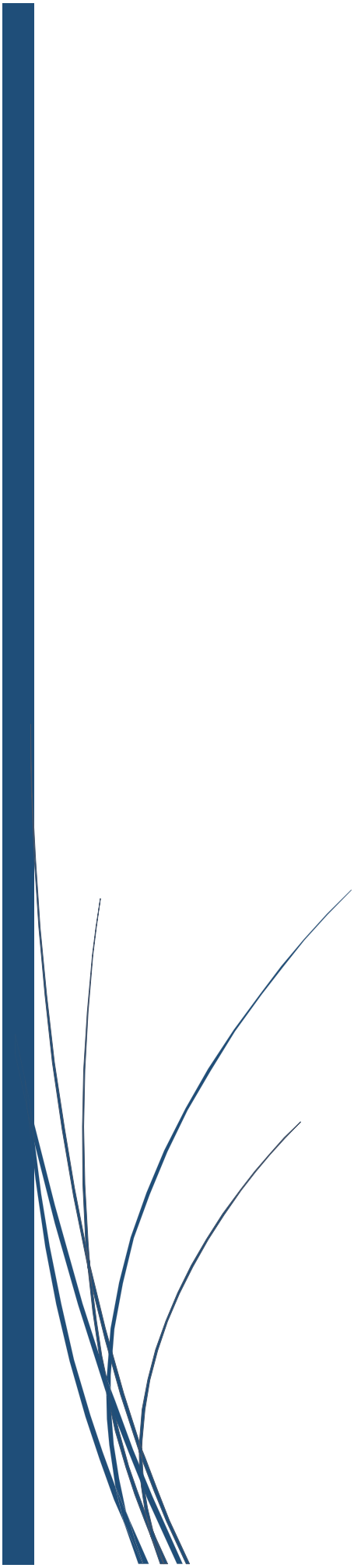
Manual Técnico
Mesa de Entrada

Sistema de Atención al Contribuyente (SAC)



HEWLETT-PACKARD COMPANY

Tabla de contenido

1. Breve descripción del Módulo de Mesa de Entrada.

Mesa de Entrada es el módulo encargado de la entrega de tiquetes para que los tramites del contribuyente sean atendidos de forma ordenada., se presenta como una herramienta que proporciona la interfaz gráfica en la cual se establece las condiciones de atención en los centros de servicio del Ministerio de Hacienda.

La pantalla de Mesa de Entrada condiciona el tramite a realizar, tiempos, prioridades entre otros parámetros que optimizan que los servicios proporcionados por el Ministerio de Hacienda sean gestionados de forma ordenada.

Dentro del sistema SAC la Mesa de Entrada posee opciones para mantener la estabilidad y optimización del servicio de atención. Cada tiquete es personalizado con los trámites a realizar por el contribuyente.

El primer paso del proceso de atención al contribuyente por medio de este sistema es por medio de la mesa entrada, la cual configura los trámites y proporciona el tiquete con su número para ser atendido en el centro de servicio.

2. Diagrama Entidad – Relación

A continuación se listan las diferentes tablas utilizadas en el módulo Mesa de Entrada. La siguiente tabla proporciona el esquema al que pertenece y el nombre de cada una de las tablas.

Tabla	Esquema	Anexo
GC_CONF_TRAMITE	GESTION.COLAS	Anexo 1
GC_SERVICIOS		
GC_TIQUETE		
GC_CONF_TIQUETE		
GC_PRIORIDAD		
GC_RESERVA_CITA		
TB_UNIDAD_RECEP	CATALOGOS	Anexo 2
RC_RUC	RUC	Anexo 3

Tabla 2.1 Diagrama ER.

3. Estructura de Componentes (Capa vista, controlador, servicio DAO y estructura de reportes)

Vistas o páginas HTML

Nombre de opción	Ubicación	Nombre
Confirmación de Cita	/Other Sources/src/main/resources/templates.me	ReservaCita.html
Reasignación de Tiquete		reasignacionTiquete.html
Escalamiento		escalamientoTiquete.html
Reimpresión de tiquete		reimpresionTiquete.html
--		TiqImp.html Home.html

Tabla 3.1 Vistas o paginas HTML

Controladores

En la siguiente tabla se proporcionan los controladores utilizados en las opciones del módulo de Mesa de Entrada. Estos controladores se encuentran almacenados en el siguiente paquete:

sv.gob.mh.dgii.colas.controllers.me
sv.gob.mh.dgii.colas.controllers

Opción	Controlador
--	MenuController.java
--	OpcionesController.java
Confirmación de Cita	ReservaCitaController.java
Reasignación de Tiquete	ReasignacionTiqueteController.java
Escalamiento	EscalamientoTiqueteController.java
Reimpresión de tiquete	ReasignacionTiqueteController.java

Tabla 3.2 Lista de Controladores

Descripción de Controladores.

A continuación se proporciona una descripción de la funcionalidad principal de los controladores utilizados en el módulo de Mesa de Entrada.

PANTALLA: MENU	
Controlador:	**MenuController.java**
Descripción de Métodos de Negocio	
Este controlador contiene los métodos que despliegan las opciones utilizadas en la Mesa de Entrada. **@RequestMapping("/me/home")** **public String irMesaEntrada(Model model):** Método que se encarga de proporcionar la pantalla de inicio para la mesa de entrada, se utiliza el método getDisplayName() para proporcionar el nombre y establecer el usuario que ha iniciado sesión. Esta pantalla se carga haciendo uso de la página home.html. **@RequestMapping("/me/reservaCita")** **public String irReservaCita(Model model):** Método utilizado para inciar la pantalla de reserva de cita, se utiliza el método getDisplayName() para establecer el nombre del usuario y por medio de la validación proporcionar el usuario y cargar la pagina reservaCita.html **@RequestMapping("/me/reasignacionTiquete")** **public String irReasignacionTiquete(Model model):** Método que se usa para iniciar la pantalla de Reasignación de Tiquete, se establece el nombre del usuario por medio del método getDisplayName() y luego por medio de la validación se obtienen los datos adicionales. Para este proceso se utiliza la página reasignacionTiquete.html **@RequestMapping("/me/escalamientoTiquete")** **public String irEscalamientoTiquete(Model model):** Este método que se usa para proporcionar la pantalla de Escalamiento de Tiquetes, esta opción es desplegada utilizando la página escalamientoTiquete.html. Se obtiene el nombre del usuario por medio de la invocación del método getDisplayName(). **@RequestMapping("/me/reimpresionTiquete")** **public String irReimpresionTiquete(Model model):** Método que se usa para cargar la pantalla de reasignación de tiquete, se captura el nombre del usuario por medio de la invocación del método getDisplayName(). Para este proceso se carga la página reimpresionTiquete.html **@RequestMapping("/admin/home")** **public String irAdministracion():** Método que carga la página index.html para la pantalla de administración.	

Tabla 3.3 Controlador Menú.

PANTALLA: CONFIRMACIÓN DE CITA	
Controlador:	**ReservaCitaController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value="/rcta/data" , method=RequestMethod.GET, headers="Accept=application/json", produces="application/json") **public @ResponseBody List<ReservaCitaPojo> getData():** Método que obtiene la información de reserva de citas asociadas, proporcionando la lista de reservaciones. Utiliza la clase pojo ReservaCitaPojo.java para este proceso y establecer los parámetros de la reserva de cita como IdReservaCita, IdTramite, fecha, nit y usando los métodos verificarReserva() y actualizarEstado() del repositorio gcReservaCitaRepository para conocer el estado de la reserva y ser actualizado si la reserva cumple con la condición determinada. Se utiliza la clase ReservaCitaPojo para llenar los parámetros de la reserva como área de servicio, código de verificación, correo, fecha, hora, nit, teléfono, tramite, unidad o centro de servicio, estado y los id de la reserva y tramite. Los métodos invocados en este proceso tienen como objetivo obtener los parámetros anteriormente mencionados tomando como referencia el id de dichos parametros. **@RequestMapping(value="/rcta/enableReservacion" , method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)** **public @ResponseBody Integer enableReservacion(@RequestBody ReservaCitaPojo rctaPojo):** Método que se usa para deshabilitar la reserva de una determinada cita. Mediante el método verifyIfExistsTramiteAsignado() del repositorio gcTiqueteRepository se verifica si e tramite ya ha sido asignado previamente y se guarda en la variable countTr. Posteriormente se genera la condición sobre la variable countTr que sea mayor que cero, es decir que el trámite exista, luego se establecen los parámetros de tipo string identificador y correlativo que determinan la zona del trámite y el correlativo. Se usa la variable de tipo fecha fhReservacion para proporcionar la fecha de reservación de la cita. Creando la instancia GcTiquete se establecen los parámetros de unidad receptora, id de trámite, número de tiquete, correlativo e identificador, nit, estado, fecha de llegada, usuario y número de la reserva de cita. Utilizando la fecha se establece la prioridad y se actualiza el estado del tiquete. **public String getUnidad():** Método que retorna la unidad, se crea la variable de tipo string unidad y se invocan los métodos getPrincipal() y getUnidadRecep(). Posteriormente si la variable unidad se encuentra nula o vacia se usa el método getCsCombinacion() de repositorio gcUnidadRecepRepository.	

Tabla 3.4 Controlador Confirmación de Cita.

PANTALLA: REASIGNACIÓN DE TIQUETE	
Controlador:	**ReasignacionTiqueteController.java**
Descripción de Métodos de Negocio	

@RequestMapping(value="/reaTiq/data" , method=RequestMethod.GET, headers="Accept=application/json", produces="application/json")
public @ResponseBody List<ReasignacionTiquetePojo> getData(): Método que se encarga obtener la información asociada al tiquete y realizar la reasignación de este hacia otro servicio o trámite. Usando la clase ReasignacionTiquetePojo se crea el objeto de tipo lista listTiquete en el cual se proporcionará la información de la reasignación del tiquete. En este proceso se establecen la variables correspondientes y se utiliza el repositorio gcTiqueteRepository para invocar el método getListTiquetes().

Se genera la instancia ReasignacionTiquetePojo para que reciba nuevos parámetros por medio del objeto reaTiq. Se le proporcionan los parámetros como estado, fecha, hora, id del servicio, etc. En este proceso se invocan métodos con la finalidad de obtener dichos parámetros. Una vez han sido establecidos los nuevos parámetros se agregan a la variable listTiquete con lo cual concluye el proceso de reasignación.

public String getUnidad(): Método que retorna la unidad, se crea la variable de tipo string unidad y se invocan los métodos getPrincipal() y getUnidadRecep(). Posteriormente si la variable unidad se encuentra nula o vacia se usa el método getCsCombinacion() de repositorio gcUnidadRecepRepository.

Tabla 3.5 Controlador Reasignación de Tiquete.

PANTALLA: ESCALAMIENTO	
Controlador:	**EscalamientoTiqueteController.java**
Descripción de Métodos de Negocio	

@RequestMapping(value="/escTiq/services" , method=RequestMethod.GET, headers="Accept=application/json", produces="application/json")
public @ResponseBody List<GcServicios> getServices(): Método utilizado para proporcionar los servicios por medio del método getAllServicios() del repositorio gcServiciosRepository. Los servicios son presentados por medio del objeto de tipo lista listServicios.

@RequestMapping(value="/escTiq/tramitesByService" , method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
public @ResponseBody List<GcConfTramite> getTramitesByService(@RequestBody Map<String, String> map): Método en el cual se establece el proceso para obtener los tramites disponibles por medio del servicio indicado. Se usa la variable de tipo lista listTramites para obtener la lista de tramites invocando al reporsitorio gcConfTramiteRepository y su método listaTramitesEscByCS(). Luego en la variable t es recorrida seteando los parámetros requeridos para el escalamiento de tiquete.

PANTALLA: ESCALAMIENTO	
Controlador:	**EscalamientoTiqueteController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value="escTiq/insertData", method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE) **public @ResponseBody GcTiquete insertData(@RequestBody Map<String, Long> map):** Este método tiene como función insertar la información al tiquete para su escalamiento y que se almacenada. Se usa la variable t de tipo GcTiquete para proporcionar los parámetros del tiquete y tramite entre los cuales se setean unidad receptora, prioridad, número de tiquete, correlativo, estado, hora de llegada, numero de prioridad. Posteriormente se verifica si el tramite asignado existe por medio del método verifyIfExistsTramiteAsignado() usando el repositorio gcTiqueteRepository este resultado se almacena en la variable countTr. La información es almacenada en la variable tiquete invocando el método save(), caso contrario se genera una nueva instancia GcTiquete para que se puedan ingresar nuevos datos. **public String getUnidad():** Método que retorna la unidad, se crea la variable de tipo string unidad y se invocan los métodos getPrincipal() y getUnidadRecep(). Posteriormente si la variable unidad se encuentra nula o vacía se usa el método getCsCombinacion() de repositorio gcUnidadRecepRepository.	

Tabla 3.6 Controlador Escalamiento.

PANTALLA: REIMPRESIÓN DE TIQUETE	
Controlador:	**ReimprimirTiqueteController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value="/reimTiq/data" , method=RequestMethod.GET, headers="Accept=application/json", produces="application/json") **public @ResponseBody List<ReasignacionTiquetePojo> getData():** Método que se encarga obtener la información asociada al tiquete y realizar la reasignación de este hacia otro tiquete. Usando la clase ReasignacionTiquetePojo se crea el objeto de tipo lista listTiquete en el cual se proporcionará la información de la reasignación del tiquete. En este proceso se establecen la variables correspondientes y se utiliza el repositorio gcTiqueteRepository para invocar el método getListTiquetesEnEspera(). Esto permite establecer el nuevo tiquete que será proporcionado al contribuyente basado en los tiquetes actuales en espera. Se genera la instancia ReasignacionTiquetePojo para que reciba nuevos parámetros por medio del objeto reaTiq. Se le proporcionan los parámetros como estado, fecha, hora, id del servicio, etc. En este proceso se invocan métodos con la finalidad de obtener dichos parámetros por ejemplo getNTramiteId(), getNServiciosId() entre otros. Una vez han sido establecidos los nuevos parámetros se agregan a la variable listTiquete con lo cual concluye el proceso de reimpresión de tiquete.	

PANTALLA: REIMPRESIÓN DE TIQUETE	
Controlador:	**ReimprimirTiqueteController.java**
Descripción de Métodos de Negocio	
public String getUnidad(): Método que retorna la unidad, se crea la variable de tipo string unidad y se invocan los métodos getPrincipal() y getUnidadRecep(). Posteriormente si la variable unidad se encuentra nula o vacia se usa el método getCsCombinacion() de repositorio gcUnidadRecepRepository.	

Tabla 3.7 Controlador Reimpresión de Tiquete.

PANTALLA: OPCIONES	
Controlador:	**OpcionesController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value="/me/getTramites" , method=RequestMethod.GET, headers="Accept=application/json", produces="application/json") **public @ResponseBody List<GcConfTramite> servicios(ModelMap map):** Método que proporciona los trámites o servicios disonibles basados en centro de servicio. La variable cunidadRecep invoca el método getUnidad() para establecer el centro de servicio, posteriormente forma parámetro del método listaTramitesByCS() invocado por medio del repositorio GcConfTramiteRepository. El proceso anterior permite acceder a la lista de tramites basado en el centro de servicio y se almacenan al objeto de tipo lista listTramites. Se crea la variable de tipo GcConfTramite tramite en la cual se setean los parámetros como usuarios, la lista de reserva de citas, lista de tiquetes, números de servicio id, y tramites. Así como también se setean los parámetros RcTramites, EdDeclaraciones y EDNotasdeAbono. **@RequestMapping(value="me/getPrioridades", method=RequestMethod.GET, headers="Accept=application/json", produces="application/json")** **public @ResponseBody List<GcPrioridad> getAllPrioridades(ModelMap map):** Método encargado de obtener las prioridades de los tramites, se listan por medio de la invocación del método getAllPrioridades() del repositorio gcPrioridadRepository en variable listPrioridades. **@SuppressWarnings("unchecked")** **@RequestMapping(value="me/leerTiquete", method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)** **public @ResponseBody String leerTiquete(@RequestBody Map<String, String> map):** Método cuya función es proporcionar la lectura del tiquete, se accede por medio del método leerTiquete() en el cual se usa como parámetro el id y cusuario, dicho método es invocado a través del repositorio GcTiqueteRepository. La imagen es obtenida por medio getimg() método que pertenece al mismo repositorio. Al objeto JSON se le proporcionan los parámetros correspondientes a la lectura del tiquete.	

PANTALLA: OPCIONES	
Controlador:	**OpcionesController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value="me/insertData", method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE) public @ResponseBody GcTiquete insertData(@RequestBody OpcionesPojo opciones, ModelMap map): Método encargado de ingresar la información correspondiente a un tiquete o reasignación basado en los parámetros proporcionados. Al ingresar un nuevo tiquete se establecen los parámetros id de correlativo, prioridad, hora de llegada; caso contrario que el tiquete no se encuentra vacío se realiza una reasignación usando el repositorio gcTiqueteRepository y su método getIdTiquete() para validar el estado del tiquete y cambiar si es igual a uno con el método changeStatusTiquete(). Se establecen los parámetros de reasignación los id de tiquete, hora inicial y fina de proceso, hora de llamando, número de tiquete reasignado. Luego los parámetros de tipo fecha como fecha de reasignación y hora de llegada por medio de los métodos getfhDateReasignacion() y getTiempoHolgura() ambos de repositorio. La tiempo de holgura para el tiquete se determina con el método getTiempoHolgura() y posteriormente se setean los datos de trámite, centro de servicio, estado, etc. Finalmente se determina por medio de la condición el trámite, si el tiquete no posee tramite se encuentra por medio de método fin done del repositorio tramitesRepository. Caso contrario se genera un nuevo tiquete seteando los parámetros correspondientes para su generación. **@RequestMapping(value="me/validarNIT", method=RequestMethod.POST, headers="Accept=application/json", produces=MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE) public @ResponseBody PersonaPojo validarNIT(@RequestBody OpcionesPojo opcion, ModelMap map):** Método que se usa para la verificación del NIT, se utiliza la clase PersonaPojo y se crea el objeto persona en el cual se proporciona los datos del contribuyente como nombres y nit. En este proceso se usa el método findByNit() desde el repositorio rcRucRepository.	

Tabla 3.8 Controlador Opciones.

4. Clases POJO

En la siguiente tabla se listan las diferentes clases POJO que se usan en el Modulo Mesa de Entrada y los diferentes paquetes en donde se pueden encontrar.

Nombre de Opción	Ubicación	Clase POJO
--	sv.gob.mh.dgii.colas.pojos.entities	OpcionesPojo.java
		PersonaPojo.java
--	sv.gob.mh.dgii.colas.pojos.security	PersonW.java
Reserva de Cita	sv.gob.mh.dgii.colas.pojos.entities	ReservaCitaPojo,java
Reasignación de Tiquete		ReasignacionTiquetePojo
Reimpresión de tiquete		ReasignacionTiquetePojo.java

Tabla 4.1 Clases Pojo.

Descripción de Clases POJO.

A continuación se proporciona una breve descripción de las clases POJO usadas en las opciones del módulo Mesa de Entrada.

Nombre de Opción	Clase POJO	Descripción
--	OpcionesPojo.java	Clase en la cual se declaran los métodos y variables que se utilizan en las opciones para el tiquete. Las variables declaradas son las siguientes: private Long tramite; private Long prioridad; private Integer holgura; private String nit; private String opt; private Long idTiquete;
	PersonaPojo.java	Clase en la cual se declaran las variables escalares que se utilizan para gestionar los datos de un determinado contribuyente. Las siguientes variables son las que se encuentran en esta clase: private static final long serialVersionUID = 1L; private String nit; private String s1apeRasoc; private String s2apeAbrev; private String SNombres; private Date FNacConst; private Integer BDomiciliado; private Date FTermino; private Integer BInteresFiscal; private String MSexo; private Integer BImportador; private Double VCapitalSoc; private String BActivo; private Integer BAlerta;
--	PersonW.java	Clase en donde se declaran las variables que se utilizan en los proceso de seguridad del contribuyente. Algunas de las variables declaradas son las siguientes:

Nombre de Opción	Clase POJO	Descripción

		public class PersonW implements Serializable{
		private static final long serialVersionUID = -4454216593897L;
		private Boolean authorities;
		private String nit;
		private String displayName;
		private String cadmTrib;
		private String rol; //uniqueMember
		private String ssello;
		private Long itecnico;
		private String cargo;
		private String tipoUsuario;
		private String ubicacionFisica;
		private String unidadRecep;
		private String rawPassword;
		private String detFtpHost;
		private String detFtpUsr;
		private String detFtpPass;

Reserva de Cita	ReservaCitaPojo.java	Clase en la cual se declaran las siguientes variables que se utilizaran para llenar los datos en el proceso de Reserva de Cita: private String codigoVerificacion; private String areaServicio; private String tramite; private String unidad; private String fecha; private String hora; private String nit; private String correo; private String telefono; private String estado; private Long idTiquete; private Long idReservaCita; private Long idTramite;
Reasignación de Tiquete	ReasignacionTiquetePojo.java	Clase que se utiliza para llenar los datos en el proceso de reasignación de tiquete. La variables declaradas en esta clase son las siguientes:

Nombre de Opción	Clase POJO	Descripción
		private String tiqueteNo; private String nit; private String servicio; private String tramite; private String estado; private String fecha; private String hora; private Long idTiquete; private Long idTramite; private Long idServicio; private Long prioridad;
Reimpresión de tiquete	ReasignacionTiquetePojo.java	Clase que se utiliza para llenar los datos en el proceso de reasignación y reimpresion de tiquete. La variables declaradas en esta clase son las siguientes: private String tiqueteNo; private String nit; private String servicio; private String tramite; private String estado; private String fecha; private String hora; private Long idTiquete; private Long idTramite; private Long idServicio; private Long prioridad;

Tabla 4.2 Clases Pojo.

5. Repositorios

En la siguiente tabla se proporcionan los repositorios utilizados en las opciones del módulo Mesa de Entrada, la ubicación de estos es en el siguiente paquete: **sv.gob.mh.dgii.colas.repositories**

Opción	Nombre del Repositorio
--	GcConfTramiteRepository.java
	GcPrioridadRepository.java
	GcTiqueteRepository.java
	GcUnidadRecepRepository.java
	GcUsuarioRepository.java
	RcRucRepository.java
	TbListasValorRepository.java
--	PersonWRepository.java
Reserva de Cita	GcReservaCitaRepository.java
	GcTiqueteRepository.java
	GcTramiteRepository.java
	GcUnidadRecepRepository.java
	GcUsuarioRepository.java
	TbListasValorRepository.java
Reasignación de Tiquete	GcTiqueteRepository.java
	GcUnidadRecepRepository.java
	TbListasValorRepository.java
Escalamiento	GcConfTramiteRepository.java
	GcPrioridadRepository.java
	GcServiciosRepository.java
	GcTiqueteRepository.java
	GcTramiteRepository.java
	GcUnidadRecepRepository.java
	GcUsuarioRepository.java
	PersonWRepository.java
	RcRucRepository.java
	TbListasValorRepository.java
Reimpresión de tiquete	GcTiqueteRepository.java
	GcUnidadRecepRepository.java
	TbListasValorRepository.java

Tabla 5.1 Repositorios.

Descripción de Repositorios.

En la siguiente tabla se describen los métodos usados de los repositorios dentro del módulo de Mesa de Entrada.

Opción	Nombre del Repositorio	Descripción de métodos
--	GcConfTramiteRepository.java	**public List<GcConfTramite> listaTramitesByCS(List<String> unidadRecep):** Método que se ejecuta la siguiente hibernate query: "SELECT c FROM GcConfTramite c where C_UNIDAD_RECEP in(?1) and (c.nAtencionProm > 0 or c.nTiempoEspera is not null or c.nTiempoHolgura is not null) and c.nTramiteId.bActiva = 1 and c.nTramiteId.bEscalamiento = 0 ORDER BY nTramiteId.nServiciosId.nServiciosId ASC ".
	GcPrioridadRepository.java	**public Long getPrioridadDuplicada(String nombre,Long id):** Método que ejecuta la siguiente consulta de Hibernate: "SELECT count(p) FROM GcPrioridad p WHERE p.bActiva in(1,0) AND REPLACE(upper(p.sNombre),' ','') = REPLACE(upper(?1) ,' ','') and p.nPrioridadId != ?2".
	GcTiqueteRepository.java	**public GcConfTiquete getimg(Long i):** Método que ejecuta la siguiente query de hibernate: "SELECT c FROM GcConfTiquete c where c.bActiva=1 and c.cUnidadRecep=(select d.cUnidadRecep.cunidadRecep from GcTiquete d where d.nTiqueteId=?1)" **public String leerTiquete(String i,String u):** Método el cual ejecuta la siguiente query nativa: "SELECT GENERATIQUETE(?1,?2) FROM dual". **public String getMaxCorrelativo(Long idTramite, String unidadRecep):** Método que ejecuta la siguiente query de hibernate:

		“SELECT NVL(MAX(TO_NUMBER(REGEXP_REPLACE(t.sCorrelativo,'[^[:digit:]]'))),0)+1 FROM GcTiquete t WHERE REGEXP_REPLACE(t.sCorrelativo,'[^[:alpha:]]') = (SELECT z.sNombre FROM GcUsuario u INNER JOIN u.nEscritorioId e INNER JOIN e.nZonaId z INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1) AND TRUNC(t.fhLlegada) = TRUNC(sysdate)”. **public Timestamp getServerDateTime():** Método que ejecuta la siguiente hibernate query: "SELECT SYSDATE FROM DUAL" **public void changeStatusTiquete(@Param("cUsuarioAtendio") String cUsuarioAtendio, @Param("tiqueteId") Long tiqueteId):** Método que ejecuta la siguiente consulta de hibernate: "UPDATE GcTiquete t SET t.mEstado = 4, t.fhfProceso = SYSDATE, t.cUsuarioAtendio = :cUsuarioAtendio WHERE t.nTiqueteId = :tiqueteId ". **public String getfhDateReasignacion(Long idTramite, String unidadRecep):** Método que se encarga de ejecuta la siguiente hibernate query: “SELECT CASE WHEN ctr.nComportamiento = 0 THEN TO_CHAR(sysdate,'dd/MM/yyyy hh12:mi:ss') WHEN ctr.nComportamiento = 1 THEN TO_CHAR(to_date(min(to_char(tiq.fhLlegada,'dd/MM/yyyy hh12:mm:ss')),'dd/MM/yyyy hh12:mm:ss')+10/(24*60),'dd/MM/yyyy hh12:mm:ss') WHEN ctr.nComportamiento = 2 THEN TO_CHAR(TO_DATE(TRUNC(AVG(TO_CHAR(tiq.fhLlegada,'J'))),'J'),'dd/MM/yyyy hh12:mm:ss') END AS fh_fecha FROM GcTiquete tiq INNER JOIN tiq.nTramiteId tr INNER JOIN tr.nServiciosId s LEFT JOIN tr.gcConfTramiteList ctr WHERE tiq.nTramiteId.nTramiteId = ?1 AND ctr.cUnidadRecep.cunidadRecep = ?2 AND tiq.mEstado = 1 AND TRUNC(tiq.fhLlegada) = TRUNC(sysdate) GROUP BY ctr.nComportamiento ".

		public Integer getTiempoHolgura(Long idTramite, String unidadRecepcion): Método utilizando para ejecutar la siguiente consulta de hibernate hql: "SELECT NVL(ct.nTiempoHolgura,0) FROM GcConfTramite ct WHERE ct.nTramiteId.nTramiteId = ?1 and ct.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1 ". **public int verifyIfExistsTramiteAsignado(Long idTramite, String unidadRecepcion):** Método que se ejecuta la siguiente sentencia de hibernate: "SELECT COUNT(*) FROM GcUsuario u INNER JOIN u.gcTramiteList ut INNER JOIN u.nEscritorioId e INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 and e.cUnidadRecep.cunidadRecep = ?2 and rownum = 1 "
	GcUnidadRecepRepository.java	**public TbUnidadRecep getUnidadEnServicioE(String cunidadRecep):** Método que ejecuta la siguiente hibernate query: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('C','M') and t.cunidadRecep = ?1 ". **public String getCsCombinacion(String unidadRecep):** Método que ejecuta la siguiente sentencia de tipo nativa: "SELECT SIIT.PKG_COLAS_UTILS.CENTRO_SERVICIO(?1) FROM DUAL".
	GcUsuarioRepository.java	**public String getZonaIdentifier(Long tramiteId, String cUnidadRecep):** Método encargado de ejecutar la siguiente sentencia: “SELECT z.sNombre FROM GcZona z INNER JOIN z.gcEscritorioList e INNER JOIN e.gcUsuarioList u INNER JOIN u.gcTramiteList t WHERE t.nTramiteId =?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1 ".

	RcRucRepository.java	**public RcRuc findByNit(String nit):** Método que ejecuta la siguiente sentencia query de hibernate: "SELECT ruc FROM RcRuc ruc LEFT JOIN FETCH ruc.tbTipoContrib WHERE ruc.nit = ?1 ".
--	PersonWRepository.java	**public PersonW findByPrimaryKey(String name):** Este método su implementación se encuentra en el repositorio PersonWRepositoryImpl.java. Se encarga de verificar el usuario que se está ingresando si se encuentra registrado.
Reserva de Cita	GcReservaCitaRepository.java	**public List<GcReservaCita> getData(List<String> unidadRecep):** Método que ejecuta la siguiente sentencia: "SELECT rs FROM GcReservaCita rs WHERE TRUNC(SYSDATE) <= TRUNC(rs.fhReservacion) AND rs.cUnidadRecep.cunidadRecep IN (?1) AND rs.bEstado IN (5,6) ORDER BY TO_NUMBER(REGEXP_REPLACE(rs.sCodVerifica, '[^[:digit:]]')) DESC" **public Integer verificarReserva(Long nReservaCitaID):** Método que ejecuta la siguiente consulta hibernate: "SELECT count(*) FROM GcTiquete t WHERE t.nReservaCitaId.nReservaCitaId = ?1 ". **public Integer actualizarEstado(@Param("bEstado") Integer bEstado, @Param("reservaId") Long reservaId):** Método que ejecuta la siguiente consulta de hibernate: "UPDATE GcReservaCita r SET r.bEstado = :bEstado WHERE r.nReservaCitaId = :reservaId ". **public GcReservaCita getReservaCitaById(Long nReservaCitaID):** Método que ejecuta la siguiente query de hibernate: "SELECT rs FROM GcReservaCita rs "

		WHERE rs.nReservaCitaId = ?1) "
	GcTiqueteRepository.java	**public int verifyIfExistsTramiteAsignado(Long idTramite, String unidadRecepcion):** Método que se ejecuta la siguiente sentencia de hibernate: "SELECT COUNT(*) FROM GcUsuario u INNER JOIN u.gcTramiteList ut INNER JOIN u.nEscritorioId e INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 and e.cUnidadRecep.cunidadRecep = ?2 and rownum = 1 " **public String getMaxCorrelativo(Long idTramite, String unidadRecep):** Método que ejecuta la siguiente query de hibernate: “SELECT NVL(MAX(TO_NUMBER(REGEXP_REPLACE(t.sCorrelativo,'[^[:digit:]]'))),0)+1 FROM GcTiquete t WHERE REGEXP_REPLACE(t.sCorrelativo,'[^[:alpha:]]') = (SELECT z.sNombre FROM GcUsuario u INNER JOIN u.nEscritorioId e INNER JOIN e.nZonaId z INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1) AND TRUNC(t.fhLlegada) = TRUNC(sysdate)”.
	GcTramiteRepository.java	**public GcTramite getTramiteById(Long nTramiteId):** Método que ejecuta la siguiente consulta de hibernate: SELECT t FROM GcTramite t where t.nTramiteId = ?1.
	GcUnidadRecepRepository.java	**public TbUnidadRecep getUnidadEnServicioE(String cunidadRecep):** Método que ejecuta la siguiente query de hibernate: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('C','M') and t.cunidadRecep = ?1 ". **public String getCsCombinacion(String unidadRecep):** Método cuya función es la ejecución de la siguiente sentencia de hibernate, la cual es una query nativa: "SELECT SIIT.PKG_COLAS_UTILS.CENTRO_SERVICIO(?1) FROM DUAL".

	GcUsuarioRepository.java	**public String getZonaIdentifier(Long tramiteId, String cUnidadRecep):** Método cuya función es ejecutar la siguiente sentencia de hibernate: "SELECT z.sNombre FROM GcZona z INNER JOIN z.gcEscritorioList e INNER JOIN e.gcUsuarioList u" INNER JOIN u.gcTramiteList t WHERE t.nTramiteId =?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1 "
Reasignación de Tiquete	GcTiqueteRepository.java	**public List<GcTiquete> getListTiquetes(List<String> unidadRecep):** Método que ejecuta la siguiente query de hibernate: "SELECT t FROM GcTiquete t INNER JOIN t.nTramiteId tr INNER JOIN tr.nServiciosId s WHERE t.nTiqueteId = (SELECT MAX(b.nTiqueteId) FROM GcTiquete b WHERE b.sCorrelativo = t.sCorrelativo) AND TRUNC(t.fhLlegada) = TRUNC(sysdate) AND t.mEstado IN(1,4) AND t.cUnidadRecep.cunidadRecep IN (?1) ORDER BY TO_NUMBER(REGEXP_REPLACE(t.sCorrelativo, '[^[:digit:]]')) DESC "
	GcUnidadRecepRepository.java	**public String getCsCombinacion(String unidadRecep):** Método cuya función es la ejecución de la siguiente sentencia de hibernate, la cual es una query nativa: "SELECT SIIT.PKG_COLAS_UTILS.CENTRO_SERVICIO(?1) FROM DUAL".
Escalamiento	GcConfTramiteRepository.java	**public List<GcConfTramite> listaTramitesEscByCS(List<String> unidadRecep, Long idServicio):** Método en el cual se invoca y se ejecuta la siguiente query de hibernate: "SELECT c FROM GcConfTramite c where C_UNIDAD_RECEP in(?1) and (c.nAtencionProm > 0 or c.nTiempoEspera is not null or c.nTiempoHolgura is not null) and c.nTramiteId.bActiva = 1 and c.nTramiteId.bEscalamiento = 1 AND

		c.nTramiteId.nServiciosId.nServiciosId = ?2 ORDER BY nTramiteId.nServiciosId.nServiciosId ASC "
	GcServiciosRepository.java	**public List<GcServicios> getAllServicios**(): Método que ejecuta la siguiente query de hibernate: "SELECT s FROM GcServicios s ORDER BY nServiciosId ASC ".
	GcTiqueteRepository.java	**public int verifyIfExistsTramiteAsignado(Long idTramite, String unidadRecepcion):** Método que se ejecuta la siguiente sentencia de hibernate: "SELECT COUNT(*) FROM GcUsuario u INNER JOIN u.gcTramiteList ut INNER JOIN u.nEscritorioId e INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 and e.cUnidadRecep.cunidadRecep = ?2 and rownum = 1 " **public String getMaxCorrelativo(Long idTramite, String unidadRecep):** Método que ejecuta la siguiente query de hibernate: “SELECT NVL(MAX(TO_NUMBER(REGEXP_REPLACE(t.sCorrelativo,'[^[:digit:]]'))),0)+1 FROM GcTiquete t WHERE REGEXP_REPLACE(t.sCorrelativo,'[^[:alpha:]]') = (SELECT z.sNombre FROM GcUsuario u INNER JOIN u.nEscritorioId e INNER JOIN e.nZonaId z INNER JOIN u.gcTramiteList tr WHERE tr.nTramiteId = ?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1) AND TRUNC(t.fhLlegada) = TRUNC(sysdate)”.
	GcUnidadRecepRepository.java	**public TbUnidadRecep getUnidadEnServicioE(String cunidadRecep):** Método que ejecuta la siguiente hibernate query: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('C','M') and t.cunidadRecep = ?1 ". **public String getCsCombinacion(String unidadRecep):** Método que ejecuta la siguiente sentencia de tipo nativa: "SELECT SIIT.PKG_COLAS_UTILS.CENTRO_SERVICIO(?1) FROM DUAL".

	GcUsuarioRepository.java	**public String getZonaIdentifier(Long tramiteId, String cUnidadRecep):** Método cuya función es ejecutar la siguiente sentencia de hibernate: "SELECT z.sNombre FROM GcZona z INNER JOIN z.gcEscritorioList e INNER JOIN e.gcUsuarioList u" INNER JOIN u.gcTramiteList t WHERE t.nTramiteId =?1 AND z.cUnidadRecep.cunidadRecep = ?2 AND ROWNUM = 1 ".
Reimpresión de tiquete	GcTiqueteRepository.java	**public List<GcTiquete> getListTiquetesEnEspera(List<String> unidadRecep):** Método que ejecuta la siguiente consulta de hibernate: "SELECT t FROM GcTiquete t INNER JOIN t.nTramiteId tr INNER JOIN tr.nServiciosId s WHERE t.mEstado = 1 AND TRUNC(t.fhLlegada) = TRUNC(sysdate) AND t.cUnidadRecep.cunidadRecep IN (?1) ORDER BY TO_NUMBER(REGEXP_REPLACE(t.sCorrelativo, '[^[:digit:]]')) DESC "
	GcUnidadRecepRepository.java	**public String getCsCombinacion(String unidadRecep):** Método que ejecuta la siguiente sentencia de tipo nativa: "SELECT SIIT.PKG_COLAS_UTILS.CENTRO_SERVICIO(?1) FROM DUAL".

Tabla 5.2 Descripción de Repositorios.

6. Componentes

La siguiente tabla proporciona la lista de componentes en este módulo.

Nombre del Componente	Ubicación
AppInfoInterceptor.java	sv.gob.mh.dgii.colas.components

TABLA 6.1 Listado y Ubicación de componentes

Descripción de Componentes.

A continuación se describen la funcionalidad del componente utilizado en el sistema SAC (Sistema de Atención al Contribuyente).

Nombre del Componente	Descripcion
AppInfoInterceptor.java	Clase que extiende de HandlerInterceptorAdapter la cual sobre-escribe los siguientes Métodos: **public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler);** Esta aplicación siempre devuelve cierto. **public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView);** Esta aplicación está vacía.

TABLA 6.2 Nombre y descripción de componentes

7. Configuración.

En la siguiente tabla se describen las clases usadas para la configuración del proyecto. Las clases se encuentran ubicadas en el paquete: **sv.gob.mh.dgii.colas.config**

Nombre del archivo de configuración	Descripción
HibernateConfig	**public LocalSessionFactoryBean alertsSessionFactory();** Método que establece el origen de datos para ser utilizados por la SessionFactory, Especifica el paquete "sv.gob.mh.dgii.model" para buscar la autodetección de sus clases de entidad en la ruta de clases, establece las propiedades de hibernate, establecer la ubicación de un único archivo de configuración de Hibernate XML, por ejemplo, como recurso de ruta de clases "ruta de clases: hibernate.cfg.xml". **public HibernateTransactionManager transactionManager();** Método que ajusta la instancia que debe gestionar las transacciones y obtiene el objeto del método alertsSessionFactory(). **public HibernateExceptionTranslator exceptionTranslation();** Método que crea una nueva instancia de **HibernateExceptionTranslator.** **final Properties hibernateProperties();** Método que establece las propiedades de la clase Properties

Nombre del archivo de configuración	Descripción
RepositoryConfig	**@EnableJpaRepositories(basePackages = { "sv.gob.mh.dgii" }, includeFilters = @ComponentScan.Filter(pattern = ".*.repositories.*", type = FilterType.REGEX))** Esta anotación me dice que paquete inyectara como repositorios. **@ComponentScan(basePackages = "sv.gob.mh.dgii", useDefaultFilters = false, includeFilters = @Filter(pattern = ".*.components.*", type = FilterType.REGEX));** Configura directivas de escaneo de componentes y me dice que paquete inyectara como repositorios.
SecurityConfig	**protected void configure(HttpSecurity http);** Método que contiene información sobre cómo autenticar a los usuarios, Asegura que cualquier petición a nuestra aplicación requiere que el usuario sea autenticado, permite que los usuarios se autentican con formulario basado entrada, permite que los usuarios se autentican con autenticación básica HTTP. **public DefaultLdapAuthoritiesPopulator ldapAuthoritiesPopulator();** Constructor de escenarios de búsqueda de grupo y suministra los contextos utilizados para buscar roles de usuario. **public DgiiFilterInvocationSecurityMetadataSource dgiiSecurityMetadataSource();** Método que invoca el paquete "sv.gob.mh.dgii.colas.security.PropertyFileSecurityBuilder". **public AuthenticationManager authenticationManagerBean();** Método de anulación authenticationManagerBean en WebSecurityConfigurerAdapter para exponer el

Nombre del archivo de configuración	Descripción
	AuthenticationManager construido usando configure(AuthenticationManagerBuilder) **public AffirmativeBased accessDecisionManager();** Método que concreta de **AccessDecisionManager** que otorga acceso si cualquier AccessDecisionVoter devuelve una respuesta afirmativa. **public RoleVoter roleVoter();** Método que especifica un prefijo de rol al usuario. **public AuthenticatedVoter authenticatedVoter();** Método para la autenticación de usuario dependiendo si es anónima o si desea que se recuerde. **public FilterSecurityInterceptor dgiiFilterSecurityInterceptor();** Método que realiza el manejo de la seguridad de los recursos HTTP a través de un filtro de aplicación. **BaseLdapPathContextSource contextSource();** Interfaz para ser implementado por ContextSources que son capaces de proporcionar la ruta LDAP base. **public void configureAuthentification(AuthenticationManager Builder auth)** Método que obtiene la configuración requerirá que cualquier URL que se solicita será necesario un usuario con el rol de "ROLE_USER".
SecurityWebApplicationInitializer	Clase que extiende de AbstractSecurityWebApplicationInitializer la cual sobre-escribe los siguientes Métodos: **public class SecurityWebApplicationInitializer extends AbstractSecurityWebApplicationInitializer.**

Nombre del archivo de configuración	Descripción
WebConfig	Clase que extiende de WebMvcConfigurerAdapter la cual sobre-escribe los siguientes Métodos: **public void configureMessageConverters(List<HttpMessageConverter<?>> converters)** Para personalizar la configuración importado, implementar la interfaz WebMvcConfigurer o más probablemente extender el método vacío clase base WebMvcConfigurerAdapter y anular métodos individuales **public String appName();** Método que obtiene el nombre completo de la app. **public void addResourceHandlers(ResourceHandlerRegistry registry) ;** Agregar controladores para servir recursos estáticos como imágenes, js y css, archivos desde ubicaciones específicas bajo raíz de la aplicación web, la ruta de clase, y otros. **public LocaleResolver localeResolver();** Interfaz de estrategias de resolución de localización basadas en web que permite tanto la resolución de la configuración regional a través de la solicitud y la modificación de la configuración regional a través de la solicitud y la respuesta. **public ViewResolver viewResolver() ;** Método para ver si el estado no cambia durante el funcionamiento de la aplicación. **public CommonsMultipartResolver multipartResolver():**

Nombre del archivo de configuración	Descripción
	Método encargado de setear por defecto la configuración utf-8 y de ¡configurar los tamaños máximos de upload y el uso de la memoria máxima.
WebSocketConfig	**public void configureMessageBroker(MessageBrokerRegistry config):** Método encargado de configurar enableSimpleBroker y setApplicationDestinationPrefixes. **public void registerStompEndpoints(StompEndpointRegistry registry):** Método encargado de configurar el sock en addEndpoint withSockJS.

TABLA 7.1 Configuraciones del Módulo de Colas

8. Documentos Relacionados a la Base de Datos

Tablas

En las siguientes páginas se describen cada uno de los campos de las tablas de la base de datos. **La descripción del campo está sujeta a los comentarios agregados en la base de datos.**

GC_CONF_TRAMITE Secuencia: SEQ_GC_CONF_TRAMITE Nombre del Atributo	Tipo de Dato	Null	Descripción
N_CONF_TRA_ID	NUMBER	No	Llave primaria de la tabla
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
N_TRAMITE_ID	NUMBER	No	Referencia al trámite a que se refiere la configuración
N_ATENCION_PROM	NUMBER	No	Tiempo promedio de atención
N_TIEMPO_ESPERA	NUMBER	Yes	Tiempo, en minutos, de espera máximo que debe tardarse en la cola
N_TIEMPO_HOLGURA	NUMBER	Yes	Tiempo, en minutos, que debe esperarse antes de incluirlo en la cola
C_USUARIO_CREA	VARCHAR2(100 BYTE)	No	Código del usuario que crea el registro
C_USUARIO_MODI	VARCHAR2(100 BYTE)	No	Código del usuario que modifica el registro
FI_VIGENCIA	DATE	No	Fecha en que el registro es creado
FF_VIGENCIA	DATE	Yes	Fecha en que el registro deja de tener vigencia
F_MODIFICA	DATE	Yes	Fecha en que el registro fue modificado
N_COMPORTAMIENTO	NUMBER(6,0)	Yes	Comportamiento: 0-al final, 1-al inicio, 2- en medio, 3 por peso
N_PESO	NUMBER(6,0)	Yes	peso de la reasignación
N_PROM_ESPERA	NUMBER	Yes	Promedio real (en segundos) de espera en cola

GC_CONF_TRAMITE Secuencia: SEQ_GC_CONF_TRAMITE Nombre del Atributo	Tipo de Dato	Null	Descripción
N_PROM_ATENCION	NUMBER	Yes	Promedio real (en segundos) de atención en escritorio

GC_SERVICIO Secuencia: SEQ_GC_SERVICIOS Nombre del Atributo	Tipo de Dato	Null	Descripción
N_SERVICIOS_ID	NUMBER(38,0)	No	Llave primaria de la tabla
D_SERVICIOS	VARCHAR2(256 BYTE)	Yes	Descripción del registro
C_USUARIO_CREA	VARCHAR2(100 BYTE)	No	Código del usuario que crea el registro
C_USUARIO_MODI	VARCHAR2(100 BYTE)	No	Código del usuario que modifica el registro
FI_VIGENCIA	DATE	No	Fecha en que el registro es creado
FF_VIGENCIA	DATE	Yes	Fecha en que el registro deja de tener vigencia
F_MODIFICA	DATE	Yes	Fecha en que el registro fue modificado
N_ORDEN	NUMBER	Yes	Orden en que se presentaran los servicios en pantalla
B_ACTIVA	NUMBER(1,0)	No	Bandera que indica si el registro está activo o no
S_NOMBRE	VARCHAR2(256 BYTE)	Yes	Nombre del centro de servicio

GC_TIQUETE Secuencia: SEQ_GC_TIQUETE Nombre del Atributo	Tipo de Dato	Null	Descripción
N_TIQUETE_ID	NUMBER	No	Llave primaria de la tabla
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio

GC_TIQUETE Secuencia: SEQ_GC_TIQUETE Nombre del Atributo	Tipo de Dato	Null	Descripción
N_RESERVA_CITA_ID	NUMBER	Yes	Referencia a la reserva de cita que dio origen al tiquete
N_PRIORIDAD_ID	NUMBER	No	Prioridad del tiquete
N_TRAMITE_ID	NUMBER	No	Referencia al trámite que está atendiendo el tiquete
N_TIQUETE_REA	NUMBER	Yes	Tiquete de donde fue reasignado al tiquete actual
S_CORRELATIVO	VARCHAR2(25 BYTE)	No	Número del tiquete
NIT	VARCHAR2(14 BYTE)	Yes	Nit del contribuyente
M_ESTADO	VARCHAR2(1 BYTE)	No	Estado en que se encuentra el tiquete
FH_LLEGADA	DATE	Yes	Fecha y hora en que el contribuyente llegó a la cola
FH_LLAMADO	DATE	Yes	Fecha y hora en que el contribuyente fue llamado
FHI_PROCESO	DATE	Yes	Fecha y hora en que la atención inició
FHF_PROCESO	DATE	Yes	Fecha y hora en que la atención finalizó
C_USUARIO_ATENDIO	VARCHAR2(256 BYTE)	Yes	Usuario que atendió el trámite
C_USUARIO_CREA	VARCHAR2(256 BYTE)	Yes	Usuario que crea el registro
N_TIEMPO_HOLGURA	NUMBER	Yes	Tiempo de holgura del trámite seleccionado

GC_CONF_TIQUETE Secuencia: SEQ_GC_CONF_TIQUETE Nombre del Atributo	Tipo de Dato	Null	Descripción
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
B_ACTIVA	NUMBER(1,0)	No	Bandera que indica si el registro está activo o no
X_CONTENIDO	CLOB	No	Formato del tiquete que se entregaran en los centros de servicio
C_USUARIO_CREA	VARCHAR2(100 BYTE)	No	Código del usuario que crea el registro

GC_CONF_TIQUETE Secuencia: SEQ_GC_CONF_TIQUETE Nombre del Atributo	Tipo de Dato	Null	Descripción
C_USUARIO_MODI	VARCHAR2(100 BYTE)	No	Código del usuario que modifica el registro
FF_VIGENCIA	DATE	Yes	Fecha en que el registro deja de tener vigencia
FI_VIGENCIA	DATE	No	Fecha en que el registro es creado
F_MODIFICA	DATE	Yes	Fecha en que el registro fue modificado
N_CONF_TIQ_ID	NUMBER	No	Llave primaria de la tabla
N_VERSION	NUMBER	Yes	Versión del tiquete en relación al centro de servicio
X_IMAGEN	BLOB	Yes	Imagen que va en el tiquete.

GC_PRIORIDAD Secuencia: SEQ_GC_PRIORIDAD Nombre del Atributo	Tipo de Dato	Null	Descripción
N_PRIORIDAD_ID	NUMBER	No	Llave primaria de la tabla
N_PESO	NUMBER	No	Cuantificación de la prioridad
D_PRIORIDAD	VARCHAR2(256 BYTE)	Yes	Descripción del registro
C_USUARIO_CREA	VARCHAR2(100 BYTE)	No	Código del usuario que crea el registro
C_USUARIO_MODI	VARCHAR2(100 BYTE)	No	Código del usuario que modifica el registro
FI_VIGENCIA	DATE	No	Fecha en que el registro es creado
FF_VIGENCIA	DATE	Yes	Fecha en que el registro deja de tener vigencia
F_MODIFICA	DATE	Yes	Fecha en que el registro fue modificado
B_ACTIVA	NUMBER(1,0)	No	Bandera que indica si el registro está activo o no
B_FILA_ESP	NUMBER(1,0)	No	Si será asignado a la fila especial

GC_PRIORIDAD Secuencia: SEQ_GC_PRIORIDAD Nombre del Atributo	Tipo de Dato	Null	Descripción
S_NOMBRE	VARCHAR2(256 BYTE)	Yes	Nombre de la prioridad

GC_RESERVA_CITA Secuencia: Nombre del Atributo	Tipo de Dato	Null	Descripción
N_RESERVA_CITA_ID	NUMBER	No	Llave primaria de la tabla
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
N_TRAMITE_ID	NUMBER	No	Referencia al trámite que está solicitando la reserva de cita
FH_RESERVACION	DATE	No	Fecha y hora en que se llevará a cabo la cita programada
S_CORREO	VARCHAR2(256 BYTE)	Yes	Correo electrónico del contribuyente que hace la cita
NIT	VARCHAR2(14 BYTE)	Yes	Nit del contribuyente
N_TELEFONO	VARCHAR2(25 BYTE)	Yes	Teléfono de contacto del contribuyente
S_OBSERVACIONES	VARCHAR2(1024 BYTE)	Yes	Observaciones de la reserva de cita.
S_COD_VERIFICA	VARCHAR2(50 BYTE)	Yes	Código de verificación de la cita
N_TIPO_RESERVA	NUMBER	Yes	Forma en que fue hecha la reserva de cita
NIT_TERCERO	VARCHAR2(14 BYTE)	Yes	Nit de la persona que hizo la cita en nombre de un tercero
B_ESTADO	NUMBER(1,0)	No	Estado en que se encuentra la reserva de cita

TB_UNIDAD_RECEP Secuencia: NO Nombre del Atributo	Tipo de Dato	Null	Descripción
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
C_USUARIO	VARCHAR2(30 BYTE)	No	Usuario
D_UNIDAD_RECEP	VARCHAR2(40 BYTE)	No	Descripción
FH_INGRESO	DATE	Yes	Fecha de ingreso
B_STATUS	NUMBER(1,0)	No	Estado

TB_UNIDAD_RECEP Secuencia: NO Nombre del Atributo	Tipo de Dato	Null	Descripción
M_TIPO_UNIDAD	VARCHAR2(1 BYTE)	No	Tipo de unidad
B_DESPLEGABLE	NUMBER(1,0)	No	Se muestra 1 o 0
C_UNIDAD	VARCHAR2(1 BYTE)	Yes	Unidad asignada
C_UNIDAD_DGT	VARCHAR2(25 BYTE)	Yes	Unidad DGT
C_DEP_MUN	VARCHAR2(4 BYTE)	Yes	Código de departamento
S_UBIC_GEOGRAF	VARCHAR2(60 BYTE)	Yes	Descripción de la ubicación geográfica.
C_UNIDAD_RECEP_SUP	VARCHAR2(5 BYTE)	Yes	
M_SERVICIO	VARCHAR2(1 BYTE)	Yes	

RC_RUC Secuencia: Nombre del Atributo	Tipo de Dato	Null	Descripción
NIT	VARCHAR2(14 BYTE)	No	número de identificación tributaria
C_PAIS	VARCHAR2(4 BYTE)	No	código de país
C_DEPARTAMENTO	VARCHAR2(2 BYTE)	No	código de departamento
C_MUNICIPIO	VARCHAR2(2 BYTE)	No	código de municipio
C_ADM_TRIB	VARCHAR2(2 BYTE)	No	código de administración tributaria
C_TERMINO	VARCHAR2(3 BYTE)	No	código de termino
C_IMPORTANCIA	VARCHAR2(1 BYTE)	No	código de importancia
C_CLASE	VARCHAR2(1 BYTE)	No	código de clase de contribuyente
C_TIPO	VARCHAR2(4 BYTE)	No	código de tipo
S_1APE_RASOC	VARCHAR2(100 BYTE)	No	primer apellido o razón social de la empresa
S_2APE_ABREV	VARCHAR2(100 BYTE)	Yes	segundo apellido
S_NOMBRES	VARCHAR2(40 BYTE)	Yes	nombres del contribuyente
F_NAC_CONST	DATE	No	fecha de nacimiento o constitución
B_DOMICILIADO	NUMBER(1,0)	No	es domiciliado o no
F_TERMINO	DATE	Yes	fecha de término del NIT
B_INTERES_FISCAL	NUMBER(1,0)	No	tiene interés fiscal
M_SEXO	VARCHAR2(1 BYTE)	Yes	tipo de sexo
B_IMPORTADOR	NUMBER(1,0)	No	es importador
V_CAPITAL_SOC	NUMBER(13,2)	Yes	valor de capital social
B_ACTIVO	VARCHAR2(1 BYTE)	No	esta activo
B_ALERTA	NUMBER(1,0)	Yes	

9. Seguridad

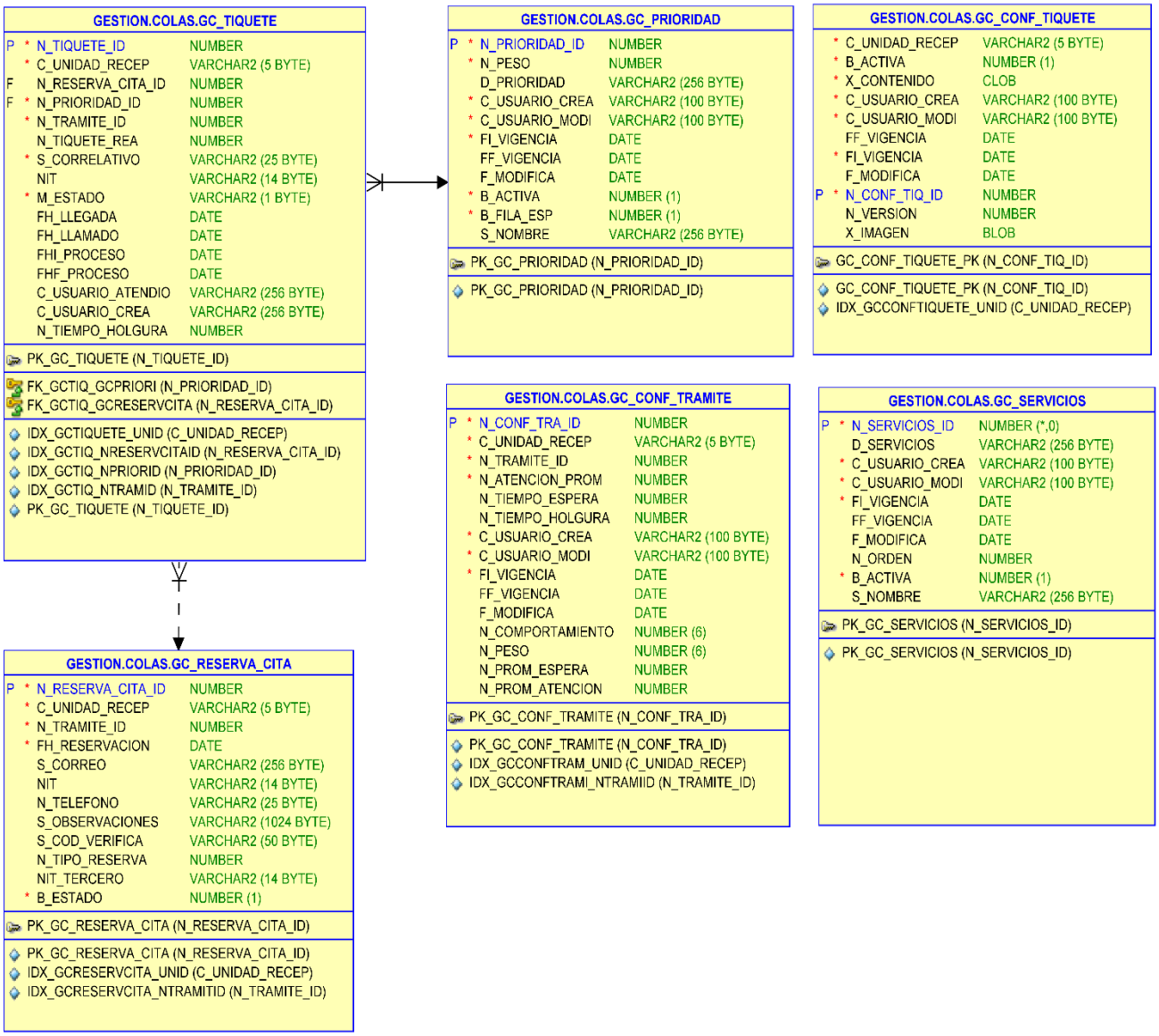
Estos son los roles de seguridad usados en el Módulo Mesa de Entrada:

Opción	URL	Roles
--	/me/home/	ROLE_GC_R1 ROLE_GC_R4
Confirmación de Cita	/me/reservaCita	ROLE_GC_R1 ROLE_GC_R5 ROLE_GC_R6
Reasignación de Tiquete	/me/reasignacionTiquete	ROLE_GC_R1 ROLE_GC_R5 ROLE_GC_R6
Escalamiento	/me/escalamientoTiquete	ROLE_GC_R1 ROLE_GC_R5 ROLE_GC_R6
Reimpresión de tiquete	/me/reimpresionTiquete	ROLE_GC_R1 ROLE_GC_R5 ROLE_GC_R6

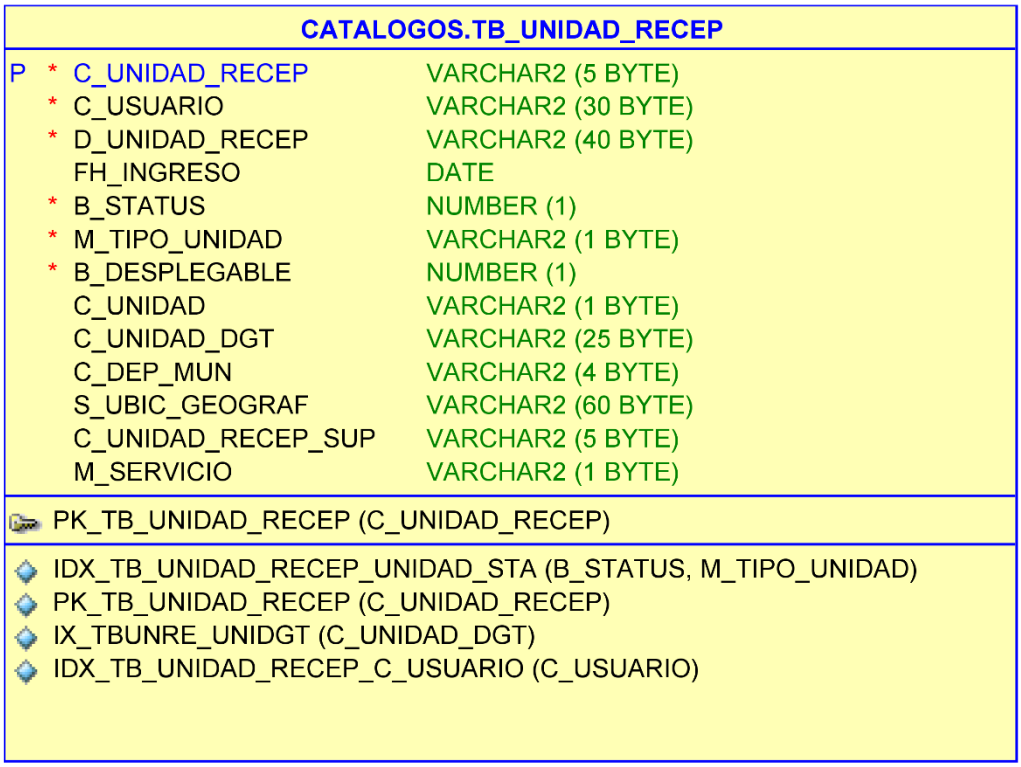
Tabla 9.1 Roles de Seguridad

10. Anexos

Anexo 1:



Anexo 2:


CATALOGOS.TB_UNIDAD_RECEP
P * C_UNIDAD_RECEP VARCHAR2 (5 BYTE)
* C_USUARIO VARCHAR2 (30 BYTE)
* D_UNIDAD_RECEP VARCHAR2 (40 BYTE)
FH_INGRESO DATE
* B_STATUS NUMBER (1)
* M_TIPO_UNIDAD VARCHAR2 (1 BYTE)
* B_DESPLEGABLE NUMBER (1)
C_UNIDAD VARCHAR2 (1 BYTE)
C_UNIDAD_DGT VARCHAR2 (25 BYTE)
C_DEP_MUN VARCHAR2 (4 BYTE)
S_UBIC_GEOGRAF VARCHAR2 (60 BYTE)
C_UNIDAD_RECEP_SUP VARCHAR2 (5 BYTE)
M_SERVICIO VARCHAR2 (1 BYTE)
PK_TB_UNIDAD_RECEP (C_UNIDAD_RECEP)
IDX_TB_UNIDAD_RECEP_UNIDAD_STA (B_STATUS, M_TIPO_UNIDAD)
PK_TB_UNIDAD_RECEP (C_UNIDAD_RECEP)
IX_TBUNRE_UNIDGT (C_UNIDAD_DGT)
IDX_TB_UNIDAD_RECEP_C_USUARIO (C_USUARIO)

Anexo 3:

RUC.RC_RUC

P	*	NIT	VARCHAR2 (14 BYTE)
	*	C_PAIS	VARCHAR2 (4 BYTE)
	*	C_DEPARTAMENTO	VARCHAR2 (2 BYTE)
	*	C_MUNICIPIO	VARCHAR2 (2 BYTE)
	*	C_ADM_TRIB	VARCHAR2 (2 BYTE)
	*	C_TERMINO	VARCHAR2 (3 BYTE)
	*	C_IMPORTANCIA	VARCHAR2 (1 BYTE)
	*	C_CLASE	VARCHAR2 (1 BYTE)
	*	C_TIPO	VARCHAR2 (4 BYTE)
	*	S_1APE_RASOC	VARCHAR2 (100 BYTE)
		S_2APE_ABREV	VARCHAR2 (100 BYTE)
		S_NOMBRES	VARCHAR2 (40 BYTE)
	*	F_NAC_CONST	DATE
	*	B_DOMICILIADO	NUMBER (1)
		F_TERMINO	DATE
	*	B_INTERES_FISCAL	NUMBER (1)
		M_SEXO	VARCHAR2 (1 BYTE)
	*	B_IMPORTADOR	NUMBER (1)
		V_CAPITAL_SOC	NUMBER (13,2)
	*	B_ACTIVO	VARCHAR2 (1 BYTE)
		B_ALERTA	NUMBER (1)

PK_RC_RUC (NIT)

- IDX_RC_RUC_IMP_ADM_TR (C_IMPORTANCIA, C_ADM_TRIB)
- IX_RCRUC_CCLASE (NIT, C_CLASE)
- IDX_RC_RUC_C_DEPARTAMENTO_C_MU (C_DEPARTAMENTO, C_MUNICIPIO)
- IDX_RC_RUC_C_ADM_TRIB (C_ADM_TRIB)
- IDX_RC_RUC_C_TERMINO (C_TERMINO)
- IDX_RC_RUC_C_CLASE_C_TIPO (C_CLASE, C_TIPO)
- IDX_RC_RUC_C_IMPORTANCIA (C_IMPORTANCIA)
- IDX_RC_RUC_C_PAIS (C_PAIS)
- IDX_RC_RUC_F_NAC_CONST (F_NAC_CONST)
- IDX_RC_RUC_S_1APE_RASOC (S_1APE_RASOC)
- IDX_RC_RUC_S_2APE_ABREV (S_2APE_ABREV)
- IDX_RC_RUC_S_NOMBRES (S_NOMBRES)