Universidad Autónoma de Manizales

Proyecto Final Bot

Caso #5 – Registro Signos Vitales

Entrega #2

Yordan Castelblanco
Juan Alberto Vidal González
Elan Francisco Perea
Héctor Daniel Cardona Londoño

BOT REGISTRO DE SIGNOS VITALES

1. Solicitud del cliente

Se desea desarrollar un bot que permita a los pacientes que sufren de hipertensión realizar el registro frecuente de sus signos vitales y a sus respectivos médicos tratantes, acceder a ellos de manera fácil y rápida.

Para esto, el bot permitirá a los usuarios registrar sus signos en cualquier momento. Los signos vitales deben incluir por lo menos el registro de la presión arterial (sistólica y diastólica), frecuencia cardiaca, peso y la fecha/hora en que fueron tomados. Estos registros podrán ser consultados por el usuario en cualquier momento y en caso de detectarse alguna incongruencia, podrán ser removidos.

Por su parte, los médicos (administradores del bot) podrán listar la información de sus pacientes y acceder a los registros de signos vitales de cada uno de ellos para revisarlos y verificar que se encuentren bajo parámetros normales. En caso de haber alguna anomalía, los médicos estarán en capacidad de agregar un mensaje con observaciones a cada uno de los registros, el cual podrá ser leído por el paciente propietario del registro cuando lo consulte.

2. Sprint

Luego de la última sesión de clase y teniendo claro la forma de abordar el bot, se opta por establecer un (1) sprint para el proyecto.

Esta definición se toma teniendo en cuenta que el equipo de trabajo, son cuatro (4) integrantes y se tiene un tiempo estimado de dos (2) semanas para su entrega.

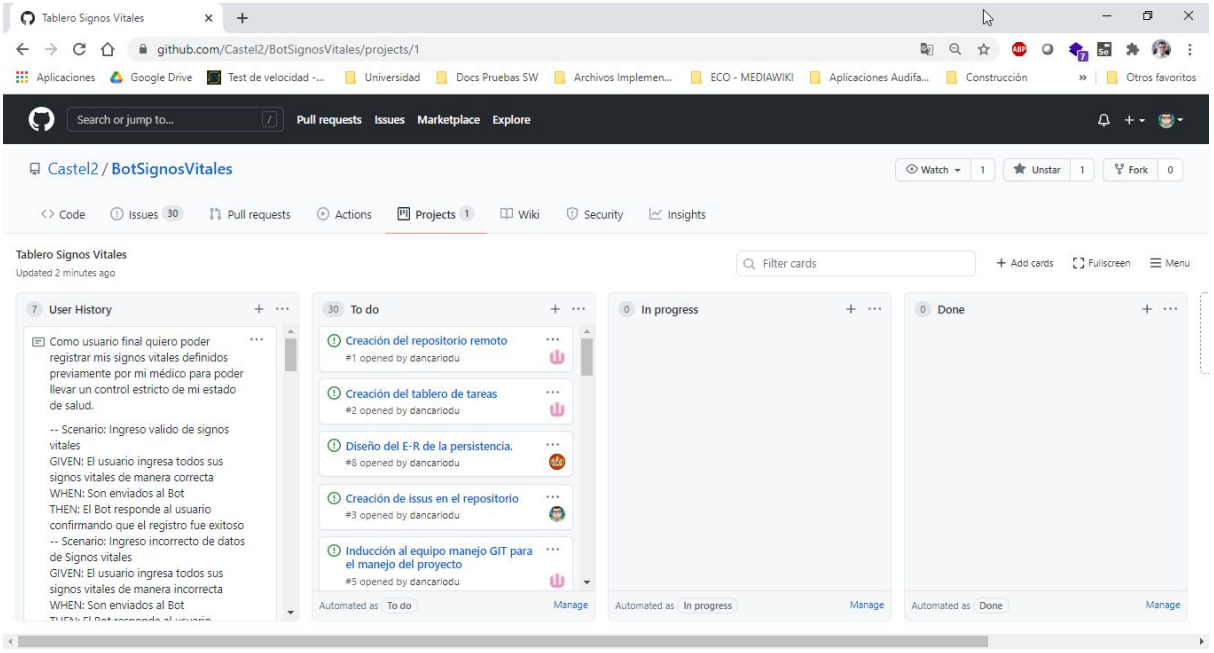
En el Product Backlog se estima un tiempo de 115 horas, distribuidas entre los cuatro(4) integrantes para un total aproximado de 29 horas hombre que con una dedicación diaria de 2.5 horas durante 12 días se cumple con el objetivo de entrega.

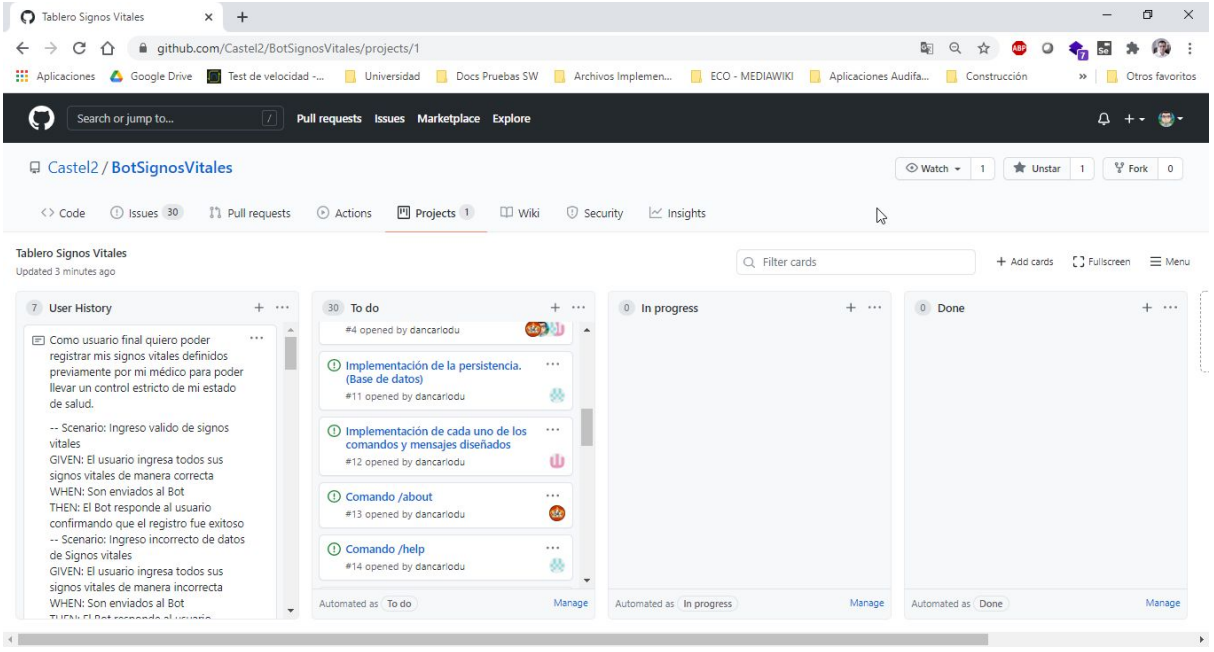
ID HU	Nombre	Prioridad	Estimado (Horas)	Orden
HU 01	Registro de Signos Vitales	Alta	20	1
HU 02	Consultar mis signos vitales	Alta	20	2
HU 03	Eliminar registro de signos vitales	Alta	20	3
HU 04	Roles	Media	10	7
HU 05	Consultar Registros de pacientes	Alta	15	4
HU 06	Registrar Anomalías	Alta	15	5
HU 07	Consultar mis anomalías	Alta	15	6
		Tiempo Estimado:	**115**	

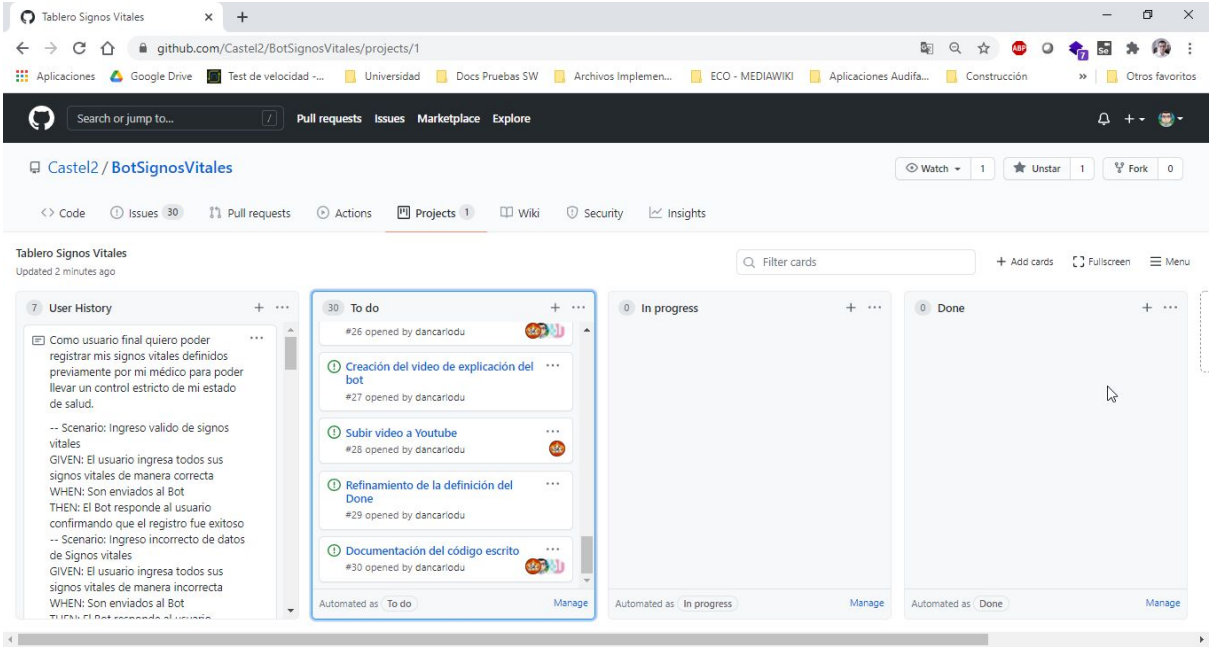
3. Tablero Kanban

Se crea tablero Kanban en el github del proyecto, en el cual se definen 30 tareas iniciales, donde algunas claramente definidas su responsable se asignan y durante el transcurso del proyecto se incorporan otras.

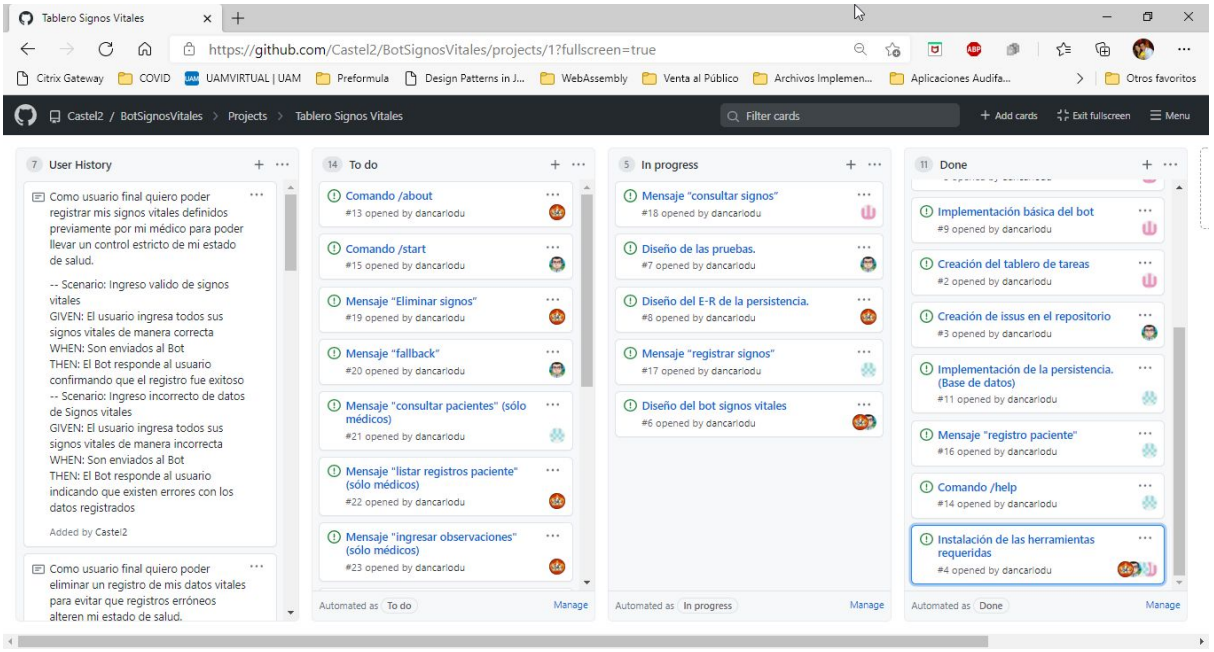
Dia 1 (11-02-2021). Creación del tablero.



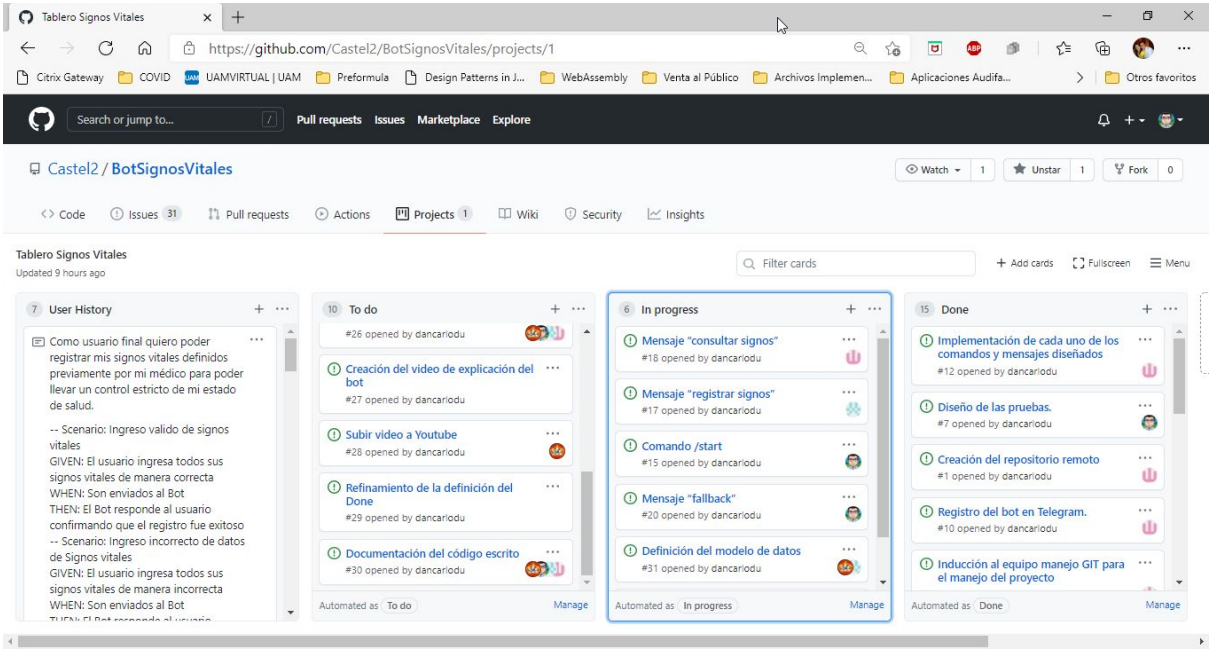



Tablero Signos Vitales
github.com/Castel2/BotSignosVitales/projects/1
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Castel2 / BotSignosVitales
Watch 1
Unstar 1
Fork 0
Code
Issues 30
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Tablero Signos Vitales
Updated 2 minutes ago
Filter cards
Add cards
Fullscreen
Menu
7 User History
Como usuario final quiero poder registrar mis signos vitales definidos previamente por mi médico para poder llevar un control estricto de mi estado de salud.
-- Scenario: Ingreso valido de signos vitales
GIVEN: El usuario ingresa todos sus signos vitales de manera correcta
WHEN: Son enviados al Bot
THEN: El Bot responde al usuario confirmando que el registro fue exitoso
-- Scenario: Ingreso incorrecto de datos de Signos vitales
GIVEN: El usuario ingresa todos sus signos vitales de manera incorrecta
WHEN: Son enviados al Bot
30 To do
#26 opened by dancariodu
Creación del video de explicación del bot
#27 opened by dancariodu
Subir video a Youtube
#28 opened by dancariodu
Refinamiento de la definición del Done
#29 opened by dancariodu
Documentación del código escrito
#30 opened by dancariodu
Automated as To do
Manage
0 In progress
Automated as In progress
Manage
0 Done
Automated as Done
Manage

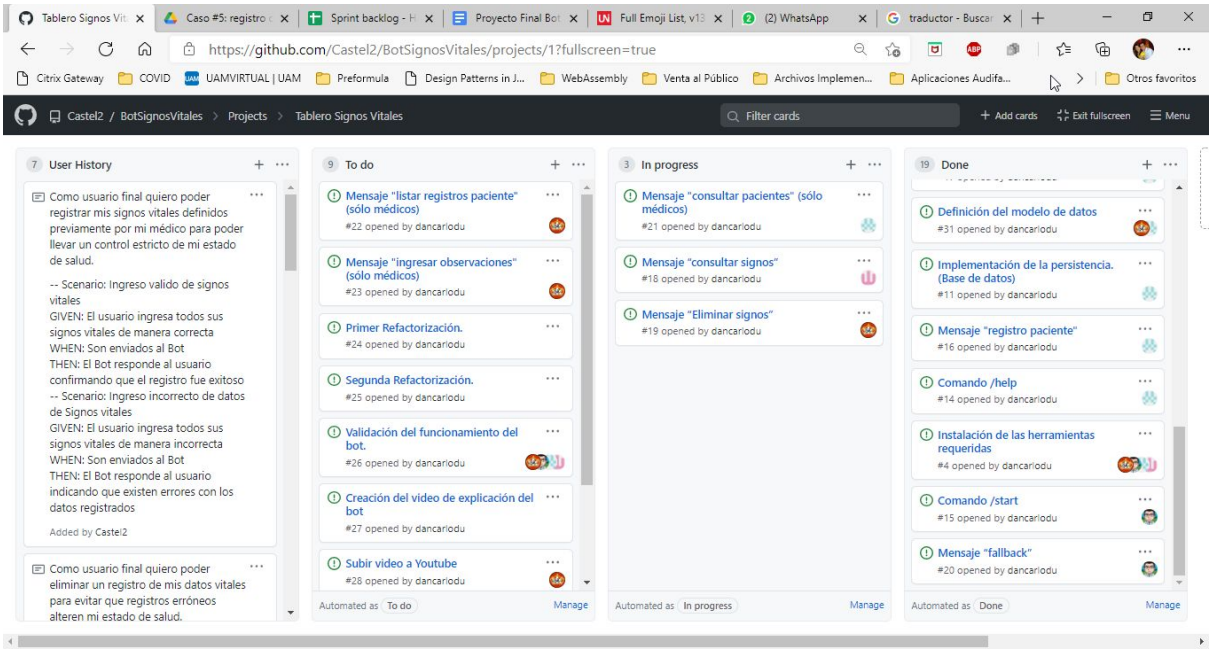
Dia (13-02-2021)



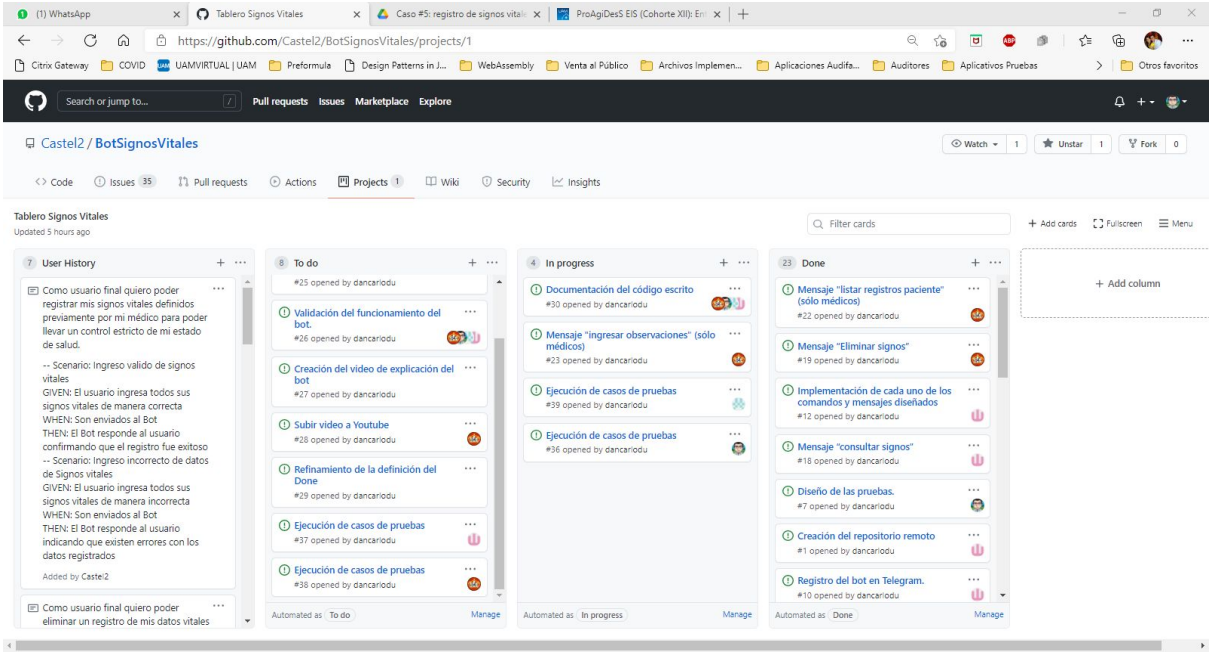
Dia (16-02-2021)



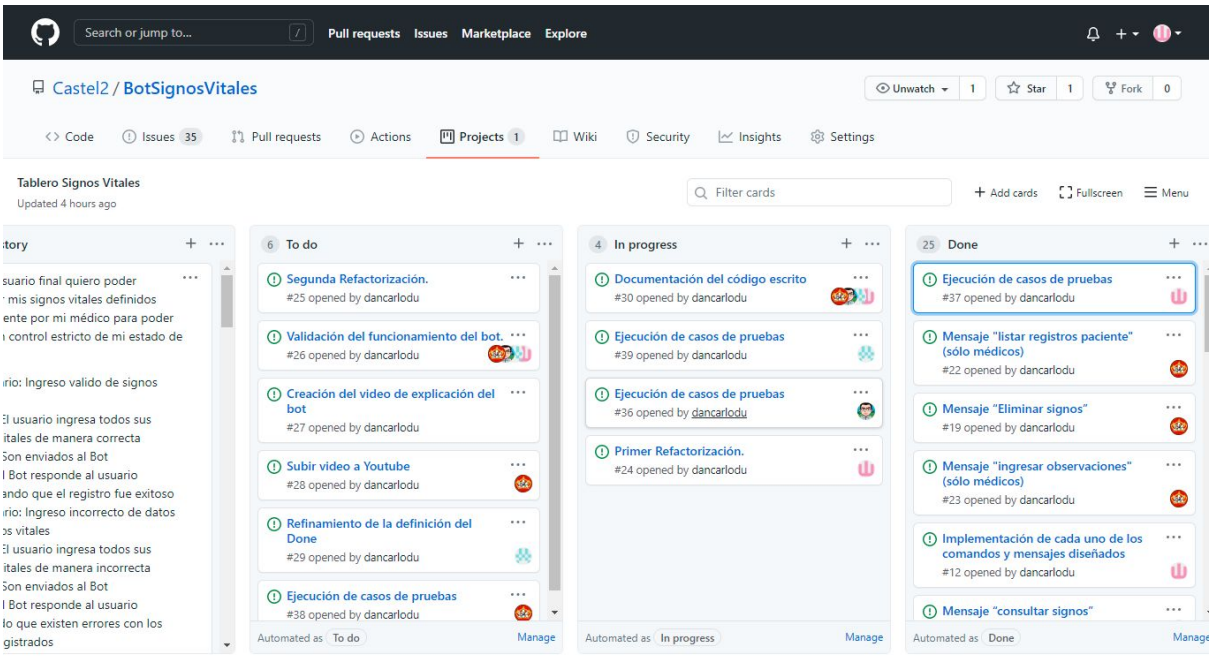
Dia (17-02-2021)



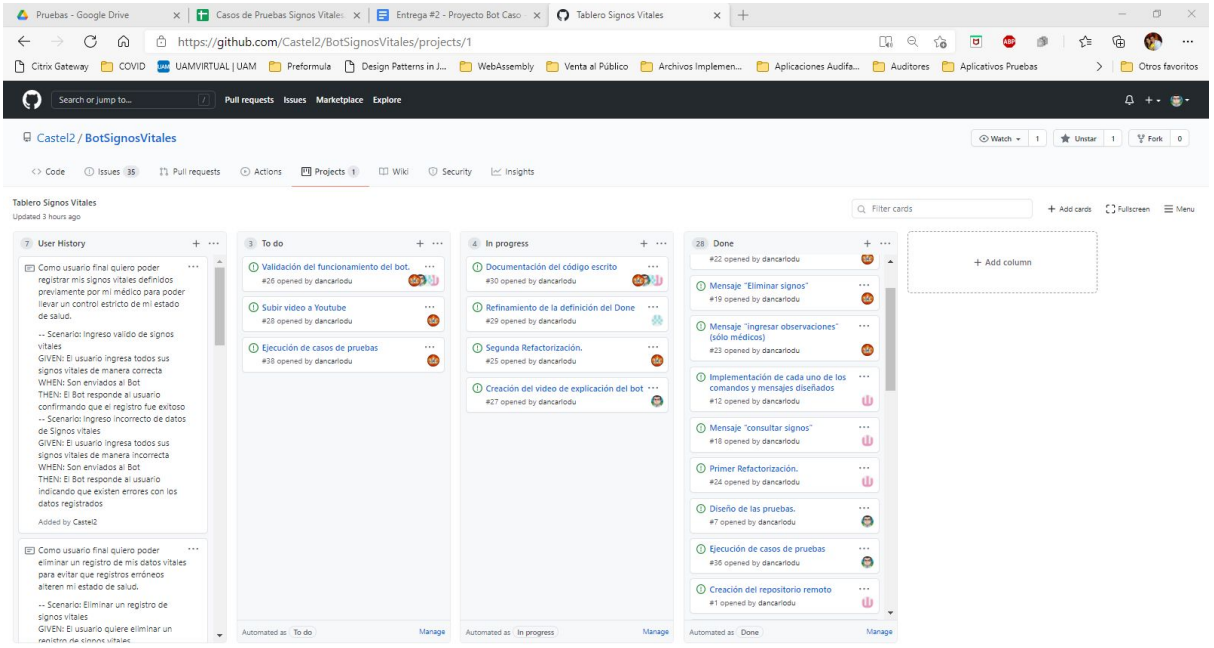
Dia (22-02-2021)



Dia (23-02-2021)



Día (24-02-2021)



4. Refinamiento de la Definición de Done (Definición de Terminado)

Al inicio del proyecto se define una DOD, orientada a cerca de 5 sprints de trabajo, posterior al proceso de planificación, se establece entonces un único sprint de trabajo, bajo el cual se planearon todas las actividades por lo tanto el DOD, original es refinado a una segunda versión, la cual resume el proceso de definition of done a un sprint.
A continuación se presenta la lista de ítems originalmente planteados:

ID	Descripción	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
Planificación						
1	Las actividades especificadas y seleccionadas para cada sprint, están vinculadas a una historia de usuario de forma explícita	No cumple	No cumple	No cumple	No cumple	No cumple
2	Las historias de usuario que se desean cubrir en la ejecución del sprint contienen criterios de aceptación claros	No cumple	No cumple	No cumple	No cumple	No cumple
3	Las actividades planeadas y ejecutadas cuentan con la respectiva priorización	No cumple	No cumple	No cumple	No cumple	No cumple
4	Las actividades son ejecutadas según la priorización definida por el product owner	No cumple	No cumple	No cumple	No cumple	No cumple
Ejecución y desarrollo						
5	Cada uno de los integrantes del equipo de desarrollo seleccionó las actividades según su experticia	No cumple	No cumple	No cumple	No cumple	No cumple
6	Cada uno de los integrantes del equipo definió claramente y socializo con el equipo la duración de las actividades seleccionadas	No cumple	No cumple	No cumple	No cumple	No cumple
7	Se socializo entre el equipo todos aquellos posibles temas técnicos a resolver, se llegó a consensos y acuerdos entre el equipo en las soluciones a implementar	No cumple	No cumple	No cumple	No cumple	No cumple
8	Cada uno de los integrantes del equipo tiene acceso completo a todos los elementos de desarrollo de software necesarios para la ejecución de sus actividades	No cumple	No cumple	No cumple	No cumple	No cumple
Pruebas y calidad						
9	La ejecución de las actividades sigue los estándares de desarrollo de software establecidos por el equipo	No cumple	No cumple	No cumple	No cumple	No cumple
10	Cada una de las actividades cuenta con la definición de pruebas unitarias para el sprint ejecutado.	No cumple	No cumple	No cumple	No cumple	No cumple
11	Se cuenta con un ambiente QA, para la ejecución de pruebas y las actividades desarrolladas fueron desplegadas en este ambiente	No aplica	No aplica	No aplica	No aplica	No aplica
12	Las pruebas unitarias fueron ejecutadas por un miembro del equipo.	No cumple	No cumple	No cumple	No cumple	No cumple
13	Se ajustaron todas las observaciones resultantes de las pruebas unitarias	No cumple	No cumple	No cumple	No cumple	No cumple
14	El desarrollo de todas las actividades se encuentran integradas en el repositorio y el sprint ejecutado cuenta con una rama única que identifique claramente todos los cambios realizados en esta interacción	No cumple	No cumple	No cumple	No cumple	No cumple
15	Los criterios de aceptación planteados se han cumplido a cabalidad	No cumple	No cumple	No cumple	No cumple	No cumple
Despliegue de la solución						

16	Las actividades ejecutadas durante el sprint se encuentran desplegadas en ambientes QA	No aplica	No aplica	No aplica	No aplica	No aplica
17	Las actividades ejecutadas se encuentran desplegadas en producción	No aplica	No aplica	No aplica	No aplica	No aplica
18	El product owner aprueba el despliegue en producción realizado	No aplica	No aplica	No aplica	No aplica	No aplica

En el refinamiento, se concluye que los ítems a evaluar, no se ven ajustados, solo se adiciona uno en el proceso de desarrollo y se reducen los sprints planteados inicialmente.

ID	Descripción	Sprint 1
Planificación		
1	Las actividades especificadas y seleccionadas para cada sprint, están vinculadas a una historia de usuario de forma explícita	Cumple
2	Las historias de usuario que se desean cubrir en la ejecución del sprint contienen criterios de aceptación claros	Cumple
3	Las actividades planeadas y ejecutadas cuentan con la respectiva priorización	Cumple
4	Las actividades son ejecutadas según la priorización definida por el product owner	Cumple
Ejecución y desarrollo		
5	Cada uno de los integrantes del equipo de desarrollo seleccionó las actividades según su experticia	Cumple
6	Cada uno de los integrantes del equipo definió claramente y socializo con el equipo la duración de las actividades seleccionadas	Cumple
7	Se socializo entre el equipo todos aquellos posibles temas técnicos a resolver, se llegó a consensos y acuerdos entre el equipo en las soluciones a implementar	Cumple
8	Cada uno de los integrantes del equipo tiene acceso completo a todos los elementos de desarrollo de software necesarios para la ejecución de sus actividades	Cumple
9	Cada uno de los integrantes gestionar sus aportes de código fuente de forma correcta al repositorio, vinculandolo a actividades asignadas	Cumple
Pruebas y calidad		
9	La ejecución de las actividades sigue los estándares de desarrollo de software establecidos por el equipo	Cumple
10	Cada una de las actividades cuenta con la definición de pruebas unitarias para el sprint ejecutado.	Cumple
11	Se cuenta con un ambiente QA, para la ejecución de pruebas y las actividades desarrolladas fueron desplegadas en este ambiente	No aplica
12	Las pruebas unitarias fueron ejecutadas por un miembro del equipo.	Cumple
13	Se ajustaron todas las observaciones resultantes de las pruebas unitarias	Cumple
14	El desarrollo de todas las actividades se encuentran integradas en el repositorio y el sprint ejecutado cuenta con una rama única que identifique claramente todos los cambios realizados en esta interacción	Cumple
15	Los criterios de aceptación planteados se han cumplido a cabalidad	Cumple
Despliegue de la solución		
16	Las actividades ejecutadas durante el sprint se encuentran desplegadas en ambientes QA	No aplica
17	Las actividades ejecutadas se encuentran desplegadas en producción	No aplica
18	El product owner aprueba el despliegue en producción realizado	No aplica

5. Refinamiento de los comandos y mensajes

Se realiza un refinamiento de los comandos y mensajes a usar en la interacción del Bot respecto a los presentados en la primera entrega:

Descripción	Comando	Datos de Entrada
Saludo Inicial	/start	
Ayuda	/help	
Acerca del este bot	/about	
Registro paciente	registro paciente | rp	{nro_identificacion}
Registrar signos	registrar signos | rs	{presion_arterial_sistólica} {presion_arterial_diastólica} {frecuencia_cardiaca} {peso} {fecha_hora}
Consultar signos	consultar signos | cs	{Fecha inicial} {Fecha Final}
Eliminar signos	eliminar signos | es	{id_registro}
Consultar pacientes (sólo médicos)	consultar paciente | cp	
Listar registros pacientes (sólo médicos)	listas registros pacientes | lrp	{nro_identificacion} {fecha_inicial} {fecha_final}
Ingresar observaciones (sólo médicos)	ingresar observaciones | io	{id_registro} {observaciones}

Teniendo en cuenta los comandos del bot descritos anteriormente, se definen las entradas de cada uno de ellos.

Variable	Tipo	Descripción
nro_identificacion	Entero > 0	Número de identificación del paciente para la consulta de los médicos
Id_registro	Entero (autoincrementable)	Identificador único para cada registro de los pacientes
presión_arterial_sistólica	Entero > 0	Dato de la presión arterial sistólica
presión_arterial_diastólica	Entero > 0	Dato de la presión arterial diastólica
frecuencia_cardiaca	Entero > 0	Dato de la frecuencia cardíaca
peso	Float	Dato del peso
Fecha_hora	Datetime	Fecha y hora de la toma de los signos vitales en formato aaaa-mm-dd hh:mm:ss
observaciones	String (2000)	Texto libre de observaciones
fecha_inicial	Date	Fecha inicial para consulta de los datos en formato aaaa-mm-dd
{fecha_final	Date	Fecha Final para consulta de los datos en formato aaaa-mm-dd

A continuación, se muestran los textos de ayuda, frases con que interactúa los pacientes y médicos en el bot y comandos

Los textos resaltados en rojo, son variables y datos que se deben extraer del sistema.

/start

Hola, soy **Yoda** el asistente virtual de **Vidalan** y quiero ayudarte con el registro y/o consulta de los signos vitales.

¡Recuerda que nuestra comunicación debe ser por texto!

El comando **/start** válida si el usuario se encuentra registrado y el tipo de usuario

PACIENTE NO REGISTRADO EN EL SISTEMA

{**usuario**}, hemos identificado que no estás registrado en el sistema.

Vidalan te brinda la bienvenida y para guiarte en el registro de tus signos vitales, escribe tu número de identificación (sin puntos o comas).

¡Usa el comando: registrar paciente !

PACIENTE REGISTRADO EN EL SISTEMA

{**usuario**}, bienvenido y quiero ayudarte con el registro de tus signos vitales.

¡Por favor escriba el comando para registrar o consultar sus datos!

MÉDICO REGISTRADO EN EL SISTEMA

Dr(a). {**usuario**}, bienvenido y quiero ayudarte con la consulta y registro de observaciones de tus pacientes.

¡Por favor escriba el comando indicado para consultar pacientes o registrar observaciones!

/about

Se presenta un texto estático con la información del Bot y los integrantes del grupo:

Bot Registro Signos Vitales (pyTelegramBot) v0.1

Desarrollado por los estudiantes de la Especialización de Ingeniería de Software de la Universidad Autónoma de Manizales:

Héctor Daniel Cardona <hectord.cardonal@autonoma.edu.co>
Yordan Castelblanco <yordan.castelblancoj@autonoma.edu.co>
Juan Alberto Vidal <juana.vidalg@autonoma.edu.co>
Elan Fco. Perea <elanf.pereaa@autonoma.edu.co>

2021

/help

Estos son los comandos disponibles:

/start - Inicio de la interacción con el bot

/help - Muestra este mensaje de ayuda

/about - Muestra detalles de esta aplicación y su equipo de desarrollo

PACIENTES

registrar paciente|rp {documento} - para registro de paciente

registrar signos|rs {Presión arterial sistólica} {presión arterial diastólica} {frecuencia cardiaca} {peso (kg)} {fecha toma (aaaa-mm-dd)} {hora toma 24h (hh:mm:ss)} - Para registro de signos vitales

consultar signos|cs {Fecha inicial (aaaa-mm-dd)} {Fecha Final (aaaa-mm-dd)} - para consultar sus signos registrados

eliminar signos|es {número de la medición} - eliminar medición, se recomienda consultar la medición para conocer su numeración

MÉDICOS

consultar pacientes|cp - para realizar esta consulta debe estar habilitado como médico en nuestro sistema, permite visualizar el número de documento y el nombre de los pacientes registrados en nuestro sistema

listar registros pacientes|lrp {documento} {Fecha inicial (aaaa-mm-dd)} {Fecha Final (aaaa-mm-dd)} - para realizar esta consulta debe estar habilitado como médico en nuestro sistema, permite consultar datos de pacientes

ingresar observaciones|io {número de la medición} {observación asociada} - permite a asociar una observación a una medición registrada por un paciente, funcionalidad solo disponible para médicos autorizados

Comando registrar signos

Este comando presenta un texto de confirmación donde se le indica al usuario los datos a registrar para su confirmación y cancelación

{**usuario**}, los datos registrados son:

Presión arterial sistólica: **122** mmHg
Presión arterial diastólica: **82** mmHg
Frecuencia cardiaca: **65** latidos por minuto
Peso: **72.5** kg
Fecha y Hora de toma: **20211-02-01 10:40 am**

1.Guardar
2.Cancelar

6. Diseño de Pruebas

Se realizaron y ejecutaron 27 casos de prueba que nos permitieron comprobar que el bot funciona acorde a los requerimientos establecidos.

Para una correcta visualización de las pruebas realizadas, a continuación se relaciona un enlace al documento original en donde se realizaron:

https://drive.google.com/file/d/1WRMtPh2D8q_-TwtQ4nbtGiuR59_AMlVq/view?usp=sharing

7. Repositorio Github

En el siguiente enlace se tiene acceso al repositorio del proyecto llamado “BotSignosVitales”

https://github.com/Castel2/BotSignosVitales

8. Refactorizaciones

a. Replace Nested Conditional with Guard Clauses

Se notó que existían algunos métodos que presentaban algunos condicionales (if, else y elif) anidados, que no permitían entender claramente el flujo del código, por lo que se decidió usar la presente técnica de refactorización, en la cual se realizan todas las validaciones primero y se maneja el flujo del método por medio de return, dejando el código funcional al final, el cual solo se ejecutará si se aprueban las validaciones.

i. Consultar Paciente

```python
@bot.message_handler(regexp=r"^(consultar pacientes|cp)$")
def on_get_paciente(message):
    medico = GestorConsultas.validar_medico(message.from_user.id)
    #Si no existe ningun paciente con ese ese documento
    if not medico:
        return bot.reply_to(message, f"\U0001F6AB Esta consulta solo puede ser realizada por usuarios médicos.",
        parse_mode="Markdown")
    else:
        pacientes = GestorConsultas.get_pacientes()
        if not pacientes:
            return bot.reply_to(message, f"No existen pacientes registrados en la base de datos.",
            parse_mode="Markdown")
        else:
            #si pasa las validadciones de imprime los listados
            text = f"Pacientes registrados en la base de datos:\n\n"
            text += f"|Documento|Nombre Completo\n"
            for m in pacientes:
                text += f"|{m.documento} | {m.nombreCompleto}\n"

            bot.reply_to(message, text, parse_mode="Markdown")
```

Después del cambio

```python
@bot.message_handler(regexp=r"^(consultar pacientes|cp)$")
def on_get_paciente(message):
    medico = GestorConsultas.validar_medico(message.from_user.id)
    #Si no es medico
    if not medico:
        return bot.reply_to(message, f"\U0001F6AB Esta consulta solo puede ser realizada por usuarios médicos.",
        parse_mode="Markdown")

    #Si no existe ningun paciente
    pacientes = GestorConsultas.get_pacientes()
    if not pacientes:
        return bot.reply_to(message, f"No existen pacientes registrados en la base de datos.",
        parse_mode="Markdown")

    #si pasa las validadciones de imprime los listados
    text = f"Pacientes registrados en la base de datos:\n\n"
    text += f"|Documento|Nombre Completo\n"
    for m in pacientes:
        text += f"|{m.documento} | {m.nombreCompleto}\n"

    return bot.reply_to(message, text, parse_mode="Markdown")
```

ii. Registrar paciente

```python
@bot.message_handler(regexp=r"^(registrar paciente|rp) ([0-9]*)$")
def on_set_paciente(message):
    bot.send_chat_action(message.chat.id, 'typing')

    parts = re.match(r"^(registrar paciente|rp) ([0-9]*)$", message.text, flags=re.IGNORECASE)

    documento = int(parts[2])

    usuario = GestorPacientes.get_paciente(documento)
    if usuario == None:
        #Usuario no existente se procede al registro
        GestorPacientes.set_paciente(message.from_user.id, documento, message.chat.first_name + " " + message.chat.
        last_name, 1)
        bot.reply_to(message, f"Paciente registrado.")
    else:
        bot.reply_to(message, f"Paciente ya registrado.")
```

Después del cambio

```python
@bot.message_handler(regexp=r"^(registrar paciente|rp) ([0-9]*)$")
def on_set_paciente(message):
    bot.send_chat_action(message.chat.id, 'typing')

    parts = re.match(r"^(registrar paciente|rp) ([0-9]*)$", message.text, flags=re.IGNORECASE)

    documento = int(parts[2])
    #si existe paciente
    usuario = GestorPacientes.get_paciente(documento)
    if usuario:
        return bot.reply_to(message, f"Paciente ya registrado.")

    #Usuario no existente se procede al registro
    GestorPacientes.set_paciente(message.from_user.id, documento, message.chat.first_name + " " + message.chat.
    last_name, 1)
    return bot.reply_to(message, f"Paciente registrado.")
```

iii. Consultar signos

```
@bot.message_handler(regexp=r"^(consultar signos|cs) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])) ([0-9]{4}-(0[1-9]|1[0-2])-(0
[1-9]|[1-2][0-9]|3[0-1]))$")
def on_get_signos(message):
    bot.send_chat_action(message.chat.id, 'typing')
    id_usuario = int(message.from_user.id)
    # Antes de realizar las operaciones se valida que el usuario este registrado
    if GestorPacientes.existencia_paciente(id_usuario):
        text = message.chat.first_name
        parts = re.match(r"^(consultar signos|cs) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])) ([0-9]{4}-(0[1-9]|1[0-2])-(0
        [1-9]|[1-2][0-9]|3[0-1]))$", message.text, flags=re.IGNORECASE)
        fecha_inicial = parts[2]
        fecha_final = parts[5]
        signos = GestorConsultas.get_signos(message.from_user.id, fecha_inicial, fecha_final)
        # En caso de que el usuario este registrado pero no tenga registros de signos vitales se le mostrara un mensaje
        if signos is None:
            bot.reply_to(message, "No existe registro de signos para el usuario " + text + "\n\n", parse_mode="Markdown")
        else:
            text = "``` Listado de los signos del usuario: " + text + "\n"
            text += f" Desde {fecha_inicial} hasta {fecha_final} \n"
            text += f"|ID|Sistolica|Diastolica|F.Cardiaca|Peso|Observación| \n"
            for sv in signos:
                text += f"| {sv.id}|   {sv.pas}   |   {sv.pad}     | {sv.fc}   | {sv.peso} |{sv.observacion}| \n"
            text += "```"
            bot.reply_to(message, text, parse_mode="Markdown")
    else:
        bot.send_message (
            message.chat.id,
            GestorConversacion.get_validacion_paciente(message.from_user.id,message.from_user.first_name, config.COMPANIA_SIGNOS),
            parse_mode="Markdown")
```

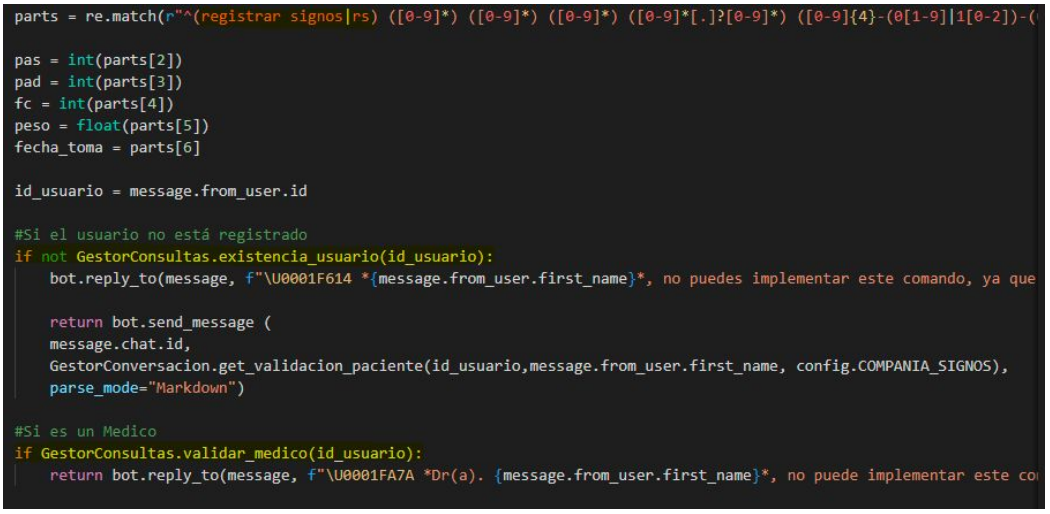
Después del cambio

```
def on_get_signos(message):
    bot.send_chat_action(message.chat.id, 'typing')
    parts = re.match(r"^(consultar signos|cs) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|
    [1-2][0-9]|3[0-1]))$", message.text, flags=re.IGNORECASE)
    fecha_inicial = parts[2]
    fecha_final = parts[5]

    id_usuario = int(message.from_user.id)
    nombre_user = message.chat.first_name
    # si no esta registrado el paciente          id_usuario: int
    if not GestorPacientes.existencia_paciente(id_usuario):
        return bot.send_message (message.chat.id,GestorConversacion.get_validacion_paciente(message.from_user.id,message.from_user.
        first_name, config.COMPANIA_SIGNOS),parse_mode="Markdown")

    signos = GestorConsultas.get_signos(message.from_user.id, fecha_inicial, fecha_final)
    # En caso de que el usuario este registrado pero no tenga registros de signos vitales se le mostrara un mensaje
    if not signos:
        return bot.reply_to(message, "No existe registro de signos para el usuario " + nombre_user + "\n\n", parse_mode="Markdown")

    #si pasa todas las validaciones
    text = "``` Listado de los signos del usuario: " + nombre_user + "\n"
    text += f" Desde {fecha_inicial} hasta {fecha_final} \n"
    text += f"|ID|Sistolica|Diastolica|F.Cardiaca|Peso|Observación| \n"
    for sv in signos:
        text += f"| {sv.id}|   {sv.pas}   |   {sv.pad}     | {sv.fc}   | {sv.peso} |{sv.observacion}| \n"
    text += "```"
    return bot.reply_to(message, text, parse_mode="Markdown")
```

b. Extract Method

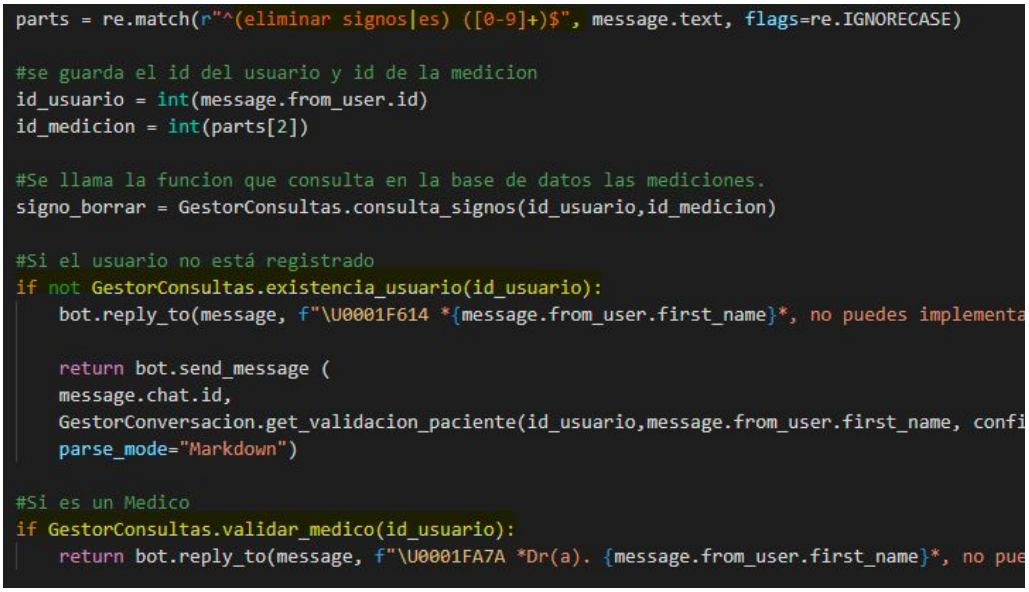
Durante el proceso de codificación detectamos que en varios métodos se realizaban siempre dos validaciones en concreto, determinar si un usuario se encontraba registrado y si este usuario era un médico, como lo vemos a continuación:

i. Registrar signos



```
parts = re.match(r"^(registrar signos|rs) ([0-9]*) ([0-9]*) ([0-9]*) ([0-9]*[.]?[0-9]*) ([0-9]{4}-(0[1-9]|1[0-2])-(

pas = int(parts[2])
pad = int(parts[3])
fc = int(parts[4])
peso = float(parts[5])
fecha_toma = parts[6]

id_usuario = message.from_user.id

#Si el usuario no está registrado
if not GestorConsultas.existencia_usuario(id_usuario):
    bot.reply_to(message, f"\U0001F614 *{message.from_user.first_name}*, no puedes implementar este comando, ya que

    return bot.send_message (
    message.chat.id,
    GestorConversacion.get_validacion_paciente(id_usuario,message.from_user.first_name, config.COMPANIA_SIGNOS),
    parse_mode="Markdown")

#Si es un Medico
if GestorConsultas.validar_medico(id_usuario):
    return bot.reply_to(message, f"\U0001FA7A *Dr(a). {message.from_user.first_name}*, no puede implementar este co
```

ii. Eliminar signos



```
parts = re.match(r"^(eliminar signos|es) ([0-9]+)$", message.text, flags=re.IGNORECASE)

#se guarda el id del usuario y id de la medicion
id_usuario = int(message.from_user.id)
id_medicion = int(parts[2])

#Se llama la funcion que consulta en la base de datos las mediciones.
signo_borrar = GestorConsultas.consulta_signos(id_usuario,id_medicion)

#Si el usuario no está registrado
if not GestorConsultas.existencia_usuario(id_usuario):
    bot.reply_to(message, f"\U0001F614 *{message.from_user.first_name}*, no puedes implementa

    return bot.send_message (
    message.chat.id,
    GestorConversacion.get_validacion_paciente(id_usuario,message.from_user.first_name, confi
    parse_mode="Markdown")

#Si es un Medico
if GestorConsultas.validar_medico(id_usuario):
    return bot.reply_to(message, f"\U0001FA7A *Dr(a). {message.from_user.first_name}*, no pue
```

iii. Consultar Signos

```
parts = re.match(r"^(consultar signos|cs) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]
fecha_inicial = parts[2]
fecha_final = parts[5]

id_usuario = int(message.from_user.id)
nombre_user = message.chat.first_name

#Si el usuario no está registrado
if not GestorConsultas.existencia_usuario(id_usuario):
    bot.reply_to(message, f"\U0001F614 *{message.from_user.first_name}*, no puedes imp

    return bot.send_message (
    message.chat.id,
    GestorConversacion.get_validacion_paciente(id_usuario,message.from_user.first_name
    parse_mode="Markdown")

#Si es un Medico
if GestorConsultas.validar_medico(id_usuario):
    return bot.reply_to(message, f"\U0001FA7A *Dr(a). {message.from_user.first_name}*,
```

Se decidió extraer estas validaciones a un método independiente que funcionara de manera general y que simplemente se llamará en los diferentes métodos que fuera necesario

```
#Funcion recurente
############################################################
def check_user(message, id_usuario):
    #Si el usuario no está registrado
    if not GestorConsultas.existencia_usuario(id_usuario):
        bot.reply_to(message, f"\U0001F614 *{message.from_user.first_name}*,
        bot.send_message (
        message.chat.id,
        GestorConversacion.get_validacion_paciente(id_usuario,message.from_us
        parse_mode="Markdown")
        return False
    #Si es un Medico
    if GestorConsultas.validar_medico(id_usuario):
        bot.reply_to(message, f"\U0001FA7A *Dr(a). {message.from_user.first_n
        return False
    return True
############################################################
```

```
parts = re.match(r"^(consultar signos|cs) ([0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])
fecha_inicial = parts[2]
fecha_final = parts[5]

id_usuario = int(message.from_user.id)
nombre_user = message.chat.first_name

if check_user(message, id_usuario):
    signos = GestorConsultas.get_signos(message.from_user.id, fecha_inicial, fecha_final)
    # En caso de que el usuario este registrado pero no tenga registros de signos vitales se l
    if not signos:
        return bot.reply_to(message, "No existe registro de signos para el usuario " + nombre_

    #si pasa todas las validaciones
    text = "``` Listado de los signos del usuario: " + nombre_user + "\n"
```

9. Retrospectiva

¿Cómo nos fue?

En términos generales para ser nuestro primer trabajo bajo esta forma de trabajo consideramos que nos fue bien y se cumple con el objetivo de finalizar el trabajo, aunque se presentan ciertas novedades en el grupo y la falta de experiencia y sinergia para el trabajo en equipo.

¿Qué salió bien?

La asignación de tareas brinda una orientación clara del trabajo a realizar.

La realización de reuniones diarias mantiene al equipo al tanto de las novedades y gestión de dificultades.

El uso de herramientas de trabajo colaborativo como Google Drive y sistemas de versionamiento como Github, ayudan a tener a la mano la información actualizada y retroalimentar a los compañeros casi que de inmediato.

¿Qué problemas se encontraron y cómo se abordaron?

Se presentaron varias dificultades de las cuales se describen las siguientes:

- Un poco enredados al inicio para establecer los daily scrum donde el primero se vuelve prácticamente un scrum planning en el cual se replantea el proyecto, la situación fue mejorando a medida que íbamos teniendo más claridad y al paso que avanza el proyecto.

- Se estableció reunirnos cada dos(2) días, pero esto nos generó que pasaba mucho tiempo para resolver limitantes que se nos presentaban, se estableció luego de una semana reunión diaria a las 7pm .

- Se generan los issus al inicio del proyecto donde se brindan las indicaciones del trabajo, pero algunos compañeros no revisan el interior de las tarjetas y solo ven el título. Esto genera que lo diseñado originalmente cambie y en otros casos se tenga que dedicar tiempo en ajustar.

- Se presenta un problema con el código en github donde no sabemos qué pasó exactamente y se borró el código de un commit de un compañero, esto nos generó un retraso en el cronograma mientras otro compañero más experimentado en la herramienta lo soluciono.

- Se presenta un problema con el modelo de datos que no estaban bien definidos, donde toca volver a realizar el diagrama Entidad-Relación y modificar el código que se tenía avanzado.

¿Ideas para mejorar?

El tablero de Github permite asignar la tarea a varios miembros del equipo pero genera que nadie termine haciéndola, es mejor generar una tarea para cada miembro del equipo se responsabilice de cumplirla.

Los issues de github manejan etiquetas en las tareas, se pueden personalizar para establecer el grado de prioridad (Alta en color rojo, Media en color Azul y baja en color verde) de la actividad que permita al equipo orientar su trabajo.

10. Video Youtube

https://www.youtube.com/watch?v=yRGf2ueneRo&feature=youtu.be

11. Documentación

BotSignosVitales/resources/documentacion at master·Castel2/BotSignosVitales · GitHub

- **Historias de usuario:**
  https://docs.google.com/document/d/1G2EoF87fjLHq7AzVCOj0vqUZ8Dh8ICKgwIePa01YJzI/edit?usp=sharing

- **Diseño del Bot:**
  https://drive.google.com/file/d/1bp7cLLPgw27MMP2RbHJEB4bMvpmAoBGt/view?usp=sharing

- **Modelo Entidad Relación - Gráfico:**
  https://drive.google.com/file/d/1DEL5BJ7TvsIbtn2YCzPal5Lx-ntejZoV/view?usp=sharing

- **Modelo Entidad Relación - BD:**
  https://drive.google.com/file/d/1eTFko2WEgjfqF3tBSpw0LVfiYdC72xEG/view?usp=sharing

- **Casos de prueba - pruebas realizadas:**
  https://drive.google.com/file/d/1eTFko2WEgjfqF3tBSpw0LVfiYdC72xEG/view?usp=sharing