


Sistema para prescribir apps a pacientes para el seguimiento de sus enfermedades crónicas: caso de uso en insuficiencia cardíaca

SIMIC 2.0

Michell Mamrut
Guillermo Alvez
Romina Pons

Directores:
Prof. Ing. Franco Simini
Prof. Ing. Antonio López

Proyecto de Grado de Ingeniería en Computación

Facultad de Ingeniería
Universidad de la República
Montevideo – Uruguay
Junio de 2021

Agradecimientos

En primer lugar, queremos agradecer a nuestras familias y amigos por el apoyo incondicional recibido desde que comenzamos con la carrera.

A los tutores del proyecto, Prof. Ing. Antonio López y Prof. Ing. Franco Simini, quienes demostraron compromiso, aportando sus experiencias y sus conocimientos para guiarnos en el transcurso de esta etapa tan importante para nuestras carreras.

A la Universidad de la República, y a todo su equipo docente, a quienes debemos la formación que nos permitió afrontar con seguridad los desafíos que surgieron durante el proyecto.

A los integrantes de la Unidad Multidisciplinaria de Insuficiencia Cardíaca (UMIC) del Hospital de Clínicas, en especial a sus médicos Gabriela Ormaechea, Pablo Álvarez y Gabriela Silvera, por su compromiso, dedicación y enseñanzas obtenidas en el tiempo compartido junto a ellos.

A los estudiantes de Medicina Lucía Belén Ribeiro, Isabel Ribeiro y Valentina Fernández Francese.

A los integrantes del Núcleo de Ingeniería Biomédica Franco Vienni y Hernán Castillo.

A todos ellos, nuestro más sincero agradecimiento.

Resumen

SIMIC 2.0 es un sistema de prescripción de reglas (recetas) para el seguimiento de pacientes. Este sistema introduce dos nuevos conceptos, el primero es recetar una app para el seguimiento del paciente a distancia y el segundo es el médico programando recetas en lenguaje Python para el seguimiento.

Como caso de estudio se toma a pacientes con insuficiencia cardíaca, pero el diseño fue pensado para que funcione con el seguimiento de cualquier patología.

La motivación surge de parte del Núcleo de Ingeniería Biomédica y el equipo médico de la UMIC (Unidad Multidisciplinaria de Insuficiencia Cardíaca), que atiende a pacientes con insuficiencia cardíaca.

Como solución se implementa una aplicación móvil que es utilizada por el paciente para realizar el seguimiento a distancia y una aplicación web que permite ver la información del seguimiento, las alertas generadas y también permite prescribir recetas de seguimiento para el mismo.

Palabras claves:
Insuficiencia Cardíaca, seguimiento del paciente, prescripción de apps, complemento de historia clínica

Índice

Índice de figuras

Capítulo 1

Introducción

1.1 Problema

Hoy en día los pacientes con enfermedades crónicas concurren a consultas presenciales en los centros de salud y los médicos le recetan fármacos, exámenes de laboratorio, entre otras cosas. Esto se da con distintas frecuencias, pero la realidad es que estas personas necesitan de un seguimiento continuo. En patologías de este tipo de una consulta a la siguiente pudieron haberse agravado los síntomas y el médico recién enterarse en el consultorio de esta situación.

Las enfermedades crónicas no transmisibles (ECNT) tienden a ser de larga duración y resultan de la combinación de factores genéticos, fisiológicos, ambientales y conductuales. Para prevenir y controlar las ECNT se debe reducir los factores de riesgo asociado a ellas. También puede ayudar la detección temprana y el tratamiento a tiempo por los centros de salud. Existen factores de riesgo asociados al comportamiento que son modificables, por ejemplo el consumo de tabaco, la inactividad física, las dietas no saludables y el uso nocivo del alcohol. Por otro lado, se tienen los cambios metabólicos como el aumento de la tensión arterial, el sobrepeso, la hiperglucemia y la hiperlipidemia que contribuyen al aumento del riesgo de las ECNT. Todos estos factores son cada vez más comunes en la vida de las personas, lo hace que se vea en aumento el número de casos de las ECNT y se torna un reto cada vez mayor para los integrantes de la salud. Para controlar estos factores se apunta a un seguimiento integral del paciente y hacer recomendaciones oportunas que mejoren su calidad de vida [42].

Algunas de las patologías que requieren este seguimiento son por ejemplo: la insuficiencia cardíaca, diabetes, entre otras.

Si bien el paciente tiene noción de su enfermedad, ya que son patologías que son tratadas durante toda la vida, estas tienen varios parámetros que pueden hacer que la persona se descompense rápidamente.

En una situación ideal el médico debería estar presente diariamente y poder ajustar dosis de medicamentos según los síntomas del paciente. Esto es inviable, no solo por la relación en cuanto a cantidad de médicos por cantidad de pacientes, sino que tampoco es posible para todas las personas concurrir asiduamente al centro de salud. A su vez, los parámetros que deben ser monitoreados son de fácil acceso para cualquier persona, como por ejemplo el peso, azúcar en sangre, o la presión sanguínea.

1.2 Motivación

Desde el Núcleo de Ingeniería Biomédica surge la idea de poder recetar “reglas de seguimientos”, para que el paciente pueda, desde su celular, seguir una rutina de seguimiento que fue escrita en lenguaje de alto nivel por un médico. Es así como nace el proyecto de grado presentado en este informe.

La idea abarca tanto el poder prescribir en el consultorio una o más reglas de seguimiento al paciente, como también propone que estas reglas hayan sido previamente escritas en un lenguaje de alto nivel por el cuerpo médico, que son los que tienen el conocimiento de las patologías tratadas (es decir, que los médicos programan las reglas).

SIMIC 2.0 es entonces una nueva herramienta para el médico de la que antes no disponía, que promueve el uso intensivo de las Tecnologías de la Información y la Comunicación (TIC) en el sector de la salud para mejorar la calidad y continuidad asistencial de cada paciente. SIMIC 2.0 constituye una posibilidad de mejorar la relación médico-paciente al recordarle al paciente hábitos saludables y al recolectar información de la vida diaria para enriquecer la consulta posterior.

1.2.1 Prescripción de apps

El caso de estudio que motiva este proyecto fue específicamente la insuficiencia cardíaca, pero el objetivo fue poder generalizar los aspectos que abarcaban a esta enfermedad para poder utilizar esta herramienta en otros casos similares, lo que significó un desafío en sí mismo. Esto introduce una innovación única, agregando el término de Prescribir una App. El médico hasta el día de hoy únicamente tenía la opción de recetar medicamentos, y de indicar seguimientos sin tener control sobre cómo lo llevaba a cabo el paciente. Esta nueva idea tiene el objetivo de que el médico pueda recetar en las consultas una aplicación móvil que será un complemento a las indicaciones del médico, y también la fuente de datos para su control.

1.3 Objetivos

Los objetivos del proyecto sobre Prescripción de reglas de seguimiento que se definieron para poder cumplir con las necesidades planteadas son:

- Desarrollar una plataforma donde se puedan codificar las reglas de seguimiento, administrar las reglas asignadas a cada paciente y gestionar las alertas hacia los médicos.
- Implementar una aplicación móvil para el uso de los pacientes, donde verán las distintas preguntas y recomendaciones de seguimiento, y desde allí podrán también brindar sus respuestas.
- Se debe contar con la posibilidad de extender la plataforma a otros casos de uso, es decir, a otros tipos de seguimiento, no restringiendo únicamente al seguimiento de pacientes con insuficiencia cardíaca.
- Generar una serie de reportes con los datos de SIMIC en Google Data Studio, para mejor visualización y análisis de los datos del sistema, ya que la UMIC presentó la dificultad en la extracción de los datos de SIMIC 1.0. La necesidad incluye un sistema de análisis de datos clínicos, que pueda ser configurado y explotado por los propios usuarios del sistema.
- Enseñar a los médicos el manejo de la herramienta para que sean capaces de generar sus nuevos reportes, utilizando como fuente de datos principal la base de datos de SIMIC 1.0.

1.4 Idea general de la solución

Se introducen conceptos importantes en el proyecto, como el **médico autor**, que será el encargado de interpretar y transcribir a código python las necesidades de seguimiento, generando así una serie de reglas. Y los **médicos clínicos** quienes desde el consultorio recetan estas reglas de seguimiento a los pacientes.

Primeramente, de forma interdisciplinaria se realizará el análisis del seguimiento que es necesario realizarle al paciente, esto luego será expresado en un diagrama de flujo como una primer manera de formalizar dicho análisis.

Una vez que se tiene el diagrama, el médico autor será el encargado de ingresar las reglas en la interfaz web de SIMIC 2.0, traduciendo el diagrama a código Python.

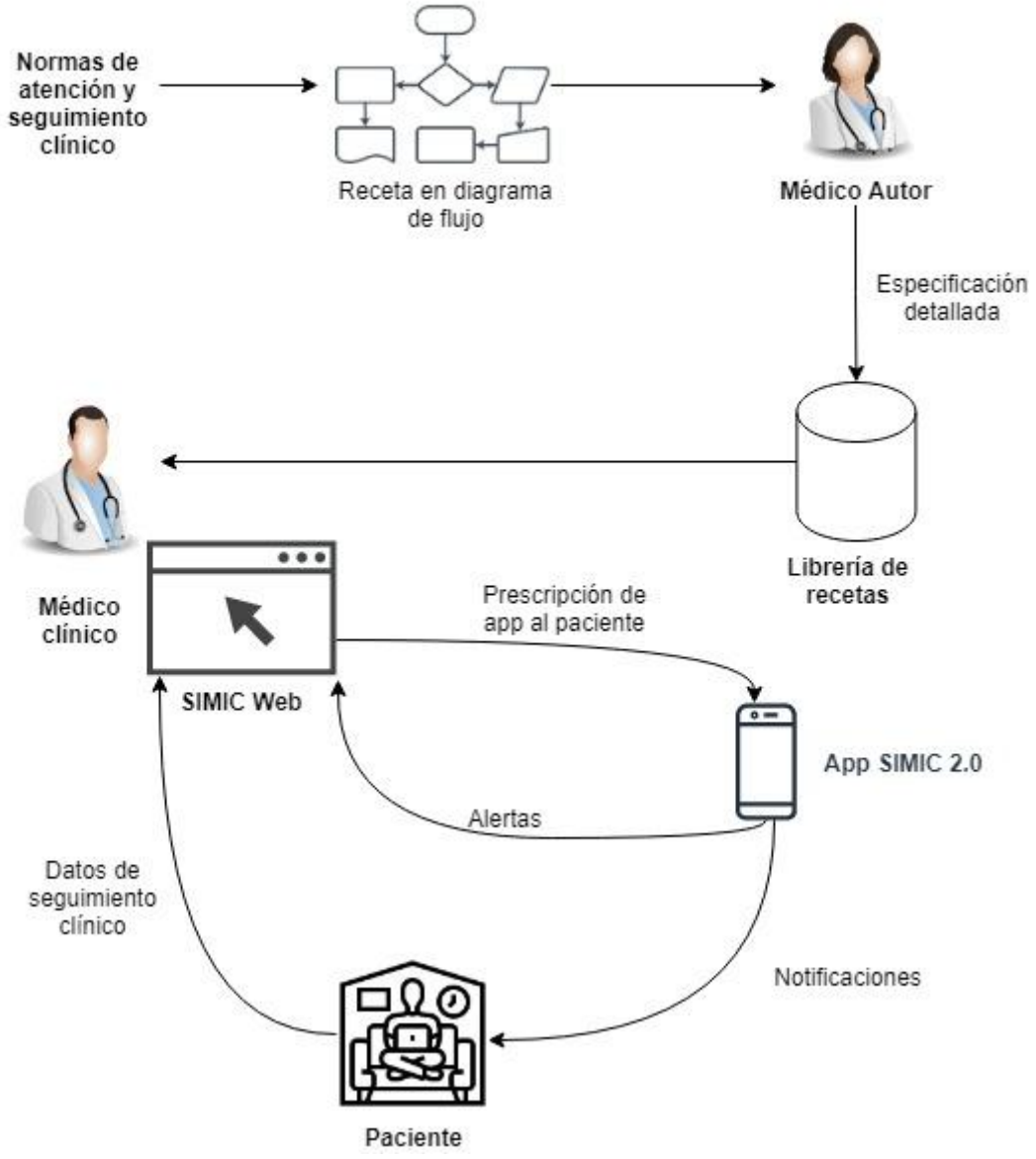


Figura 1: diagrama general de SIMIC 2.0

Una vez que estas reglas son ingresadas, y el paciente concurre a la consulta, el médico clínico le receta la app SIMIC 2.0, e ingresa el seguimiento personalizado que tendrá el paciente, esto lo hace en la web SIMIC 2.0 eligiendo las reglas que le asignará al paciente. Cuando el paciente se lleva la app, este va a contribuir con datos que le van a ser útiles al médico para el seguimiento a distancia. En este seguimiento la app le podrá sugerir (conforme lo programado en la regla) hábitos, medicamentos, concurrir a una nueva consulta, entre otros.

1.5 Resultados esperados

Los resultados de este proyecto son variados dado su enfoque multidisciplinario.

Por un lado se espera un mejor seguimiento de pacientes con enfermedades crónicas, y así poder mejorar su calidad de vida.

Por otro lado se apunta a introducir un nuevo concepto en el mundo de la medicina, como el poder recetar aplicaciones móviles. Con este resultado se pretende dar el primer paso hacia lo que será una nueva modalidad de trabajo para el médico.

También se espera que el médico pueda acostumbrarse al mundo de la programación y que no sea una tarea específica solamente de los informáticos, siendo esto un aporte a la informática médica, dado su carácter inter-disciplinario.

1.6 Desafíos del trabajo

El equipo espera encontrarse con los siguientes desafíos:

- **Diferencia de lenguaje entre la disciplina de ingeniería y la médica:** Todo trabajo multidisciplinario implica cierta barrera en el entendimiento que parte de lenguajes diferentes, esto hace que la comunicación interdisciplinaria se vea afectada, la medicina y la ingeniería no son la excepción. El equipo pondera este punto como el de mayor importancia, siendo crucial la reducción de la brecha a la hora del análisis y especificación de los requerimientos, que en definitiva, derivará en la implementación y posterior aceptación del sistema.

- **Intereses contrapuestos entre los tutores del proyecto, los estudiantes y los médicos:** Encontrar el equilibrio entre distintos intereses es difícil, los proyectos de grados generalmente cuentan con un cliente ficticio o con cierta idea dada por él/los tutores, lo que constituye un ida y vuelta más que nada entre los 2 actores. El proyecto SIMIC 2.0 además cuenta con la expectativa del cliente (los médicos), esperándose que el sistema pueda utilizarse como herramienta en un escenario real de seguimiento al paciente. Se considera muy probable que existan diferencias entre lo que necesita la unidad médica con los intereses de los tutores, es un gran desafío encontrar el equilibrio que sea satisfactorio y valioso para todos.

- **Continuación de un proyecto de grado:** El sistema SIMIC 1.0 dejó cierto trabajo a futuro, la dificultad radica en entender lo que ya existe y lo que se necesita.

- **Tecnologías diferentes:** en el ámbito técnico el equipo eligió un proyecto diferente a lo que venía acostumbrado tanto académica como laboralmente, los integrantes del equipo vienen de ramas un poco distintas, como ser la seguridad informática y el

desarrollo más que nada enfocado al backend. En una tentativa de salida de su zona de confort y exploración de nuevas disciplinas, se procura mejorar los conocimientos (y a su vez aportar valor con ello) en lenguajes como python, el desarrollo frontend, y el desarrollo mobile.

1.7 Organización del documento

El documento se organiza en capítulos.

El primero de ellos es la introducción, donde se aborda la motivación del problema a resolver, los antecedentes y resultados esperados.

Luego el capítulo 2 aborda los principales conceptos para comprender el documento. Aquí se puede encontrar conceptos médicos como la insuficiencia cardíaca y también conceptos técnicos que son utilizados en otros capítulos como el motor de ejecución de recetas.

El capítulo 3 contiene el análisis y diseño de la solución, en particular se ven los objetivos, el alcance del proyecto y también la especificación funcional.

El capítulo 4 explica las decisiones más importantes que se tomaron en el proyecto, la arquitectura, el modelo de dominio y los principales casos de uso con sus respectivos diagramas de secuencia.

El capítulo 5 especifica detalles de la implementación de los componentes, contiene traducciones de diagrama de flujo a código, y se muestran ejemplos de los capítulos anteriores.

El capítulo 6 corresponde a las pruebas que fueron realizadas. Estas incluyen pruebas unitarias, pruebas exploratorias, pruebas de integración y pruebas de aceptación.

El capítulo 7 muestra detalles de la gestión del proyecto y la metodología que se decidió utilizar.

El capítulo 8 da un cierre al trabajo, presentando conclusiones generales, lecciones aprendidas y proponiendo mejoras futuras.

El capítulo 9 muestra las referencias.

Y por último en el capítulo 10 se dejan como anexos el manual del médico clínico, médico autor, de implantación y el video presentado en ingeniería de muestra.

Capítulo 2

Estado del arte

En esta sección se detallarán los principales conceptos para comprender el contexto del trabajo realizado. Es necesario abarcar varias disciplinas para entender el objetivo y el camino que llevó hacia el resultado final del proyecto. Se especificará sobre el uso de la tecnología en la rama de la medicina y el contacto del personal médico con la misma. Se estudiará un caso de estudio específico como es el de la insuficiencia cardíaca, qué es y cuál es el tratamiento necesario para esta patología. Dentro de este caso de estudio se hablará sobre el centro multidisciplinario que se ocupa de tratar a los pacientes, y de su influencia en los desarrollos tecnológicos aquí planteados.

También en este marco contextual, se expondrán las problemáticas encontradas al momento de iniciar a desarrollar el proyecto y las soluciones tecnológicas pensadas en un comienzo para llevarlo a cabo. Luego se verá la necesidad de que el desarrollo se haga desde un comienzo de una manera genérica, sin centrarse en el caso de estudio utilizado para la implementación.

2.1 Seguimiento continuo en enfermedades crónicas

Las enfermedades crónicas son aquellas de larga duración y en general de progresión lenta, entre ellas se encuentran [43]:

- Enfermedades cardiovasculares
- Diabetes
- Enfermedades neoplásicas
- Enfermedades respiratorias crónicas (EPOC),
- Enfermedades osteoarticulares
- Enfermedades invalidantes

- HTA (Hipertensión Arterial)

Vivir con una enfermedad crónica supone un reto importante para la persona, ya que es una condición que la acompañará en el largo plazo. Hoy en día, las enfermedades crónicas han ido tomando importancia debido al aumento de su prevalencia pero también a la dificultad para su control y seguimiento, según el artículo titulado “Programa para mejorar la atención de las enfermedades crónicas. Aplicación del Modelo de Cuidados para Enfermedades Crónicas” [44] publicada por la editorial de libros de medicina “Elsevier”, la dificultad para el control de las enfermedades crónicas tiene algunos problemas a resolver, entre ellos están que “El sistema sanitario está preparado para la atención reactiva de problemas agudos, reagudizaciones y complicaciones, no para la atención proactiva de los enfermos crónicos” y por otro lado “El control de las enfermedades crónicas requiere grandes cambios en el comportamiento de los pacientes, los sanitarios y la organización del sistema.”

Por tanto, es necesario que el paciente conozca la enfermedad que enfrenta y sea partícipe de su control y seguimiento periódico, el artículo anterior menciona además que “El Modelo de Cuidados para Enfermedades Crónicas enfatiza la participación y un papel de decisión principal al paciente a través del Servicio de Salud. El concepto de empowerment, de capacitar al paciente para responsabilizarse de su situación de salud, es el gran desafío en la implementación de este modelo” [44].

2.2 Insuficiencia cardíaca y contexto en Uruguay

La IC (insuficiencia cardíaca) es un complejo síndrome clínico vinculado a alteraciones fisiopatológicas, que ocurre cuando una anormalidad de la función cardíaca determina la incapacidad del corazón para cumplir con las necesidades metabólicas tisulares, o lo hace a expensas de un incremento en las presiones diastólicas finales. La misma puede deberse a múltiples potenciales etiologías, siendo la posible vía final común que alcanzan las diferentes cardiopatías estructurales según su gravedad y tiempo de evolución.

Es una enfermedad de creciente incidencia y prevalencia, de elevada morbimortalidad, determinante de elevados costos en salud [1].

La importancia que adquiere el diagnóstico etiológico de la IC está íntimamente relacionada con la adopción de conductas terapéuticas específicas, que potencialmente podrían revertir (al menos en parte), o enlentecer el deterioro de la función cardíaca [1].

En Uruguay la definición precisa de Unidades de IC aún no está completamente desarrollada. El objetivo de las mismas es el manejo integrado de la enfermedad. Lo que se busca con las mismas es, mediante un equipo multidisciplinario, trabajando en forma coordinada, encargarse del control y seguimiento de los pacientes con IC en las diferentes etapas de la enfermedad, fundamentalmente en las más complejas, y en los diferentes niveles de la cadena de atención sanitaria que vive el paciente ya sea extra e intrahospitalaria.

El informe “Una mirada a la salud de los uruguayos y las uruguayas en el largo plazo” [45] realiza un análisis sobre las consecuencias de la transición demográfica y la transición epidemiológica sobre la salud de la población uruguaya y la sostenibilidad del sistema de salud en el largo plazo, tanto en términos de modelo asistencial como financiero. A su vez, identifica lineamientos estratégicos para el sector salud a ser incluidos en la Estrategia Nacional de Desarrollo, Uruguay 2050, y a considerar en las estrategias, planes y demás instrumentos de planificación de políticas públicas.

Según el informe mencionado anteriormente, el estilo de vida moderno ha traído consigo problemas explicados por la falta de ejercicio, desórdenes en la alimentación, demasiada ingesta de grasas saturadas, sal y a su vez poco consumo de frutas y verduras. Esto en conjunto con el tabaquismo, entre otras conductas, han constituido una epidemia de sobrepeso, hipertensión y diabetes. Todos estos factores influyen altamente en las enfermedades del tipo cardiovasculares.

Uruguay lamentablemente acompaña la tendencia mundial, sumándole al problema una población envejecida proyectándose que el porcentaje de mayores de 65 años pasaría de 14% en 2016 al 22,9% para 2050 [45] sumado al hecho de que el 64,9% de la población tiene sobrepeso u obesidad, 36,6% padece hipertensión arterial y 21,5% presenta colesterol elevado. Estos factores hacen que la salud pública y privada deba poner especial foco en las enfermedades no transmisibles (ENT) entre ellas, las enfermedades cardiovasculares, ya que al aumentar el peso relativo de las personas mayores es de esperarse que también lo haga el número de casos totales de ENT.

En la siguiente imágen podemos ver la mortalidad proporcional en Uruguay, cifras tomadas del año 2015.

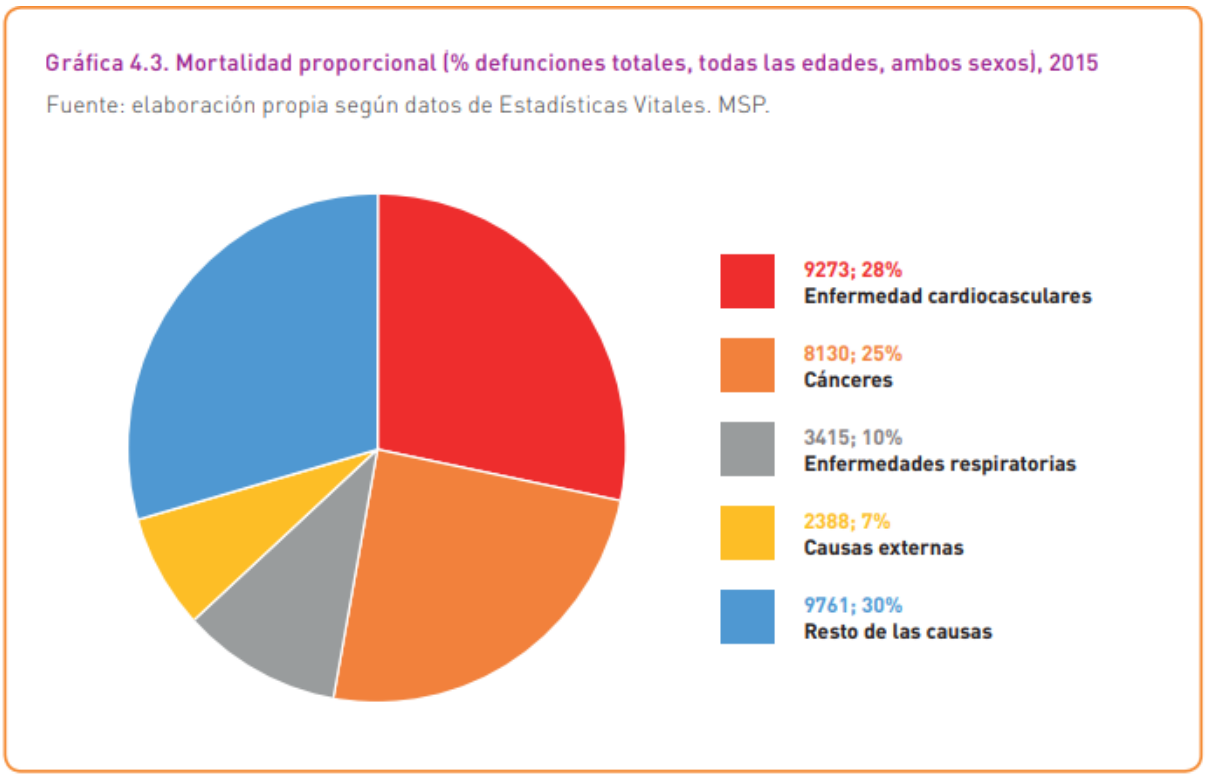


Figura 65: Gráfica de mortalidad proporcional en Uruguay

La Unidad Multidisciplinaria de Insuficiencia Cardíaca (UMIC) está integrada por un grupo de profesionales de la salud pertenecientes al Hospital de Clínicas “Dr. Manuel Quintela” de la República Oriental del Uruguay, quienes conforman un equipo multidisciplinario, de trabajo interdisciplinario, bajo la concepción de los programas de manejo de enfermedades crónicas, para atender a los pacientes portadores de insuficiencia cardíaca (IC) crónica por disfunción sistólica.

El método de trabajo es el de una policlínica especializada en la cual participan médicos internistas, cardiólogos, psiquiatras, ecocardiografistas, nutricionistas, licenciados en enfermería y asistentes sociales.

Estas unidades, han mostrado a nivel internacional, grandes beneficios, fundamentalmente en lo que tiene que ver con el mayor conocimiento de la enfermedad por el paciente y familia, adhesión al tratamiento, disminución franca de las hospitalizaciones, así como del número de días de internación, aunque aún no se pueda extraer datos en cuenta al impacto en la sobrevida [2].

2.3 Seguimiento de la insuficiencia cardíaca

Varios estudios han demostrado el impacto negativo en la evolución de la IC que tiene un pobre seguimiento médico, así como un tratamiento inadecuado. Por otra parte, en los

últimos 10 años ha tomado fuerza el concepto de la IC como un síndrome clínico complejo que requiere el control periódico de un equipo multidisciplinario lo cual, llevado a la práctica, ha demostrado claros beneficios en la reducción de reingresos hospitalaria por esta patología.

Los pacientes con IC crónica y sus familiares deben recibir información y consejos generales sobre la enfermedad, a cargo del médico tratante o de un equipo destinado a tal fin. La educación constituye un pilar fundamental del tratamiento de esta entidad. El manejo no farmacológico del paciente con IC puede tener un gran impacto en la estabilidad del mismo, en su capacidad funcional, en la mortalidad y en su calidad de vida.

Existen hábitos y una base alimenticia fuertemente recomendada para que el paciente pueda tener una buena calidad de vida y bajar los reingresos hospitalarios.

Una de las causas más frecuentes de descompensación de la IC es el incumplimiento de la dieta es por esto que se debe controlar periódicamente. Luego de establecido el diagnóstico de IC, se le va a informar al paciente de una dieta alimenticia estricta. Dentro de esta dieta se encuentran restringidos los alimentos con exceso de sodio y de grasas saturadas, los líquidos están limitados en su cantidad diaria y la ingesta de alcohol y el consumo de cigarros están altamente desaconsejados [8].

Una ingesta alta de sal puede causar aumentos bruscos de peso y ser una causa frecuente de descompensaciones. Es por esto que se requiere un control estricto del peso como signo precoz de retención hidrosalina. Por lo tanto, se debe recomendar controlar el peso de manera diaria y, en caso de observarse un aumento de peso en pocos días, es posible que se deba aumentar la dosis de diuréticos o eventualmente consultar a su médico.

2.4 Las aplicaciones móviles de salud en el modelo de cuidados centrado en el paciente

En la sociedad actual, las TIC se han convertido en una herramienta fundamental para el acceso a la información y el mantenimiento de las comunicaciones, configurándose así Internet como una fuente importante de conocimiento. Todos los sectores, incluido el sector sanitario, han sufrido el impacto de este avance tecnológico, adaptándose a los cambios e incorporando diferentes herramientas disponibles. Según los datos de la última Encuesta sobre equipamiento y uso de tecnologías de información y comunicación en los hogares del Instituto Nacional de Estadística (INE), en 2018 el 83% de las viviendas tenían acceso a Internet, estando equipadas en un 71.2% con algún tipo de ordenador y en un 89% con teléfono móvil. En cuanto a su utilización, se constataba que un 78% de la población accede a Internet al menos una vez a la semana y el dispositivo más usado para ello es el teléfono móvil (96.3%) [10]. La incorporación en el ámbito de la salud de las TIC está provocando un cambio en el paradigma de la atención sanitaria, convirtiéndose en una potente herramienta de apoyo al paciente y consiguiendo, a su vez, la optimización de los recursos. En concreto, las aplicaciones móviles de salud (apps de salud) están dando lugar a la creación de una nueva dimensión en las relaciones profesional-paciente, ya que se configuran como un

espacio abierto de comunicación, favoreciendo el intercambio de información de manera fácil y comprensible. Esto convierte a las apps de salud en importantes sistemas de apoyo para el paciente crónico, pues ofrecen distintos servicios que facilitan los procesos de información, educación, gestión y relación social, los cuales son esenciales para una adecuada autogestión de la salud [9].

2.5 Software en la medicina

En la medicina se maneja gran cantidad de información sobre los pacientes, proveniente de fuentes muy variadas. El uso de software para la manipulación de esta información aporta un acceso rápido, fiable (sin introducción de errores humanos en la comprensión de los formularios usados en la recolección de datos) y además, la realización de respaldos que servirán para la posibilidad de pérdidas.

En Uruguay en su visión de Gobierno Electrónico, AGESIC [37] entendió prioritario abordar el área de la Salud con el fin de aportar una fuerza propulsora para la modernización de los procesos, avanzar en la aplicación de las políticas de gobierno en el área y viabilizar mejoras en la calidad de las prestaciones de salud recibidas por los ciudadanos.

Para implementar esta estrategia, se firmó un Convenio entre Presidencia, AGESIC, el Ministerio de Salud Pública y el Ministerio de Economía y Finanzas, a través del cual se creó el Programa Salud.uy fuerza ejecutiva a cargo de la instrumentación de la iniciativa.

El objetivo del Programa Salud.uy es fortalecer el Sistema Nacional Integrado de Salud (SNIS) apoyando la conformación de la red asistencial a través del uso de las Tecnologías de la Información y la Comunicación (TIC), creando herramientas que contribuyan a mejorar el acceso de los ciudadanos a servicios de salud de calidad.
Su finalidad es desarrollar los mecanismos, las acciones transversales y la infraestructura tecnológica necesaria, a los efectos de generar las condiciones para que los prestadores de salud puedan prestar sus servicios en forma integrada, complementaria y centrada en el usuario.

Historia Clínica Electrónica Nacional (HCEN), se trata de una herramienta que permite el almacenamiento, transferencia y consulta de información sobre la prestación de servicios de salud y datos clínicos del usuario. Entre sus cometidos está promover la continuidad de la atención sanitaria y la calidad del registro a través de la normalización de las estructuras clínicas, así como generar una base sustantiva de información clínica que permita complementar los servicios asistenciales y su prestación a distancia.

Uno de sus principales objetivos es garantizar que la información clínica vital del ciudadano esté disponible y accesible para el profesional de la salud, de forma oportuna, segura y en línea.

2.6 Antecedentes

2.6.1 SIMIC 1.0

SIMIC, proyecto de grado que realizó una plataforma web enfocada a centralizar los datos de los pacientes de la Unidad Multidisciplinaria de Insuficiencia Cardíaca (UMIC) del Hospital de Clínicas. Funciona como historia clínica de cada paciente y ayuda a gestionar sus antecedentes, enfermedades y tratamientos [3]. SIMIC tenía una aplicación móvil en la cual hacía un seguimiento del paciente muy básico y restringido para la patología nombrada anteriormente. Este sistema generaba alertas a la web para informar al médico de alteraciones del paciente y tenía generación de indicadores en tiempo real para ayudar a la toma de decisiones del equipo médico.

Este sistema tenía la idea del seguimiento a distancia pero al ser muy básico fue restrictivo y no se podía modificar el seguimiento a menos que se modificara el código fuente del mismo.

Algunas de las funcionalidades principales de SIMIC 1.0 son:

- Consulta de pacientes
- Indicadores de gestión de la atención
- Agenda de médicos
- Integración con estándar de historia clínica electrónica (XDS [38])
- Integración con identificador único para personas (EMPI [39])
- Integración con servidor terminológico (SNOMED [40])

El sistema cuenta con dos tipos de roles que permiten vínculos con otros sectores del sistema con los cuales la UMIC tiene relación y que pueden ser intensificadas usando SIMIC:

- Médico UMIC: son los médicos que trabajan en UMIC en el hospital de clínicas. Podrán ver todos los pacientes ingresados en el sistema.
- Médico Externo: médicos que atienden a los pacientes que ingresaron a UMIC y además, realizan controles en policlínicas externas a UMIC. Solo podrán ver los pacientes asignados tanto como médico responsable o tratante.

2.6.2 MediSafe

Se trata de una aplicación que ayuda a recordar la toma de medicamentos mediante el envío de notificaciones. Es especialmente útil para seguir el tratamiento de enfermedades crónicas. Permite gestionar gran número de afecciones y se puede sincronizar con los

pastilleros de familiares para controlar su medicación [4]. Esta aplicación mejora la continuidad asistencial del paciente.

2.6.3 RecuerdaMed

Sirve de recordatorio para la toma de fármacos y ayuda a llevar el control, siendo especialmente útil en las enfermedades crónicas y en pacientes polimedicados (tanto para los pacientes como para los profesionales sanitarios). Cuenta con el aval del Observatorio para la Seguridad del Paciente de la Consejería de Salud de la Junta de Andalucía, la Red Ciudadana de Formadores en Seguridad del Paciente y de la Agencia de Calidad Sanitaria de Andalucía [5].

2.6.4 Social Diabetes

Sirve como herramienta de ayuda al control de la diabetes tipo 1 y 2, resultando útil a los pacientes para calcular las dosis de hidratos y la administración de insulina, así como a los médicos, que pueden controlar de forma remota los parámetros de sus pacientes [6]. Esta aplicación ayuda al seguimiento a distancia pero solamente de pacientes con la enfermedad de Diabetes, si se quisiera utilizar para otra enfermedad crónica se debería adaptar la misma y modificar el código.

2.6.5 Welvi

Esta app ayuda a promover hábitos saludables para conseguir una vida activa y seguir una dieta sana a través de una cuidada planificación y personalización. También permite mejorar el entrenamiento registrando parámetros físicos y aporta consejos útiles en función de las características del usuario. Está respaldada por un amplio equipo de médicos, nutricionistas, fisioterapeutas, entrenadores y psicólogos [7].

2.6.5 eCardioSurf

Ecardiosurf es una App para el manejo de la enfermedad en pacientes con insuficiencia cardíaca. El desarrollo consiste en una aplicación móvil para el autocontrol de signos y síntomas de insuficiencia cardiaca por parte del paciente, de forma independiente. Los datos son, a su vez, analizados por la unidad de cardiología del Hospital de Basurto en tiempo real.

La aplicación permite al paciente, además de registrar y visualizar sus datos de salud, obtener en su teléfono móvil recomendaciones por parte de sus cardiólogos de referencia; recomendaciones que llegan al teléfono móvil a modo de notificaciones [41].

<u>Conclusiones de antecedentes:</u>
Todas estas aplicaciones ayudan particularmente a una enfermedad crónica pero ninguna es extensible para el seguimiento del resto de las enfermedades.
Por otra parte, cada una de estas aplicaciones tiene su seguimiento ya programado en el código fuente de la misma y no es posible la modificación del seguimiento a no ser que se programe nuevamente.
Por lo tanto se concluye que estas aplicaciones no satisfacen los objetivos del proyecto.

2.7 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI) [13]. Generalmente, un IDE cuenta con las siguientes características:

- Editor de código fuente: editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico del lenguaje y la comprobación de errores a medida que se escribe el código.

- Automatización de compilación local: herramientas que automatizan tareas sencillas e iterativas como parte de la creación de una compilación local del software para su uso por parte del desarrollador, como la compilación del código fuente de la computadora en un código binario, el empaquetado del código binario y la ejecución de pruebas automatizadas.

- Depurador: programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

VS Code (Visual Studio Code) es un entorno de desarrollo implementado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto [14].

En primera instancia, cumpliría con las necesidades del proyecto que serán detalladas más adelante.

2.8 Motores de ejecución de reglas

2.8.1 Nuevo lenguaje como motor de ejecución de reglas

La idea es crear un nuevo lenguaje para especificar las reglas de seguimiento para la insuficiencia cardíaca. El parser de este lenguaje debía ser implementado en java y se deberían tener en cuenta todos los elementos necesarios que se pueden tener en el diagrama de flujo para pasarlo al lenguaje de alto nivel que se quiere generar [15].

2.8.2 Python como motor de ejecución de reglas

Python es uno de los lenguajes de programación más usados actualmente y la tendencia sigue aumentando. Es de código abierto, una sintaxis sencilla y fácil de entender, por lo que

ahorra tiempo y recursos. Es uno de los mejores para iniciarse en el mundo de la programación.

2.9 Diagrama de flujo

El diagrama de flujo, también conocido como flujograma, es una herramienta utilizada para representar la secuencia de las actividades en un proceso. Para ello, muestra el comienzo del proceso, los puntos de decisión y el final. Todo esto proporciona una visualización del funcionamiento del proceso, volviendo la descripción más clara. Esta herramienta expresa el flujo de la información, las derivaciones del proceso y el número de pasos.

Es una de las siete herramientas básicas de gestión de calidad [16]. Su objetivo principal es asegurar la calidad y aumentar la productividad del equipo. El diagrama de flujo se puede utilizar para desarrollar y mejorar la presentación gráfica de un proceso y para identificar el costo de la calidad (COQ). Estos son sus beneficios:

Control de calidad. El diagrama de flujo se utiliza como una herramienta para identificar actividades sin valor agregado en la ejecución del proceso y, de ese modo, mejorar el rendimiento.

Visión transparente. El diagrama de flujo mejora la comprensión del proceso. La diagramación hace posible aprender el conjunto de actividades, relaciones e incidencias de un proceso, enfocándose en aspectos específicos del mismo.

Identificación de clientes. Gracias al diagrama de flujo, es más fácil conocer las necesidades de los clientes y ajustar el proceso hacia la satisfacción de sus necesidades y expectativas.

Comunicación eficaz. El diagrama introduce un lenguaje común que mejora la comunicación de todo el equipo. Para ello, se deben realizar capacitaciones a los profesionales que lo administrarán.

Mejora de tiempos y costes. El diagrama de flujo facilita la aplicación de acciones en la optimización del tiempo y los costes de actividad. De esta manera, mejora la eficacia y la eficiencia del proceso.

El uso de un diagrama de flujo ayudará a mejorar los resultados del negocio en varias áreas o departamentos. Es una herramienta genérica que puede adaptarse para una amplia variedad de propósitos y para describir varios procesos, como un proceso de fabricación, un proceso administrativo o de servicio, o un plan de proyecto [17].

Capítulo 3

Análisis

3.1 Introducción

El proyecto SIMIC 2.0 surge como un complemento de un proyecto de grado anterior (SIMIC). SIMIC es actualmente utilizado como un sistema de historia clínica a la hora de la consulta cara a cara con el paciente.

En este proyecto se hizo un análisis exhaustivo sobre la insuficiencia cardíaca, basado en la experiencia de la Unidad Multidisciplinaria de Insuficiencia Cardíaca (UMIC) del Hospital de Clínicas, donde se desprendía la necesidad de un seguimiento periódico del paciente, que continuase luego de la cita en el consultorio.

Este seguimiento se lleva a cabo a través de una aplicación móvil que interactúa con el paciente, basado en un motor de reglas. Estas reglas son una secuencia de instrucciones que indican los pasos a seguir según las respuestas de los pacientes, y a su vez, las mismas son escritas por los médicos.

SIMIC 2.0 captura datos del estilo de vida del paciente en su entorno, genera recomendaciones y evalúa la situación antes de alertar al equipo médico.

3.2 Objetivos generales del proyecto

Dentro de los objetivos del proyecto se tuvo en cuenta:

- Implementar un sistema innovador y que sea un desafío técnico que implique el aprendizaje de nuevas tecnologías por parte de los estudiantes de Ingeniería.
- Que el sistema desarrollado contribuya con la causa particular (UMIC) y general (solución universal).

- Cumplir con las expectativas del equipo médico tanto con el nuevo sistema, como también poder generar soluciones complementarias para el mejor aprovechamiento del sistema anterior.

- Involucrar a los médicos y estudiantes de medicina en el nuevo sistema y en la informática, para dar paso a una nueva modalidad pensando en un futuro tener médicos programadores.

3.3 Alcance del sistema

Este punto tiene como objetivo plantear el proceso de negociación del alcance. Dada la particularidad de este proyecto por el hecho de ser multidisciplinario, se tuvo que tener en cuenta los objetivos de cada uno de los clientes, en este caso, el cuerpo médico y los tutores del proyecto.

Se presentan a continuación las etapas que tuvo este proceso y las decisiones que llevaron al alcance final.

Primera versión

- Mejorar la aplicación SIMIC 1.0 que incluya módulo de análisis y extracción de datos, mantenimiento del ambiente de producción, solución de bugs existentes en el sistema.

- Aplicación móvil para pacientes: Incorporar reglas de seguimiento de la insuficiencia cardíaca en domicilio.

- Las reglas deberían estar escritas en un lenguaje de fácil lectura y comprensión por parte de los médicos. Se debía especificar un nuevo lenguaje que tenga todos los elementos necesarios para su traducción a partir de un diagrama de flujo [15].

Problemas encontrados

- Dado la poca documentación técnica generada para SIMIC 1.0 no fue posible generar un ambiente de desarrollo que replicara el ambiente de producción, lo que hizo que no se pudiera implementar sobre SIMIC 1.0.

- Especificar un nuevo lenguaje para escribir recetas no era viable debido a la gran cantidad de sentencias que se necesitaban crear y tampoco era escalable ya que si se quisiera agregar una nueva sentencia se tenía que agregar al lenguaje.

- Los médicos expresaron su urgente necesidad del módulo de extracción de datos para SIMIC 1.0.

Decisiones tomadas

- No implementar sobre SIMIC 1.0 y hacer un sistema paralelo, utilizando datos de este sistema consumiendo mediante consultas a la base de datos.
- No implementar un nuevo lenguaje sino utilizar uno ya existente junto con un intérprete conocido. Esto nos beneficiaría ya que el lenguaje nos provee de varias funciones que iban a ser necesarias y las cuales no iban a tener que especificarse.
- Se utilizó la herramienta Google Data Studio para el análisis de datos.

Alcance del proyecto

A continuación, se presenta cada uno de los componentes de la solución que son independientes de SIMIC 1.0.

- Sistema web de reglas: Motor y administración:
    - Motor de reglas**:** Se les ofrecerá un editor de escritura de las reglas (recetas) con un motor implementado en Python. Este motor tendrá disponible una interfase con funcionalidades para que los médicos puedan escribir las recetas, pero de manera controlada. Se tendrá que tener en cuenta el asincronismo de las recetas al momento de esperar la respuesta de los pacientes.
    - Administración de las reglas: Se definirán roles dentro del cuerpo médico. Por un lado, habrán médicos que serán los encargados de programar las reglas. Por otro lado, estarán los médicos que le asignen dichas reglas en las consultas, según la condición del paciente.
    - La comunicación desde SIMIC 1.0 a SIMIC 2.0 se hará por medio de una extensión de Chrome que comunicará ambas aplicaciones.
    - Alertas por paciente: El médico clínico podrá ver las alertas generadas por paciente desde la aplicación web
- Aplicación para pacientes: Se implementará una aplicación móvil para que se pueda utilizar tanto en Android como en iOS. Esta aplicación será para los pacientes, y tendrá la funcionalidad de correr las reglas que implementaron los médicos y prescribieron a cada uno de ellos.
- Módulo de análisis: Se dejará disponible con reportes dinámicos implementado por Google Data Studio. Esto les permitirá a los médicos acceder a los datos de la base de datos y descargarlos, como también tendrán la posibilidad de ver distintos reportes gráficos que tenga disponible la herramienta.

3.4 Actores

3.4.1 Cliente

UMIC está integrada por un grupo de profesionales de la salud pertenecientes al Hospital de Clínicas “Dr. Manuel Quintela” de la República Oriental del Uruguay.

Los integrantes de UMIC que tendrán el rol de clientes son: Prof. Dra. Gabriela Ormaechea, Prof. Adj. Dr. Pablo Álvarez, Dra. Gabriela Silvera.

Por otra parte, también serán clientes del proyecto el Prof. Ing. Franco Simini y los estudiantes de Pregrado de Medicina: Lucia Ribeiro, Isabel Ribeiro y Valentina Fernández.

3.4.2 Perfiles de usuario

Médico Autor: Médico encargado de la traducción del diagrama de flujo a código Python, es el encargado de la creación de reglas (recetas).

Médico Clínico: Médico encargado de atender al paciente en la consulta y de asignarle las recetas a los pacientes.

Paciente: Son pacientes de la UMIC, estos van a poder tener acceso mediante la aplicación móvil SIMIC 2.0 a las recetas que el Médico Clínico le indicó y podrán tener el seguimiento a distancia necesario.

3.5 Especificación funcional

3.5.1 Requerimientos funcionales

A continuación, se presentan casos de uso separados para la web y para el dispositivo móvil.

WEB

Nombre: Login web administración de reglas
Objetivo: El usuario ingresa al sistema
Actores: Médico autor y médico clínico
Precondiciones: ● Estar registrado en SIMIC 1.0

Descripción: El usuario ingresa sus credenciales y es autenticado y validado por el sistema
Flujo Normal: 1. El usuario ingresa su email asociado y contraseña. 2. El sistema verifica las credenciales del usuario y el usuario es autorizado a continuar.
Flujos Alternativos: ● 2.A: El usuario ingresa email o clave incorrecta ○ El sistema despliega un mensaje de error correspondiente
Poscondiciones:

Nombre: Visualización detalle del paciente
Objetivo: Médico en consulta visualiza detalle de paciente en SIMIC 2.0
Actores: Médico clínico
Precondiciones: ● Estar registrado en SIMIC 1.0
Descripción: El usuario ingresa a SIMIC 2.0 desde SIMIC 1.0 a ver detalles del paciente
Flujo Normal: 1. Médico clínico busca paciente en SIMIC 1.0 2. Mediante la extensión de chrome se lo redirige a SIMIC 2.0 3. Paciente ya registrado, se le muestra la pantalla de información del paciente
Flujos Alternativos: ● 3.A: El paciente no se encuentra registrado en SIMIC 2.0 ○ El sistema despliega los datos que trae de SIMIC 1.0 ○ Se confirma el registro
Poscondiciones:

Nombre: Ingreso de nueva regla
Objetivo: Médico autor genera una nueva regla en Python
Actores: Médico autor
Precondiciones: ● Médico autor logueado en el sistema
Descripción: El médico autor traduce el diagrama de flujo de una regla de seguimiento en código Python y la guarda para posterior uso.
Flujo Normal:

1. Médico autor ingresa a SIMIC 2.0
2. Ingresa a Menú de reglas
3. Ingreso a Editor de reglas
4. Escribir código Python
5. Guardar archivo

Flujos Alternativos:

Poscondiciones: Se crea una nueva receta de alias igual a nombre de archivo

Nombre: Asociar regla a un paciente

Objetivo: Asociar un seguimiento(regla) a un paciente para que comience a ejecutarse en la aplicación.

Actores: Médico clínico

Precondiciones:
- Estar logueado en SIMIC 2.0
- El paciente debe existir en SIMIC 2.0
- La regla debe estar creada en SIMIC 2.0

Descripción: El médico clínico en la consulta con el paciente, le receta una de las reglas de seguimiento creadas, al paciente según las necesidades de este último

Flujo Normal:
1. Ir a pantalla de detalle del paciente
2. Ir a pantalla de asignar receta
3. Completar: nombre, periodicidad, y fecha de inicio de la receta
4. Emitir receta

Flujos Alternativos:
4.A Si se habilita el icono de Agregar variable seguir caso de uso: *Agregar variable paciente* hasta que el icono se deshabilite, indicando esto que ya se han completado todas las variables necesarias

Poscondiciones: Receta queda asociada al paciente, y comienza a ejecutarse la misma.

Nombre: Agregar variable al paciente

Objetivo: Agregar variables necesarias para las recetas ya asociadas

Actores: Médico clínico

Precondiciones:
- Receta debe estar asociada a un paciente

● Receta debe contener variables a completar
Descripción: Se completan las variables contenidas en el código python de la receta, necesarias para comenzar a ejecutar.
Flujo Normal: 1. En la pantalla de detalle de paciente, ir a Agregar variable 2. Completar nombre eligiendo de los que sugiere el sistema 3. Completar valor 4. Guardar variable
Flujos Alternativos: N/C
Poscondiciones: Variable queda con un valor asignado

Nombre: Obtener código de invitación para paciente
Objetivo: Entregarle al paciente el código de invitación a la aplicación móvil
Actores: médico clínico
Precondiciones: ● Se debe haber registrado al paciente en SIMIC 2.0
Descripción: Para que el paciente pueda ingresar a la aplicación móvil de SIMIC 2.0 es necesario un código de invitación que es proporcionado en la consulta por única vez
Flujo Normal: 1. Ir a la pantalla de detalle del paciente 2. Entregarle el código que aparece en la pantalla
Flujos Alternativos: N/C
Poscondiciones: Luego de que el paciente utiliza el código, no se visualiza más en el detalle del paciente

Nombre: Ver respuestas de pacientes
Objetivo: Poder visualizar las respuestas del paciente.
Actores: médico clínico
Precondiciones: ● Se debe haber registrado al paciente en SIMIC 2.0 ● Médico le debe haber asignado alguna receta y ésta debe estar en ejecución
Descripción: Luego que las recetas de seguimiento estén en ejecución, el paciente irá respondiendo a las preguntas que se le presentan en la aplicación móvil. Es de gran importancia ver la interacción de los pacientes con las recetas, como también el nivel de

adherencia a las mismas. Es decir, el nivel de respuesta sobre el total de las preguntas realizadas.
Flujo Normal: 1. Ir a la pantalla de detalle del paciente 2. Elegir una receta para ver respuestas 3. Visualizar respuestas y nivel de adherencia a la regla
Flujos Alternativos: N/C
Poscondiciones:

Nombre: Ver alertas
Objetivo: Ver alertas generadas para un paciente
Actores: médico clínico
Precondiciones: ● Se debe haber registrado al paciente en SIMIC 2.0 ● Médico le debe haber asignado alguna receta y ésta debe estar en ejecución
Descripción: Existen recetas que generan alertas al equipo médico, y se podrán visualizar por paciente.
Flujo Normal: 3. Ir a la pantalla de detalle del paciente 4. Visualizar alertas
Flujos Alternativos: N/C
Poscondiciones:

APLICACIÓN MÓVIL

Nombre: Autenticación en la aplicación
Objetivo: Paciente se autentica con éxito en la aplicación
Actores: Paciente
Precondiciones: ● Paciente debe estar ingresado en SIMIC 2.0 ● Médico le debe haber entregado el código de invitación ● Paciente debe haber descargado la aplicación móvil
Descripción: El paciente debe ingresar un código de invitación que le entrega el médico

en la consulta presencial antes de poder utilizar la aplicación
Flujo Normal: 1. Paciente ingresa el código de invitación 2. Ingresa a la aplicación
Flujos Alternativos: N/C
Poscondiciones: Paciente queda logueado en la aplicación de SIMIC 2.0

Nombre: Respuesta a preguntas desde la aplicación móvil
Objetivo: Paciente responde las preguntas que se le presentan
Actores: Paciente
Precondiciones: ● Paciente debe estar autenticado en la aplicación de SIMIC 2.0 ● Médico le debe haber asignado alguna receta y ésta debe estar en ejecución
Descripción: Luego de que el médico le asoció una receta de seguimiento al paciente, le llegarán preguntas o recomendaciones que este debe responder
Flujo Normal: 1. Lee y responde la pregunta 2. Envía respuesta
Flujos Alternativos: 1. A. El paciente no responde la pregunta
Poscondiciones: Se envía la respuesta deseada o se genera una notificación push a modo de recordatorio en caso de que el paciente no responda (Ver *Notificación de pregunta a la aplicación móvil*)

Nombre: Notificación de pregunta a la aplicación móvil
Objetivo: Avisarle al usuario cuando tiene una pregunta para responder
Actores: Paciente
Precondiciones: ● Paciente debe estar autenticado en la aplicación de SIMIC 2.0 ● Debe tener una pregunta para responder
Descripción:
Flujo Normal: 1. Le llega una notificación al celular que tiene una nueva pregunta por responder 2. Ir a la notificación
Flujos Alternativos: N/C

Poscondiciones: N/C

3.5.2 Requerimientos no funcionales

SIMIC 1.0 No Modificable
SIMIC 1.0 no dispone de suficiente documentación técnica para poder realizar cambios en el código fuente e introducir nuevas funcionalidades. Tampoco fue posible levantar un ambiente de prueba con el código disponible actualmente.

Portabilidad
Uno de los propósitos del proyecto es obtener un seguimiento continuo de los pacientes.
Es por esto que SIMIC 2.0 cumple este objetivo por medio de los dispositivos móviles. En este caso se publicará la aplicación solo para el sistema operativo Android, aunque estará implementada en un sistema multiplataforma, y en un futuro se podrá liberar también para iOS.

Usabilidad / Interfaz
Los pacientes que utilizan la aplicación móvil, son personas de diversas edades, en su mayoría adultos mayores, es por esto que es necesario lograr un diseño amigable y simple, que facilite su usabilidad.

Operatividad
El tiempo de las consultas al sistema es reducido y es necesario que el ingreso de datos sea lo más rápido y eficiente posible. Es por esto que SIMIC 2.0 se pensó teniendo en cuenta estas consideraciones.

Seguridad
Se debe manipular adecuadamente la información ya que se trata de datos personales de los involucrados. Se tuvo en cuenta tanto la seguridad de la web como la de la aplicación móvil.

Interoperabilidad y extensibilidad
Es de gran importancia la integración con SIMIC 1.0 como caso de estudio, asimismo que pudiera conectarse en un futuro con cualquier otra herramienta de manejo de enfermedades crónicas. Es así que se diseñó el sistema de manera genérica aceptando nuevas integraciones.

3.6 Diagramas de flujo utilizados

El médico autor y el médico clínico tienen un mismo lenguaje en común que es el diagrama de flujo. Este va a ser el lenguaje que van a poder utilizar para poder entenderse y generar las recetas para el seguimiento a distancia.

A la UMIC les interesa particularmente el seguimiento del peso como la disnea. A continuación se presentan los diagramas de flujo que fueron parte del análisis de SIMIC 2.0.

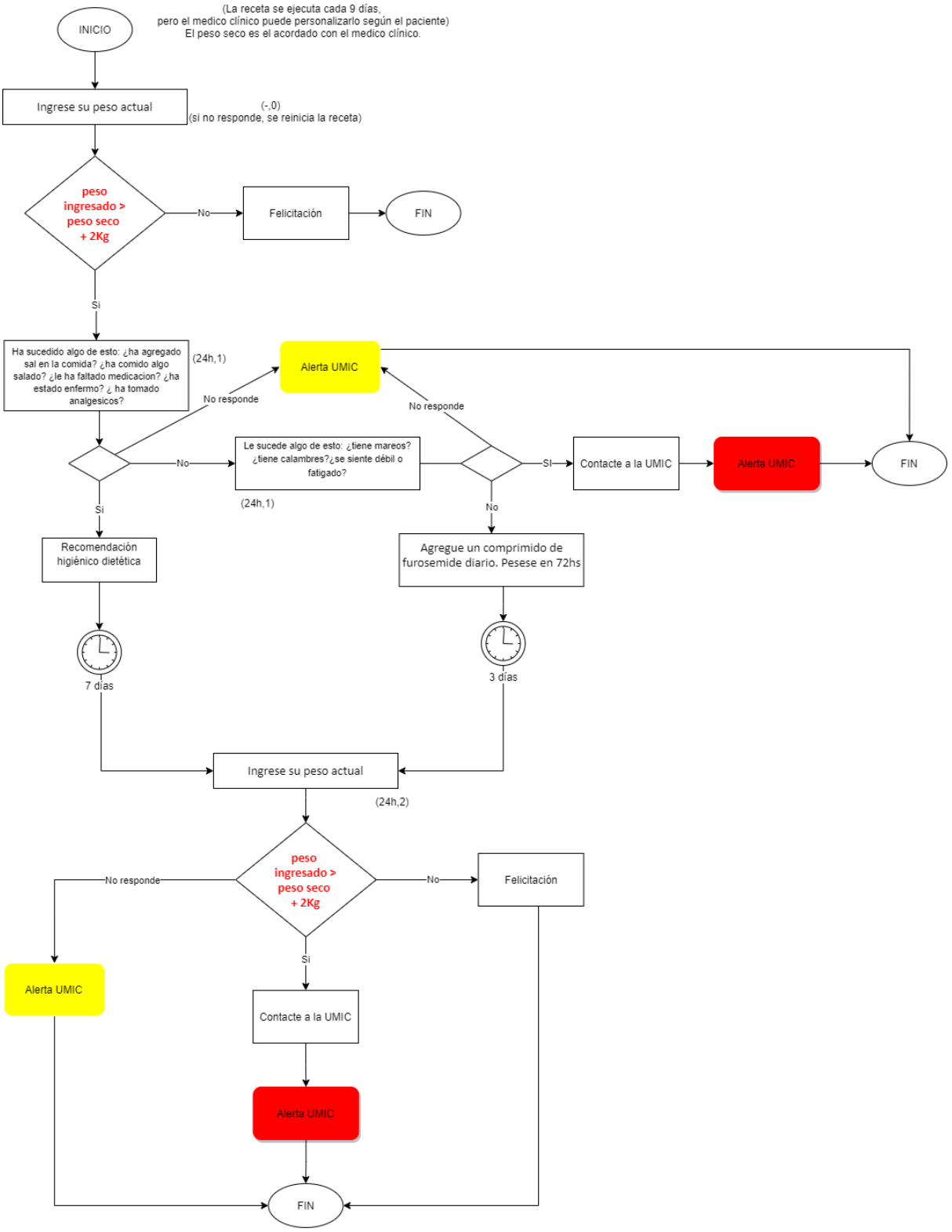


Figura 2: Receta peso

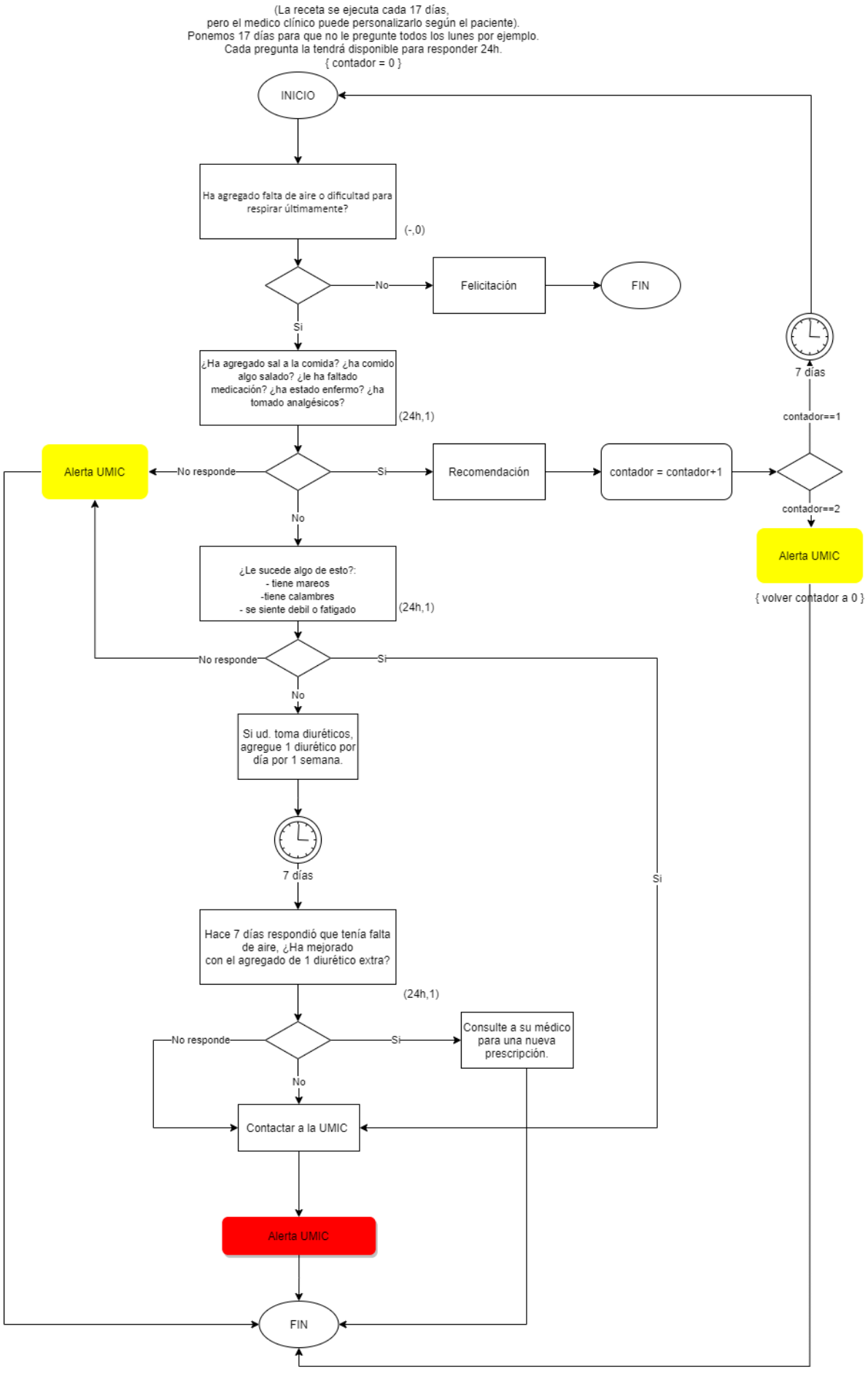

(La receta se ejecuta cada 17 días,
pero el medico clínico puede personalizarlo según el paciente).
Ponemos 17 días para que no le pregunte todos los lunes por ejemplo.
Cada pregunta la tendrá disponible para responder 24h.
{ contador = 0 }
INICIO
Ha agregado falta de aire o dificultad para respirar últimamente?
(-,0)
No
Felicitación
FIN
Si
¿Ha agregado sal a la comida? ¿ha comido algo salado? ¿le ha faltado medicación? ¿ha estado enfermo? ¿ha tomado analgésicos?
(24h,1)
Alerta UMIC
No responde
Si
Recomendación
contador = contador+1
7 días
contador==1
contador==2
Alerta UMIC
{ volver contador a 0 }
No
¿Le sucede algo de esto?:
- tiene mareos
-tiene calambres
- se siente debil o fatigado
(24h,1)
No responde
Si
No
Si ud. toma diuréticos, agregue 1 diurético por día por 1 semana.
7 días
Hace 7 días respondió que tenía falta de aire, ¿Ha mejorado con el agregado de 1 diurético extra?
(24h,1)
No responde
Si
Consulte a su médico para una nueva prescripción.
No
Contactar a la UMIC
Alerta UMIC
FIN

Figura 3: Receta disnea

Luego de que se crearon los diagramas de flujo se notó que se necesitaban los siguientes ítems para poder crear las recetas:

A. Interacción con el paciente: Estas interacciones están marcadas en el diagrama de flujo con un rectángulo y lo que se quiere representar es una pregunta hacia el paciente.

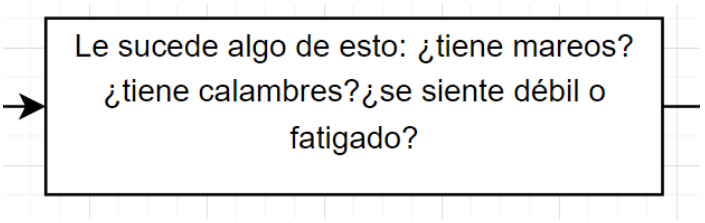


Figura 4: Interacción con el paciente

B. Periodicidad de la receta: Se notó que las recetas debían tener una fecha de comienzo y también repetirse cada cierto tiempo.

C. Tiempo de espera por interacción (TimeOut): A los médicos les interesa particularmente realizar ciertas acciones cuando el paciente no responde a las preguntas realizadas en cierto tiempo.

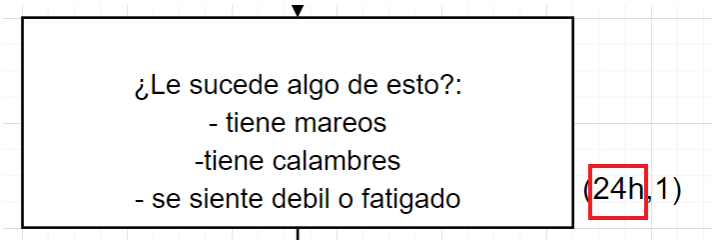


Figura 5: Tiempo de espera por interacción

D. Repregunta: Hay momentos en que se quiere volver a realizar la misma pregunta si el tiempo de espera en esa interacción caducó.

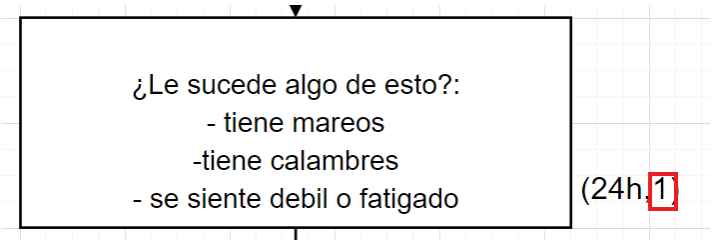


Figura 6: Repregunta

E. Condición: Dependiendo de las respuestas que va dando el paciente el médico clínico quiere elegir la acción a realizar vía programación.

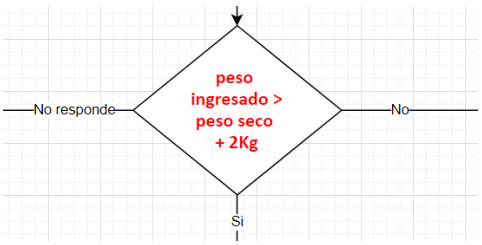


Figura 7: Condición de la acción a realizar en la receta

F. Espera: En ciertos momentos se debe esperar cierto tiempo para continuar con el flujo de la receta.

Figura 8: Espera en la receta

G. Alerta: A los médicos clínicos les interesaba recibir ciertas alertas cuando los pacientes indican ciertos comportamientos. Estas alertas deben tener distintos niveles para poder indicar la gravedad.

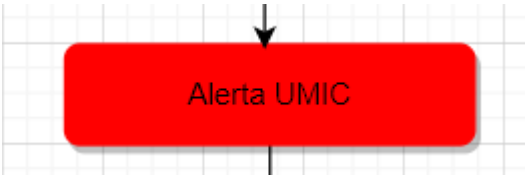


Figura 9: Alerta en la receta

H. Variable: Cierta información es necesaria para poder realizar el flujo y debe tener un nombre y un valor inicial para poder ir manipulando.

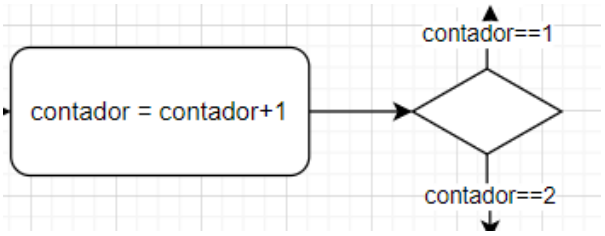


Figura 10: Variable en la receta

Capítulo 4

Diseño

4.1 Decisiones tomadas

4.1.1 Utilización de Python como motor de reglas

El lenguaje de programación Python es el preferido a nivel internacional, según el PopularitY of Programming Language Index (acortado como PYPL). Así, en enero del 2021 este era utilizado por casi el 26% de los programadores web, tal y como se estima en un análisis de las búsquedas en Google de tutoriales sobre programación [18].

Figura 11: Lenguajes de programación más utilizados

A continuación, se hace una comparativa de los lenguajes de programación más utilizados:

Lenguaje	Python	Java	C#
Compilado/ Interpretado	Interpretado	Compilado	Compilado
Orientado a Objetos	Si	Si	Si
Estático/Dinámico	Dinámico	Estático	Estático

Dado que uno de los objetivos de este proyecto es que el médico autor sea el responsable de la traducción del diagrama de flujo a código, se utilizó el lenguaje de programación Python que tiene menor curva de aprendizaje, es el más utilizado y no necesita ser compilado.

4.1.2 VS Code como ambiente para escribir recetas

Dado que el médico autor es el encargado de escribir las recetas, se vió la necesidad de brindarle un ambiente que le resulte amigable y que le sea sencillo poder escribirlas.

A continuación, se hace una comparativa de los dos editores de código más populares:

Ambiente	VS Code	Sublime
Open-source	Si	No
Intellisense	Si	Si
Python	Si	Si
Utilizado por la comunidad de desarrolladores	Si	Si

Se tomaron en cuenta los siguientes puntos para la elección de VS Code como el editor para crear recetas:

- Open-source: Era importante este punto ya que era necesario poder desplegarlo como una aplicación web.
- Intellisense: La función de autocompletado era imprescindible ya que los módulos que le proveía SIMIC 2.0 iban a ser utilizados y por lo tanto no era necesario que el médico autor se fijará cada vez los parámetros de cada función.

- Python: Al elegir Python como lenguaje para crear las recetas, el ambiente necesitaba soportar el lenguaje.
- Utilizado por la comunidad de desarrolladores: Era importante seleccionar un ambiente que ya esté probado por la comunidad y recomendado.

Se utilizó la versión web de VS Code [29]. Esto brinda facilidad al médico autor a la hora de codificar ya que simplemente creando un nuevo archivo ya está escribiendo el código de una nueva receta. También facilita la integración con la aplicación web de SIMIC 2.0.

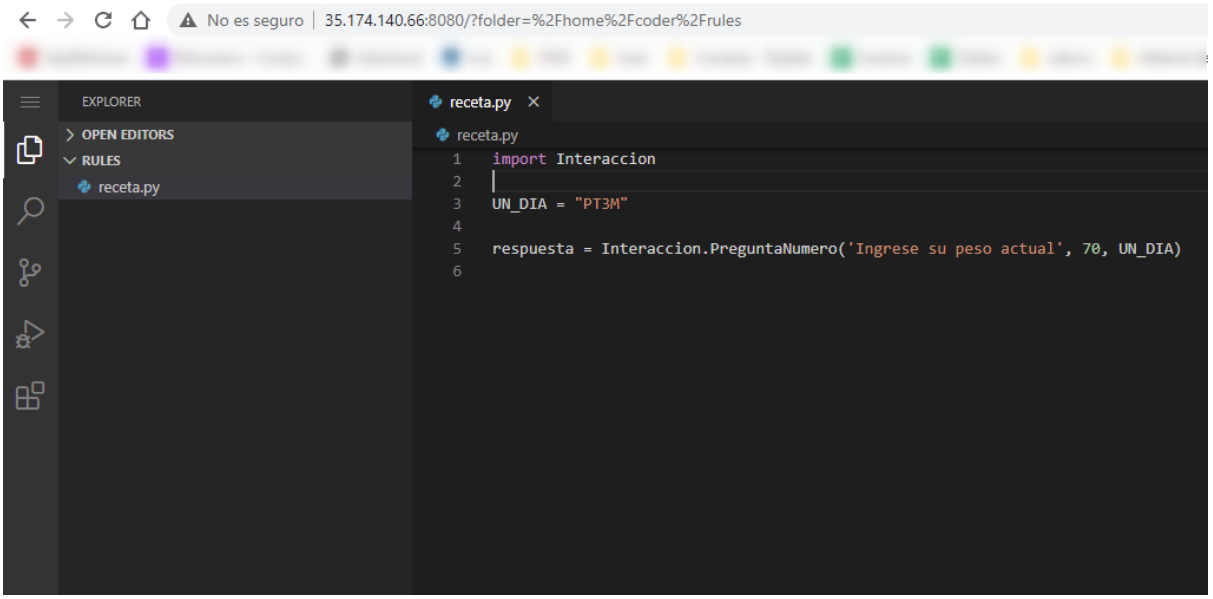


Figura 12: Ambiente para crear recetas

4.1.3 Pausa y reanudación de la regla

Al efectuarse una interacción con el paciente, se debe aguardar la respuesta del mismo para continuar la regla. Como la regla no es más que un script de programación, este se ejecuta secuencialmente, no es una tarea sencilla aguardar la respuesta del paciente para proseguir con la ejecución del código con todo su contexto.

De aquí surgen dos opciones para solucionar este problema:

A. Guardar el estado del proceso

Guardar el estado del proceso nos daría la posibilidad de saber qué usuario estaba ejecutando la regla y en qué parte de la regla había quedado esperando, para esto se investigaron diversas librerías de Python. Pero esto traía un gran inconveniente que era tanto la escalabilidad como la indisponibilidad, ya que si por algún motivo se caía el servidor todos los procesos que estaban corriendo también se perderían y por lo tanto esto no era una solución óptima.

B. Utilizar la capa de persistencia para conocer el estado de la regla

La otra opción analizada fue utilizar la persistencia para administrar qué interacciones han sido resueltas y cuáles no, para esto se debe poder identificar una pregunta, en una receta,

para un cierto paciente. Es por esto, que decidimos utilizar hashes para identificar una cierta interacción.

Se decidió utilizar un hash que tenga poca probabilidad de colisiones, como SHA-256 [30], en base a la CI del paciente, el identificador de la receta en ejecución y el texto de la pregunta.

Si este hash es encontrado en la base de datos, esto quiere decir que la respuesta a esa pregunta ya fue respondida, entonces no es necesaria una nueva interacción, por lo tanto se puede continuar con la ejecución del resto del código.

En cambio, si el hash no es encontrado, implica una nueva interacción con el paciente, por lo tanto, una nueva espera por la respuesta.

A modo de ejemplo, se explica el siguiente fragmento de código ejecutado en el contexto de SIMIC 2.0 y las interacciones que genera:

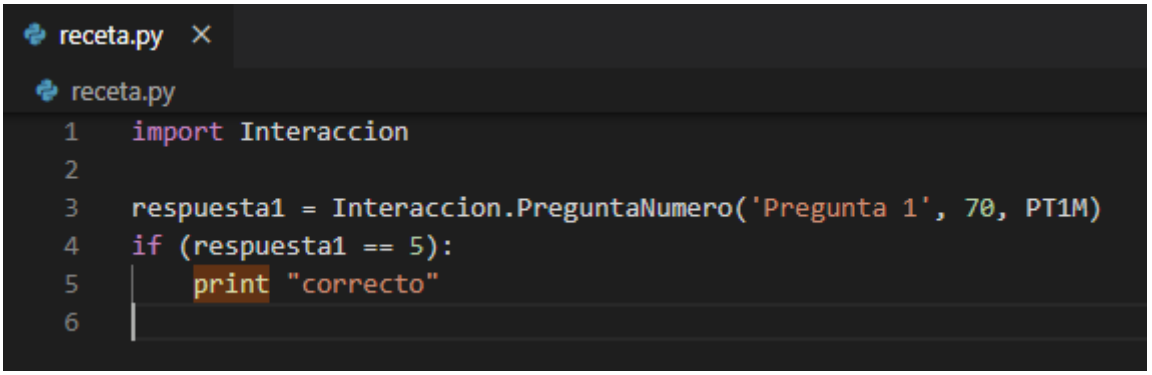


Figura 13: fragmento de código ejecutado en el contexto SIMIC 2.0

Suponiendo que la receta es ejecutada para el paciente con cédula 12345678, y la receta contiene el identificador 1. Al ejecutarse la tercer línea se genera el hash en base al siguiente texto: “123456781Pregunta1”, resultando en el siguiente Hash: *28832fba9d2407ca6a182135e4c9f2e2ecf5167f7dcb061a5eecce74f4599968.*

Al ser la primera ejecución de la interacción, el hash no se encuentra en la base de datos, por lo que se envía la pregunta al celular del paciente.

Una vez que el paciente responde, se persiste la respuesta para ese hash y se ejecuta la regla nuevamente desde el comienzo, pero de esta vez, al ejecutarse la tercer línea se generará el mismo hash, el cual se encontrará en la base de datos, indicando que ya se tiene la respuesta a esta pregunta, continuando con la ejecución en la línea 4.

4.1.4 Extracción de datos con Google Data Studio

En el comienzo del proyecto, los médicos expresaron la necesidad de poder acceder a los datos de SIMIC 1.0 de una manera cómoda para ellos. Cuando se descartó la posibilidad de hacerle modificaciones a SIMIC 1.0, se originó la idea de la utilización de la herramienta Google Data Studio [27]. Esta decisión surgió a partir de los beneficios que este tiene, presentados a continuación:

- Facilidad de conexión de varias fuentes de datos a la plataforma. Era necesario que se pudiera conectar directamente a la base de datos MySQL de SIMIC 1.0 y poder realizar consultas SQL, y este requisito se cumple de forma correcta.

- Transforma los datos presentándolos de forma amigable y permitiendo personalizar los gráficos. Además de que los médicos pudieran extraer planillas de datos, era importante que se pudiera visualizar la información de forma simple y rápida como es a través de gráficas.

- La información se actualiza a tiempo real. Era importante también que la información fuera accesible en todo momento y sin necesidad de la intervención de un programador. Google Data Studio realiza la consulta SQL automáticamente cada cierto tiempo definido por el usuario al momento de configuración, sin necesidad de que se tenga que intervenir manualmente.

- Permite editar la presentación de los informes y compartirlos fácilmente. Es un ambiente de fácil acceso, se necesita únicamente una cuenta de Google y el vínculo para poder acceder.

- Posibilidad de trabajar en un mismo informe de forma conjunta desde distintas computadoras. Tiene la funcionalidad de poder acceder en forma concurrente por varios usuarios.

- Supone un ahorro de tiempo y de recursos al crear los informes. Si bien es necesario una investigación a la base de datos de SIMIC 1.0 para realizar las consultas SQL, no significa un gran esfuerzo al momento de montar las conexiones y generar las planillas.

4.2 Arquitectura

La arquitectura planteada sigue el esquema de desarrollo en capas, la principal ventaja de este tipo de programación es que las distintas lógicas se separan y tienen una estructura bien definida. Las capas que conforman esta arquitectura son:

Capa de presentación: Es la que ve el usuario, le comunica la información y captura la información del usuario. Debe tener la característica de ser “amigable” (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina así porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Capa de datos: Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

La arquitectura de la solución fue pensada para que pueda ser integrada luego con cualquier otro sistema que tenga la necesidad de un seguimiento a distancia. Por lo que en las siguientes secciones se presentará la arquitectura general de la solución y luego la arquitectura propuesta para SIMIC 2.0.

4.2.1 Arquitectura General de la solución

En la siguiente imagen se puede ver un diagrama general de la solución:

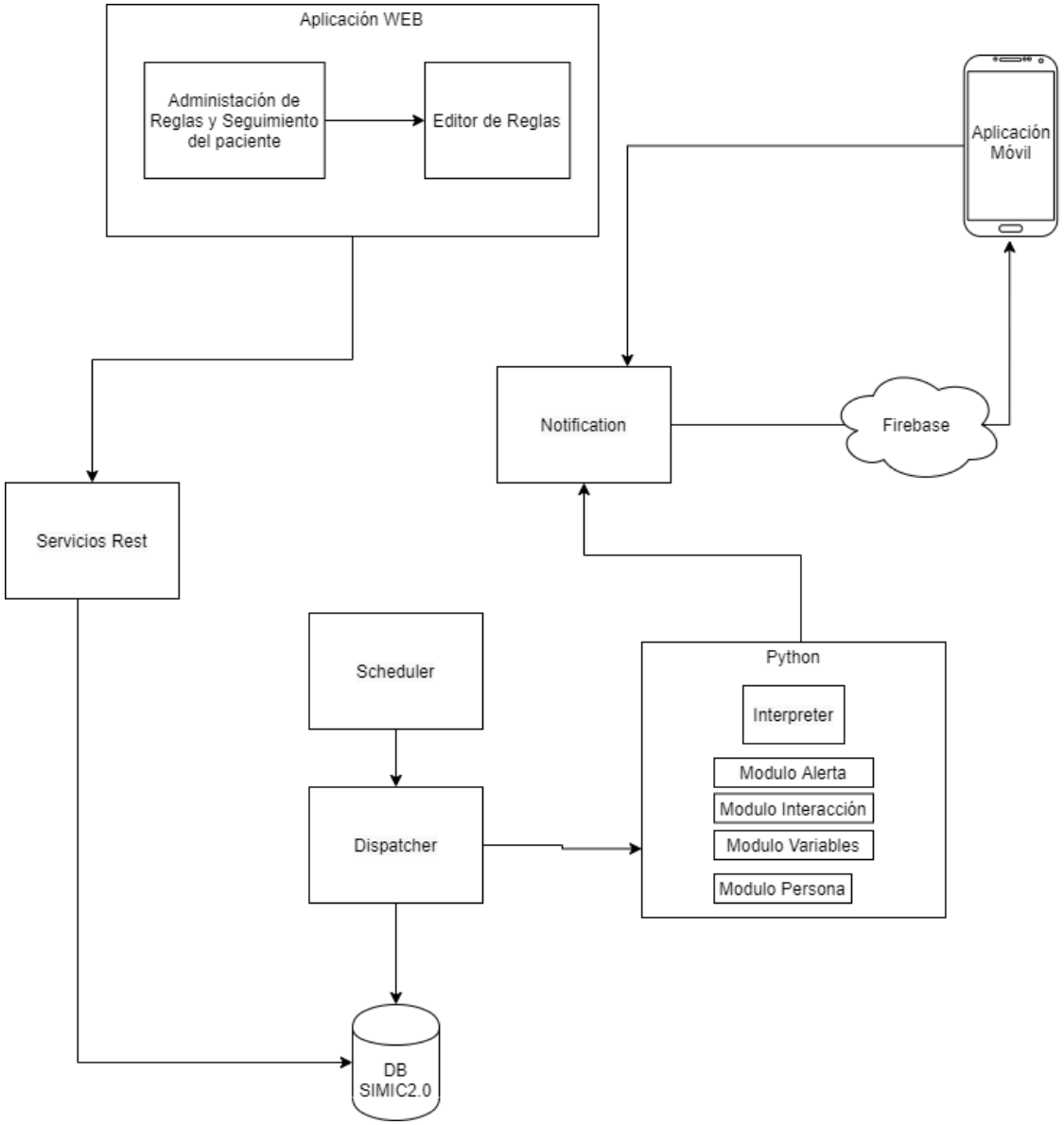


Figura 14: Arquitectura general de la solución

En la cual se detallan los siguientes componentes:

Aplicación Web: La aplicación web tiene dos partes:

A. Editor de Reglas: Esta aplicación es utilizada por los médicos autores. En ella se tiene la posibilidad de escribir las reglas de seguimiento en lenguaje Python.
   Como se mencionó anteriormente, aquí se utilizó la aplicación llamada VS Code versión web, en un container docker [29].

   Para sincronizar los archivos python (extensión .py) de VS Code con SIMIC 2.0 se utilizan volúmenes para compartir la carpeta en donde VS Code almacena dichos archivos y así persistirlos a la base de datos.

B. Seguimiento del paciente y administración de Reglas: Esta aplicación es utilizada por los médicos clínicos en la cual se puede dar de alta usuarios para asignarles reglas de seguimiento, ver las alertas generadas y las respuestas del paciente que realizó a distancia.
   Aquí se utilizó React para desarrollar una SPA (Single Page Application), este componente se comunica con el componente "Servicios Rest" para poder implementar las funcionalidades importantes mencionadas. Dicha comunicación se realiza mediante el protocolo HTTP utilizando el formato Json.

Aplicación Móvil: Aplicación que receta el médico clínico en la consulta con el paciente, esta tiene un código de invitación con la cual el paciente se autentica a la aplicación y responde las preguntas que el médico clínico necesita para hacerle el seguimiento a distancia.
A nivel de red se comunica por HTTP con el componente Notification utilizando el formato Json. La aplicación accede a los endpoints relacionados a la validación del código de invitación, listado de preguntas y respuesta de una pregunta.

Servicios Rest: Servicios necesarios de la capa lógica para que la capa de presentación Web pueda cumplir los requerimientos del médico clínico, estos servicios son:

A. Alta, baja o modificación de usuario

B. Alta, baja o modificación de reglas

C. Asociación de reglas a usuarios

D. Listado de respuestas por usuario

E. Listado de alertas por usuario

F. Indicadores tales como:

   a. Adhesión de una receta (porcentaje de preguntas respondidas de la asignación de la regla al paciente)

   b. Adhesión de una regla (porcentaje de preguntas respondidas de la regla)

c. Adhesión del paciente (porcentaje de preguntas respondidas por el paciente)

Scheduler: Es el componente encargado de verificar qué reglas necesitan ser ejecutadas en cada momento. Se utiliza la librería de Spring boot llamada Scheduling.

Dispatcher: Una vez que el Scheduler tiene las recetas a ejecutarse, notifica al Dispatcher, este es el encargado de comunicarse con el componente Python para enviarle el script que tiene que ejecutar, junto con su contexto (identificador del paciente, identificador de la regla y variables cargadas).
El Dispatcher se comunica vía HTTP con el componente Python, utilizando Json.

Python: Este componente tiene varios Sub componentes que se detallan a continuación:

A. Interpreter: Expone un servicio que es el encargado de recibir el script Python (por parte del Dispatcher) y ejecutarlo. En caso que el script contenga la importación de algún módulo deberá ser cualquiera que posea el núcleo de Python o alguno de los descritos a continuación.

B. Módulo Interacción: Posee funciones que permiten la interacción del usuario con la aplicación móvil, estas funciones son:

    a. Pregunta con opciones: Permite realizar una pregunta al usuario en la cual se tienen varias opciones para elegir alguna de ellas.

    b. Pregunta número: Permite realizar una pregunta la cual la respuesta va a ser un número.

    c. Pregunta fecha: Permite realizar una pregunta la cual la respuesta va a ser una fecha.

C. Módulo Alerta: Este módulo permite generar alertas en la aplicación web para que el médico clínico pueda verlas. Estas alertas pueden configurarse con niveles de gravedad (por defecto el nivel de gravedad es cero).

D. Módulo Variable: Este módulo fue creado para que en la regla se pueda obtener información del usuario necesaria para la ejecución de la misma, las funciones son Obtener Variable (a partir del nombre se obtiene el valor de la variable) o Setear Variable (a partir del nombre se guarda el valor de la variable).

E. Módulo Persona: Este módulo tiene la definición de los métodos (interfaz) que deben ser programados para poder utilizar el módulo, los métodos son:

    a. celular: devuelve el número de celular de la persona

    b. estadocivil: devuelve el estado civil de la persona

    c. fechanacimiento: devuelve la fecha de nacimiento de la persona

d. mail: devuelve el mail de la persona

e. niveleducativo: devuelve el nivel educativo que tiene la persona

f. primerapellido: devuelve el primer apellido de la persona

g. primernombre: devuelve el primer nombre de la persona

h. segundoapellido: devuelve el segundo apellido de la persona

i. segundonombre: devuelve el segundo nombre de la persona

j. sexo: devuelve el sexo de la persona

k. telefono: devuelve el teléfono de la persona

l. etiologia: devuelve la etiología de la persona

m. situacionVital: devuelve la situación vital de la persona

n. ultimaConsulta: devuelve la última consulta que tuvo la persona con el médico

o. consultas: devuelve las consultas que tuvo la persona con el médico

Notification: Es el componente encargado de comunicarse con la aplicación móvil mediante notificaciones push, estas son enviadas a la aplicación mediante Firebase. También expone el servicio de listado de preguntas, y hace que sea posible responder las preguntas.
Los endpoints son expuestos por HTTP, utilizando Json.

DB SIMIC 2.0: Capa de persistencia del sistema SIMIC 2.0

4.2.2 Arquitectura Específica de SIMIC 2.0

Para que la arquitectura general propuesta pueda ser adaptada para integrarse con SIMIC 1.0 se agregaron los siguientes componentes marcados en rojo:

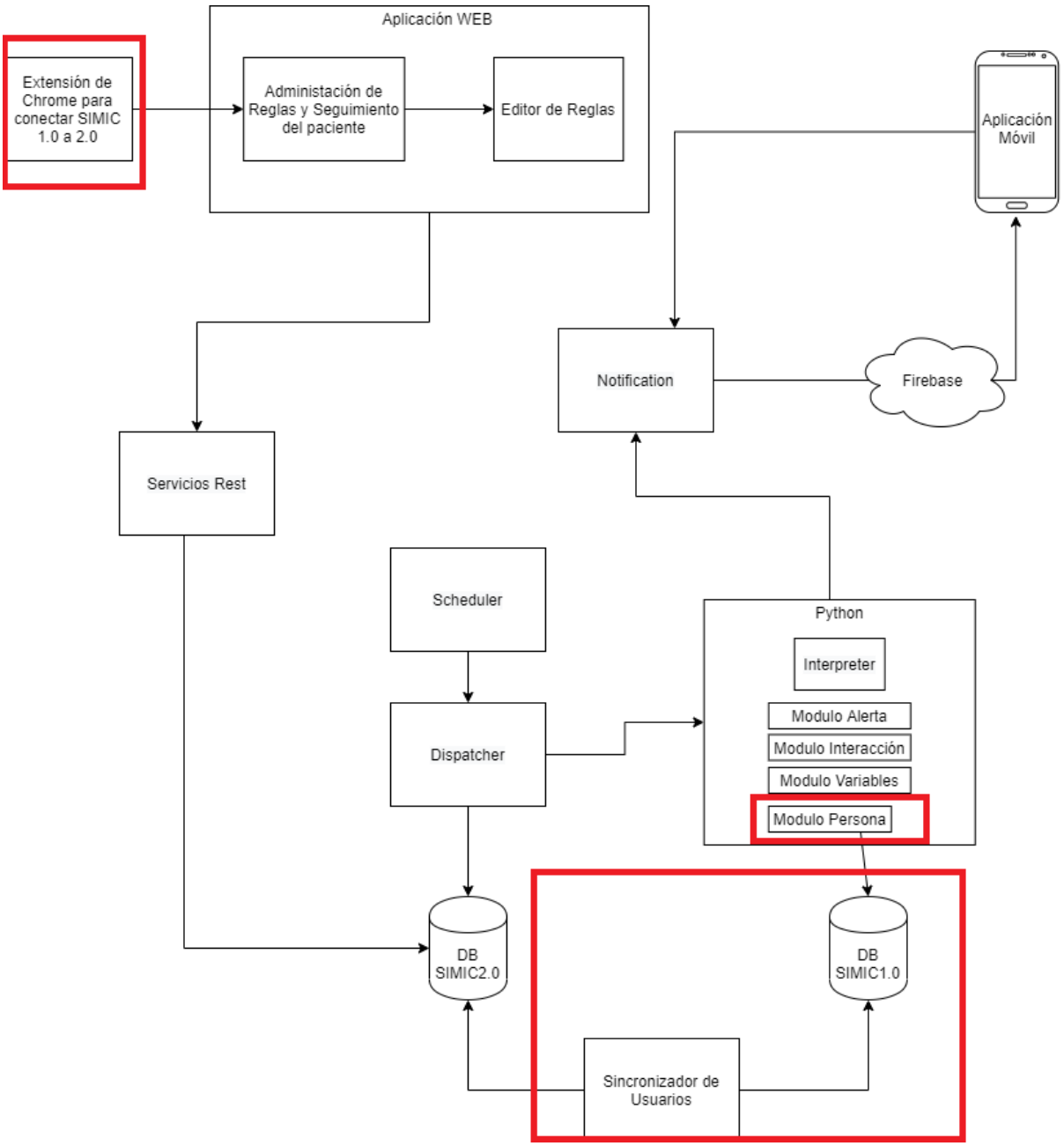


Figura 15: Arquitectura específica SIMIC 2.0

A continuación se explican las modificaciones:

Extensión de Chrome para conectar SIMIC 1.0 a 2.0: Al ser SIMIC 1.0 un sistema independiente a SIMIC 2.0, se vio la necesidad de tener un componente que sea el nexo para obtener los datos más relevantes del paciente (por ejemplo: el nombre y la CI) y darlos de alta en SIMIC 2.0, para esto se desarrolló una extensión de Chrome. Esto simplifica y minimiza los errores de tipeo que el médico clínico podría tener a la hora de querer registrar al paciente en SIMIC 2.0.

A continuación se muestra el flujo:

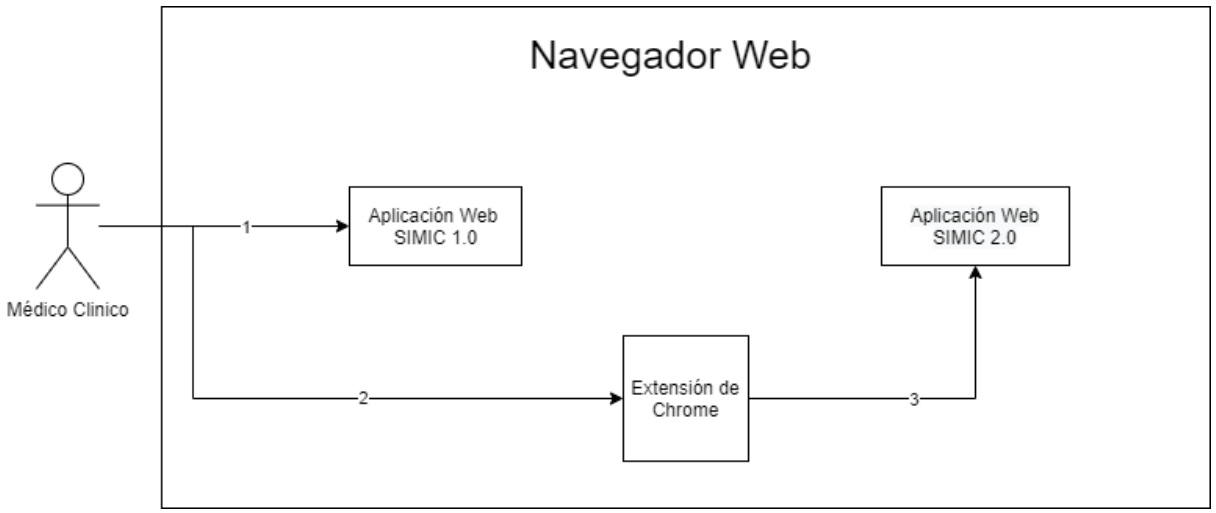


Figura 16: Flujo de extensión Chrome

Como se puede ver en la imagen anterior el médico clínico accede a través de un navegador web a SIMIC 1.0, luego de esto y habiendo seleccionado el paciente aprieta la extensión de chrome y esto abre una nueva pestaña en el navegador que redirige a SIMIC 2.0 con los datos del paciente. Si el paciente no estaba dado de alta se le da la opción al médico para que lo registre y en caso contrario se muestra el perfil del paciente.

Módulo PersonaSimic: Es la implementación de la interfaz del módulo Persona, esta implementación se conecta con la base de datos de SIMIC 1.0 para obtener los datos del paciente (solo realiza consultas).

Sincronizador de Usuarios: Componente que se encarga de sincronizar los usuarios con el rol cuerpo_medico de SIMIC 1.0 a 2.0. Este componente es importante ya que permite a los médicos realizar la autenticación en la aplicación Web de SIMIC 2.0 con las mismas credenciales que en SIMIC 1.0.

4.3 Diagramas de secuencia

Se expone a continuación los diagramas de la implementación de los casos de uso más relevantes para SIMIC 2.0:

Caso de uso - crear regla:
El caso de uso se considera el más importante, ya que permite implementar el concepto de “recetar” . El actor involucrado en el caso de uso es el médico autor, quien utiliza el editor de reglas (Interfaz de Visual Studio Code) para crear y escribir el código de la regla.

A su vez, un proceso se ejecuta cada cierto tiempo para sincronizar los archivos del *file system* del editor de reglas al sistema SIMIC 2.0. Dicho proceso se comunica con los servicios rest para persistir en la base de datos de las recetas.

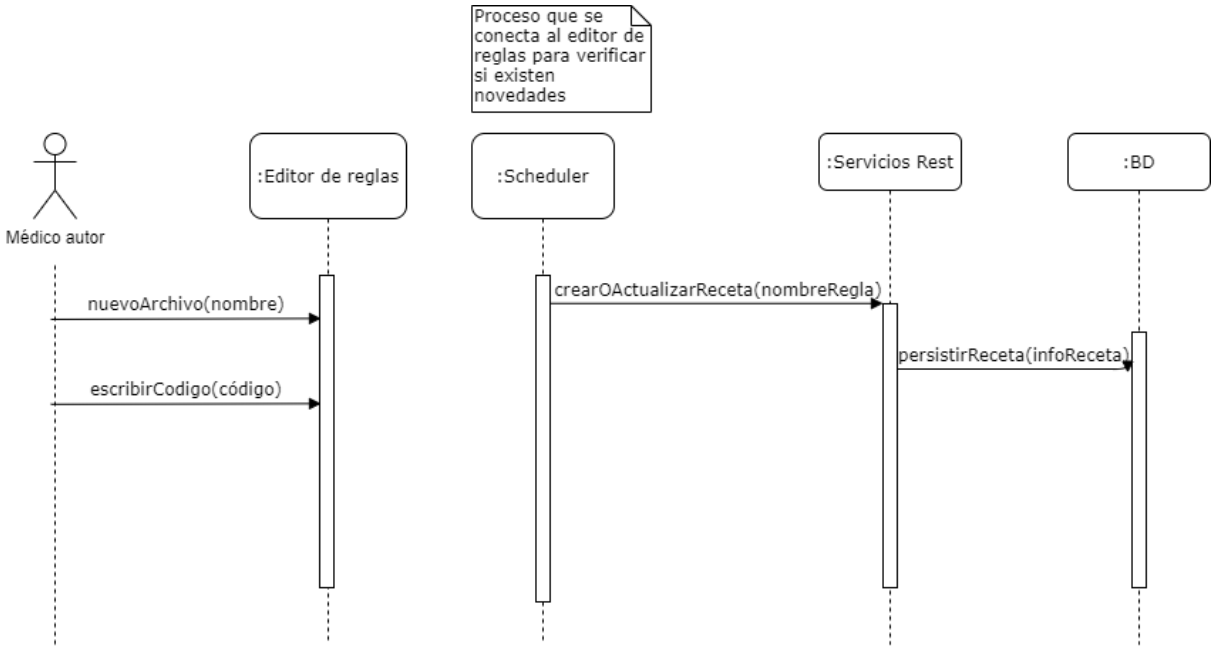


Figura 17: Caso de uso - Crear regla

Caso de uso - ver información del paciente:
La información que se muestra en este caso de uso es:

- Nombre
- Cédula
- Recetas asignadas
- Variables
- Alertas médicas

El caso de uso inicia cuando el médico clínico, estando en la historia clínica del paciente atendido (esto es, en SIMIC 1.0) quiere ingresar a SIMIC 2.0. Para esto hace click en la extensión creada y esta lo redirige a SIMIC 2.0 iniciando el caso de uso Ver info del paciente. En caso de que el usuario no exista, se sincroniza el usuario en SIMIC 2.0 luego se muestra la información del paciente. En caso de existir simplemente se muestra su información mencionada anteriormente.

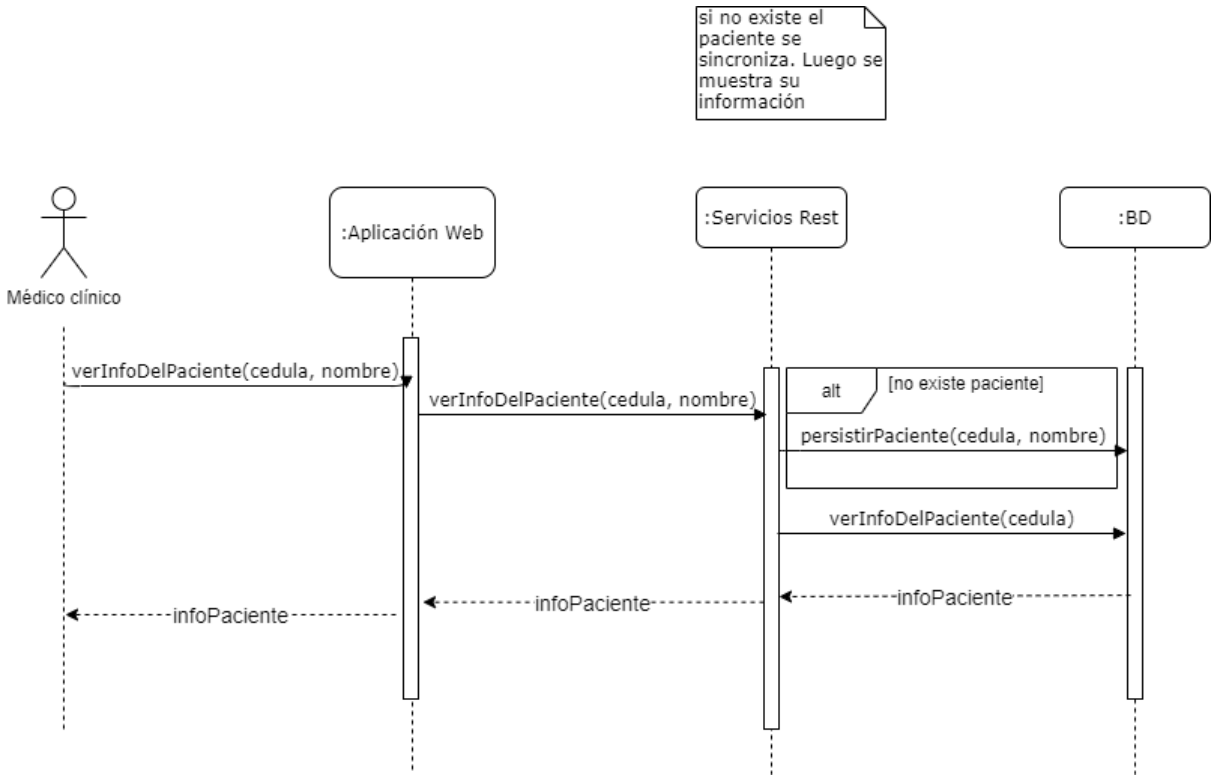


Figura 18: Caso de uso - Ver información del paciente

Caso de uso - asignar receta:

El caso de uso inicia cuando el médico clínico desea recetar una regla al paciente, selecciona la regla, la periodicidad y la fecha de inicio de la regla.

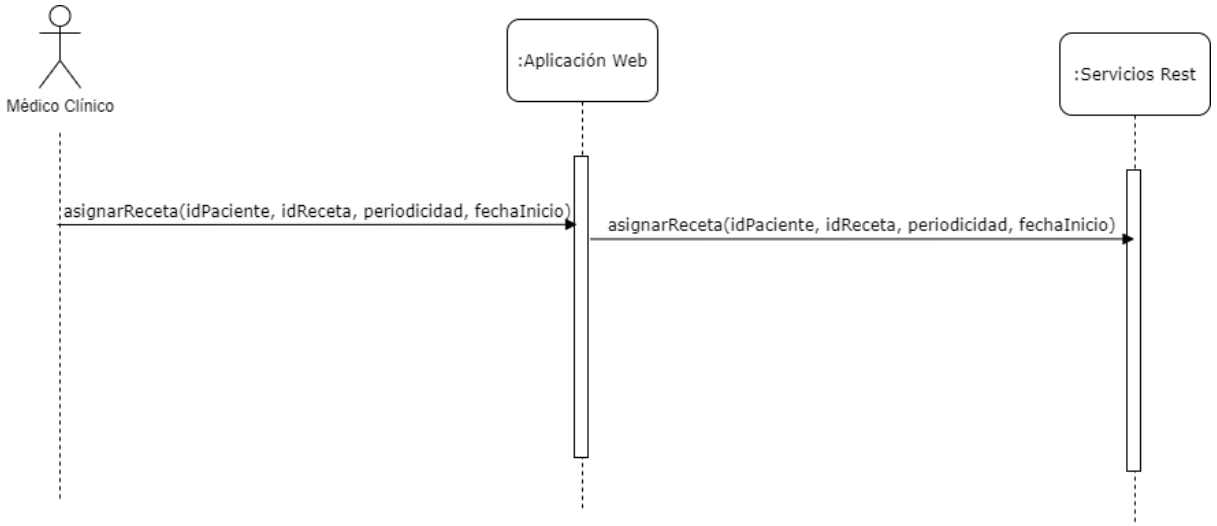


Figura 19: Caso de uso - Asignar receta

Caso de uso - ejecutar receta:
Cada cierto tiempo, el componente Scheduler verifica si existe alguna receta que debe ser ejecutada para algún paciente. Todas estas recetas que deben ser ejecutadas son enviadas al componente Dispatcher, este se encarga de buscar el código de la receta y a su vez, cargar el contexto. En el contexto se cargan las variables asignadas, y la cédula del paciente (identificador).

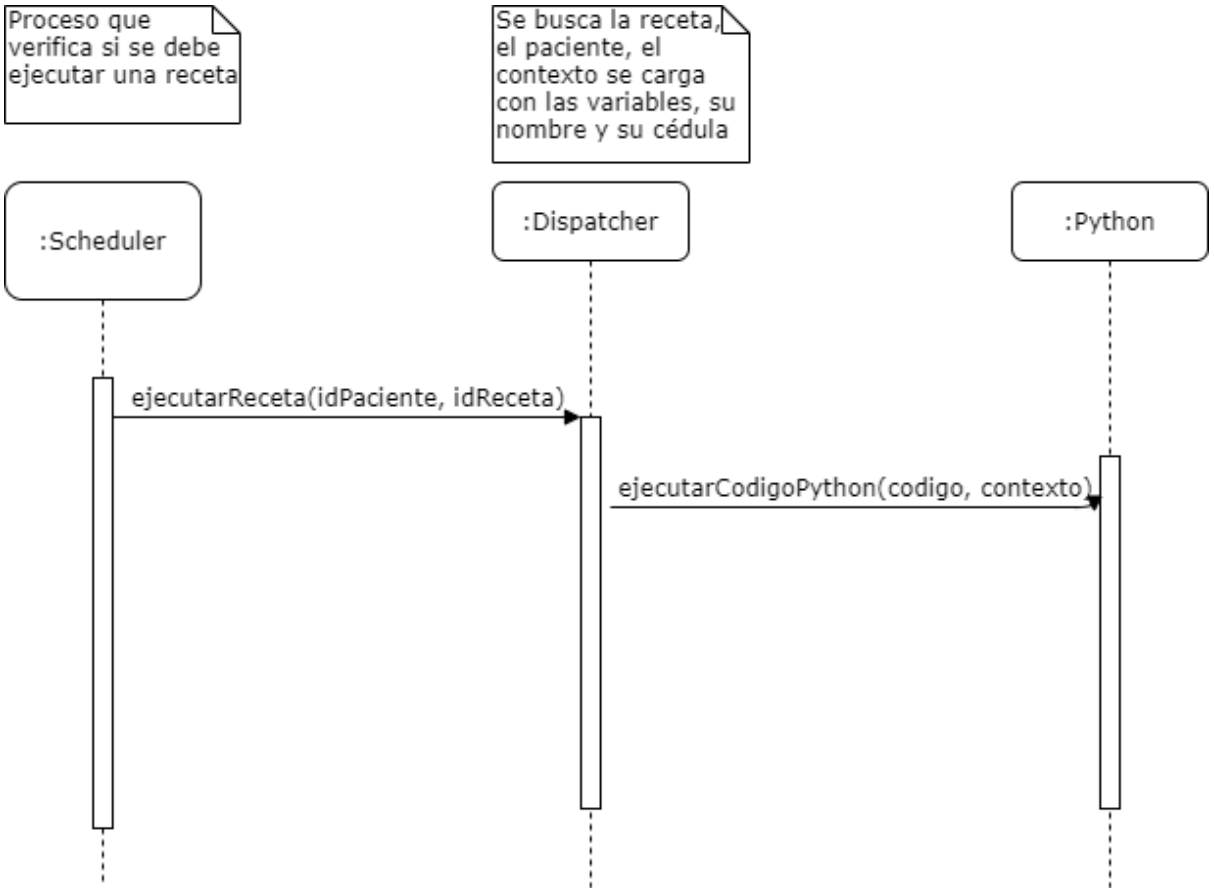


Figura 20: Caso de uso - Ejecutar receta

El código escrito previamente por el médico autor es ejecutado en el componente Python. Una vez que existe una línea de código con una interacción con el paciente, como por ejemplo la siguiente:
Interaccion.PreguntaConOpciones('¿Ha agregado sal a la comida?',['si','no'], PT3M)
se genera internamente un hash para identificar la pregunta en esa receta, se consulta al componente Notification si existe respuesta para esa pregunta, en caso de existir, el componente Python obtiene esa respuesta y sigue su ejecución, en caso de no existir respuesta la ejecución es frenada a través de una excepción , notificando al dispatcher de que se debe esperar una respuesta para la “reanudación” del código Python. Esto se explicó mas en detalle en la sección 4.1.3.
Cabe destacar que el componente Notification es el único expuesto a internet para que sean posibles las respuestas y preguntas de y hacia los pacientes, así como también, las notificaciones push para informar de las nuevas preguntas a los mismos.

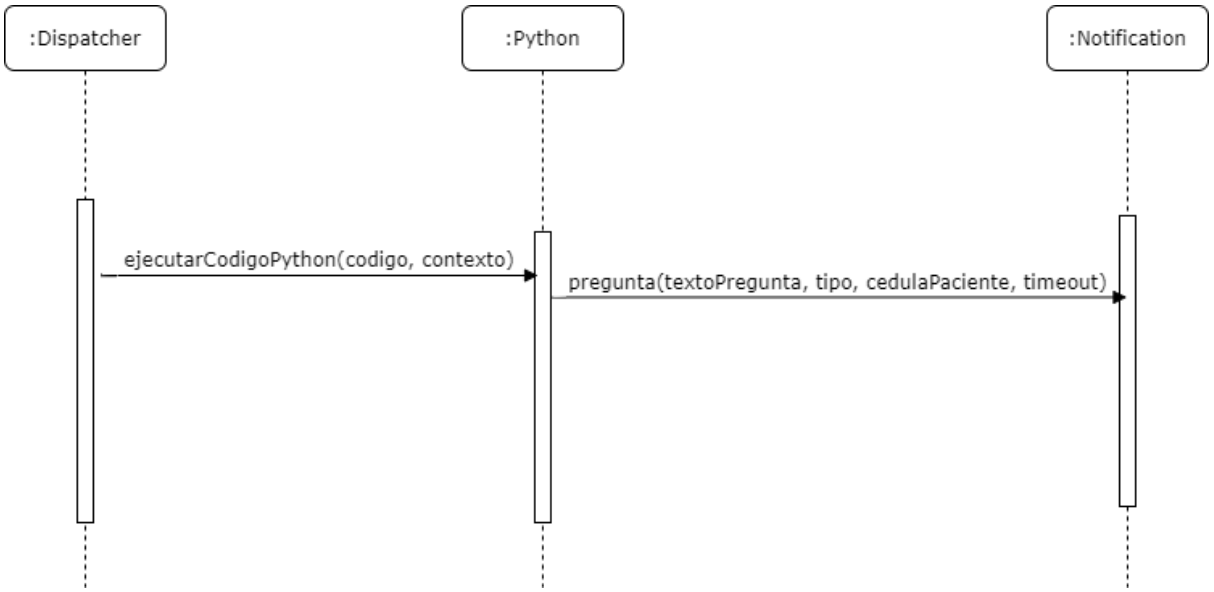


Figura 21: Caso de uso - Ejecutar receta nueva

Caso de uso - listar preguntas:

El caso de uso inicia cuando el paciente ingresa a la aplicación móvil (previamente autenticado con su cédula y código de verificación). El componente Notification se encarga de resolver las preguntas sin responder para el paciente en cuestión.

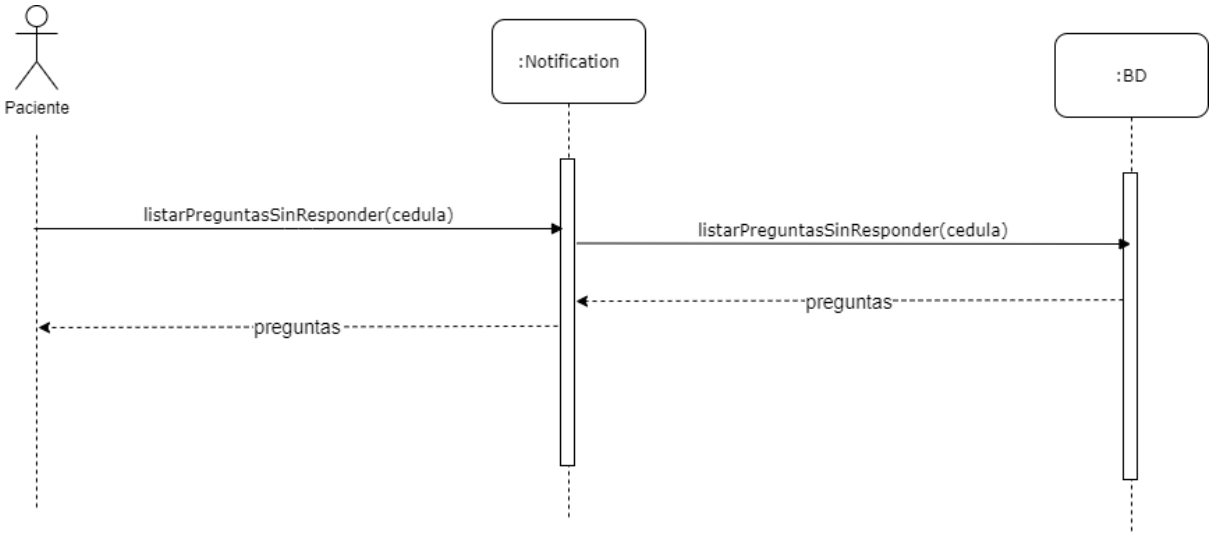


Figura 22: Caso de uso - Listar preguntas

Caso de uso - responder pregunta:

Uno de los desafíos que enfrentó el equipo estuvo en la “pausa” de la regla al generarse una pregunta, y en la “reanudación” de la misma al generarse una respuesta (asincronicidad). Que en definitiva, se traduce a una pausa y reanudación de un código python (la solución se explicó más en detalle en la sección 4.1.3).

Como se puede ver en el diagrama de flujo, el paciente al responder una pregunta también genera una reejecución del código python, esta vez con una respuesta a la última pregunta almacenada en el componente Notification.

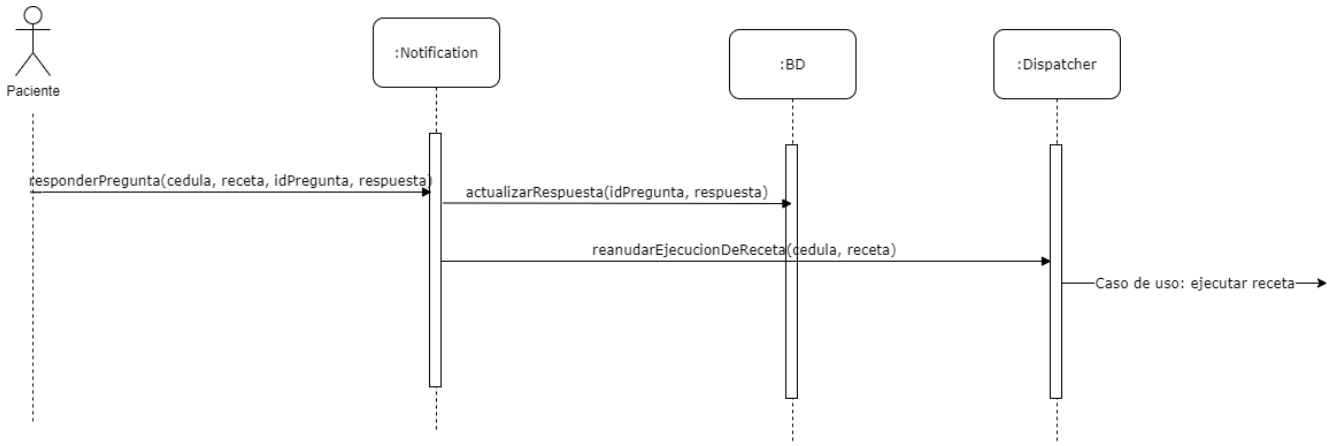


Figura 22: Caso de uso - Responder pregunta

Caso de uso - ver respuestas de paciente:

El actor de este caso de uso es el médico clínico, quien en la consulta, desea ver las preguntas que se le hicieron al paciente y todo lo que respondió para poder analizarlo en el consultorio.

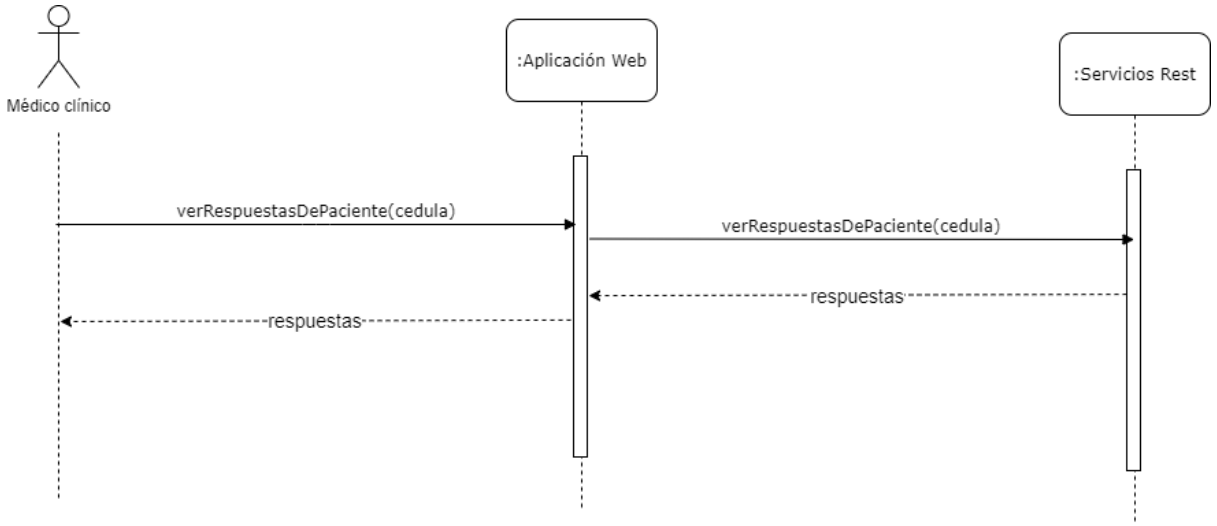


Figura 23: Caso de uso - Ver respuesta de paciente

4.4 Modelo de dominio

En la siguiente imagen se muestra el modelo de dominio de SIMIC 2.0.

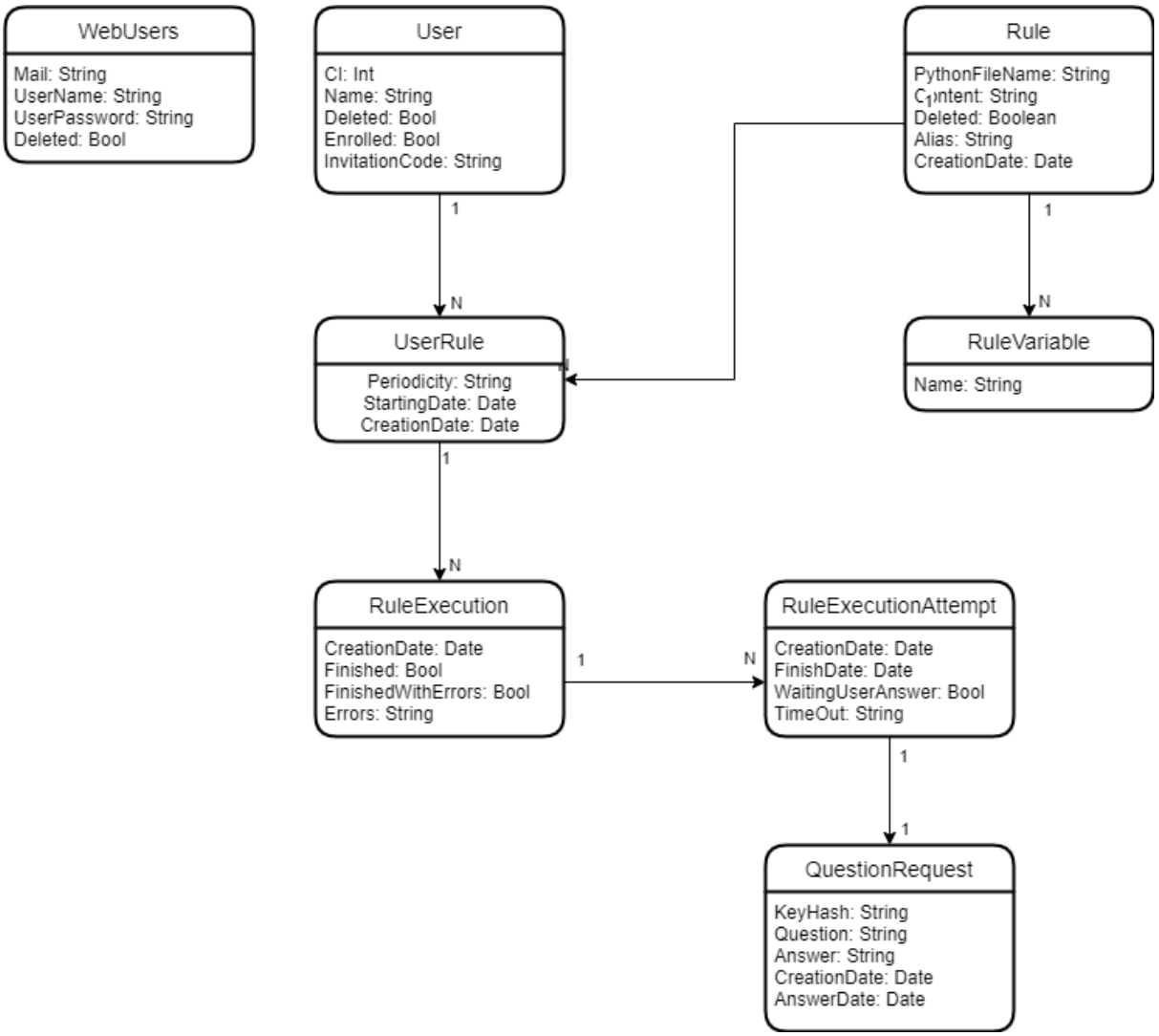


Figura 24: Modelo de dominio SIMIC 2.0

Tomando en cuenta los requerimientos, se necesita modelar tanto usuarios como reglas y la relación entre ellos que le llamamos UserRule. Cada UserRule va a tener muchas RuleExecution ya que esto se ejecuta con una determinada periodicidad que la elige el médico clínico al asignar la regla al usuario.

Por otra parte cada RuleExecution va a tener N intentos que es la tabla llamada RuleExecutionAttempt, esto va a depender exclusivamente de lo que haya programado el médico autor en la regla de seguimiento.

Por último, como se mencionó en la arquitectura específica de SIMIC 2.0 se necesitó sincronizar los usuarios de SIMIC a SIMIC 2.0 y esto se logró persistir con la tabla WebUsers.

4.5 Lenguajes y tecnologías utilizadas

Java + Spring
Los integrantes del proyecto SIMIC 2.0 cuentan con alrededor de 4 años de experiencia en Java. Siendo este el principal lenguaje usado en las experiencias laborales que han tenido. El hecho de conocer esta tecnología fue el principal motivo para decidirse a usar Java como lenguaje de programación del lado del servidor. Cuenta con una gran cantidad de bibliotecas conocidas por los integrantes del equipo, que brindan facilidades a la solución de problemas descritos en la arquitectura.

Spring es un framework de Java que ofrece como elemento clave el soporte de infraestructura a nivel de aplicación, brindando un completo modelo tanto para la configuración como para la programación de aplicaciones empresariales desarrolladas bajo Java, sin discriminación en cuanto al despliegue de la plataforma. Todo esto trae consigo una gran ventaja, ya que permite que los equipos de desarrollo puedan enfocarse directamente en la lógica empresarial que requiere la aplicación, haciendo el proceso más corto, rápido y eficaz, ahorrando líneas de código evitando tareas repetitivas [19].

Los componentes “Servicios Rest”, “Scheduler”, “Dispatcher” y “Notification” fueron programados en Java y exponen sus servicios gracias a la facilidad que provee Spring a la hora de la creación de APIs Rest.

Django
Django es un framework de Python para el desarrollo web, de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático [20].

Esta tecnología fue utilizada para el componente Python descrito anteriormente en la arquitectura, que expone un servicio para que el componente Dispatcher le pueda indicar la ejecución de cierto código python (regla).

Git
Se utilizó Git como repositorio del proyecto, este es un software que permite que varios desarrolladores trabajen al mismo tiempo y en paralelo en un proyecto con un acceso compartido. Así como también en identificar que usuario y cuando ha realizado cada modificación. Además, cada desarrollador cuenta con una copia local de todo el proyecto y de los cambios generados. Esta tecnología permite comparar, fusionar o restaurar versiones de una aplicación y contar con una copia del código fuente para volver atrás ante imprevistos. Otra de las ventajas es que es software libre, open source y es multiplataforma por lo que se puede utilizar en todos los sistemas operativos [21].

React

Para el desarrollo del frontend Web se consideraron dos tecnologías, React.js [25] y Angular [26]. Se descartó la idea de utilizar únicamente HTML + javascript porque supondría un desarrollo extenso, se prefirió usar una herramienta que garantice un desarrollo más ordenado, de menor extensión en cuanto a código, con mayor legibilidad y mantenibilidad del mismo.

Tabla comparativa entre Angular y React:

Tecnología	Angular	React
Tipo	Framework completamente desarrollado: ofrece una fuerte opinión sobre cómo debe estructurarse la aplicación y tiene muchas bibliotecas pequeñas integradas que lo ayudan a crear aplicaciones complejas.	Es una biblioteca que necesita de otras para lograr la funcionalidad de un framework, pero brinda más libertad de la forma en que se organiza el código.
Data binding	Data binding bidireccional, lo que significa que si cambiamos la entrada en la interfaz de usuario, cambiará el estado del modelo y viceversa.	Data binding unidireccional, lo que significa que un elemento de la interfaz de usuario no puede cambiar el estado de un componente.
Lenguaje	TypeScript: superconjunto de JavaScript y lenguaje estáticamente escrito.	JavaScript XML (JSX): se puede escribir en TypeScript, aunque no se incluye de forma nativa.
Estrellas en GitHub	40.963	111.927
Contribuyentes en GitHub	732	1.242
Reportes de errores en GitHub	2.162	287
Conocimiento del equipo de desarrollo	Nivel bajo	Nivel medio

Debido a que los integrantes tenían más dominio de React y que en comparación ambas tecnologías son muy utilizadas se decidió utilizar React.

React Native
React Native es un marco de aplicación móvil de código abierto creado por Facebook. Se utiliza para desarrollar aplicaciones para Android y iOS y permite a los desarrolladores usar React junto con las capacidades de la plataforma nativa [22].

Se decidió utilizar React Native ya que por un lado React Native utiliza el núcleo de React entonces íbamos a estar familiarizados y por otro ya que con el mismo código íbamos a poder generar la aplicación tanto para Android como para iOS. Cabe aclarar que los integrantes no teníamos experiencia con el desarrollo móvil.

MySQL
Para la base de datos se utilizó MySQL, ya que es independiente del lenguaje de programación que se utilice y también está disponible para casi cualquier sistema operativo, esto lo hace un sistema de base de datos muy potente. Es fácil de usar y tiene una amplia comunidad que lo respalda, por lo que se encuentran respuestas a dudas con facilidad [23]. Además es gratis, open source y ya se contaba con experiencia trabajando con esta herramienta por lo que el equipo se siente cómodo usando este motor de base de datos.

Docker
Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos [24].

Esta tecnología se utilizó para el despliegue de cada uno de los componentes mencionados en la arquitectura anteriormente.

Google Firebase
Firebase Cloud Messaging (FCM) es una solución de mensajería multiplataforma que te permite enviar mensajes de forma segura y gratuita.

Con FCM, se puede notificar a una app cliente que un correo electrónico nuevo o que otros datos están disponibles para la sincronización. También se puede enviar mensajes de notificación para volver a atraer a más usuarios y aumentar su retención [32].

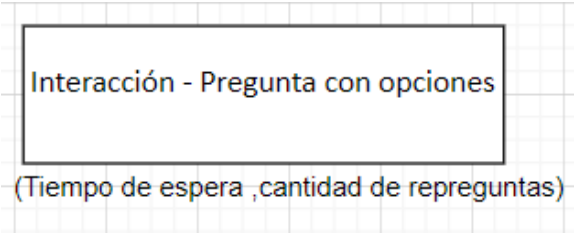
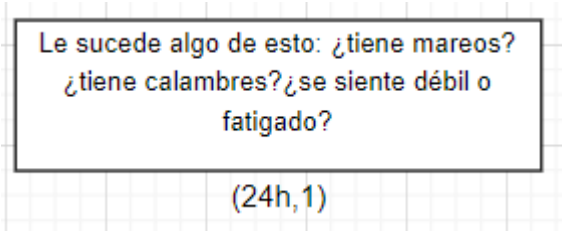
Capítulo 5

Implementación

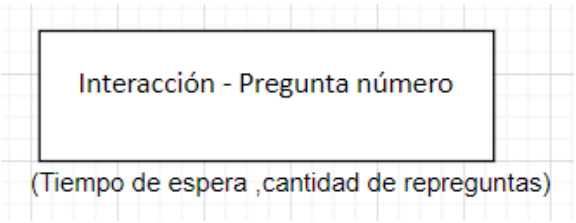
5.1 Lista de traducciones

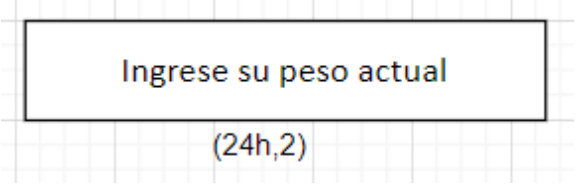
En esta sección se muestra una tabla con objetos genéricos para la traducción del diagrama de flujo a receta Python.

Estructura en diagrama de flujo	Traducción en Python

Estructura general:
 
 	
Ejemplo:
 
 	
	Interfaz del método:
	`PreguntaConOpciones(pregunta, opciones, timeOut, cantidadDeRepreguntas=0, secuencial=0)`
	
	Ejemplo:
	`UN_DIA = "P1D"`
	`respuesta = Interaccion.PreguntaConOpciones('Le sucede algo de esto: ¿tiene mareos?¿tiene calambres?¿se siente débil o fatigado?',['si','no'], UN_DIA,1)`

Estructura general:	Interfaz del método:
	`PreguntaNumero(pregunta, defaultNumber, timeOut, cantidadDeRepreguntas=0, secuencial=0)`
	
	Ejemplo:
	`UN_DIA = "P1D"`

<table>
<tr>
<td> 

Ejemplo:

</td>
<td>

```
respuesta =
Interaccion.PreguntaNumero('Ingrese
su peso actual', 70, UN_DIA,2)
```

</td>
</tr>
<tr>
<td>Estructura general:


Ejemplo:

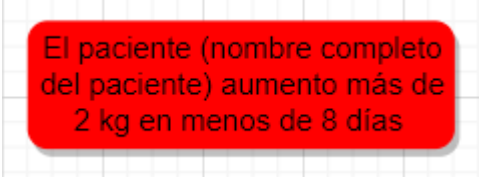
</td>
<td>Interfaz del método:

```
Esperar(tiempo)
```

Ejemplo:

```
SIETE_DIAS = "P7D"
Interaccion.Esperar(SIETE_DIAS)
```

</td>
</tr>
<tr>
<td>Estructura general:


Ejemplo:

</td>
<td>Interfaz del método:

```
CrearAlerta(mensaje, gravedad=0)
```

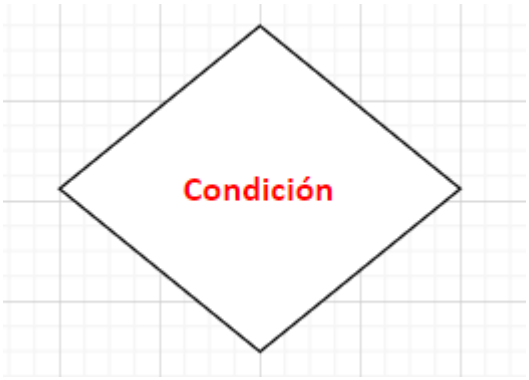
Ejemplo:

```
Alerta.CrearAlerta('El paciente ' +
PersonaSimic().primernombre() + ' ' +
PersonaSimic().primerapellido() + '
aumento mas de 2 kg en menos de 8
días', 2)
```

</td>
</tr>
<tr>
<td>Estructura general:</td>
<td>Ejemplo:

```
if (pesoIngresado - pesoSeco) > 2:
else:
```

</td>
</tr>
</table>



Ejemplo:

5.2 Traducción del diagrama de flujo a receta Python

En esta sección se muestra la traducción de la receta Peso, luego como Anexo se adjunta la traducción de la receta Disnea. Por otra parte se adjunta como anexo el Manual del Médico Autor donde se explican todas las funciones que nos provee los módulos de SIMIC 2.0 y los parámetros de cada función.

Traducción receta Peso:

La receta comienza con la inclusión de los módulos que provee SIMIC 2.0.

```
import Interaccion
import Variable
import Alerta
from PersonaSimic import PersonaSimic
```

Figura 25: módulos SIMIC 2.0

El diagrama de flujo de la receta Peso comienza con la siguiente secuencia:

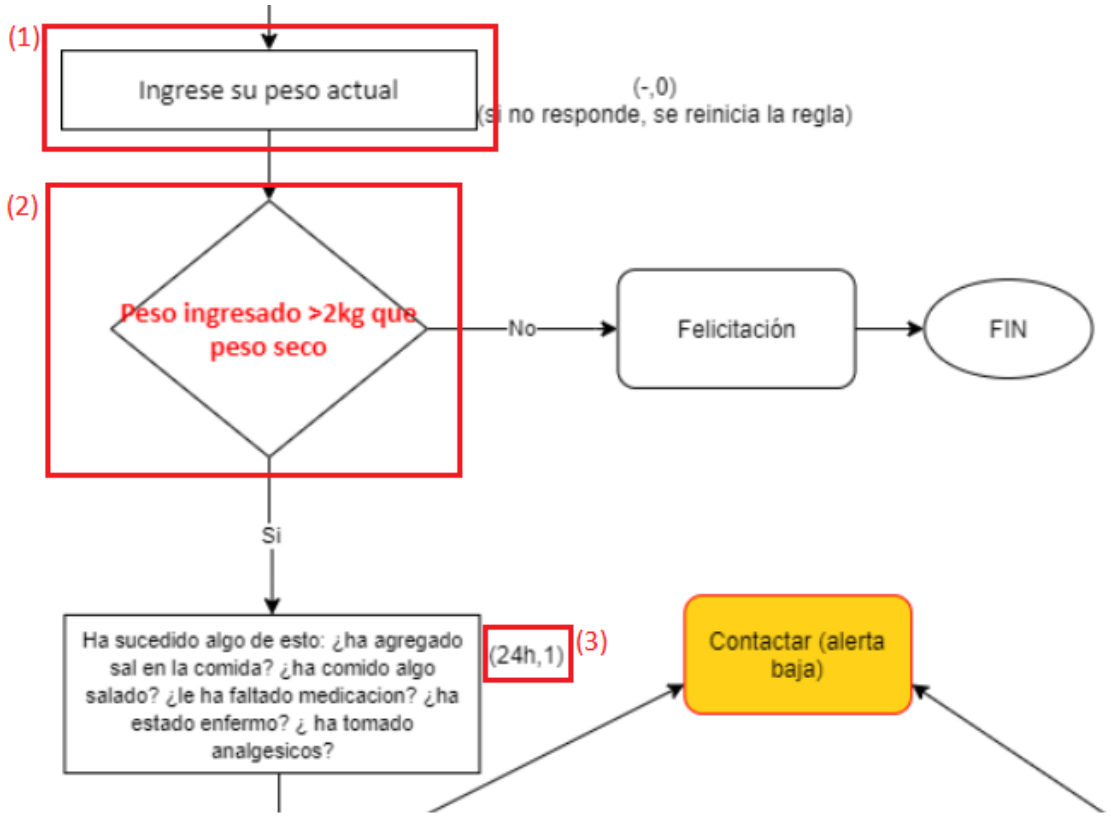


Figura 26: Primera secuencia receta Peso

Al comienzo vemos que se realiza una interacción con el paciente, el cual le realiza una pregunta que se tiene que responder con un número marcado en rojo con (1), luego con la respuesta del paciente se realiza una comparación, esta comparación depende de un dato almacenado en SIMIC 2.0 que es el peso seco y de la respuesta (2). Dependiendo del resultado de la comparación es si el flujo sigue por el Sí o por el No. Si el resultado de la comparación es Si, la siguiente pregunta hacia el usuario se puede responder como máximo en 24 horas, de lo contrario SIMIC 2.0 entenderá que se debe seguir el flujo pero con un valor “timeout” a menos que se tenga que volver a preguntar nuevamente que este es el caso mostrado en el punto (3).

Todo esto se traduce a Python de la siguiente forma:

```
respuesta = Interaccion.PreguntaNumero('Ingrese su peso actual', 70, UN_DIA)
pesoSeco = Variable.ObtenerVariable('peso_seco')
if (respuesta - pesoSeco) > 2:
    respuesta = Interaccion.PreguntaConOpciones('Ha sucedido algo de esto: ¿ha
agregado sal en la comida? ¿ha comido algo salado? ¿le ha faltado medicación?
¿ha estado enfermo? ¿ ha tomado analgesicos?', ['si','no'], UN_DIA, 1)
```

Figura 27: Primera secuencia receta Peso en código Python

Luego se muestra una parte interesante de la receta peso en el diagrama de flujo:

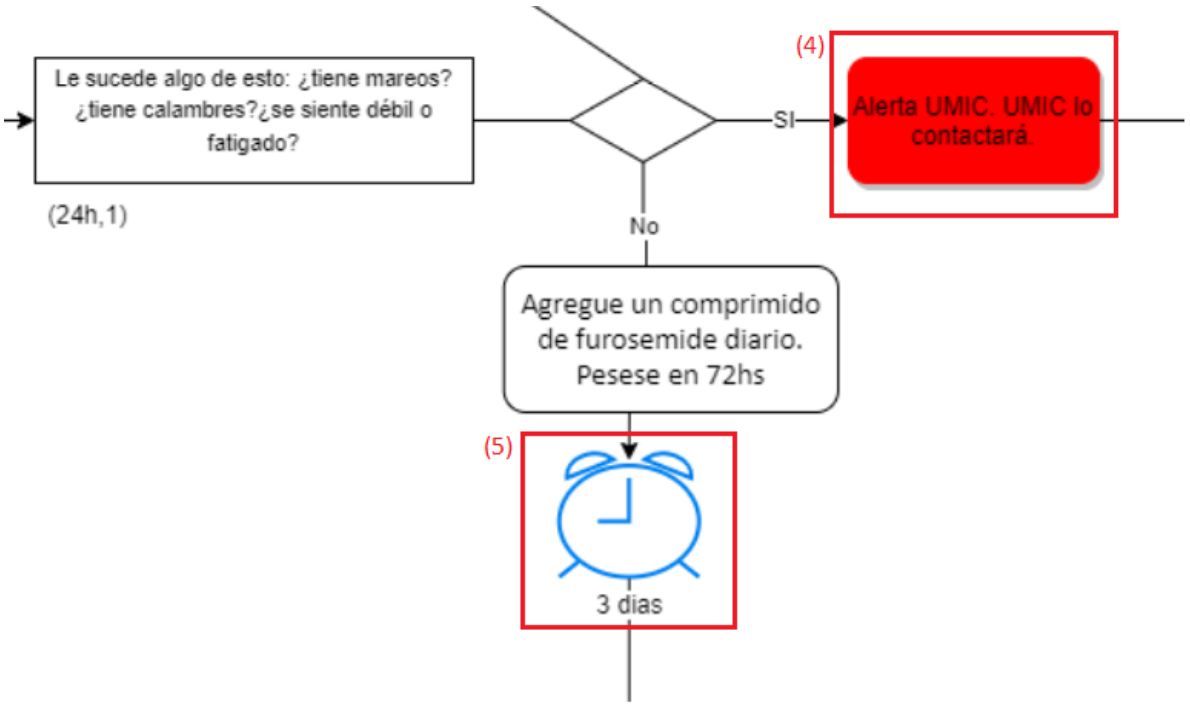


Figura 28: Secuencia interesante en la receta Peso

Como se puede ver en (4) se necesita alertar al cuerpo médico de la UMIC. Luego en (5) se necesita pausar la receta por determinado tiempo y luego seguir. Esto se puede ver implementado en el siguiente fragmento de código:

```
      respuesta = Interaccion.PreguntaConOpciones('Le sucede algo de esto:
¿tiene mareos?¿tiene calambres?¿se siente débil o fatigado?', ['si','no'],
UN_DIA, 1)
       if respuesta == 'si':
            #Alertamos a el medico y le decimos al paciente que umic lo
contactara
            Interaccion.PreguntaConOpciones('Pongase en contacto con la UMIC
al telefono 2487 1515', ['Entendido'], UN_DIA)
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' se siente mareado', 2)
        elif respuesta == 'no':
            Interaccion.PreguntaConOpciones('Agregue un comprimido de
furosemide diario',['Entendido'], UN_DIA)
            Interaccion.Esperar(TRES_DIAS)
```

Figura 29: Secuencia interesante en la receta Peso en código Python

A continuación se muestra la receta completa en Python:

```
import Interaccion
import Variable
```

```
import Alerta
from PersonaSimic import PersonaSimic

## RECETA PESO ##
UN_DIA = "P1D"
TRES_DIAS = "P3D"
SIETE_DIAS = "P7D"

respuesta = Interaccion.PreguntaNumero('Ingrese su peso actual', 70, UN_DIA)
pesoSeco = Variable.ObtenerVariable('peso_seco')
if (respuesta - pesoSeco) > 2:
    respuesta = Interaccion.PreguntaConOpciones('Ha sucedido algo de esto: ¿ha agregado sal en la comida? ¿ha comido algo salado? ¿le ha faltado medicacion? ¿ha estado enfermo? ¿ ha tomado analgesicos?', ['si','no'], UN_DIA, 1)
    if respuesta == 'si':
        Interaccion.PreguntaConOpciones('Recomendación higiénico dietética', ['Entendido'], UN_DIA)
        Interaccion.Esperar(SIETE_DIAS)
        respuesta = Interaccion.PreguntaNumero('Ingrese su peso actual', 70, UN_DIA, 2, 1)
        if (respuesta - pesoSeco) > 2:
            #Alertamos a el medico y le decimos al paciente que umic lo contactara
            Interaccion.PreguntaConOpciones('Pongase en contacto con la UMIC al telefono 2487 1515', ['Entendido'], UN_DIA)
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + ' aumento mas de 2 kg en menos de 8 días', 2)
        elif respuesta == 'no':
            Interaccion.PreguntaConOpciones('Continue así!', ['Entendido'], UN_DIA)
        else:
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + ' no responde su peso actual')
    elif respuesta == 'no':
        respuesta = Interaccion.PreguntaConOpciones('Le sucede algo de esto: ¿tiene mareos?¿tiene calambres?¿se siente débil o fatigado?', ['si','no'], UN_DIA, 1)
        if respuesta == 'si':
            #Alertamos a el medico y le decimos al paciente que umic lo
```

```
contactara
            Interaccion.PreguntaConOpciones('Pongase en contacto con la UMIC
al telefono 2487 1515', ['Entendido'], UN_DIA)
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' se siente mareado', 2)
        elif respuesta == 'no':
            Interaccion.PreguntaConOpciones('Agregue un comprimido de
furosemide diario',['Entendido'], UN_DIA)
            Interaccion.Esperar(TRES_DIAS)
            respuesta = Interaccion.PreguntaNumero('Ingrese su peso actual',
70, UN_DIA, 1)
            if (respuesta - pesoSeco) > 2:
                #Alertamos a el medico y le decimos al paciente que umic lo
contactara
                Interaccion.PreguntaConOpciones('Pongase en contacto con la
UMIC al telefono 2487 1515', ['Entendido'], UN_DIA)
                Alerta.CrearAlerta('El paciente ' +
PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + '
aumento mas de 2 kg en menos de 8 días', 2)
            elif respuesta == 'no':
                Interaccion.PreguntaConOpciones('Continue así!',
['Entendido'], UN_DIA)
            else:
                Alerta.CrearAlerta('El paciente ' +
PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + ' no
responde su peso actual')
        else:
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' no responde a la pregunta de si
tiene mareos, calambres o se siente debil')
    else:
        Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre() + '
' + PersonaSimic().primerapellido() + ' no responde a la pregunta de si agrego
sal a la comida')
elif respuesta != 'timeout':
    Interaccion.PreguntaConOpciones('Felicitaciones! continue asi', ['OK'],
UN_DIA)
```

Figura 30: Receta Peso en código Python

Luego se muestra la traducción de la receta Disnea:

```
import Interaccion
import Variable
import Alerta
from PersonaSimic import PersonaSimic

## REGLA DISNEA ##
UN_DIA = "PT3M"
SIETE_DIAS = "PT4M"

respuesta = Interaccion.PreguntaConOpciones('¿Ha agregado falta de aire o
dificultad para respirar últimamente?',['si','no'], UN_DIA)
if respuesta == 'si':
    respuesta = Interaccion.PreguntaConOpciones('¿ha agregado sal en la
comida? ¿ha comido algo salado? ¿le ha faltado medicacion? ¿ha estado enfermo?
¿ ha tomado analgesicos?',['si','no'],UN_DIA,1)
    if respuesta == 'si':
        Interaccion.PreguntaConOpciones('No agregue sal a la
comida',['Entendido'],UN_DIA)
        valorNuevoDelContador = Variable.ObtenerVariable('contador') + 1
        Variable.SetearVariable('contador',valorNuevoDelContador)
        if valorNuevoDelContador == 1:
            Interaccion.CambiarPeriodicidad(SIETE_DIAS)
        else:
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' Ha agregado sal a la comida varias
veces')
            Variable.SetearVariable('contador',0)
    elif respuesta == 'no':
        respuesta = Interaccion.PreguntaConOpciones('¿Le sucede algo de esto?
Tiene mareos, calambres o se siente debil?',['si','no'],UN_DIA,1)
        if respuesta == 'si':
            #Alertamos a el medico y le decimos al paciente que umic lo
contactara
            Interaccion.PreguntaConOpciones('Pongase en contacto con la UMIC
al telefono 2487 1515',['Entendido'],UN_DIA)
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' tiene mareos, o se siente debil',
2)
```

```
        elif respuesta == 'no':
            Interaccion.PreguntaConOpciones('Si usted toma diureticos, agregue
1 diuretico por dia por una semana',['ok'],UN_DIA)
            Interaccion.Esperar(SIETE_DIAS)
            respuesta = Interaccion.PreguntaConOpciones('Hace 7 días respondió
que tenía falta de aire, ¿Ha mejorado con el agregado de 1 diurético extra?
',['si','no'],UN_DIA,1)
            if respuesta == 'si':
                Interaccion.PreguntaConOpciones('Consulte a su médico para una
nueva prescripción',['Entendido'],UN_DIA)
            elif respuesta == 'no':
                #Alertamos a el medico y le decimos al paciente que umic lo
contactara
                Interaccion.PreguntaConOpciones('Pongase en contacto con la
UMIC al telefono 2487 1515',['Entendido'],UN_DIA)
                Alerta.CrearAlerta('El paciente ' +
PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + ' no
mejoro con el agregado del diuretico extra', 2)
            else:
                Alerta.CrearAlerta('El paciente ' +
PersonaSimic().primernombre() + ' ' + PersonaSimic().primerapellido() + ' no
respondió si mejoro con el agregado del diuretico')
        else:
            Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre()
+ ' ' + PersonaSimic().primerapellido() + ' no a respondido la pregunta de si
tiene mareos, calambres o si se siente debil')
    elif respuesta == 'timeout':
        Alerta.CrearAlerta('El paciente ' + PersonaSimic().primernombre() + '
' + PersonaSimic().primerapellido() + ' no a respondido la pregunta de si
comio con sal la comida')
```

Figura 49: Receta Disnea en código Python

5.3 Data Studio

Permisos

Data Studio es una herramienta gratuita y accesible fácilmente desde la web, únicamente es necesaria una cuenta de Google [27]. Al acceder a esta funcionalidad, se podrán crear y compartir fácilmente los proyectos que se creen, enviando el vínculo generado para

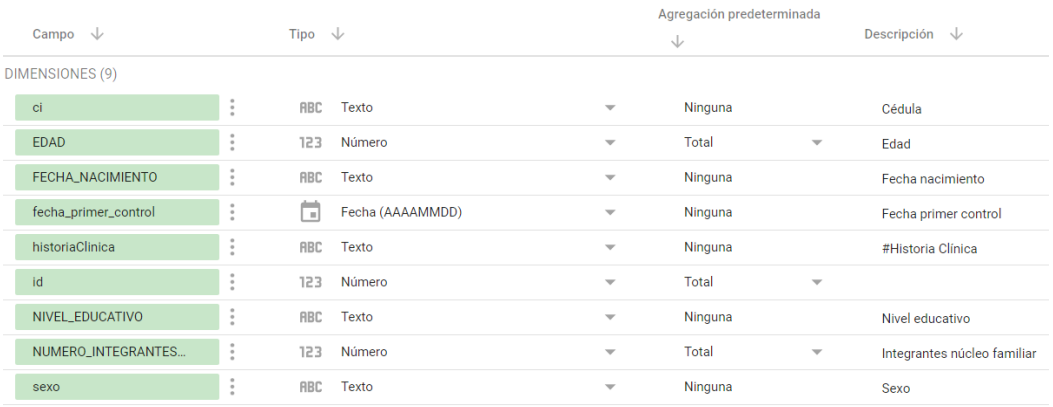
compartir o añadiendo nuevos miembros. A su vez, se podrá gestionar el acceso a estas nuevas cuentas con permisos de edición o de visualización.

El permiso de edición será para aquellas personas que sean los responsables de generar fuentes de datos, y crear los gráficos. Mientras que el de visualización tendrá acceso a los gráficos pudiendo únicamente interactuar con estos a través de los filtros que se hayan puesto a disposición.

Fuentes de datos

La herramienta Data Studio presenta las ventajas mencionadas en el sección 4, aquí se menciona el hecho de que es muy fácil la conexión hacia la base de datos y luego el uso de estos para generar gráficas y listados.

Para poder conectarse hacia la base de datos de SIMIC 1.0 se utilizó una opción de Data Studio que permite ingresar una consulta SQL personalizada y luego interactuar con los campos obtenidos. Esta misma fuente de datos se puede reutilizar en los distintos gráficos y en cada uno de ellos agruparlos de distintas formas.



Campo	Tipo	Agregación predeterminada	Descripción
DIMENSIONES (9)			
ci	ABC Texto	Ninguna	Cédula
EDAD	123 Número	Total	Edad
FECHA_NACIMIENTO	ABC Texto	Ninguna	Fecha nacimiento
fecha_primer_control	Fecha (AAAAMMDD)	Ninguna	Fecha primer control
historiaClinica	ABC Texto	Ninguna	#Historia Clínica
id	123 Número	Total	
NIVEL_EDUCATIVO	ABC Texto	Ninguna	Nivel educativo
NUMERO_INTEGRANTES...	123 Número	Total	Integrantes núcleo familiar
sexo	ABC Texto	Ninguna	Sexo

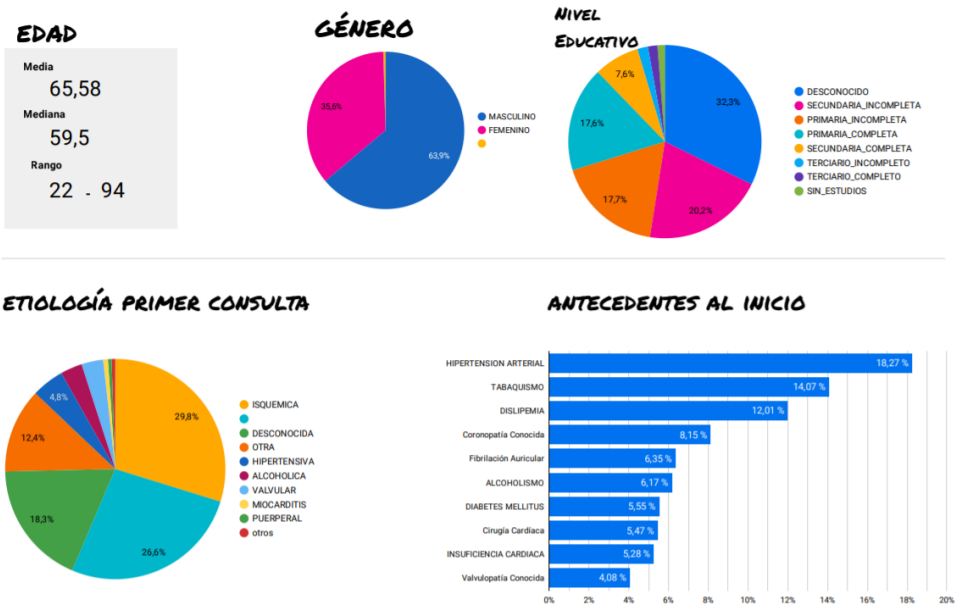
Figura 31: Herramienta Google Data Studio

Elección de gráficos

Luego de tener la o las fuentes de datos agregadas, a continuación se elige el tipo de gráfico a crear.
Se generaron gráficas de tipo circulares, de barra, indicadores de datos relevantes y planillas con filtros.

Filtros

Dentro de los listados es posible agregar filtros de aquellos campos que se hayan agregado anteriormente, y estos filtros podrán ser utilizados en tiempo real en el momento de visualización del gráfico.



DATOS BÁSICOS DEL PACIENTE

sexo | historiaClinica | EDAD

Igual a | Introduzca un... | Igual a | Introduzca un...

	ci	historiaClini...	sexo	FECHA_NACIMIENTO	EDAD	fecha_primer_control	situacion_vital	NIVEL_EDUCATI...	NUMERO_INTEGRANTES_NUCLEO_FAMILIAR
1.	9984596	733775	MASCULINO	1947-12-20 00:00:00.0	73	3 nov 2008	FALLECIDO	DESCONOCIDO	0
2.	9951692	232174	FEMENINO	1939-07-19 00:00:00.0	81	23 abr 2010	FALLECIDO	DESCONOCIDO	0
3.	9829499	732664	FEMENINO	1932-12-23 00:00:00.0	88	5 mar 2012	CENSURADO	DESCONOCIDO	0
4.	9810218	713658	MASCULINO	1943-01-06 00:00:00.0	78	31 may 2010	CENSURADO	DESCONOCIDO	0
5.	9809039	774568	MASCULINO	1937-12-26 00:00:00.0	83	30 nov 2007	CENSURADO	DESCONOCIDO	0
6.	9768348	711841	MASCULINO	1944-04-04 00:00:00.0	76	31 oct 2017	VIVO	DESCONOCIDO	0
7.	9751248	902417	FEMENINO	1944-05-04 00:00:00.0	76	11 abr 2011	VIVO	DESCONOCIDO	0
8.	9739757	741751	MASCULINO	1942-10-20 00:00:00.0	78	3 mar 2015	CENSURADO	DESCONOCIDO	0
9.	9723261	715375	MASCULINO	1942-04-26 00:00:00.0	78	15 oct 2010	FALLECIDO	DESCONOCIDO	0
10.	9696892	761854	FEMENINO	1944-05-07 00:00:00.0	76	26 jul 2013	VIVO	DESCONOCIDO	0

Figura 32: Gráficos obtenidos a partir de Google Data Studio

Capítulo 6

Gestión de calidad

En esta sección se tratará aquellos puntos relacionados con la calidad de software que tienen como objetivo garantizar el correcto funcionamiento del sistema.

6.1 Pruebas unitarias y de componentes

Se realizaron pruebas unitarias a aquellos componentes más relevantes en la hora del desarrollo para corroborar un correcto funcionamiento.

Para probar los componentes de los servicios REST, de forma aislada, lo que se hizo fue utilizar la aplicación Postman.

Desde Postman se simularon los llamados a los módulos de los servicios REST, tanto de los métodos POST y GET.

También se utilizó esta aplicación para simular endpoints para los métodos que consumen apis externas, devolviendo en cada caso, la respuesta esperada para cada prueba.

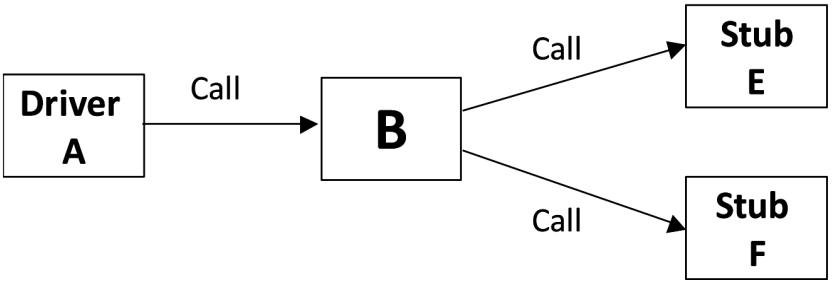


Figura 33: Pruebas unitarias

6.2 Pruebas de integración

Dado que la arquitectura contiene múltiples componentes, durante cada iteración de desarrollo se tuvo que probar el correcto funcionamiento de cada uno de ellos. Esto implicó hacer pruebas de regresión, para probar funcionalidades que ya habían sido testeadas anteriormente.

A continuación se detallan los componentes que se testean en cada prueba de regresión:

Componente	**Estado**
Aplicación Web	Chequeado
Extensión de Chrome	Chequeado
Dispatcher	Chequeado
Notification	Chequeado
Servicios Rest	Chequeado
Scheduler	Chequeado
Python	Chequeado
Aplicación móvil	Chequeado
VS Code	Chequeado
Firebase	Chequeado
Google DataStudio	Chequeado

6.3 Pruebas del sistema

A continuación se muestran los requerimientos y su aprobación por el equipo de médicos de la UMIC y los estudiantes de medicina:

Requerimiento	**Estado**
Login web administración de reglas	Chequeado
Visualización detalle del paciente	Chequeado
Ingreso de nueva regla	Chequeado
Asociar regla a un paciente	Chequeado
Agregar variable al paciente	Chequeado
Obtener código de invitación para paciente	Chequeado
Ver respuestas de pacientes	Chequeado
Ver alertas	Chequeado
Autenticación en la aplicación	Chequeado
Respuesta a preguntas desde la aplicación móvil	Chequeado
Indicadores sobre datos SIMIC 1.0	Chequeado
Obtención de datos del paciente SIMIC 1.0	Chequeado
Obtención de medicamentos por paciente SIMIC 1.0	Chequeado
Obtención de datos por consulta SIMIC 1.0	Chequeado

6.4 Pruebas de aceptación

Las pruebas de aceptación son aquellas que se le muestra el sistema al cliente y “se verifica que cumpla con sus expectativas”.

En cada reunión se les mostró al equipo de médicos de la UMIC y a los estudiantes de medicina una demo de lo que se había desarrollado hasta el momento. En primera instancia solo se les mostraba la aplicación, y más adelante se les proporcionó un sistema de prueba,

y se les entregó una aplicación para que pudieran instalarselas en el celular y realizar el ciclo completo de recetar un seguimiento.

Además como última instancia se realizó una capacitación que constó de dos talleres virtuales, donde se les proporcionó una guía de uso y se creó un canal por el EVA, que oficia de repositorio de información y lugar de intercambio.

En esta instancia también se probó con éxito que cada participante de este taller pudiera instalarse correctamente la aplicación móvil en su celular Android.

Como anexos se adjuntan los manuales del médico clínico, médico autor y del paciente.

6.5 Pruebas de performance

El sistema desarrollado tiene como requisitos de infraestructura los citados en la sección 10.4. En este apartado se explica que el código fuente de SIMIC 2.0 está empaquetado de manera que sea sencillo el despliegue. Esto se hizo con la ayuda de Docker, que tiene la finalidad de que la aplicación se comporte de manera agnóstica al servidor en que se despliega, y fue también una forma de incluir buenas prácticas, y de aprender nuevas tecnologías.

Dicho esto, se entiende que no es un escenario fácil de replicar para poder probar sin afectar el funcionamiento de la aplicación, que sería lo ideal antes de poner el sistema en producción.

Por otro lado, dado el conocimiento adquirido durante todo el proyecto, se puede proyectar la cantidad de usuarios máximos y en forma concurrente que se espera tener. También se hizo un análisis de los distintos escenarios en los cuales el sistema podría tener la mayor carga.

Teniendo estas consideraciones se tomó la decisión de no realizar pruebas de performance, dado que las condiciones en las cuales se encuentra desplegada la solución no son las adecuadas para tomar como referencia estas pruebas.

6.6 Pruebas de seguridad

Para realizar pruebas de seguridad tanto en la aplicación Web como en la Móvil se decidió utilizar el Estándar de Verificación de Seguridad en Aplicaciones de OWASP (ASVS por sus siglas en inglés). El estándar de verificación de seguridad en aplicaciones es una lista de requerimientos de seguridad o pruebas que pueden ser utilizadas por arquitectos, desarrolladores, testers, profesionales de seguridad e incluso consumidores, para definir qué tan segura es una aplicación [28]. Tanto para la aplicación web como la móvil se decidió chequear contra el ASVS nivel 2 que es para aplicaciones que contienen datos sensibles y que requieren protección. El Anexo 10.6 contiene las pruebas realizadas.

Capítulo 7

Gestión del proyecto

7.1 Descripción

Este proyecto se ha realizado en base al trabajo en conjunto de sus integrantes, mediante reuniones semanales y además trabajo remoto.

Las reuniones fueron clave para el intercambio de conocimiento, resolución del problema planteado así como también de integración en momentos de desarrollo.

Por otra parte, hubo reuniones de tutoría, aquí se hizo especial énfasis en definir el enfoque y el alcance del proyecto durante las primeras etapas, y de guiar y ayudar a darle valor al mismo, durante todo el proceso.

Dentro de los encuentros, se incluyeron a los médicos especializados en la insuficiencia cardíaca y estudiantes de medicina.

7.2 Metodología

La metodología utilizada a lo largo del proyecto, fue un híbrido entre desarrollo basado en planes y desarrollo ágil.

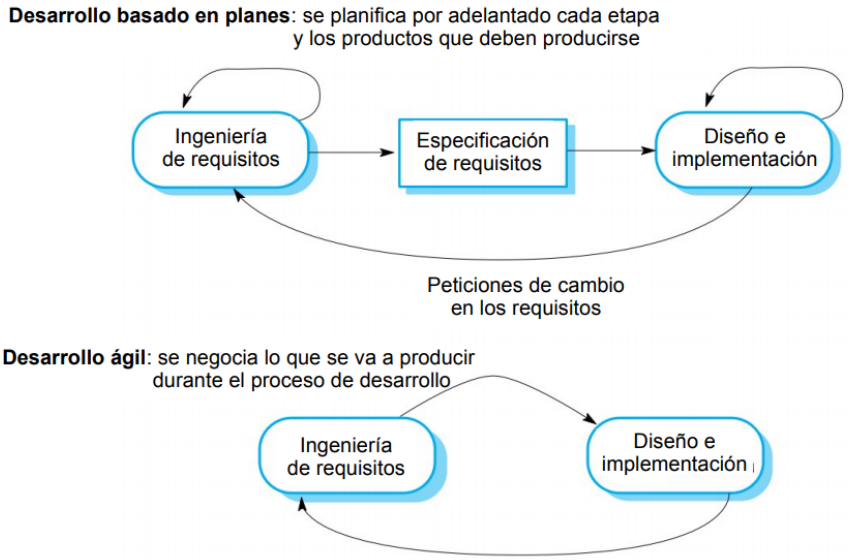


Figura 34: Metodología utilizada en el proyecto

Cada etapa comenzó con un proceso de análisis y definición de un plan, para luego separar las tareas, con sus fechas límites asociadas.
Para este proceso fue utilizada la herramienta Trello, donde se podían ver de una manera práctica la división de tareas y la asignación por usuario.
A continuación podemos ver como fue utilizada esta herramienta a medida que avanzaba el proyecto:

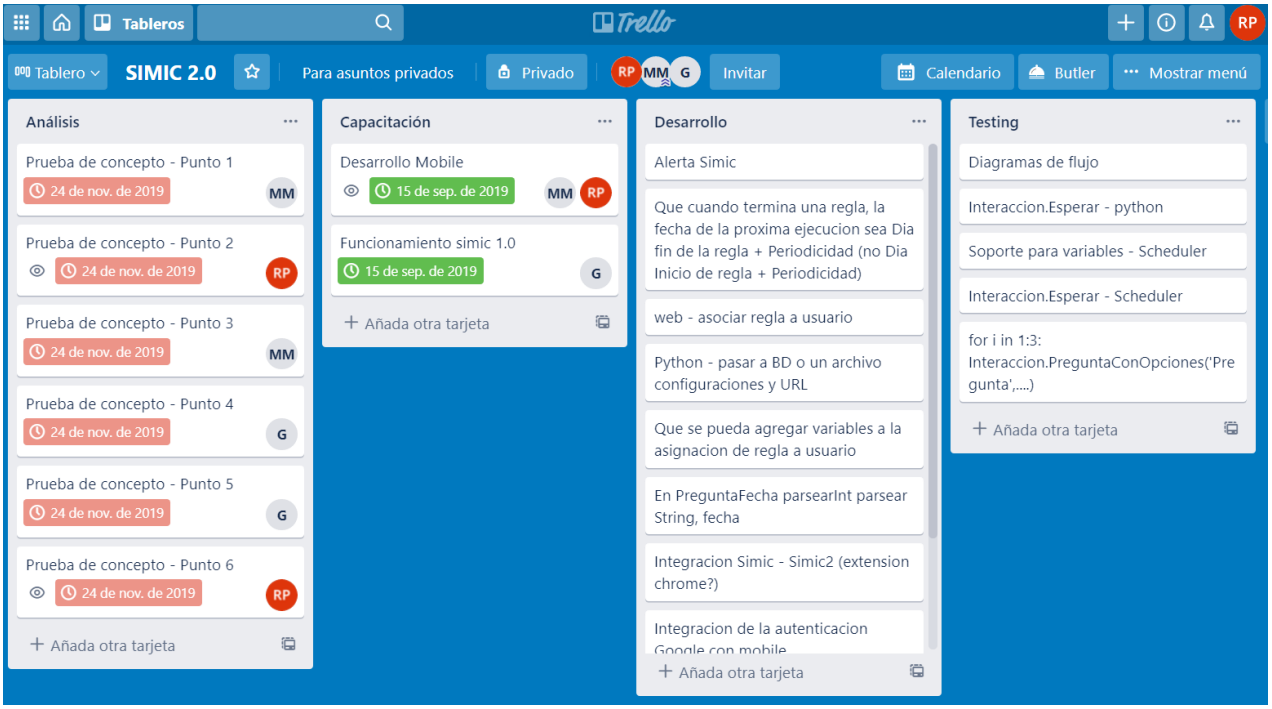


Figura 35: Herramienta Trello utilizada para la gestión del proyecto

En la etapa de implementación además de tener una planificación global de lo que se iba a hacer, se tuvieron objetivos puntuales por semana y por persona. Al término de cada

semana se integraban los distintos componentes desarrollados y se realizaba testing de cada parte y del producto completo. Esta estrategia favoreció al proyecto, de manera que se tenían resultados continuos y se podía ver junto con el cliente, mejoras o fallas del sistema.

7.3 Hitos del proyecto

Se enumeran a continuación los diferentes hitos que han marcado diferentes etapas del proyecto.

24/07/2019 - Primera reunión con el tutor

21/08/2019 - Primera reunión con el equipo de la UMIC

11/10/2019 - Primera reunión con los estudiantes de medicina

13/11/2019 - Definición de alcance con los tutores

14/06/2020 - Primer receta escrita en Python en conjunto con médicos y estudiantes de medicina

16/09/2020 - Demo de aplicación funcionando con recetas para tutores, médicos y estudiantes de medicina

14/10/2020 - Presentación en Ingeniería de muestra: “Mi proyecto en 180 segundos” es un concurso de videos realizados por los estudiantes de fin de carrera de la Facultad, donde se debió presentar su proyecto final en menos de 3 minutos y para todo público [31][33].

18/12/2020 - Entrega de aplicación para aceptación del usuario

7.4 Etapas del proyecto

Se comenzó por ajustar el alcance y conocer los requerimientos de cada una de las partes.

Luego de cerrado el alcance, se comenzó a planificar el desarrollo del sistema.

Identificado el plan a seguir, hubo un período de creación de reglas de seguimiento, que abarcó la creación de los diagramas de flujo y la traducción de estos diagramas a código Python, mientras se continuaba con el desarrollo.

Hacia el final del proyecto, avanzada la implementación, las reuniones fueron de validación y testing sobre el sistema.

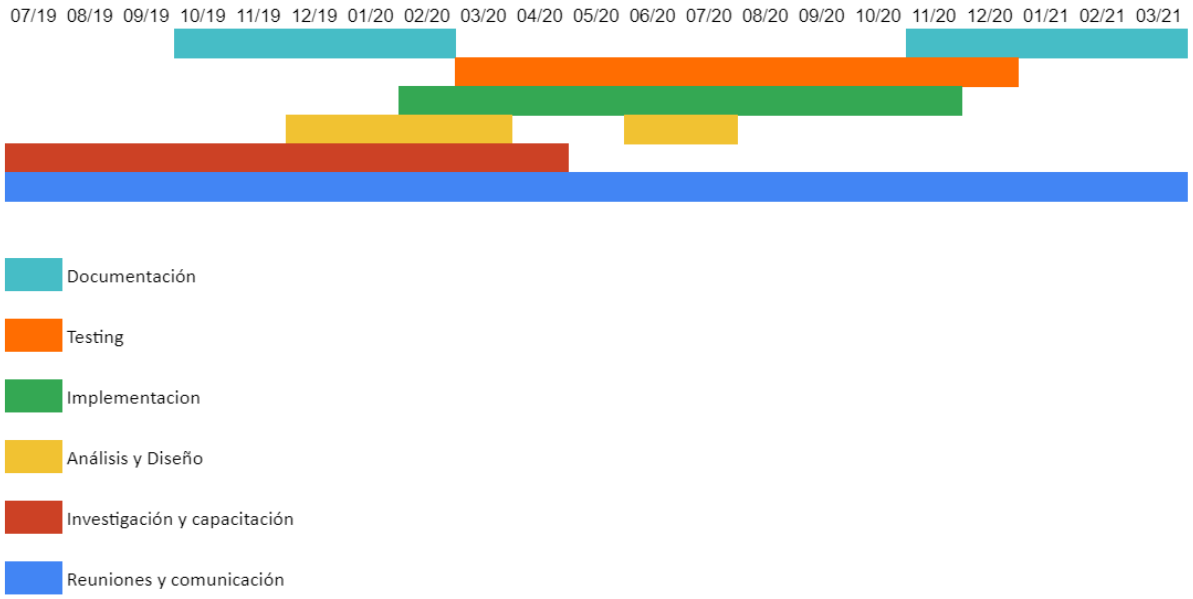


Figura 36: Etapas del proyecto

Julio 2019 - Noviembre 2019 - Definición del alcance, relevamiento de requerimientos

Diciembre 2019 - Enero 2020 - Análisis del problema específico de las reglas de seguimiento.

Febrero 2020 - Setiembre 2020 - Construcción de diagramas de flujo junto con los estudiantes de medicina y médicos de la UMIC, y paralelamente, etapa de desarrollo de la web, motor de reglas y aplicación móvil.

Abril 2020 - Octubre 2020 - Etapa de validación del sistema, donde se hicieron pruebas y correcciones con los médicos, estudiantes de medicina y tutores.

Noviembre 2020 - Diciembre 2020 - Entrega de aplicación y pruebas de aceptación con el usuario

Setiembre 2020 - Marzo 2021 - Redacción del informe final y preparación de la defensa final

7.5 División de horas por centro de costo

Analizando la división de horas por mes en la figura 37, se puede ver que este proyecto consumió un total de 1.934 horas, lo que da un promedio de 645 horas por estudiante.

Luego, el porcentaje de horas por centro de costo que se puede ver en la figura 38 demuestra que el área que abarca un mayor porcentaje de las horas dedicadas es la de las

especificación interdisciplinaria. Esto se puede ver que tuvo un gran impacto en el comienzo del proyecto, pero marca también, el trabajo en conjunto que se mantuvo a lo largo de los meses junto con los tutores y los médicos como usuarios finales. Lo más complejo de estas reuniones fue que cada integrante pudiera entender lo que realmente se necesitaba y que diera valor.

Seguido se encuentra la implementación y como tercer puesto, el de la investigación y capacitación. Estos dos puntos se corresponden con el hecho de que el sistema que se construyó fue a partir de una idea innovadora. Los primeros meses no hubo implementación, esto fue así hasta tener una idea clara de qué y cómo se iba a realizar la idea planteada en un principio.

De manera similar sucede con las pruebas, estas ocupan un 12.4 % del total, y estas estuvieron presentes en cada iteración del desarrollo, y se concentran más horas hacia el final con el sistema implementado.

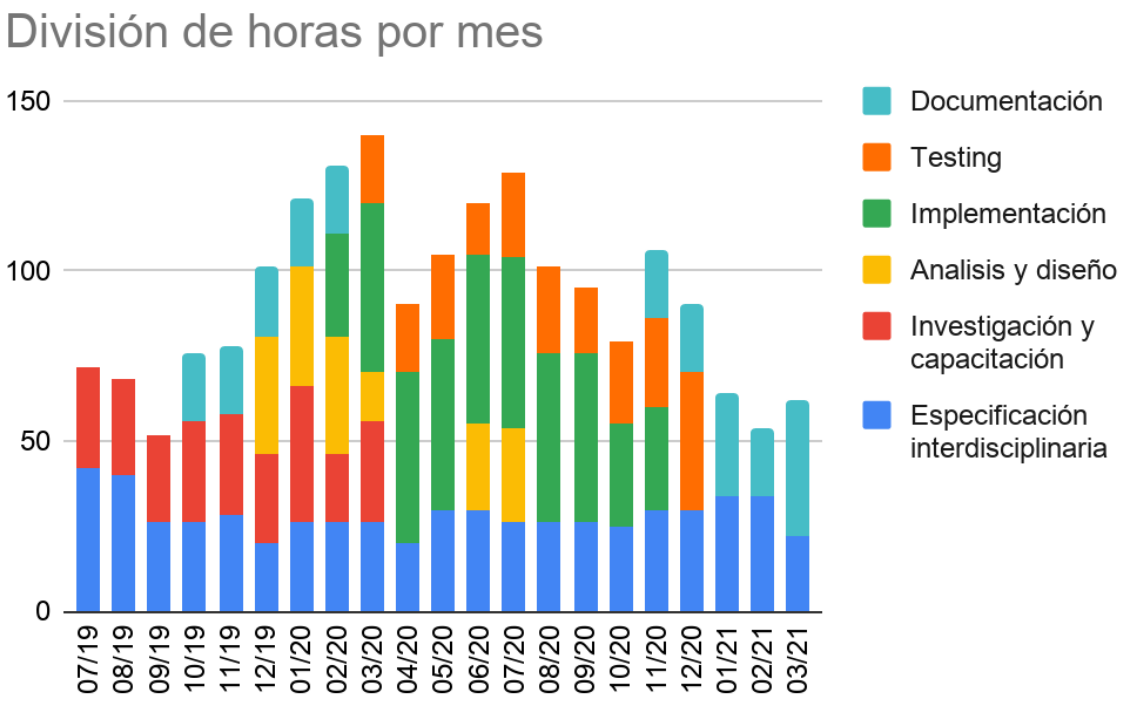


Figura 37: División de horas por mes

Figura 38: Porcentajes de horas por centro de costo

7.6 Horas de implementación por sistema

Se presentan gráficas que corresponden a la implementación de algunos de los componentes del sistema. Se puede ver que los primeros meses la mayor dedicación estuvo en el backend del sistema, y los últimos meses se puso más énfasis en el frontend. La dedicación está medida en commits, que es el conjunto de cambios en el código que se suben al repositorio.

Core

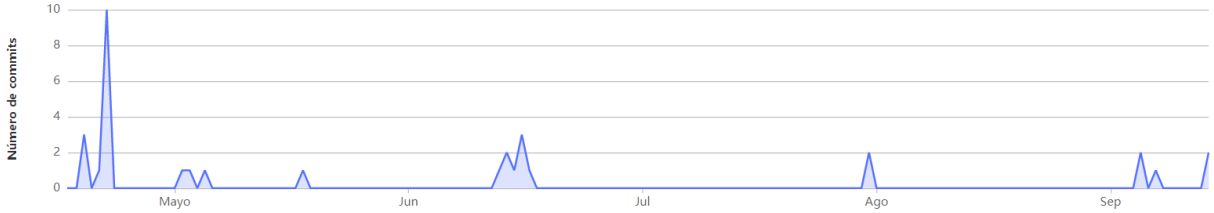


Figura 39: Número de commits en el componente Core

Rest rules

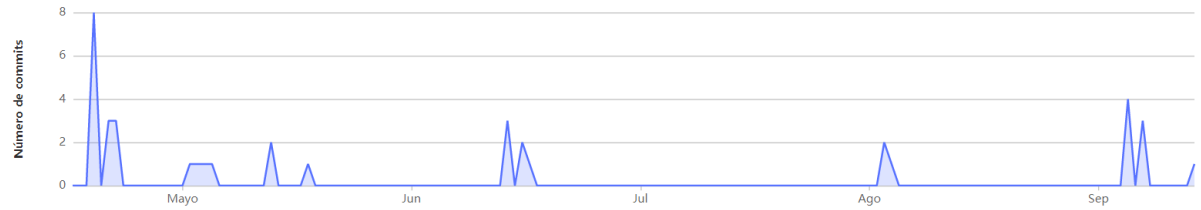


Figura 40: Número de commits en el componente Rest Rules

Django server

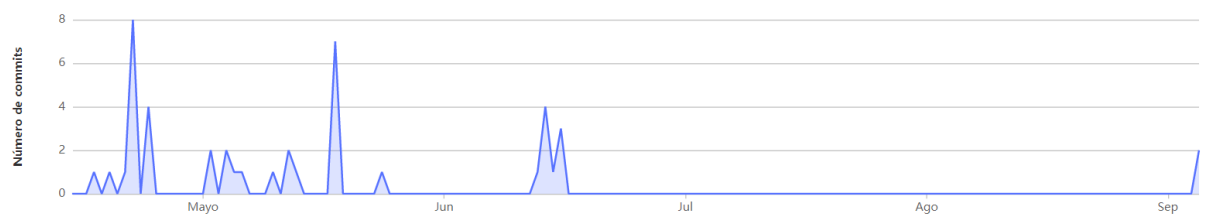


Figura 41: Número de commits en el componente Django Server

Web

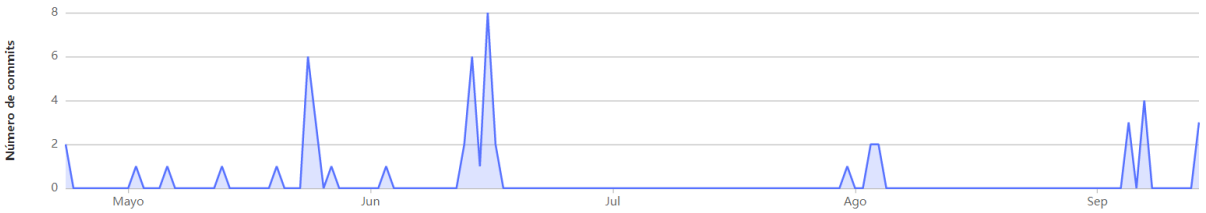


Figura 42: Número de commits en el componente Web

Móvil

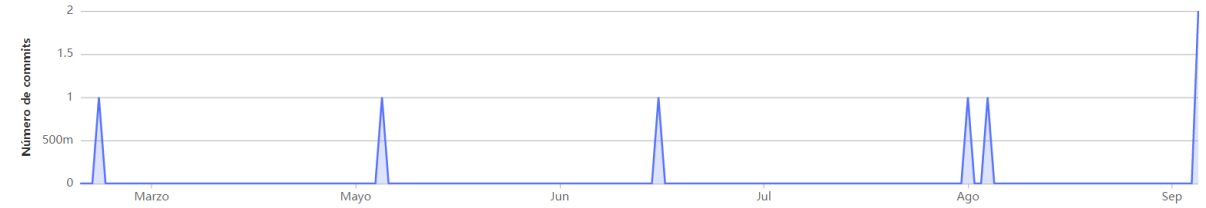


Figura 43: Número de commits en el componente Móvil

7.7 Desafíos del proyecto

El proyecto tuvo desafíos de varias índoles, que se fueron superando mientras transcurrió el mismo. De estos desafíos se generó un conocimiento muy valioso, que suma a la formación académica de cada estudiante, tanto de forma profesional como de forma humana.

Estuvo marcado desde un principio con el hecho de que fue un proyecto interdisciplinario. Esto hizo que no solo se tuviera un cliente al cual satisfacer con el producto final, sino que también se tuviera que trabajar en conjunto con él, para que el producto tuviera valor

significativo. Fue complejo el trabajo dado que cada integrante tenía una formación académica distinta, y objetivos distintos en cuanto al producto deseado.

Los médicos tenían una necesidad inicial hacia SIMIC 1.0, que tenía que ver con la extracción de datos. Y por otro lado, los tutores planteaban una nueva idea que había que desarrollarla de forma independiente. Este tema se resolvió como ya se vio anteriormente con una solución simple y con bajo costo como fue incluir la herramienta Data Studio.

Este trabajo interdisciplinario también implicó la creación en conjunto de diagramas de flujo y la traducción a lenguaje de programación, donde fue necesario la ayuda mutua entre los estudiantes de medicina y los de ingeniería para que se pudiera modelar lo que realmente necesitaban y de una forma correcta.

Al final se pudieron sortear las dificultades, y se fue llegando a un punto en común.

Otro desafío que tuvo el proyecto fue que se utilizaron tecnologías no conocidas como por ejemplo Python, React, y desarrollo móvil. Esto si bien tuvo su peso en el momento de la implementación, se pudo sortear gracias al estudio y capacitación de cada uno de los integrantes del equipo.

Capítulo 8

Conclusiones y trabajo a futuro

8.1 Conclusiones

El sistema SIMIC 2.0 cumple con los objetivos planteados al comienzo del proyecto:

- Se implementó un sistema independiente a SIMIC 1.0, genérico, y que puede ser extendido a cualquier enfermedad crónica.
- Se desarrolló un sistema para el médico autor donde se puede escribir las recetas y gestionarlas. También se implementó una aplicación móvil para el seguimiento del paciente y por último se implementó un sistema web para uso del médico clínico que permite visualizar alertas generadas por las recetas, ver las interacciones del paciente con la aplicación móvil y asignar recetas a los pacientes.
- Se logró un sistema el cual el médico tenga la libertad de programar el comportamiento de una receta, con las únicas restricciones de un lenguaje de programación. A su vez, se crearon bloques de código como ejemplo, y estos fueron entendidos por los médicos autores de manera rápida, logrando que la creación de una receta sea fácil e intuitiva dentro de las complejidades de un lenguaje de programación.
- Se pudo incluir tanto a los médicos como a los estudiantes de medicina en el diseño de cada una de las etapas de creación de las recetas, esto aportó un gran valor y enseñanza en un equipo multidisciplinario.
- Se ayudó en la creación de templates para la extracción y cruzamiento de datos clínicos de SIMIC 1.0 por medio de la herramienta Google Data Studio, logrando aportar valor a la UMIC.

- El equipo se lleva un gran aprendizaje a nivel técnico, ya que no contaba con amplia experiencia en el área de desarrollo web, móvil y python.
- El proyecto SIMIC 2.0 significó un gran crecimiento y enriquecimiento para los integrantes del proyecto ya que representó el desafío de pensar el problema uniendo los conocimientos tanto de la ingeniería como de la medicina.

En conclusión, se consiguió implementar un sistema que cumple con los requerimientos mencionados, beneficiando tanto a los pacientes de la UMIC como a sus médicos.

8.2 Trabajo a futuro

Se presentan a continuación futuras mejoras al sistema, así como extensiones, que permitan mejorar la calidad de información obtenida y mejoras a la experiencia de usuario.

- Que el sistema traduzca de diagramas de flujo exportado en formato xml a código Python y que el médico autor tenga la responsabilidad de validar la traducción y no de la implementación.
- Agregar la posibilidad de simular el tiempo en otras escalas, para que el médico autor pueda verificar rápidamente lo que especifica en Python. Esta herramienta también será útil para que el médico clínico antes de recetar, verifique el comportamiento futuro de la App, confirmando que es lo que efectivamente quiere para el paciente.
- En el motor de reglas, agregar la posibilidad de validar la correctitud de las reglas escritas en Python como manera de anticiparse a posibles escenarios no deseados. Pudiendo correr el código en un modo de prueba para asegurar de que efectivamente el código hace lo que se necesita e interactúa con el paciente de la forma deseada.
- Posibilidad de Parametrizar rango horario de ejecuciones para todas las preguntas de una regla.
- Integración con SALUD.UY, agregando información a la historia clínica electrónica nacional.
- Mejoras a la interfaz web y móvil agregando mayor valor a la usabilidad de los sistemas.
- Agregar soporte multi-idioma al sistema (aplicación web y móvil)

Capítulo 9

Referencias

[1] UMIC. (2012). Manual Práctico para el Manejo del Paciente con Insuficiencia Cardíaca Crónica por Disfunción Sistólica. http://www.umic.hc.edu.uy/images/Manual_UMIC_-_Insuficiencia_cardiaca_crnica_por_disfuncin_sistlica.pdf

[2] “HOSPITAL DE CLÍNICAS ‘Dr. Manuel Quintela.’” UMIC, www.umic.hc.edu.uy/.

[3] Alejandro Cardone, Rodrigo González y Viterbo García. “SIMIC, Sistema Médico de Insuficiencia Cardíaca”, 2016

[4] “Medisafe App - Download the App and Never Miss Another Med.” Medisafe, 1 May 2020, www.medisafeapp.com/.

[5] “RecuerdaMed, La App Que Permite Controlar La Medicación Del Paciente - Diario Dicen.” Enfermería21, 28 Dec. 2016, www.enfermeria21.com/diario-dicen/recuerdamed-la-app-que-permite-controlar-la-medicacion-del-paciente-DDIMPORT-046405/.

[6] “Integral Management for Diabetes Control.” SocialDiabetes, www.socialdiabetes.com/.

[7] “Blog De Salud y Belleza " Welvi, App Para Una Dieta Sana y Conseguir Una Vida ActivaBlog De Salud y Belleza.” Blog De Salud y Belleza, www.cosasdesalud.es/welvi-app-para-una-dieta-sana-y-conseguir-una-vida-activa/.

[8] Usuario, Super. “HOSPITAL DE CLÍNICAS ‘Dr. Manuel Quintela’ -FACULTAD DE MEDICINA - UNIVERSIDAD DE LA REPÚBLICA. Montevideo, Uruguay.” UMIC, www.medicaa.hc.edu.uy/index.php/medica-a/15-policlinicas/21-umic.

[9] Presidencia de la República Oriental del Uruguay. “Gobierno Planea Llegar Con Conexión a Internet Al 90 % De Los Hogares En 2020.” Presidencia De La República Oriental Del Uruguay,

www.presidencia.gub.uy/comunicacion/comunicacionnoticias/agenda-gobierno-conectividad-internet.

[10] “Las aplicaciones móviles de salud como herramientas de apoyo a la autogestión de cuidados del paciente crónico”. Cristina Rey Iborra, Mayo 2019, https://repositorio.uam.es/bitstream/handle/10486/687937/rey_iborra_cristinatfg.pdf?sequence=1

[11] 20m. “Bill Gates y Mark Zuckerberg Animan a Los Niños a Programar.” Www.20minutos.Es - Últimas Noticias, 27 Feb. 2013, www.20minutos.es/noticia/1743841/0/gates/zuckerberg/programar/.

[12] Sinc. “El Futuro Está En Enseñar a Programar En La Escuela.” Libertad Digital, 9 May 2015, www.libertaddigital.com/ciencia-tecnologia/tecnologia/2015-05-09/el-futuro-de-la-educacion-esta-en-ensenar-a-programar-en-la-escuela-1276547574/.

[13] “El Concepto De IDE.” Red Hat - We Make Open Source Technologies for the Enterprise, www.redhat.com/es/topics/middleware/what-is-ide.

[14] “Visual Studio Code - Code Editing. Redefined.” RSS, Microsoft, 14 Apr. 2016, code.visualstudio.com/.

[15] "High-Level Language to Specify an Adaptive Heart Failure Follow up Strategy", SABI2020, 22 Congreso de Bioingeniería, Piriápolis, paper 147, Uruguay, 4-6 marzo 2020.

[16] López, Bryan Salazar, et al. “Las Siete Herramientas De La Calidad.” Ingenieria Industrial Online, 6 July 2020, www.ingenieriaindustrialonline.com/gestion-de-calidad/las-siete-herramientas-de-la-calidad/.

[17] Business, ESAN Graduate School of. “El Uso Del Diagrama De Flujo Para La Gestión De Calidad.” ESAN Graduate School of Business, www.esan.edu.pe/apuntes-empresariales/2019/11/el-uso-del-diagrama-de-flujo-para-la-gestion-de-calidad/#:~:text=El diagrama de flujo, también,las actividades en un proceso.&text=El diagrama de flujo se,de la calidad (COQ).

[18] “PYPL PopularitY of Programming Language Index.” Index, pypl.github.io/PYPL.html.

[19] “Why Spring?” Spring, spring.io/why-spring

[20] “Django Overview.” Django, www.djangoproject.com/start/overview/

[21] Atlassian. “Why Git: Atlassian Git Tutorial.” Atlassian, www.atlassian.com/git/tutorials/why-git#:~:text=One of the biggest advantages,every change to your codebase

[22] Martinez, Emilio Rodriguez. React: Cross-Platform Application Development with React Native: Harness the Power of React Native to Build 4 Real-World Apps. Packt, 2018

[23] What Is a Database and What Are the Advantages of Using MySQL Database?, www.tutorialspoint.com/What-is-a-database-and-what-are-the-advantages-of-using-MySQL-database.

[24] “Why Docker?” Docker, www.docker.com/why-docker

[25] “React – A JavaScript Library for Building User Interfaces.” – A JavaScript Library for Building User Interfaces, reactjs.org/

[26] Angular, angular.io/.

[27] Google Data Studio Overview, Google, datastudio.google.com/overview?authuser=1.

[28] OWASP Application Security Verification Standard, owasp.org/www-project-application-security-verification-standard/

[29] Docker Hub, hub.docker.com/r/codercom/code-server/tags?page=1&ordering=last_updated.

[30] Yu, Hongbo, et al. “Evaluate the Security Margins of SHA-512, SHA-256 and DHA-256 against the Boomerang Attack.” Science China Information Sciences, vol. 59, no. 5, 2016, doi:10.1007/s11432-015-5389-4.

[31] IdM, idm.fing.edu.uy/concurso-videos.html

[32] “Firebase Cloud Messaging.” Google, Google, firebase.google.com/docs/cloud-messaging/

[33] “Seguimiento De Enfermedades Crónicas Mediante Receta De Apps Aplicado a La Insuficiencia Cardíaca.” YouTube, YouTube, 27 Oct. 2020, www.youtube.com/watch?v=vDDhT4eC5TY&list=PLHJE-8Uu23MURLNkVA_nH4waX9xd8eCGK&index=26

[34] Franco Simini, Sharon Da Costa, Camilo De los Santos, Valentina Fernández, Marcelo Hernández, Lucia Ribeiro, Isabel Ribeiro. “Formalización de guías clínicas del seguimiento de la Insuficiencia Cardíaca en lenguaje de lógica formal”. Ciclo de Metodología Científica II-2019

[35] Hernán Castillo, Lucia Ribeiro, Gabriela Silvera, Pablo Álvarez-Rocha, Gabriela Ormaechea y Franco Simini. “Transcripción formal de guías clínicas para el Sistema Informático de Manejo de Insuficiencia Cardíaca SIMIC”, Semana Académica 2020

[36] Franco Simini, Matías Galnares, Gabriela Silvera, Pablo Álvarez-Rocha, Richard Low y Gabriela Ormaechea. “Pattern Recognition to Automate Health Followup and as Medical Assistant: Chronic Patients at Home and Out-Patient Diagnostics”, 30 Agosto del 2019

[37] “Creación y Evolución Histórica.” Agencia De Gobierno Electrónico y Sociedad De La Información y Del Conocimiento, www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/institucional/creacion-evolucion-historica.

[38] AGESIC, “Guía de implementación estructura mínima nacional del documento clínico HL7 V3 CDA-R2 para uso en el dominio de Salud,” p. 28, 2013.

[39] P. Salud and H. Cl, “Guía para la Identificación de Persona en el dominio de Salud.”

[40] N. Archivo, “Guía SNOMED CT,” pp. 1–21, 2013.

[41] “Ecardiosurf.” Ideable, 18 Mar. 2021, ideable.net/casos-exito/ecardiosurf/.

[42] “Enfermedades No Transmisibles.” World Health Organization, World Health Organization, www.who.int/es/news-room/fact-sheets/detail/noncommunicable-diseases. 13 Abr. 2021

[43] “Cómo hacer un control y seguimiento del enfermo crónico?”, Residencia de ancianos Palau de Can Sunyer, https://palaudecansunyer.com/como-hacer-control-y-seguimiento-enfermo-cronico/

[44] “Programa para mejorar la atención de las enfermedades crónicas. Aplicación del Modelo de Cuidados para Enfermedades Crónicas”, Revista Elsevier https://www.elsevier.es/es-revista-atencion-primaria-27-articulo-programa-mejorar-atencion-enfermedades-cronicas--13065832

[45] Pablo Anzalone, Wilson Benia, Andrés Coitiño, Alvaro Díaz Berenguer, Darío Fuletti, Daniel Macadar, Daniel Olesker. “Una mirada a la salud de los uruguayos y las uruguayas en el largo plazo”. Uruguay, Febrero 2020.

Capítulo 10

Anexos

10.1 Grupo de profesionales de la UMIC

Figura 44: Grupo de profesionales de la UMIC

10.2 Manual del médico autor

El objetivo de la siguiente guía es que el médico autor adquiera los conocimientos necesarios para poder utilizar de forma correcta SIMIC 2.0.

Requisitos:

1. Tener los conocimientos básicos de programación, en particular del lenguaje Python 3
2. Tener creados los diagramas de flujos aprobados por el/los Médicos Clínicos
3. Tener creado el usuario con el rol médico en SIMIC 1.0, para poder acceder con las mismas credenciales a SIMIC 2.0

Consideración:

Una vez creada una receta y asignada a un paciente para su posterior seguimiento, **NO** se deberá modificar la receta, ya que si no el sistema podría presentar resultados no deseados. En este caso si se quiere modificar una receta existente, se debe crear una nueva con un nombre distinto.

Por ejemplo: si se tenía la receta *peso.py* creada y asignada a un paciente para su seguimiento y se quiere modificar para agregar nuevas mejoras, se debe crear una nueva receta llamada *pesov2.py*

Manual:

El médico autor es el encargado de la traducción del diagrama de flujo a código Python.

¿Cómo acceder al entorno para crear recetas?

Para esto se debe acceder a SIMIC 2.0 e ingresar las credenciales: http://35.174.140.66/login

Luego hacer click en lo seleccionado en rojo:

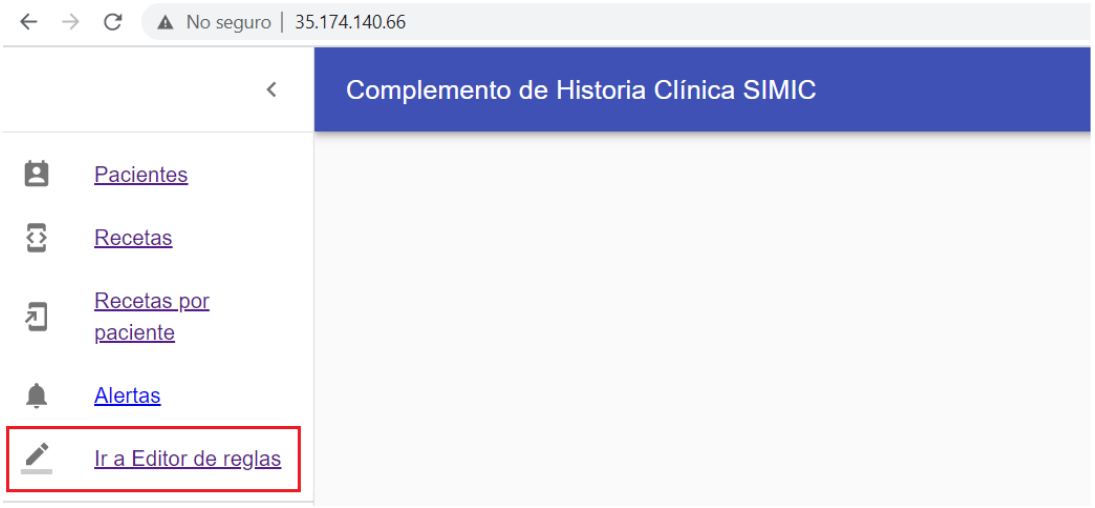


Figura 45: Acceso al editor de recetas

Se nos va a abrir una nueva pestaña con el acceso al editor de recetas y colocar la contraseña simic2 (esto es el ambiente de test).

¿Cómo crear una nueva receta?

Bajo la carpeta RULES se deben crear las nuevas recetas que siempre contendrán al final la extensión **.py** como se muestra en la siguiente imagen:

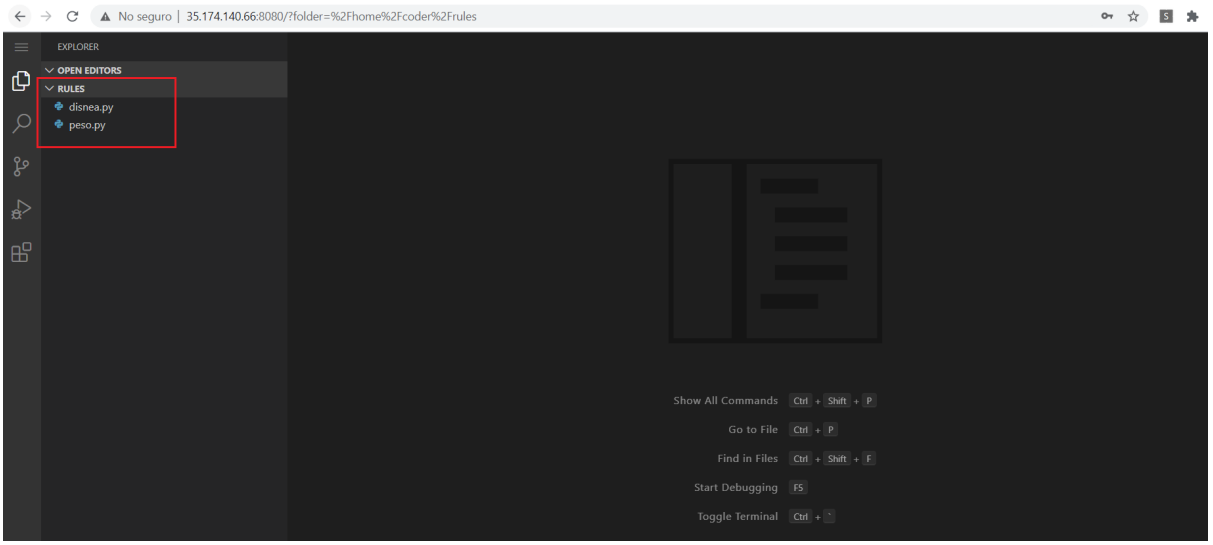


Figura 46: Crear una nueva receta en el ambiente

Para crear una nueva solamente damos click derecho, seleccionamos new file y escribimos el nombre de la receta.

Luego de esto siempre agregar al principio de la receta las siguientes 4 líneas de código:

```
import Interaccion
import Variable
import Alerta
from PersonaSimic import PersonaSimic
```

Figura 47: Módulos a agregar al comienzo de la receta

Estas cuatro líneas nos permite utilizar las funciones que nos provee SIMIC 2.0 para la traducción del diagrama de flujo a código Python.

¿Cuáles son las funciones que nos brinda SIMIC 2.0 para traducir del diagrama de flujo a código Python?

1 - Módulo Interacción

Este módulo provee todas las formas de interactuar con la aplicación móvil SIMIC 2.0 y obtener respuestas del usuario

Interaccion.PreguntaConOpciones(ARGUMENTOS)

Esta función tiene los siguientes argumentos:

```
PreguntaConOpciones(pregunta, opciones, timeOut,
cantidadDeRepreguntas=0, secuencial=0)
```

- pregunta: es la pregunta que se quiere mandar al usuario
- opciones: las opciones para responder la pregunta
- timeOut: la cantidad de tiempo que tiene el usuario para responder a esa pregunta
- cantidadDeRepreguntas: la cantidad de veces que se le va a hacer la pregunta sino la respondió (este argumento es **opcional**, por defecto es 0)
- secuencial: si en la misma receta se hace la misma pregunta y las mismas opciones, con el mismo timeout y la mismas cantidad de repreguntas, entonces hay que setear este argumento con un número distinto del anterior (este argumento es **opcional**, por defecto es 0)

Ejemplo de cómo utilizar esta función:

```
UN_DIA = "P1D"
respuesta = Interaccion.PreguntaConOpciones('¿Ha agregado falta de aire o dificultad para respirar últimamente?',['si','no'], UN_DIA)
```

Interaccion.PreguntaNumero(ARGUMENTOS)

```
PreguntaNumero(pregunta, defaultNumber, timeOut,
cantidadDeRepreguntas=0, secuencial=0)
```

- pregunta: es la pregunta que se quiere mandar al usuario
- defaultNumber: es el número que por defecto le va a aparecer al usuario en la pantalla de la aplicación
- timeOut: la cantidad de tiempo que tiene el usuario para responder a esa pregunta
- cantidadDeRepreguntas: la cantidad de veces que se le va a hacer la pregunta sino la respondió (este argumento es **opcional**, por defecto es 0)
- secuencial: si en la misma receta se hace la misma pregunta y las mismas opciones, con el mismo timeout y la mismas cantidad de repreguntas, entonces hay que setear este argumento con un número distinto del anterior (este argumento es **opcional**, por defecto es 0)

Ejemplo de cómo utilizar esta función:

```
UN_DIA = "P1D"
respuesta = Interaccion.PreguntaNumero('Ingrese su peso actual', 70, UN_DIA)
```

2 - Módulo Alerta

Este módulo sirve para mandar alertas a los médicos clínicos en la web SIMIC 2.0

Interaccion.CrearAlerta(ARGUMENTOS)

Esta función tiene los siguientes argumentos:

```
CrearAlerta(mensaje, gravedad=0)
```

- mensaje: El mensaje que se le quiere mandar al médico clínico sobre la alerta
- gravedad: El valor de la gravedad, siendo 0 menor grave y 2 mayor gravedad (este argumento es **opcional**, por defecto es 0)

Ejemplo de cómo utilizar esta función:

```
Alerta.CrearAlerta('El paciente no mejoró con el agregado del diurético extra', 2)
```

3 - Módulo Variable

Este módulo sirve para tener información que sirve para la ejecución de la receta, como por ejemplo el peso seco del paciente, este peso no se tenía en la información de SIMIC 1.0 y era necesario para la receta peso.

Variable.SetearVariable(ARGUMENTOS)

```
SetearVariable(nombre_de_variable, valor_de_variable)
```

- nombre_de_variable: es el nombre de la variable que se quiere guardar su valor
- valor_de_variable: es el valor de la variable

Ejemplo de cómo utilizar esta función:

```
Variable.SetearVariable('contador',1)
```

Variable.ObtenerVariable(ARGUMENTOS)

```
ObtenerVariable(nombre_de_variable)
```

- nombre_de_variable: es el nombre de la variable que se quiere obtener su valor

Ejemplo de cómo utilizar esta función:

```
valorDelContador = Variable.ObtenerVariable('contador')
```

4 - Modulo PersonaSimic

Este módulo sirve para acceder a toda la información almacenada en la base de datos SIMIC 1.0. Por ejemplo su información personal y la de las consultas.

Retorna el celular del paciente:

```
PersonaSimic().celular()
```

Retorna el primer nombre del paciente:

```
PersonaSimic.primernombre()
```

Retorna el segundo nombre del paciente:

```
PersonaSimic.segundonombre()
```

Retorna el primer apellido del paciente:

```
PersonaSimic.primerapellido()
```

Retorna el segundo apellido del paciente:

```
PersonaSimic.segundoapellido()
```

Retorna el sexo del paciente:

```
PersonaSimic.sexo()
```

Retorna la situación vital del paciente:

```
PersonaSimic.situacionVital()
```

Retorna el teléfono del paciente:

```
PersonaSimic.telefono()
```

Retorna las consultas que tuvo el paciente en SIMIC 1.0:

```
PersonaSimic.consultas()
```

Retorna la última consulta registrada del paciente en SIMIC 1.0:

```
PersonaSimic.ultimaConsulta()
```

10.3 Manual del médico clínico

El objetivo de la siguiente guía es que el médico clínico adquiera los conocimientos necesarios para poder utilizar de forma correcta SIMIC 2.0.

Requisitos:

1. Para que el médico clínico pueda utilizar SIMIC 2.0 es necesario tener instalado el navegador Chrome en la computadora. Se puede descargar del siguiente link: https://www.google.com/chrome/

2. Luego de haber instalado el navegador, es necesario instalar la extensión SIMIC 2.0 para Chrome. Para esto se debe descargar la carpeta del siguiente link: https://drive.google.com/drive/folders/1RjQ_0nbVJ-UpiECMX4kqD1DyV9Gkwv-2?usp=sharing

3. Para instalar la extensión se debe realizar lo siguiente:

    a. Poner en la barra del navegador lo siguiente: chrome://extensions/ y darle enter

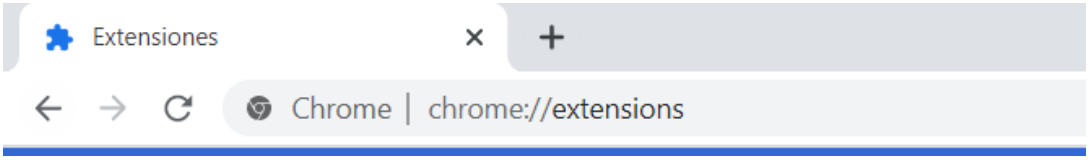


Figura 50: URL de la extensión de Chrome

b. Luego activar el modo desarrollador apretando el siguiente botón:

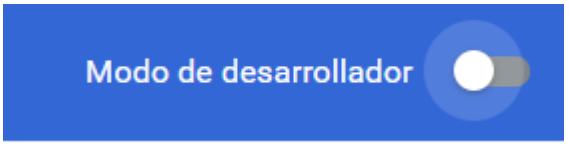


Figura 51: Botón de "Modo desarrollador" en Chrome

c. Por último apretar en el siguiente botón (Cargar extensión sin empaquetar) marcado en rojo:

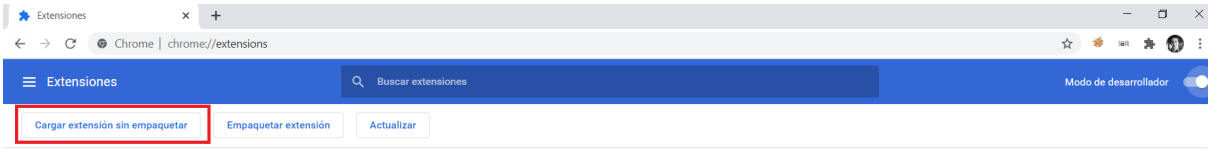


Figura 52: Botón "Cargar extensión sin empaqueta" en Chrome

d. Y seleccionar la carpeta descargada descomprimida:

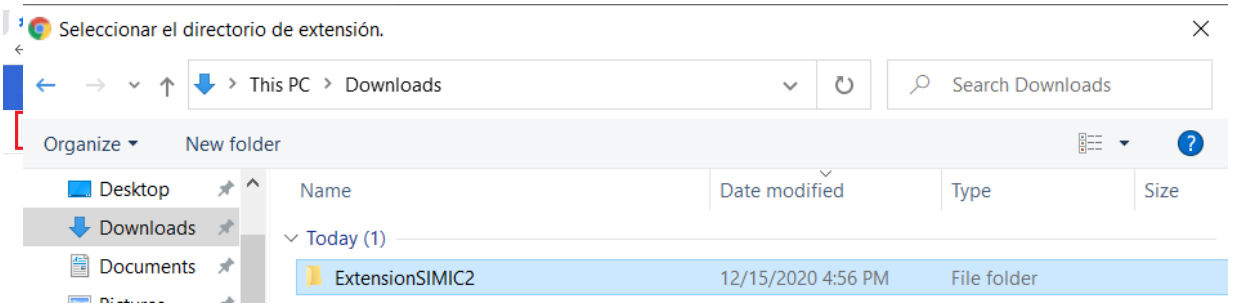


Figura 53: Carga de la carpeta de la extensión en Chrome

e. Se debería ver lo siguiente:

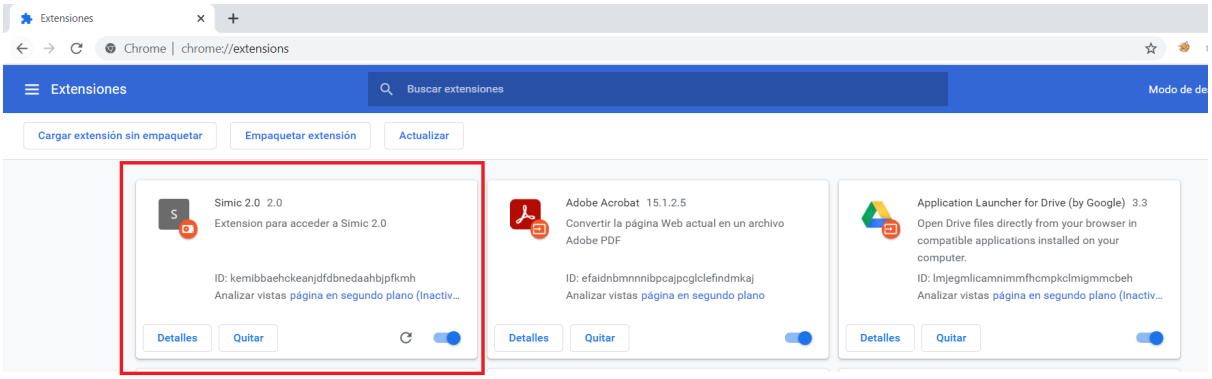


Figura 54: Visualización de la extensión cargada correctamente en Chrome

f. Para que la extensión aparezca en la barra del navegador se debe anclar, para esto seleccionar lo siguiente marcado en rojo:

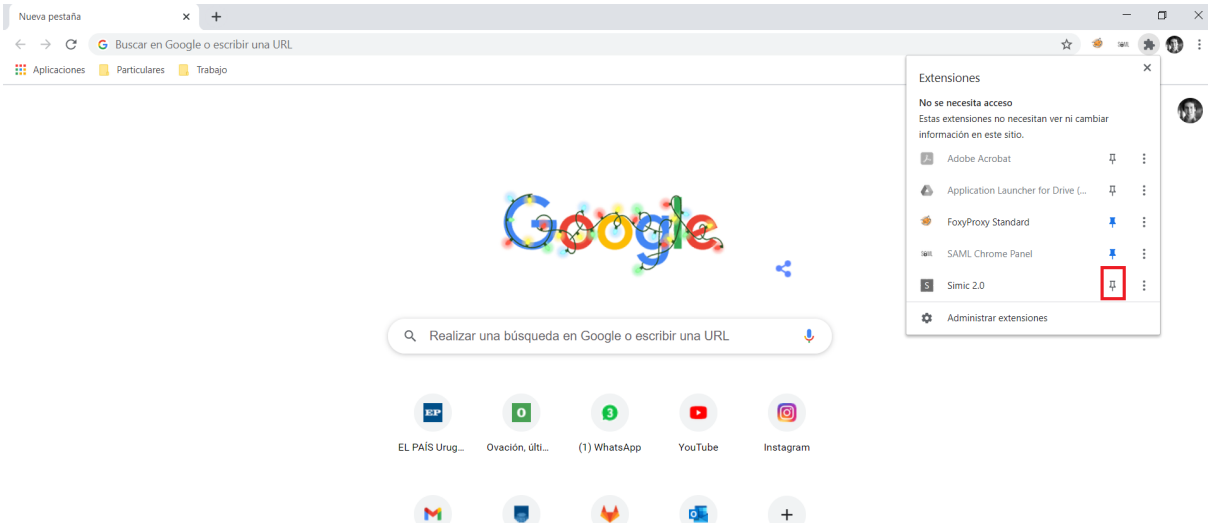


Figura 55: Botón para anclar la extensión a la barra del navegador Chrome

Manual:

El médico clínico debe ser el encargado de asociar recetas a los pacientes, poder realizar el seguimiento y ver las alertas generadas.

Para esto se plantean distintos escenarios:

Escenario 1: El paciente concurre por primera vez a la consulta

En este caso el Médico se tiene que haber logueado en SIMIC 1.0 y realizar los siguientes pasos:

A. Buscar por CI o por nombre y apellido el usuario en la barra del buscador

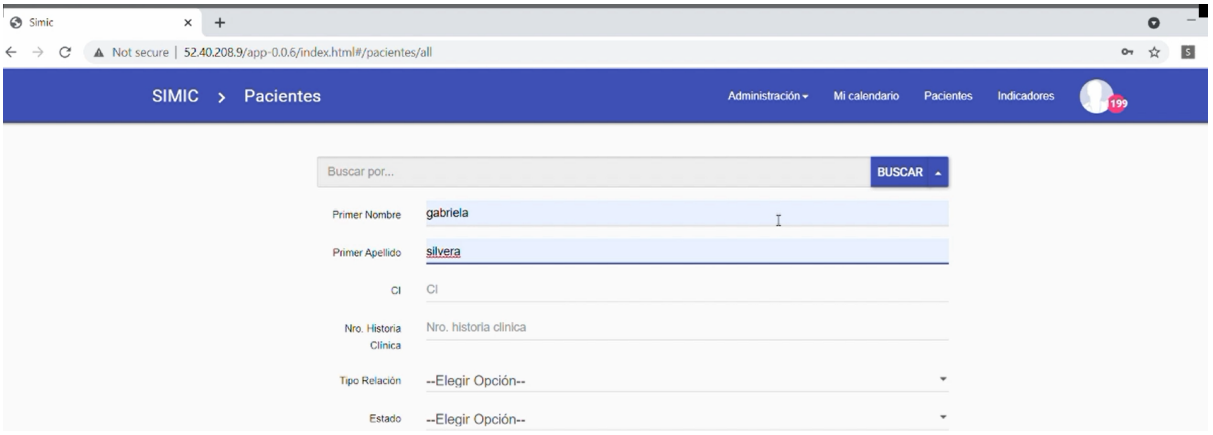


Figura 56: Búsqueda de un usuario en SIMIC 1.0

B. Luego de clickear el botón buscar seleccionar la extensión de Chrome

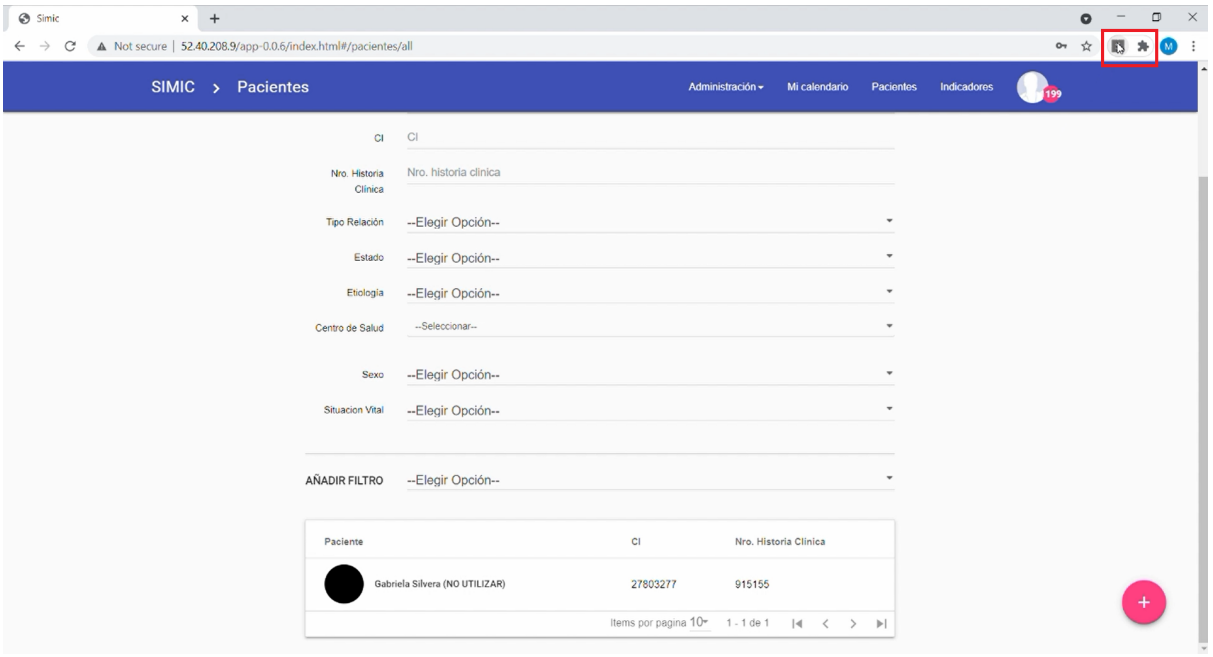


Figura 57: Clickear extensión

C. Este nos abrirá una nueva pestaña con la aplicación web SIMIC 2.0. Se deben autenticar con las mismas credenciales que en SIMIC 1.0 y luego nos aparecerá para registrar al paciente ya que es la primera vez que fue a la consulta:

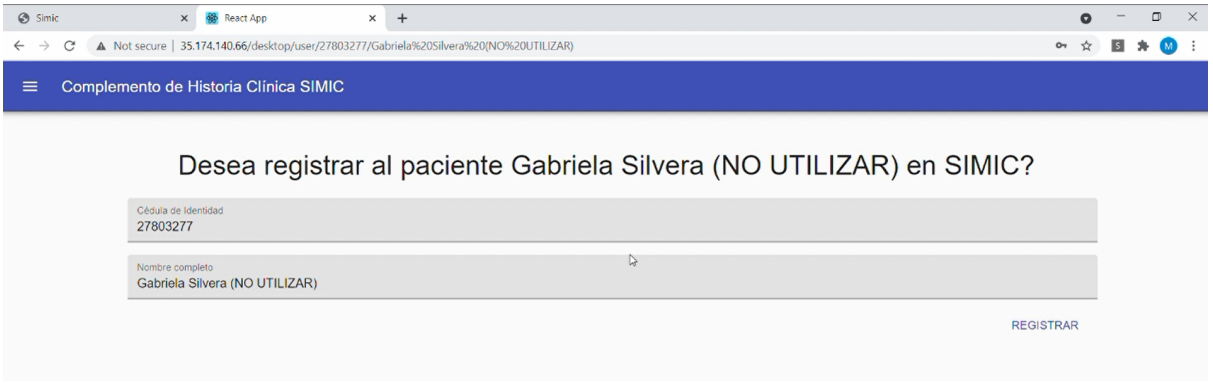


Figura 58: Registro de nuevo paciente en SIMIC 2.0

D. Luego de esto ya podemos asignar las recetas que cargo el Médico Autor en SIMIC 2.0 haciendo click en el siguiente botón:

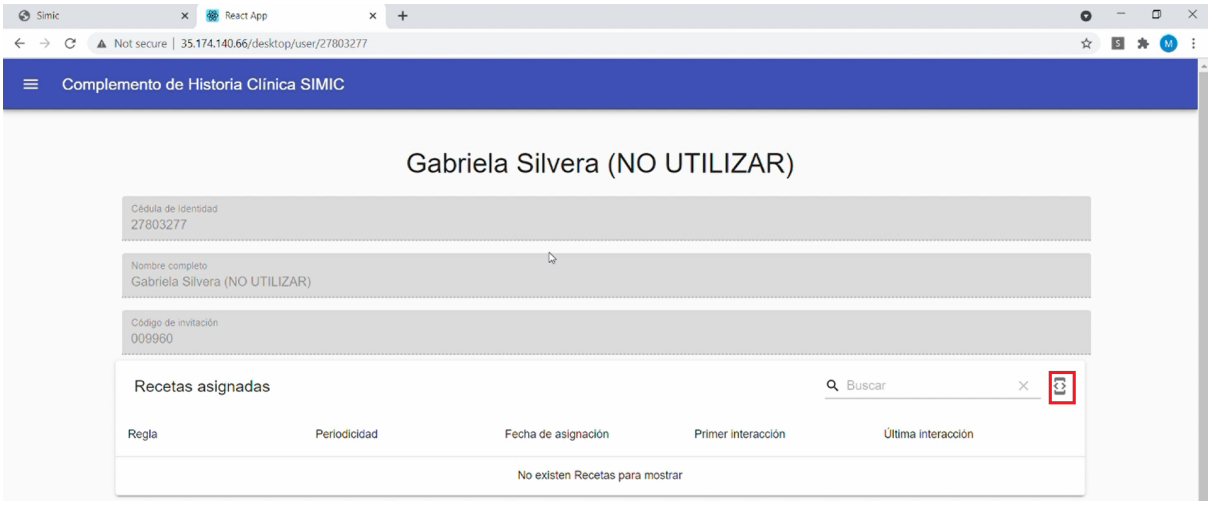


Figura 59: Botón para asignar recetas a un paciente en SIMIC 2.0

E. Para asignar receta se debe poner el nombre de la receta, la periodicidad con la que se quiere que la receta vuelva a comenzar y la fecha de inicio:

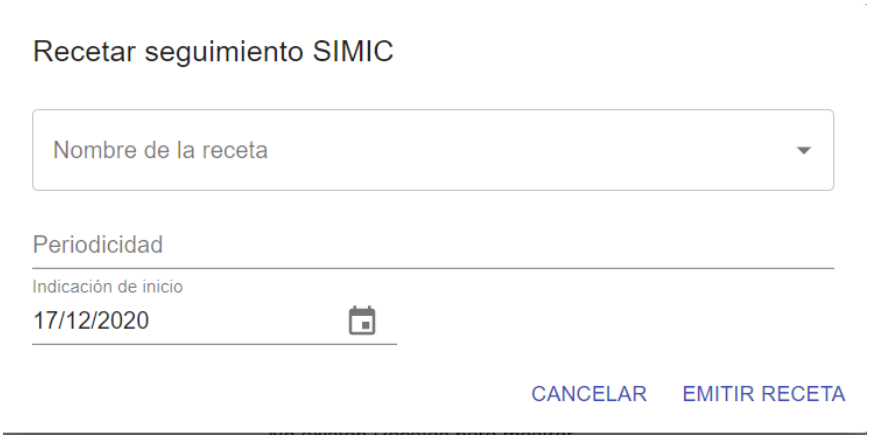


Figura 60: Formulario para ingresar nueva receta

F. Las recetas que necesiten variables para poder comenzar a funcionar también deben ser cargadas con el siguiente botón:

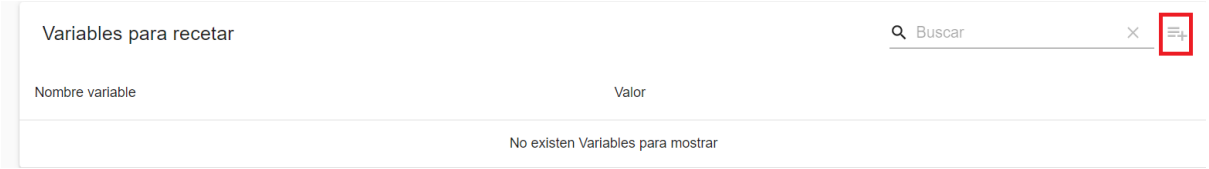


Figura 61: Botón para ingresar variables para las recetas

G. Por último el código de invitación debe ser entregado al paciente para poder utilizar la aplicación:

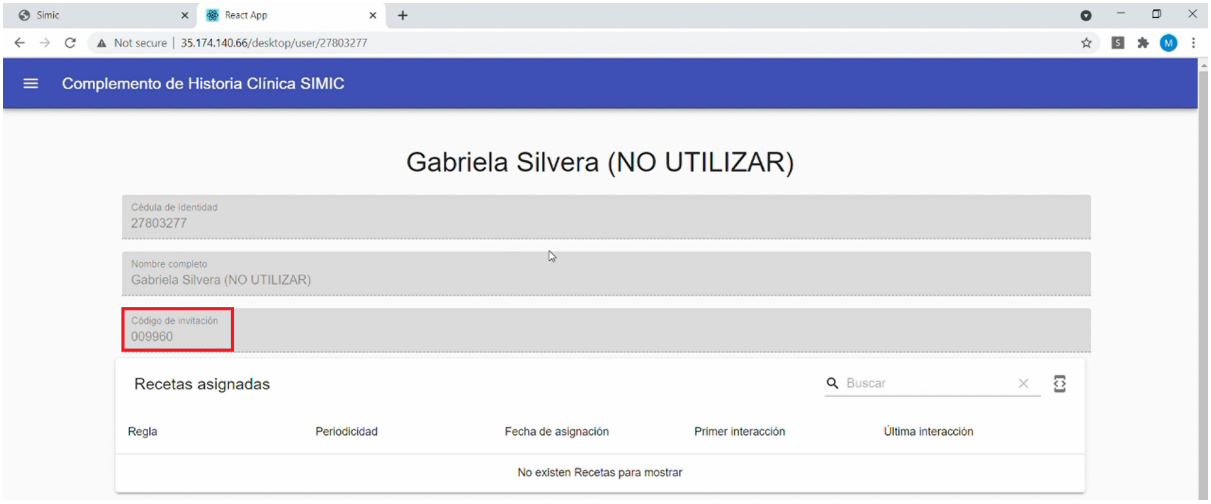


Figura 62: Código de invitación para un paciente

Escenario 2: Siguientes consultas

Para las siguientes consultas se puede acceder tanto a SIMIC 2.0 con la siguiente URL http://35.174.140.66/ o utilizando el mismo procedimiento que en escenario 1, salvo que en este caso no va a requerir el registro del paciente.

Escenario 3: Médico Clínico quiere verificar que el paciente esté utilizando correctamente la app SIMIC 2.0.

Para esto ir a SIMIC 2.0 con la siguiente URL: http://35.174.140.66/
Luego clickar en donde dice Pacientes:

Figura 63: Click en Pacientes en la web de SIMIC 2.0

Por último filtrar por la CI del paciente al que se le quiere ver el seguimiento y clickear sobre la receta.

Ejercicios:

1 - Crear un paciente en SIMIC 2.0 a partir de SIMIC 1.0
2 - Asignar una receta al paciente creado en SIMIC 2.0 (puede ser la receta peso o disnea)
3 - Ingresar a la APP de SIMIC 2.0 utilizando el usuario y el código de invitación
4 - Responder al seguimiento como si fuese un paciente real
5 - Verificar el seguimiento en SIMIC 2.0 WEB

Descargar la aplicación desde el siguiente link: https://drive.google.com/file/d/1ngl-2XOAR7BzyA4zIDbQ_R80WFDro2ej/view?usp=sharing

10.4 Manual de implantación del sistema

Este manual está dirigido a el/los administrador/es de la infraestructura en donde se desea desplegar SIMIC 2.0.

El objetivo del siguiente es proporcionar una guía técnica detallada a la hora de la puesta en producción de SIMIC 2.0, guiando a el/los administradores en la elección del hardware y software mínimos para un correcto funcionamiento de la solución.

Requisitos:

1. **Sistema operativo Linux (arquitectura de 64-bit):** El sistema fue probado en Ubuntu pero es posible instalarlo en cualquier distro basada en debian.
    a. *RAM: 8GB*
    b. *Disco: 500GB*
    c. *CPUs: 4 cores*
2. **Acceso a internet**
3. **Puertos libres y accesibles solamente desde una red interna/vpn:**
    a. puerto 8080 (se desplegará Vs Code para la escritura de reglas)
    b. puerto 80 (se desplegará la web para administrar recetas por parte de los médicos).
4. **Puertos libres y accesibles hacia internet:**
    a. puerto 8085 (se desplegará el componente *notification-server* para hacer posible la comunicación del dispositivo del paciente con SIMIC 2.0).
5. **Acceso a servidores de terceros:**
    a. Debe ser posible la comunicación con el componente que almacena los usuarios de Historia clínica (en la actual implementación de SIMIC 2.0 se necesita comunicación a la base de datos de SIMIC 1.0).
    b. Debe ser posible la comunicación con el componente que almacena la historia clínica de los pacientes (en la actual implementación de SIMIC 2.0 se necesita comunicación a la base de datos de SIMIC 1.0)
6. **GIT**
7. **Docker**
8. **Docker compose**

Manual:

El código fuente de SIMIC 2.0 está empaquetado de manera que sea sencillo el despliegue. Con la ayuda de Docker y Docker-compose se persiguió el objetivo de que la aplicación se comporte de manera agnóstica al servidor en que se despliega.

Lo primero que se debe hacer es descargar el código fuente que se encuentra en el repositorio de código de la FING, con los siguientes comandos:

git clone https://gitlab.fing.edu.uy/simic2.0/docker-devops.git

Luego ejecutaremos el iniciador de la aplicación de la siguiente manera:

cd docker-devops
sh start.sh {IP_DEL_SERVIDOR}

Luego de ingresar las credenciales de acceso al GitLab de la FING, comenzará el despliegue de la aplicación:

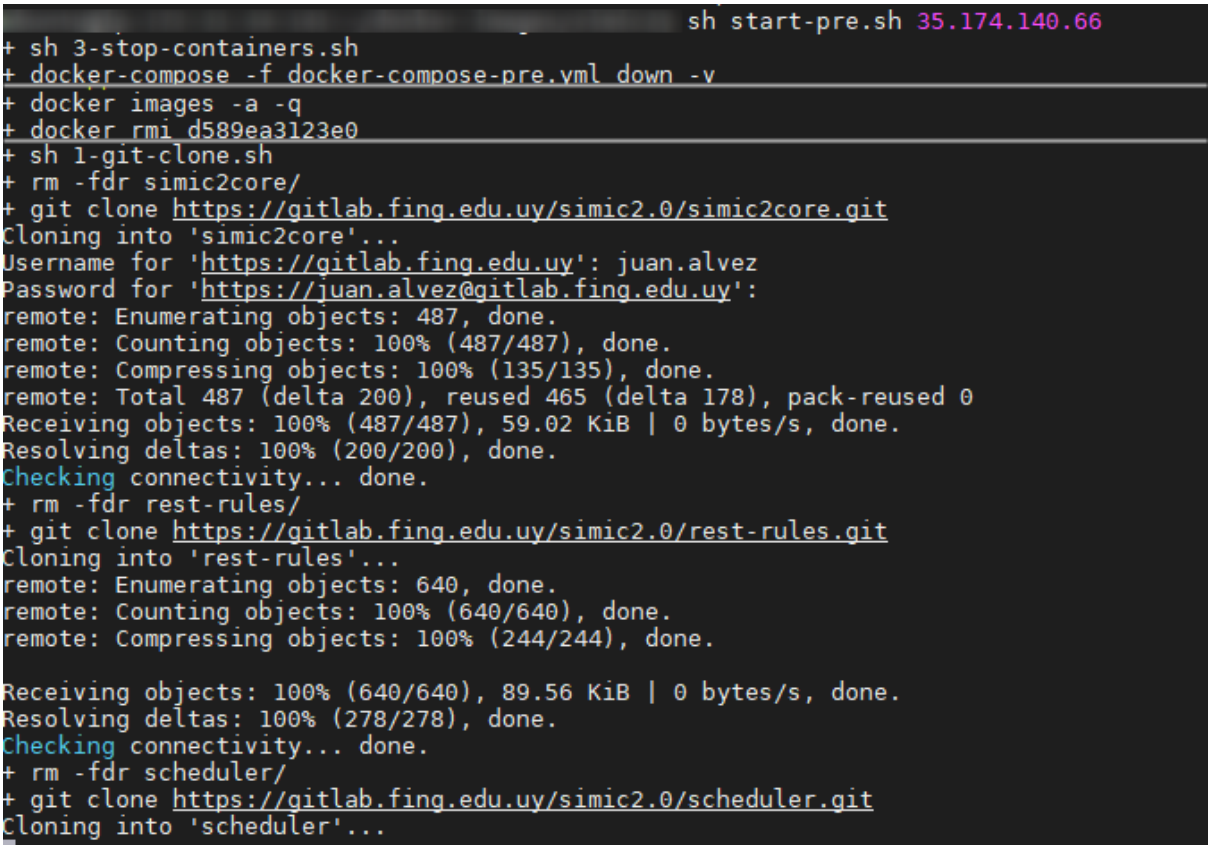


Figura 48: Despliegue de la aplicación

Al finalizar la ejecución deberíamos tener la aplicación lista y funcionando.
Para comprobar el correcto funcionamiento se debe poder ingresar correctamente a través del navegador a la URL http://{*IP_DEL_SERVIDOR*} o su equivalente nombre en caso de utilizar DNS.

10.5 Manual del paciente

A través del WhatsApp de la UMIC se le entregará al paciente el siguiente link para poder descargarse el APK (Android Application Package) e instalarlo en su dispositivo móvil: https://drive.google.com/file/d/1ngl-2XOAR7BzyA4zIDbQ_R80WFDro2ej/view?usp=sharing

Luego de instalada la aplicación en el celular se debe acceder con el código de invitación que el médico clínico le otorgara en la consulta.

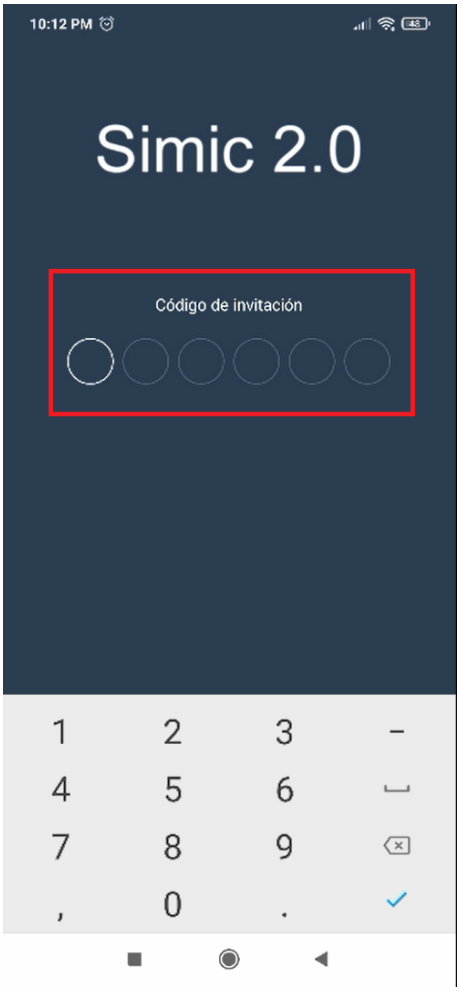


Figura 64: Paciente accede con el código de invitación que se le otorga en la consulta

Luego de acceder con el código de invitación, ya se puede comenzar con el seguimiento a distancia.

10.6 Pruebas de seguridad

ASVS Web

V1: Arquitectura y diseño

#	Descripción	L2	Estado
1.1	Verificar que todos los componentes de la aplicación se encuentran identificados y asegurar que son necesarios.	✓	Chequeado

V2: Requisitos de verificación de autenticación

#	Descripción	L2	Estado
2.1	Verificar que todas las páginas y recursos requieren autenticación excepto aquellos que sean específicamente destinados a ser públicos (Principio de mediación completa).	✓	Chequeado
2.2	Verificar que todos los campos de credenciales no reflejen las contraseñas del usuario. Cargar la credencial por parte de la aplicación implica que la misma fue almacenada de forma reversible o en texto plano, lo que se encuentra explícitamente prohibido.	✓	Chequeado
2.4	Verificar que todos los controles de autenticación se realicen del lado del servidor.	✓	Chequeado
2.6	Verificar que los controles de autenticación fallan de forma segura para evitar que los atacantes no puedan iniciar sesión.	✓	Chequeado

2.7	Verificar que los campos de contraseñas permiten o fomentan el uso de frases como contraseñas (passphrases) y no impiden el uso de gestores de contraseñas, contraseñas largas o altamente complejas.	✓	Chequeado
2.8	Verificar que toda función relacionada con la autenticación (como registro, actualización del perfil, olvido de nombre de usuario, recuperación de la contraseña, token perdido / deshabilitado, funciones de help desk o IVR) que pueda ser utilizada de forma indirecta como mecanismo de autenticación, sea al menos tan resistente a ataques como el mecanismo primario.	✓	Chequeado
2.9	Verificar que la funcionalidad de cambio de contraseña solicite la contraseña anterior, la nueva contraseña y una confirmación de la contraseña.	✓	Chequeado
2.16	Verificar que las credenciales son transportadas mediante un enlace cifrado adecuadamente y que todas las páginas/funciones que requieren que el usuario introduzca credenciales se realicen utilizando enlaces cifrados.	✓	Fuera de alcance por ser ambiente de TEST
2.17	Verificar que las funciones de recuperar contraseña y acceso no revelen la contraseña actual y que la nueva contraseña no se envíe en texto plano al usuario.	✓	Chequeado
2.18	Verificar que no es posible enumerar información mediante las funcionalidades de: inicio de sesión, reinicio o recuperación contraseñas.	✓	Chequeado
2.19	Verificar que no se utilizan contraseñas por defecto en la aplicación o cualquiera de los	✓	Chequeado

	componentes utilizados por la misma (como "admin/password").		
2.20	Verificar que existen mecanismos de anti-automatización que previenen la verificación de credenciales obtenidas de forma masiva, ataques de fuerza bruta y ataques de bloqueos de cuentas.	✓	Chequeado
2.22	Verificar que las funcionalidades de recuperar contraseña y otras formas de recuperar la cuenta utilizan mecanismos de TOTP (Time-Based One-Time Password) u otro tipo de soft token, push a dispositivo móvil u otro tipo de mecanismo de recuperación offline. El uso de un valor aleatorio en un correo electrónico o SMS debe ser la última opción ya que son conocidas sus debilidades.	✓	Chequeado
2.24	Verificar que si la aplicación hace uso de conocimiento basado en preguntas (también conocido como "secreto"), las preguntas no violan leyes de privacidad y son lo suficientemente fuertes para proteger la cuenta de recuperaciones maliciosas.	✓	Chequeado
2.27	Verificar que existen medidas para bloquear el uso de contraseñas comúnmente utilizadas y contraseñas débiles.	✓	Chequeado
2.32	Verificar que las interfaces administrativas de la aplicación no sean accesibles a intrusos.	✓	Chequeado
2.33	Autocompletar de navegadores e integración con gestores de contraseñas deben estar permitidos a no ser que se encuentren prohibidos por políticas de riesgos.	✓	Chequeado

V3: Requisitos de verificación de gestión de sesiones

#	Descripción	L2	Estado
3.1	Verificar que no se utiliza un gestor de sesiones personalizado, o que, si el gestor de sesiones es personalizado, éste sea resistente contra los ataques más comunes.	✓	Chequeado
3.2	Verificar que las sesiones se invalidan cuando el usuario cierra la sesión.	✓	Chequeado
3.3	Verificar que las sesiones se invalidan luego de un período determinado de inactividad.	✓	Chequeado
3.5	Verificar que todas las páginas que requieren autenticación poseen acceso fácil y visible a la funcionalidad de cierre de sesión.	✓	Chequeado
3.6	Verificar que el identificador de sesión nunca se revele en URLs, mensajes de error o registros de bitácora. Esto incluye verificar que la aplicación no es compatible con la re-escritura de URL incluyendo el identificador de sesión.	✓	Chequeado
3.7	Verificar que toda autenticación exitosa y re-autenticaciones generen un nuevo identificador de sesión.	✓	Chequeado
3.11	Verificar que los identificadores de sesión son suficientemente largos, aleatorios y únicos para las sesiones activas.	✓	Chequeado
3.12	Verificar que los identificadores de sesión almacenados en cookies poseen su atributo “path” establecido en un valor adecuadamente restrictivo y que además contenga los atributos "Secure" y "HttpOnly"	✓	Fuera de alcance por ser ambiente de TEST

3.16	Verificar que la aplicación limita el número de sesiones concurrentes activas.	✓	Chequeado
3.17	Verificar que una lista de sesiones activas esté disponible en el perfil de cuenta o similar para cada usuario. El usuario debe ser capaz de terminar cualquier sesión activa.	✓	N/A
3.18	Verificar que al usuario se le sugiera la opción de terminar todas las otras sesiones activas después de un proceso de cambio de contraseña exitoso.	✓	N/A

V4: Requisitos de verificación del Control de acceso

#	Descripción	L2	Estado
4.1	Verificar que existe el principio de privilegio mínimo - los usuarios sólo deben ser capaces de acceder a las funciones, archivos de datos, URL, controladores, servicios y otros recursos, para los cuales poseen una autorización específica. Esto implica protección contra suplantación de identidad y elevación de privilegios.	✓	Chequeado
4.4	Verificar que el acceso a registros sensibles esté protegido, tal que sólo objetos autorizados o datos sean accesibles por cada usuario (por ejemplo, proteger contra la posible manipulación hecha por usuarios sobre un parámetro para ver o modificar la cuenta de otro usuario).	✓	Chequeado

4.5	Verificar que la navegación del directorio esté deshabilitada a menos que esto sea deliberadamente deseado. Además, las aplicaciones no deben permitir el descubrimiento o divulgación de metadatos de archivos o directorios, como carpetas que contengan Thumbs.db, DS_Store, o directorios .git o SVN.	✓	Chequeado
4.8	Verificar que los controles de acceso fallen de forma segura.	✓	Chequeado
4.9	Verificar que las mismas reglas de control de acceso implícitas en la capa de presentación son aplicadas en el servidor.	✓	Chequeado
4.13	Verificar que la aplicación o su infraestructura emite tokens anti-CSFR aleatorios existe otro mecanismo de protección de la transacción.	✓	Chequeado
4.16	Verificar que la aplicación aplique correctamente la autorización contextual para no permitir la manipulación de parámetros de la URL.	✓	Chequeado

V5: Requisitos de verificación para Manejo de entrada de datos maliciosos

#	Descripción	L2	Estado
5.1	Verificar que el entorno de ejecución no es susceptible a desbordamientos de búfer, o que los controles de seguridad previenen desbordamientos de búfer.	✓	Chequeado
5.3	Verificar que las fallas de validación de entradas de datos del lado del servidor sean rechazadas y registradas.	✓	Chequeado

5.5	Verificar que se aplican las rutinas de validación de entradas de datos del lado del servidor.	✓	Chequeado
5.10	Verificar que todas las consultas de SQL, HQL, OSQL, NOSQL y procedimientos almacenados, llamadas de procedimientos almacenados están protegidos por la utilización de declaraciones preparadas o parametrización de consultas, y por lo tanto no sean susceptibles a la inyección de SQL	✓	Chequeado
5.11	Verificar que la aplicación no es susceptible a la inyección LDAP, o que los controles de seguridad previenen inyección LDAP.	✓	Chequeado
5.12	Verificar que la aplicación no es susceptible a la inyección de comandos del sistema operativo, o que los controles de seguridad previenen la inyección de comandos del sistema operativo.	✓	Chequeado
5.13	Verificar que la aplicación no es susceptible a la inclusión de archivo remoto (RFI) o inclusión de archivo Local (LFI) cuando el contenido es utilizado como una ruta a un archivo.	✓	Chequeado
5.14	Verificar que la aplicación no es susceptible a ataques comunes de XML, como manipulación de consultas XPath, ataques de entidad externa XML, y ataques de inyección XML.	✓	Chequeado
5.15	Asegurar que todas las variables string utilizadas dentro de HTML u otro lenguaje web interpretado en cliente se encuentra apropiadamente codificada manualmente o se utiliza plantillas que automáticamente codifican contextualmente para asegurar que la aplicación no sea susceptible a ataques DOM Cross-Site Scripting (XSS).	✓	Chequeado

5.22	Verificar que HTML no confiable proveniente de editores WYSIWYG o similares sean debidamente sanitizados con un sanitizador de HTML y se manejen apropiadamente según la validación de entrada y codificación.	✓	Chequeado

V7: Requisitos de verificación para la criptografía en el almacenamiento

#	Descripción	L2	Estado
7.2	Verificar que todos los módulos criptográficos fallen de forma segura, y que los errores sean manejados de tal manera que no permitan ataques Oracle padding.	✓	Chequeado
7.7	Verificar que los algoritmos criptográficos utilizados por la aplicación hayan sido validados contra FIPS 140-2 o un estándar equivalente.	✓	Chequeado

ASVS Móvil

V1: Requisitos de Arquitectura y Diseño

#	Descripción	L2	Estado
1.1	Todos los componentes se encuentran identificados y asegurar que son necesarios.	✓	Chequeado
1.2	Los controles de seguridad nunca se aplican sólo en el cliente, sino que también en los respectivos servidores.	✓	Chequeado
1.3	Se definió una arquitectura de alto nivel para la aplicación y los servicios y se incluyeron controles de seguridad en la misma.	✓	Chequeado

1.4	Se identificó claramente la información considerada sensible en el contexto de la aplicación móvil.	✓	Chequeado
1.5	Todos los componentes de la aplicación están definidos en términos de la lógica de negocio o las funciones de seguridad que proveen.	✓	Chequeado
1.6	La implementación de los controles de seguridad se encuentra centralizada.	✓	Chequeado
1.7	Existe una política explícita para el manejo de las claves criptográficas (si se usan) y se refuerza su ciclo de vida. Idealmente siguiendo un estándar del manejo de claves como el NIST SP 800-57.	✓	Chequeado
1.8	Existe un mecanismo para imponer las actualizaciones de la aplicación móvil.	✓	N/A
1.9	Se realizan tareas de seguridad en todo el ciclo de vida de la aplicación.	✓	N/A

V2: Requerimientos en el Almacenamiento de datos y la Privacidad

#	Descripción	L2	Estado
2.1	Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar la información sensible, como la información personal, credenciales del usuario y claves criptográficas.	✓	Chequeado
2.2	No se debe almacenar información sensible fuera del contenedor de la aplicación o del almacenamiento de credenciales del sistema.	✓	Chequeado
2.3	No se escribe información sensible en los registros de la aplicación.	✓	Chequeado

2.4	No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura.	✓	Chequeado
2.5	Se desactiva el caché del teclado en los campos de texto donde se maneja información sensible.	✓	Chequeado
2.6	No se expone información sensible mediante mecanismos entre procesos (IPC).	✓	Chequeado
2.7	No se expone información sensible como contraseñas y números de tarjetas de crédito a través de la interfaz o capturas de pantalla.	✓	Chequeado
2.8	No se incluye información sensible en los respaldos generados por el sistema operativo.	✓	Chequeado
2.9	La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano.	✓	Chequeado
2.10	La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria es limpiada luego de su uso.	✓	Chequeado
2.11	La aplicación obliga a que exista una política mínima de seguridad en el dispositivo, como que el usuario deba configurar un código de acceso.	✓	Chequeado
2.12	La aplicación educa al usuario acerca de los tipos de información personal que procesa y de las mejores prácticas en seguridad que el usuario debería seguir al utilizar la aplicación.	✓	Chequeado

V3: Requerimientos de Criptografía

#	Descripción	L2	Estado
3.1	La aplicación no depende únicamente de criptografía simétrica con claves escritas en el código.	✓	Chequeado
3.2	La aplicación utiliza implementaciones de criptografía probadas.	✓	Chequeado
3.3	La aplicación utiliza primitivas de seguridad que son apropiadas para el caso particular y su configuración y sus parámetros siguen las mejores prácticas de la industria.	✓	Chequeado
3.4	La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad.	✓	Chequeado
3.5	La aplicación no utiliza una misma clave criptográfica para varios propósitos.	✓	Chequeado
3.6	Los valores aleatorios son generados utilizando un generador de números suficientemente aleatorios.	✓	Chequeado

V4: Requerimientos de Autenticación y Manejo de Sesiones

#	Descripción	L2	Estado
4.1	Si la aplicación provee acceso a un servicio remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto.	✓	Chequeado

4.2	Si se utiliza la gestión de sesión por estado, el servidor remoto usa tokens de acceso aleatorios para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno.	✓	Chequeado
4.3	Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro.	✓	Chequeado
4.4	Cuando el usuario cierra sesión se termina la sesión también en el servidor.	✓	Chequeado
4.5	Existe una política de contraseñas y es aplicada en el servidor.	✓	Chequeado
4.6	El servidor implementa mecanismos, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces.	✓	Chequeado
4.7	Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad.	✓	Chequeado
4.8	La autenticación biométrica, si hay, no está atada a un evento (usando una API que simplemente retorna "true" o "false"). Sino que está basado en el desbloqueo del keychain (iOS) o un keystore (Android).	✓	N/A
4.9	Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente.	✓	N/A
4.10	Para realizar transacciones se requiere una re-autenticación.	✓	N/A

4.11	La aplicación informa al usuario acerca de los accesos a su cuenta. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos.	✓	N/A

V5: Requerimientos de Comunicación a través de la red

#	Descripción	L2	Estado
5.1	La información es enviada cifrada utilizando TLS. El canal seguro es usado consistentemente en la aplicación.	✓	N/A - Ambiente de prueba
5.2	Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite.	✓	N/A - Ambiente de prueba
5.3	La aplicación verifica el certificado X.509 del servidor al establecer el canal seguro y solo se aceptan certificados firmados por una CA válida.	✓	N/A - Ambiente de prueba

V6: Requerimientos de Interacción con la Plataforma

#	Descripción	L2	Estado
6.1	La aplicación requiere la mínima cantidad de permisos.	✓	Chequeado
6.2	Toda entrada del usuario y fuentes externas es validada y si es necesario será sanitizada. Esto incluye información recibida por la UI, y mecanismo IPC como los intents, URLs y fuentes de la red.	✓	Chequeado

6.3	La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.	✓	Chequeado
6.4	La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.	✓	Chequeado
6.5	JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.	✓	Chequeado
6.6	Los WebViews se encuentran configurados para permitir el mínimo de los manejadores (idealmente, solo https). Manejadores peligrosos como file, tel y app-id se encuentran deshabilitados.	✓	Chequeado
6.7	Si objetos nativos son expuestos en WebViews, verificar que solo se cargan JavaScripts contenidos del paquete de la aplicación.	✓	Chequeado
6.8	Serialización de objetos, si se realiza, se implementa utilizando API seguras	✓	Chequeado