MANUAL TECNICO MONITOREO

SISTEMA DE ATENCIÓN AL CONTRIBUYENTE (SAC)

Tabla de contenido

[1. Breve Descripción del Módulo de Monitoreo 2](#_Toc433725676)

[2. Diagrama Entidad – Relación 3](#_Toc433725677)

[3. Estructura de Componentes (Capas Vista y Controlador) 4](#_Toc433725678)

[3.1 Vistas o páginas HTML 4](#_Toc433725679)

[3.2 Controladores 5](#_Toc433725680)

[3.3 Descripción de Controladores 6](#_Toc433725681)

[4. Clases POJOS 10](#_Toc433725682)

[4.1 Listado de clases POJOS 10](#_Toc433725683)

[4.2 Descripción de clases POJOS 11](#_Toc433725684)

[5. Repositorios 15](#_Toc433725685)

[5.1 Listado de Repositorios 15](#_Toc433725686)

[5.2 Descripción de Repositorios 16](#_Toc433725687)

[6. Componentes 24](#_Toc433725688)

[6.1 Listado de Componentes 24](#_Toc433725689)

[6.2 Descripción de Componentes 24](#_Toc433725690)

[7. Configuración del Proyecto (@Configuration y @Beans) 25](#_Toc433725691)

[8. Estructura de Reportes 28](#_Toc433725692)

[9. Documentos Relacionados a la Base de Datos 29](#_Toc433725693)

[9.1 Tablas 29](#_Toc433725694)

[10. Seguridad 32](#_Toc433725695)

[11. Anexos 33](#_Toc433725696)

# 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.

# 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

# 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

# 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

# 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

# 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

# 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();**  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**](http://docs.spring.io/autorepo/docs/spring-security/3.2.3.RELEASE/apidocs/org/springframework/security/access/AccessDecisionManager.html)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](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/config/annotation/WebMvcConfigurer.html) o más probablemente extender el método vacío clase base [WebMvcConfigurerAdapter](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/config/annotation/WebMvcConfigurerAdapter.html) 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():**  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

# Estructura de Reportes

El módulo de Monitoreo no utiliza reportes

# 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

# Seguridad

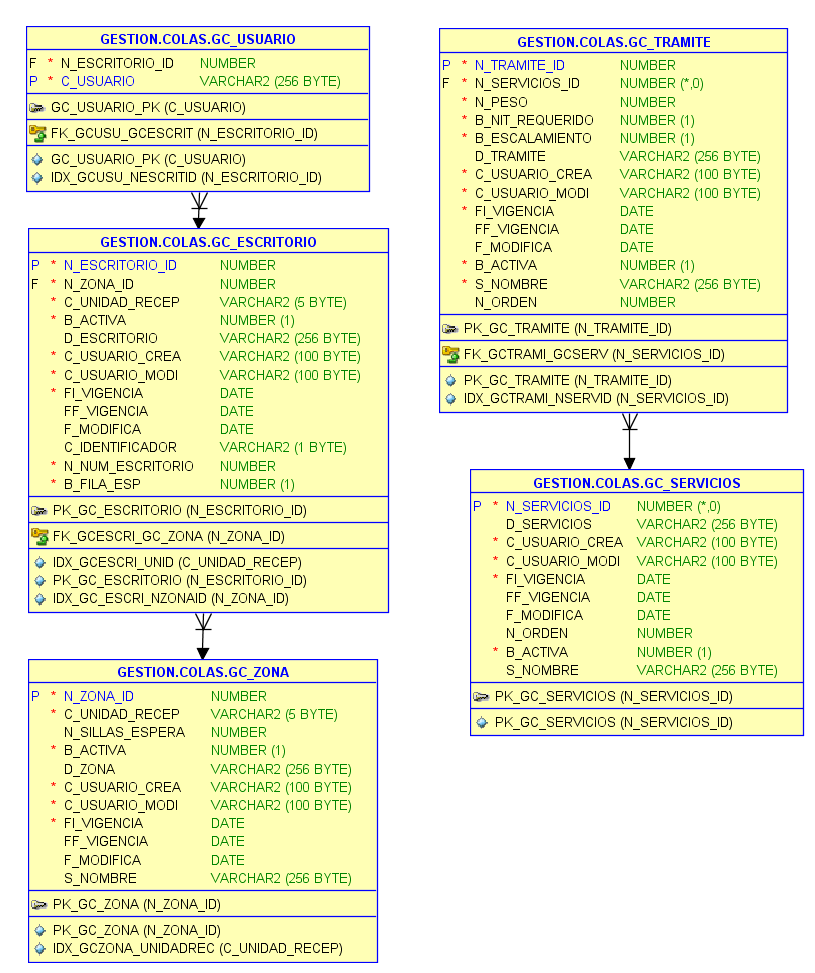
Tabla conteniendo los roles de seguridad usados en el Módulo de Monitoreo

|  |  |  |
| --- | --- | --- |
| 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

# Anexos

Anexo 1



Anexo 2

