> resp;
        Long id;
        try {
            E entity=commonConverter.VoToE(vo);
            id = commonService.insertAndReturnKey(entity);
            resp = new CommonResponseVo<V>("Se Guardo correctamente", vo, id,true);
        } catch (Exception e) {
            resp = new CommonResponseVo<V>(e.getMessage(), vo,false);
            return new ResponseEntity<CommonResponseVo<V>>(resp, HttpStatus.ACCEPTED);
        }

        return new ResponseEntity<CommonResponseVo<V>>(resp, HttpStatus.OK);
    }
}

```

Ilustración 77 Clase común para los controller

3.6.4. Siepdoc_micro_eureka

Este microservicio se encarga de localizar todos los demás microservicios y registra todas las instancias junto a su estado en el servidor Netflix Eureka Server, de esta manera el servidor de Eureka mantiene en su registro la información de todos los microservicios del ecosistema. Eureka nos permite una fácil localización de nuestros microservicios mediante el nombre asignado a cada uno, solo debemos configurar los microservicios como cliente de Eureka y así registrarse al servidor, esto nos facilita la comunicación entre microservicios y permite mantenerlos monitoreados. En la ilustración 78 podemos observar cómo se configura el servidor de Eureka.

```

@EnableEurekaServer
@SpringBootApplication
public class MicroserviciosEurekaApplication {

```

Ilustración 78 Configuración Netflix Eureka Server

3.6.5. *Siepdoc_micro_config*

Este microservicio se encarga principalmente de externalizar nuestras configuraciones, nos ayuda para cargar y gestionar nuestros archivos de configuración de manera centralizada. Esto se hace mediante el servidor de Spring Cloud Config Server. En la ilustración 79 podemos observar cómo se configura el servidor de Config Server.

```
@EnableConfigServer
@SpringBootApplication
public class MicroserviciosConfigApplication {
```

Ilustración 79 Configuración Config Server

Por último, debemos agregar en el archivo de configuraciones de nuestro microservicio, la URL al repositorio de nuestros archivos de configuración como se muestra en la ilustración 80.

```
spring.cloud.config.server.git.uri=https://github.com/DeimerJC/config-server-repository-siepdoc.git
#spring.cloud.config.server.git.search-paths=governacion
```

Ilustración 80 Conexión al repositorio de archivos de propiedades

3.6.6. *Siepdoc_micro_scgu*

En este microservicio se encuentra la lógica del negocio relacionada con las tablas del esquema de base de datos “scgu”. En la ilustración 81 podemos observar la estructura de paquetes del microservicio.

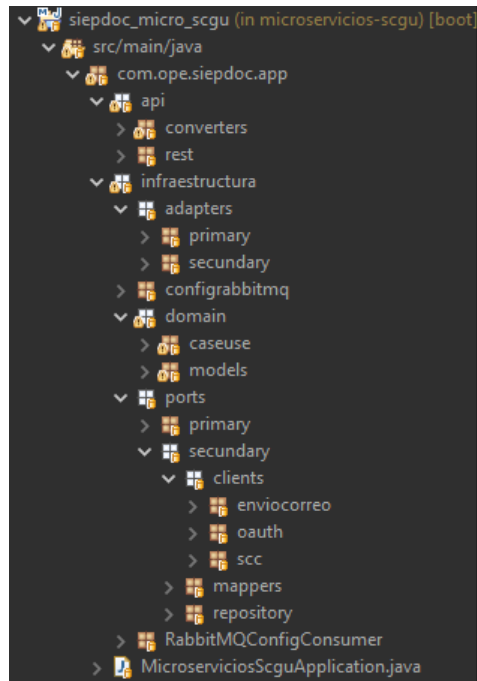


Ilustración 81 Estructura de paquetes Siepdoc_micro_scgu

A continuación, observaremos en detalle cada una de las capas del microservicio para un solo CRUD, cabe destacar que todos los microservicios contienen esta misma estructura, por lo tanto, no entraremos en detalle en los demás microservicios.

En la ilustración 82 podemos observar el controller para la tabla usuarios, estos se almacenan en el paquete “rest”. En esta clase se extiende de la clase común “CommonController” la cual contiene los servicios que ofrece las API como vimos anteriormente en la librería “Siepdoc_commons_services”. Solo debemos enviar como parámetros de la clase, el modelo de nuestra tabla, el Value Object (VO), el converter, el servicio de la tabla y el SearchCriteria para los criterios de búsquedas avanzadas.

```

@RestController
@RequestMapping("/api/usuarios")
public class ScguUsuariosController
extends CommonController<ScguUsuarios, ScguUsuariosVo, ScguUsuariosConverter, ScguUsuariosService, ScguUsuariosSearchCriteria>{

    @Autowired
    private ScguUsuariosService scguUsuariosService;

    @Override
    @PostMapping
    public ResponseEntity<?> create(@RequestBody ScguUsuariosVo vo) {

        CommonResponseVo<ScguUsuariosVo> resp;
        Long id;
        try {
            ScguUsuarios entity=commonConverter.VoToE(vo);
            id = scguUsuariosService.insertAndReturnKey(entity,vo.getLinkCrearPasvorto());
            resp = new CommonResponseVo<ScguUsuariosVo>("Se Guardo correctamente", vo, id,true);
        } catch (Exception e) {
            resp = new CommonResponseVo<ScguUsuariosVo>(e.getMessage(), vo,false);
            return new ResponseEntity<CommonResponseVo<ScguUsuariosVo>>(resp, HttpStatus.ACCEPTED);
        }

        return new ResponseEntity<CommonResponseVo<ScguUsuariosVo>>(resp, HttpStatus.OK);
    }
}

```

Ilustración 82 Controller CRUD tabla usuarios

En la ilustración 83 podemos observar la interface para los converter, estos se almacenan en el paquete “converters”, con la ayuda de la librería MapStruct generamos la implementación, lo único que necesitamos es anotar la interfaz con @Mapper de la librería, y generar el jar del microservicio y esta creará las implementaciones en el paquete “target” del proyecto. Estos converters nos permite mapear los VO que llegan a las API, hacia los modelos y viceversa.

```

3 @Mapper
3 public interface ScguUsuariosConverter
1 extends CommonConverter<ScguUsuarios, ScguUsuariosVo> {
2
3 }

```

Ilustración 83 Interface del converter para la tabla usuarios

En la ilustración 84 podemos observar la implementación de los converter para la tabla usuarios, estos se almacenan en el paquete “convertersimpl” dentro del paquete “converters”, de esa manera la librería crea estos convertidores, algo que nos ayuda a minimizar mucho trabajo repetitivo y extenso.

```

@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2022-04-01T13:02:44-0500",
    comments = "version: 1.4.2.Final, compiler: javac, environment: Java 17.0.1 (Oracle Corporation)"
)
@Component
public class ScguUsuariosConverterImpl implements ScguUsuariosConverter {

    @Override
    public ScguUsuarios VoToE(ScguUsuariosVo vo) {
        if ( vo == null ) {
            return null;
        }

        ScguUsuarios scguUsuarios = new ScguUsuarios();

        scguUsuarios.setId( vo.getId() );
        scguUsuarios.setUsername( vo.getUsername() );
        scguUsuarios.setSccDependenciasid( vo.getSccDependenciasid() );
        scguUsuarios.setSccCargosid( vo.getSccCargosid() );
        scguUsuarios.setScguFuncionariosid( vo.getScguFuncionariosid() );
        scguUsuarios.setPerfil( vo.getPerfil() );
        scguUsuarios.setIdentificacion( vo.getIdentificacion() );
        scguUsuarios.setPrimernombre( vo.getPrimernombre() );
        scguUsuarios.setSegundonombre( vo.getSegundonombre() );
        scguUsuarios.setPrimerapellido( vo.getPrimerapellido() );
        scguUsuarios.setSegundoapellido( vo.getSegundoapellido() );
    }
}

```

Ilustración 84 Implementación de converter de la tabla usuarios

En la ilustración 85 podemos observar la interface para los services de la tabla usuarios, estos se almacenan en el paquete “primary” dentro del paquete “ports”.

```

public interface ScguUsuariosService extends CommonService<ScguUsuarios,ScguUsuariosSearchCriteria>{

    public Long insertAndReturnKey(ScguUsuarios entity, String linkCrearPasvorto) throws Exception;

    public SccCargosVo findByIdCargo(Long id);

    public SccDependenciasVo findByIdDependencia(Long id);
}

```

Ilustración 85 Interface del service para la tabla usuarios

En la ilustración 86 podemos observar la implementación para el service de la ilustración 85, estos se almacenan en el paquete “primary” dentro del paquete “adapters”. Aquí se inyecta los casos de uso correspondiente para cada service.

```

@Service
public class ScguUsuariosServiceImpl
    extends CommonServiceImpl<ScguUsuarios, ScguUsuariosRepository, ScguUsuariosSearchCriteria>
    implements ScguUsuariosService {

    @Autowired
    private CaseUseListarCargo listarCargo;

    @Autowired
    private CaseUseListarDependencia listarDependencia;
}

```

Ilustración 86 Implementación del service para la tabla usuarios

En la ilustración 87 podemos observar el caso uso para crear usuarios, estos se almacenan en el paquete “caseuse”. Aquí se inyecta los repositorios correspondientes y se hacen también las validaciones necesarias como se puede observar.

```

@Component
public class CaseUseCreateUsuarios
    extends CasoUseCreate<ScguUsuarios, ScguUsuariosRepository, ScguUsuariosSearchCriteria> {

    @Override
    public Long create(ScguUsuarios entity) throws Exception {

        entity.setEstado("verificando email");
        entity.setPasvorto(" ");

        super.create(entity);

        return entity.getId();
    }

    @Override
    public void validarDatos(ScguUsuarios entity) throws Exception {
        if (entity != null) {
            if (entity.getUsername() != null) {
                if (entity.getUsername().isEmpty()) {
                    throw new Exception("El nombre de usuario es obligatorio");
                } else {
                    if (validarUserName(entity.getUsername())) {
                        throw new Exception("El nombre de usuario "+entity.getUsername()+" ya existe");
                    }
                }
            } else {
                throw new Exception("El nombre de usuario es obligatorio");
            }
        }
    }
}

```

Ilustración 87 Caso de uso create para tabla usuarios

Los repositorios trabajan de la misma manera que los services, pero comunicándose con los Mappers que nos proporciona MyBatis y los que creamos personalizados. La interface de los repositorios se almacena en el paquete “secondary” dentro del paquete “ports” y la implementación se almacena en el paquete “secondary” dentro del paquete “adapters”.

En la ilustración 88 podemos ver el Mapper que crea MyBatis con Mybatis Generator, una extensión instalada en Eclipse IDE. Este Mapper proporciona los métodos para los CRUD básicos de las tablas en código Java, además también se crean estos mismos Mapper en XML donde podemos realizar las consultas SQL.

```
@Mapper
public interface ScguUsuariosMapper extends CommonMapper<ScguUsuarios> {

    /**
     * This method was generated by MyBatis Generator. This method corresponds to the SQL statement
     * @mbg.generated Thu Aug 18 08:54:51 COT 2022
     */
    long countByExample(ScguUsuariosExample example);

    /**
     * This method was generated by MyBatis Generator. This method corresponds to the SQL statement
     * @mbg.generated Thu Aug 18 08:54:51 COT 2022
     */
    int deleteByExample(ScguUsuariosExample example);

    /**
     * This method was generated by MyBatis Generator. This method corresponds to the SQL statement
     * @mbg.generated Thu Aug 18 08:54:51 COT 2022
     */
    int deleteByPrimaryKey(Long id);

    /**
     * This method was generated by MyBatis Generator. This method corresponds to the SQL statement
     * @mbg.generated Thu Aug 18 08:54:51 COT 2022
     */
}
```

Ilustración 88 Mapper básico de la tabla usuarios

En la ilustración 89 podemos ver el Mapper extendido de Java que creamos para agregar nuevos métodos en la aplicación. Cabe destacar que los MapperExt que se crean tanto en Java como en XML se crean manualmente basados en los generados por MyBatis Generator.

```
@Mapper
public interface ScguUsuariosMapperExt extends CommonMapperExt<ScguUsuarios, ScguUsuariosSearchCriteria> {

    public ScguUserLogin loadByUsername(String username);

    public int actualizarCargo(Long id, String nombre, String codigo);

    public int actualizarDependencia(Long id, String nombre, String codigo);

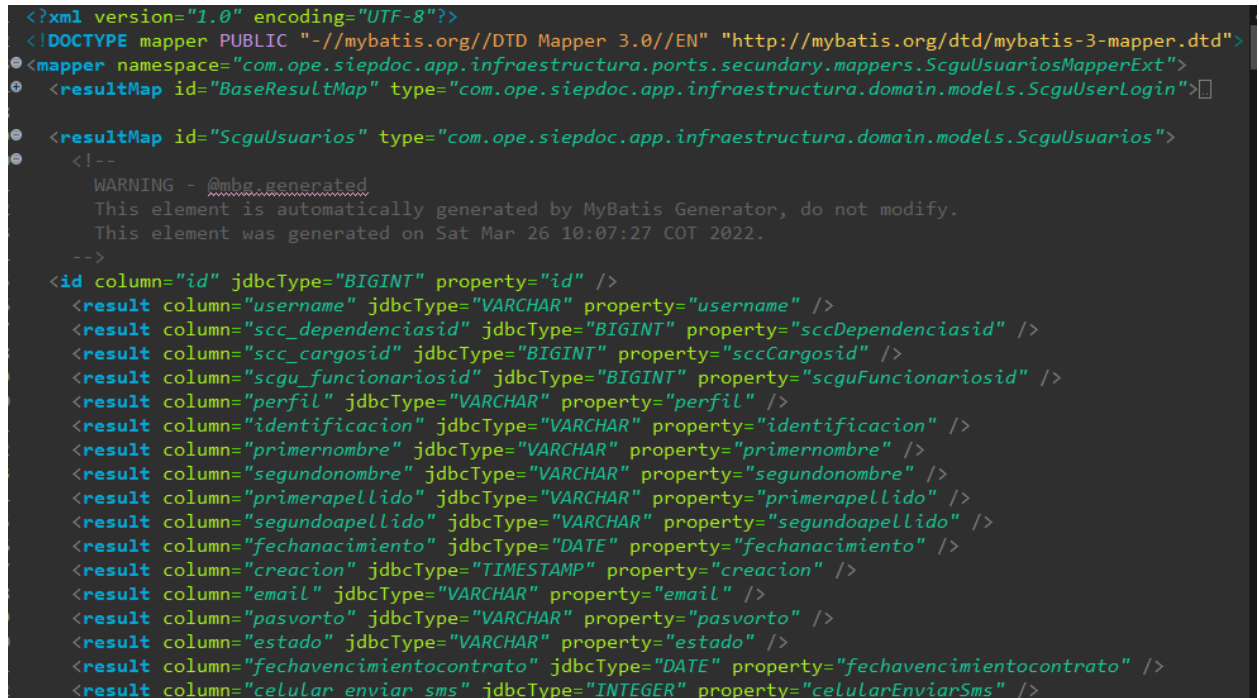
    public ScguUsuarios findByUsername(String username);

    public List<ScguUsuarios> filtroUsuariosPorCodigoDependencia(Long idDependencia);
}
```

Ilustración 89 Mapper extendido de la tabla usuarios

En la ilustración 90 y 91 podemos ver un MapperExt XML que creamos para agregar nuevos métodos en la aplicación, en este archivo XML es donde interactuamos directamente con la base de datos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org/DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.ope.siepdoc.app.infraestructura.ports.secondary.mappers.ScguUsuariosMapperExt">
  <resultMap id="BaseResultMap" type="com.ope.siepdoc.app.infraestructura.domain.models.ScguUserLogin">
  
```



```

  <!--
    WARNING - @mbg.generated
    This element is automatically generated by MyBatis Generator, do not modify.
    This element was generated on Sat Mar 26 10:07:27 COT 2022.
  -->
  <id column="id" jdbcType="BIGINT" property="id" />
  <result column="username" jdbcType="VARCHAR" property="username" />
  <result column="scc_dependenciasid" jdbcType="BIGINT" property="sccDependenciasid" />
  <result column="scc_cargosid" jdbcType="BIGINT" property="sccCargosid" />
  <result column="scgu_funcionariosid" jdbcType="BIGINT" property="scguFuncionariosid" />
  <result column="perfil" jdbcType="VARCHAR" property="perfil" />
  <result column="identificacion" jdbcType="VARCHAR" property="identificacion" />
  <result column="primernombre" jdbcType="VARCHAR" property="primernombre" />
  <result column="segundonombre" jdbcType="VARCHAR" property="segundonombre" />
  <result column="primerapellido" jdbcType="VARCHAR" property="primerapellido" />
  <result column="segundoapellido" jdbcType="VARCHAR" property="segundoapellido" />
  <result column="fechanacimiento" jdbcType="DATE" property="fechanacimiento" />
  <result column="creacion" jdbcType="TIMESTAMP" property="creacion" />
  <result column="email" jdbcType="VARCHAR" property="email" />
  <result column="pasvorto" jdbcType="VARCHAR" property="pasvorto" />
  <result column="estado" jdbcType="VARCHAR" property="estado" />
  <result column="fechavencimientocontrato" jdbcType="DATE" property="fechavencimientocontrato" />
  <result column="celular_enviar_sms" jdbcType="INTEGER" property="celularEnviarSms" />

```

Ilustración 90 MapperExt xml de la tabla usuarios parte 1

```
<select id="findByUsername" resultMap="ScguUsuarios">
  select * from scgu.scgu_usuarios where username= #{username,jdbcType=VARCHAR}
</select>
```

Ilustración 91 MapperExt xml de la tabla usuarios parte 2

En la ilustración 92 podemos ver los archivos necesarios para el uso de Mybatis. En el archivo “generatorConfig.xml” se encuentra la configuración para generar los Mapper Java y XML básicos mencionados anteriormente, para generarlos basta con dar clic derecho sobre el archivo y ejecutar con MyBatis Generator. En el archivo “mybatis-config.xml” se encuentra la configuración para que mybatis pueda localizar los Mapper xml que vamos a utilizar. En el

archivo “mybatis-velocity.properties” se encuentra algunas propiedades necesarias para realizar consultas dinámicas.

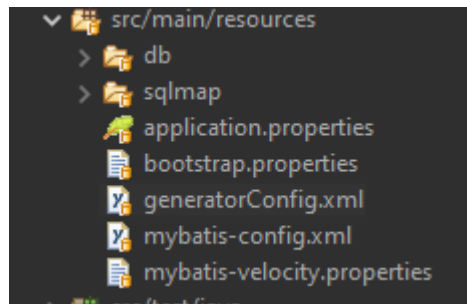


Ilustración 92 Archivos MyBatis

3.6.7. Siepdoc_micro_scc

En este microservicio se encuentra la lógica del negocio relacionada con las tablas del esquema de base de datos “scc”. En la ilustración 93 podemos observar la estructura de paquetes del microservicio.

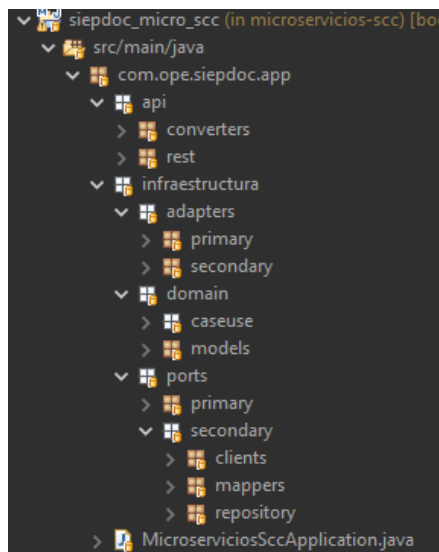


Ilustración 93 Estructura de paquetes Siepdoc_micro_scc

3.6.8. *Siepdoc_micro_scgr*

En este microservicio se encuentra la lógica del negocio relacionada con las tablas del esquema de base de datos “scgd”. En la ilustración 94 podemos observar la estructura de paquetes del microservicio.

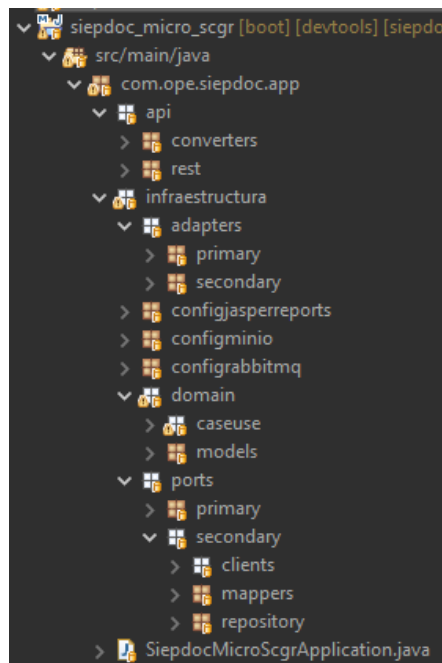


Ilustración 94 Estructura de paquetes Siepdoc_micro_scgr

En la ilustración 95 podemos observar el caso de uso para crear una radicación, funcionalidad de la cual se encarga este microservicio, aquí se puede observar las validaciones que se realizan en este caso de uso, como buscar la carpeta a la cual pertenece el tipo de radicado para asignarla, buscar el estado del radicado, validar si el ciudadano existe para crearlo o actualizarlo, calcular la fecha de respuesta del radicado y finalmente registrar el histórico.

```

@Override
public Long create(ScgrRadicados entity) throws Exception {

    entity = findCarpeta(entity);

    entity = findEstadoRadicado(entity);

    if (!(entity.getScgdCiudadanoid() != null)) {
        validaExisteCiudadano(entity);
        entity = saveCiudadano(entity);
    } else {
        updateCiudadano(entity);
    }

    entity = calcularFechaRespuesta(entity);

    super.create(entity);

    historicoRadicado.sendHistoricoRadicado(datosHistorico(entity));

    return entity.getId();
}

```

Ilustración 95 Caso de uso Crear Radicación

En la ilustración 96 podemos observar el caso de uso para almacenar el documento principal del radicado, aquí se puede observar las validaciones que se realizan antes de almacenar el pdf en minio, como validar el tamaño, crear la ruta donde se va almacenar el documento y actualizar el radicado.

```

public CommonResponseFileVo<ScgrTransaccionRadicadoVo> save(UploadFileVo uploadFileVo) throws Exception {

    ScgrRadicados scgrRadicados = null;

    try {
        scgrRadicados = scgrRadicadosRepository.read(uploadFileVo.getIdRadicado());
    } catch (Exception e) {
        throw new Exception("Ocurrió un error al buscar el radicado.");
    }

    validarTamaño(uploadFileVo.getLengthFile(), uploadFileVo.getFileName(), scgrRadicados);

    String rutaAbsoluta = scgrRadicados.getVigencia() + "/" +
        scgrRadicados.getCodigoDependenciaActual() + "/" +
        scgrRadicados.getNumeroRadicado() + "/" + uploadFileVo.getFileName();

    try {
        scgrRadicados.setRutaDocumentoPrincipal(rutaAbsoluta);
        scgrRadicadosRepository.update(scgrRadicados);
    } catch (Exception e) {
        throw new Exception("Ocurrió un error al actualizar el radicado.");
    }

    CommonResponseFileVo<ScgrTransaccionRadicadoVo> commonResponseFileVo = new CommonResponseFileVo<ScgrTransaccionRadicadoVo>
        llenarTransaccionVo(uploadFileVo.getLengthFile(), uploadFileVo.getFileName(), uploadFileVo.getMimeType(),
            true,
            rutaAbsoluta);

    return commonResponseFileVo;
}

```

Ilustración 96 Caso de uso para para almacenar documento principal del radicado

En la ilustración 97 podemos observar el repositorio para almacenar archivos en minio, se almacenan en el “Bucket” especificado en el archivo de propiedades del microservicio.

```
@Override
public void save(File file, String rutaAbsoluta) {
    try {
        minioClient.uploadObject(UploadObjectArgs.builder()
            .bucket(defaultBucketName)
            .object(rutaAbsoluta) //ruta ej "2022/000/12345/"+nombrearchivo.ext
            .filename(file.getAbsolutePath())
            .build());

        System.gc();
    } catch (Exception e) {
        throw new RuntimeException("No se puede guardar el archivo. Error " + e);
    }
}
```

Ilustración 97 Repositorio para almacenar archivos en minio

En la ilustración 98 podemos observar el repositorio para obtener la URL de la ubicación de archivos en minio, solo debemos enviar la ruta del archivo en el Bucket que se guarda en el radicado.

```
@Override
public String getFileURL(String rutaAbsoluta) {
    String url="";
    try {
        url =
            minioClient.getPresignedObjectUrl(
                GetPresignedObjectUrlArgs.builder()
                    .method(Method.GET)
                    .bucket(defaultBucketName)
                    .object(rutaAbsoluta)
                    .expiry(1, TimeUnit.DAYS)
                    .build());

        System.out.println(url);
    } catch (Exception e) {
        System.out.println("error"+e);
    }

    return url;
}
```

Ilustración 98 Repositorio para obtener URL de ubicación de un archivo de minio

En la ilustración 99 podemos observar el repositorio para obtener los bytes de los archivos en minio, solo debemos enviar la ruta del archivo en el Bucket que se guarda en el radicado. Este repositorio se utiliza para visualizar los PDF de los documentos principales en el visor del FrontEnd.

```

@Override
public byte[] getFilesBytes(String rutaAbsoluta) {
    InputStream stream=null;
    byte bais[] = null;
    try {
        stream = minioClient.getObject(
            GetObjectArgs.builder()
                .bucket(defaultBucketName)
                .object(rutaAbsoluta)
                .build());
        bais=stream.readAllBytes();
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return bais;
}

```

Ilustración 99 Repositorio para obtener los bytes de los archivos de minio

En la ilustración 100 podemos observar el repositorio para eliminar archivos en minio, solo debemos enviar la ruta del archivo en el Bucket que se guarda en el radicado.

```

@Override
public String deleteFile(String rutaAbsoluta){
    try {
        minioClient.removeObject(
            RemoveObjectArgs.builder()
                .bucket(defaultBucketName)
                .object(rutaAbsoluta)
                .build());
        return "Se eliminó exitosamente.";
    } catch (Exception e){
        e.printStackTrace();
        return "Ocurrió un error al eliminar.";
    }
}

```

Ilustración 100 Repositorio para eliminar un archivo de minio

3.6.9. Siepdoc_micro_envio_correo

En este microservicio se encuentra la lógica para el envío de correos electrónicos. En la ilustración 101 podemos observar la estructura de paquetes del microservicio.

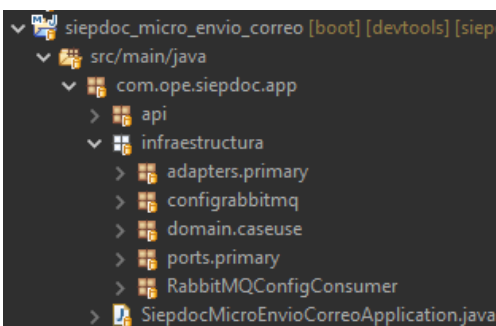


Ilustración 101 Estructura de paquetes Siepdoc_micro_envio_correo

En la ilustración 102 podemos observar el caso de uso para el envío de emails, se debe generar un código para terceros en la cuenta del correo electrónico del cual se van a enviar los emails para que el microservicio pueda hacer los envíos.

```
public void enviarCorreo(EnvioCorreoParamsVo envioCorreoParamsVo) throws EmailException {
    enviar(envioCorreoParamsVo);
}

public void enviar(EnvioCorreoParamsVo envioCorreoParamsVo) throws EmailException {
    ScguUsuariosVo scguUsuariosVo=envioCorreoParamsVo.getScguUsuariosVo();
    Email email = new SimpleEmail();
    email.setSmtpport(587);
    email.setHostName("smtp.gmail.com");
    email.setAuthenticator(new DefaultAuthenticator("deimercegarra.ope", "██████████"));
    email.setSSLonConnect(true);
    email.setFrom("deimercegarra.ope@gmail.com");
    email.setSubject(envioCorreoParamsVo.getAsunto());
    email.setMsg(envioCorreoParamsVo.getCuerpoCorreo());
    email.addTo(envioCorreoParamsVo.getCorreo());
    try {
        scguUsuariosVo.setEstado("Email Enviado");
        email.send();
    } catch (Exception e) {
        scguUsuariosVo.setEstado("Verificando Email");
    }
    respuestaEnvioCorreo.sendRespuestaEnvioCorreo(scguUsuariosVo);
}
```

Ilustración 102 Caso de uso para envío de emails

3.6.10. Siepdoc_micro_reports

En este microservicio se encuentra la lógica para la creación de reportes, este tiene acceso a todos los esquemas de base de datos mencionados anteriormente. En la ilustración 103 podemos observar la estructura de paquetes del microservicio.

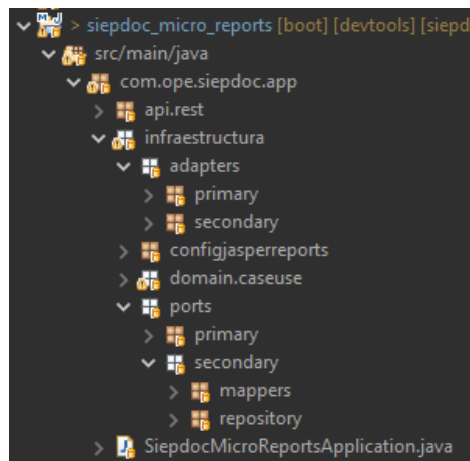


Ilustración 103 Estructura de paquetes Siepdoc_micro_reports

En la ilustración 104 podemos observar el caso de uso para la creación de los reportes, aquí se puede observar que existe una clase llamada “SimpleReportFiller”, esta se encarga de almacenar y procesar la información del reporte. También hay una clase llamada “SimpleReportExporter”, esta se encarga de exportar los reportes a los diferentes formatos de archivo que queramos, en este caso se utilizan dos formatos: PDF y XLSX. Los “parameters” son los datos que se envían para el encabezado del reporte, y el “datasource” es la lista con la información recolectada de la consulta a la base de datos.

```

public byte[] generarPlanillaBytes(JasperParamsVo paramsVo) throws Exception {
    AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();
    ctx.register(JasperReportsSimpleConfig.class);
    ctx.refresh();

    SimpleReportFiller simpleReportFiller = ctx.getBean(SimpleReportFiller.class);

    simpleReportFiller.setReportFileName(reportFileName(paramsVo.getPlanilla()));
    simpleReportFiller.compileReport();

    simpleReportFiller.setParameters(parameters(paramsVo));
    simpleReportFiller.fillDataSource(dataSource(paramsVo));
    simpleReportFiller.fillReport();

    SimpleReportExporter simpleExporter = ctx.getBean(SimpleReportExporter.class);
    simpleExporter.setJasperPrint(simpleReportFiller.getJasperPrint());

    return generarBytes(simpleReportFiller, paramsVo.getTipoArchivo(), simpleExporter);
}

```

Ilustración 104 Caso de uso para generar reportes

En la ilustración 105 podemos observar la plantilla, en este caso, para el reporte de Correspondencia de entrada en el editor utilizado, Jaspersoft Studio.

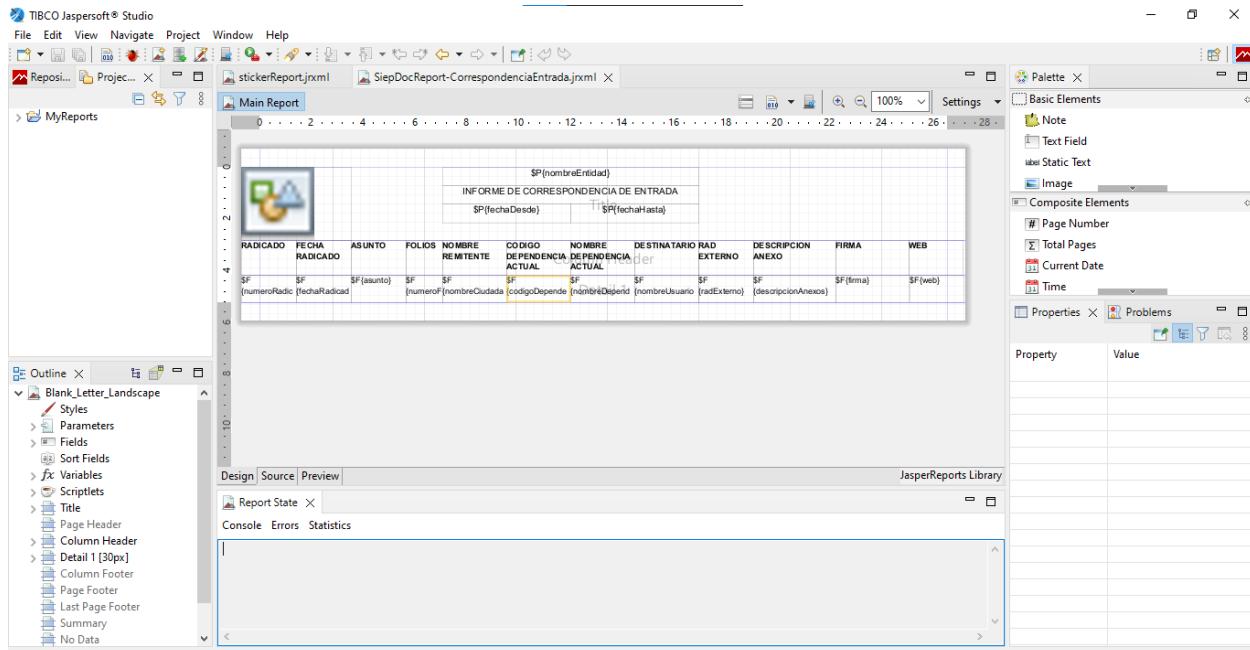


Ilustración 105 Plantilla reporte Correspondencia de entrada en Jaspersoft Studio

3.6.11. Siepdoc_micro_oauth

En este microservicio se encuentra parte de la seguridad de la aplicación, este se encarga de la autenticación y autorización de los usuarios. En este microservicio se utiliza Spring Security y la librería Java-jwt. En la ilustración 106 podemos observar la estructura de paquetes del microservicio.

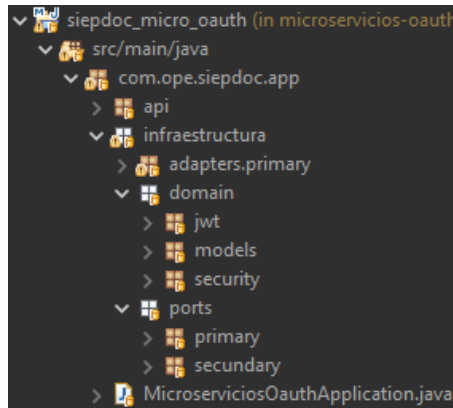


Ilustración 106 Estructura de paquetes Siepdoc_micro_oauth

En la ilustración 107 se puede observar la clase de configuración de Spring Security, aquí se filtran las peticiones a las EndPoint que podemos ver, el EndPoint “/login” es el utilizado para el inicio de sesión desde el FrontEnd.

```

@Configuration
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired
    private UserDetailsService usuarioService;

    @Autowired
    private UsuarioFeingRepository usuarioFeingRepository;

    @Value("${jwtsecret}")
    private String secret;

    @Value("${expiration}")
    private int expiration;

    @Bean
    public static BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.authorizeRequests().antMatchers("/login").permitAll();
        http.authorizeRequests().antMatchers("/generarToken").permitAll();
        http.authorizeRequests().antMatchers("/generarToken/RecoveryPasvorto").permitAll();
        http.authorizeRequests().anyRequest().authenticated();
        http.addFilter(new CustomAuthenticationFilter(authenticationManagerBean(),
            usuarioFeingRepository,secret,expiration));
    }
}

```

Ilustración 107 Clase de configuración de Spring Security

En la ilustración 108 podemos observar la clase “CustomAuthenticationFilter”, la cual se encarga de autenticar los usuarios y proporcionar un token para la sesión, y realizar otras validaciones sobre los usuarios.

```

public class CustomAuthenticationFilter extends UsernamePasswordAuthenticationFilter {

    private String secret;
    private int expiration;

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UsuarioFeingRepository usuarioFeingRepository;

    private RememberMeServices rememberMeServices = new NullRememberMeServices();

    private AuthenticationFailureHandler failureHandler = new SimpleUrlAuthenticationFailureHandler();

    public CustomAuthenticationFilter(AuthenticationManager authenticationManager,
        UsuarioFeingRepository usuarioFeingRepository, String secret, int expiration) {
        this.authenticationManager = authenticationManager;
        this.usuarioFeingRepository = usuarioFeingRepository;
        this.secret = secret;
        this.expiration = expiration;
    }
}

```

Ilustración 108 Clase CustomAuthenticationFilter parte 1

En la ilustración 109 podemos observar que allí se hace el intento de inicio de sesión con el método “authenticate” de la interface “AuthenticationManager”.

```
@Override
public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
    throws AuthenticationException {

    String username = request.getParameter("username");
    String password = request.getParameter("password");
    UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(username,
        password);
    return authenticationManager.authenticate(authenticationToken);
}
```

Ilustración 109 Clase CustomAuthenticationFilter parte 2

En la ilustración 110 podemos observar que allí se captura el intento de inicio de sesión exitoso, aquí se hacen validaciones sobre el usuario y se crea el token para la sesión con los permisos del usuario.

```
@Override
protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain,
    Authentication authentication) throws IOException, ServletException {
    UserLoginNueva user = (UserLoginNueva) authentication.getPrincipal();

    validarIntentosInicioSesionExitoso(user.getUsername());

    Algorithm algorithm = Algorithm.HMAC256(secret.getBytes());
    String access_token = JWT.create().withSubject(user.getUsername())
        .withExpiresAt(new Date(System.currentTimeMillis() + expiration))
        .withIssuer(request.getRequestURL().toString())
        .withClaim("permisos",
            user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()))
        .withClaim("idUsuario", user.getIdUsuario())
        .withClaim("rol", user.getRol())
        .sign(algorithm);

    Map<String, String> tokens = new HashMap<>();
    tokens.put("access_token", access_token);
    response.setContentType("application/json");
    new ObjectMapper().writeValue(response.getOutputStream(), tokens);
}
```

Ilustración 110 Clase CustomAuthenticationFilter parte 3

En la ilustración 111 podemos observar que allí se captura el intento de inicio de sesión fallido, aquí se hacen validaciones sobre el usuario, se contabiliza los intentos fallidos para cuando sean 3 el usuario será suspendido.

```

@Override
protected void unsuccessfulAuthentication(HttpServletRequest request, HttpServletResponse response,
    AuthenticationException failed) throws IOException, ServletException {

    String username = request.getParameter("username");

    validarIntentosInicioSesionErroneos(username);

    SecurityContextHolder.clearContext();
    this.logger.trace("Failed to process authentication request", failed);
    this.logger.trace("Cleared SecurityContextHolder");
    this.logger.trace("Handling authentication failure");
    this.rememberMeServices.loginFail(request, response);
    this.failureHandler.onAuthenticationFailure(request, response, failed);
}

```

Ilustración 111 Clase CustomAuthenticationFilter parte 4

En la ilustración 112 podemos observar que las peticiones al EndPoint “/login” se filtran hacia la clase “CustomAuthenticationFilter” de la ilustración 108, como se ha configurado en la clase de configuracion de Spring Security de la ilustración 107.

```

public class CustomAuthorizationFilter extends OncePerRequestFilter {

    @Value("${jwt.secret}")
    private String secret;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
        FilterChain filterChain) throws ServletException, IOException {

        if (request.getServletPath().equals("/login")) {
            filterChain.doFilter(request, response);
        } else {

```

Ilustración 112 Clase CustomAutorizationFilter

En la ilustración 113 podemos observar que allí se hace un Override sobre el método “UserDetails” de la interface “UserDetailsService”, en este método se consulta al usuario en la base de datos, si el usuario no está suspendido o su fecha de contrato no está vencida entonces la consulta a la base de datos por medio de la API retornará los datos que necesitamos para crear el usuario del UserDetails, de esta manera Spring security valida las credenciales enviadas en la petición con las credenciales que son correctas del UserDetails en la ilustración 109 con el método “authenticate”.

```

@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    CommonResponseVo<ScguUserLoginVo> resp = usuarioFeingClient.LoadByUsername(username);

    try {
        if (resp.getVo() == null) {
            throw new Exception(resp.getMensaje());
        }
        List<GrantedAuthority> authorities= resp.getVo().getPermisos()
            .stream()
            .map(SimpleGrantedAuthority::new)
            .collect(Collectors.toList());

        return new UserLoginNueva(resp.getVo().getUsername(), resp.getVo().getIdUsuario(),
            resp.getVo().getRol(), resp.getVo().getPassword(),true, true,
            true,true,authorities);
    } catch (Exception e) {

        tracer.currentSpan().tag("error.mensaje", resp.getMensaje());
        log.error(resp.getMensaje());
        System.err.println("error "+e.getMessage());
        throw new UsernameNotFoundException(resp.getMensaje());
    }
}

```

Ilustración 113 Creación del UserDetails

3.6.12. Siepdoc_micro_gateway

En este microservicio se encuentra parte de la seguridad de la aplicación, este se utiliza para enrutar todas las peticiones a las API del BackEnd. En este microservicio se utiliza Spring Security y la librería Java-jwt. En la ilustración 114 podemos observar la estructura de paquetes del microservicio.

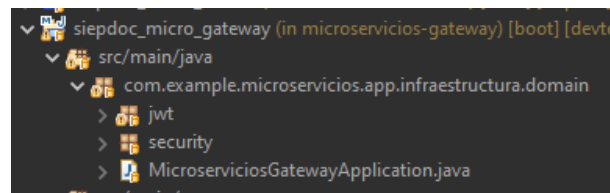


Ilustración 114 Estructura de paquetes Siepdoc_micro_gateway

En la ilustración 115 podemos observar la clase de configuración de Spring Security, en esta se configuran cuáles rutas pueden ser públicas y cuáles privadas. También se filtran las peticiones con la clase personalizada “JwtAuthenticationFilter”.

```

@EnableWebFluxSecurity
public class SpringSecurityConfig {

    @Autowired
    private JwtAuthenticationFilter authenticationFilter;

    @Bean
    public SecurityWebFilterChain configure(ServerHttpSecurity http) {
        return http.authorizeExchange()
            .pathMatchers("/api/security/login").permitAll()
            .pathMatchers("/api/security/generarToken/RecoveryPasvorto").permitAll()
            .pathMatchers("/api/usuarios/api/usuarios/actualizarPasvorto").hasAnyAuthority
            .anyExchange().authenticated()
            .and().addFilterAt(authenticationFilter, SecurityWebFiltersOrder.AUTHENTICATIO
            .csrf().disable()
            .build();
    }
}

```

Ilustración 115 Clase de configuracion de Spring Security

En la ilustración 116 podemos observar la clase de configuracion “JwtAuthenticationFilter”. En esta clase se hace un Override al método “filter” de la clase “WebFilter” y se valida la autenticación del token que se envía en los Headers de la petición con el método “authenticate” de la clase “ReactiveAuthenticationMagager”.

```

Component
public class JwtAuthenticationFilter implements WebFilter {

    @Autowired
    private ReactiveAuthenticationManager authenticationManager;

    public Mono<Void> filter(ServerWebExchange exchange, WebFilterChain chain) {
        return Mono.justOrEmpty(exchange.getRequest().getHeaders().getFirst(HttpHeaders.AUTHORIZATION))
            .filter(authHeader -> authHeader.startsWith("Bearer "))
            .switchIfEmpty(chain.filter(exchange).then(Mono.empty()))
            .map(token -> token.replace("Bearer ", ""))
            .flatMap(token -> authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(null, token)))
            .flatMap(authentication -> chain.filter(exchange)
                |.contextWrite(ReactiveSecurityContextHolder.withAuthentication(authentication)));
    }
}

```

Ilustración 116 Clase JwtAuthenticationFilter

En la ilustración 117 podemos observar la clase de configuracion “AuthenticationManagerJwt”. En esta clase se hace un Override al método “authenticate” de la clase “ReactiveAuthenticationMagager” donde se intercepta la autenticación de la clase “JwtAuthenticationFilter” de la ilustración 116 y se verifica que el token sea válido, de lo contrario, genera un error de autenticación y rechaza las peticiones.

```

@Component
public class AuthenticationManagerJwt implements ReactiveAuthenticationManager {

    @Value("${jwt.llaveJwt}")
    private String llaveJwt;

    public Mono<Authentication> authenticate(Authentication authentication) {
        return Mono.just(authentication.getCredentials().toString())
            .map(token-> {
                Algorithm algorithm = Algorithm.HMAC256(llaveJwt.getBytes());
                JWTVerifier verifier = JWT.require(algorithm).build();
                return verifier.verify(token);
            }).map(claims->{
                String username = claims.getSubject();
                List<String> roles = claims.getClaim("permisos").asList(String.class);
                Collection<SimpleGrantedAuthority> authorities=roles.stream()
                    .map(rol-> new SimpleGrantedAuthority(rol)).collect(Collectors.toList());
                return new UsernamePasswordAuthenticationToken(username, null,authorities);
            });
    }
}

```

Ilustración 117 Clase AuthenticationManagerJwt

En la ilustración 118 podemos observar la configuracion de rutas de los microservicios, esta se hace en un archivo YML.

```

spring:
  cloud:
    gateway:
      globalcors:
        corsConfigurations:
          '[/**]':
            allowedOrigins: 'http://vaadin-micro:8084'
            allowedHeaders:
              - Content-Type
            allowedMethods:
              - GET
              - POST
              - PUT
              - DELETE
      routes:
        - id: microservicio-scgw
          uri: lb://microservicio-scgw
          predicates:
            - Path=/api/usuarios/**
          filters:
            - StripPrefix=2
        - id: microservicio-scc
          uri: lb://microservicio-scc
          predicates:
            - Path=/api/configuracion/**
          filters:
            - StripPrefix=2

```

Ilustración 118 Configuración de rutas de los microservicios

3.7. Fase de pruebas

En esta sección se describen las pruebas de aceptación para las historias de usuarios del módulo principal del presente trabajo, Radicación. Los casos de prueba que se describen son

prueba de radicación con campos obligatorios completos, radicación con campos vacíos, y radicación con campos obligatorios incompletos.

Tabla 25 Registrar datos obligatorios del formulario de radicación de entrada.

NUMERO DE PRUEBA: 1	NUMERO HISTORIA DE USUARIO: 21
DESCRIPCION DE LA PRUEBA	
Registrar datos obligatorios del formulario de radicación de entrada.	
CONDICIONES DE EJECUCION	
El funcionario debe tener permiso para radicar entradas.	
El tipo de radicado de entrada debe tener asignado el tipo de documento que se radica.	
Debe estar registrado el medio de recepción y los datos de ubicación como Continente, País, Departamento y Municipio tanto del ciudadano como del lugar donde se radica.	
ENTRADAS Y PASOS DE EJECUCION	
En una alcaldía se radica una entrada que no requiere respuesta, con los siguientes datos en el formulario:	
Sección CIUDADANO: Tipo Identificación: Cédula, Identificación: 1004912874, Primer Nombre: José, Primer Apellido: Pérez.	
Sección Otros datos del Ciudadano: Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario.	
Sección INFORMACION A RADICAR: Medio Recepción: Personal, Folios: 0, Tipo Documento: Petición, Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario, Asunto: petición para alumbrado de parques públicos por motivo de mes de diciembre.	
Sección ASIGNACION: Dependencia: Cultura, Funcionario: Mario Andrés Monsalve Reyes.	
Finalmente se da clic en el botón 'Radicar' en la parte inferior de la pantalla para generar el radicado.	
RESULTADOS ESPERADOS	
Se debe generar el radicado y mostrarse en pantalla el número del radicado, junto a los botones: 'Nuevo', 'Modificar', 'Sticker', 'Documento Principal', 'Adjuntar Documentos', 'Informar'.	
Ejemplo de numero de radicado: 20229920000232.	
EVALUACION DE LA PRUEBA	
Prueba satisfactoria.	

Tabla 26 Registrar datos obligatorios incompletos del formulario de radicación de entrada.

NUMERO DE PRUEBA: 2	NUMERO HISTORIA DE USUARIO: 21
DESCRIPCION DE LA PRUEBA	
Registrar datos obligatorios incompletos del formulario de radicación de entrada.	
CONDICIONES DE EJECUCION	
El funcionario debe tener permiso para radicar entradas.	
ENTRADAS Y PASOS DE EJECUCION	
En una alcaldía se radica una entrada que no requiere respuesta, con los siguientes datos en el formulario:	
Sección CIUDADANO: Tipo Identificación: Cédula, Identificación: 1004912874, Primer Nombre: José, Primer Apellido: ____.	
Sección Otros datos del Ciudadano: Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: ____.	

Sección INFORMACION A RADICAR: Medio Recepción: Personal, Tipo Documento: ____, Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario, Asunto: ____.
 Sección ASIGNACION: Dependencia: ____, Funcionario: ____.
 Finalmente se da clic en el botón ‘Radicar’ en la parte inferior de la pantalla para generar el radicado.

RESULTADOS ESPERADOS

Se deben mostrar los campos obligatorios vacíos en color rojo y un mensaje debajo del campo diciendo que el campo es obligatorio.

EVALUACION DE LA PRUEBA

Prueba satisfactoria.

Tabla 27 Radicar sin registrar datos del formulario de radicación de entrada.

NUMERO DE PRUEBA: 3

NUMERO HISTORIA DE USUARIO: 21

DESCRIPCION DE LA PRUEBA

Radical sin registrar datos del formulario de radicación de entrada.

CONDICIONES DE EJECUCION

El funcionario debe tener permiso para radicar entradas.

ENTRADAS Y PASOS DE EJECUCION

En una alcaldía se radica una entrada que no requiere respuesta, con los siguientes datos en el formulario:

Sección CIUDADANO: Tipo Identificación: ____, Identificación: ____, Primer Nombre: ____, Primer Apellido: ____.

Sección Otros datos del Ciudadano: Continente: ____, País: ____, Departamento: ____, Municipio: ____.

Sección INFORMACION A RADICAR: Medio Recepción: ____, Tipo Documento: ____, Continente: ____, País: ____, Departamento: ____, Municipio: ____, Asunto: ____.

Sección ASIGNACION: Dependencia: ____, Funcionario: ____.

Finalmente se da clic en el botón ‘Radical’ en la parte inferior de la pantalla para generar el radicado.

RESULTADOS ESPERADOS

Se deben mostrar los campos obligatorios vacíos en color rojo y un mensaje debajo del campo diciendo que el campo es obligatorio.

EVALUACION DE LA PRUEBA

Prueba satisfactoria.

Tabla 28 Registrar datos obligatorios del formulario de radicación de salida.

NUMERO DE PRUEBA: 4

NUMERO HISTORIA DE USUARIO: 22

DESCRIPCION DE LA PRUEBA

Registrar datos obligatorios del formulario de radicación de salida.

CONDICIONES DE EJECUCION

El funcionario debe tener permiso para radicar salidas.

El tipo de radicado de salida debe tener asignado el tipo de documento que se radica.

Debe estar registrado el medio de recepción y los datos de ubicación como Continente, País, Departamento y Municipio, tanto del ciudadano como del lugar donde se radica.

ENTRADAS Y PASOS DE EJECUCION

Se radica una salida que tiene como antecedente el radicado de entrada de la prueba número 1, con los siguientes datos en el formulario:

Sección CIUDADANO: Tipo Identificación: Cédula, Identificación: 1004912874, Primer Nombre: José, Primer Apellido: Pérez.

Sección Otros datos del Ciudadano: Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario.
Sección ANTECEDENTE: Antecedente: 20229920000232. (ponemos como antecedente el radicado de entrada de la prueba numero 1)
Sección INFORMACION A RADICAR: Medio Recepción: interno, Folios: 0, Tipo Documento: Tutela, Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario, Asunto: Desde la alcaldía se está trabajando para atender la solicitud de alumbrados públicos para el mes de diciembre, como es de costumbre todos los años.
Sección ASIGNACION: Dependencia: Cultura, Funcionario: Deimer Joel Cegarra Cordero.
Finalmente se da clic en el botón 'Radificar' en la parte inferior de la pantalla para generar el radicado.
RESULTADOS ESPERADOS
Se debe generar el radicado y mostrarse en pantalla el número del radicado, junto a los botones: 'Nuevo', 'Modificar', 'Sticker', 'Documento Principal', 'Adjuntar Documentos', 'Informar', 'Otros Antecedentes', 'Otros Destinatarios'.
Ejemplo de numero de radicado: 20229920000241.
EVALUACION DE LA PRUEBA
Prueba satisfactoria.

Tabla 29 Registrar datos obligatorios incompletos del formulario de radicación de salida.

NUMERO DE PRUEBA: 5	NUMERO HISTORIA DE USUARIO: 22
DESCRIPCION DE LA PRUEBA	
Registrar datos obligatorios incompletos del formulario de radicación de salida.	
CONDICIONES DE EJECUCION	
El funcionario debe tener permiso para radicar salidas.	
ENTRADAS Y PASOS DE EJECUCION	
Se radica una salida que tiene como antecedente el radicado de entrada de la prueba número 1, con los siguientes datos en el formulario:	
Sección CIUDADANO: Tipo Identificación: Cédula, Identificación: 1004912874, Primer Nombre: José, Primer Apellido: ____.	
Sección Otros datos del Ciudadano: Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: ____.	
Sección ANTECEDENTE: Antecedente: 20229920000232. (ponemos como antecedente el radicado de entrada de la prueba número 1)	
Sección INFORMACION A RADICAR: Medio Recepción: Personal, Tipo Documento: ____, Continente: América, País: Colombia, Departamento: Norte de Santander, Municipio: Villa del Rosario, Asunto: ____.	
Sección ASIGNACION: Dependencia: ____, Funcionario: ____.	
Finalmente se da clic en el botón 'Radificar' en la parte inferior de la pantalla para generar el radicado.	
RESULTADOS ESPERADOS	
Se deben mostrar los campos obligatorios vacíos en color rojo y un mensaje debajo del campo diciendo que el campo es obligatorio.	
EVALUACION DE LA PRUEBA	
Prueba satisfactoria.	

Tabla 30 Radicar sin registrar datos del formulario de radicación de salida.

NUMERO DE PRUEBA: 6	NUMERO HISTORIA DE USUARIO: 22
DESCRIPCION DE LA PRUEBA	

Radicar sin registrar datos del formulario de radicación de salida.

CONDICIONES DE EJECUCION

El funcionario debe tener permiso para radicar salidas.

ENTRADAS Y PASOS DE EJECUCION

Se radica una salida que tiene como antecedente el radicado de entrada de la prueba número 1, con los siguientes datos en el formulario:

Sección CIUDADANO: Tipo Identificación: __, Identificación: __, Primer Nombre: __, Primer Apellido: __.

Sección Otros datos del Ciudadano: Continente: __, País: __, Departamento: __, Municipio: __.

Sección ANTECEDENTE: Antecedente: 20229920000232. (ponemos como antecedente el radicado de entrada de la prueba número 1)

Sección INFORMACION A RADICAR: Medio Recepción: __, Tipo Documento: __, Continente: __, País: __, Departamento: __, Municipio: __, Asunto: __.

Sección ASIGNACION: Dependencia: __, Funcionario: __.

Finalmente se da clic en el botón 'Radicar' en la parte inferior de la pantalla para generar el radicado.

RESULTADOS ESPERADOS

Se deben mostrar los campos obligatorios vacíos en color rojo y un mensaje debajo del campo diciendo que el campo es obligatorio.

EVALUACION DE LA PRUEBA

Prueba satisfactoria.

Conclusiones

El análisis de los requerimientos, diseño y modelado del software nos permitió tener una visión general de las funcionalidades que fueron desarrolladas en el presente proyecto, lo cual permitió llevar a cabo exitosamente el desarrollo de cada uno de los requerimientos iniciales.

Las recomendaciones aplicadas del estándar ISO 29110 fueron importantes para el desarrollo del producto final, pues se logró un mejor resultado de acuerdo al nivel de calidad exigido por la empresa OPE Colombia en cuanto a la optimización de recursos y proceso de desarrollo; esto gracias a su experiencia con la certificación en el estándar.

El marco de desarrollo Vaadin Flow nos permitió realizar satisfactoriamente la construcción de los módulos Radicación y, Administración de Usuarios y Roles de la aplicación de gestión documental SIEPDOG, gracias al aporte de componentes web pre diseñados que facilitaron, en muchas ocasiones, la elaboración de las interfaces de usuario. La plataforma Vaadin también proporciona cada vez mejor documentación lo cual permitió encontrar soluciones rápidas y eficaces.

Este proyecto constituye una base de conocimiento para futuras investigaciones en lo relacionado con el uso de arquitecturas de vanguardia que aportan soluciones a los problemas que surgen en las aplicaciones web como lo son la arquitectura de microservicios y la arquitectura REST, la aplicación de estándar ISO 29110 que optimiza el proceso de desarrollo gracias a la documentación sugerida, y la metodología Scrum que agiliza la productividad del equipo de desarrollo.

Recomendaciones

Se recomienda tener conocimientos previos para la aplicación de las tecnologías utilizadas en el presente proyecto, debido a que se pueden disminuir los costos de desarrollo por concepto de curva de aprendizaje.

Para investigaciones donde se desee utilizar el marco de desarrollo Vaadin Flow se recomienda tener claro lo que se quiere realizar debido a las limitaciones de algunos componentes que este proporciona, como lo es la complejidad para la modificación de estilos y los derechos de propietario. De esta manera, se deben efectuar los análisis correspondientes antes de comenzar a desarrollar.

Para proyectos planeados con microservicios, se recomienda hacer un análisis detallado del software que se quiera desarrollar debido a que los microservicios tienen un alto consumo de recursos y su curva de aprendizaje también genera costos altos si no se tienen conocimientos previos.

En cuanto al producto obtenido en este proyecto, se puede mejorar el aspecto visual de la aplicación para futuras versiones e integrar otras tecnologías como kubernetes que facilita la manipulación de contenedores Docker.

Referencias

- A. G. Bonelli Toro, M. J. Iofrida, & M. P. Gómez. (2019). *Interfaz para visualizar y procesar señales de emisión acústica*. <https://dialnet.unirioja.es/servlet/articulo?codigo=7940724>
- Abad Gómez-Pastrana, R. (2018). *VISALipse Plugin para el IDE Eclipse para la visualización de algoritmos bajo el paradigma de Orientación a Aspectos*. [Universidad de Castilla-La Mancha]. <https://ruidera.uclm.es/xmlui/handle/10578/19074>
- Berdugo Baez, L. E. (2016). *Análisis de los sistemas de gestión documental que existen actualmente en colombia, que cumplen con la legislación vigente para empresas públicas de acuerdo con el programa de gestión documental*. <https://repository.unimilitar.edu.co/handle/10654/14557>
- Cabrera, P., & Marlon, P. (2018). Software SistWebDigit para mejorar el proceso de la gestión documental en la empresa Servicios Integrados de Limpieza, 2018. *Universidad Privada Norbert Wiener*. <http://repositorio.uwiener.edu.pe/handle/123456789/2503>
- Cambarieri, M., Difabio, F., & García, N. (2020). *Implementación de una arquitectura de software guiada por el dominio*. <http://sedici.unlp.edu.ar/handle/10915/115198>
- Carvajal Suarez, F. A., Clemen, G., & Torres Chavez, I. (2017). SISTEMATIZACIÓN DEL PROCESO DE PRESTACIÓN DEL SERVICIO DE LABORATORIOS DE LA FACULTAD DE INGENIERÍAS Y ARQUITECTURA DE LA UNIVERSIDAD DE PAMPLONA. *REVISTA AMBIENTAL AGUA, AIRE Y SUELO*, 8(1). <https://doi.org/10.24054/19009178.V1.N1.2017.3275>
- Cesar. (2017). *PostgreSQL: qué es, cómo funciona y cuáles son sus ventajas*. https://platzi.com/blog/que-es-postgresql/?utm_source=google&utm_medium=cpc&utm_campaign=12915366154&utm_a

dgroup=&utm_content=&gclid=Cj0KCQiAkNiMBhCxARIsAIDDKNVlT9ldEayeuY9wIt
O5rXeVwgUBy2aZpE-JLin2r4KLKuasy7zQm3MaAmnsEALw_wcB&gclsrc=aw.ds

defensoria.gov.co. (2021). *Ley 23 de 1982, sobre derechos de autor*.
[https://www.defensoria.gov.co/public/Normograma 2013_html/Normas/Ley_23_1982.pdf](https://www.defensoria.gov.co/public/Normograma%202013_html/Normas/Ley_23_1982.pdf)

docs.docker.com. (2022a). *Docker overview | Docker Documentation*.
<https://docs.docker.com/get-started/overview/>

docs.docker.com. (2022b). *Use Docker Compose | Docker Documentation*.
https://docs.docker.com/get-started/08_using_compose/

Freire, L. (2014). Estudio de factibilidad para la implementación del Sistema de Gestión Documental “Quipux” en la Universidad Regional Autónoma de los Andes. *Revista UNIANDES Episteme*, 1(1), 95–106.
<https://dialnet.unirioja.es/servlet/articulo?codigo=6756307&info=resumen&idioma=SPA>

funcionpublica.gov.co. (2019). *Decreto 2106 de 2019 - EVA - Función Pública*.
<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=103352>

funcionpublica.gov.co. (2021). *Ley 1581 de 2012 - EVA - Función Pública*.
<https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981>

Ginestà Gibert, M., & Pérez Mora, O. (2012). *Bases de datos en PostgreSQL*.
<https://artpatcusco.com/sis/pdf/20140527172742en.pdf>

hub.docker.com. (2022). *mongo-express - Official Image | Docker Hub*.
https://hub.docker.com/_/mongo-express

java.com. (2021). *¿Qué es Java y para qué es necesario?*
https://www.java.com/es/download/help/whatis_java.html

jenkins.io. (2022). *Jenkins*. <https://www.jenkins.io/>

- López, D., & Maya, E. (2017). *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web*.
- Martelo, R. J., Maderay, J. E., & Betín, A. D. (2015). Software para Gestión Documental, un Componente Modular del Sistema de Gestión de Seguridad de la Información (SGSI). *SCIELO*, 26(2), 129–134. <https://doi.org/10.4067/S0718-07642015000200015>
- Maza Valles, A. de la. (2020). *API REST para la gestión de partidas multijugador de un juego serio*. <https://uvadoc.uva.es/handle/10324/44417>
- MDN contributors. (2021a, November 17). CSS. <https://developer.mozilla.org/es/docs/Web/CSS>
- MDN contributors. (2021b, November 17). *Empezar con HTML*. https://developer.mozilla.org/es/docs/Learn/HTML/Introduction_to_HTML/Getting_started
- min.io. (2022). *Almacenamiento de objetos de alto rendimiento MinIO: almacenamiento de objetos MinIO para contenedores*. <https://min.io/docs/minio/container/index.html>
- mongodb.com. (2022). *¿Qué Es MongoDB? | MongoDB*. <https://www.mongodb.com/es/what-is-mongodb>
- Montaña Posada, A. (2018). *Diseño de un modelo de gestión documental para industrias Haceb S.A para su posterior automatización a través de un software de gestión* [Universidad Pontificia Bolivariana]. <https://repository.upb.edu.co/handle/20.500.11912/5417>
- MyBatis – MyBatis 3 | Introducción. (2022). <https://mybatis.org/mybatis-3/es/index.html>
- normativa.archivogeneral.gov.co. (2012). *CIRCULAR EXTERNA 005 DE 2012 | Normatividad AGN*. <https://normativa.archivogeneral.gov.co/circular-externa-005-de-2012/>
- normativa.archivogeneral.gov.co. (2018). *LEY 594 DE 2000 | Normatividad AGN*. <https://normativa.archivogeneral.gov.co/ley-594-de-2000/>
- nycecolombia.co. (2015). *ISO/IEC 29110 – Ingeniería de Software – NYCE Colombia*.

<https://nycecolombia.co/isoiec-29110/>

opecolombia.com. (2022). *OPE Colombia | OPEColombia | SIEPDOC*.

<https://www.opecolombia.co/>

platzi.com. (2021). *Desarrollo Backend con Java*. <https://platzi.com/desarrollo-java/>

rabbitmq.com. (2022). *Messaging that just works — RabbitMQ*.

<https://www.rabbitmq.com/#features>

Ramírez Pérez, S. (2020). *Estudio del framework Spring, Spring Boot y microservicios*.

<https://ebuah.uah.es/dspace/handle/10017/45107>

Reyes Vargas, A. N., Buitrago Espitia, L. D., Arévalo Montes, S. Y., & Salazar Meza, J. M.

(2019). APLICATIVO WEB DE GESTIÓN DOCUMENTAL DE PROCESOS JUDICIALES PARA LA ORGANIZACIÓN, ACCESIBILIDAD Y SEGURIDAD DE LOS EXPEDIENTES Y ANEXOS DE CASOS JURÍDICOS. *Encuentro Internacional de Educación En Ingeniería*. <https://acofipapers.org/index.php/eiei/article/view/208>

Scrum.org. (2021). *What is Scrum?* <https://www.scrum.org/resources/what-is-scrum>

spring.io. (2021). *Spring | Web Applications*. <https://spring.io/web-applications>

Tovar Gonzales, M. (2021). *Concepto Técnico - Radicación y Ordenación de documentos*.

[https://www.archivogeneral.gov.co/sites/default/files/conceptostecnicos/2013/SUBGPD/Ra
dicado 1-2013-1296.pdf](https://www.archivogeneral.gov.co/sites/default/files/conceptostecnicos/2013/SUBGPD/Radicado%201-2013-1296.pdf)

vaadin.com. (2021). *Introduction*. <https://vaadin.com/docs/v14/guide/introduction>

ANEXO 1. PRODUCT BACKLOG

En este anexo se puede observar el formato de Estimación de Tamaño del Product Backlog basado en la ISO 29110. En la ilustración 119 y 120 se puede observar cómo se describen cada uno de los criterios para la estimación del tamaño, y en la ilustración 121 y 122 se puede apreciar el formato de Estimación de Tamaño para este trabajo.

Criterio	Descripción	Resultado esperado	Descripción del resultado
Funcionalidad	Se estima la funcionalidad del requerimiento.	D: Documento Co: Consulta R: Reporte P:Proceso	Documento: Master - Detalle Consulta: Modificación de un reporte ya realizado con anterioridad Reporte. Proceso: Lógica
Complejidad	Se estima la complejidad del requerimiento.	4-6: Simple 7-9: Medio 10-19: Complejo > 20: Muy Complejo	
CO	Cantidad de Operaciones (CO) que se estiman para el requerimiento	Cantidad de lecturas, escrituras, validaciones y cálculos que realizará el requerimiento.	
EI	Entidades Internas (EI) que se estiman para el requerimiento.	Cantidad de objetos propios de la aplicación que interactuarán en el requerimiento.	
EE	Entidades Externas (EE) que se estiman para el requerimiento.	Cantidad de objetos de otras aplicaciones que interactuarán en el requerimiento.	
O	Optimización (O): Optimizaciones especiales que requiere el requerimiento.	1 -> Ninguna 1.1 -> Concurrencia 1.2 -> Tunning 1.3 -> Ambas	
R	Reusabilidad (R): Reutilización de componentes y herramientas ya desarrolladas.	0.2 -> Alta 0.5 -> Media 0.85 -> Escasa 1 -> Nula	
D	Dominio (D): Conocimiento que se posee sobre el dominio del problema.	0.5 -> Conocimiento Alto 0.75 -> Conocimiento Medio 1 -> Sin conocimiento	

Ilustración 119 Descripción de criterios para estimación del tamaño parte 1

TU	Testing Unitario (TU): Realización de testing unitario.	1 -> Sin Testing 1.5 -> Con Testing	
DE	Documentación Específica (DE): Confección de documentación específica no estándar.	1 -> No 1.5 -> Sí	
TAM	Tamaño del Caso de Uso: (TAM).	$(CO+EI+EE) * O * R * D * TU * DE$	
T	Ajuste por Tecnología(T).	1 = Java, Flex4 1.5 = C#, PHP, .Net, HTML5, otros	
FA	Factor de Ajuste.	1-> Proceso 0.8 -> Documento 0.5-> Consulta, Reporte, Catalogo	
TAMA	Tamaño Ajustado por Tecnología (TAMA).	$TAM * T * FA$	

Ilustración 120 Descripción de criterios para estimación del tamaño parte 1

														
Estimación de Tamaño														
												937.4625	917.96	
Nº	Nombre de la historia de usuario	Funcionalidad	Complejidad	CO	EI	EE	O	R	D	TU	DE	Esfuerzo	T	FA
				Casos de operación	Entidad a interactuar	Entidad a consultar	Optimización	Reusabilidad	Domínio	Testing	Doc. Especifica	Tamaño de casos de uso	Ajuste por tecnología	Factor de ajuste
1	Login	P	MC	15	5	0	1,3	1,0	1,00	1,0	1,0	39,00	1,0	
2	Cerrar sesion	P	S	2	1	0	1,3	1,0	1,00	1,0	1,0	5,85	1,0	
3	Reestablecer contraseña de usuario	P	MC	10	5	0	1,3	0,9	1,00	1,0	1,0	24,86	1,0	
4	CRUD de usuarios	P	C	20	10	0	1,3	0,9	1,00	1,0	1,0	49,73	1,0	
5	CRUD de Roles y permisos	P	MC	20	10	0	1,3	0,9	1,00	1,0	1,0	49,73	1,0	
6	CRUD de funcionarios	P	C	10	3	0	1,3	0,9	1,00	1,0	1,0	21,55	1,0	
7	CRUD de los tipos de identificación	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
8	CRUD de los continentes	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
9	CRUD de los países	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
10	CRUD de los departamentos	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
11	CRUD de los municipios	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
12	CRUD de los medios de recepción	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
13	CRUD de los días no hábiles	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
14	CRUD de los tipos de radicados	P	C	15	4	0	1,3	0,9	1,00	1,0	1,0	31,49	1,0	
15	CRUD de los cargos	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
16	CRUD de los tipos de transacción	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	

Ilustración 121 Estimación de tamaño parte 1

														
Estimación de Tamaño														
												937.4625	917.96	
Nº	Nombre de la historia de usuario	Funcionalidad	Complejidad	CO	EI	EE	O	R	D	TU	DE	Esfuerzo	T	FA
				Casos de operación	Entidad a interactuar	Entidad a consultar	Optimización	Reusabilidad	Domínio	Testing	Doc. Especifica	Tamaño de casos de uso	Ajuste por tecnología	Factor de ajuste
9	CRUD de los países	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
10	CRUD de los departamentos	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
11	CRUD de los municipios	P	C	10	4	0	1,3	0,9	1,00	1,0	1,0	23,21	1,0	
12	CRUD de los medios de recepción	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
13	CRUD de los días no hábiles	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
14	CRUD de los tipos de radicados	P	C	15	4	0	1,3	0,9	1,00	1,0	1,0	31,49	1,0	
15	CRUD de los cargos	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
16	CRUD de los tipos de transacción	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
17	CRUD de las dependencias	P	C	20	6	0	1,3	0,9	1,00	1,0	1,0	43,10	1,0	
18	CRUD de los estados de los radicados	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
19	CRUD de los ciudadanos	P	C	20	12	0	1,3	0,9	1,00	1,0	1,0	53,04	1,0	
20	CRUD de los tipos de documentos	P	C	10	2	0	1,3	0,9	1,00	1,0	1,0	19,89	1,0	
21	Radicaciones de entrada	P	MC	40	30	0	1,3	0,9	1,00	1,0	1,0	116,03	1,0	
22	Radicaciones de salida	P	MC	40	30	0	1,3	0,9	1,00	1,0	1,0	116,03	1,0	
23	Otros tipos de radicaciones	P	MC	40	30	0	1,3	0,9	1,00	1,0	1,0	116,03	1,0	
24	Generar reportes	R	MC	15	5	0	1,3	1,0	1,00	1,0	1,0	39,00	0,5	

Ilustración 122 Estimación de tamaño parte 2

ANEXO 2. DAILY SCRUM

En este anexo podemos observar el formato de documento del Daily Scrum, o Scrum diario. El formato completo se puede observar en el documento anexo “Anexo 2. Scrum diario”, en la hoja “Scrum diario”.

Tabla 31 Daily Scrum

Fecha	Persona	Hoy	Mañana	Problemas
221011	Deimer Joel Cegarra Cordero	Reunión siepdoc java, terminar de paginar métodos en radicación, configuraciones docker a jhonathan bautista	reunión siepdoc java, terminar de paginar métodos en radicación, configuraciones docker al Scrum Master	
221010	Deimer Joel Cegarra Cordero	Agregar paginación a los métodos que hay que paginar y no son findAll, ida a la empresa	reunión siepdoc java, terminar de paginar métodos en radicación	
221008	Deimer Joel Cegarra Cordero	en la mañana sin luz a la hora del scrum, agregar paginación a los métodos que hay que paginar y no son findAll	agregar paginación a los métodos que hay que paginar y no son findAll, ida a la empresa	
221007	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, agregar paginación a los métodos que hay que paginar y no son findAll		
221006	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, reunión jhonathan bautista, problemas al paginar otros métodos diferentes al findAll, pero se solucionó, agregar paginación a los métodos que hay que paginar y no son findAll	reunión equipo siepdoc java, agregar paginación a los métodos que hay que paginar y no son findAll	
221005	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, agregar paginación los métodos que hay que paginar y no son findAll		
221004	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, liberar espacio en instancias de siepdoc, corregir componente paginación para paginar cualquier método	reunión equipo siepdoc java, agregar paginación los métodos que hay que paginar y no son findAll	
221003	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, continuar paginación en diálogos	reunión equipo siepdoc java, corregir componente paginación para paginar cualquier método	
221001	Deimer Joel Cegarra Cordero	reunión equipo siepdoc java, continuar paginación en diálogos, despliegue de siepdoc en las dos máquinas virtuales	reunión equipo siepdoc java, continuar paginación en diálogos	

ANEXO 3. HITOS

En este anexo podemos observar el formato de documento de los hitos basado en la ISO 29110. El formato completo se puede observar en el documento anexo “Anexo 3. Hitos”.

Tabla 32 Hitos

Descripción (hito)	Fecha Reunión
Reunión hitos iniciales	220330
Microservicios de configuracion (siepdocus_micro_eureka, siepdocus_micro_config, siepdocus_micro_sccu, siepdocus_micro_scc)	
Pantallas de configuración (crud de dependencia, usuarios, permisos, etc.)	
Reunión nuevos hitos (verificar avance, definición tiempos microservicios listado enviado por mario)	220423
Gestión de usuarios, falta FrontEnd (vistas)	
Microservicio de radicación BackEnd	
Reunión verificar avance	220430
Gestión de usuarios, falta FrontEnd Revisión del crud de cargos -> 220501 Martes, nueva revisión de vistas si no es aprobada, si es aprobado todo lo de usuarios	
Radicador BackEnd (se generó los mappers, proceso de generar las apis de todas las tablas)	
Reunión verificar avance	220507
Vista de roles, permisos	
Gestión de documentos, BackEnd (se generó los mappers, proceso de generar las apis de todas las tablas)	
Ya se tienen las tablas y se empezaron a hacer los micros Radicador - Ya se tienen las tablas	
Reunión avance	2205014
API de las tablas de configuracion (nuevas 13 tablas) (Deimer)	
Vistas de las tablas de configuracion (nuevas 13 tablas) (Deimer)	
Radicador (Deimer) ya se tienen las tablas	

ANEXO 4. MANUAL DE OPERACIONES Y MANTENIMIENTO

En esta sección se presentan parte del manual de operación y mantenimiento, en el documento anexo “Anexo 4. SIEPDOG - Manual de operación y mantenimiento” podemos observar el manual completo con la configuración del entorno de producción.

Instalación de la aplicación

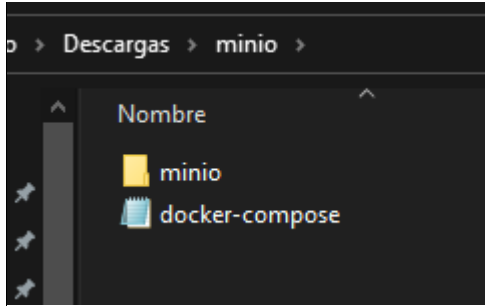
Aplicaciones que son necesarias para el desarrollo de la aplicación.

MinIo

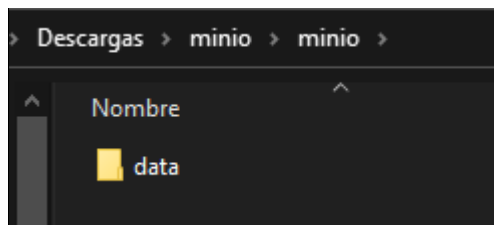
Para la instalación de minio en docker desktop debemos tener la siguiente configuración en un archivo llamado ‘docker-compose.yml’:

```
version: '3.0'
services:
  minio:
    image: minio/minio
    ports:
      - "9000:9000"
      - "9001:9001"
    networks:
      - default
    volumes:
      - "./minio/data:/data"
    environment:
      MINIO_ROOT_USER: user
      MINIO_ROOT_PASSWORD: password
    command: server /data --console-address ":9001"
```

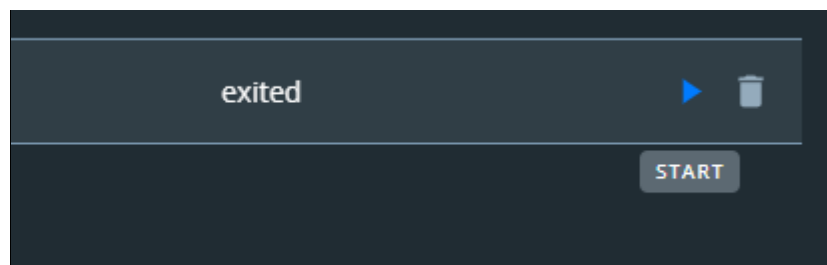
Este archivo debe estar dentro de una carpeta preferiblemente llamada ‘minio’, y junto a otra carpeta llamada ‘minio’, ejemplo:



Dentro de la segunda carpeta 'minio' debemos crear otra carpeta llamada 'data':



Por último, nos ubicamos en la ruta de la primera carpeta llamada 'minio' en la consola de comandos CMD o símbolo del sistema y ejecutamos el comando 'docker-compose up', este comando descarga la imagen docker y crea el contenedor que luego aparece en el docker desktop. Después de este último paso el contenedor ya queda desplegado, pero podemos cerrar la consola CMD y desplegarlo desde docker desktop, dando click en el botón 'start'.



RabbitMQ

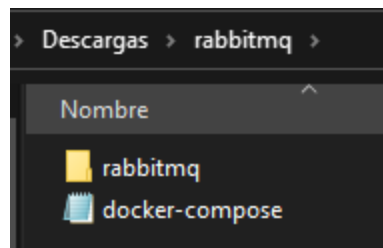
Seguir los mismos pasos de la instalación de minio.

```

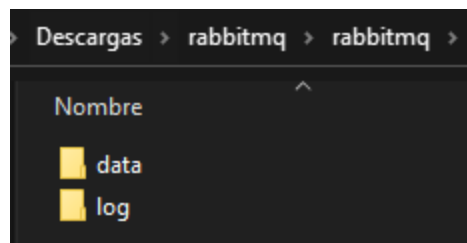
version: '3.2'
services:
  minio:
    image: rabbitmq:3.10.5-management
    container_name: 'rabbitmq'
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - rabbitmq_go_net
    volumes:
      - "./rabbitmq/data:/var/lib/rabbitmq/"
      - "./rabbitmq/data:/var/log/rabbitmq/"
networks:
  rabbitmq_go_net:
    driver: bridge

```

crear carpeta llamada rabbitmq:



subcarpeta rabbitmq:



Mongo y Mongo-Express

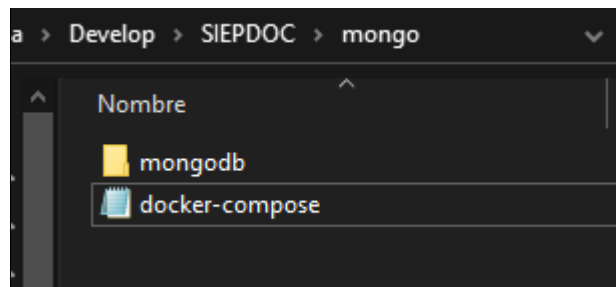
Seguir los mismos pasos de la instalación de minio.

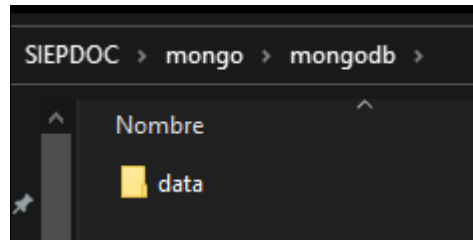
```

version: '3.0'
services:
  mongodb:
    image: mongo
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: siepdoc
      MONGO_INITDB_DATABASE: db_prueba_mongo
    volumes:
      - "./mongodb/data:/data/db"
    ports:
      - "27017:27017"
    networks:
      - default
  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - "8081:8081"
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: siepdoc
      ME_CONFIG_MONGODB_URL: mongodb://root:siepdoc@mongodb:27017/
      ME_CONFIG_MONGODB_AUTH_USERNAME: root
      ME_CONFIG_MONGODB_AUTH_PASSWORD: siepdoc
    networks:
      - default
  depends_on:
    - mongodb

```

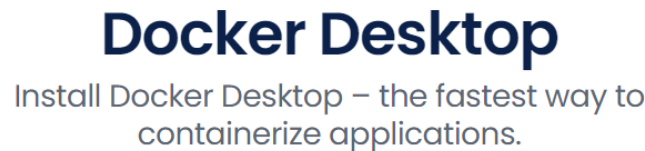
Estructura de carpetas, la primera carpeta preferiblemente llamarla 'mongo' y la subcarpeta 'mongodb':





Docker Desktop

Para la instalacion de docker desktop para Windows debemos ir a la dirección <https://www.docker.com/products/docker-desktop/> y descargar la opción para Windows.

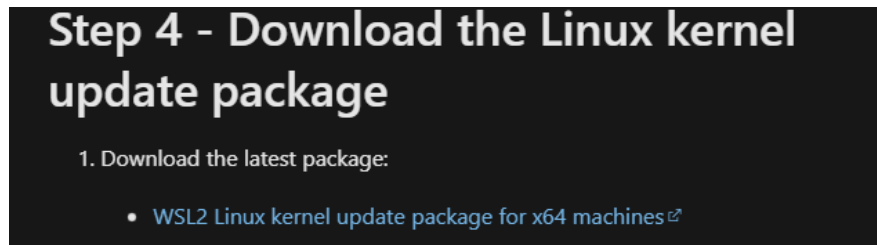


Luego ejecutamos el instalador descargado para que comience a descarga los componentes necesarios. Nos debe salir una pantalla de configuracion donde sale una opción que dice “Install required Windows components for WSL 2” y la seleccionamos, luego damos “ok”.

Importante: antes de continuar debemos tener habilitada la virtualización en nuestro equipo.

Luego de terminar las instalaciones debemos reiniciar nuestro equipo, si luego de esto nos aparece un aviso al abrir docker desktop que dice “WSL 2 installation is incomplete”, debemos ir

a la dirección que nos dice para descargar un instalador de paquetes para Linux o a la dirección <https://learn.microsoft.com/en-us/windows/wsl/install-manual> y dar clic en el siguiente enlace:



Luego de descargarlo solo debemos ejecutar el instalador y seguir los pasos. Ahora debemos habilitar el paquete WSL. Para esto, abrimos una consola de PowerShell como administrador (CMD no sirve en este caso) y ejecutamos el siguiente comando:

```
Enable-WindowsOptionalFeature -Online -FeatureName $("VirtualMachinePlatform",  
"Microsoft-Windows-Subsystem-Linux")
```

El comando anterior habilitará el paquete WSL, ahora podemos cerrar la consola y debemos reiniciar nuevamente nuestro equipo. Ahora ya podemos abrir docker desktop y utilizarlo.

PostgreSQL versión 14

Para la instalación de PostgreSQL es necesario dirigirse a la siguiente dirección para descargar el sistema de gestión y administración de base de datos, PgAdmin 4: <https://www.pgadmin.org/download/pgadmin-4-windows/>. Elegimos la descarga de la versión que queremos como se muestra en la siguiente imagen:

pgAdmin 4 (Windows)

Download

Maintainer: pgAdmin Development Team

pgAdmin is available for 64 bit Windows™ 7 SP1 (desktop) or 2008R2 (server) and at v5.0 and later are supported on Windows 8 (desktop) or 2012 (server) and above.

32 bit Windows support is available for versions up to v4.29.

The packages below include both the Desktop Runtime and Web Application:

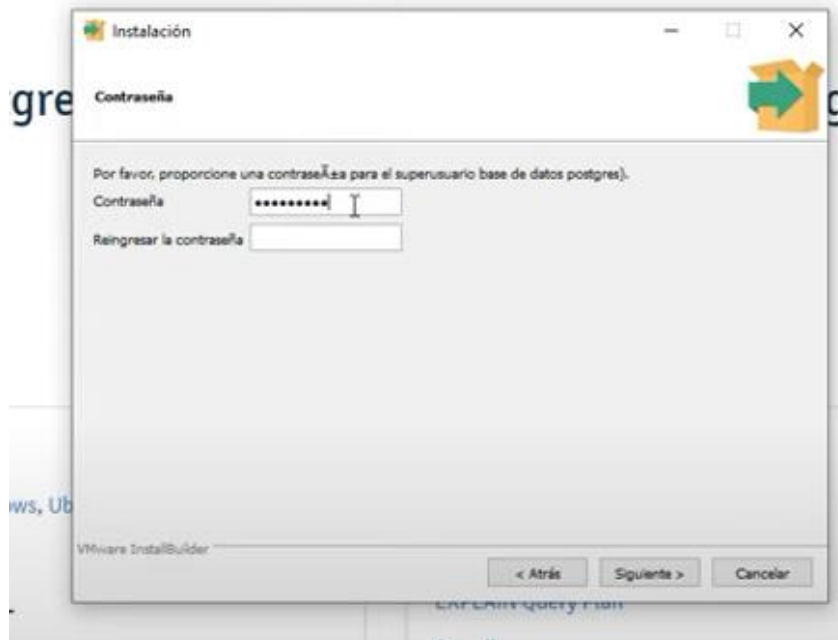
- 📄 pgAdmin 4 v6.17 (released Dec. 2, 2022)
- 📄 pgAdmin 4 v6.16 (released Nov. 19, 2022)
- 📄 pgAdmin 4 v6.15 (released Oct. 20, 2022)
- 📄 pgAdmin 4 v6.14 (released Sept. 22, 2022)
- 📄 pgAdmin 4 v6.13 (released Aug. 25, 2022)
- 📄 pgAdmin 4 v6.12 (released July 28, 2022)
- 📄 pgAdmin 4 v6.11 (released June 30, 2022)

Luego ejecutamos el instalador y seguimos los pasos, dejamos todas las opciones por defecto.

Ahora debemos instalar el motor de base de datos PostgreSQL 14, para esto nos debemos dirigir a la dirección <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads> y descargar la versión 14 como se muestra en la siguiente imagen:

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64
15.1	postgresql.org 📄	postgresql.org 📄		
14.6	postgresql.org 📄	postgresql.org 📄		
13.9	postgresql.org 📄	postgresql.org 📄		
12.13	postgresql.org 📄	postgresql.org 📄		

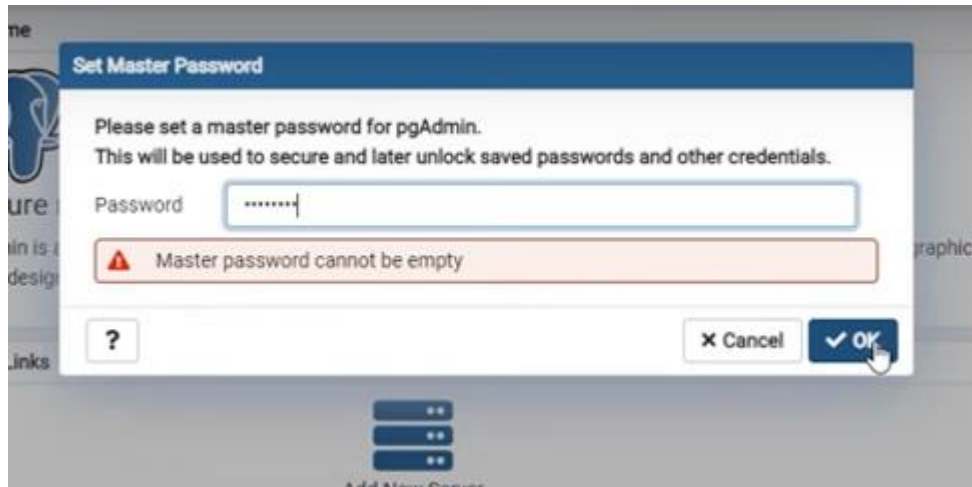
Luego ejecutamos el instalador y seguimos los pasos, nos pedirá ingresar una contraseña para el servidor de PostgreSQL 14 como se muestra en la siguiente imagen:



Luego dejamos el puerto y las demás configuraciones por defecto. Finalmente nos saldrá una pantalla para instalaciones adicionales si lo deseamos o damos en cancelar.



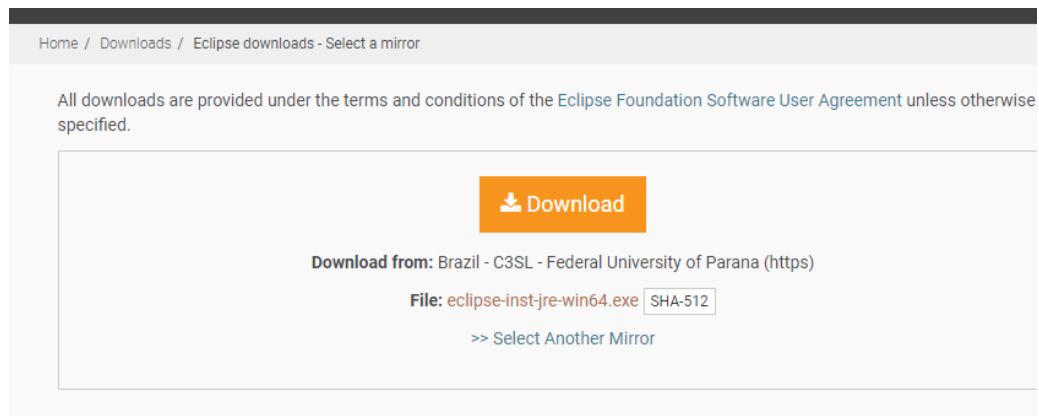
Al ingresar al PgAdmin por primera vez nos solicitara crear una contraseña como se muestra en la siguiente imagen:



Damos en ok y ya deberíamos poder acceder a nuestros servidores, en este caso al de PostgreSQL 14.

Eclipse IDE

Para la instalacion de Eclipse IDE debemos dirigirnos a la dirección <https://www.eclipse.org/downloads/download.php?file=/oomph/products/latest/eclipse-inst-jre-win64.exe> para instalar la versión de Windows. Damos clic en el botón “Download” para descargar el ejecutable.



Ejecutamos el .exe y buscamos “Eclipse IDE for Enterprise Java and Web Developers”, luego seleccionamos la versión de java que tenemos en nuestro equipo para configurar eclipse con esa versión, como se muestra en la siguiente imagen:



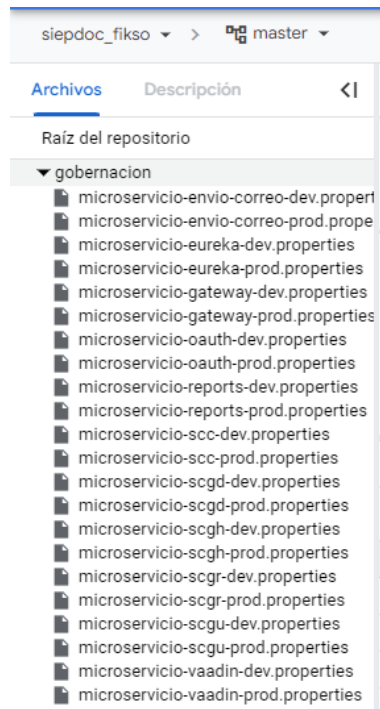
Finalmente damos clic en la opción “INSTALL” y esperamos que termine la instalación.

Configuración de la aplicación

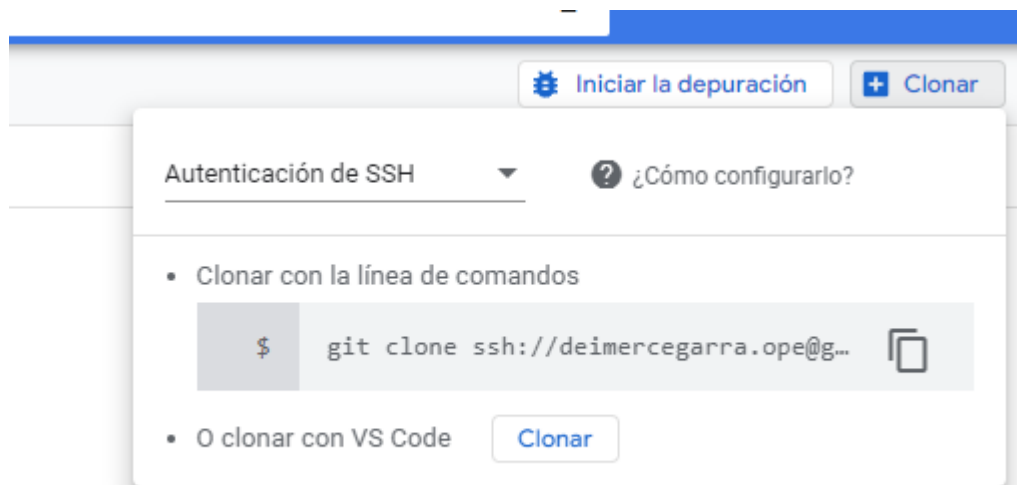
Se definen los componentes ya instalados que se deben configurar para el correcto funcionamiento de la aplicación, para estas configuraciones debemos tener las instalaciones del capítulo “Instalar entorno de desarrollo”.

Configuración archivos de configuración

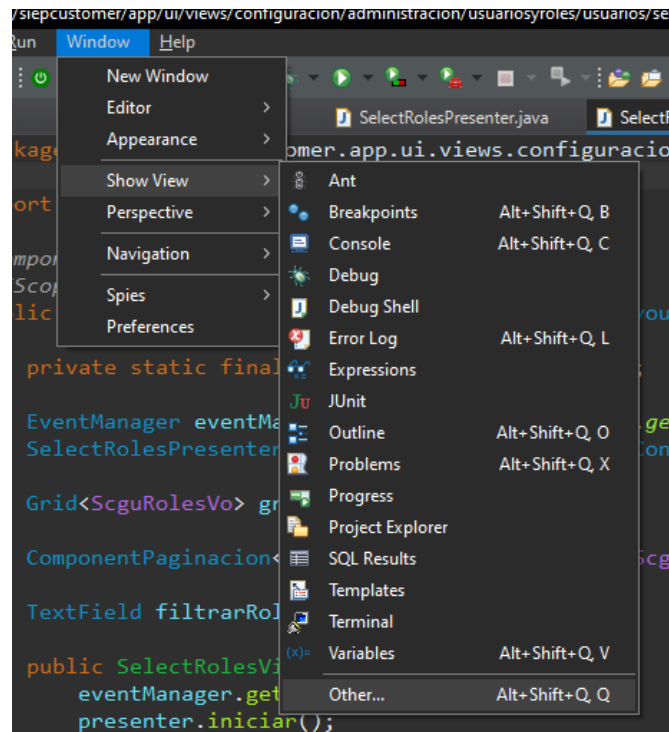
Debemos clonar el repositorio que posee los archivos de configuración externalizados de la aplicación. Por ejemplo, clonamos el siguiente repositorio:



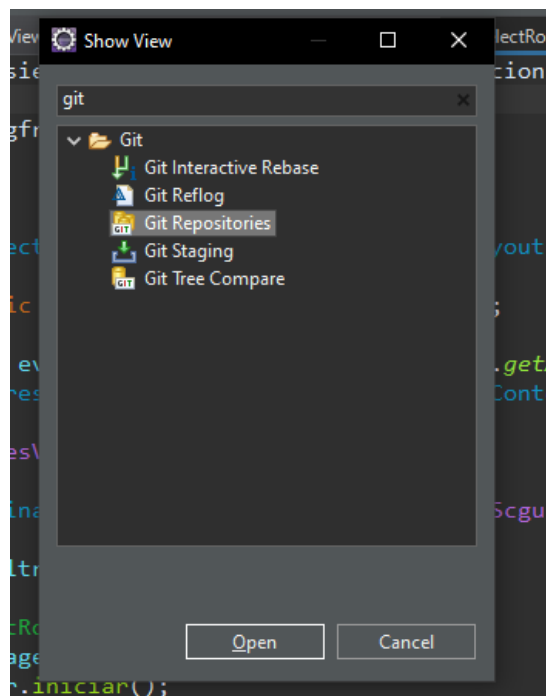
Para clonarlo, debemos configurar el repositorio como se muestra en el capítulo 4.1.2, luego debemos copiar el comando ssh para poder clonarlo e irnos a eclipse:



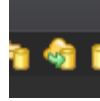
En eclipse debemos ubicarnos en Window > Show View > Other:



Luego buscamos y seleccionamos Git Repositories:



Luego damos en la opción de clonar repositorio:



Debería salir la siguiente pantalla con la información copiada automáticamente, debemos ingresar la password de nuestra clave ssh y marcamos la opción “Store in secure store” para que no la vuelva a solicitar:

Clone Git Repository

Source Git Repository
Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

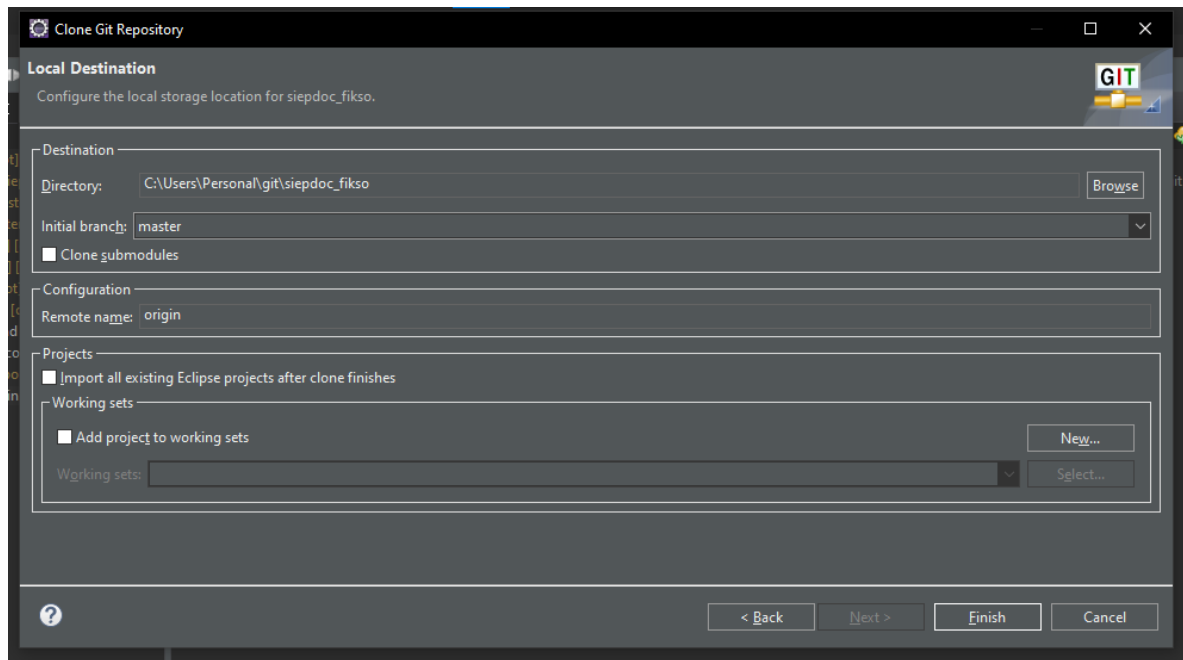
Authentication

User:

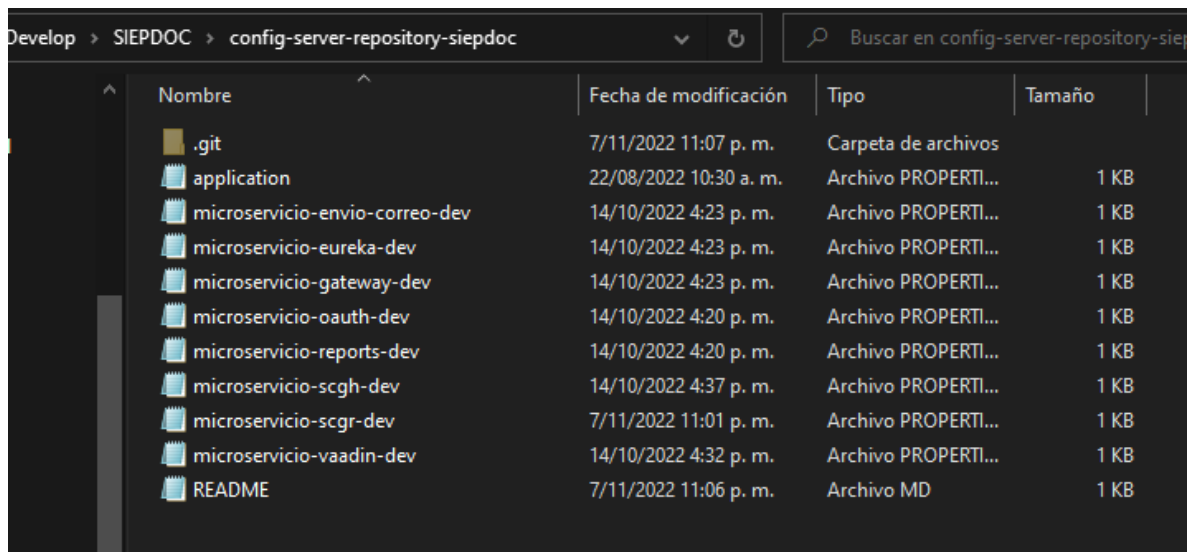
Password:

☒ Store in Secure Store

Luego damos next > next y nos debería salir la siguiente pantalla donde seleccionamos la ubicación donde queremos que quede nuestro proyecto clonado, normalmente se ubican en un mismo Workspace todos los proyectos:



Luego de esto debemos crear un repositorio local donde copiamos los archivos de propiedades dev, los cuales son los que necesitamos para desarrollo.



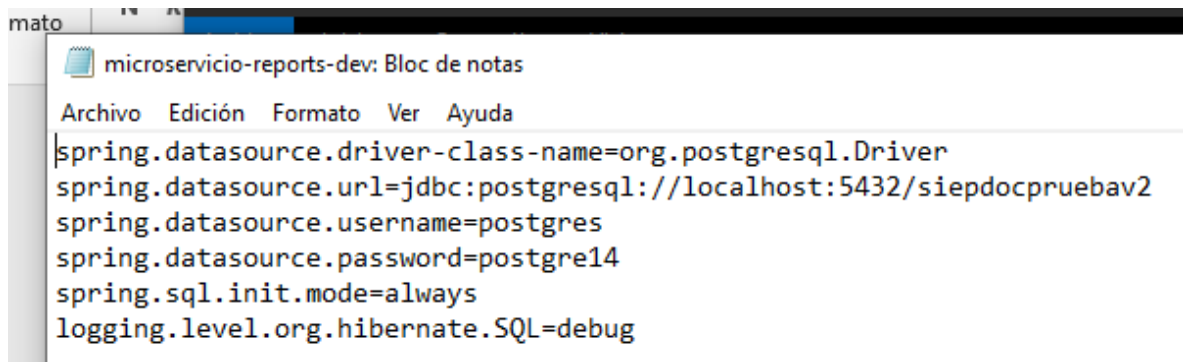
Después de esto debemos irnos al archivo de propiedades del microservicio Siepdoc_micro_config y reemplazar la dirección del repositorio que aparezca por la de nuestro

repositorio, en la siguiente imagen aparece la dirección de un repositorio que se encuentra en Github:

```
1 spring.application.name=servicio-config-server
2 server.port=8888
3
4 spring.cloud.config.server.git.uri=https://github.com/DeimerJC/config-server-repository-siepdoc.git
5 #spring.cloud.config.server.git.search-paths=gobernacion
```

Configurar base de datos

En esta sección solo debemos ir a crear nuestra base de datos y luego ir a cada uno de los archivos de propiedades del capítulo anterior y configurar nuestra base de datos:

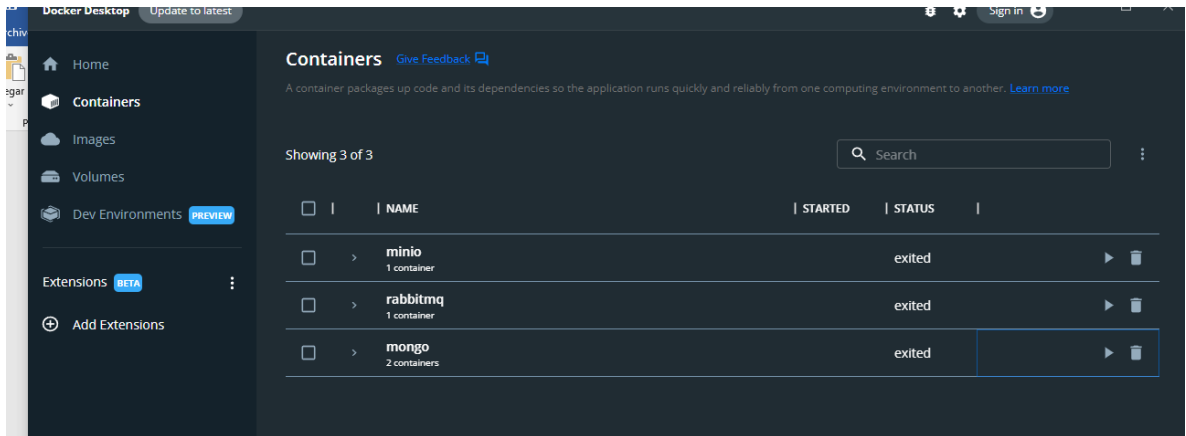


```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/siepdocpruebav2
spring.datasource.username=postgres
spring.datasource.password=postgre14
spring.sql.init.mode=always
logging.level.org.hibernate.SQL=debug
```

Prueba inicial de instalacion correcta

Para el correcto inicio y funcionamiento de SIEPDOC en entorno de desarrollo debemos ejecutar los microservicios y demás aplicaciones en el siguiente orden:

1. Primero debemos abrir Docker desktop e inicializar los contenedores creados anteriormente de MinIo, RabbitMQ y Mongo:



Para iniciarlos debemos dar clic en el botón Start:



2. Debemos ejecutar los microservicios en eclipse en el siguiente orden:

Siepdoc_micro_config > Siepdoc_micro_eureka > Siepdoc_micro_scgu >

Siepdoc_micro_scc > Siepdoc_micro_scgd > Siepdoc_micro_scgr >

Siepdoc_micro_scgh > Siepdoc_micro_reports > Siepdoc_micro_envio_correo >

Siepdoc_micro_oauth > Siepdoc_micro_gateway > Siepdoc_micro_vaadin.

Finalmente, cuando termine de ejecutarse vaadin, debería abrirse automáticamente en el navegador predeterminado el login para el inicio de sesión en el puerto 8084.