

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE DE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

**SISTEMA DE INFORMACIÓN CLÍNICA DE PACIENTES, EN CASO DE EMERGENCIAS POR
ENFERMEDADES DE RIESGO**

Trabajo de Titulación para optar al Título de
Técnico Universitario en INFORMÁTICA

Alumnos:
Pablo Hernández Fuentes
Johann Román Santis

Profesor Guía:
Catherine Gómez Barrera

RESUMEN.

KEYWORDS: *Emergencia, Enfermedad, Paciente, Consultas, Información Pública, Información Médica.*

Este Trabajo de Título propone una solución, para mejorar las posibilidades de supervivencia de pacientes que sufran enfermedades crónicas y/o degenerativas, ante situaciones de emergencia; desplegando información útil a las personas involucradas durante el rescate o ingreso a sala de urgencias.

Los requerimientos fueron obtenidos mediante entrevistas con estudiantes, académicos y médicos en ejercicio de la Universidad Diego Portales: Se percibió un problema en el tiempo de respuesta frente a estas situaciones y en la desinformación a la que se enfrentan las personas que intentan ayudar al paciente y el equipo médico que se enfrenta a una situación en la que se deben tomar decisiones en las que cualquier información pertinente aumenta las posibilidades de supervivencia del paciente.

Para mejorar esta situación de desinformación, se propone un sistema que permite registrar un paciente y su historial de consultas médicas, para guardar su diagnóstico y tratamientos; y poder proporcionar información de valor ante una situación de riesgo, disponible en línea tanto para el público general, como para médicos y personal de urgencias. Dando la oportunidad de retroalimentación con los usuarios y una constante mejora del sistema; además de beneficiar los estudios demográficos referentes a enfermedades crónicas tratadas, gracias a las distintas consultas estadísticas presentes en el sistema.

El desarrollo del software se basará en el Framework Web Django (Python) con base de datos en SQLite, gestionada a través del ORM de Django. Lo cual permite una reacción mucho más fluida a los cambios tanto en el servidor como en el cliente. Se describe en detalle la situación actual del entorno médico y la entrega de información, el ambiente computacional en el cual es desarrollado el proyecto, y el Modelo de Datos que se diseñó.

Finalmente se describen las principales funcionalidades que permiten dar solución a esta problemática y que son desarrolladas en este trabajo, las cuales son: Visualización de información de paciente (vía formulario), gestión de Paciente, Gestión de Atenciones Ambulatorias, Gestión de Atenciones de Urgencia y Generación de Estadísticas.

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1: Aspectos relevantes del diseño lógico.	3
1.1 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL.	4
1.2 PROBLEMAS DETECTADOS.	5
1.3 DESCRIPCIÓN DEL SISTEMA PROPUESTO.	6
1.3.1 Objetivos.	6
1.3.2 Beneficios.	7
1.3.3 Descripción general de la solución propuesta.	7
1.4 DIAGRAMA DE FLUJO ADMINISTRATIVO.	8
1.5 ESTRUCTURA FUNCIONAL DEL SISTEMA.	9
1.6 DESCRIPCIÓN DE FÓRMULAS Y CÁLCULOS EMPLEADOS.	11
1.7 INFORMACIÓN QUE SE MANEJARÁ.	12
1.7.1 Entradas.	12
1.7.2 Salidas.	13
1.7.3 Entidades de información.	14
1.7.4 Estructura de código.	15
1.7.5 Condicionantes de Diseño.	16
CAPÍTULO 2: Medio ambiente computacional y descripción de archivos.	17
2.1 DESCRIPCIÓN DE LAS CARACTERÍSTICAS PROPIAS DEL RECURSO COMPUTACIONAL.	18
2.1.1 Configuración del sistema.	18
2.1.2 Software utilizado.	18
2.2 DESCRIPCIÓN DE ARCHIVOS.	19
CAPÍTULO 3: Descripción de programas.	29
3.1 DIAGRAMA MODULAR.	30
3.2 DIAGRAMA DE MENÚS.	31
3.3 TABLA DE PROGRAMAS DESARROLLADOS.	32
3.4 DESCRIPCIÓN DE PROGRAMAS DESARROLLADOS.	33
3.4.1 Información de paciente (formulario).	33
3.4.2 Gestor de Paciente.	35
3.4.3 Gestor de Atención Ambulatoria.	39
3.4.4 Gestor de Atención de Urgencia.	41
3.4.5 Generador de Estadísticas.	43
CONCLUSIONES.	45
BIBLIOGRAFÍA.	47

ANEXOS.	49
VALIDADORES REGEX.	49
INFORMACIÓN DE PACIENTE (VÍA FORMULARIO).	50
a) Información de paciente vía formulario, Vista.	50
b) Información de Paciente vía Formulario, Formulario.	50
GESTOR DE PACIENTE.	51
a) Registrar Paciente, Vista.	51
b) Registrar paciente, formulario.	52
c) Consultar Paciente, Vista.	54
d) Consultar Paciente, Formulario.	54
e) Modificar Paciente, Vista.	55
f) Modificar Paciente, Formulario.	57
GESTOR DE ATENCIÓN AMBULATORIA.	59
a) Ingresar Atención Ambulatoria, Vista.	59
b) Ingresar Atención Ambulatoria, Formulario.	60
c) Consultar Atención Ambulatoria, Vista.	62
GESTOR DE ATENCIÓN DE URGENCIA.	63
a) Ingresar Atención de Urgencia, Vista.	63
b) Ingresar Atención de Emergencia, Formulario.	65
c) Consultar Atención de emergencia, Vista.	67
GENERADOR DE ESTADÍSTICAS.	67
a) Generador de estadísticas, Vista.	67
b) Generador de estadísticas, Formulario.	69

INDICE DE FIGURAS.

Fig. 1-1 Diagrama de flujo administrativo	8
Fig. 1-2 Formula de promedio de edad en una comuna	11
Fig. 1-3Modelo Relacional de Datos	14
Fig. 3-1 Diagrama Modular	30
Fig. 3-2 Diagrama de Menús	31
Fig. 3-3 Información de paciente vía Formulario (Público), pantalla de Inicio.	34
Fig. 3-4 Información de paciente vía Formulario (Público), pantalla de Información.	34
Fig. 3-5 Información de paciente vía Formulario (Autenticado), pantalla de Inicio.	34
Fig. 3-6 Información de paciente vía Formulario (Autenticado), pantalla de Información.	35
Fig. 3-7 Gestor de paciente, Ingresar Paciente.	36
Fig. 3-8 Gestor de paciente, Consulta de Información.	37
Fig. 3-9 Gestor de paciente, Modificar Información.	37
Fig. 3-10 Gestor de paciente, Consulta de Información – Personal de Urgencias.	38
Fig. 3-11 Gestor de Atención Ambulatoria, Ingresar Atención.	40
Fig. 3-12 Gestor de Atención Ambulatoria, Consultar Atenciones.	40
Fig. 3-13 Gestor de Atención Urgencia, Ingresar Atención Urgencia.	42
Fig. 3-14 Gestor de Atención Urgencia, Consultar Atenciones de Urgencia.	42
Fig. 3-15 Selección de filtros, Generador de estadísticas.	44
Fig. 3-16 Despliegue de estadísticas, Generador de estadísticas.	44

INDICE DE TABLAS.

Tabla 2-1, Médico.	19
Tabla 2-2, Paciente.	20
Tabla 2-3, Establecimiento.	21
Tabla 2-4, Medico Establecimiento.	21
Tabla 2-5, Enfermedad.	22
Tabla 2-6, Solicitud enfermedad.	23
Tabla 2-7, Administrador.	24
Tabla 2-8, Atención Urgencia de Paciente.	25
Tabla 2-9, Atención Ambulatoria de Paciente.	26
Tabla 2-10, Tickets.	27
Tabla 2-11, Usuarios.	28
Tabla 3-1, Programas Desarrollados.	32

SIMBOLOGÍA, SIGLAS, ABREVIATURAS.

EAR: Enfermedad de Alto Riesgo, enfermedades que tengan la característica de inducir un estado de pérdida de conciencia, capacidad de discernimiento o ponga en peligro a las personas que están cerca del paciente.

Observaciones clínicas: Indicaciones proporcionadas por un profesional de la salud -Médico- a cargo del tratamiento de la persona.

Observación pública: Indicaciones proporcionadas por un profesional de la salud -Médico- orientadas a informar a una persona que no tiene conocimientos médicos a socorrer a un paciente con EAR en una situación de emergencia lejos de un Centro de Salud, proporcionando información sin llegar a transgredir la privacidad del paciente.

Situación de emergencia: Cualquier trastorno, cuadro, ataque, desmayo o condición que imposibilite al paciente EAR comunicarse, identificarse o tomar decisiones.

Enfermedad WHO: Enfermedad tipificada y descrita por la Organización Mundial de la Salud.

UDP: Universidad Diego Portales.

IFMSA: International Federation of Medical Students.

MVC: Modelo-Vista-Controlador (Arquitectura de software).

ORM: Mapeo Objeto-Relacional.

INTRODUCCIÓN

Desde que se tiene registro, la humanidad ha tenido que lidiar con enfermedades; no hasta hace mucho contraer un simple resfrío podía significar la muerte. A finales del siglo XIX, los comienzos de la medicina moderna eran poco esperanzadores, se seguía con la visión “Curar pocas veces, aliviar a menudo, consolar siempre” y durante el siglo XX, junto con los diversos avances científicos, la medicina se consolida como una ciencia; una disciplina basada en la evidencia. Esto permitió que los avances ocurrieran a pasos agigantados y en los tiempos que corren se puede contar con diagnósticos precisos y al alcance de una visita a un centro médico.

Pese a esto, muchas enfermedades no dan tregua y cualquier persona puede estar en peligro constante, siendo una preocupación el monitoreo y el acceso a información de los pacientes. Este trabajo tiene por objetivo aportar una solución a la falta de información al momento de socorrer a una persona que haya sido víctima de una situación que haya puesto en peligro su vida o integridad debido a una enfermedad. Para poder lograr esto se desarrollará un sistema de información que permita almacenar información sobre pacientes y sus diferentes consultas clínicas, comentarios proporcionados por los médicos tratantes según la enfermedad consultada y la posibilidad de consultar estos datos durante una situación de peligro para el paciente; todo esto desarrollado con una base de datos y sistema alojados en un servidor y disponibles en línea a disposición del Público, Médicos y Personal de Urgencias.

CAPÍTULO 1:
Aspectos relevantes del diseño lógico.

1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO.

1.1 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL.

Las enfermedades son parte de la vida diaria del ser humano, en algunos casos no influyen mucho en la vida de una persona común y corriente; como pasa con los resfriados, alergias leves y gripes estacionales. En otros casos se vuelve un problema a gran escala que continuamente amenaza su vida: Enfermedades mentales como el Alzheimer, Demencia y Esquizofrenia, Enfermedades Cardiovasculares en general, e incluso enfermedades degenerativas como los males de Huntington o Parkinson pueden aparecer. Este tipo de enfermedades son la antesala de una situación que puede comprometer la vida del afectado, para cualquier persona que lea esto no debe ser desconocida la situación de ver a una persona desmayada, con fuertes convulsiones o una actitud evidentemente errática que necesita ayuda, y no saber qué hacer para ayudar o incluso tener miedo de hacer algo y empeorar la situación.

Teniendo en cuenta esa situación, y considerando a una persona en riesgo vital o que comprometa gravemente su calidad de vida; una vez que se identifica a esta persona -o paciente- lo primero que se piensa es “¿Cómo ayudarla?”, lo más acertado es intentar darle los primeros auxilios, al mismo tiempo que se llame una ambulancia o se traslade directamente a un Centro de Salud. Según el “Manual Administrativo para Servicios de Atención Primaria de Urgencia” del Ministerio de Salud, cuando una ambulancia es solicitada:

- El médico de turno debe dar la indicación del traslado del paciente al centro de derivación correspondiente, mediante una hoja de traslado.
- En caso de que el móvil sea requerido por una urgencia en la cual el paciente no puede acceder al SAPU, será el médico de turno quién decidirá la concurrencia de la ambulancia al sitio del suceso. En aquellos SAPU que están conectados a la red de emergencia prehospitolaria (SAMU), será el médico regulador quien tomará la decisión. El paciente debe ser derivado al SAPU para su evaluación, excepto si es de riesgo vital inminente, caso en que el Médico Regulador decidirá su destino.
- Se debe informar previamente al establecimiento de derivación el traslado del paciente, especialmente si se trata de un paciente crítico.
- Se debe acompañar un informe de la condición del paciente, hipótesis diagnóstica, signos vitales, tratamiento y procedimientos realizados (hoja de traslado).
- Se debe registrar la atención del paciente, la hora de llegada al SAPU y de salida al centro de derivación.
- Se debe registrar el resultado de la derivación: recepción en urgencia, retorno al SAPU o a domicilio.
- La ambulancia no hará salidas innecesarias y sin justificación. Su uso será exclusivo para el traslado de pacientes, salvo autorización específica del jefe de turno.
- Cualquier irregularidad que se presente en uso del vehículo debe ser corregido a la brevedad posible por el Coordinador(a) e informada por escrito al director del Centro de Salud o a la autoridad responsable.

Este procedimiento aplica para Servicios de Atención Primaria de Urgencia (SAPU), sin embargo, no difiere significativamente para el resto de los centros de salud y sus llamados de emergencia. En ambos casos, al solicitar ayuda a un centro de salud se requiere que la información proporcionada sea lo más específica y completa posible para poder asistir eficazmente al paciente.

Mediante entrevistas a médicos, estudiantes de medicina y profesores de dicha carrera de la Universidad Diego Portales, se descubrió que si bien existen redes de hospitales que comparten información del paciente (historia clínica, participación en protocolos de tratamiento), se limitan a servicios de salud definidos, por ejemplo: Los hospitales de la circunscripción Quillota-Viña del Mar pertenecen al Servicio de Salud del mismo nombre (en total son 29 en todo el país), una persona que se atiende en un determinado Servicio de Salud no tendrá un historial médico en ninguno de los otros 28, y para compartir dicho historial se debe solicitar directamente entre servicios, que se envían por correo o e-mail, dependiendo de si existe física o digitalmente.

En los portales y bases de datos públicas se encontraron bases de datos generales sobre enfermedades y su tipificación, además de un informe de concurrencia a centros de urgencia por año.

1.2 PROBLEMAS DETECTADOS.

- a) Los pacientes de EAR son susceptibles a situaciones de emergencia en la vía pública, sin tener un método con el cual puedan ser asistidos por personas en los alrededores.
- b) Una persona que encuentra a un paciente de EAR en una situación de riesgo, no siempre tendrá conocimiento necesario para proporcionar ayuda.
- c) Los médicos que atienden en sala de urgencias deben ayudar al paciente de EAR sin tener información de la condición previa del paciente (observaciones clínicas).
- d) Un médico que prescribe un tratamiento a su paciente de EAR no tiene acceso a las atenciones de emergencia y atenciones de otros médicos a la que ha sido sometido su paciente.

1.3 DESCRIPCIÓN DEL SISTEMA PROPUESTO.

1.3.1 Objetivos.

1.3.1.1 Objetivo Principal.

Crear un sistema informático que permita acceder a información médica; correspondiente a información entregada por un Médico usuario del sistema, orientada a ser útil en una situación de emergencia sufrida por un paciente inscrito. Y que además permita registrar y consultar las atenciones -tanto consultas ambulatorias como de urgencia- prestadas a cada paciente.

1.3.1.2 Objetivos específicos.

- a) Permitir a cualquier médico que se registre, y considere que su paciente tiene una EAR -y por ende está en peligro permanente de sufrir una situación de emergencia-, ingresar a dicho paciente al sistema indicando sus datos personales, una observación pública en caso de que la situación de emergencia se produzca fuera de un centro de salud y un texto de observaciones clínicas para el personal de urgencias, en caso de que el paciente se encuentre ya en un centro de salud. A continuación, darle una pulsera que contenga: el RUT del paciente y un código QR, que estén asociados a los datos del paciente.
- b) Mostrar una observación pública de manera oportuna a las personas que encuentren a un paciente de EAR en una situación de emergencia, mediante una aplicación web en el navegador ingresando el RUT o escaneando el código QR con la aplicación por defecto que tenga el dispositivo móvil.
- c) Mostrar la observación clínica al personal de urgencias en el centro de salud, de igual manera que en el caso anterior, usando la aplicación web desde un terminal en la sala de urgencias o usando un dispositivo móvil (en este caso preferentemente un tablet) para escanear el código QR.
- d) Registrar las atenciones prestadas en la sala de urgencias y atenciones ambulatorias, para ser consultadas posteriormente por el médico tratante o el personal del hospital cuando el paciente ya haya sido estabilizado.
- e) Generar informes orientados a proporcionar distintos tipos de información relevante para su posterior uso académico o de investigación.
- f) Permitir a los médicos entregar sugerencias y observaciones a través de un formulario en la aplicación web, a modo de retroalimentación para hacer mejoras y corregir errores, esto se definirá como "Tickets" de retroalimentación.

1.3.2 Beneficios.

Los pacientes EAR mejorarán sus posibilidades de sobrevivencia respecto a la situación de emergencia en la vía pública y en sala de urgencia.

Reducción considerable de la demora en la obtención de información para el tratamiento inmediato en situación de emergencia.

Permitirá recopilar datos de ingresos a centros de salud, enfermedades en comunas, regiones, edad y sexo, para uso del Ministerio de Salud o investigación académica.

1.3.3 Descripción general de la solución propuesta.

En un caso de urgencia, se tiene a una persona (Paciente EAR) con incapacidad de poder pedir ayuda o valerse por sí misma, en un lugar lejano a un Centro de Salud. Quien lo encuentra, se percata de la presencia de una pulsera con una página web, un RUT, y un código Qr inscritos y decide consultar la web; iniciando el proceso del sistema propuesto. Una vez en la página, la persona ve un mensaje pidiendo un RUT, e ingresa el que está inscrito en la pulsera; el sistema comprueba que el paciente existe en los registros y despliega en pantalla la información pública que haya dispuesto su médico tratante en una atención previa. Una vez que la información sea leída, se le aplican los cuidados primarios al paciente EAR, y se solicita una ambulancia para su posterior traslado a un centro de salud. Al ingresar a sala de urgencias, el personal inicia sesión en el sistema; el cual verifica el usuario con el fin de desplegar la página web para centros de salud. Posteriormente, se ingresa el RUT del paciente en esta página y el sistema verifica nuevamente su existencia, para desplegar la información médica correspondiente, que haya ingresado el médico tratante. Una vez entregada la información, se trata de urgencia al paciente. Al finalizar los procedimientos de urgencia, el personal designado rellena el “formulario de atención de urgencia” y registra una nueva atención de urgencia, si este paso es exitoso finaliza el proceso.

1.4 DIAGRAMA DE FLUJO ADMINISTRATIVO.

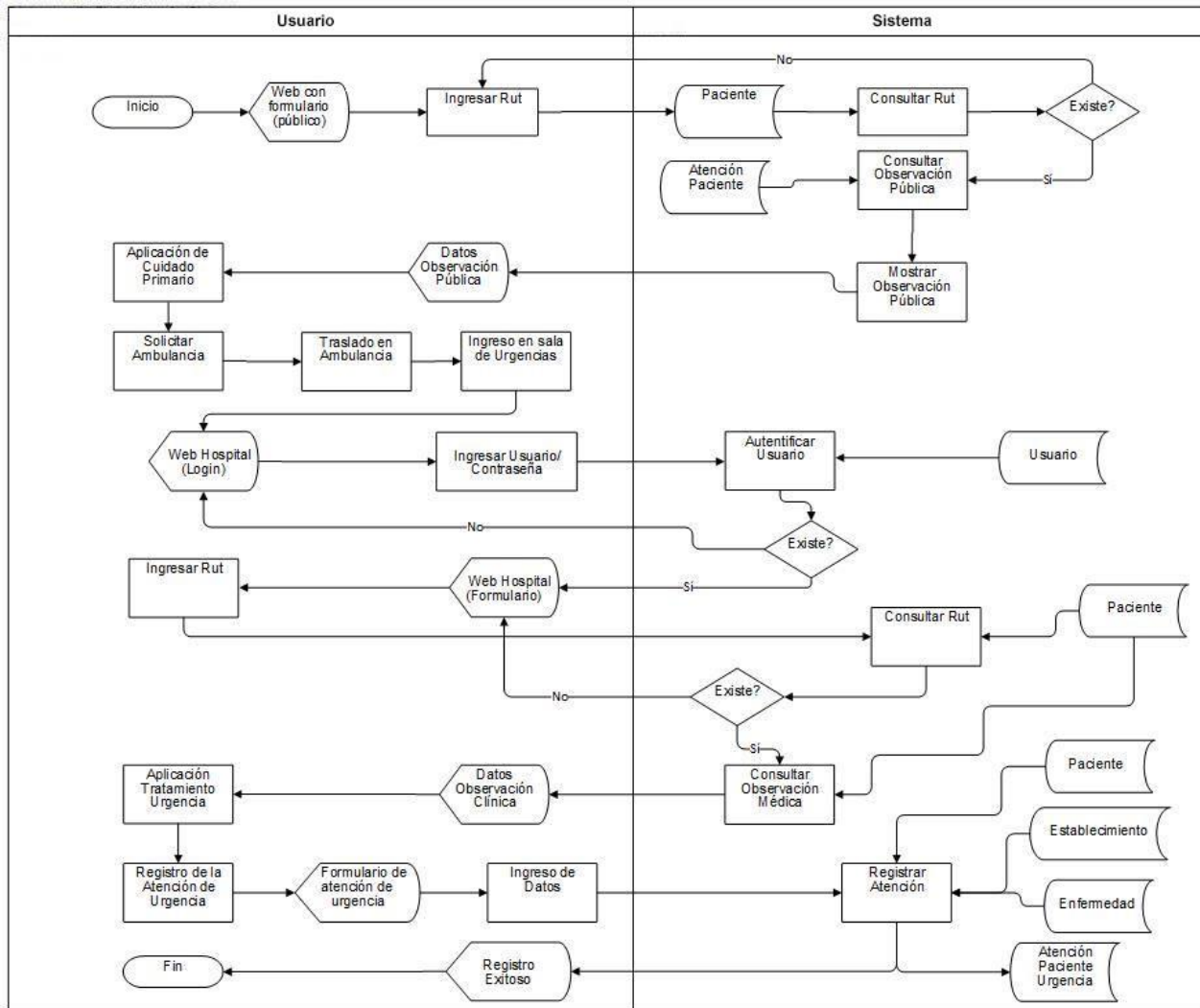


Fig. 1-1 Diagrama de flujo Administrativo.

Fuente: Elaboración propia.

1.5 ESTRUCTURA FUNCIONAL DEL SISTEMA.

A continuación, se presentan las funcionalidades del Sistema, detallando su utilidad:

a) Gestor de Atenciones Médicas: Permite agregar y consultar las atenciones de médicos de cada paciente con el fin de tener un historial médico.

b) Gestor de Atenciones de Emergencia: Permite agregar y consultar las atenciones de emergencia de cada paciente con el fin de tener un historial médico.

c) Gestor de Retroalimentación: Permite agregar y consultar los Tickets de retroalimentación enviados por los médicos.

d) Mantenedor Pacientes: Permite a los Médicos Agregar, Modificar, Consultar y Eliminar a los pacientes en el sistema.

e) Mantenedor de Médicos: Permite Agregar, Modificar, Consultar y Eliminar a médicos que quieran ingresar al sistema como usuarios.

f) Gestor de Enfermedades: Permite Agregar y Consultar enfermedades que se consideren como EAR.

g) Función Consulta de Establecimientos: Permite Consultar centros de salud en una lista de establecimientos, estos datos vienen de la base de datos del MINSAL en formato Excel.

h) Gestor de Solicitudes de enfermedades: Permite agregar y consultar solicitudes de enfermedades.

i) Mantenedor de Administradores: Permite Agregar, Modificar, Consultar y Eliminar administradores para que aprueben o rechacen las solicitudes de agregación de enfermedades.

j) Mantenedor de Usuarios: Permite Agregar, Modificar, Consultar y Eliminar usuarios del sistema, cada uno con determinado perfil.

k) Función de Consulta en caso de Emergencia (Fuera de Hospitales, Usuario Público): Entrega la observación pública, a quien esté asistiendo al paciente en una situación de emergencia lejos de un centro de salud; al haber consultado el RUT indicado en la identificación física del paciente, mediante la aplicación web o escaneado el código QR.

l) Función de Consulta en caso de Emergencia (Dentro de las dependencias del hospital): Entrega la observación clínica, que considera sólo los aspectos más importantes para la atención en sala de Urgencias, según el RUT del paciente o su código QR (con opción de ingresar Rut en caso de que la identificación física esté dañada o perdida).

m) Función de Consulta de Atenciones (uso del médico): Se mostrará por pantalla un historial de atenciones de emergencia y atenciones de médicos para el uso de médicos y hospitales registrados en el sistema.

n) Función de solicitud de ingreso de nuevas enfermedades: A través de otra sección de la aplicación web se recibirán solicitudes de médicos registrados como usuarios en el sistema, para ingresar nuevas enfermedades (posteriormente estudiadas y confirmadas con autoridades competentes)

ñ) Función de confirmación de solicitud: Una vez que una solicitud de enfermedad sea aprobada para su ingreso al sistema (por autoridades competentes, quienes determinarán además el riesgo y clasificación de la enfermedad), se notificará a quien haya realizado la petición mediante mail, el administrador añadirá los campos de riesgo y clasificación de dicha enfermedad.

o) Función de Notificación vía mail a Médico tratante: Si un paciente del sistema es ingresado de urgencia a un Centro Médico, se notificará a su Médico tratante vía mail.

p) Función Generadora de informes: Permitirá llevar los datos recopilados a informes que se pueden generar en la aplicación web con filtros como: región, edad, sexo y estatura.

1.6 DESCRIPCIÓN DE FÓRMULAS Y CÁLCULOS EMPLEADOS.

$$promedio = \frac{\sum EPE}{NTPE}$$

Fig. 1-2 Fórmula de Promedio de Edad en una comuna.

Fuente: Elaboración propia.

Promedio de edad por enfermedad en una comuna determinada:

Donde EPE es la Edad de la Persona Enferma y NTPE es el número total de personas enfermas con la misma enfermedad.

1.7 INFORMACIÓN QUE SE MANEJARÁ.

1.7.1 Entradas.

-Datos de Pacientes (Rut, Nombre, Apellido_Paterno, Apellido_Materno, Direccion, Fecha_nacimiento, Peso, Estatura, Sexo, Fono, Fono_emergencia)

-Datos de Médico (Rut, Nombre, Apellido_Paterno, Apellido_Materno, Fono, Mail, Especialidad, id_usuario, Clave).

-Datos de Administrador (Rut, Nombre, Apellido_Paterno, Apellido_Materno, Mail, id_usuario, Clave).

-Datos de atención de urgencia (Rut_paciente, Rut_encargado_urg, Nombre_encargado_urg, Cargo_encargado_urg, id_who, codigo_establecimiento, Tratamiento, Observacion_urgencia)

-Datos de atención de médico (Rut_paciente, Rut_medico, id_who, codigo_establecimiento, Tratamiento, Observacion_publica, observacion_medica)

-Datos de Solicitud de Ingreso de Enfermedades (Rut_medico, id_who, Observación).

-Datos de Aprobación/rechazo de Ingreso de Enfermedades (Nro_Solicitud, riesgo, clasificación)
En este caso todos los campos menos riesgo y clasificación no serán editables, solo se podrá editar riesgo y clasificación.

-Datos de Ticket de Retroalimentación (Rut_medico, Descripción del Problema/Recomendación)

-Archivo Excel de la base de datos de los establecimientos de salud (MINSAL), al importarse los datos desde este archivo se crearán los usuarios/contraseña de todos los establecimientos, para ser entregados después al director cada establecimiento y que él/ella informe de estas credenciales al personal de urgencia.

1.7.2 Salidas.

-Informe de Paciente en Exteriores (nombre_paciente, observación_publica): Informe de datos de paciente; destinado a ser presentado fuera de centros de salud a un socorrista/transeúnte, ante una situación de emergencia.

-Informe de Paciente en Urgencia (Rut_paciente, Nombre_paciente, observacion_medica, sexo, riesgo, fono_emergencia, edad (derivado de fecha_nacimiento)): Informe de datos de paciente; destinado a ser presentado dentro de centros de salud, al equipo de urgencias, con el fin de agilizar la evaluación y tratamiento del paciente.

-Informe de Atenciones de Emergencia del Paciente: (Rut_paciente, nombre_paciente, fecha_hora_atencion, Rut_encargado_urg, Nombre_encargado_urg, Cargo_encargado_urg, nombre_enfermedad, nombre_establecimiento, Tratamiento, Observacion_urgencia): Informe de datos sobre las atenciones de emergencia prestadas a un paciente, destinado a ser presentado tanto en atenciones ambulatorias como en atenciones de urgencia.

-Informe de Atenciones ambulatorias del Paciente: (Rut_paciente, nombre_paciente, fecha_hora_atencion, rut_medico, nombre_medico, nombre_enfermedad, nombre_establecimiento, observacion_medica, observacion_publica): Informe de datos sobre las atenciones ambulatorias prestadas a un paciente, destinado a ser presentado en atenciones ambulatorias a su médico(s) tratante(s).

-Notificación sobre ingreso de paciente a urgencias, a su médico(s) tratante(s) (nombre_paciente, nombre_establecimiento, nombre_enfermedad, tratamiento, observacion_urgencia): Notificación vía email al médico(s) tratante(s), en el momento que se efectúe una atención de urgencia al paciente.

-Notificación de ticket atendido por administrador (Nro_ticket, Respuesta, fecha_emitido, fecha_respondido(derivado de fecha_emitido)): Notificación vía email, al usuario que haya enviado un ticket de feedback y este haya sido respondido por un administrador.

-Informe de Enfermedades más comunes por región: Informe de conteo decreciente de la cantidad de distintos pacientes tratados por enfermedad, organizados en grupos por región.

-Informe de Enfermedades comunes por temporada: Informe de incidencia de las distintas enfermedades y su aumento en las consultas o emergencias, en distintas etapas del año.

-Informe de Edad promedio por enfermedad y región: Informe de la edad promedio de todos los pacientes dentro de una misma enfermedad y región.

-Informe de Concurrencia a Centros de salud: Informe de la cantidad de consultas tanto de urgencia como de control, agrupados por región.

-Informe de Incidencia de enfermedades por edad y sexo: Informe sobre la cantidad de pacientes agrupados en distintos rangos etarios y su sexo.

1.7.3 Entidades de información.

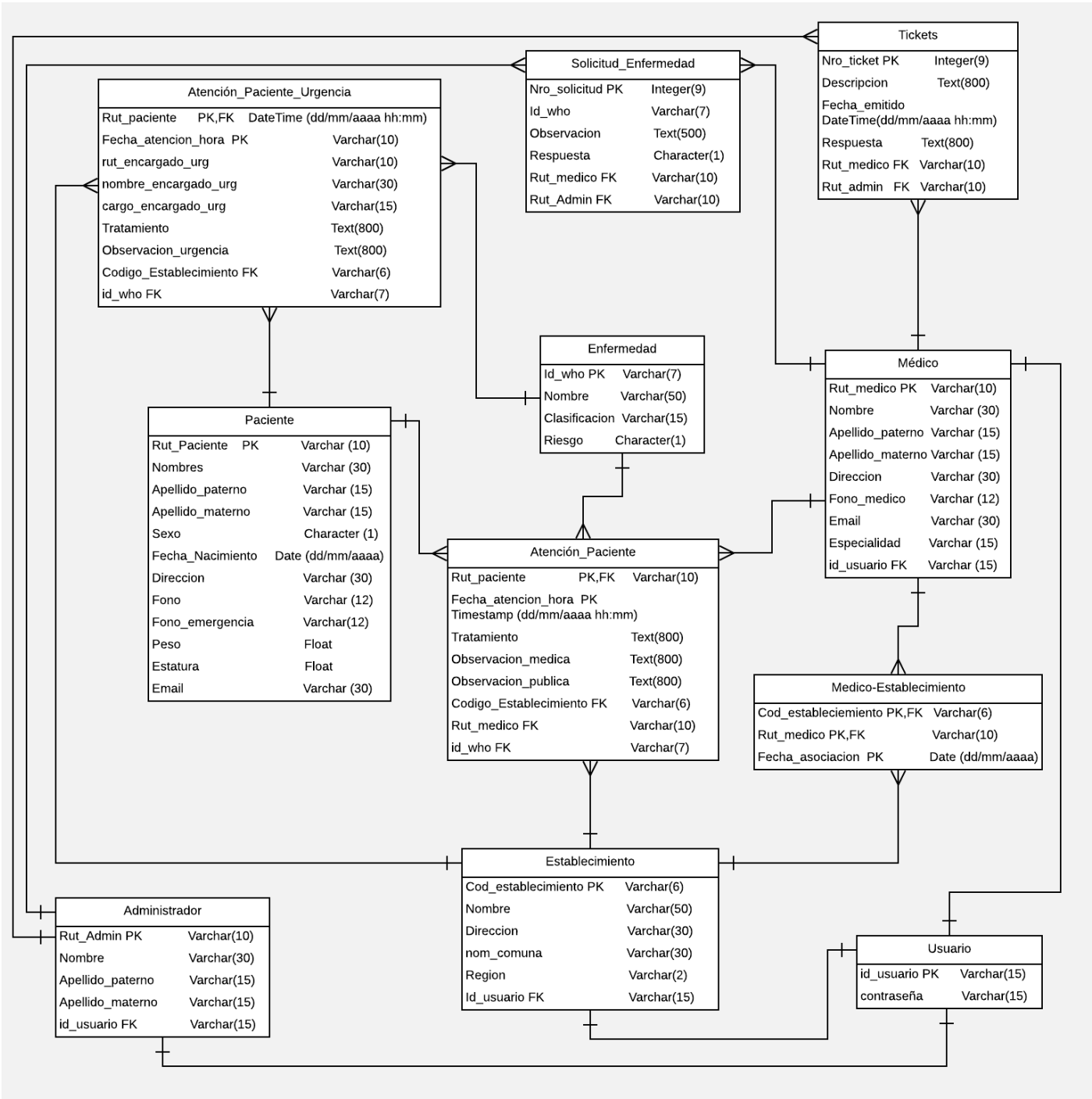


Fig. 1-3 Modelo Relacional de datos.
Fuente: Elaboraci3n propia.

1.7.4 Estructura de código.

a) Códigos Naturales

- Paciente: El RUT es un código natural propio de la persona, por lo que se utilizará como clave primaria para la tabla Paciente.
- Médico: Al igual que el paciente el RUT es código natural de la persona, por lo que se utilizará como clave primaria para la tabla Médico.
- Establecimiento: El Ministerio de Salud tiene una base de datos de todos los centros de salud legales y al día del país, trae una clave única para cada establecimiento, por lo que se usará como clave primaria de la tabla Establecimiento.
- Enfermedad: La WHO (World Health Organization) tiene códigos únicos para cada enfermedad en su base de datos, se usará este código como clave primaria en la tabla Enfermedad.
- Administrador: Al igual que los pacientes y los médicos, las personas que vayan a desempeñarse como administrador de solicitudes tienen un RUT, así que se usará como clave primaria en Administrador.

b) Códigos Creados

- Ticket: Se usará un número correlativo.
- Usuario: Es la combinación de un número correlativo generado al registrar un nuevo usuario, con el nombre de usuario asociado a la cuenta.
- Solicitud de enfermedad: Se usará un número correlativo.
- Atención paciente urgencia: Al combinar el RUT del paciente junto con la fecha/hora de atención, se puede distinguir entre cada atención sin problemas de redundancia, así que se usarán juntas como clave primaria.
- Atención paciente: Al combinar el RUT del paciente junto con la fecha/hora de atención, se puede distinguir entre cada atención sin problemas de redundancia, así que se usarán juntas como clave primaria.

1.7.5 Condicionantes de Diseño.

Lenguajes de programación a utilizar: Web FrontEnd (HTML,CSS,Javascript), Web BackEnd Django (Python), responsiva para la mayoría de los dispositivos móviles y de escritorio.

Base de Datos: Oracle

Se tiene en cuenta que la privacidad y confidencialidad de los pacientes, y sus implicaciones legales podrían restringir el uso de este sistema, se tiene contemplado alguna forma de documento legal firmado por parte del paciente para que autorice la manipulación de sus datos clínicos.

Al ser un sistema en tiempo real se prevé un requerimiento fuerte en materia de servidores y disponibilidad de la red en caso de que los tiempos de respuesta sean altos para el estándar de tratamiento de urgencias (peor caso posible) con servidores externos o servicios de cloud computing.

Respecto a la seguridad en redes, se tiene contemplado un sistema de tunneling o de encriptación de los paquetes en caso de que la seguridad en la capa de transporte no sea suficiente, en el caso de la consulta en sala de urgencias se pretende hacer un login que dure 1 día y que solo pueda ser accedido a través de una IP determinada (la del establecimiento).

CAPÍTULO 2:
***Medio ambiente computacional y descripción de
archivos.***

2 MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS.

2.1 DESCRIPCIÓN DE LAS CARACTERÍSTICAS PROPIAS DEL RECURSO COMPUTACIONAL.

2.1.1 Configuración del sistema

-Equipo de Desarrollo 1:

- PC de escritorio.
- Tipo de procesador: Intel i5 4460, 3,2 GHz.
- Almacenamiento primario: Kingston HyperX Fury Black 8GB.
- Almacenamiento secundario: Western Digital blue 1 Tb.
- LAN Ethernet 1Gbps.
- Pantalla de 24 pulgadas LCD, 1920x1080px, extendida con 20 pulgadas LCD, 1600x900 px.

-Equipo de Desarrollo 2:

- PC de escritorio.
- Tipo de procesador: Intel i5 650, 3,2 Ghz.
- Almacenamiento primario: Corsair Dual-Channel DDR3 8GB.
- Almacenamiento secundario: Western Digital Blue 1Tb.
- LAN Ethernet 1Gbps.
- Pantalla única de 22 pulgadas LCD, 1680x1050 px.

-Requisitos mínimos del cliente:

- Navegador web y conexión a internet activa.

-Servidor:

- Tipo de procesador: Intel® Xeon® Processor E7 v2 Family 2.80 GHz.
- Almacenamiento primario: Kingston HyperX Fury Black 64GB.
- Almacenamiento secundario: Storage Dell PowerEdge 2900 10TB.
- LAN Ethernet Fibra Óptica 10Gbps.

2.1.2 Software utilizado.

-Desarrollador 1:

- Sistema operativo: Windows 7.
- Herramientas de desarrollo de software: MySQL/phpMyAdmin y Sublime Text 2.
- Lenguajes de comandos: PowerShell de Windows.

-Desarrollador 2:

- Sistema operativo: Windows 10.
- Herramientas de desarrollo de software: Django 1.11.2 (Python 3.6) y virtualenv.
- Lenguajes de comandos: PowerShell de Windows.

2.2 DESCRIPCIÓN DE ARCHIVOS.

1) Tabla Médicos:

- Nombre Lógico: médico.
- Nombre Físico: medico.
- Descripción: Esta tabla almacena los datos propios de cada médico.
- Clave Primaria: Rut_medico.
- Clave Foránea: Id_usuario (referencia a tabla Usuario).

Nombre	Tipo de Dato	Longitud	Descripción
Rut_medico	varchar2	10	Identificador único de médico (Rol único tributario)
Nombres	varchar2	30	Nombres de médico
Apellido_paterno	varchar2	15	Apellido paterno del médico
Apellido_materno	varchar2	15	Apellido materno del médico
Direccion	varchar2	30	Dirección de la consulta (u oficina) del médico
Fono	varchar2	12	Teléfono de contacto médico
Email	varchar2	30	Email de contacto médico
Especialidad	varchar2	15	Área de especialidad del médico
Id_usuario	varchar2	15	Identificador de la cuenta de usuario asociada al médico

Tabla 2-1, Médico.

2) Tabla Pacientes:

- Nombre Lógico: Paciente.
- Nombre Físico: Paciente.
- Descripción: Esta tabla almacena los datos propios del paciente.
- Clave Primaria: Rut_paciente.
- Clave Foránea: Ninguna.

Nombre	Tipo de Dato	Longitud	Descripción
Rut_paciente	varchar2	10	Identificador único paciente (Rol único tributario)
Nombres	varchar2	30	Nombres del paciente
Apellido_paterno	varchar2	15	Apellido paterno del paciente
Apellido_materno	varchar2	15	Apellido materno del paciente
Sexo	char	1	Sexo del paciente (Femenino o Masculino), valores “F” ó “M”
Fecha_nacimiento	date	10	Día/Mes/Año de nacimiento
Direccion	varchar2	30	Dirección particular del paciente
Fono	varchar2	12	Teléfono particular del paciente
Fono_emergencia	varchar2	12	Teléfono de un familiar o persona responsable del paciente
Peso	float	4	Peso del paciente en Kilogramos
Estatura	float	4	Estatura del paciente en Centímetros
Email	varchar2	30	Email de contacto Paciente

Tabla 2-2, Paciente.

3) Tabla Establecimiento:

- Nombre Lógico: Establecimiento.
- Nombre Físico: Establecimiento.
- Descripción: Tabla que almacena los datos de los distintos centros de salud en el país.
- Clave Primaria: Cod_establecimiento.
- Clave Foránea: Id_usuario (referencia a tabla Usuario).

Nombre	Tipo de Dato	Longitud	Descripción
Cod_establecimiento	varchar2	6	Identificador único del establecimiento inscrito en el servicio de salud
Nombre	varchar2	50	Nombre del establecimiento
Direccion	varchar2	30	Dirección del establecimiento
Nom_comuna	varchar2	30	Comuna en la que se encuentra el establecimiento
Region	varchar2	2	Región donde se encuentra el establecimiento, valores I a XV o RM
Id_usuario	varchar2	15	Identificador de la cuenta de usuario asociada al establecimiento

Tabla 2-3, Establecimiento.

4) Tabla Médicos-Establecimientos:

- Nombre Lógico: Medico_Establecimiento.
- Nombre Físico: Medico_Establecimiento.
- Descripción: Tabla para la asociación de un médico con un establecimiento.
- Clave Primaria: Cod_establecimiento+Rut_medico+Fecha_asociacion.
- Clave Foránea: Cod_establecimiento (referencia a tabla Establecimiento), Rut_medico (referencia a tabla Medico).

Nombre	Tipo de Dato	Longitud	Descripción
Cod_establecimiento	varchar2	6	Identificador único del establecimiento inscrito en el servicio de salud
Rut_medico	varchar2	10	Identificador único médico (Rol único tributario)
Fecha_asociacion	date	10	Fecha en la que se lleva a cabo la asociación

Tabla 2-4, Establecimiento.

5) Tabla Enfermedades:

- Nombre Lógico: Enfermedad.
- Nombre Físico: Enfermedad.
- Descripción: Tabla que contiene los datos propios de cada enfermedad.
- Clave Primaria: Id_who.
- Clave Foránea: Ninguna.

Nombre	Tipo de Dato	Longitud	Descripción
Id_who	varchar2	7	Identificador único de enfermedad provisto por la WHO (World Health Organization)
Nombre	varchar2	50	Nombre formal de la enfermedad
Clasificacion	varchar2	15	Taxonomía de la enfermedad, por ejemplo, cardiovascular, intestinal, gástrica, neuronal, entre otros
Riesgo	char	1	En escala de 1 a 5, siendo 1 riesgo controlado, 2 riesgo leve,3 riesgo moderado,4 riesgo alto y 5 riesgo urgente

Tabla 2-5, Enfermedad.

6) Tabla Solicitudes de Enfermedad:

- Nombre Lógico: Solicitud enfermedad.
- Nombre Físico: Solicitud_enfermedad.
- Descripción: Tabla que almacena las solicitudes de médicos para ingresar una enfermedad nueva al sistema.
- Clave Primaria: Nro_solicitud.
- Clave Foránea: Rut_admin (referencia a tabla Administrador), Rut_medico (referencia a la tabla Medico).

Nombre	Tipo de Dato	Longitud	Descripción
Nro_solicitud	number	9	Identificador correlativo único
Id_who	varchar2	7	Identificador único de enfermedad provisto por la WHO (World Health Organization)
Observación	varchar2	500	Consideraciones del médico que propone incluir la enfermedad
Respuesta	char	1	Estado de la respuesta: 1: aprobado, 2: en proceso o 3: rechazado
Rut_admin	varchar2	10	Identificador único administrador (Rol único tributario)
Rut_medico	varchar2	10	Identificador único médico (Rol único tributario)

Tabla 2-6, Enfermedad.

7) Tabla Administradores:

- Nombre Lógico: Administrador.
- Nombre Físico: Administrador.
- Descripción: Tabla que contiene los datos propios de un administrador de sistema.
- Clave Primaria: Rut_admin.
- Clave Foránea: Id_usuario (referencia a la tabla Usuario).

Nombre	Tipo de Dato	Longitud	Descripción
Rut_admin	varchar2	10	Identificador único administrador (Rol único tributario)
Nombres	varchar2	30	Nombres del administrador
Apellido_paterno	varchar2	15	Apellido paterno del administrador
Apellido_materno	varchar2	15	Apellido materno del administrador
Id_usuario	varchar2	15	Identificador de la cuenta de usuario asociada al administrador

Tabla 2-7, Administrador.

8) Tabla Atenciones de Urgencia:

- Nombre Lógico: Atención Urgencia de Paciente.
- Nombre Físico: Atencion_paciente_urgencia.
- Descripción: En esta tabla se almacenan atenciones de urgencia provistas por establecimientos a los pacientes, incluyendo al encargado de la atención, observaciones de la atención y tratamientos provistos.
- Clave Primaria: Rut_paciente+Fecha_hora_atencion.
- Clave Foránea: Cod_establecimiento (referencia a la tabla Establecimiento), Id_who (referencia a la tabla Enfermedad), Rut_paciente (que referencia a la tabla Paciente).

Nombre	Tipo de Dato	Longitud	Descripción
Fecha_hora_atencion	timestamp	16	Fecha y hora de la atención prestada
Rut_paciente	varchar2	10	Identificador único del paciente (Rol único tributario)
Rut_encargado_urg	varchar2	10	Identificador único del encargado de la atención (Rol único tributario)
Nombre_encargado_urg	varchar2	30	Nombre completo del encargado
Cargo_encargado_urg	varchar2	15	Cargo del encargado de la atención
Tratamiento	varchar2	800	Tratamientos que se le administraron al paciente
Observacion_urgencia	varchar2	800	Observaciones o apuntes del encargado con respecto a la atención
Codigo_establecimiento	varchar2	6	Identificador único del establecimiento inscrito en el servicio de salud
Id_who	varchar2	7	Identificador único de enfermedad provisto por la WHO (World Health Organization)

Tabla 2-8, Atención Paciente Urgencia.

9) Tabla Atenciones ambulatorias de paciente:

- Nombre Lógico: Atención Ambulatoria de Paciente.
- Nombre Físico: Atencion_paciente.
- Descripción: En esta tabla se almacenan las atenciones ambulatorias que recibe un paciente de un médico determinado.
- Clave Primaria: Rut_paciente+fecha_hora_atencion.
- Clave Foránea: Rut_medico (referencia a la tabla Medico), Rut_paciente (referencia a la tabla Pacientes), Id_who (referencia a la tabla Enfermedad),Codigo_Establecimiento (referencia a tabla Establecimiento).

Nombre	Tipo de Dato	Longitud	Descripción
Fecha_hora_atencion	Timestamp	16	Fecha y hora de la atención prestada
Rut_paciente	varchar2	10	Identificador único del paciente (Rol único tributario)
Tratamiento	varchar2	800	Tratamientos que se le administraron al paciente
Observacion_medica	varchar2	800	Observaciones o apuntes del médico con respecto a la condición fisiológica del paciente en términos médicos (destinado al personal de urgencias)
Observacion_publica	varchar2	800	Indicaciones del médico sobre cuidados y auxilios para el paciente, puesto en palabras simples (destinado al público general)
Codigo_establecimiento	varchar2	6	Identificador único del establecimiento inscrito en el servicio de salud
Rut_medico	varchar2	10	Identificador único del médico (Rol único tributario)
Id_who	varchar2	7	Identificador único de enfermedad provisto por la WHO (World Health Organization)

Tabla 2-9, Atención Ambulatoria de Paciente.

10) Tabla Tickets:

- Nombre Lógico: Tickets.
- Nombre Físico: Tickets.
- Descripción: En esta tabla se almacenan los tickets de feedback que envíen los médicos, con tal de mejorar el servicio o sistema y/o aportar ideas nuevas.
- Clave Primaria: Nro_ticket.
- Clave Foránea: Rut_medico (referencia a la tabla Medico), Rut_admin (referencia a la tabla Administrador).

Nombre	Tipo de Dato	Longitud	Descripción
Nro_ticket	number	9	Identificador correlativo único
Descripcion	varchar2	800	Observaciones / sugerencias del médico respecto al funcionamiento del sistema
Fecha_emitido	timestamp	16	Fecha en la que se emitió el ticket
Respuesta	varchar2	800	Respuesta del administrador
Rut_medico	varchar2	10	Identificador único del médico (Rol único tributario)
Rut_admin	varchar2	10	Identificador único administrador (Rol único tributario)

Tabla 2-10, Tickets.

11) Tabla Usuarios:

- Nombre Lógico: Usuario.
- Nombre Físico: Usuario.
- Descripción: En esta tabla se almacenan los distintos usuarios que existen en el sistema.
- Clave Primaria: Id_usuario.

Nombre	Tipo de Dato	Longitud	Descripción
Id_usuario	varchar2	15	Identificador único de una cuenta de usuario
Contraseña	varchar2	15	Contraseña para el acceso de la cuenta

Tabla 2-11, Usuarios.

CAPÍTULO 3:
Descripción de programas.

3 DESCRIPCIÓN DE PROGRAMAS.

3.1 DIAGRAMA MODULAR.

A continuación, se presentan los distintos módulos del sistema y sus funcionalidades correspondientes

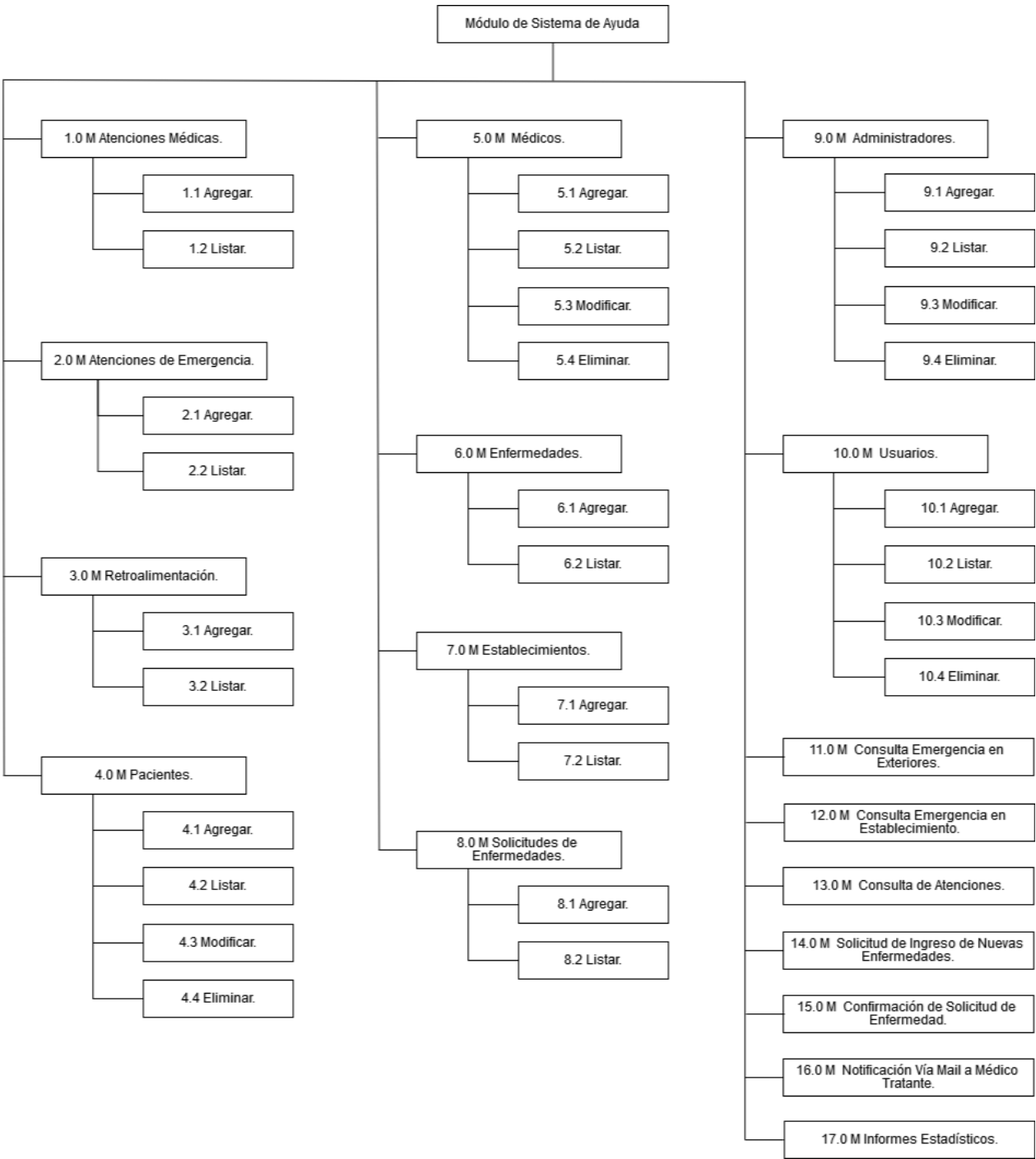


Fig. 3-1 Diagrama Modular.
Fuente: Elaboración propia.

3.2 DIAGRAMA DE MENÚS.

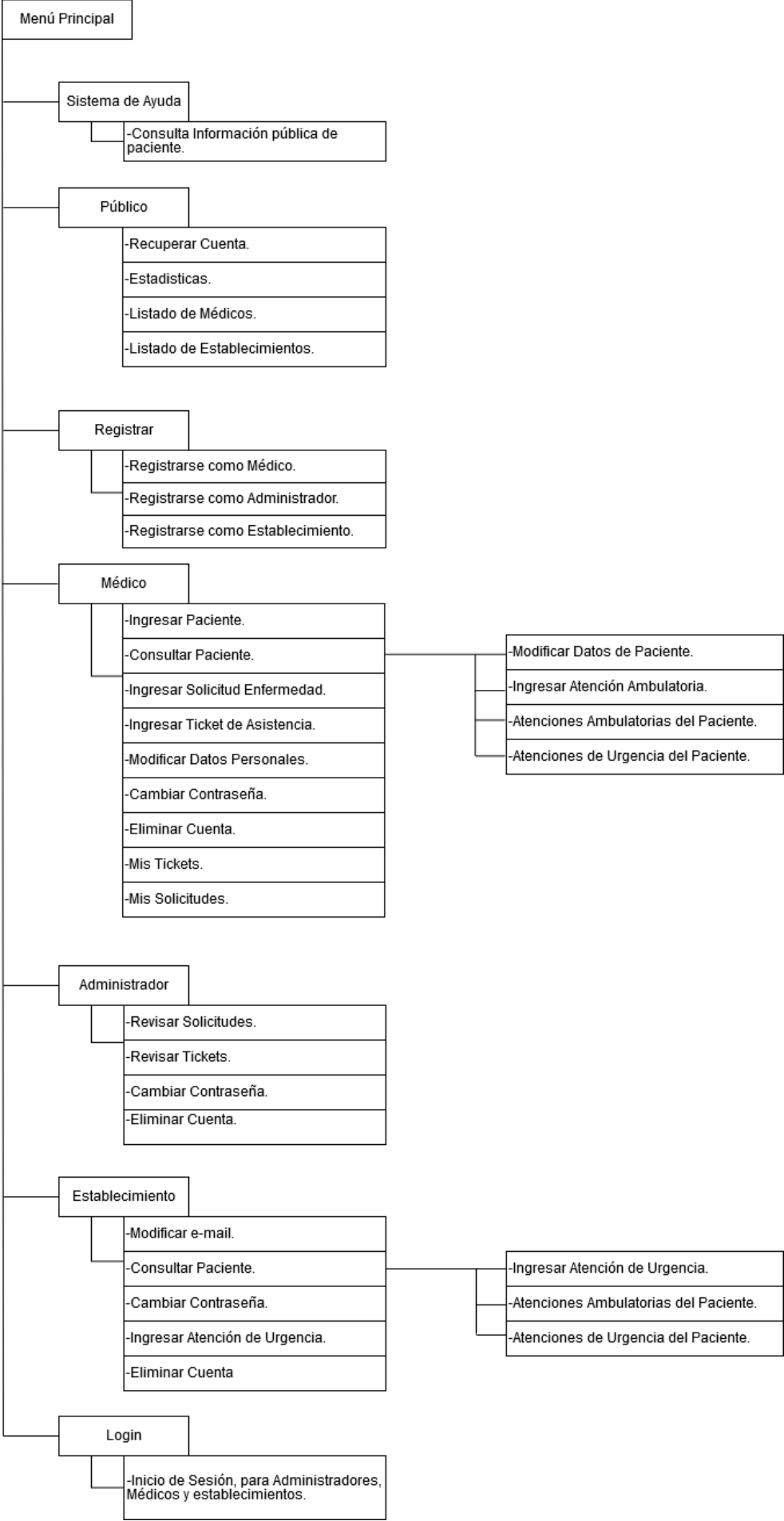


Fig. 3-2 Diagrama de Menús.
Fuente: Elaboración propia.

3.3 TABLA DE PROGRAMAS DESARROLLADOS.

Nombre	Objetivo
Menú Principal	Permite acceder a las funcionalidades del software dependiendo si el usuario está o no autenticado y que privilegios posee.
Inicio de Sesión	Provee la vista para que el usuario se autentifique.
Información de Paciente (vía URL)	Despliega la información clínica o pública del paciente (dependiendo de si el usuario está autenticado o no) directamente a través de la petición de una URL.
Información de paciente (vía formulario) *	Despliega la información clínica o pública del paciente (dependiendo de si el usuario está autenticado o no) directamente a través del envío de formulario en menú principal.
Mantenedor de Médico	Permite al usuario realizar las funciones CRUD sobre un Médico en específico.
Gestor de Paciente*	Permite al usuario Ingresar Pacientes al Sistema, hacer consultas a sus datos, o actualizar su información.
Mantenedor de Establecimiento	Permite al usuario realizar las funciones CRUD sobre un Establecimiento en específico.
Mantenedor de Administrador	Permite al usuario realizar las funciones CRUD sobre un Administrador en específico.
Gestor de Usuario	Permite la creación, consulta y modificación de los distintos usuarios del sistema.
Gestor de Enfermedad	Permite añadir y consultar enfermedades a la lista de afecciones tratadas en el sistema.
Gestor de Solicitud de Enfermedad	Permite al usuario ingresar, consultar y modificar solicitudes de ingreso de enfermedades al sistema.
Gestor de Atención Ambulatoria*	Permite a los Médicos registrados en el sistema, crear y consultar las atenciones ambulatorias prestadas a cada paciente del sistema.
Gestor de Atención de Urgencia*	Permite al personal de urgencia de los establecimientos registrados en el sistema, crear y consultar las atenciones de urgencia prestadas a cada paciente del sistema.
Gestor de Ticket	Permite a los Usuarios ingresar, consultar y modificar los diversos tickets de retroalimentación del sistema.
Generador de Estadísticas*	Permite al usuario solicitar datos estadísticos por región y enfermedad

Tabla 3-1, Programas Desarrollados.

Los diagramas destacados con ‘**’ se describirán detalladamente a continuación:

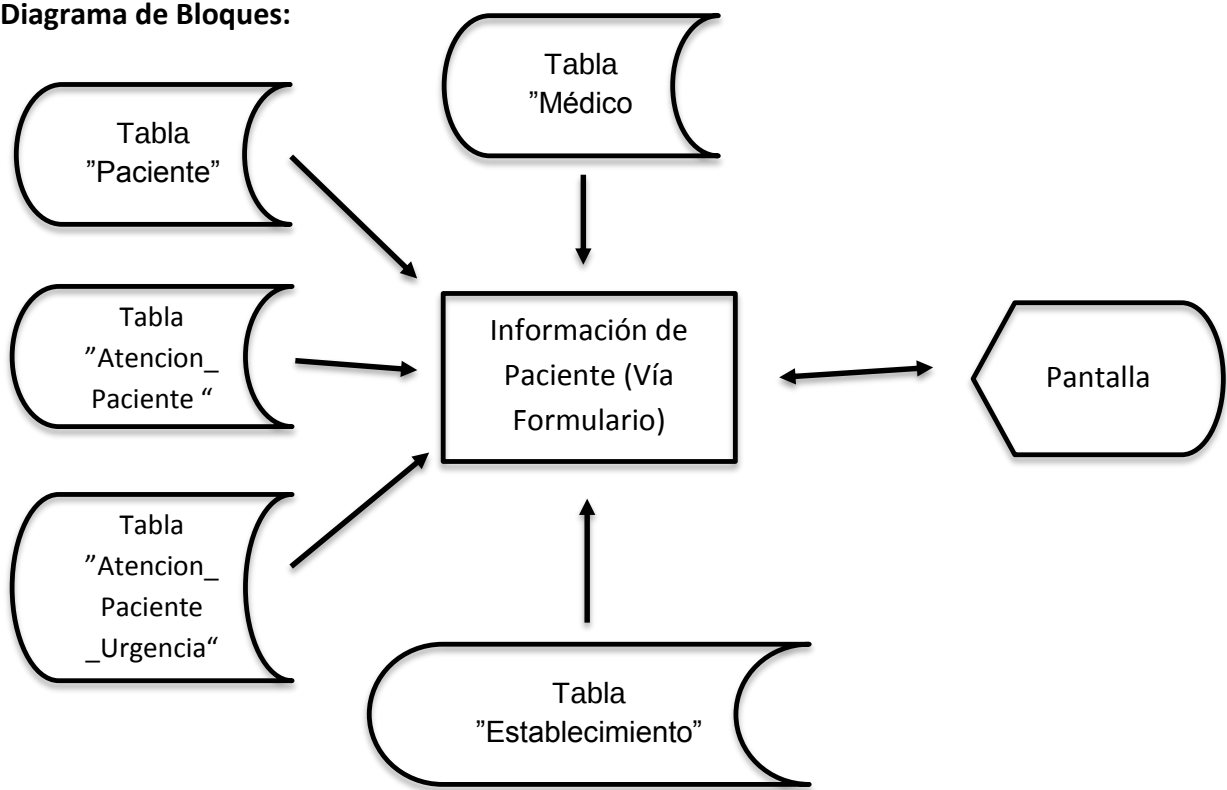
3.4 DESCRIPCIÓN DE PROGRAMAS DESARROLLADOS.

3.4.1 Información de paciente (formulario).

Nombre: Información de paciente (vía formulario).

Objetivo: Permitir la consulta de información pública o médica de paciente, dependiendo de si hay un usuario con sesión iniciada -y de qué tipo es en caso de estarlo-, ingresando el RUT del paciente en cuestión en el formulario web.

Diagrama de Bloques:



Reglas del Proceso: Se usa para desplegar información de un paciente en concreto, ante una situación de emergencia, al momento de usar el RUT del paciente para efectuar una consulta en el sitio del sistema (*Fig. 3-3 y Fig. 3-5*), mostrando información pertinente según quien realice la consulta (*Fig. 3-4 y 3-6*). Los archivos involucrados figuran por permitir presentar esta información, proporcionando información sobre el paciente, su enfermedad, entorno, y quien lo ha tratado.

Diseño de pantallas:

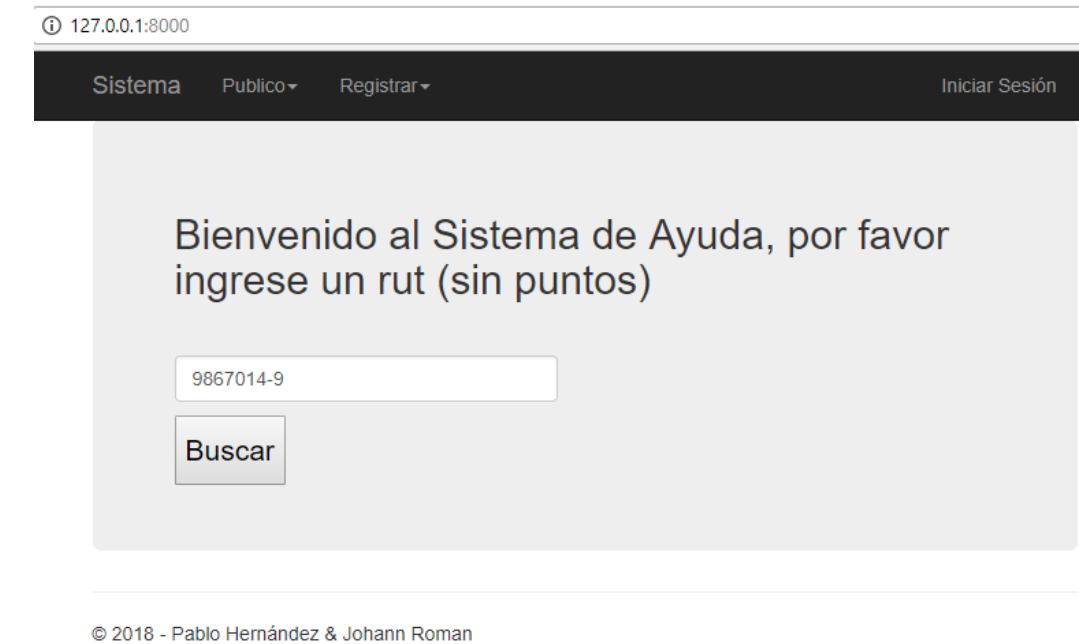


Fig. 3-3 Información de paciente vía Formulario (Público), pantalla de Inicio. Fuente: Elaboración propia.

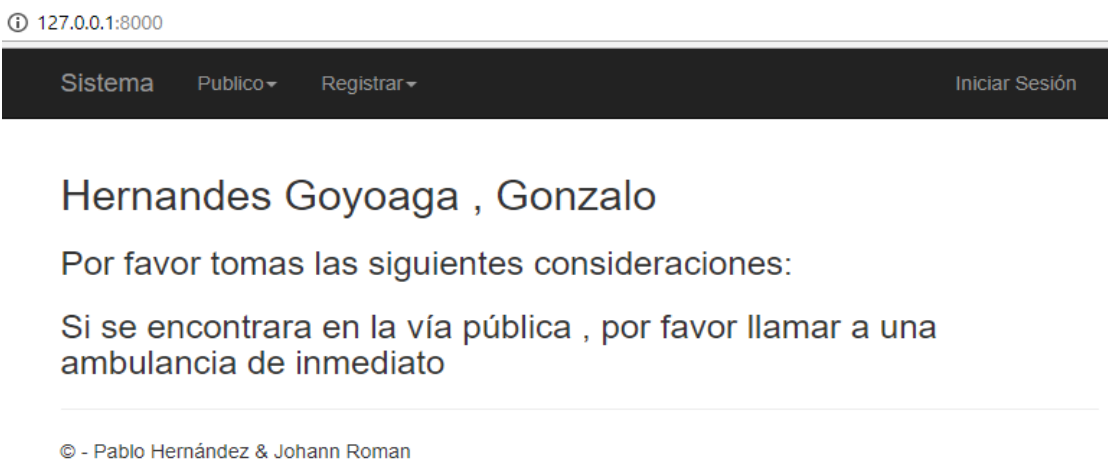


Fig. 3-4 Información de paciente vía Formulario (Público), pantalla de Información. Fuente: Elaboración propia.

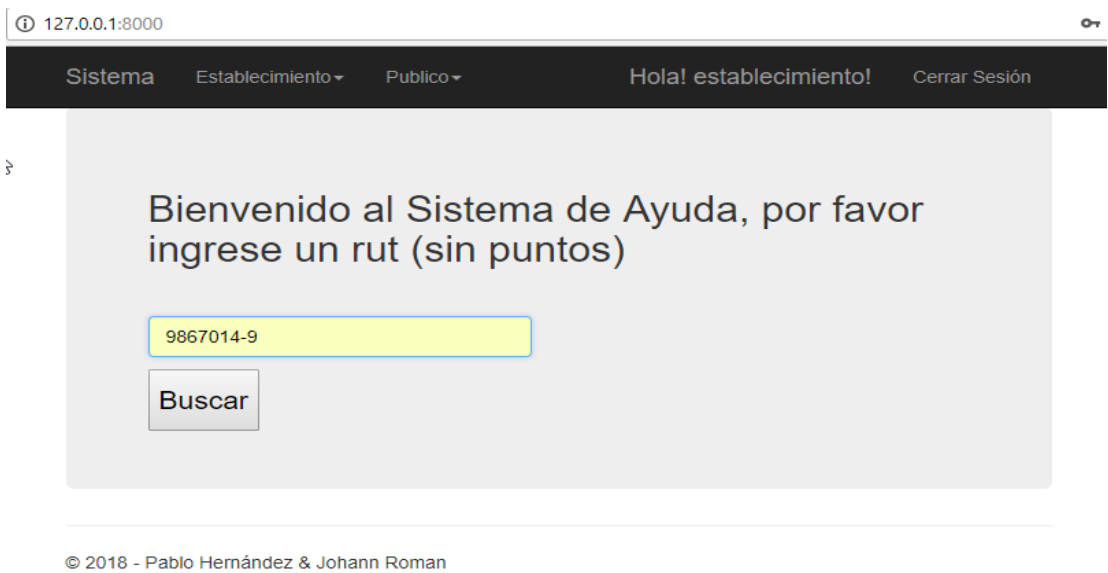


Fig. 3-5 Información de paciente vía Formulario (Autenticado), pantalla de Inicio. Fuente: Elaboración propia.

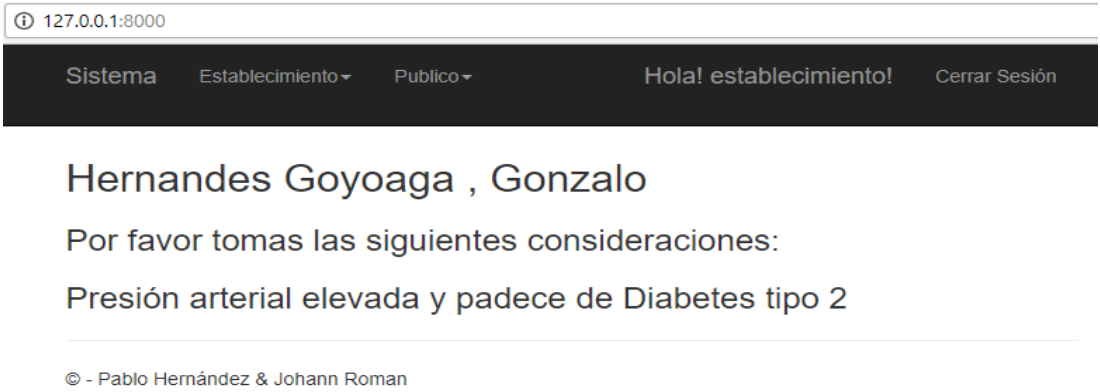


Fig. 3-6 Información de paciente vía Formulario (Autenticado), pantalla de Información. Fuente: Elaboración propia.

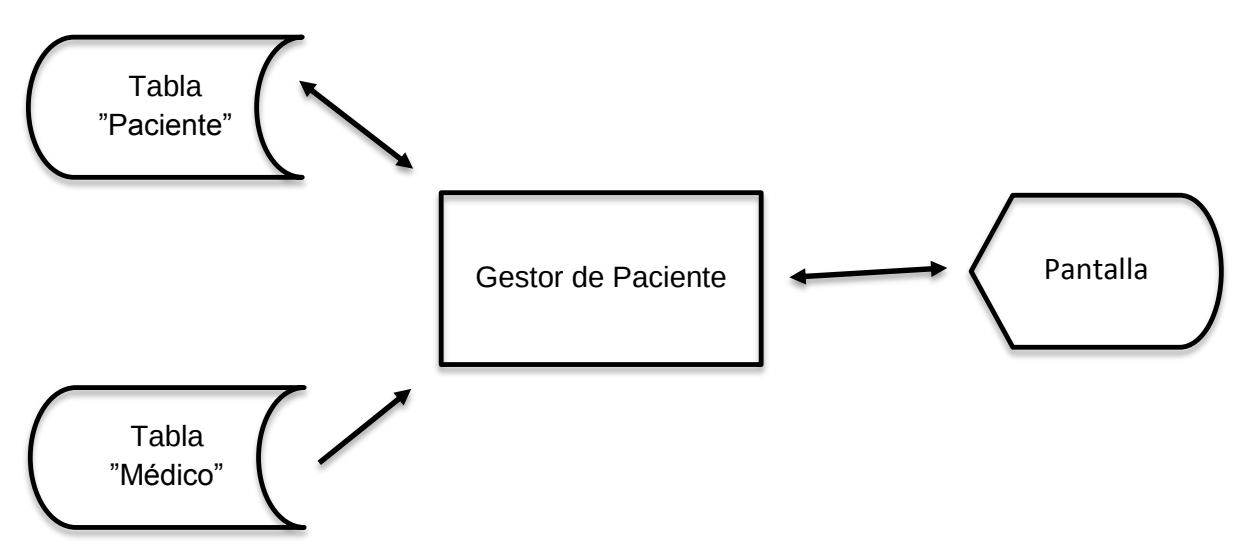
Referencias: Pág. 50 a 51.

3.4.2 Gestor de Paciente.

Nombre: Gestor de Paciente

Objetivo: Permitir a los Médicos registrados en el sistema, ingresar a sus pacientes, acceder a sus datos, y poder editar la información de acuerdo con cómo lo estime conveniente.

Diagrama de Bloques:



Reglas del Proceso: Es usado cuando un médico registrado en el sistema, inicia sesión y necesita ingresar un paciente, debiendo completar un formulario (Fig. 3-7). También es utilizado al momento de que un médico necesite editar la información de un paciente (Fig. 3-9), y se puede utilizar para consultar la información, desplegando resultados distintos dependiendo de quién efectúe la consulta (Fig. 3-8 y Fig. 3-10). Los archivos involucrados, son utilizados para permitir almacenar, editar y desplegar esta información.

Diseño de pantallas:

127.0.0.1:8000/registrar_paciente/

SistemaMedico▼Publico▼

Hola! medico123!Cerrar Sesión

Registrar Paciente

*Todos los campos son requeridos

Rut Paciente:

18658620-4

Nombres:

Pablo

Apellido Paterno:

Hernández

Apellido Materno:

Fuentes

Sexo :

Hombre

Seleccione uno por favor

Fecha de Nacimiento:

30/121993

Email:

pjhfx@gmail.com

Celular:

+56989079205

Celular de Emergencia:

+56989079205

Dirección:

Direccion #123

Peso:

80.3

En kilogramos

Estatura:

1.80

En metros

Registrar

© - Pablo Hernández & Johann Roman

Fig. 3-7 Gestor de paciente, Ingresar Paciente. Fuente: Elaboración propia.

127.0.0.1:8000/consultar_paciente/

SistemaMedicoPublico

Hola! medico123!Cerrar Sesión

Hernandes Goyoaga , Gonzalo

RUT: 9867014-9

Sexo: M

Fecha de Nacimiento: 12 de Agosto de 1969

Fono: +56988451211

Fono Emergencia: +56995791578

Direccion: Direccion 1234

E-mail: emailprueba@gmail.com

Peso: 80,9 kilogramos

Estatura: 1,80 metros

Modificar Datos de Paciente

Ingresar Atencion Ambulatoria

Atenciones Ambulatorias del Paciente

Atenciones de Urgencia del Paciente

© - Pablo Hernández & Johann Roman

Fig. 3-8 Gestor de paciente, Consulta de Información. Fuente: Elaboración propia.

127.0.0.1:8000/modificar_paciente/

SistemaMedicoPublico

Hola! medico123!Cerrar Sesión

Modificar Paciente

Nombres:

Gonzalo

Apellido Paterno:

Hernandes

Apellido Materno:

Goyoaga

Sexo:

Hombre

Seleccione uno por favor

Fecha de Nacimiento:

12/08/1969

Email:

emailprueba@gmail.com

Celular:

+56988455679

Celular de Emergencia:

+56995791578

Dirección:

Direccion Nueva 1234

Peso:

70.9

En kilogramos

Estatura:

1.79

En metros

Modificar

© - Pablo Hernández & Johann Roman

Fig. 3-9 Gestor de paciente, Modificar Información. Fuente: Elaboración propia.



Fig. 3-10 Gestor de paciente, Consulta de Información – Personal de Urgencias. Fuente: Elaboración propia.

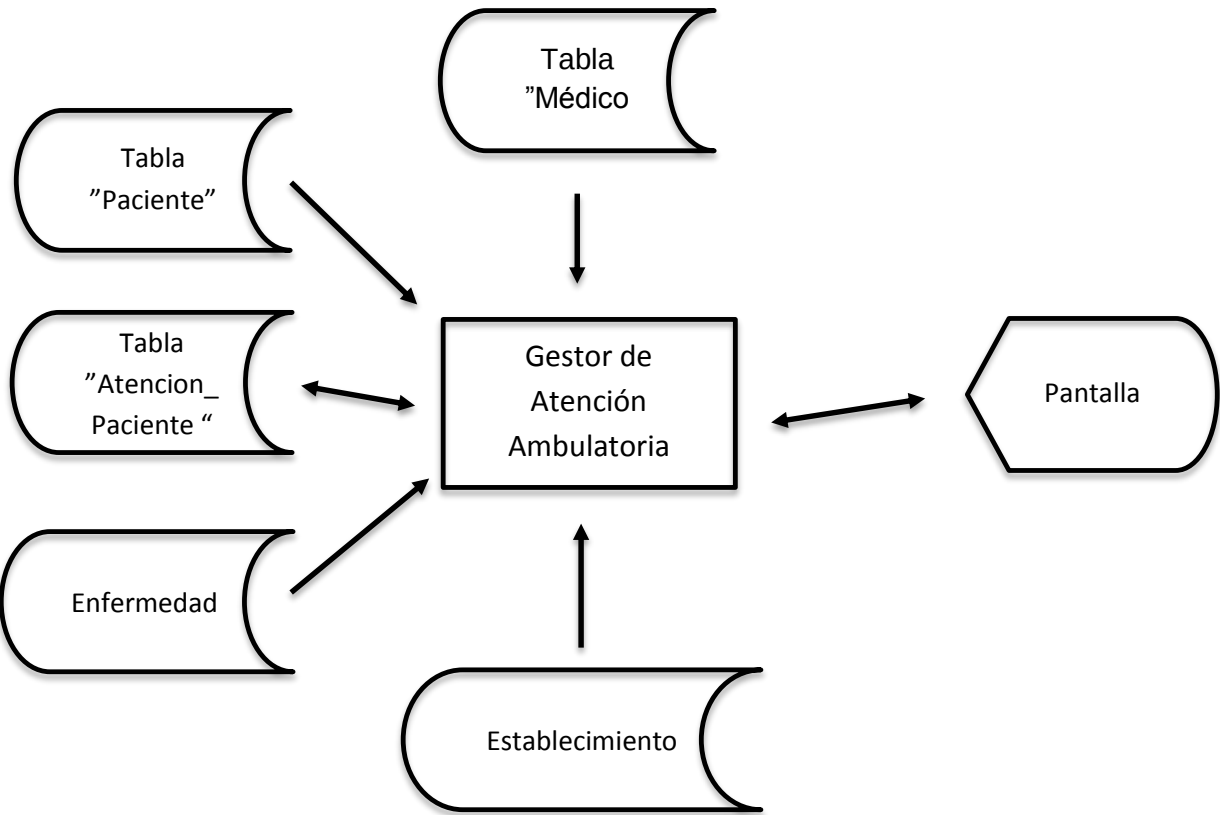
Referencias: *Pág. 51 a 58.*

3.4.3 Gestor de Atención Ambulatoria.

Nombre: Gestor de Atención Ambulatoria

Objetivo: Permitir que los Médicos puedan registrar y consultar las distintas atenciones que les sean prestadas a los pacientes, además en cada una de ellas registrar información respecto a cómo se puede ayudar al paciente en caso de emergencia.

Diagrama de Bloques:



Reglas del Proceso: Se utiliza cuando un médico presta una atención del tipo ambulatoria a un paciente, y registra los datos pertinentes (*Fig. 3-11*), además es usado para consultar las atenciones prestadas a un paciente en concreto, desplegando los detalles de estas (*Fig. 3-12*). Los archivos involucrados, son utilizados para permitir almacenar y desplegar esta información.

Diseño de pantallas:

127.0.0.1:8000/ingresar_atencion/

SistemaMedicoPublico

Hola! medico123!Cerrar Sesión

Ingresar Atencion Ambulatoria

Paciente : 9867014-9

Nombre: Gonzalo Hernández Goyoaga

Fecha y Hora de la atención:

Ej: 01/09/2017 09:25

Tratamiento:

Observación Médica:

Paciente con narcolepsia severa y epilepsia, atendido por resfriado común

Observación Pública:

De encontrar a este paciente desmayado, favor de acercarlo a la sombra y llamar a la ambulancia de

Enfermedad :

Enfermedad no relacionada

Establecimiento :

Fig. 3-11 Gestor de Atención Ambulatoria, Ingresar Atención. Fuente: Elaboración propia.

127.0.0.1:8000/consultar_atencion/

SistemaEstablecimientoPublico

Hola! establecimiento!Cerrar Sesión

Atenciones ambulatorias

Consultas del paciente : 9867014-9

Nombre: Gonzalo Hernández Goyoaga

Listado

Enfermedad	Tratamiento	Observación Médica	Observación Pública	Fecha/Hora Atención
aneurisma cerebral cronico	Potenciadores cognitivos, antidepresivos y estimulantes	Paciente con narcolepsia severa	De encontrar a este paciente desmayado, favor de acercarlo a la sombra y llamar a la ambulancia	24 de Noviembre de 2017 a las 15:18
fibrosis pulmonar	Potenciadores cognitivos, antidepresivos, anticonvulsivos y estimulantes	Paciente con narcolepsia severa y epilepsia	De encontrar a este paciente desmayado, favor de acercarlo a la sombra y llamar a la ambulancia, de encontrarse convulsionando, proteger la boca con ropa para evitar mordidas en la lengua	20 de Diciembre de 2017 a las 15:32

Fig. 3-12 Gestor de Atención Ambulatoria, Consultar Atenciones. Fuente: Elaboración propia.

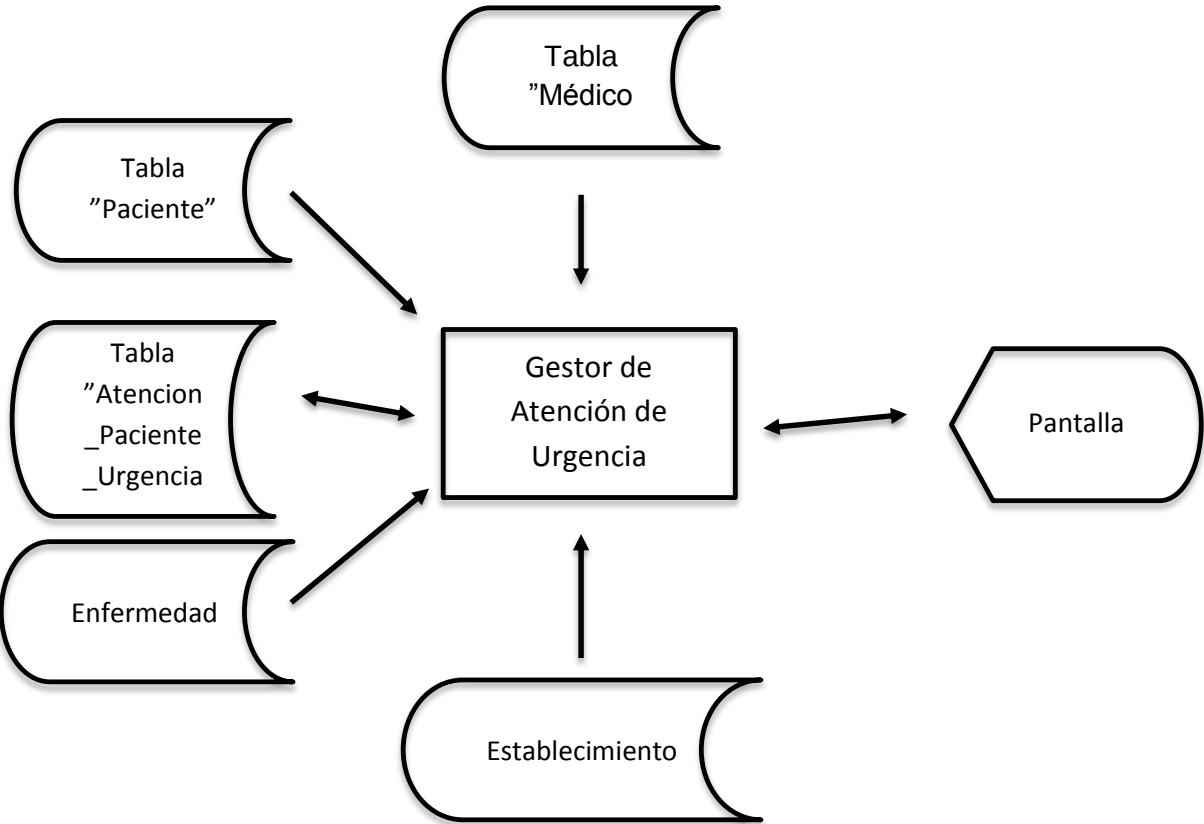
Referencias: Pág. 59 a 62.

3.4.4 Gestor de Atención de Urgencia.

Nombre: Gestor de Atención de Urgencia

Objetivo: Permitir que el personal de urgencias de un centro de salud pueda registrar las distintas atenciones que les sean prestadas a los pacientes. Además, permitir a los Médicos consultar las atenciones de un paciente en particular y mostrar la información registrada destinada a ayudar a los pacientes en caso de emergencia.

Diagrama de Bloques:



Reglas del Proceso: Se utiliza cuando un paciente es ingresado a sala de urgencias, y un encargado del equipo médico registra los datos pertinentes (*Fig. 3-13*), a su vez puede ser usado en esta misma instancia para consultar las atenciones de urgencia prestadas al paciente en cuestión, desplegando los detalles de estas (*Fig. 3-14*). Los archivos involucrados, son utilizados para permitir almacenar y desplegar esta información.

Diseño de pantallas:

127.0.0.1:8000/ingresar_atencion_urg/

SistemaEstablecimientoPublico

Hola! establecimiento!Cerrar Sesión

Ingresar Atencion Urgencia

Paciente: 9867014-9

Nombre: Gonzalo Hernández Goyoaga

Fecha y Hora de la atención :

Ej: 02/05/2016 15:30

Rut del Paciente :

Ej: 9867014-9

Rut del encargado de la atención :

Ej: 2219376-4

Nombre del encargado de la atención :

Ej: Yasna Fuentes

Cargo del encargado de la atención :

Ej: Jefa de Urgencias

Fig. 3-13 Gestor de Atención Urgencia, Ingresar Atención Urgencia. Fuente: Elaboración propia.

127.0.0.1:8000/consultar_atencion_urg/

SistemaEstablecimientoPublico

Hola! establecimiento!Cerrar Sesión

Atenciones de Urgencia

Consultas del paciente : 9867014-9

Nombre: Gonzalo Hernández Goyoaga

Listado			
Enfermedad	Tratamiento	Observacion Urgencia	Fecha/Hora atención
aneurisma cerebral cronico	Inyección de Epinefrina	Paciente se encontraba con síntomas severos de alergia alimentaria	4 de Diciembre de 2017 a las 22:40
fibrosis pulmonar	Sutura de heridas provocadas por accidente vial	Baja presión arterial y obstrucción bronquial severa	30 de Diciembre de 2017 a las 21:11

Fig. 3-14 Gestor de Atención Urgencia, Consultar Atenciones de Urgencia. Fuente: Elaboración propia.

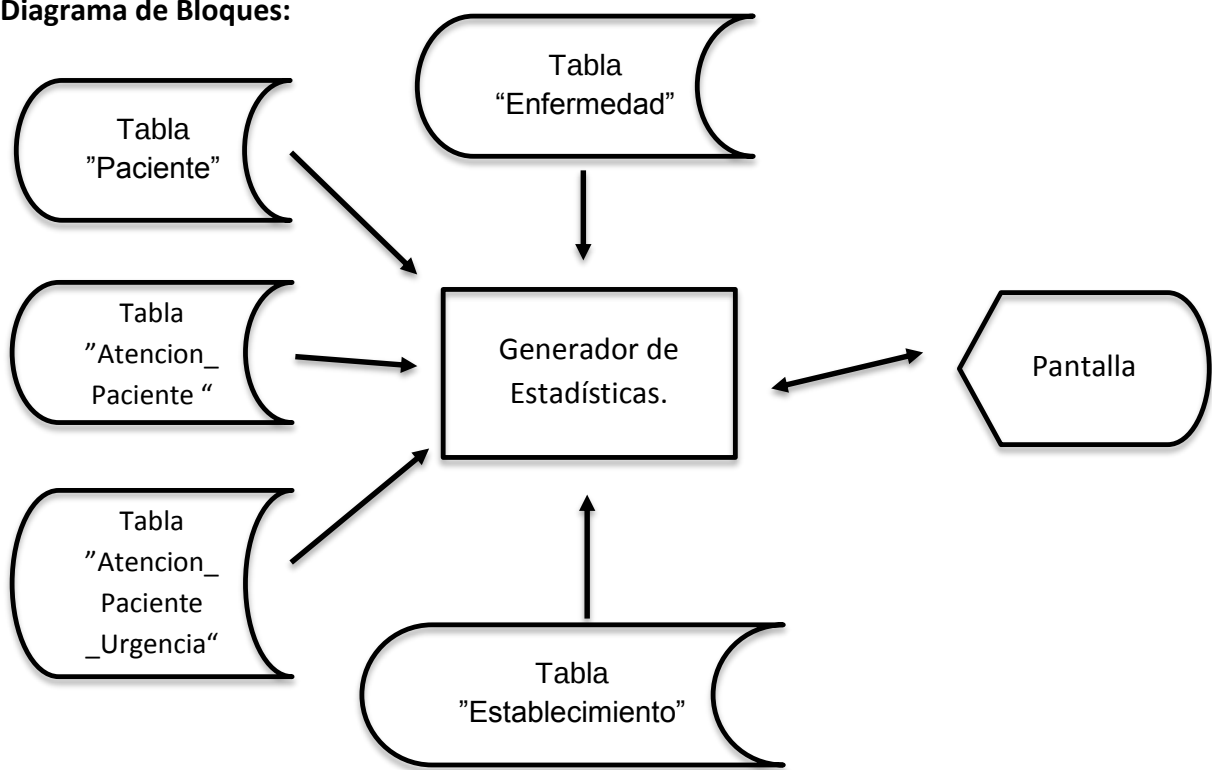
Referencias: Pág. 63 a 67.

3.4.5 Generador de Estadísticas.

Nombre: Generador de Estadísticas.

Objetivo: Permitir que el personal de urgencias de un centro de salud pueda registrar las distintas atenciones que les sean prestadas a los pacientes. Además, permitir a los Médicos consultar las atenciones de un paciente en particular y mostrar la información registrada destinada a ayudar a los pacientes en caso de emergencia.

Diagrama de Bloques:



Reglas del Proceso: Un usuario de cualquier tipo, accede al sistema e ingresa a la funcionalidad "Estadísticas" (Fig. 3-15), donde puede seleccionar los filtros para generar distintas estadísticas mostradas en vivo según los datos almacenados en el sistema (Fig. 3-16).

Diseño de pantallas:

127.0.0.1:8000/estadisticas/

SistemaPublico

Hola! admin!Cerrar Sesión

Generar Estadísticas

Edad:

25

Año:

2015

Mes:

Mayo

Tipo de Atención:

Ambas

Sexo:

Cualquiera

Región:

RM

Enfermedad:

Tuberculosis pulmonar

Establecimiento:

Cualquiera

Fig. 3-15 Selección de filtros, Generador de estadísticas. Fuente: Elaboración propia.

127.0.0.1:8000/estadisticas/

SistemaPublico

Hola! admin!Cerrar Sesión

Sexo : Cualquiera

Tipo de consulta : Ambas

Edad : 25

Región : 13

Año : 2015

Mes : Mayo

Establecimiento : Cualquiera

Cantidad de atenciones ambulatorias : 20

Cantidad de atenciones de urgencia 25

Promedio de atenciones por región: 3,75

Porcentaje de atenciones respecto al total: 0,03%

Fig. 3-16 Despliegue de estadísticas, Generador de estadísticas. Fuente: Elaboración propia.

Referencias: Pág. 67 a 70.

CONCLUSIONES

Django es un framework Web basado en Python, de patrón de diseño MVC de alto nivel de abstracción para modelos de datos y control que permite desarrollar aplicaciones web de manera rápida y modular. Es particularmente bueno para el desarrollo de prototipos y se integra con herramientas de desarrollo como ambientes virtuales, diversas bases de datos y una variedad extensa de paquetes de python (open source) para implementaciones de distintas lógicas de negocio.

En términos de la organización, ejecución y mantención de código, este framework separa la aplicación en las capas típicas de Modelo-Vista-Controlador, los datos son manejados a través de objetos definidos por el desarrollador y mapeados a la base de datos a través de ORM (Modelos, Vistas y Motor de Plantillas). Esta sintaxis es muy diferente a la de SQL regular y tienden a confundirse ciertos términos en etapas iniciales (como el uso de objetos Q y funciones de la API de Base de Datos).

Es necesario tener sólidos conocimientos previos de expresiones regulares, ya que estas gobiernan el enrutamiento hacia las diversas vistas. A su vez, la relación entre la base de datos y los modelos que están escritos en Django debe ser cuidadosamente manejada, cada vez que se cambia algo en los modelos se deben aplicar migraciones constantemente, lo que significa sincronizar los modelos de datos con la base de datos en sí.

Al momento de desarrollar este trabajo, no se contaba con los conocimientos necesarios respecto a programación web con controladores, los cuales serían adquiridos posteriormente en una asignatura al final de la carrera; por lo que se tuvo que hacer una investigación previa y práctica más extensa a lo normal, dejando un tiempo total de desarrollo cercano a los cuatro meses.

Una vez superadas estas barreras y desarrollado el proyecto, la siguiente meta fue realizar las pruebas de software y obtener opiniones respecto a las funcionalidades.

Se contó con la ayuda de alumnos y docentes de la facultad de medicina de la Universidad Diego Portales, y mediante las pruebas realizadas, y con el feedback recibido luego de las mismas; se puede concluir:

- Los tiempos de respuesta en sala de emergencia disminuyen considerablemente al poseer la información básica y personal del paciente en situación de riesgo.
- Se sugiere crear campañas de concientización respecto a este tipo de enfermedades y como puede cambiar el destino del paciente si recibe la ayuda adecuada, al mismo tiempo que se da a conocer el sistema al público.
- La información del paciente y su enfermedad, destinada a su uso en la sala de urgencias, es discutida por su subjetividad y se sugieren mejoras; tales como que se muestren las distintas anotaciones a modo de historial. Se toma sugerencia usando la funcionalidad de feedback y se planifica añadir funcionalidad en una próxima versión.
- Se planean reuniones con nuevas entidades relacionadas con el área de la medicina, particularmente con IFMSA.

Las mayores complicaciones fueron no contar con conexiones formales y un feedback constante durante todo el proceso de desarrollo; lo cual sugiere que las próximas actualizaciones al sistema, debiesen estar orientadas a seguir el trabajo en conjunto a un comité especializado con profesionales del área de la salud.

Se espera que, con las modificaciones propuestas, se pueda acelerar aún más el procedimiento de urgencia y colaborar más eficazmente con la información pertinente. Se pretende dar a conocer el sistema a autoridades competentes.

BIBLIOGRAFÍA

Django Software Foundation. Django: The web framework for perfectionists with deadlines. [En línea] <<http://www.djangoproject.com>> [consulta: octubre 2017].

Freitas, Vitor. Stories about Python, Django and Web Development. [En línea] <www.simpleisbetterthancomplex.com> [consulta: noviembre 2017].

García M., Saúl. Introducción a Django. The Django Corporation, 2015.

Holovaty, Adrian y Kaplan Mosss, Jacob. La guía definitiva de Django. The Django Corporation, 2015.

Russell Severance, Charles. Python for everybody. Createspace Independent Publishing Platform, 2016.

Stack Exchange Inc. Learn, Share, Build. [En línea] <<http://www.stackoverflow.com>> [consulta: Octubre 2017].

ANEXOS

VALIDADORES REGEX.

- a) rutregex = RegexValidator(r'^0*(\d{1,3}(\.?\d{3})*)\-(?![dkK])\$', 'Solo rut con o sin puntos y guion ')

- b) celularregex = RegexValidator(r'^\+?56(\s?)(0?9)(\s?)[98765]\d{7}\$', 'Solo celulares en chile')

- c) passwordregex = RegexValidator(r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,30}\$', 'Minimo ocho caracteres, Maximo 30, debe contener por lo menos una letra mayuscula, una minuscula y un numero')

- d) nombreregex = RegexValidator(r'^(?=.*[a-z])[A-Za-zñÑáéíóúÁÉÍÓÚ\s]{2,30}\$', 'Maximo treinta caracteres, no se admiten numeros')

- e) apellidosregex = RegexValidator(r'^(?=.*[a-z])[A-Za-zñÑáéíóúÁÉÍÓÚ\s]{2,15}\$', 'Maximo quince caracteres, no se admiten numeros')

- f) direccionregex = RegexValidator(r'^[A-Za-zñÑáéíóúÁÉÍÓÚ\s,#0-9]{,30}\$', 'maximo 30 caracteres')

- g) usuarioregex = RegexValidator(r'^[A-Za-z0-9ñ_]{5,15}\$', 'Minimo 5 caracteres, maximo 15')

- h) pesoregex = RegexValidator(r'^\d{1,3}(\.\d{1,2})?', 'peso en kilos, max 2 decimales y el separador es un punto ')

- i) estaturaregex = RegexValidator(r'^\d{1,3}(\.\d{1,2})?', 'estatura en centimetros, max 2 decimales y el separador es un punto ')

- j) fechahoraregex = RegexValidator('^(\d{1,2}\/\d{1,2}\/\d{4})\ \d{2}:\d{2}\$', 'dd/mm/aaaa HH:MM')

- k) fecharegex = RegexValidator('^\d{1,2}\/\d{1,2}\/\d{4}\$', 'dd/mm/aaaa')

INFORMACIÓN DE PACIENTE (VÍA FORMULARIO).**a) Información de paciente vía formulario, Vista.**

```

def home(request):
if request.method == 'POST':
    form = SearchForm(request.POST)
if form.is_valid():
    try:
        rut=form.cleaned_data['rut_paciente']
        rut_limpio= rut.replace('.', '')
        atencion=
        AtencionPacienteAmbulatorio.objects.filter(rut_paciente__exact=rut_limpio).latest()
        paciente = Paciente.objects.get(rut_paciente__exact=rut_limpio)
    except:
        return render(request, '../templates/app/no_info.html')
        return render(request, '../templates/app/info_paciente.html', {'atencion': atencion,
            'paciente':paciente})
else:
    form = SearchForm()
    return render(request, '../templates/app/index.html', {'form': form, 'title': 'Home Page',
        'year':datetime.now().year,})

```

b) Información de Paciente vía Formulario, Formulario.

```

class ListFormPaciente(forms.Form):
rut_paciente=forms.CharField(max_length=10,          required=True,label='&#39;Rut          del
Paciente:&#39;;,
widget=forms.TextInput(attrs={&#39;placeholder&#39;:          &#39;Ej:          18658620-
4&#39;;&#39;class&#39;: &#39;form-
control&#39;});),validators=[rutregex])
#Verificar que el rut sea válido
def clean_rut_paciente(self):
rut_form=str(self.cleaned_data[&#39;rut_paciente&#39;])
rut_limpio = rut_form.replace('&quot;.&quot;,&quot;.&quot;')
digito_verificador = rut_limpio[-1:]
if digito_verificador==&quot;k&quot; or digito_verificador==&quot;K&quot;:
digito_verificador = 10
rut_limpio = rut_limpio[:-2]
reversed_digits = map(int, reversed(rut_limpio))
factors = cycle(range(2, 8))
suma = sum(d * f for d, f in zip(reversed_digits, factors))
digito_resultado= 11 -(suma%11)
if digito_resultado==11:

```

```

digito_resultado=0
if int(digito_verificador)!=digito_resultado:
raise forms.ValidationError(_('El rut no es válido'), code='invalid')

#validar que exista en el sistema
if not Paciente.objects.filter(rut_paciente=rut_form).exists():
raise forms.ValidationError(_('El paciente NO existe en el sistema'),
code='invalid')
return rut_form

```

GESTOR DE PACIENTE.

a) Registrar Paciente, Vista.

```

def registrarPaciente(request):
    if request.method == 'POST':
        form = RegistrationFormPaciente(request.POST)
        if form.is_valid():
            rutp = form.cleaned_data['rut_paciente']
            nombresp = form.cleaned_data['nombres']
            apellidopp = form.cleaned_data['apellido_paterno']
            apellidomp = form.cleaned_data['apellido_materno']
            sexop = form.cleaned_data['sexo']
            emailp = form.cleaned_data['email']
            fecha_nacimientop=form.cleaned_data['fecha_nacimiento']
            fonop= form.cleaned_data['fono']
            fono_emergenciap= form.cleaned_data['fono_emergencia']
            direccionp = form.cleaned_data['direccion']
            pesop = form.cleaned_data['peso']
            estaturap = form.cleaned_data['estatura']
            new_paciente = Paciente (rut_paciente=rutp, nombres=nombresp, apellido_paterno =
            apellidopp,                apellido_materno=apellidomp,                sexo=sexop,
            fecha_nacimiento=fecha_nacimientop,                email=emailp,                fono=fonop,
            fono_emergencia=fono_emergenciap,                direccion=direccionp,                peso=pesop,
            estatura=estaturap)
            new_paciente.save()
            return redirect('home')
        else:
            form = RegistrationFormPaciente()
    return render(request, '../templates/app/registrar_paciente.html', {'form': form})

```

b) Registrar paciente, formulario.

```

class RegistrationFormPaciente(forms.Form):
    sexo_choice = ( ('M', _("Hombre")), ('F', _("Mujer")), ('O', _("Otros")), )
    rut_paciente=forms.CharField(max_length=10, required=True,label='Rut Paciente:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej:18658620-4', 'class': 'form-control'}) ,
        validators=[rutregex])
    nombres=forms.CharField(max_length=30, required=True, label='Nombres:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: Juan','class': 'form-control'}),
        validators=[nombregex])
    apellido_paterno=forms.CharField(max_length=30, required=True, label='Apellido Paterno:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: Pardo','class': 'form-control'}),
        validators=[apellidosregex] )
    apellido_materno=forms.CharField(max_length=30, required=True,label='Apellido Materno:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: Rodríguez','class': 'form-control'}),
        validators=[apellidosregex])
    sexo=forms.ChoiceField(choices = sexo_choice, label="Sexo ", initial='1',
        widget=forms.Select(attrs={'class': 'form-control'}), required=True,help_text='Seleccione
        uno por favor')
    fecha_nacimiento=forms.DateField(required=True,label='Fecha de Nacimiento:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: 20/12/1993','class': 'form-control'}),
        input_formats=['%d/%m/%Y'])
    email=forms.EmailField(max_length=30,label='Email:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: correo@gmail.com','class': 'form-
        control'})))
    fono=forms.CharField(max_length=12, required=True,label='Celular:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej:+56989079205','class': 'form-control'}),
        validators=[celularregex])
    fono_emergencia=forms.CharField(max_length=12,label='Celular de Emergencia:',
        required=True, widget=forms.TextInput(attrs={'placeholder': 'Ej:18658620-4','class': '
        form-control'}), validators=[celularregex])
    direccion=forms.CharField(max_length=30,label='Dirección:', required=True,
        widget=forms.TextInput(attrs={'placeholder': 'Ej: Dirección #123','class': 'form-control'}),
        validators=[direccionregex])
    peso=forms.DecimalField( required=True,label='Peso:', help_text='En kilogramos',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: 90.4','class': 'form-control'}),
        max_value=300.0, min_value=35.0, max_digits=3,decimal_places=1,
        validators=[pesoregex] )
    estatura=forms.DecimalField( required=True, label='Estatura:', help_text='En
        metros',max_value=3.0, min_value=1.0,max_digits=3, decimal_places=2,
        widget=forms.TextInput(attrs={'placeholder': 'Ej: 1.81','class': 'form-control'})
        ,validators=[estaturaregex])

```

```

def clean_rut_paciente(self):
    rut_form=str(self.cleaned_data['rut_paciente'])
    rut_limpio = rut_form.replace(".", "")
    digito_verificador = rut_limpio[-1:]
    if digito_verificador=="k" or digito_verificador=="K":
        digito_verificador = 10
    rut_limpio = rut_limpio[:-2]
    reversed_digits = map(int, reversed(rut_limpio))
    factors = cycle(range(2, 8))
    suma = sum(d * f for d, f in zip(reversed_digits, factors))
    digito_resultado= 11 -(suma%11)
    if digito_resultado==11:
        digito_resultado=0
    if int(digito_verificador)!=digito_resultado:
        raise forms.ValidationError(_("El rut no es valido"), code='invalid')

#Validar que el paciente no exista ya en el sistema
if Paciente.objects.filter(rut_paciente=rut_form).exists():
    raise forms.ValidationError(_("El paciente ya existe en el sistema"), code='invalid')
return rut_form

def clean_fecha_nacimiento(self):
    fecha_nacimiento=self.cleaned_data['fecha_nacimiento']
    fecha_hoy=date.today()

    if fecha_nacimiento > fecha_hoy:
        raise forms.ValidationError(_("La fecha de nacimiento no puede ser mayor a la fecha actual"),
            code='invalid')
    return fecha_nacimiento

```

c) Consultar Paciente, Vista.

```
def consultarPaciente(request):
    try:
        current_user = request.user
        current_username = str(current_user.username)
    except:
        messages.info(request, 'No puedes realizar esta acción sin estar autenticado')
        return redirect('home')

    if request.method == 'POST':
        form = ListFormPaciente(request.POST)
        if form.is_valid():
            rut_form=form.cleaned_data['rut_paciente']
            rut_limpio=rut_form.replace('.', '')
            request.session['rut_paciente'] = rut_limpio
            paciente = Paciente.objects.get(rut_paciente=rut_limpio)
            return render(request, '../templates/app/pagina_paciente.html', {'paciente': paciente})
        else:
            form = ListFormPaciente()
    return render(request, '../templates/app/consultar_paciente.html', {'form': form})
```

d) Consultar Paciente, Formulario.

```
class ListFormPaciente(forms.Form):
    rut_paciente=forms.CharField(max_length=10, required=True,label='Rut del Paciente:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: 18658620-4','class': 'form-control'}),
        validators=[rutregex])

    #Verificar que el rut sea valido

    def clean_rut_paciente(self):
        rut_form=str(self.cleaned_data['rut_paciente'])

        rut_limpio = rut_form.replace(".", "")

        digito_verificador = rut_limpio[-1:]

        if digito_verificador=="k" or digito_verificador=="K":
            digito_verificador = 10
        rut_limpio = rut_limpio[:-2]
        reversed_digits = map(int, reversed(rut_limpio))
        factors = cycle(range(2, 8))
        suma = sum(d * f for d, f in zip(reversed_digits, factors))
        digito_resultado= 11 -(suma%11)
```

```

if digito_resultado==11:
    digito_resultado=0

if int(digito_verificador)!=digito_resultado:
    raise forms.ValidationError(_("El rut no es valido"), code='invalid')

#validar que exista en el sistema
if not Paciente.objects.filter(rut_paciente=rut_form).exists():
    raise forms.ValidationError(_("El paciente NO existe en el sistema"), code='invalid')
return rut_form

```

e) Modificar Paciente, Vista.

```
def modificarPaciente(request):
```

```

    try:
        rut_paciente_session= request.session['rut_paciente']
        rut_limpio=rut_paciente_session.replace('.', '')
        paciente = Paciente.objects.get(rut_paciente=rut_limpio)
    except:
        return redirect('consultar_paciente')

    data = {'nombres': paciente.nombres,
            'apellido_paterno': paciente.apellido_paterno,
            'apellido_materno': paciente.apellido_materno,
            'sexo': paciente.sexo,
            'fecha_nacimiento': paciente.fecha_nacimiento.strftime('%d/%m/%Y'),
            'email': paciente.email,
            'fono': paciente.fono,
            'fono_emergencia': paciente.fono_emergencia,
            'direccion': paciente.direccion,
            'peso': paciente.peso,
            'estatura': paciente.estatura,
            }

    form = UpdateFormPaciente(data,rut_paciente=rut_paciente_session)

    if request.method == 'POST': #terminar la actualizacion con el commit
        form = UpdateFormPaciente(request.POST,rut_paciente=rut_paciente_session)
        if form.is_valid():
            nombresp = form.cleaned_data['nombres']
            apellidopaterno = form.cleaned_data['apellido_paterno']
            apellidomaterno = form.cleaned_data['apellido_materno']
            sexop = form.cleaned_data['sexo']
            fechap = form.cleaned_data['fecha_nacimiento']
            fonop = form.cleaned_data['fono']
            fonop_emergencia = form.cleaned_data['fono_emergencia']
            emailp = form.cleaned_data['email']
            direccionp = form.cleaned_data['direccion']
            pesop = form.cleaned_data['peso']
            estaturap = form.cleaned_data['estatura']
            Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(nombres=nombresp)

```

```

Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(apellido_paterno=apelli
do_paternop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(apellido_materno=apelli
do_maternop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(sexo=sexop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(fecha_nacimiento=fecha
_nacimientop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(fono=fonop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(fono_emergencia=fono_
emergenciap)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(email=emailp)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(direccion=direccionp)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(peso=pesop)
Paciente.objects.filter(rut_paciente__exact=rut_limpio).update(estatura=estaturap)
return redirect('home')
return render(request, '../templates/app/modificar_paciente.html', {'form': form})

```

f) Modificar Paciente, Formulario.

```
class UpdateFormPaciente(forms.Form):
```

```
    sexo_choice = (
        ('M', _("Hombre")),
        ('F', _("Mujer")),
        ('O', _("Otros")),
    )
```

```
    nombres = forms.CharField(max_length=30, required=True, label='Nombres:',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: Ninoska Javiera','class': 'form-control'}),
    validators=[nombregex])
```

```
    apellido_paterno = forms.CharField(max_length=30, required=True, label='Apellido Paterno:',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: Bardo','class': 'form-control'}),
    validators=[apellidosregex])
```

```
    apellido_materno = forms.CharField(max_length=30, required=True, label='Apellido Materno:',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: Millar','class': 'form-control'}),
    validators=[apellidosregex])
```

```
    sexo = forms.ChoiceField(choices = sexo_choice, label="Sexo:", initial='1',
    widget=forms.Select(attrs={'class': 'form-control'}), required=True, help_text='Seleccione uno por
    favor')
```

```
    fecha_nacimiento = forms.DateField(required=True, label='Fecha de Nacimiento:',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: 12/04/1997','class': 'form-control'}),
    input_formats=['%d/%m/%Y'])
```

```
    email = forms.EmailField(max_length=30, label='Email:', widget =
    forms.TextInput(attrs={'placeholder': 'Ej: pjhfx@gmail.com','class': 'form-control'}))
```

```
    fono = forms.CharField(max_length=12, label='Celular:', required=True, widget=
    forms.TextInput(attrs={'placeholder': 'Ej: +56989079205','class': 'form-control'}),v
    alidators=[celularregex])
```

```
    fono_emergencia = forms.CharField(max_length=12, label='Celular de Emergencia:',
    required=True, widget=forms.TextInput(attrs={'placeholder': 'Ej: +56989079205','class': 'form-
    control'}), validators=[celularregex])
```

```
    direccion = forms.CharField(max_length=30, required=True, label='Dirección:',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: Dirección #231','class': 'form-control'}),
    validators=[direccionregex])
```

```
    peso = forms.DecimalField( required=True, help_text='En kilogramos',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: 90.3','class': 'form-control'}), max_value=300.0,
    min_value=35.0,max_digits=3,decimal_places=1)
```

```
    estatura =forms.DecimalField( required=True, help_text='En metros',
    widget=forms.TextInput(attrs={'placeholder': 'Ej: 1.81','class': 'form-control'}), max_value=3.0,
    min_value=1.0,max_digits=3, decimal_places=2)
```

```

def __init__(self,*args,**kwargs):
    self.rut_paciente = kwargs.pop('rut_paciente')
    super(UpdateFormPaciente,self).__init__(*args,**kwargs)

def clean_fecha_nacimiento(self):
    #validar que la fecha no sea mayor a la actual
    fecha_nacimiento= self.cleaned_data['fecha_nacimiento']
    fecha_hoy= date.today()

    if fecha_nacimiento > fecha_hoy:
        raise forms.ValidationError(_("La fecha de nacimiento NO puede ser mayor a la fecha
        actual"), code='invalid')
    return fecha_nacimiento

def clean_email(self):
    email=self.cleaned_data['email']
    paciente = Paciente.objects.get(rut_paciente=self.rut_paciente)

    if str(paciente.email) == email:
        pass
    elif str(paciente.email) != email:
        if Paciente.objects.filter(email=email).exists():
            raise forms.ValidationError(_('Este email ya está asociado a un paciente'),
            code='invalid')
    return email

```

GESTOR DE ATENCIÓN AMBULATORIA.**a) Ingresar Atención Ambulatoria, Vista.**

```
def ingresarAtencionAmbulatoria(request):
```

```
    try:
```

```
        current_user = request.user
        current_username = str(current_user.username)
        medico = Medico.objects.get(user__username__exact=current_username)
        rut_paciente_session= request.session['rut_paciente']
        rut_limpio=rut_paciente_session.replace('.', '')
        paciente = Paciente.objects.get(rut_paciente=rut_limpio)
```

```
    except:
```

```
        messages.info(request, 'No puedes realizar esta acción sin estar autenticado')
        return redirect('home')
```

```
    if request.method == 'POST':
```

```
        form = RegistrationFormAtencionAmbulatoria(request.POST)
```

```
        if form.is_valid():
```

```
            rut_paciente = rut_limpio
            fecha_atencion_hora = form.cleaned_data['fecha_atencion_hora']
            tratamiento = form.cleaned_data['tratamiento']
            observacion_medica = form.cleaned_data['observacion_medica']
            observacion_publica = form.cleaned_data['observacion_publica']
            codigo_establecimiento = form.cleaned_data['codigo_establecimiento']
            id_who= form.cleaned_data['id_who']
            paciente = Paciente.objects.get(rut_paciente=rut_paciente)
            establecimiento=
                Establecimiento.objects.get(codigo_establecimiento=codigo_establecimiento)
            enfermedad = Enfermedad.objects.get(id_who=str(id_who))
            new_atencion =
                AtencionPacienteAmbulatorio(fecha_atencion_hora=fecha_atencion_hora,
                rut_paciente=paciente, tratamiento=tratamiento,
                observacion_medica=observacion_medica,
                observacion_publica=observacion_publica,
                codigo_establecimiento=establecimiento, rut_medico=medico,
                id_who=enfermedad)
            new_atencion.save()
            Return redirect('home')
```

```
    else:
```

```

    try:
        atencion = AtencionPacienteAmbulatorio.objects.filter(rut_paciente=rut_limpio).latest()
        data = {'observacion_publica': atencion.observacion_publica,
                'observacion_medica': atencion.observacion_medica,}
        form = RegistrationFormAtencionAmbulatoria(initial=data)
    except:
        form = RegistrationFormAtencionAmbulatoria()
    return render(request, '../templates/app/ingresar_atencion_ambulatoria.html', {'form': form})

```

b) Ingresar Atención Ambulatoria, Formulario.

```
class RegistrationFormAtencionAmbulatoria (forms.Form):
```

```

    fecha_atencion_hora = forms.DateTimeField(required=True, label='Fecha y Hora de la atención:',
        widget=forms.TextInput(attrs={'placeholder': 'Ej: 01/09/2017 09:25','class': 'form-control'}),
        input_formats=['%d/%m/%Y %H:%M'])

```

```

    tratamiento = forms.CharField(max_length=800, required=True, label='Tratamiento:',
        widget=forms.Textarea(attrs={'placeholder': '', 'class': 'form-control'}))

```

```

    observacion_medica= forms.CharField(max_length=800, required=True, label='Observación
    Médica:', widget=forms.Textarea(attrs={'placeholder': '', 'class': 'form-control'}))

```

```

    observacion_publica= forms.CharField(max_length=800, required=True, label='Observación
    Pública:', widget=forms.Textarea(attrs={'placeholder': '', 'class': 'form-control'}))

```

```

    id_who = forms.ChoiceField(required=True, label='Enfermedad :', choices=[(enfermedad.id_who,
    enfermedad.nombre) for enfermedad in Enfermedad.objects.all()],
    widget=forms.Select(attrs={'class': 'form-control'}))

```

```

    codigo_establecimiento = forms.ChoiceField(required=True, label='Establecimiento :',
    choices=[(establecimiento.codigo_establecimiento, establecimiento.nombre) for
    establecimiento in Establecimiento.objects.all()], widget=forms.Select(attrs={'class': 'form-
    control'}))

```

```
def clean_fecha_atencion_hora(self):
```

```
    fecha_atencion_hora =self.cleaned_data['fecha_atencion_hora']
```

```
    fecha_hoy= datetime.today()
```

```
    fecha_hoy = fecha_hoy.replace(tzinfo=None)
```

```
    fecha_atencion_hora=fecha_atencion_hora.replace(tzinfo=None)
```

```
    if fecha_atencion_hora > fecha_hoy:
```

```
        raise forms.ValidationError(_("La fecha/hora de la atención NO puede ser mayor a la
        fecha actual"), code='invalid')
```

```
    return fecha_atencion_hora
```

```

def clean_rut_paciente(self):
    rut_form=self.cleaned_data['rut_paciente']
    rut_limpio = rut_form.replace(".", "")

    #MODULO 11
    digito_verificador = rut_limpio[-1:]

    if digito_verificador=="k" or digito_verificador=="K":
        digito_verificador = 10

    rut_limpio = rut_limpio[:-2]

    reversed_digits = map(int, reversed(rut_limpio))
    factors = cycle(range(2, 8))
    suma = sum(d * f for d, f in zip(reversed_digits, factors))
    digito_resultado= 11 -(suma%11)

    if int(digito_verificador)!=digito_resultado:
        raise forms.ValidationError(_("El rut no es valido"), code='invalid')
    #Verifica que el paciente EXISTA en el sistema
    if not Paciente.objects.filter(rut_paciente=rut_form).exists():
        raise forms.ValidationError(_('Este rut de paciente no existe en el sistema'),
        code='invalid')

    return rut_form

def clean (self):
    cleaned_data = super(RegistrationFormAtencionAmbulatoria,self).clean()
    rut_paciente= cleaned_data.get("rut_paciente")
    fecha_atencion_hora= cleaned_data.get("fecha_atencion_hora")

    if      AtencionPacienteAmbulatorio.objects.filter(Q(rut_paciente=rut_paciente)      |
        Q(fecha_atencion_hora=fecha_atencion_hora)).exists():
        msg="Ya existe una atencion prestada al paciente en esta fecha/hora"
        self.add_error('fecha_atencion_hora',msg)

```

c) Consultar Atención Ambulatoria, Vista.

```
def consultarAtencionAmbulatoria(request):
    try:
        rut_paciente_session= request.session['rut_paciente']
        rut_limpio=rut_paciente_session.replace('.', '')
        paciente = Paciente.objects.get(rut_paciente=rut_limpio)
    except:
        return redirect('consultar paciente')

    atenciones = AtencionPacienteAmbulatorio.objects.filter(rut_paciente__exact=rut_limpio)
    enfermedades = []
    for atencion in atenciones:
        enfermedad = Enfermedad.objects.get(id_who=atencion.id_who)
        enfermedades.append(enfermedad)
    lista = zip(atenciones,enfermedades)
    return render(request, '../templates/app/listado_atencion_ambulatoria.html', {'lista': lista})
```

6.5 GESTOR DE ATENCIÓN DE URGENCIA.

a) Ingresar Atención de Urgencia, Vista.

```
def ingresarAtencionUrgencia(request):
    try:
        current_user = request.user
        current_username = str(current_user.username)
        establecimiento = Establecimiento.objects.get(user__username__exact=current_username)
    except:
        return redirect('home')

    if request.method == 'POST':
        form = RegistrationFormAtencionUrgencia(request.POST)
        if form.is_valid():

            rut_paciente = form.cleaned_data['rut_paciente']
            fecha_atencion_hora = form.cleaned_data['fecha_atencion_hora']
            tratamiento = form.cleaned_data['tratamiento']
            observacion_urgencia = form.cleaned_data['observacion_urgencia']
            rut_encargado_urgencia = form.cleaned_data['rut_encargado_urgencia']
            nombre_encargado_urgencia = form.cleaned_data['nombre_encargado_urgencia']
            cargo_encargado_urgencia = form.cleaned_data['cargo_encargado_urgencia']
            codigo_establecimiento = establecimiento.codigo_establecimiento
            id_who= form.cleaned_data['id_who']
            paciente = Paciente.objects.get(rut_paciente=rut_paciente)
            enfermedad= Enfermedad.objects.get(id_who__exact=id_who)
            establecimiento=
            Establecimiento.objects.get(codigo_establecimiento=codigo_establecimiento)
            try:
                atenciones = AtencionPacienteAmbulatorio.objects.filter(rut_paciente=
rut_paciente).values('rut_medico').distinct()
                mails=[]
                for atencion in atenciones:
                    rut_medico=atencion['rut_medico']
                    medico = Medico.objects.get(rut_medico=rut_medico)
                    mails.append(medico.email)
                send_mail('Ingreso de paciente a Sala de Urgencias de paciente
'+str(paciente.rut_paciente)+' '+str(paciente.nombres)+' '+str(paciente.apellido_paterno)+'
'+str(paciente.apellido_materno) ,
                    'El paciente ha sido atendido en '+str(establecimiento.nombre)+' a causa de :
'+str(enfermedad.nombre)+' ', 'noreply@sistema.cl', mails)
            except:
                render(request, '../templates/app/problema_mail.html')
```

```

    new_atencion =
        AtencionPacienteUrgencia(fecha_atencion_hora=fecha_atencion_hora,rut_paciente=pa
ciente,tratamiento=tratamiento, observacion_urgencia=observacion_urgencia,
cargo_encargado=cargo_encargado_urgencia, rut_encargado=rut_encargado_urgencia,
codigo_establecimiento=establecimiento, id_who=enfermedad)
    new_atencion.save()
    return redirect('home')
else:
    form =RegistrationFormAtencionUrgencia()
return render(request, '../templates/app/ingresar_atencion_urgencia.html', {'form': form})

```

```
def consultarAtencionUrgencia(request):
```

```
    try:
```

```
        rut_paciente_session= request.session['rut_paciente']
```

```
        rut_limpio=rut_paciente_session.replace('.',',')
```

```
        paciente = Paciente.objects.get(rut_paciente=rut_limpio)
```

```
    except:
```

```
        return redirect('consultar paciente')
```

```
atenciones = AtencionPacienteUrgencia.objects.filter(rut_paciente__exact=rut_limpio)
```

```
enfermedades =[]
```

```
for atencion in atenciones:
```

```
    enfermedad = Enfermedad.objects.get(id_who=atencion.id_who)
```

```
    enfermedades.append(enfermedad)
```

```
lista = zip(atenciones,enfermedades)
```

```
return render(request, '../templates/app/listado_atencion_urgencia.html', {'lista': lista})
```

b) Ingresar Atención de Emergencia, Formulario.

```

class RegistrationFormAtencionUrgencia (forms.Form):
    fecha_atencion_hora = forms.DateTimeField( required=True,label='Fecha y Hora de la atención
    :', widget=forms.TextInput(attrs={'placeholder': 'Ej: 02/05/2016 15:30','class': 'form-control'}) ,
    input_formats=['%d/%m/%Y %H:%M'], )

    rut_paciente      =      forms.CharField(max_length=10,label='Rut      del      Paciente      :',
    widget=forms.TextInput(attrs={'placeholder':      'Ej:      9867014-9','class':      'form-control'}),
    required=True,validators=[rutregex])

    rut_encargado_urgencia = forms.CharField(max_length=10,label='Rut del encargado de la
    atención :',widget=forms.TextInput(attrs={'placeholder': 'Ej: 2219376-4','class': 'form-control'}),
    required=True,validators=[rutregex])

    nombre_encargado_urgencia=forms.CharField(max_length=30,label='Nombre del encargado de
    la atención :',widget=forms.TextInput(attrs={'placeholder': 'Ej: Yasna Fuentes','class': 'form-
    control'})), required=True,validators=[nombreregex])

    cargo_encargado_urgencia=forms.CharField(max_length=30,label='Cargo del encargado de la
    atención :',widget=forms.TextInput(attrs={'placeholder': 'Ej: Jefa de Urgencias','class': 'form-
    control'})), required=True, validators=[nombreregex])

    tratamiento      =      forms.CharField(max_length=800,label='Tratamiento      :',      required=True,
    widget=forms.Textarea(attrs={'placeholder': '', 'class': 'form-control'}))

    observacion_urgencia = forms.CharField(max_length=800,label='Observación de la atención de
    urgencia :', required=True, widget=forms.Textarea(attrs={'placeholder': '', 'class': 'form-control'}))

    choices      =      [(establecimiento.codigo_establecimiento,      establecimiento.nombre)      for
    establecimiento in Establecimiento.objects.all()], label='Establecimiento')

    id_who = forms.ChoiceField(required=True,choices=[(enfermedad.id_who, enfermedad.nombre)
    for enfermedad in Enfermedad.objects.all()], widget=forms.Select(attrs={'class': 'form-control'}),
    label='Enfermedad')

    def clean_fecha_atencion_hora(self):
        fecha_atencion_hora =self.cleaned_data['fecha_atencion_hora']
        fecha_hoy= datetime.today()
        fecha_hoy = fecha_hoy.replace(tzinfo=None)
        fecha_atencion_hora=fecha_atencion_hora.replace(tzinfo=None)
        if (fecha_atencion_hora) > (fecha_hoy):
            raise forms.ValidationError(_("La fecha/hora de la atención NO puede ser mayor a la
            fecha actual"), code='invalid')
        return fecha_atencion_hora

    def clean_rut_encargado_urgencia(self):
        rut_form=str(self.cleaned_data['rut_encargado_urgencia'])
        rut_limpio = rut_form.replace(".", "")

        digito_verificador = rut_limpio[-1:]
        if digito_verificador=="k" or digito_verificador=="K":

```

```

    digito_verificador = 10
    rut_limpio = rut_limpio[:-2]
    reversed_digits = map(int, reversed(rut_limpio))
    factors = cycle(range(2, 8))
    suma = sum(d * f for d, f in zip(reversed_digits, factors))
    digito_resultado= 11 -(suma%11)
    if digito_resultado==11:
        digito_resultado=0

    if int(digito_verificador)!=digito_resultado:
        raise forms.ValidationError(_("El rut no es valido"), code='invalid')
    return rut_form

def clean_rut_paciente(self):
    rut_form=str(self.cleaned_data['rut_paciente'])

    rut_limpio = rut_form.replace(".", "")

    digito_verificador = rut_limpio[-1:]

    if digito_verificador=="k" or digito_verificador=="K":
        digito_verificador = 10
    rut_limpio = rut_limpio[:-2]
    reversed_digits = map(int, reversed(rut_limpio))
    factors = cycle(range(2, 8))
    suma = sum(d * f for d, f in zip(reversed_digits, factors))
    digito_resultado= 11 -(suma%11)
    if digito_resultado==11:
        digito_resultado=0

    if int(digito_verificador)!=digito_resultado:
        raise forms.ValidationError(_("El rut no es valido"), code='invalid')

    #Verifica que el paciente EXISTA en el sistema
    if not Paciente.objects.filter(rut_paciente=rut_form).exists():
        raise forms.ValidationError(_('Este rut de paciente no existe en el sistema'),
            code='invalid')

    return rut_form

def clean (self):
    cleaned_data = super(RegistrationFormAtencionUrgencia,self).clean()
    rut_paciente= cleaned_data.get("rut_paciente")
    fecha_atencion_hora= cleaned_data.get("fecha_atencion_hora")

    if      AtencionPacienteUrgencia.objects.filter(Q(rut_paciente=rut_paciente)      &
        Q(fecha_atencion_hora=fecha_atencion_hora)):
        msg="Ya existe una atencion prestada al paciente en esta fecha/hora"
        self.add_error('fecha_atencion_hora',msg)

```

c) Consultar Atención de emergencia, Vista.

```
def consultarAtencionUrgencia(request):
    try:
        rut_paciente_session= request.session['rut_paciente']
        rut_limpio=rut_paciente_session.replace('.', '')
        paciente = Paciente.objects.get(rut_paciente=rut_limpio)
    except:
        return redirect('consultar paciente')

    atenciones = AtencionPacienteUrgencia.objects.filter(rut_paciente__exact=rut_limpio)
    enfermedades = []
    for atencion in atenciones:
        enfermedad = Enfermedad.objects.get(id_who=atencion.id_who)
        enfermedades.append(enfermedad)
    lista = zip(atenciones, enfermedades)
    return render(request, '../templates/app/listado_atencion_urgencia.html', {'lista': lista})
```

GENERADOR DE ESTADÍSTICAS.a) Generador de estadísticas, Vista.

```
def generarEstadisticas(request):
```

```
    if request.method == 'POST':
        form = GenerateStatisticsForm(request.POST)
        if form.is_valid():
            #Extraer la info del form
            region=form.cleaned_data['region']
            id_who=form.cleaned_data['enfermedad']
            edad=form.cleaned_data['edad']
            tipo_consulta=form.cleaned_data['tipo_consulta']
            sexo=form.cleaned_data['sexo']
            codigo_establecimiento=form.cleaned_data['establecimiento']
            anho=form.cleaned_data['anho']
            mes=form.cleaned_data['mes']

            enfermedad = Enfermedad.objects.get(id_who=id_who)

            filtros ={'id_who': id_who}

            #Aplicar filtros

            if sexo != "Cualquiera":
                filtros['rut_paciente__sexo__exact'] = str(sexo)
            if region!="Cualquiera":
                filtros['codigo_establecimiento__region__exact'] = str(region)
            if edad:
                filtros['rut_paciente__fecha_nacimiento__year']=
                    str(int(datetime.now(tz=None).year)-int(edad))
```

```

if codigo_establecimiento!="Cualquiera":
    filtros['codigo_establecimiento__codigo_establecimiento']=str(codigo_establecimiento)
    establecimiento =
        Establecimiento.objects.get(codigo_establecimiento=codigo_establecimiento)

if anho:
    filtros['fecha_atencion_hora__year']=str(anho)
if mes!="Cualquiera":
    filtros['fecha_atencion_hora__month']=str(mes)

if tipo_consulta == "Ambulatoria":
    atenciones_ambulatorias = AtencionPacienteAmbulatorio.objects.filter(**filtros)
elif tipo_consulta == "Urgencia":
    atenciones_urgencia = AtencionPacienteUrgencia.objects.filter(**filtros)
else:
    atenciones_ambulatorias = AtencionPacienteAmbulatorio.objects.filter(**filtros)
    atenciones_urgencia = AtencionPacienteUrgencia.objects.filter(**filtros)
cantidad_atenciones_ambulatorias=0
cantidad_atenciones_urgencias=0
#Realizar conteos
if 'atenciones_ambulatorias' in locals():
    cantidad_atenciones_ambulatorias= atenciones_ambulatorias.count()
if 'atenciones_urgencia' in locals():
    cantidad_atenciones_urgencias = atenciones_urgencia.count()
    promedio =
        round((cantidad_atenciones_urgencias+cantidad_atenciones_ambulatorias)/15,
            2)
total_atenciones_enfermedad=AtencionPacienteUrgencia.objects.filter(id_who=id_who).count() + \
        AtencionPacienteAmbulatorio.objects.filter(id_who=id_who).count()
try:
    porcentaje = round((cantidad_atenciones_urgencias +
        cantidad_atenciones_ambulatorias)/total_atenciones_enfermedad,2)
except:
    porcentaje = 0

#Traducir data

if sexo=="M":
    sexo="Masculino"
elif sexo=="F":
    sexo="Femenino"
elif sexo=="O":
    sexo="Otros"
return render(request, '../templates/app/resultado_estadistica.html', locals())
else:
    form = GenerateStatisticsForm()
    return render(request, '../templates/app/generar_estadisticas.html', {'form': form})

```

b) Generador de estadísticas, Formulario.

```
class GenerateStatisticsForm(forms.Form):
```

```
    list_regiones=(
        (1, _("I")),
        (2, _("II")),
        (3, _("III")),
        (4, _("IV")),
        (5, _("V")),
        (6, _("VI")),
        (7, _("VII")),
        (8, _("VIII")),
        (9, _("IX")),
        (10, _("X")),
        (11, _("XI")),
        (12, _("XII")),
        (13, _("RM")),
        (14, _("XIV")),
        (15, _("XV")),
        ("Cualquiera", _("Cualquiera")),
    )
    list_tipos=(("Ambulatoria", _("Ambulatoria")),
                ("Urgencia", _("Urgencia")),
                ("Ambas", _("Ambas")),
                )
    list_sexo=(("M", _("Masculino")),
               ("F", _("Femenino")),
               ("O", _("Otros")),
               ("Cualquiera", _("Cualquiera")),
               )
    list_meses=(("01", _("Enero")),
                 ("02", _("Febrero")),
                 ("03", _("Marzo")),
                 ("04", _("Abril")),
                 ("05", _("Mayo")),
                 ("06", _("Junio")),
                 ("07", _("Julio")),
                 ("08", _("Agosto")),
                 ("09", _("Septiembre")),
                 ("10", _("Octubre")),
                 ("11", _("Noviembre")),
                 ("12", _("Diciembre")),
                 ("Cualquiera", _("Cualquiera"))
                )
```

```

list_establecimiento=[]
list_establecimiento.append(("Cualquiera", _("Cualquiera")))

for establecimiento in Establecimiento.objects.all():
    list_establecimiento.append((establecimiento.codigo_establecimiento,
                                establecimiento.nombre))

list_establecimiento=tuple(list_establecimiento)

edad = forms.CharField(max_length=3,label='Edad',
                        widget=forms.TextInput(attrs={'placeholder': 'Ej: 23', 'class': 'form-control'}))
anho = forms.CharField(max_length=4,
                        label='Año',widget=forms.TextInput(attrs={'placeholder': 'Ej: 2016', 'class': 'form-control'}))
mes = forms.ChoiceField(required=True, choices=list_meses, label="Mes:", initial='1',
                        widget=forms.Select(attrs={'placeholder': 'Ej: jroman@gmail.com', 'class': 'form-control'}),help_text="")
tipo_consulta = forms.ChoiceField(required=True, choices=list_tipos, label="Tipo de Atención:", initial='1', widget=forms.Select(attrs={'class': 'form-control'}), help_text="")
sexo =forms.ChoiceField(required=True, choices=list_sexo, label="Sexo:", initial='1',
                        widget=forms.Select(attrs={ 'class': 'form-control'}), help_text="")
region = forms.ChoiceField(required=True,choices=list_regiones, label="Región:", initial='1',
                        widget=forms.Select(attrs={'placeholder': 'Ej: jroman@gmail.com', 'class': 'form-control'}),help_text="")
enfermedad = forms.ChoiceField(required=True, choices=[(enfermedad.id_who,
                                                        enfermedad.nombre) for enfermedad in
                                                        Enfermedad.objects.all()]),widget=forms.Select(attrs={'placeholder':
                                                        'Ej: jroman@gmail.com', 'class': 'form-control'}), label='Enfermedad:')
establecimiento=forms.ChoiceField(required=True,
choices=list_establecimiento,widget=forms.Select(attrs={'placeholder':
                                                        'Ej: jroman@gmail.com', 'class': 'form-control'}), label='Establecimiento:')

```