Universidad del Valle de Guatemala
Facultad de Ingeniería
Programación Orientada a Objetos - 2021
Sección: 10

Fase 4 Proyecto (POO)

Integrantes:
Abner Iván García Alegría - 21285
Sebastián José Solorzano Pérez - 21826
Javier Alejandro Prado Ramírez - 21486
Ángel Gabriel Pérez Figueroa - 21298

Identificación de requisitos Funcionales

- **Funcionalidad del Programa:**
    1. Brindar a los usuarios una herramienta que les facilite el registro de datos de sus pacientes
    2. Permitir a los doctores/enfermer@s brindar información a sus pacientes sobre el medicamento que deben ingerir, cuantos días, cada cuantas horas, etc. Esto con la finalidad de tener un registro guardado de cada paciente.
    3. Registrar la información necesaria de cada paciente, obteniendo cuáles son sus problemas, síntomas, medicamentos en necesidad, ayudas, etc.
    4. Cuando se obtengan los datos del paciente y sus enfermedades o síntomas, a cada uno se le proporcionará una cama en la que podrán descansar y seguir con su tratamiento.
    5. Los doctores/enfermer@s tendrán la facilidad de buscar la información de los pacientes, enfermedades y tratamientos con el número de DPI de cada uno.
    6. Guardado de Información ingresada en el programa en un archivo txt.

- **Programa a utilizar:** NetBeans y Visual Studio Code (Java)

- **Lista de opciones:**
    1. Escoger un Rol:
        a) Doctor
            a) Registro de Pacientes
            b) Mostrar Pacientes
            c) Buscar Pacientes
            d) Mostrar a todos los pacientes
            e) Buscar paciente por DPI
            f) Regresar
        b) Enfermero
            a) Registro de Pacientes
            b) Mostrar Pacientes
            c) Buscar Pacientes
            d) Control Médico
            e) Mostrar a todos los pacientes
            f) Buscar paciente por DPI
            g) Regresar
        c) Salir

Cuando se registra un nuevo paciente, los datos se van almacenando directamente en el archivo.

Identificación y Descripción de clases

- **Clase Display:**

  (**Funcionalidad:** A través de esta clase se llevará a cabo la impresión de todas las respuestas en que el usuario puede interactuar, se implementará un menú que servirá como referencia para conocer la información necesaria de cada paciente)**.**

Propiedades	**Visibilidad**	**Métodos**	**Visibilidad**
Scanner: Scanner	Privada	Directory: String	Pública
Array: Arraylist	Privada	ControlMedico(): void	Pública
StringScan: SS	Privada	addPaciente(): void	Pública
Array: Arraylist	Privada	getPaciente(): void	Pública
		menuRol(): int	Pública
		doctor(): int	Pública
		enfermero(): int	Pública
		asignarCama(): void	Pública
		DPI(): void	Pública
		BuscarPorDPI(): void	Pública

- **Clase Driver:**

  (**Funcionalidad:** Recogerá los valores introducidos en la ejecución del programa. Es el que servirá como punto de partida para el funcionamiento del programa. En otras palabras, controlará el programa direccionando las llamadas a las otras clases (Manipulador del archivo)).

Imports	Métodos
IOException	Main(String[] args)
FileWriter	
BufferedReader	
ClaseFile	

Composiciones: InputMismatchException

- **Clase Medicamento:**

(**Funcionalidad:** Tiene como objetivo principal el control y análisis de los medicamentos que el/la paciente debe ingerir, según sea su caso. La única responsabilidad de esta clase es asignar un control médico al paciente en cuestión)

Propiedades	Visibilidad	Métodos	Visibilidad
nameMedicina: String	Pública	Medicamento()	Pública
Medicación: String	Pública	getNameMedicina(): String	Pública
time: String	Pública	setNameMedicina(): void	Pública
times: String	Pública	getMedicación(): String	Pública
Intervals: String	Pública	setMedicación(): void	Pública
		getTime(): String	Pública

		setTime(): void	Pública
		getTimes(): String	Pública
		setTimes(): void	Pública
		getIntervals(): String	Pública
		setIntervals(): void	Pública
		info(): void	Pública

Agregaciones: Arraylist

- **Clase Paciente**

(**Funcionalidad:** Información completa del paciente, es decir, todos los datos que se anotarán en el registro proporcionado con anticipación, los datos que se tomarán tendrán cohesión con la clase medicación, ya que a través de esta información se podrá realizar el proceso de medicamento a cada paciente.)

Propiedades	**Visibilidad**	**Métodos**	**Visibilidad**
name: String	Privada	Paciente (String[] Data)	Pública
DPI: int	Privada	getNombre(): String	Pública
Sangre: String	Privada	setName(): void	Pública
Diagostico: String	Privada	getDPI(): String	Pública
fecha: String	Privada	setDPI(): String	Pública
		getSangre(): String	Pública

		setSangre(): void	Pública
		getDiagnostico(): String	Pública
		setDiagnostico(): void	Pública
		getFecha(): String	Pública
		setFecha(): void	Pública

- **Clase Camas**

  Funcionalidad (Asignar una cama o paciente y hacer que una cama esté disponible o no**).**

Propiedades	**Visibilidad**	**Métodos**	**Visibilidad**
nombre(): String	privada	getState(camas): void	pública
disponible(): camas	privada	setState(camas): void	pública
camas(): String[]	privada	pacientes: String[]	pública
		Write: String	pública

  La clase Camas recibió la herencia de la clase Pacientes para poder asignarle a cada paciente una cama.

Errores (Exception) esperados:

- **InputMismatch:**

  Cuando el usuario ingrese un tipo de dato erróneo, es decir que se le esté pidiendo un número e ingrese una letra. Si eso sucede, por medio de un catch, se evita que el programa colapse, indicando al usuario que su entrada de datos no es válida y que intente de nuevo. Un claro ejemplo puede ser en la implementación del menú de opciones y cuando se crean instancias con los datos dados por el médico, enfermero(a).

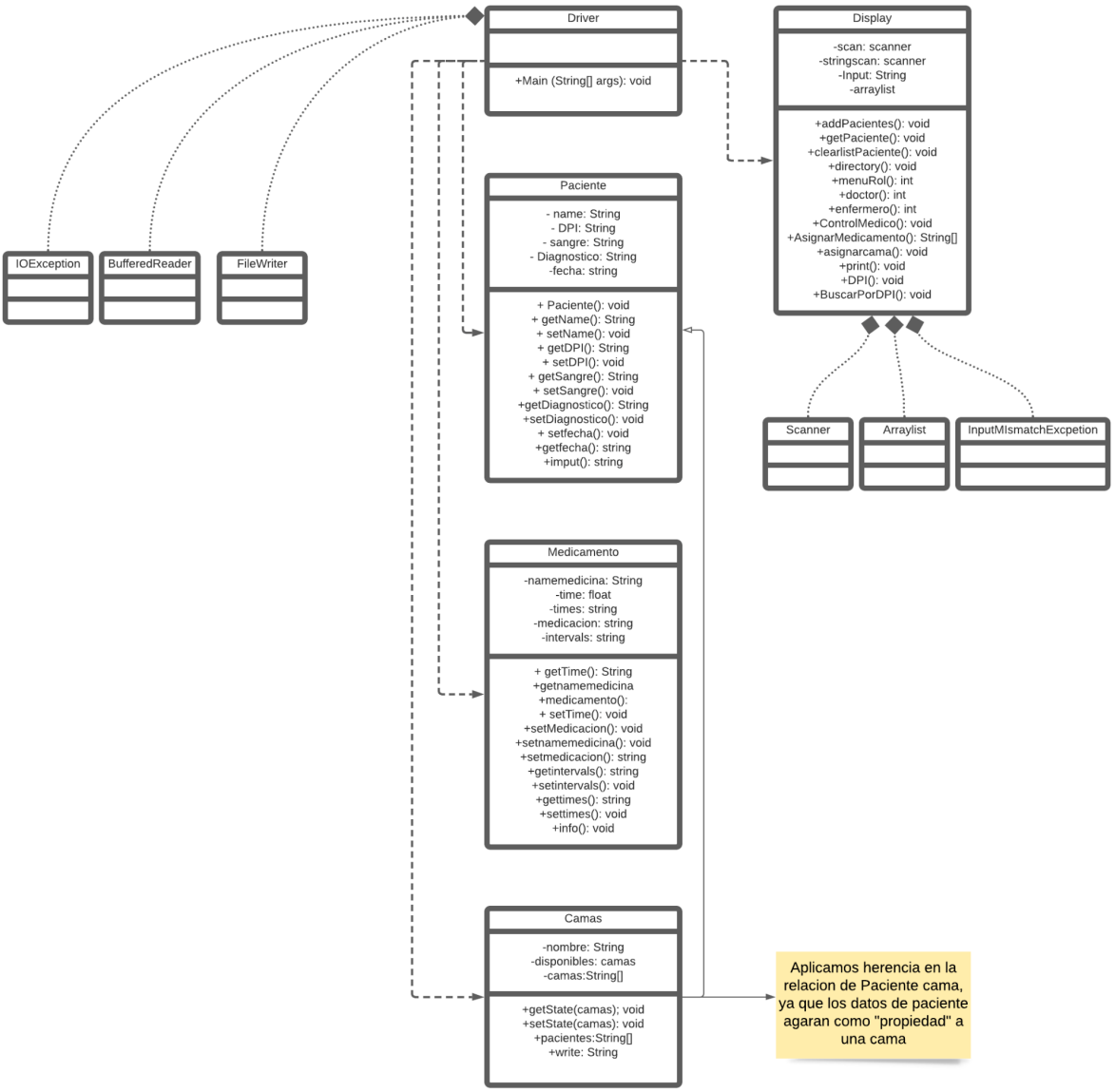
- **IOException:**

  Cuando se interrumpe de manera forzosa el registro de datos. Para esto se le puede avisar al usuario que el archivo que ha sido generado puede contener problemas de ejecución, para evitar esto se le informará al usuario que el archivo está mal, y se le pedirá que sea eliminado del directorio donde se tiene guardado.

- **FileNotFoundException:**

  Al usar el programa por primera vez no existirá un archivo con toda la información, por ende, en caso de no se haya creado, el programa auto generará uno para su implementación. De esta manera al hacer uso del programa, ya se tendrá un archivo con la información requerida.

Diseño del Sistema:



Planificación y gestión

Formulario 1. Gestión del tiempo en el cumplimiento de las tareas planificadas.

Ángel Gabriel Pérez Figueroa **Carné: 21298**

Fecha	Inicio	Fin	Tiempo Int.	Delta Time	Tarea	Comentarios
Martes 16/11	12:15pm	2:00	Ninguno	105 min	Actualización del Documento y UML	En el periodo de clase y una hora después nos enfocamos en la actualización de nuestro archivo y diagrama.
Miércoles 17/11	8:40am	10:15	Ninguno	95 min	Desarrollo del y planteamiento de nuevas ideas para el programa.	Buscamos una manera de ampliar nuestro menú de opciones de una manera que se pueda brindar toda la información necesaria para el programa.
Viernes 19	5:00pm	6 pm	Ninguno	60 min	Nuevos métodos para la clase display	
Sábado	10pm	12 am	30 min	90 min	Terminar el programa con una implementación de información en la clase “medicamento”.	Media hora de descanso y pensar en qué forma podríamos mostrar y hacer la búsqueda de DPI por cada paciente que se ingresaba.

División de requisitos funcionales: Búsqueda de pacientes por DPI y mostrar a todos los pacientes de manera directa.

Formulario 2
Sebastián José Solorzano Pérez **Carné 21826**

Fecha	Inicio	Fin	Tiempo Int.	Delta Time	Tarea	Comentarios
Martes 16/11	12:15pm	2:00	Ninguno	105 min	Actualización del Documento y UML	En el periodo de clase y una hora después nos enfocamos en la actualización de nuestro archivo y diagrama.
Miércoles 17/11	8:40am	10:15	Ninguno	95 min	Desarrollo del y planteamiento de nuevas ideas para el programa.	Buscamos una manera de ampliar nuestro menú de opciones de una manera que se pueda brindar toda la información necesaria para el programa.
Viernes 19	5:00pm	6 pm	Ninguno	60 min	Nuevos métodos para la clase display	
Sábado	10pm	12 am	30 min	90 min	Terminar el programa con una implementación de información en la clase “medicamento”.	Media hora de descanso y pensar en qué forma podríamos mostrar y hacer la búsqueda de DPI por cada paciente que se ingresaba.

División de requisitos funcionales: Búsqueda de pacientes por DPI y mostrar a todos los pacientes de manera directa.

Formulario 3.
Abner Iván García Alegría **Carné: 21285**

Fecha	Inicio	Fin	Tiempo Int.	Delta Time	Tarea	Comentarios
Martes 16/11	12:15pm	2:00	Ninguno	105 min	Actualización del Documento y UML	En el periodo de clase y una hora después nos enfocamos en la actualización de nuestro archivo y diagrama.
Miércoles 17/11	8:40am	10:15	Ninguno	95 min	Desarrollo del y planteamiento de nuevas ideas para el programa.	Buscamos una manera de ampliar nuestro menú de opciones de una manera que se pueda brindar toda la información necesaria para el programa.
Viernes 19	5:00pm	6 pm	Ninguno	60 min	Nuevos métodos para la clase display	
Sábado	10pm	12 am	30 min	90 min	Terminar el programa con una implementación de información en la clase “medicamento”.	Media hora de descanso y pensar en qué forma podríamos mostrar y hacer la búsqueda de DPI por cada paciente que se ingresaba.

División de requisitos funcionales: Búsqueda de pacientes por DPI y mostrar a todos los pacientes de manera directa.

Formulario 4.
Javier Alejandro Prado Ramírez **Carné:21486**

Fecha	Inicio	Fin	Tiempo Int.	Delta Time	Tarea	Comentarios
Martes 16/11	12:15pm	2:00	Ninguno	105 min	Actualización del Documento y UML	En el periodo de clase y una hora después nos enfocamos en la actualización de nuestro archivo y diagrama.
Miércoles 17/11	8:40am	10:15	Ninguno	95 min	Desarrollo del y planteamiento de nuevas ideas para el programa.	Buscamos una manera de ampliar nuestro menú de opciones de una manera que se pueda brindar toda la información necesaria para el programa.
Viernes 19	5:00pm	6 pm	Ninguno	60 min	Nuevos métodos para la clase display	
Sábado	10pm	12 am	30 min	90 min	Terminar el programa con una implementación de información en la clase “medicamento”.	Media hora de descanso y pensar en qué forma podríamos mostrar y hacer la búsqueda de DPI por cada paciente que se ingresaba.

División de requisitos funcionales: Búsqueda de pacientes por DPI y mostrar a todos los pacientes de manera directa.

Implementación

Cabe aclarar que nuestro programa en la entrega anterior ya casi cumplía con todas sus funcionalidades, así que en esta ocasión lo pendiente era mínimo.

Debido a que ya eran pocas cosas pendientes, las tareas que quedaron faltantes las trabajamos en conjunto, haciendo las mismas actividades a los mismos horarios y días

Agregaciones
1. Buscar paciente por su número de DPI
2. Mostrar a todos los pacientes almacenados de manera directa en el programa.
3. En la clase medicamento se agregó un método “info()” que constaba de mostrar toda la información del paciente cuando desee buscar pacientes por DPI.

Nota

Cuando se desee probar la funcionalidad de Buscar Por DPI, es necesario que se registre un paciente primero, para poder observar toda su información. En otras palabras si se cierra el programa y se vuelve a correr hay que registrar un nuevo paciente para observar esta nueva funcionalidad.

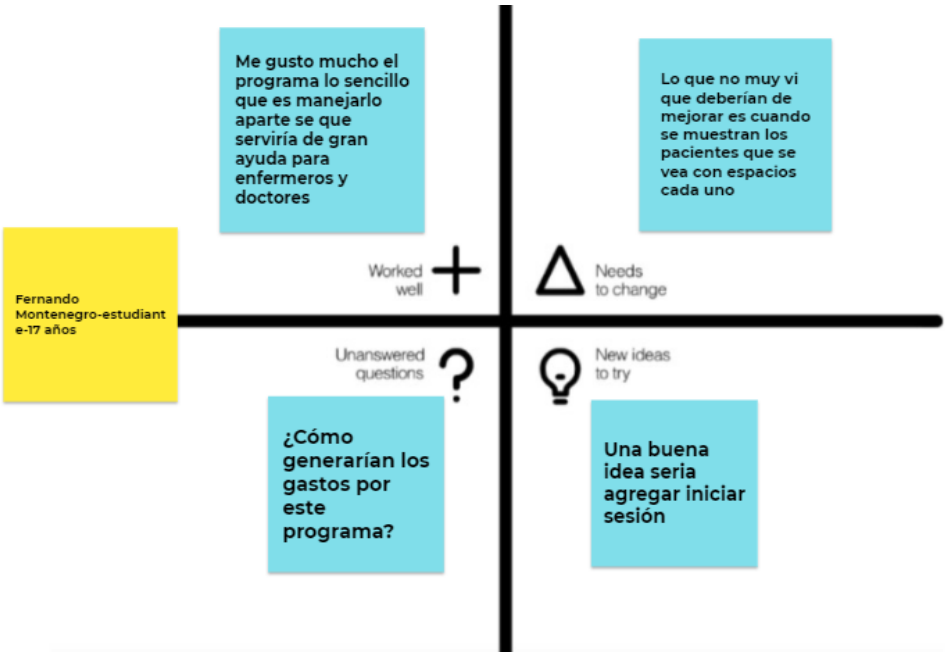
https://youtu.be/Py2lvqjZdpI == Video de muestra

Toma de retroalimentación en base al prototipo fase 4

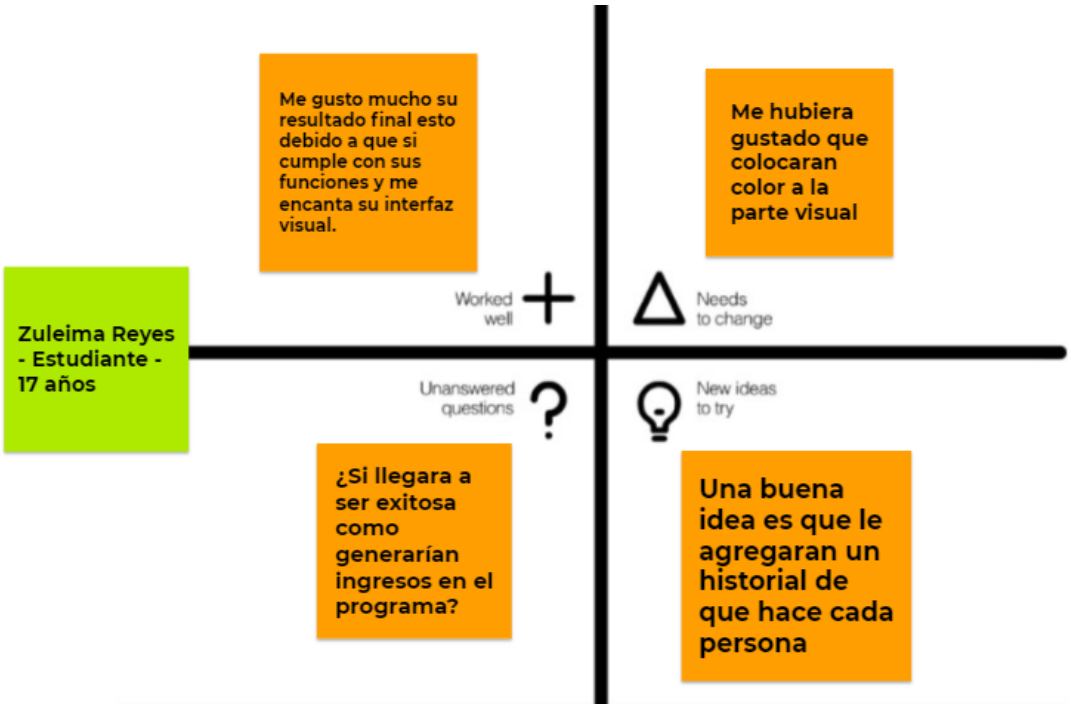
Método de Organización

Debido a que se han mostrado nuevos cambios en el programa, se buscó la forma de que las personas a través de un video, pudieran ver nuestro programa funcionando de manera que pueda interactuar, quizá no de una manera directa, pero de manera que se pueda entender.

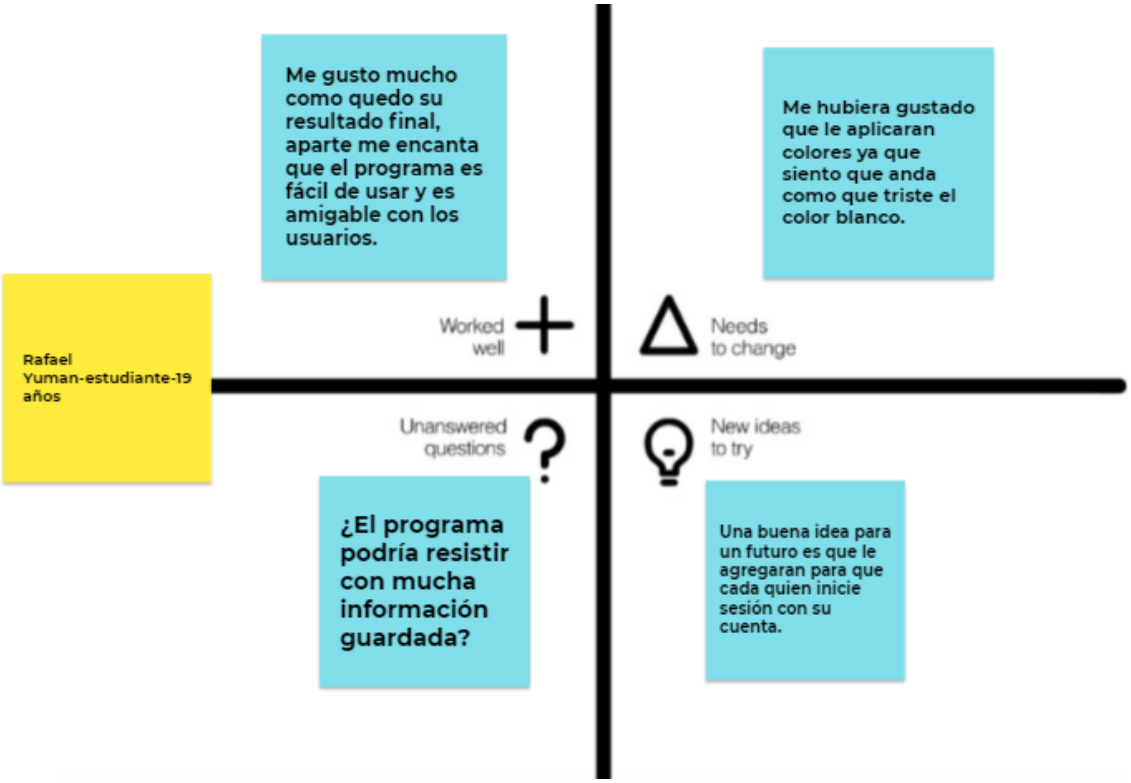
Retroalimentación #1



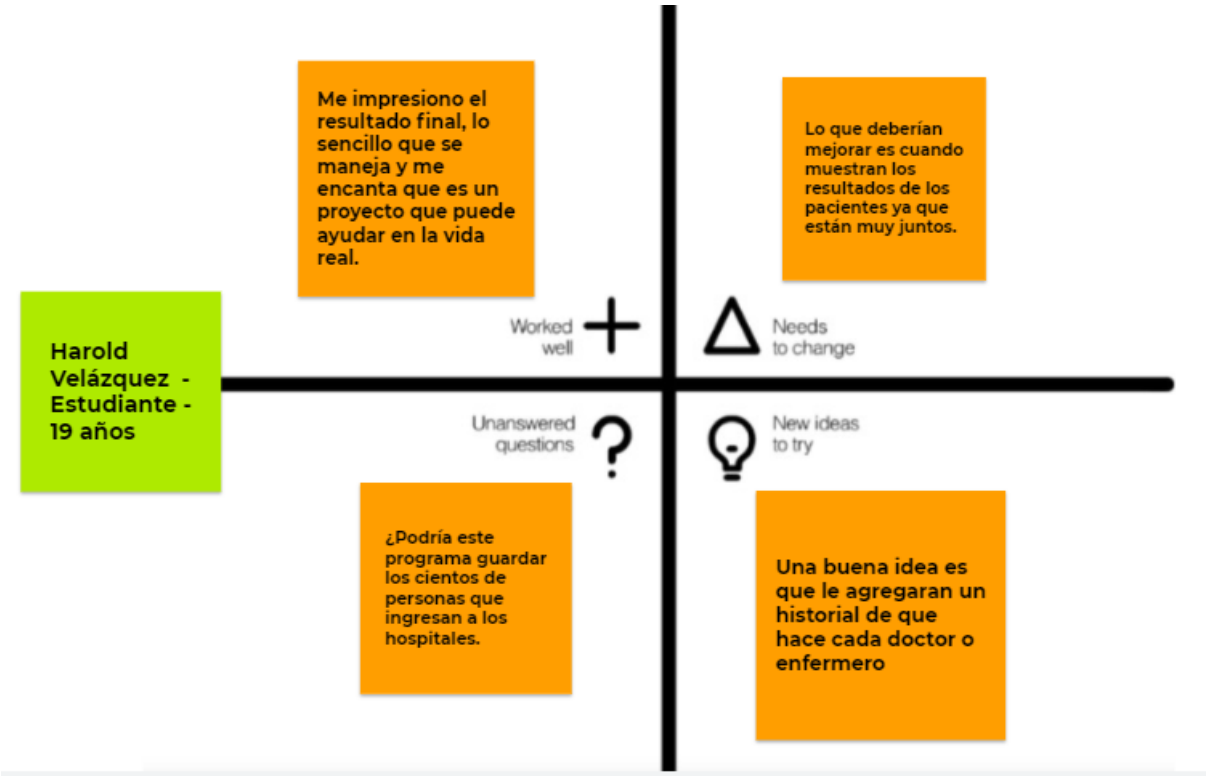
Retroalimentación #2



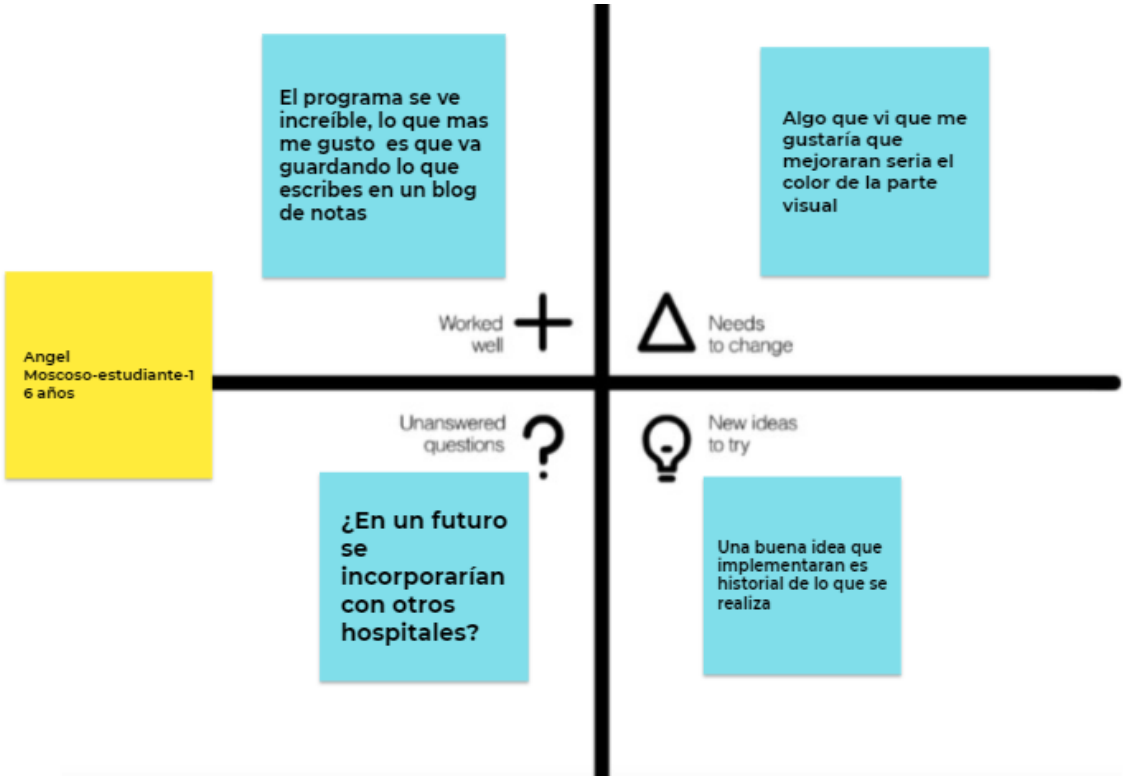
Retroalimentación #3



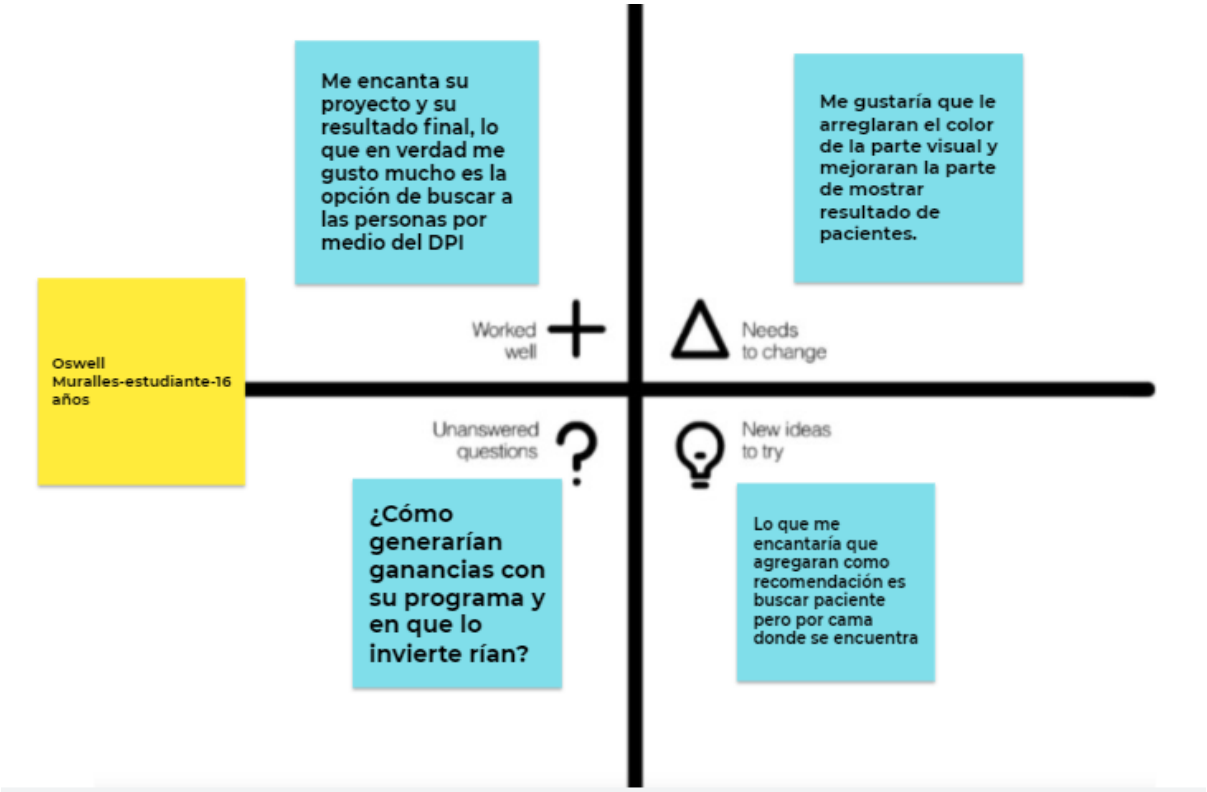
Retroalimentación #4



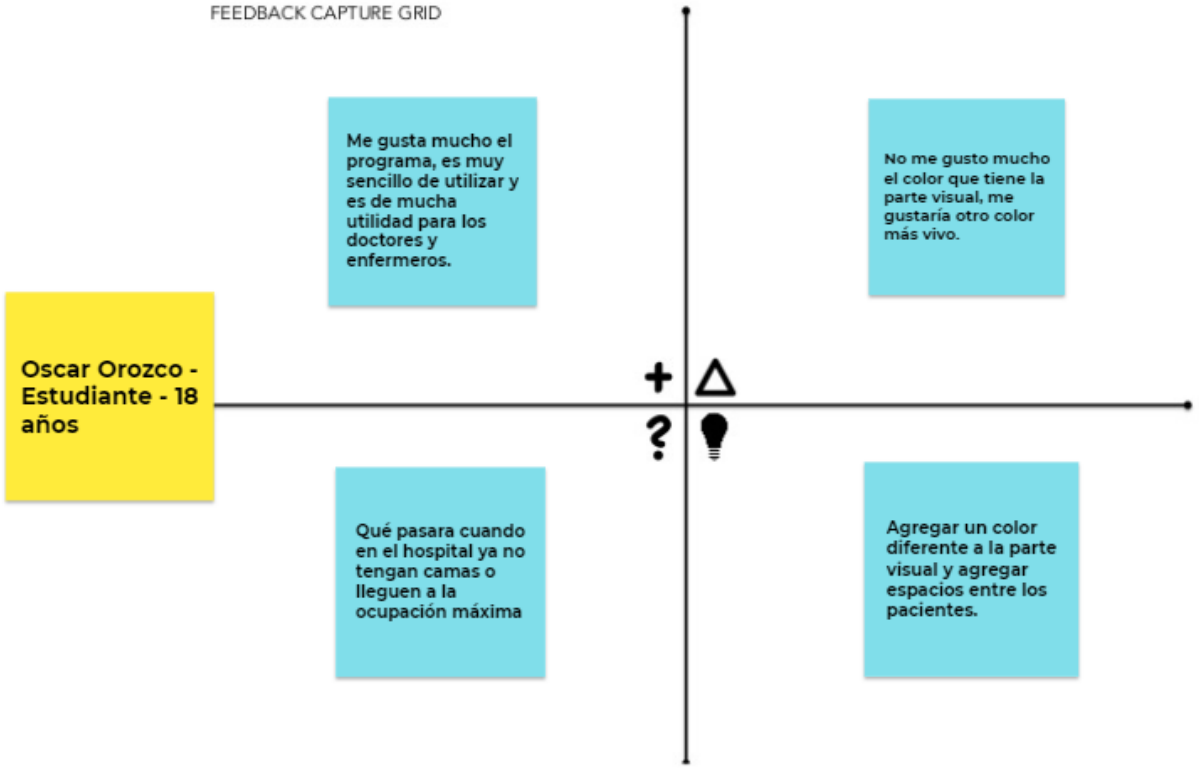
Retroalimentación #5



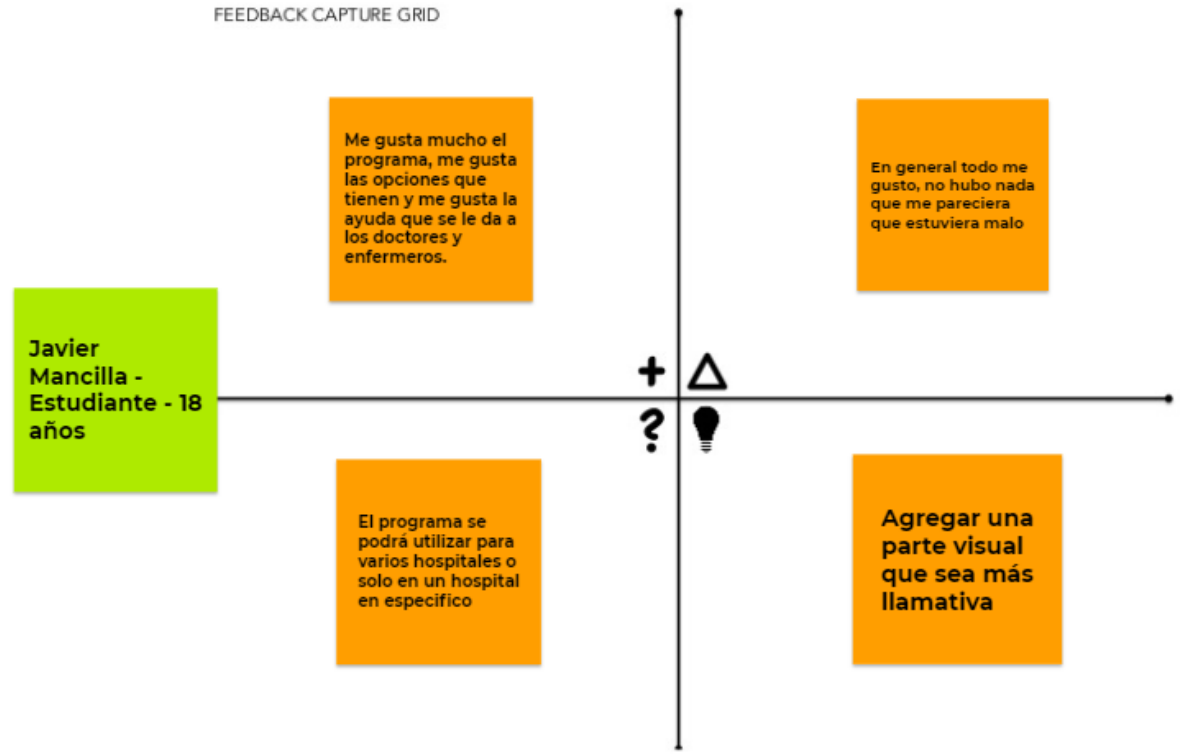
Retroalimentación #6



Retroalimentación #7



Retroalimentación #8



Link al repositorio de GitHub:

https://github.com/21298/Proyecto-POO-10