MANUAL TECNICO MONITOREO

SISTEMA DE ATENCIÓN AL CONTRIBUYENTE (SAC)

Tabla de contenido

1. Breve Descripción del Módulo de Monitoreo

El módulo de monitoreo el cual muestra información estadística por medio de gráficos y tablas en tiempo real de colas, tramites, usuarios y alertas que se requieren para el monitoreo de los diferentes centros de servicios, además permite emitir alertas, avisos y advertencias programadas en caso de desbordamiento, o casos especiales con los usuarios.

2. Diagrama Entidad – Relación

A continuación se listan las diferentes tablas utilizadas en el módulo de Monitoreo, el esquema al que pertenecen y el anexo para visualizar el diagrama.

Tabla	Esquema	Anexo
GC_TRAMITE **GC_SERVICIOS** **GC_ZONA** **GC_ESCRITORIO** **GC_USUARIO**	GESTION.COLAS	Ver anexo 1
TB_UNIDAD_RECEP	CATALOGOS	Ver anexo 2

Tabla 2.1 tablas con sus esquemas respectivos

3. Estructura de Componentes (Capas Vista y Controlador)

3.1 Vistas o páginas HTML

En la siguiente tabla se muestran todas las páginas HTML usadas en el Módulo de Monitoreo y el lugar en donde se pueden encontrar (**Other Sources src/main/sources/**).

Nombre de opción	Ubicación	Nombre
Página índex de Monitoreo	templates.monitoreo	**monitoreos.html**
Centros de Servicios	templates.monitoreo	**centrodeservicio.html**
Secciones	templates.monitoreo	**mosecciones.html**
Trámites	templates.monitoreo	**mTramites.html**
Usuarios	templates.monitoreo	**musuario.html**
Alertas	templates.monitoreo	**malertas.html**

Tabla 3.1 nombres y ubicación de las páginas HTML por opción

3.2 Controladores

En la siguiente tabla se muestran todas las clases controladores que se usan en el Módulo de Monitoreo y la ubicación del paquete (**sv.gob.mh.dgii.colas.controllers.monitoreo**) en donde se encuentran.

Nombre de Opción	Nombre
Página índex de Monitoreo	**monitoreoController.java**
Centros de Servicios	**monitoreoCentrosController.java**
Secciones	**monitoreoSeccController.java**
Trámites	**monitoreoTramiteController.java**
Usuarios	**monitoreousuarioController.java**
Alertas	**monitoreoAlertController.java**

Tabla 3.2 nombres de los controladores por opción

3.3 Descripción de Controladores

En las siguientes tablas se describen brevemente cada uno de los métodos de los diferentes controladores usados en el Módulo de Monitoreo.

PANTALLA: Página índex de Monitoreo	
Controlador:	**monitoreoController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value = "/", method = RequestMethod.GET) **public String indexMonitoreo(HttpServletRequest request,** **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model) {:** Método que redirige a la siguiente url /monitoreo/monitoreos.html; es decir carga la página índex del módulo de Monitoreo, en donde se muestran las diferentes opciones de dicho módulo.	

Tabla 3.3 Controlador monitoreoController

PANTALLA: Centros de Servicios	
Controlador:	**monitoreoCentrosController.java**
Descripción de Métodos de Negocio	
@ModelAttribute("monitoreoTramitePojo") **public MonitoreoTramitePojo monitoreoTramite() {:** Método que crea una nueva instancia del pojo MonitoreoTramitePojo y lo retorna. **public String indexMonitoreo(HttpServletRequest request,** **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model) {:** Método que carga el listado de centros de servicios por medio del repositorio GcUnidadRecepRepository y el llamado de método getUnidadesList y asigna dicho listado al pojo MonitoreoTramitePojo para cargarlos en la session. Carga la página de centros de servicios, por medio del return de la url /monitoreo/centrodeservicio.html. **@RequestMapping(value = "/getall", method = RequestMethod.POST, headers = "Accept=application/json", produces = "application/json")** **@ResponseBody** **public MonitoreoCsPojo getMonitoreoJson(HttpServletRequest request, @RequestParam(value = "unidad", required = false) String unidad){:** Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las zonas que éste tiene por medio del método getZonasByUnidad del repositorio GcZonaRepository. Se recorren las zonas para determinar los usuarios técnicos que éstas tienen, medir el porcentaje de saturación y la cantidad de trámites que están en cola por cada zona. **public List<TbUnidadRecep> getUnidadesList() {:** Método que obtiene el o los centros de servicios por medio del repositorio GcUnidadRecepRepository y lo retorna en forma de lista.	

Tabla 3.4 Controlador monitoreoCentrosController

PANTALLA: Centros de Servicios	
Controlador:	**monitoreoCentrosController.java**
Descripción de Métodos de Negocio	
public String getUnidad() {: Método que obtiene el centro de servicio por medio del método getUnidadRecep, de no encontrarse lo busca por la ubicación física llamando al método getUbicacionFisica y retorna dicho centro de servicio. **@RequestMapping(value = "/getGrafica", method = RequestMethod.GET, headers = "Accept=application/json", produces = "application/json")** **@ResponseBody** **public List getMonitoreoGraficaJson(HttpServletRequest request, @RequestParam(value = "unidad", required = false) String unidad){:** Método que carga los diferentes servicios activos por medio del método getAllServiciosActivo del repositorio GcServiciosRepository, luego recorre el listado de dichos servicios para contar la cantidad de trámites por servicio que se encuentran en un momento dado por zona.	

Tabla 3.4 Controlador monitoreoCentrosController, continuación

PANTALLA: Secciones	
Controlador:	**monitoreoSeccController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value = "/", method = RequestMethod.GET) **public String indexMonitoreo(HttpServletRequest request, @ModelAttribute MonitoreoSeccionPojo monitoreoSeccionPojo, final ModelMap model) {:** Método que carga los diferentes servicios por medio del método getAllServicios del repositorio GcServiciosRepository y carga dicho listado de servicios a la session. Carga la página de Secciones, por medio del return de la url /monitoreo/mosecciones.html. **@RequestMapping(value = "/getall", method = RequestMethod.GET, headers = "Accept=application/json", produces = "application/json")** **@ResponseBody** **public MonitoreoSeccionPojo getMonitoreoJson(HttpServletRequest request, @RequestParam(value = "seccion", required = false) String seccion){:** Método que obtiene la sección seleccionada del combo para asignarle los diferentes valores mostrados en las gráficas y cuadros, los cuales son: nombre de la sección, tiempos de espera (máxima, mínima, promedio, entre otros), los diferentes trámites que dicha sección atiende y los diferentes tiempos de procesamiento de cada trámite, el listado de técnicos que atienden dicha sección. **public String getUnidad() {:** Método que obtiene el centro de servicio por medio del método getUnidadRecep, de no encontrarse lo busca por la ubicación física llamando al método getUbicacionFisica y retorna dicho centro de servicio. **public String reemplazar(String cadena){:** Método que toma todas las vocales con tildes y eñes en un solo String y las reemplaza por sus correspondientes códigos html de tildes y eñes, luego retorna dicho String.	

Tabla 3.5 Controlador monitoreoSeccController

PANTALLA: Trámites	
Controlador:	**monitoreoTramiteController.java**
Descripción de Métodos de Negocio	

@ModelAttribute("monitoreoTramitePojo")
public MonitoreoTramitePojo monitoreoTramite() {:

Método que crea una nueva instancia del pojo MonitoreoTramitePojo y lo retorna.

@RequestMapping(value = "/", method = RequestMethod.GET)
public String indexEscritorio(HttpServletRequest request, @ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model) {:

Método que carga el listado de trámites y centros de servicio, asigna dichos listados al pojo MonitoreoTramitePojo para cargarlos a la session. Carga la página de Trámites, por medio del return de la url /monitoreo/mTramites.html.

@RequestMapping(value = "/getall", method = RequestMethod.GET, headers = "Accept=application/json", produces = "application/json")
@ResponseBody
public MonitoreoTramitePojo getall(HttpServletRequest request, @RequestParam(value = "tramite", required = false) String tramite){:

Método que obtiene el trámite seleccionado del combo para poder asignarle los diferentes valores mostrados en las gráficas y cuadros, los cuales son: tiempos máximos, mínimos, promedios y la moda para los trámites en espera, llamada y en proceso y el listado de técnicos que atienden dicho trámite.

public String getUnidad() {:

Método que obtiene el centro de servicio por medio del método getUnidadRecep, de no encontrarse lo busca por la ubicación física llamando al método getUbicacionFisica y retorna dicho centro de servicio.

Tabla 3.6 Controlador monitoreoTramiteController

PANTALLA: Usuarios	
Controlador:	**monitoreousuarioController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value = "/", method = RequestMethod.GET) public String indexMonitoreo(HttpServletRequest request, @ModelAttribute MonitoreoUsuarioPojo monitoreoUsuarioPojo, final ModelMap model) {: Método que carga la página de Usuarios, por medio del return de la url /monitoreo/musuario.html. **@RequestMapping(value = "/getall", method = RequestMethod.GET, headers = "Accept=application/json", produces = "application/json") @ResponseBody public List<MonitoreoUsuarioPojo> getUsuariosJson(HttpServletRequest request, @RequestParam(value = "unidad", required = false) String unidad){:** Método que carga el listado de técnicos por centro de servicio, recorre dicho listado para asignarles a cada técnico el nombre, escritorio, rol, estado de actividad actual, trámites atendidos y la sección a la que pertenece. **@RequestMapping(value = "/getEscritoriosDisponibles", method = RequestMethod.GET, headers = "Accept=application/json", produces = "application/json") @ResponseBody public List getEscritoriorsDisponiblesJson(HttpServletRequest request, @RequestParam(value = "unidad", required = false) String unidad){:** Método que busca el listado de escritorios que se encuentra disponibles por centro de servicio y retorna dicho listado. **public String getUnidad() {:** Método que obtiene el centro de servicio por medio del método getUnidadRecep, de no encontrarse lo busca por la ubicación física llamando al método getUbicacionFisica y retorna dicho centro de servicio.	

Tabla 3.7 Controlador monitoreousuarioController

PANTALLA: Alertas	
Controlador:	**monitoreoAlertController.java**
Descripción de Métodos de Negocio	
@RequestMapping(value = "/", method = RequestMethod.GET) public String indexMonitoreo(HttpServletRequest request, @ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model) {: Método que carga la página de Alertas, por medio del return de la url /monitoreo/malertas.html. **@RequestMapping(value = "/getAlerta", method = RequestMethod.POST) public @ResponseBody String getAlerta(HttpServletRequest request,@RequestBody Map<String, String> map) {:** Método que carga el listado de alertas por trámites, zonas y por centro de servicio y luego retorna dicho listado.	

Tabla 3.8 Controlador monitoreoAlertController

4. Clases POJOS

4.1 Listado de clases POJOS

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

Nombre de Opción	Ubicación	Nombre
Página índex de Monitoreo	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoTramitePojo.java
Centros de Servicios	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoColasPojo.java MonitoreoCsPojo.java MonitoreoSeccionPojo.java MonitoreoSeccionTramites.java MonitoreoTramitePojo.java MonitoreoUsuarioPojo.java MonitoreoZonasPojo.java
Secciones	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoSeccionPojo.java MonitoreoSeccionTramites.java MonitoreoSeccionUsuarios.java
Trámites	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoSeccionUsuarios.java MonitoreoTramitePojo.java
Usuarios	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoUsuarioPojo.java
Alertas	sv.gob.mh.dgii.colas.pojos.entities	MonitoreoTramitePojo.java

Tabla 4.1 Nombre y ubicación de las clases pojo

4.2 Descripción de clases POJOS

En la siguiente tabla se describen las diferentes clases POJO que se usan en el Módulo de Monitoreo.

Pantalla: Página índex de Monitoreo	
Pojo	Descripción
MonitoreoTramitePojo.java	Esta clase permite definir las variables escalares para capturar información de los trámites y unidades receptoras. Las variables que se encuentran dentro de esta clase son las siguientes: • **private List<GcTramite> tramiteList;** • **private List<TbUnidadRecep> UnidadesList;** • **private Long nTramiteId;** • **private String cunidadRecepId;** • **private String unidadRecep;** • **private String valora;** • **/*variables para los tramites*/** • **private String maxEspera;** • **private String maxLlamado;** • **private String maxProceso;** • **private String minEspera;** • **private String minLlamado;** • **private String minProceso;** • **private String promEspera;** • **private String promLlamado;** • **private String promProceso;** • **private String modaEspera;** • **private String modaLlamado;** • **private String modaProceso;** • **private MonitoreoSeccionTramites mst;**

Tabla 4.2 Descripción de las clases pojo

Pantalla: Centros de Servicios	
Pojo	Descripción
MonitoreoColasPojo.java	Esta clase permite definir las variables escalares para capturar información de los trámites. Las variables que se encuentran dentro de esta clase son las siguientes: • **private String tramite;** • **private Long totEspera;** • **private Long totProcesados;** • **private Long enProceso;** • **private Long totAnulados;** • **private Long saturacion;** • **private String tiquetes;**

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Centros de Servicios	
Pojo	Descripción
MonitoreoCsPojo.java	Esta clase permite definir las variables escalares para capturar información de las zonas de un centro de servicio. Las variables que se encuentran dentro de esta clase son las siguientes: • **private List<MonitoreoZonasPojo> monitoreoZonasPojo;** • **private String nombre;** • **private String csId;** • **private Long saturacion;**
MonitoreoSeccionPojo.java	Esta clase permite definir las variables escalares para capturar información de las zonas de un centro de servicio, usuarios y trámites. Las variables que se encuentran dentro de esta clase son las siguientes: • **private Long seccionId;** • **private String scNombre;** • **private String maximo;** • **private String minimo;** • **private String promedio;** • **private String moda;** • **private Long esperando;** • **private Long Procesados;** • **private List<MonitoreoSeccionUsuarios> usuarios;** • **private List<MonitoreoSeccionTramites> tramites;**
MonitoreoSeccionTramites.java	Esta clase permite definir las variables escalares para capturar información de los trámites. Las variables que se encuentran dentro de esta clase son las siguientes: • **private String tramite;** • **private Long tramiteId;** • **private Long esperando;** • **private Long hora;** • **private Long min45;** • **private Long min30;** • **private Long min15;** • **private Long actual;** • **private Long procesados;**
MonitoreoTramitePojo.java	Ya se explicó en la pantalla Página índex de Monitoreo

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Centros de Servicios	
Pojo	Descripción
MonitoreoUsuarioPojo.java	Esta clase permite definir las variables escalares para capturar información de los usuarios técnicos que atienden los trámites. Las variables que se encuentran dentro de esta clase son las siguientes: • **private String nombre;** • **private Long escritorioid;** • **private String escritorio;** • **private String rol;** • **private String estado;** • **private String tramites;** • **private String secciones;** • **private String toatendidos;** • **private String isesion;** • **private String iproceso;**
MonitoreoZonasPojo.java	Esta clase permite definir las variables escalares para capturar información de los usuarios técnicos y las zonas. Las variables que se encuentran dentro de esta clase son las siguientes: • **private List<MonitoreoUsuarioPojo> monitoreoUsuarioPojo;** • **private List<MonitoreoColasPojo> monitoreoColasPojo;** • **private String nombre;** • **private Long zonaId;** • **private Long saturacion;**

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Secciones	
Pojo	Descripción
MonitoreoSeccionPojo.java	Ya se explicó en la pantalla de Centros de Servicios
MonitoreoSeccionTramites.java	Ya se explicó en la pantalla de Centros de Servicios
MonitoreoSeccionUsuarios.java	Esta clase permite definir las variables escalares para capturar información de los usuarios técnicos que atienden los trámites. Las variables que se encuentran dentro de esta clase son las siguientes: • **private String usuario;** • **private String estado;** • **private Long escritorioId;**

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Trámites	
Pojo	Descripción
MonitoreoSeccionUsuarios.java	Ya se explicó en la pantalla de Secciones
MonitoreoTramitePojo.java	Ya se explicó en la pantalla Página índex de Monitoreo

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Usuarios	
Pojo	Descripción
MonitoreoUsuarioPojo.java	Ya se explicó en la pantalla de Centros de Servicios

Tabla 4.2 Descripción de las clases pojo, continuación

Pantalla: Alertas	
Pojo	Descripción
MonitoreoTramitePojo.java	Ya se explicó en la pantalla Página índex de Monitoreo

Tabla 4.2 Descripción de las clases pojo, continuación

5. Repositorios

5.1 Listado de Repositorios

En la siguiente tabla se listan las diferentes clases repositorios y el paquete (**sv.gob.mh.dgii.colas.repositories**) en donde se pueden encontrar.

Nombre de Opción	Nombre
Centros de Servicios	**GcServiciosRepository.java** **GcTramiteRepository.java** **GcUnidadRecepRepository.java** **GcZonaRepository.java**
Secciones	**GcServiciosRepository.java** **GcTramiteRepository.java** **GcUsuarioRepository.java**
Trámites	**GcTramiteRepository.java** **GcUnidadRecepRepository.java**
Usuarios	**GcEscritorioRepository.java** **GcUsuarioRepository.java**

Tabla 5.1 Listado de clases repositorios

5.2 Descripción de Repositorios

En la siguiente tabla se describen cada uno de los repositorios usados en el módulo de Monitorio.

Pantalla: Centros de Servicios	
Repositorio	Descripción
GcServiciosRepository.java	Los métodos usados son los siguientes: **public List<GcServicios> getAllServiciosActivo():** Método que construye un named query para buscar el listado de centros de servicio que se encuentran activos, el cual es: "SELECT s FROM GcServicios s where s.bActiva=1 ORDER BY nServiciosId ASC" **public List getMonitoreoSecTramites(String unidadRecep,Long servicioId):** Método que construye un query nativo para buscar el listado de trámites en espera por zona, el cual es: “select ti.n_tramite_id, SIIT.PKG_COLAS_UTILS.SUSTITUIR_SIMBOLOS(ti.s_nombre),\n (select count(*) from gc_tiquete where c_unidad_recep=?1 \n and n_tramite_id=ti.n_tramite_id and trunc(fh_llegada)=trunc(sysdate) and m_estado=1) esperando\n from gc_tramite ti where n_servicios_id=?2”.
GcTramiteRepository.java	El método usado es el siguiente: **public List getColasByZona(String unidadRecep, Long zonaID, Long zonaID2,String unidadRecep2):** Método que construye un query nativo para buscar los trámites que se encuentran en cola por zona, el cual es: “SELECT \n s_nombre,\n SUM(espera) espera,\n SUM(procesados) procesados,\n SUM(procesando) procesando,\n SUM(anulados) anulados,\n trunc(SIIT.PKG_COLAS_UTILS.TRAMITE_SATURACION(n_tramite_id, ?1, ?2)) saturacion,\n nvl(wm_concat(DECODE(m_estado,3, tiquetes,NULL)),'--') tiquete\n FROM (SELECT tra.n_tramite_id ,tra.s_nombre,\n wm_concat(ti.s_correlativo) tiquetes ,\n ti.m_estado,\n SUM(DECODE(ti.m_estado,1,1,0)) espera,\n SUM(DECODE(ti.m_estado,3,1,0)) procesando,\n SUM(DECODE(ti.m_estado,4,1,0)) procesados,\n SUM(DECODE(ti.m_estado,5,1,0)) anulados\n FROM gc_tiquete ti\n LEFT OUTER JOIN gc_tramite tra\n ON ti.n_tramite_id =tra.n_tramite_id\n WHERE m_estado IN (1,2,3,4,5)\n AND ti.n_tramite_id IN\n (SELECT DISTINCT ut.n_tramite_id\n FROM GC_USR_TRA ut\n WHERE ut.c_usuario IN\n (SELECT us.c_usuario\n FROM gc_escritorio es\n LEFT OUTER JOIN gc_usuario us\n ON es.n_escritorio_id=us.n_escritorio_id\n WHERE us.c_usuario IS NOT NULL\n AND es.n_zona_id = ?3))\n AND TRUNC(ti.fh_llegada)=TRUNC(sysdate)\n AND ti.c_unidad_recep = ?4 \n GROUP BY tra.s_nombre,\n ti.m_estado, tra.n_tramite_id)\n GROUP BY s_nombre,n_tramite_id”.

Tabla 5.2 Descripción de las clases repositorios

Pantalla: Centros de Servicios	
Repositorio	Descripción
GcUnidadRecepRepository.java	Los métodos usados son los siguientes: **public List<TbUnidadRecep> getUnidadesEnServicio():** Método que construye un named query para buscar el listado de centros de servicios o unidades receptoras, el cual es: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('M','C') AND c_unidad_Recep NOT IN(SELECT d.id.clistaDet FROM TbListasValorDet d where c_modulo='GC')". **public TbUnidadRecep getUnidadEnServicioE(String cunidadRecep):** Método que construye un named query para buscar el centro de servicio o unidad receptora activo, el cual es: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('C','M') and t.cunidadRecep = ?1 ". **public List getUserMonitoreo(String unidadRecep):** Método que construye un query nativo para buscar el listado de técnicos activos por zona para un centro de servicio dado, el cual es: “select nombre,escritorioid,escritorio,rol,estado,tramites,secciones \n from (SELECT us.c_usuario nombre,\n es.n_escritorio_id escritorioid ,\n es.c_identificador ||es.n_num_escritorio escritorio,\n 'Operador' rol ,\n SIIT.PKG_COLAS_UTILS.ESTADO_USUARIO(us.c_usuario) estado ,\n SIIT.PKG_COLAS_UTILS.TRAMITES_USUARIO(us.c_usuario) tramites,\n SIIT.PKG_COLAS_UTILS.SECCIONES_USUARIO(us.c_usuario) secciones\n FROM GC_USUARIO us\n LEFT OUTER JOIN gc_escritorio es\n ON us.n_escritorio_id=es.n_escritorio_id\n WHERE es.n_zona_id = ?1) \n where tramites is not null”. **public Long getCsSaturacion(String unidadRecep):** Método que llama a una función para conocer el porcentaje de saturación de un centro de servicio dado, la cual es: "SELECT SIIT.PKG_COLAS_UTILS.UNIDAD_SATURACION(?1) FROM DUAL".
GcZonaRepository.java	Los métodos usados son los siguientes: **public List<GcZona> getZonasByUnidad(String cunidadRecepId):** Método que construye un named query para buscar el listado de zonas por centro de servicio o unidad receptora, el cual es: "SELECT t FROM GcZona t where C_UNIDAD_RECEP = ?1 and t.ffVigencia is null ". **public Long getSaturacionZona(String cenServ, Long zonaId,String zona):** Método que llama a una función para conocer el porcentaje de saturación de una zona dada, la cual es: "SELECT SIIT.PKG_COLAS_UTILS.ZONA_SATURACION(?1 , ?2 , ?3) SATURACION FROM DUAL".

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Secciones	
Repositorio	Descripción
GcServiciosRepository.java	Los métodos usados son los siguientes: **public List<GcServicios> getAllServicios():** Método que construye un named query para buscar el listado todos los servicios, el cual es: "SELECT s FROM GcServicios s ORDER BY nServiciosId ASC ". **public T findOne(ID id):** Método genérico que devuelve una entidad en base a su llave primaria, para este caso permite buscar un servicio en base a una sección. **public List getMonitoreoServicio(@Param("servicioId")Long servicioId,@Param("unidadRecep")String unidadRecep):** Método que construye un query nativo para buscar información de los servicios por unidad receptora, el cual es: "SELECT SIIT.PKG_COLAS_UTILS.SECCION_MAXIMO_ESPERA(:servicioId,:unidadRecep) MAXIMO,\n SIIT.PKG_COLAS_UTILS.SECCION_MINIMO_ESPERA(:servicioId,:unidadRecep) MINIMO,\n SIIT.PKG_COLAS_UTILS.SECCION_PROMEDIO_ESPERA(:servicioId,:unidadRecep) PROMEDIO,\n SIIT.PKG_COLAS_UTILS.SECCION_MODA_ESPERA(:servicioId,:unidadRecep) MODA,\n (SELECT COUNT(*) FROM GC_TIQUETE WHERE M_ESTADO=1 AND \n C_UNIDAD_RECEP=:unidadRecep AND N_TRAMITE_ID IN (SELECT N_TRAMITE_ID \n FROM GC_TRAMITE WHERE N_SERVICIOS_ID=:servicioId) AND TRUNC(FH_LLEGADA)=TRUNC(SYSDATE)) ESPERANDO,\n (SELECT COUNT(*) FROM GC_TIQUETE WHERE M_ESTADO=4 AND C_UNIDAD_RECEP=:unidadRecep \n AND N_TRAMITE_ID IN (SELECT N_TRAMITE_ID FROM GC_TRAMITE WHERE N_SERVICIOS_ID=:servicioId) \n AND TRUNC(FH_LLEGADA)=TRUNC(SYSDATE)) PROCESADOS FROM DUAL".
GcTramiteRepository.java	Los métodos usados son los siguientes: **public List getTramitesBySeccion(@Param("seccionId")Long seccionId,@Param("unidadRecep")String unidadRecep):** Método que construye un query nativo para buscar el listado de trámites por sección, el cual es: "select tra.n_tramite_id, SIIT.PKG_COLAS_UTILS.SUSTITUIR_SIMBOLOS(tra.s_nombre),\n (select count(*) from gc_tiquete where n_tramite_id=tra.n_tramite_id \n and m_estado=1 and trunc(fh_llegada)=trunc(sysdate) and c_unidad_recep=:unidadRecep) coun, \n (select count(*) from gc_tiquete where n_tramite_id=tra.n_tramite_id \n" and m_estado=4 and trunc(fh_llegada)=trunc(sysdate) and c_unidad_recep=:unidadRecep) procesados from gc_tramite tra where n_servicios_id=:seccionId". **public List getSeriesTramiteEsperando(@Param("tramiteId")Long tramiteId,@Param("unidadRecep")String unidadRecep):** Método que construye un query nativo para buscar el listado de trámites en espera por centro de servicio el cual es: "SELECT (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-1/24)\n and (fh_llamado > (sysdate-((1/24)-15/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) HORA,\n (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-45/1440)\n and (fh_llamado > (sysdate-(1/24)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN45,\n …

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Secciones	
Repositorio	Descripción
GcTramiteRepository.java	… (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-30/1440)\n and (fh_llamado > (sysdate-(45/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN30 ,\n (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-15/1440)\n and (fh_llamado > (sysdate-(30/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN15,\n (select COUNT(*) from gc_tiquete \n where c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) ESPERANDO FROM DUAL".
GcUsuarioRepository.java	El método usado es el siguiente: **public List getUsuariosBySeccion(Long seccionId):** Método que construye un query nativo para buscar el listado de usuarios técnicos por sección seleccionada, el cual es: "select n_escritorio_id,SIIT.PKG_COLAS_UTILS.SUSTITUIR_SIMBOLOS(c_usuario) usuario,SIIT.PKG_COLAS_UTILS.ESTADO_USUARIO(c_usuario) ESTADO \n from gc_usuario where c_usuario in(select c_usuario from gc_usr_tra \n where n_tramite_id in (select n_tramite_id from gc_tramite where n_servicios_id=?1))".

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Trámites	
Repositorio	Descripción
GcTramiteRepository.java	Los métodos usados son los siguientes: **public Iterable<T> findAll():** Método genérico que devuelve todas las instancias de una entidad dada, para este caso permite buscar el listado de trámites. **public List<GcTramite> getAllTramites():** Método que construye un query nativo para buscar el listado de todos los trámites activos, el cual es: "SELECT t FROM GcTramite t where t.bActiva=1". **public List getTramitesByEstados(@Param("tramiteId")Long seccionId,@Param("unidadRecep")String unidadRecep):** Método que construye un query nativo para busca el listado de trámites por estado y por centro de servicio, el cual es: "select tra.n_tramite_id, SIIT.PKG_COLAS_UTILS.SUSTITUIR_SIMBOLOS(tra.s_nombre),\n (select count(*) from gc_tiquete where n_tramite_id=tra.n_tramite_id \n and m_estado=1 and trunc(fh_llegada)=trunc(sysdate) and c_unidad_recep=:unidadRecep) coun, \n (select count(*) from gc_tiquete where n_tramite_id=tra.n_tramite_id \n and m_estado=4 and trunc(fh_llegada)=trunc(sysdate) and c_unidad_recep=:unidadRecep) procesados from gc_tramite tra where n_tramite_id=:tramiteId". **public List getSeriesTramiteEsperando(@Param("tramiteId")Long tramiteId,@Param("unidadRecep")String unidadRecep):** Método que construye un query nativo busca el listado de trámites en espera por centro de servicio, el cual es: "SELECT (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-1/24)\n and (fh_llamado > (sysdate-((1/24)-15/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n …

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Trámites	
Repositorio	Descripción
GcTramiteRepository.java	… and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) HORA,\n (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-45/1440)\n and (fh_llamado > (sysdate-(1/24)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN45,\n (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-30/1440)\n and (fh_llamado > (sysdate-(45/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN30 ,\n (select COUNT(*) from gc_tiquete \n where fh_llegada <= (sysdate-15/1440)\n and (fh_llamado > (sysdate-(30/1440)) or fh_llamado is null)\n and c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) MIN15,\n (select COUNT(*) from gc_tiquete \n where c_unidad_recep=:unidadRecep\n and n_tramite_id=:tramiteId and m_estado=1\n and trunc(fh_llegada)=trunc(sysdate)) ESPERANDO FROM DUAL". **public List getTramiteEspera(Long tramiteId,String unidadRecep):** Método que construye un query nativo para buscar el listado de trámites en espera por centro de servicio, el cual es: "SELECT \n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MAXIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MAXIMO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MAXIMO),86400),3600),60),'09'),'00'),' ','') maximo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MINIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MINIMO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MINIMO),86400),3600),60),'09'),'00'),' ','') manimo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(PROMEDIO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(PROMEDIO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(PROMEDIO),86400),3600),60),'09'),'00'),' ','') PROM_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MODA),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MODA),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MODA),86400),3600),60),'09'),'00'),' ','') MODA_espera\n FROM(SELECT MAX(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MAXIMO,\n MIN(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MINIMO,\n AVG(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 PROMEDIO,\n stats_mode(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MODA\n FROM GC_TIQUETE WHERE N_TRAMITE_ID=?1 \n AND M_ESTADO=1 AND TRUNC(FH_LLEGADA)=TRUNC(SYSDATE) AND C_UNIDAD_RECEP=?2)". **public List getTramiteLlamado(Long tramiteId,String unidadRecep):** Método que construye un query nativo para buscar el listado de trámites en llamado por centro de servicio, el cual es: "SELECT \n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MAXIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MAXIMO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MAXIMO),86400),3600),60),'09'),'00'),' ','') maximo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MINIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MINIMO),86400),3600)/60),'09'),'00') ||':'||\n …

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Trámites	
Repositorio	Descripción
GcTramiteRepository.java	… NVL(TO_CHAR(MOD(MOD(MOD(SUM(MINIMO),86400),3600),60),'09'),'00'),' ',") manimo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(PROMEDIO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(PROMEDIO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(PROMEDIO),86400),3600),60),'09'),'00'),' ',") PROM_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MODA),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MODA),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MODA),86400),3600),60),'09'),'00'),' ',") MODA_espera\n FROM(SELECT MAX(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MAXIMO,\n MIN(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MINIMO,\n AVG(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 PROMEDIO,\n stats_mode(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MODA\n FROM GC_TIQUETE WHERE N_TRAMITE_ID=?1 \n AND M_ESTADO=1 AND TRUNC(FH_LLEGADA)=TRUNC(SYSDATE) AND C_UNIDAD_RECEP=?2)". **public List getTramiteProceso(Long tramiteId,String unidadRecep):** Método que construye un query nativo para buscar el listado de trámites en proceso por centro de servicio, el cual es: " SELECT \n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MAXIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MAXIMO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MAXIMO),86400),3600),60),'09'),'00'),' ',") maximo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MINIMO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MINIMO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MINIMO),86400),3600),60),'09'),'00'),' ',") manimo_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(PROMEDIO),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(PROMEDIO),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(PROMEDIO),86400),3600),60),'09'),'00'),' ',") PROM_espera,\n REPLACE(NVL(TO_CHAR(trunc(MOD(SUM(MODA),86400)/3600),'09'),'00') ||':'||\n NVL(TO_CHAR(trunc(MOD(MOD(SUM(MODA),86400),3600)/60),'09'),'00') ||':'||\n NVL(TO_CHAR(MOD(MOD(MOD(SUM(MODA),86400),3600),60),'09'),'00'),' ',") MODA_espera\n FROM(SELECT MAX(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MAXIMO,\n MIN(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MINIMO,\n AVG(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 PROMEDIO,\n stats_mode(NVL(FH_LLAMADO,SYSDATE)-FH_LLEGADA)*24*60*60 MODA\n FROM GC_TIQUETE WHERE N_TRAMITE_ID=?1 \n AND M_ESTADO=1 AND TRUNC(FH_LLEGADA)=TRUNC(SYSDATE) AND C_UNIDAD_RECEP=?2)". **public List getUsuarioByTramite(Long tramiteId,String unidadRecep):** Método que construye un query nativo para buscar el listado de usuarios técnicos por trámite y por centro de servicio, el cual es: "SELECT C_USUARIO,SIIT.PKG_COLAS_UTILS.ESTADO_USUARIO(C_USUARIO) estado FROM GC_USUARIO \n WHERE C_USUARIO IN(SELECT C_USUARIO FROM GC_USR_TRA WHERE N_TRAMITE_ID =?1) AND N_ESCRITORIO_ID IN(SELECT N_ESCRITORIO_ID FROM GC_ESCRITORIO WHERE C_UNIDAD_RECEP=?2)".

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Trámites	
Repositorio	Descripción
GcUnidadRecepRepository.java	El método usado es el siguiente: **public List<TbUnidadRecep> getUnidadesEnServicio():** Método que construye un named query para busca el listado de centros de servicio o unidades receptoras, el cual es: "SELECT t FROM TbUnidadRecep t WHERE t.mservicio IN('M','C') AND c_unidad_Recep NOT IN(SELECT d.id.clistaDet FROM TbListasValorDet d where c_modulo='GC')".

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Usuarios	
Repositorio	Descripción
GcEscritorioRepository.java	Los métodos usados son los siguientes: **public List allEscritoriosDisponibles(String unidadRecep):** Método que construye un query nativo para buscar el listado de los escritorios disponibles por centro de servicio, el cual es: “select n_escritorio_id,c_identificador||n_num_escritorio escritorio \n from gc_escritorio where c_unidad_recep=?1 \n and n_escritorio_id not in(select n_escritorio_id from gc_usuario)\n and b_activa=1 and ff_vigencia is null”. **public T findOne(ID id):** Método genérico que devuelve una entidad en base a su llave primaria, para este caso permite buscar un escritorio.
GcUsuarioRepository.java	Los métodos usados son los siguientes: **public List getUsuariosbyCS(String unidadRecep):** Método que construye un query nativo para buscar el listado de usuarios técnicos por centro de servicio, el cual es: “select nombre,n_zona_id,escritorioid,escritorio,rol,estado,tramites,secciones ,'6:55am' isesion,'7:00am' iproceso, \n (select count(*) from gc_tiquete where c_usuario_atendio=nombre and trunc(fh_llegada)=trunc(sysdate)) PROCESADOS \n from (SELECT us.c_usuario nombre,es.n_zona_id, \n es.n_escritorio_id escritorioid , \n es.c_identificador ||es.n_num_escritorio escritorio, \ 'Operador' rol , \n SIIT.PKG_COLAS_UTILS.ESTADO_USUARIO(us.c_usuario) estado , \n SIIT.PKG_COLAS_UTILS.TRAMITES_USUARIO(us.c_usuario) tramites, \n SIIT.PKG_COLAS_UTILS.SECCIONES_USUARIO(us.c_usuario) secciones \n FROM GC_USUARIO us \n LEFT OUTER JOIN gc_escritorio es \n ON us.n_escritorio_id=es.n_escritorio_id \n WHERE c_unidad_recep =?1) \n where tramites is not null” **public T findOne(ID id):** Método genérico que devuelve una entidad en base a su llave primaria, para este caso permite buscar un usuario técnico. **public <S extends T> S save(S s):** Método genérico que permite guardar en la bd una entidad dada, para este caso guarda un usuario técnico.

Tabla 5.2 Descripción de las clases repositorios, continuación

Pantalla: Alertas	
Repositorio	Descripción
GcMonitoreoRepository.java	Los métodos usados son los siguientes: **public List<String> findZonas(String CS):** Método que construye un query nativo para busca el listado de zonas por centros de servicio o unidades receptoras, el cual es: "select to_char(n_zona_id) from gc_zona where c_unidad_recep=?1". **public List<String> findtramites(String zona,String CS):** Método que construye un query nativo para buscar el listado de trámites por zona y centro de servicio, el cual es: “select distinct to_char(n_tramite_id) from gc_usuario a inner join gc_usr_tra b on a.c_usuario=b.c_usuario inner join gc_escritorio c on a.n_escritorio_id=c.n_escritorio_id where n_zona_id=?1 and c_unidad_recep=?2 order by 1”. **public String findalertas(String tramite,String unidad,String zona):** Método que llama a una función para conocer el porcentaje de saturación de un trámite, la cual es: "select pkg_colas_utils.TRAMITE_SATURACION_ALERTA(?1,?2,?3) from dual". **public List<String> zona_id(String unidad):** Método que construye un query nativo para buscar el listado de nombres de las zonas por centro de servicio, el cual es: "select s_nombre||';'||n_zona_id from gc_zona where C_UNIDAD_RECEP=?1 and b_activa=1". **public String findZonas(String unidad,String zona_id,String zona):** Método que llama a una función para conocer el porcentaje de saturación de una zona, la cual es: "select pkg_colas_utils.ZONA_SATURACION_ALERTA(?1,?2,?3) from dual". **public String findUnidad(String unidad):** Método que llama a una función para conocer el porcentaje de saturación de un centro de servicio, la cual es: "select pkg_colas_utils.UNIDAD_SATURACION_ALERTA(?1) from dual".

Tabla 5.2 Descripción de las clases repositorios, continuación

6. Componentes

6.1 Listado de Componentes

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

Tabla 6.1 Listado de Componentes

6.2 Descripción de Componentes

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 Descripción de Componentes

7. Configuración del Proyecto (@Configuration y @Beans)

Tabla conteniendo las clases de configuración del proyecto y la explicación de sus métodos, en general, esto es todo el paquete de **sv.gob.mh.dgii.colas.config**.

Nombre del archivo de configuración	Ubicación sv.gob.mh.dgii.colas.config	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
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();**

Nombre del archivo de configuración	Ubicación sv.gob.mh.dgii.colas.config	Descripción
	Método que invoca el paquete "sv.gob.mh.dgii.colas.security.PropertyFileSecurityBuilder". **public AuthenticationManager authenticationManagerBean();** Método de anulación authenticationManagerBean en WebSecurityConfigurerAdapter para exponer el 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(AuthenticationManagerBuilder 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.**	
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.	

Nombre del archivo de configuración	Ubicación sv.gob.mh.dgii.colas.config	Descripción
	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():** 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. Estructura de Reportes

El módulo de Monitoreo no utiliza reportes

9. Documentos Relacionados a la Base de Datos

9.1 Tablas

En las siguientes páginas se describen cada uno de los campos de las tablas de la base de datos. Cuando la **descripción no tiene nada**, quiere decir que dicha tabla no **tiene comentarios en la base de datos**.

GC_TRAMITE			
SECUENCIA: SEQ_GC_TRAMITE			
Nombre del atributo	**Tipo de dato**	**Null**	Descripción
N_TRAMITE_ID	NUMBER	No	Llave primaria de la tabla
N_SERVICIOS_ID	NUMBER	No	Referencia al servicio a que pertenece el trámite
N_PESO	NUMBER	No	Cuantificación de la prioridad que debe tener el tramite luego de aplicarle todos los criterios
B_NIT_REQUERIDO	NUMBER(1,0)	No	Bandera que indica si es obligatorio proporcionar el nit
B_ESCALAMIENTO	NUMBER(1,0)	No	Bandera que indica si el tramite será escalado al supervisor
D_TRAMITE	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 esta activo o no
S_NOMBRE	VARCHAR2(256 BYTE)	No	Nombre del tramite
N_ORDEN	NUMBER	Yes	Orden en que se presentaran los tramites en pantalla

Tabla 9.1 GC_TRAMITE

GC_SERVICIOS			
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 esta activo o no
S_NOMBRE	VARCHAR2(256 BYTE)	Yes	Nombre del centro de servicio

Tabla 9.2 GC_SERVICIOS

GC_ZONA			
SECUENCIA: SEQ_GC_ZONA			
Nombre del atributo	**Tipo de dato**	**Null**	Descripción
N_ZONA_ID	NUMBER	No	Llave primaria de la tabla
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
N_SILLAS_ESPERA	NUMBER	Yes	Número de sillas en la zona de atención
B_ACTIVA	NUMBER(1,0)	No	Bandera que indica si el registro esta activo o no
D_ZONA	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
S_NOMBRE	VARCHAR2(256 BYTE)	Yes	Nombre de la zona

Tabla 9.3 GC_ZONA

GC_ESCRITORIO			
SECUENCIA: SEQ_GC_ESCRITORIO			
Nombre del atributo	**Tipo de dato**	**Null**	Descripción
N_ESCRITORIO_ID	NUMBER	No	Llave primaria de la tabla
N_ZONA_ID	NUMBER	No	Referencia a la zona a que pertenece el escritorio
C_UNIDAD_RECEP	VARCHAR2(5 BYTE)	No	Código del centro de servicio
B_ACTIVA	NUMBER(1,0)	No	Bandera que indica si el registro esta activo o no
D_ESCRITORIO	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
C_IDENTIFICADOR	VARCHAR2(1 BYTE)	Yes	Letra que identifica a un escritorio al momento del llamado
N_NUM_ESCRITORIO	NUMBER	No	Número con el cual se identifica al escritorio al momento del llamado
B_FILA_ESP	NUMBER(1,0)	No	Si el escritorio será Fila Especial

Tabla 9.4 GC_ESCRITORIO

GC_USUARIO			
SECUENCIA: No tiene			
Nombre del atributo	**Tipo de dato**	**Null**	Descripción
N_ESCRITORIO_ID	NUMBER(38,0)	No	Escritorio asociado al usuario
C_USUARIO	VARCHAR2(256 BYTE)	No	Código del usuario

Tabla 9.5 GC_USUARIO

TB_UNIDAD_RECEP			
SECUENCIA: No tiene			
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	Descripcion
FH_INGRESO	DATE	Yes	Fecha de ingreso
B_STATUS	NUMBER(1,0)	No	Estado
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	Codigo de departamento
S_UBIC_GEOGRAF	VARCHAR2(60 BYTE)	Yes	Descripcion de la ubicacion geografica.
C_UNIDAD_RECEP_SUP	VARCHAR2(5 BYTE)	Yes	
M_SERVICIO	VARCHAR2(1 BYTE)	Yes	

Tabla 9.6 TB_UNIDAD_RECEP

10. Seguridad

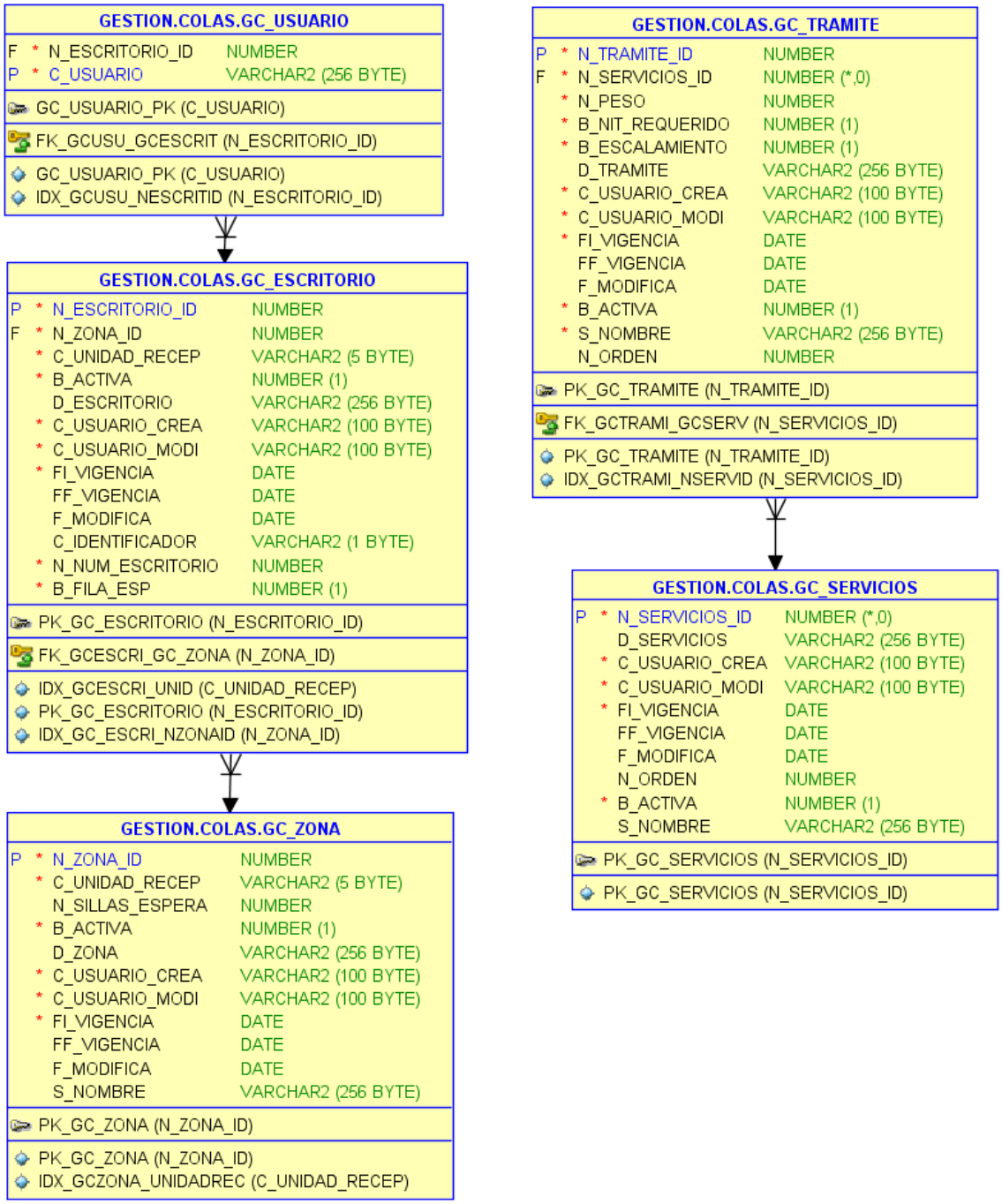
Tabla conteniendo los roles de seguridad usados en el Módulo de Monitoreo

Nombre de la opción	Url	Roles
Página índex de Monitoreo	/monitoreo/	ROLE_GC_
Centros de Servicios	/cdservicio/	
Secciones	/seccionm/	
Trámites	/monitoreoT/	
Usuarios	/usuariom/	
Alertas	/alrts/	

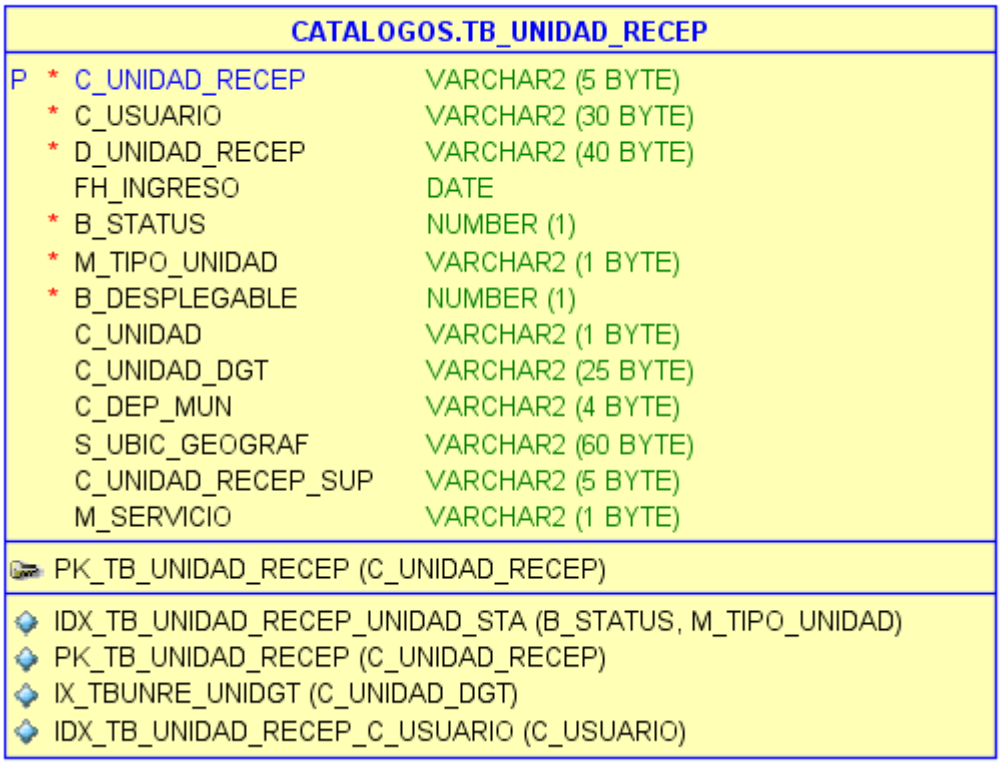
Tabla 10.1 Roles del Módulo de Monitoreo

11. 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)