**Planeación:**

- No me queda clara la diferencia entre “especificación de requerimientos” y “ECU” (que me imagino que es especificación caso de uso).

- Los productos no se hacen por semanas sino por casos de uso. Por ejemplo, no es correcto decir: “diseño de clases primera semana” sino “diseño de clases caso uso registrar pacientes”, o “contrato para método principal del caso de uso tal”.

- Cada tarea debe tener como duración máxima 4 horas. Si dura más de ese tiempo se debe dividir en tareas más pequeñas. Esto ayuda a tener más control.

- Cada tarea debe estar asignada a una sola persona. Las que dicen “Todos” deben dividirlas en tareas concretas para cada una. Esto permite revisar la distribución de las actividades para que sea equitativa.

- La distribución de las actividades para la primera iteración no fue equitativa: Julián tiene casi 20 horas, Jenny 12, Juan Castaño 10 y Juan Esteban 3.5 horas.

- Las tareas tampoco están distribuidas en cuanto a funcionalidad, especialmente para Juan Castaño y Juan Esteban que solo aparecen esporádicamente.

- La planeación se hace para una sola iteración (3 semanas).

- Falta planear la ejecución de las pruebas, no solo las de unidad. No hay que esperar a que se termine todo para hacer pruebas, se pueden hacer por casos de uso.

**Seguimiento:**

- En valor ganado no se coloca “Listo” sino el valor planeado que corresponde a esa actividad.

- Faltan columnas para el seguimiento: valor ganado acumulado, tiempo real invertido y semana real de terminación.

 - No es adecuado avanzar en tareas de otras iteraciones sino se han terminado las tareas de la iteración actual.

**Productos**

**\* Caso de uso “registrar paciente”:**

 - Los dos primeros pasos del flujo básico de eventos no son necesarios. Tampoco el último paso (el 8) porque son acciones antes o después del caso de uso.

 - Faltan validaciones: ¿cómo se sabe si el paciente ya fue registrado o no?

 - Después de ingresar el documento del propietario, se debe mostrar el nombre (para ayudar a verificar visualmente que se ingresé el documento correcto).

**\* Caso de uso “registrar propietario”:**

 - Falta validar si ya está registrado.

**\* Caso de uso “registrar MVZ”:**

 - El flujo alterno de “cancelar” no es necesario.

 - El flujo “no se registró el cambio” no sería un flujo alterno sino una excepción. Si lo van a considerar deben considerarlo para todos los casos de uso.

 - Faltan las validaciones (revisar si el profesional ya está registrado).

**\* Caso de uso “Modificar Información Básica Propietarios”:**

 - Falta indicar cómo se “selecciona el perfil del propietario” (¿hay una consulta primero o consiste solo en ingresar la cédula?)

 - Mencionan DNI, en otro caso mencionaron cédula y en otro documento, deben estandarizar estos términos. DNI es un término usado en España, no en Colombia.

 - El flujo alterno “no hay perfiles disponibles” se contradice con la precondición.

**\* Caso de uso “Modificar Información Básica Pacientes”**

 - Tampoco es clara la parte de seleccionar perfil.

 - Aparece un campo “procedencia” que no estaba en el caso de uso de registrar el paciente.

- Lo mismo para Modificar Información Básica MVZ

 - En los casos de uso de modificar falta validar que los campos no queden en blanco.

**\* Listas de chequeo:**

 - La idea de aplicar estas listas es encontrar errores. Si todas las respuestas dan “si” puede ser un indicador de que no se están aplicando correctamente.

- Además de las listas de chequeo deben validar los casos de uso con los usuarios, para confirmar si los campos de entrada son los que necesitan y las validaciones adicionales que se requieran.

**\* Contratos**

 - No vale la pena hacer contratos para métodos que tengan que ver con la persistencia. Es mejor hacer contratos para la parte de lógica, para los casos de uso más complejos (los tres primeros casos de uso que tienen no son tan complejos).

**\* Casos de prueba:**

 - ¿Han considerado parametrizar especie, raza  y sexo? Esto puede ser útil para realizar análisis estadísticos posteriormente. Esto no es un error pero es para que lo analicen.

 - En los resultados esperados **falta lo que queda en la base de datos.** Esto es muy importante, no se pueden confiar solo en los mensajes.

 - No es adecuado que en un caso de prueba la condición previa sea que exista la cédula 23456789 y que en otro caso de prueba la condición sea que esa misma cédula no exista. Esto no facilita las pruebas.

- Una precondición no es que exista una tabla. Las precondiciones tienen que ver con los datos de las tablas. Tampoco se debe indicar de forma general (que no exista la cédula que se va a ingresar), sino de forma concreta, por ejemplo: que no esté registrada la cédula 34561299.

- En el caso de prueba para “Registrar Propietario”, el listado **no está correcto**. Se hacen combinaciones de clases no válidas.

**\* Diagrama de clases:**

 - Cuando en una clase hay un atributo que hace referencia a otra, el tipo del atributo debe ser un objeto de la otra clase, o una colección de objetos, no un tipo básico. Esto lo tienen en “Paciente” donde tienen “códigoPropietario”, “raza” y “especie” como String, lo cual no es correcto. Esto es modelado de una base de datos, no es diseño orientado a objetos. PILAS: esto lo deben corregir para que revisen cómo pueden lograr esto. Deben “independizarse” de la base de datos y no programar basado solo en tablas.

- Se debe hacer un diagrama por cada caso de uso, no uno general.

- En el diagrama solo hay clases de entidad, faltaron las clases de control. Al elaborar los contratos mencionaron las clases “DAOPropietario” y “Conexión”, pero estas clases no aparecen en el diagrama. Por lo tanto el diagrama no está completo.

**\* Código:**

 - Las clases DAO **no** deben heredar de las clases de entidad. Entre ellas debe existir asociación, no herencia.

 - Falta documentación en los métodos

 - Las consultas no deberían retornar String si no objetos del negocio.

- La clase “Conexión” es una auxiliar para la parte de persistencia, no es precisamente controladora.

 - Faltan clases de control. Como tienen organizado el código, directamente desde la interfaz gráfica piden servicios a la capa de persistencia, lo cual no es correcto. La lógica del negocio está quedando en la misma interfaz. ¿Qué pasó con hacer un buen DISEÑO y no quedarse solo en la funcionalidad?