Manual Técnico Reporte y Estadística

Sistema de Atención al Contribuyente (SAC)

Tabla de contenido

[1. Breve descripción del Módulo de Reporte y Estadística. 1](#_Toc437937042)

[2. Diagrama Entidad – Relación 2](#_Toc437937043)

[3. Estructura de Componentes (Capa vista, controlador, servicio DAO y estructura de reportes) 3](#_Toc437937044)

[3.1 Vistas o páginas HTML 3](#_Toc437937045)

[3.2 Controladores 4](#_Toc437937046)

[3.3 Descripción de Controladores. 6](#_Toc437937047)

[4. Clases POJO 32](#_Toc437937048)

[4.1 Listado de clases pojo 32](#_Toc437937049)

[4.2 Descripción de Clases POJO. 34](#_Toc437937050)

[5. Repositorios 35](#_Toc437937051)

[5.1 Descripción de Repositorios. 37](#_Toc437937052)

[6. Componentes 39](#_Toc437937053)

[6.1 Listado de Componentes 39](#_Toc437937054)

[6.2 Descripción de Componentes 39](#_Toc437937055)

[7. Configuración. 40](#_Toc437937056)

[8. Documentos Relacionados a la Base de Datos 43](#_Toc437937057)

[8.1 Tablas 43](#_Toc437937058)

[9. Seguridad 45](#_Toc437937059)

[10. Anexos 46](#_Toc437937060)

# Breve descripción del Módulo de Reporte y Estadística.

El módulo de Reportes y Estadística tiene como finalidad mostrar los datos en información necesaria para la toma de decisiones, dicho módulo cuenta con 20 diferentes tipos de reportes y estadísticas, siendo estos clasificados en reportes de atención de trámites y reportes de usuarios.

La pantalla principal (menú) de Reportes y Estadística muestra los diferentes reportes o estadísticas a las cuales puede accesar y realizar el filtrado de datos de acuerdo al tipo de reporte o estadística previamente seleccionado para generar.

# Diagrama Entidad – Relación

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

|  |  |  |
| --- | --- | --- |
| Tabla | Esquema | Anexo |
| GC\_TRAMITE | GESTION.COLAS | Anexo 1 |
| GC\_SERVICIOS |
| GC\_TIQUETE |
| GC\_EVENTOS |
| GC\_PRIORIDAD |
| GC\_RESERVA\_CITA |
| GC\_USER\_LOG |
| GC\_USUARIO |
| GC\_PREGUNTAS |
| GC\_PREGUNTAS\_RESPUESTAS |
| GC\_RESPUESTAS |
| GC\_ESTADOS |

Tabla 2.1 Diagrama ER.

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

## 3.1 Vistas o páginas HTML

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre de opción | Ubicación | | Nombre | |
| Reporte de afluencia de contribuyentes por hora. | | /Other Sources/src/main/resources/templates/reportes | | 0701.html | |
| Reporte de tiempos de espera de atención. | | 0702.html | |
| Reporte de contribuyentes atendidos por trámites. | | 0703.html | |
| Reporte de atención de Fila especial. | | 0704.html | |
| Reporte de confirmación de citas. | | 0705.html | |
| Reporte de tiquetes emitidos. | | 0706.html | |
| Reporte de tiquetes reasignados. | | 0707.html | |
| Reporte de evaluación del servicio. | | 0708.html | |
| Reporte consolidado de afluencia de contribuyentes. | | 0709.html | |
| Reporte Tiquetes emitidos, atendidos y no atendidos. | | 0710.html | |
| Comparativo de trámites por centro de servicios. | | 0711.html | |
| Comparativo de tiempo de trámites por centros de servicios. | | 0712.html | |
| Comparativo de Tiquetes emitidos, atendidos y no atendidos. | | 0713.html | |
| Comparativo de afluencia por centro atención. | | 0714.html | |
| Reporte de tiempos de atención por Usuario. | | 0715.html | |
| Reporte de Tiquetes atendidos por Usuario. | | 0716.html | |
| Reporte de Índice de trámites atendidos por Usuario. | | 0717.html | |
| Reporte de Actividades Diarias del Usuario. | | 0718.html | |
| Reporte general de Usuarios. | | 0719.html | |
| Reporte de pausas, recesos y permisos por Usuario. | | 0720.html | |
| -- | | home.html | |
| Atención de Trámites | | mTramites.html | |
| Reportes de Usuarios | | mUsuarios.html | |

Tabla 3.1 Vistas o paginas HTML

## 3.2 Controladores

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

**sv.gob.mh.dgii.colas.controllers.reportes**

| Opción | Controlador |
| --- | --- |
| -- | reportesController.java |
| Reporte de afluencia de contribuyentes por hora. | Impresion0701.java |
| reporteAfluenciaporhora.java |
| Reporte de tiempos de espera de atención. | Impresion0702.java |
| reportetiempoespera.java |
| Reporte de contribuyentes atendidos por trámites. | Impresion0703.java |
| reportecontatendidos.java |
| Reporte de atención de Fila especial. | Impresion.java |
| reporteFilaEpecial.java |
| Reporte de confirmación de citas. | Impresion0705.java |
| confirmacionCitas.java |
| Reporte de tiquetes emitidos. | Impresion0706.java |
| reporteTiqueteE.java |
| Reporte de tiquetes reasignados. | Impresion0707.java |
| reporteTiqueteR.java |
| Reporte de evaluación del servicio. | Impresion0708.java |
| reporteEvalServicio.java |
| Reporte consolidado de afluencia de contribuyentes. | Impresion0709.java |
| reporteConsAfluenciaCont.java |
| Reporte Tiquetes emitidos, atendidos y no atendidos. | Impresion0710.java |
| reporteTiqueteEmiAteNoAte.java |
| Comparativo de trámites por centro de servicios. | Impresion0711.java |
| reporteComparaTramCS.java |
| Comparativo de tiempo de trámites por centros de servicios. | Impresion0712.java |
| reporteComparaTiempoTraCS.java |
| Comparativo de Tiquetes emitidos, atendidos y no atendidos. | Impresion0713.java |
| reporteComparaTiqueEmiAteNoAte.java |
| Comparativo de afluencia por centro atención. | Impresion0714.java |
| reporteComparaAfluenciaCA.java |
| Reporte de tiempos de atención por Usuario. | Impresion0715.java |
| reporteTiempoAten.java |
| Reporte de Tiquetes atendidos por Usuario. | Impresion0716.java |
| reporteTiquetesAte.java |
| Reporte de Índice de trámites atendidos por Usuario. | Impresion0717.java |
| reporteIndiceTramit.java |
| Reporte de Actividades Diarias del Usuario. | Impresion0718.java |
| reporteActDiarias.java |
| Reporte general de Usuarios. | Impresion0719.java |
| reporteGralUsuario.java |
| Reporte de pausas, recesos y permisos por Usuario. | Impresion0720.java |
| Reporteprp.java |

Tabla 3.2 Lista de Controladores

## 3.3 Descripción de Controladores.

A continuación se proporciona una descripción de la funcionalidad principal de los controladores utilizados en el módulo de Reporte y Estadística.

|  |  |
| --- | --- |
| **PANTALLA: Página índex de Reportes y Estadística** | |
| **Controlador:** | **reporteController.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 /reportes/home.html; es decir carga la página índex del módulo de Reportes y Estadísticas, en donde se muestran las dos opciones para acceder a los diferentes tipos de reportes de dicho módulo.  **@RequestMapping("/contribuyente ")**  **public String indexContribuyente(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model):**  Método utilizado para cargar en una pantalla modal las opciones de los diferentes reportes de atención de trámites.  **@RequestMapping("/usuario")**  **public String indexUsuario(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model):**  Método utilizado para cargar en una pantalla modal las opciones de los diferentes reportes de usuarios. | |

Tabla 3.3 Controlador reporteController

|  |  |
| --- | --- |
| **PANTALLA: Reporte de afluencia de contribuyentes por hora.** | |
| **Controlador:** | **Impresion0701.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de afluencia de contribuyentes por hora, a la vez este método invoca al jasper con el nombre 0701.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de afluencia de contribuyentes por hora, a la vez este método invoca a jasper con el nombre 0701.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.4 Controlador impresion0701

|  |  |
| --- | --- |
| **PANTALLA: Reporte de afluencia de contribuyentes por hora.** | |
| **Controlador:** | **reporteAfluenciaporhora.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 del reporte de Afluencia por hora, por medio del return de la url /reportes/0701.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.5 Controlador reporteAfluenciaporhora

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiempos de espera de atención.** | |
| **Controlador:** | **Impresion0702.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiempos de espera de atención, a la vez este método invoca al jasper con el nombre 0702.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiempos de espera de atención, a la vez este método invoca a jasper con el nombre 0702.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.6 Controlador impresion0702

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiempos de espera de atención.** | |
| **Controlador:** | **reportetiempoespera.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 del reporte, por medio del return de la url /reportes/0702.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 reportetiempoespera

|  |  |
| --- | --- |
| **PANTALLA: Reporte de contribuyentes atendidos por trámites.** | |
| **Controlador:** | **Impresion0703.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de contribuyentes atendidos por trámites, a la vez este método invoca al jasper con el nombre 0703.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de contribuyentes atendidos por trámites, a la vez este método invoca a jasper con el nombre 0703.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.8 Controlador impresion0703

|  |  |
| --- | --- |
| **PANTALLA: Reporte de contribuyentes atendidos por trámites.** | |
| **Controlador:** | **reportecontatendidos.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 del reporte, por medio del return de la url /reportes/0703.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.9 Controlador reportecontatendidos

|  |  |
| --- | --- |
| **PANTALLA: Reporte de atención de fila especial.** | |
| **Controlador:** | **Impresion0704.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de atención de fila especial, a la vez este método invoca al jasper con el nombre 0704.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de atención de fila especial, a la vez este método invoca a jasper con el nombre 0704.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.10 Controlador impresion0704

|  |  |
| --- | --- |
| **PANTALLA: Reporte de atención de fila especial.** | |
| **Controlador:** | **reporteFilaEspecial.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 del reporte, por medio del return de la url /reportes/0704.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.11 Controlador reporteFilaEspecial

|  |  |
| --- | --- |
| **PANTALLA: Reporte de confirmación de citas.** | |
| **Controlador:** | **Impresion0705.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de confirmación de citas, a la vez este método invoca al jasper con el nombre 0705.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de confirmación de citas, a la vez este método invoca a jasper con el nombre 0705.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.12 Controlador impresion0705

|  |  |
| --- | --- |
| **PANTALLA: Reporte de confirmación de citas.** | |
| **Controlador:** | **confirmacionCitas.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 del reporte, por medio del return de la url /reportes/0705.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.13 Controlador confirmacionCitas

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes emitidos.** | |
| **Controlador:** | **Impresion0706.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes emitidos, a la vez este método invoca al jasper con el nombre 0706.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes emitidos, a la vez este método invoca a jasper con el nombre 0706.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.14 Controlador impresion0706

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes emitidos.** | |
| **Controlador:** | **reporteTiqueteE.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 del reporte, por medio del return de la url /reportes/0706.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.15 Controlador reporteTiqueteE

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes reasignados.** | |
| **Controlador:** | **Impresion0707.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes reasignados, a la vez este método invoca al jasper con el nombre 0707.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes reasignados, a la vez este método invoca a jasper con el nombre 0707.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.16 Controlador impresion0707

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes reasignados.** | |
| **Controlador:** | **reporteTiqueteR.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 del reporte, por medio del return de la url /reportes/0707.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.17 Controlador reporteTiqueteR

|  |  |
| --- | --- |
| **PANTALLA: Reporte de evaluación del servicio.** | |
| **Controlador:** | **Impresion0708.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de evaluación del servicio, a la vez este método invoca al jasper con el nombre 0708.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de evaluación del servicio, a la vez este método invoca a jasper con el nombre 0708.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.18 Controlador impresion0708

|  |  |
| --- | --- |
| **PANTALLA: Reporte de evaluación del servicio.** | |
| **Controlador:** | **reporteEvalSeervicio.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 del reporte, por medio del return de la url /reportes/0708.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.19 Controlador reporteEvalServicio

|  |  |
| --- | --- |
| **PANTALLA: Reporte consolidad de afluencia de contribuyentes.** | |
| **Controlador:** | **Impresion0709.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte consolidado de afluencia de contribuyentes, a la vez este método invoca al jasper con el nombre 0709.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte consolidado de afluencia de contribuyentes, a la vez este método invoca a jasper con el nombre 0709.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.20 Controlador impresion0709

|  |  |
| --- | --- |
| **PANTALLA: Reporte consolidado de afluencia de contribuyentes.** | |
| **Controlador:** | **reporteConsAfluenciaCont.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 del reporte, por medio del return de la url /reportes/0709.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.21 Controlador reporteConsAfluenciaCont

|  |  |
| --- | --- |
| **PANTALLA: Reporte tiquetes emitidos, atendidos y no atendidos.** | |
| **Controlador:** | **Impresion0710.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte tiquetes emitidos, atendidos y no atendidos, a la vez este método invoca al jasper con el nombre 0710.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte tiquetes emitidos, atendidos y no atendidos, a la vez este método invoca a jasper con el nombre 0710.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.22 Controlador impresion07010

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes emitidos, atendidos y no atendidos.** | |
| **Controlador:** | **reporteTiqueteEmiAteNoAte.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 del reporte, por medio del return de la url /reportes/0710.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.23 Controlador reporteEmiAteNoAte

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de trámites por centros de servicios.** | |
| **Controlador:** | **Impresion0711.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de trámites por centros de servicios, a la vez este método invoca al jasper con el nombre 0711.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de trámites por centros de servicios, a la vez este método invoca a jasper con el nombre 0711.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.24 Controlador impresion07011

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de trámites por centros de servicios.** | |
| **Controlador:** | **reporteComparaTramCS.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 del reporte, por medio del return de la url /reportes/0711.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.25 Controlador reporteComparaTramCS

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de tiempo de trámites por centros de servicios.** | |
| **Controlador:** | **Impresion0712.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de tiempo de trámites por centros de servicios, a la vez este método invoca al jasper con el nombre 0712.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de tiempo de trámites por centros de servicios, a la vez este método invoca a jasper con el nombre 0712.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.26 Controlador impresion07012

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de tiempo de trámites por centros de servicios.** | |
| **Controlador:** | **reporteComparaTiempoTraCS.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 del reporte, por medio del return de la url /reportes/0712.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.27 Controlador reporteComparaTiempoTraCS

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de tiquetes emitidos, atendidos y no atendidos.** | |
| **Controlador:** | **Impresion0713.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de tiquetes emitidos, aendidos y no atendidos, a la vez este método invoca al jasper con el nombre 0713.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de tiquetes emitidos, aendidos y no atendidos, a la vez este método invoca a jasper con el nombre 0713.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.28 Controlador impresion07013

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de tiquetes emitidos, atendidos y no atendidos.** | |
| **Controlador:** | **reporteComparaTiqueEmiAteNoAte.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 del reporte, por medio del return de la url /reportes/0713.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.29 Controlador reporteComparaTiqueEmiAteNoAte

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de afluencia por centro de atención.** | |
| **Controlador:** | **Impresion0714.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de afluencia por centro de atención, a la vez este método invoca al jasper con el nombre 0714.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el comparativo de afluencia por centro de atención, a la vez este método invoca a jasper con el nombre 0714.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.30 Controlador impresion07014

|  |  |
| --- | --- |
| **PANTALLA: Comparativo de afluencia por centro de atención.** | |
| **Controlador:** | **reporteComparaAfluenciaCA.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 del reporte, por medio del return de la url /reportes/0714.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.31 Controlador reporteComparaAfluenciaCA

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiempos de atención por Usuario.** | |
| **Controlador:** | **Impresion0715.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiempos de atención por usuario, a la vez este método invoca al jasper con el nombre 0715.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiempos de atención por usuario, a la vez este método invoca a jasper con el nombre 0715.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.32 Controlador impresion07015

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiempos de atención por usuario.** | |
| **Controlador:** | **reporteTiempoAten.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 del reporte, por medio del return de la url /reportes/0715.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.33 Controlador reporteTiempoAten

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes atendidos por Usuario.** | |
| **Controlador:** | **Impresion0716.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes atendidos por usuario, a la vez este método invoca al jasper con el nombre 0716.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de tiquetes atendidos por usuario, a la vez este método invoca a jasper con el nombre 0716.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.34 Controlador impresion07016

|  |  |
| --- | --- |
| **PANTALLA: Reporte de tiquetes atendidos por usuario.** | |
| **Controlador:** | **reporteTiquetesAte.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 del reporte, por medio del return de la url /reportes/0716.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.35 Controlador reporteTiquetesAte

|  |  |
| --- | --- |
| **PANTALLA: Reporte de índice de trámites atendidos por Usuario.** | |
| **Controlador:** | **Impresion0717.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de ídice de trámites atendidos por usuario, a la vez este método invoca al jasper con el nombre 0717.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de índice de trámites atendidos por usuario, a la vez este método invoca a jasper con el nombre 0717.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.36 Controlador impresion07017

|  |  |
| --- | --- |
| **PANTALLA: Reporte de índice de trámites atendidos por usuario.** | |
| **Controlador:** | **reporteIndiceTramit.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 del reporte, por medio del return de la url /reportes/0717.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.37 Controlador reporteIndiceTramit

|  |  |
| --- | --- |
| **PANTALLA: Reporte de actividades diarias del Usuario.** | |
| **Controlador:** | **Impresion0718.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de actividades diarias del usuario, a la vez este método invoca al jasper con el nombre 0718.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de actividades diarias del usuario, a la vez este método invoca a jasper con el nombre 0718.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.38 Controlador impresion07018

|  |  |
| --- | --- |
| **PANTALLA: Reporte de actividades diarias del usuario.** | |
| **Controlador:** | **reporteActDiarias.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 del reporte, por medio del return de la url /reportes/0718.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.39 Controlador reporteActDiarias

|  |  |
| --- | --- |
| **PANTALLA: Reporte general de usuarios.** | |
| **Controlador:** | **Impresion0719.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte general de usuarios, a la vez este método invoca al jasper con el nombre 0719.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte general de usuarios, a la vez este método invoca a jasper con el nombre 0716.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.40 Controlador impresion07019

|  |  |
| --- | --- |
| **PANTALLA: Reporte general de usuarios.** | |
| **Controlador:** | **reporteGralUsuario.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 del reporte, por medio del return de la url /reportes/0719.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.41 Controlador reporteGralUsuario

|  |  |
| --- | --- |
| **PANTALLA: Reporte de pausas, recesos y permisos por usuario.** | |
| **Controlador:** | **Impresion0720.java** |
| **Descripción de Métodos de Negocio** | |
| **@RequestMapping(value = "/pdf", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de pausas, recesos y permisos por usuario, a la vez este método invoca al jasper con el nombre 0720.jasper el cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte en formato pdf.  **@RequestMapping(value = "/html", method = RequestMethod.*GET*)**  **@ResponseBody**  **public void getRpt1(HttpServletResponse response, HttpServletRequest req) throws JRException, IOException, SQLException, ClassNotFoundException:**  Método que recibe los diferentes parámetros utilizados para generar el reporte de pausas, recesos y permisos por usuario, a la vez este método invoca a jasper con el nombre 0720.jasper es cual está ubicado en el directorio /Other Sources/src/main/resources/templates/reportes/jasperreport, que genera el reporte para ser mostrado en pantalla. | |

Tabla 3.42 Controlador impresion07020

|  |  |
| --- | --- |
| **PANTALLA: Reporte de pausas, recesos y permisos por usuario.** | |
| **Controlador:** | **reporteprp.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 del reporte, por medio del return de la url /reportes/0720.html.  **@RequestMapping(value = "/getTramites", method = RequestMethod.*GET*)**  **public String getTramitesXSeccion(HttpServletRequest request,**  **@ModelAttribute MonitoreoTramitePojo monitoreoTramitePojo, final ModelMap model,**  **@RequestParam(value = "servicio", required = false) String servicio) {:**  Método que obtiene el centro de servicio por medio del repositorio GcUnidadRecepRepository, para poder buscar las áreas de atención y los trámites que éste tiene por medio del método getTramitesByService del repositorio gcTramiteRepository. Se obtienen todos los trámites correspondientes a cada área de atención.  **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.  **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 3.43 Controlador reporteprp

# Clases POJO

## 4.1 Listado de clases pojo

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

| Nombre de Opción | Ubicación | Nombre |
| --- | --- | --- |
| Página índex de Reportes | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java |
| Reporte de afluencia de contribuyentes por hora. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de tiempos de espera de atención. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de contribuyentes atendidos por trámites. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de atención de Fila especial. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de confirmación de citas. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de tiquetes emitidos. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de tiquetes reasignados. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de evaluación del servicio. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte consolidado de afluencia de contribuyentes. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte Tiquetes emitidos, atendidos y no atendidos. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Comparativo de trámites por centro de servicios. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Comparativo de tiempo de trámites por centros de servicios. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Comparativo de Tiquetes emitidos, atendidos y no atendidos. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Comparativo de afluencia por centro atención. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de tiempos de atención por Usuario. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de Tiquetes atendidos por Usuario. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de Índice de trámites atendidos por Usuario. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de Actividades Diarias del Usuario. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte general de Usuarios. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |
| Reporte de pausas, recesos y permisos por Usuario. | sv.gob.mh.dgii.colas.pojos.entities | MonitoreoTramitePojo.java  ServicioPojo.java |

Tabla 4.1 Clases Pojo.

## 4.2 Descripción de Clases POJO.

A continuación se proporciona una breve descripción de las clases POJO usadas en las opciones del módulo de Reportes y Estadísticas.

|  |  |
| --- | --- |
| Pantalla: Página índex de Reporte y Estadística | |
| 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 la clases pojo para página index de reportes

|  |  |
| --- | --- |
| Pantalla: Página de todos los reportes del módulo Reporte y Estadística | |
| Pojo | Descripción |
| MonitoreoTramitePojo.java | Esta clase descrita en la tabla 4.2 |
| ServicioPojo.java | Esta clase permite definir las variables escalares para capturar información de las áreas de atención. Las variables que se encientran son las siguientes:   * **private Long nServiciosId;** * **private String sNombre;** * **private String dServicios;** * **private String cUsuarioCrea;** * **private String cUsuarioModi;** * **private Date fiVigencia;** * **private Date ffVigencia;** * **private Date fModifica;** * **private BigInteger nOrden;** * **private short bActiva;** * **private List<GcTramite> gcTramiteList;** * **private List<GcServicios> gcServiciosList;** * **private Long orden;** |

Tabla 4.3 Descripción de la clases pojo para todas las páginas del módulo de reportes y estadística

# Repositorios

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

| Nombre de Opción | Nombre |
| --- | --- |
| Reporte de afluencia de contribuyentes por hora. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de tiempos de espera de atención. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de contribuyentes atendidos por trámites. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de atención de Fila especial. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de confirmación de citas. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de tiquetes emitidos. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de tiquetes reasignados. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de evaluación del servicio. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte consolidado de afluencia de contribuyentes. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte Tiquetes emitidos, atendidos y no atendidos. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Comparativo de trámites por centro de servicios. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Comparativo de tiempo de trámites por centros de servicios. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Comparativo de Tiquetes emitidos, atendidos y no atendidos. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Comparativo de afluencia por centro atención. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de tiempos de atención por Usuario. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de Tiquetes atendidos por Usuario. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de Índice de trámites atendidos por Usuario. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte de Actividades Diarias del Usuario. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |
| Reporte general de Usuarios. | **GcServiciosRepository.java**  **GcTramiteRepository.java**  **GcUnidadRecepRepository.java** |

Tabla 5.1 Listado de clases repositorios

## 5.1 Descripción de Repositorios.

En la siguiente tabla se describen los métodos usados de los repositorios dentro del módulo de Reporte y Estadística.

| 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”. |
| 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”. |

Tabla 5.2 Descripción de Repositorios.

# Componentes

## 6.1 Listado de Componentes

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

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

Tabla 6.1 Listado de Componentes

## 6.2 Descripción de Componentes

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

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

Tabla 6.2 Descripción de Componentes

# Configuración.

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

| Nombre del archivo de configuración | 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

# Documentos Relacionados a la Base de Datos

## 8.1 Tablas

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

|  |  |  |  |
| --- | --- | --- | --- |
| GC\_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 8.1 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 8.2 GC\_SERVICIOS

|  |  |  |  |
| --- | --- | --- | --- |
| 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 8.3 TB\_UNIDAD\_RECEP

# Seguridad

Estos son los roles de seguridad usada en el Módulo de Reportes y Estadística:

|  |  |  |
| --- | --- | --- |
| Opción | URL | Roles |
| -- | /reportes/home/ | ROLE\_GC\_R1 |
| Reporte de afluencia de contribuyentes por hora. | /reportes/0701 | ROLE\_GC\_R1 |
| Reporte de tiempos de espera de atención. | /reportes/0702 | ROLE\_GC\_R1 |
| Reporte de contribuyentes atendidos por trámites. | /reportes/0703 | ROLE\_GC\_R1 |
| Reporte de atención de Fila especial. | /reportes/0704 | ROLE\_GC\_R1 |
| Reporte de confirmación de citas. | /reportes/0705 | ROLE\_GC\_R1 |
| Reporte de tiquetes emitidos. | /reportes/0706 | ROLE\_GC\_R1 |
| Reporte de tiquetes reasignados. | /reportes/0707 | ROLE\_GC\_R1 |
| Reporte de evaluación del servicio. | /reportes/0708 | ROLE\_GC\_R1 |
| Reporte consolidado de afluencia de contribuyentes. | /reportes/0709 | ROLE\_GC\_R1 |
| Reporte Tiquetes emitidos, atendidos y no atendidos. | /reportes/0710 | ROLE\_GC\_R1 |
| Comparativo de trámites por centro de servicios. | /reportes/0711 | ROLE\_GC\_R1 |
| Comparativo de tiempo de trámites por centros de servicios. | /reportes/0712 | ROLE\_GC\_R1 |
| Comparativo de Tiquetes emitidos, atendidos y no atendidos. | /reportes/0713 | ROLE\_GC\_R1 |
| Comparativo de afluencia por centro atención. | /reportes/0714 | ROLE\_GC\_R1 |
| Reporte de tiempos de atención por Usuario. | /reportes/0715 | ROLE\_GC\_R1 |
| Reporte de Tiquetes atendidos por Usuario. | /reportes/0716 | ROLE\_GC\_R1 |
| Reporte de Índice de trámites atendidos por Usuario. | /reportes/0717 | ROLE\_GC\_R1 |
| Reporte de Actividades Diarias del Usuario. | /reportes/0718 | ROLE\_GC\_R1 |
| Reporte general de Usuarios. | /reportes/0719 | ROLE\_GC\_R1 |
| Reporte de pausas, recesos y permisos por Usuario. | /reportes/0720 | ROLE\_GC\_R1 |

Tabla 9.1 Roles de Seguridad

# Anexos

Anexo 1:

