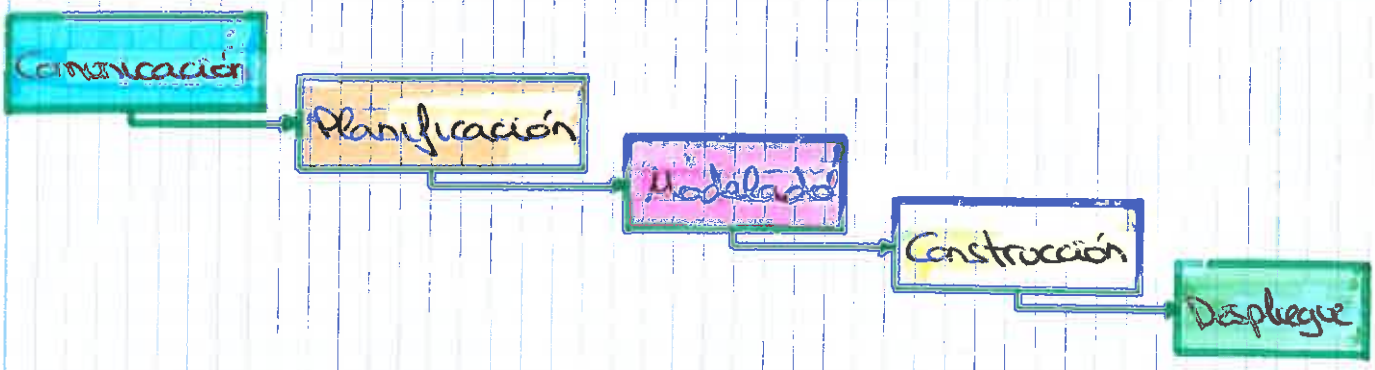
* Examenes:

1. Describa en profundidad como funciona el modelo en cascada y dibuje un pequeño esquema sobre él.

El modelo de proceso en cascada es un modelo prescriptivo clásico, que consta de cinco actividades que se realizan de forma secuencial, es decir, se inicia la primera actividad y cuando finaliza comienza la siguiente actividad, y así hasta la finalización completa del software.
Las cinco actividades que comprende este proceso son:

- Comunicación: Entender los requisitos de los participantes de lo que se requiere.
- Planificación: Crear un plan de proyecto y la planificación de las actividades.
- Modelado: Se crea un modelo para entender los requisitos del software y el diseño final que los cumplirá.
- Construcción: Generar el código y realizar las pruebas necesarias.
- Despliegue: Entrega del software al usuario final.



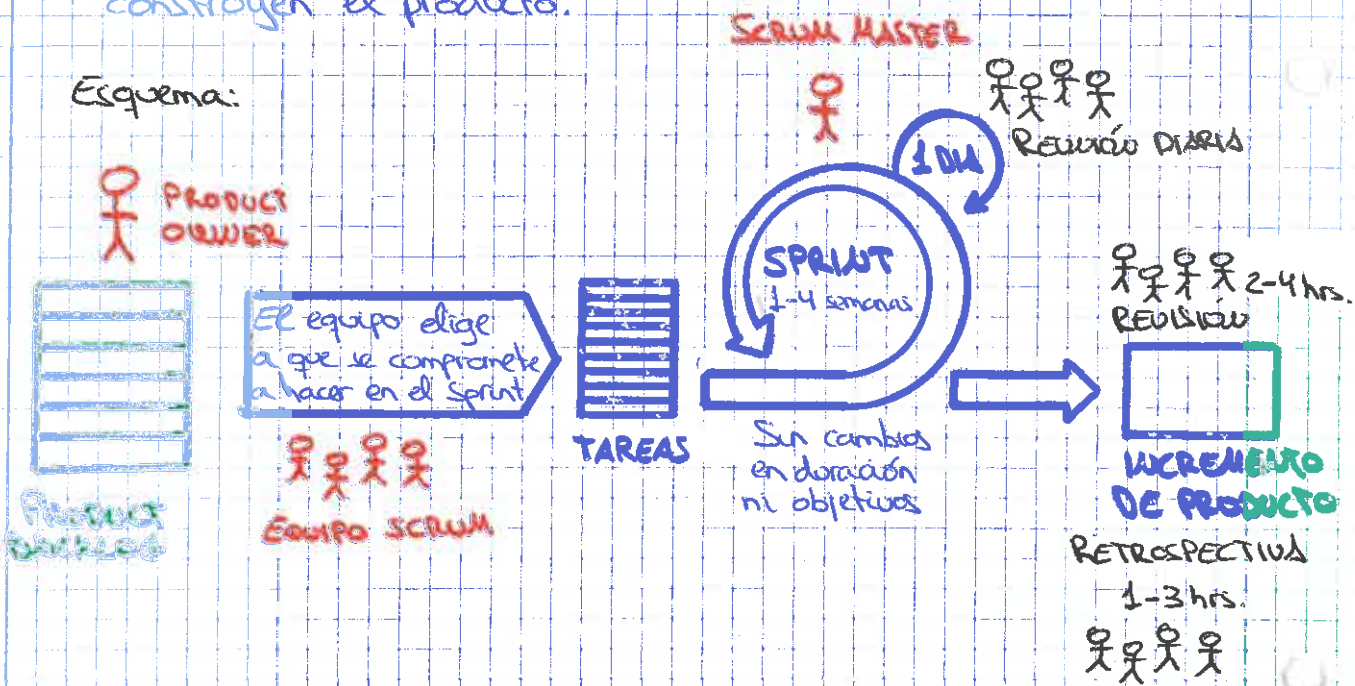
2. Describa el funcionamiento de la metodología SCRUM. Dibuje un esquema del proceso.

Sabemos que cuanto más avanzado este un proyecto, más costoso es realizar cambios. Los procesos ágiles tratan de reducir el coste de estos cambios, dando la oportunidad de evaluar las funcionalidades del software durante el desarrollo.
Scrum es una metodología agil basada en cinco actividades estructurales: requerimientos, analisis, diseño, evolución y entrega.

Roles:

- Product owner: es el responsable de la gestión de la lista de funcionalidades (product backlog).
- Scrum master: es la persona que gestiona el proceso Scrum, ayuda al equipo Scrum a mantenerse enfocado en los objetivos del proyecto y elimina los impedimentos que van apareciendo durante el camino.
- Equipo Scrum: es un equipo multidisciplinar, consistente en un grupo de personas con las habilidades necesarias para transformar los requerimientos del cliente en incrementos de desarrollo en cada Sprint. Son los que realmente construyen el producto.

Esquema:



Flujo del proceso Scrum:

Lo primero de todo es la comunicación con los interesados en el proyecto, en el que recabaremos las características necesarias para el cliente.

En segundo lugar, el product owner se encargara de que las características correctas entren en el product backlog.

El tercer paso es la reunión del equipo scrum, en la que se decidira y planificara lo que el equipo se compromete a hacer en el sprint. De esa reunión se obtienen una serie de tareas que se iran realizando en el sprint.

Los sprint consisten en las unidades de trabajo necesarias para alcanzar un requerimiento definido en la lista de tareas obtenida en la reunión previa del equipo, y que debe ajustarse a unos tiempos (lo común son 30 dias).

Durante el sprint no se introducen cambios, así, el sprint permite al equipo trabajar en un ambiente de corto plazo pero estable.

Reuniones scrum: son una reuniones breves (unos 15 minutos) que el equipo scrum ejecuta a diario.

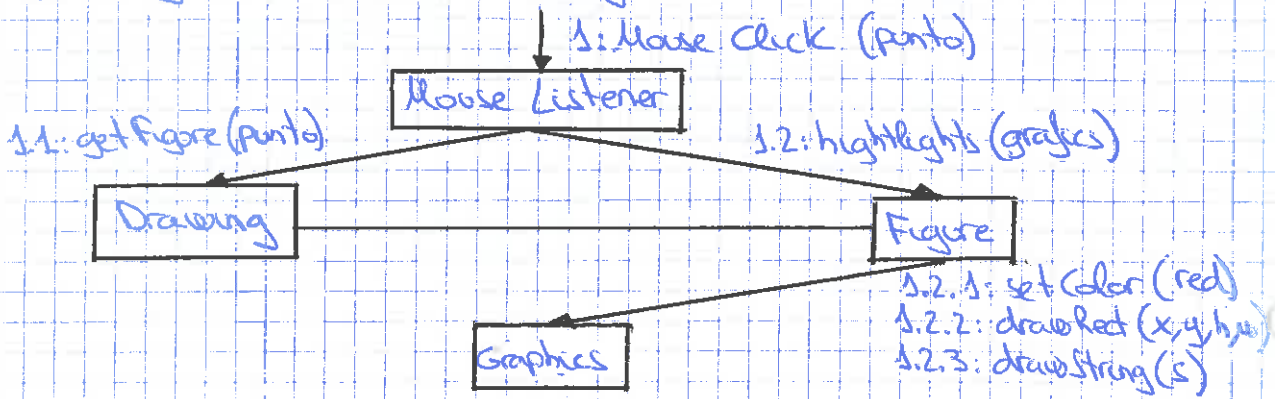
En estas reuniones es donde cada miembro del equipo cuenta los inconvenientes encontrados y buscan la manera de solucionarlo.

Demostraciones preliminares: consisten en entregar el incremento del software al cliente de modo que pueda evaluar la funcionalidad implementada.

3. Describa las características de los diagramas de comunicación. ¿Para qué son útiles?

Los diagramas de comunicación muestran la comunicación dinámica entre los objetos para llevar a cabo una tarea, pero enfatiza las relaciones entre los objetos y clases en lugar del orden temporal.

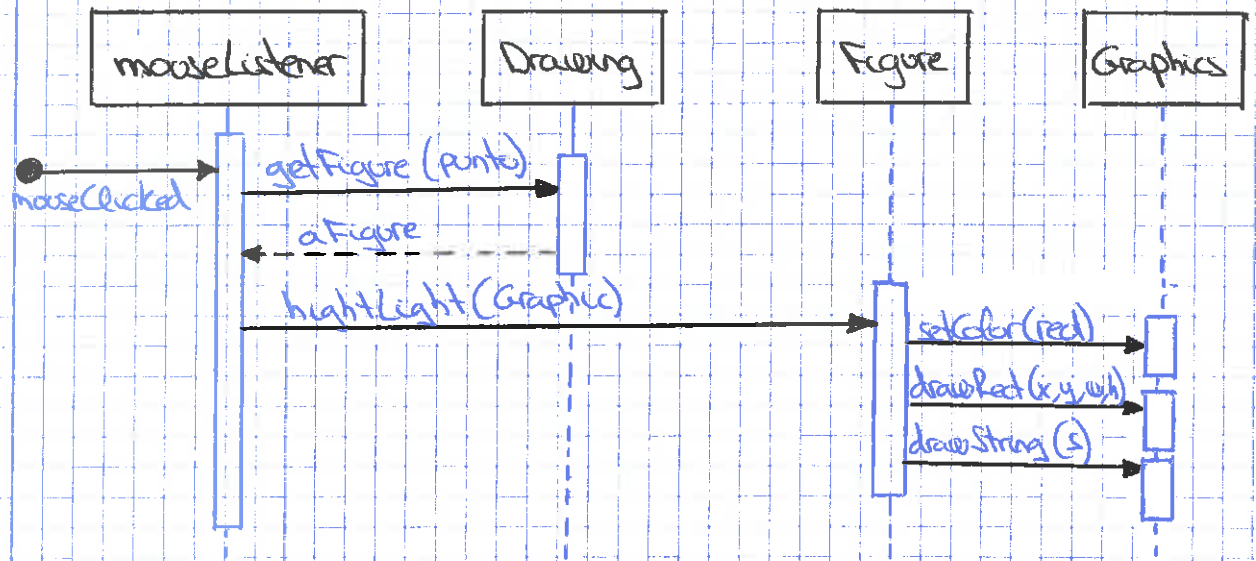
Los mensajes son etiquetados por un número cronológico para poder seguirlos entre un objeto y otro.



4. Describe las características del diagrama de secuencia. ¿Para qué son útiles?

Los diagramas de secuencia describen como y en que orden un grupo de objetos funcionan en un conjunto para llevar a cabo una tarea.

Son útiles para las iteraciones en un caso de uso o en un escenario de un sistema de software.

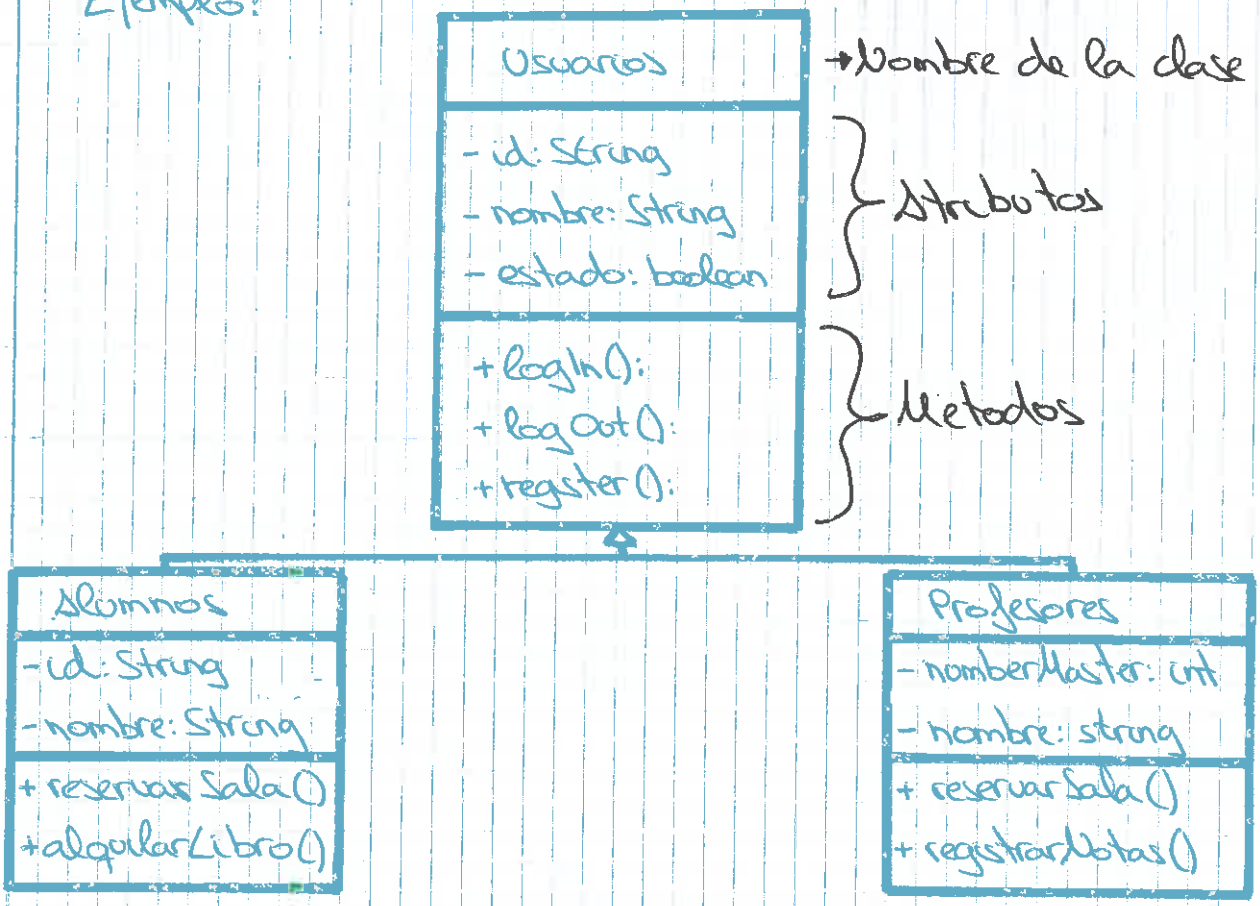


5. ¿Que representa el modelo basado en clases? Por un ejemplo.

El modelo basado en clases representa:

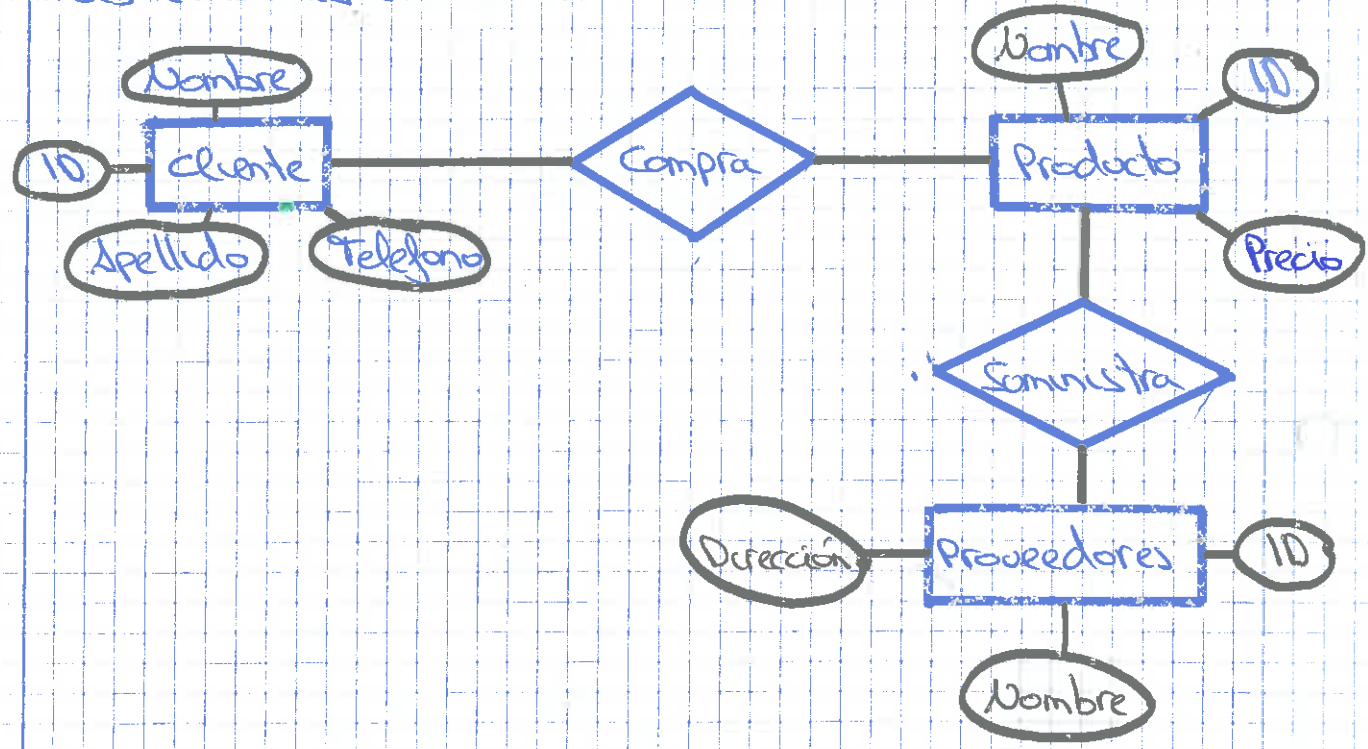
- Objetos → Clases
- Datos → Atributos
- Operaciones → Metodos
- Las interrelaciones entre las clases.

Ejemplo:



6. ¿Como se define el modelado de datos? Realiza un ejemplo de diagrama de entidad-relación.

El modelado de datos define los objetos de datos que se procesan en el sistema, los atributos que tienen y las relaciones entre ellos.



7. Enlista las 6 tareas para el diseño de componentes:

- Identificar las clases de diseño que corresponden con el dominio del problema.
- Identificar las clases de diseño que corresponden con el dominio de la infraestructura.
- Elaborar las clases de diseño no reutilizables.
- Describir las fuentes de datos e identificar las clases necesarias para administrarlos.
- Elaborar representaciones del comportamiento para cada clase o componente.
- Elaborar diagramas de despliegue para dar más detalles de la implementación.

¿Cual es el papel de los líderes de equipo dentro del proyecto?

El líder de équipo debe de motivar, organizar y aportar ideas.
Deben evitar:

- Ambientes de trabajo estresantes.
- Conflictos entre miembros del equipo.
- Mala organización y coordinación de los procesos.
- Vaga definición de los roles y responsabilidades.
- Sensaciones de fracaso.

Las características de un líder son:

- Carisma
- Resolución de problemas y conflictos.
- Delegar, reconocer y valorar al equipo.
- Empatía
- Autocontrol.

¿Como debe ser la cohesión y el acoplamiento en el desarrollo de un sistema de software? ¿Porque?

Debe haber una alta cohesión y bajo acoplamiento.
Cuanto más acoplamiento tengamos, más aumentara el riesgo de errores, pues el acoplamiento es hacer que unas clases dependan de otras.
La alta cohesión implicara un alcance definido, unos límites claros y un contenido delimitado.
Al tener alta cohesión, cada clase hara algo específico y bajo acoplamiento hara que no dependa de otras clases.

Principios relativos al modelado de requisitos:

- Entender y representar el dominio del problema.
- Definir las funciones que realizara el software.
- Representar el comportamiento del software.
- Dividir los modelos de forma jerarquica.
- Avanzar desde lo esencial hasta el final.

Principios relativos al diseño de software:

- El diseño se podra rastrear hasta el modelo de requisitos.
- Tener en cuenta la arquitectura.
- El diseño de los datos es igual de importante.
- Las interfaces se diseñaran con cuidado.
- La interfaz de usuario se ajustara al usuario final.
- Los componentes tendran independencia funcional.
- Los modelos deben ser faciles de entender.
- Se desarrollara de forma iterativa.

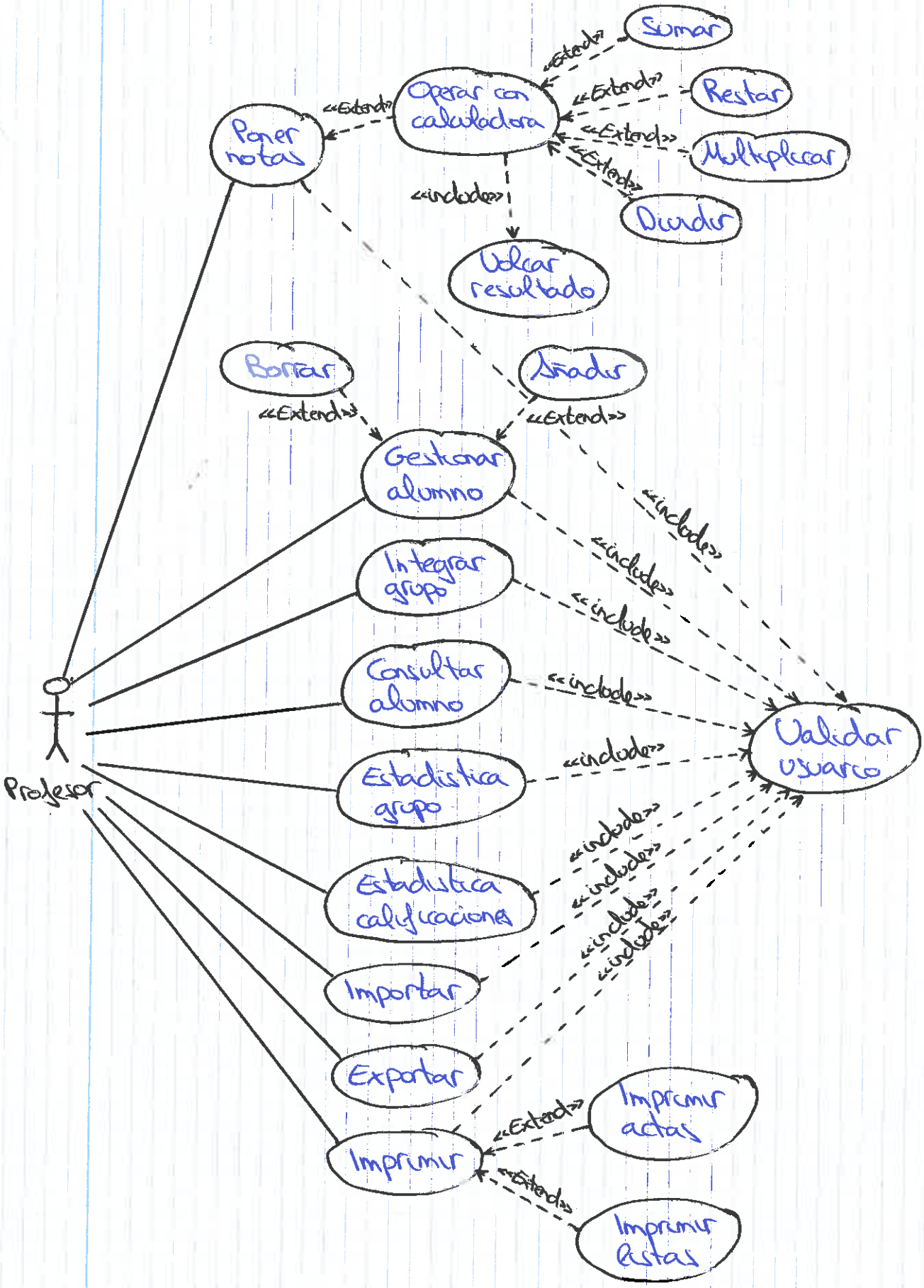
Aseguramiento de la calidad:

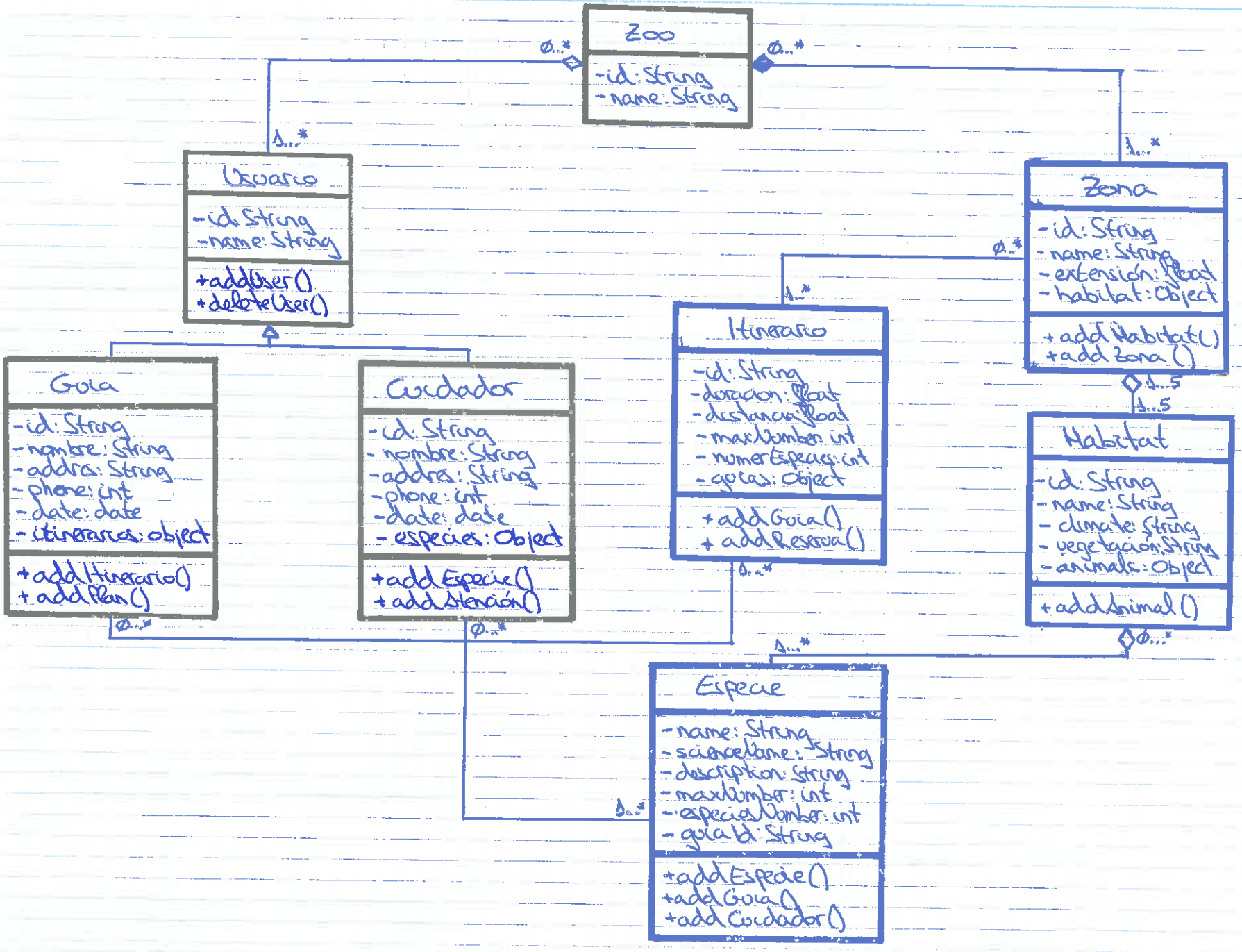
- Realizar auditorias e informes
- Control de la calidad sobre el control de la calidad.
  - Revisiones y auditorias.
  - Pruebas de software
  - Analisis de software
  - Gestionar los cambios
  - Formar al equipo
  - Gestionar los proveedores
  - Gestionar los riesgos
  - Gestionar la seguridad.

Arquitectura en capas:

Se define por una serie de capas donde cada una ejecuta operaciones de software.

- Capa de nucleo
- Capa de utiles
- Capa de aplicación
- Capa de interface de usuario


Sumar
Operar con calculadora
Restar
Poner notas
Multiplicar
Dividir
Volcar resultado
Borrar
Añadir
Gestionar alumno
Integrar grupo
Consultar alumno
Validar usuario
Estadística grupo
Estadística calificaciones
Importar
Exportar
Imprimir
Imprimir actas
Imprimir listas
Profesor
«Extend»
«include»


Zoo
-id: String
-name: String
Usuario
-id: String
-name: String
+addUser()
+deleteUser()
Zona
-id: String
-name: String
-extensión: float
-habitat: Object
+add Habitat()
+add Zona()
Guia
-id: String
-nombre: String
-addra: String
-phone: int
-date: date
-itinerarios: object
+add Itinerario()
+add Plan()
Cuidador
-id: String
-nombre: String
-addra: String
-phone: int
-date: date
-especies: Object
+add Especie()
+add Atención()
Itinerario
-id: String
-duracion: float
-distancia: float
-maxNumber: int
-numerEspecies: int
-guias: object
+add Guia()
+ add Reserva()
Habitat
-id: String
-name: String
-clima: String
-vegetación: String
-animals: Object
+addAnimal()
Especie
-name: String
-scienceName: String
-description: String
-maxNumber: int
-especiesNumber: int
-guiaId: String
+addEspecie()
+addGuia()
+addCuidador()

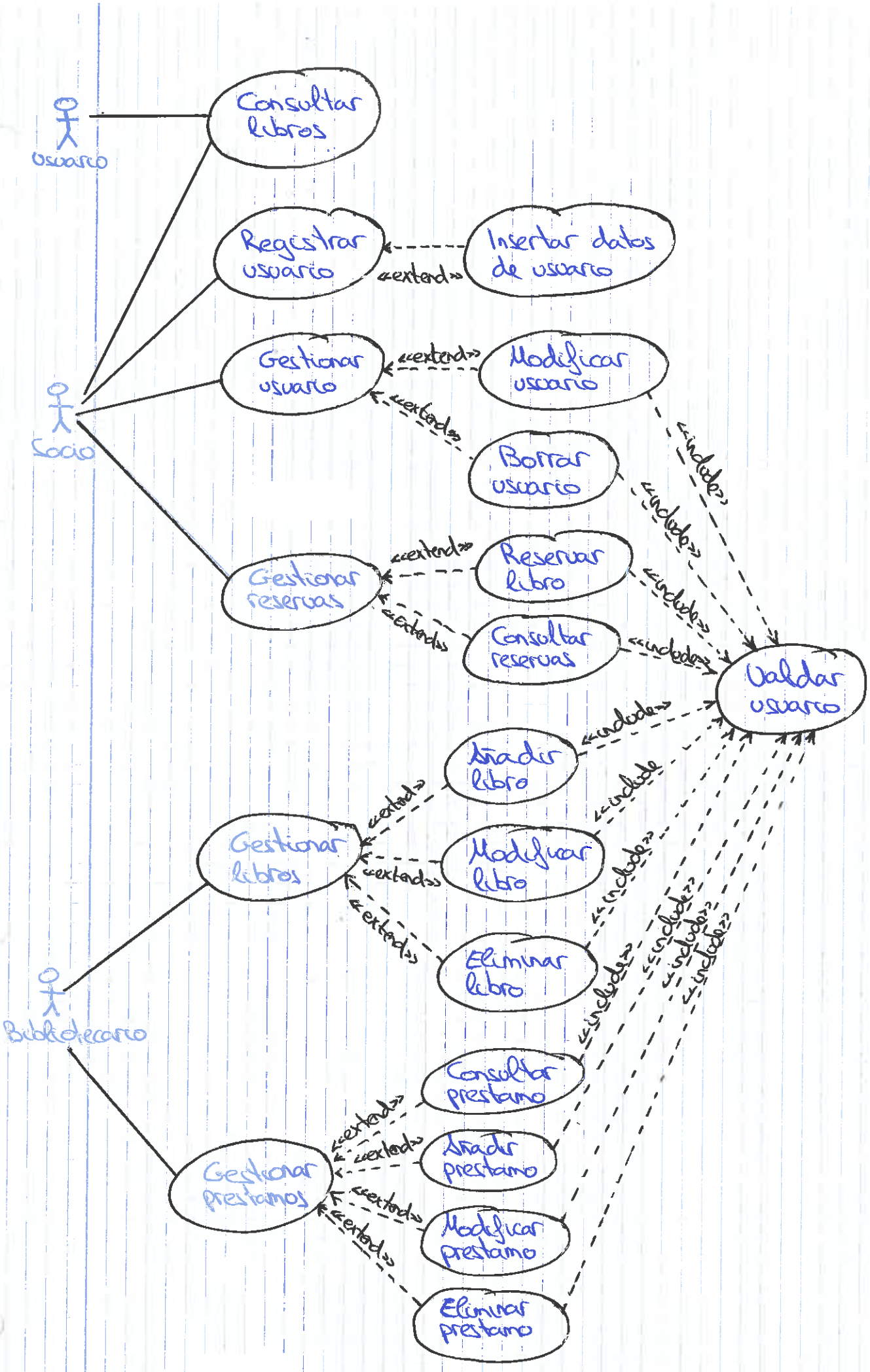

Usuario
Consultar libros
Registrar usuario
Insertar datos de usuario
«extend»
Socio
Gestionar usuario
«extend»
Modificar usuario
«extend»
Borrar usuario
«include»
Gestionar reservas
«extend»
Reservar libro
«extend»
Consultar reservas
«include»
Validar usuario
«include»
Bibliotecario
Gestionar libros
«extend»
Añadir libro
«extend»
Modificar libro
«extend»
Eliminar libro
«include»
Gestionar prestamos
«extend»
Consultar prestamo
«extend»
Añadir prestamo
«extend»
Modificar prestamo
«extend»
Eliminar prestamo

Diagrama casos de uso de la APP.

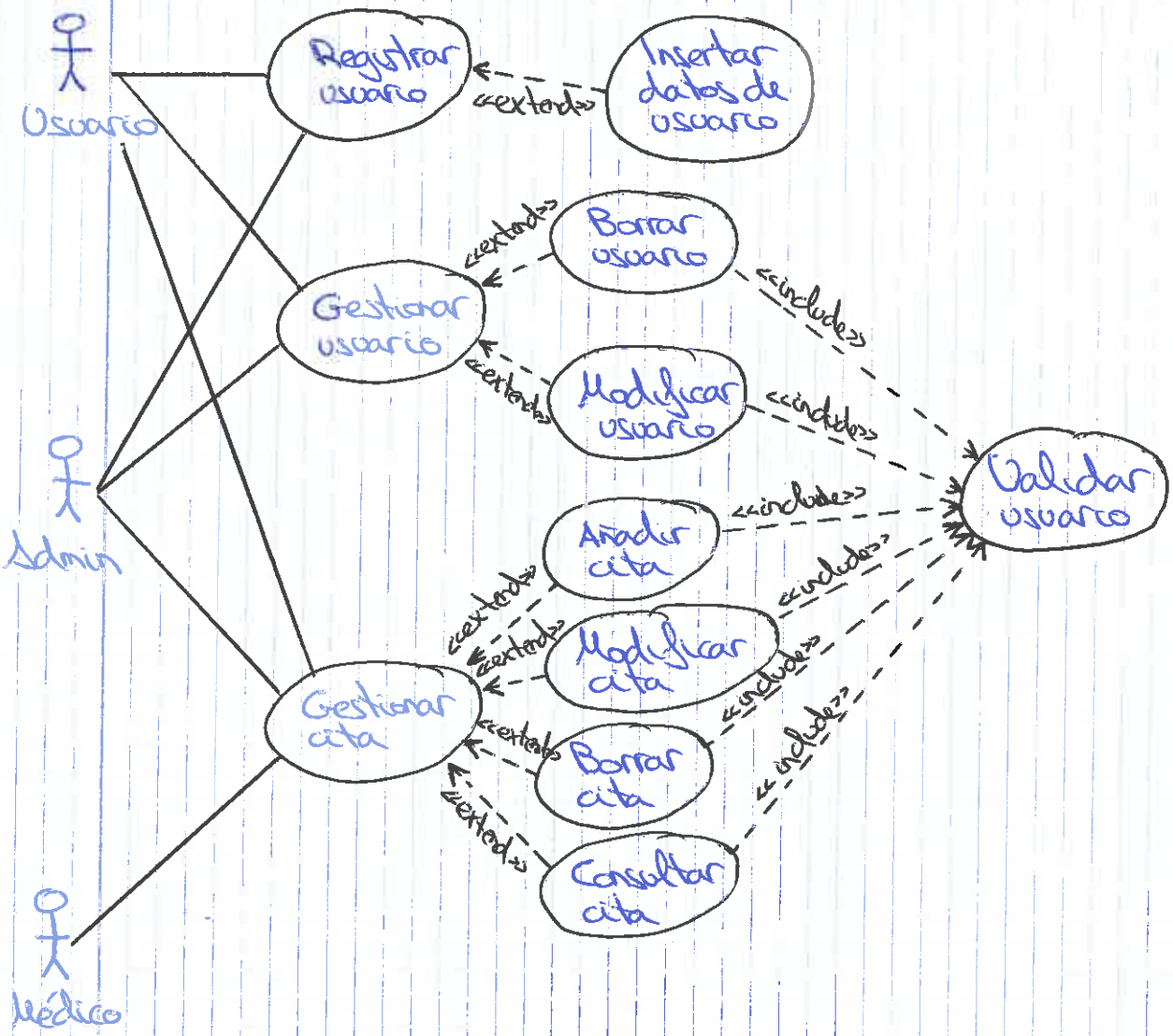
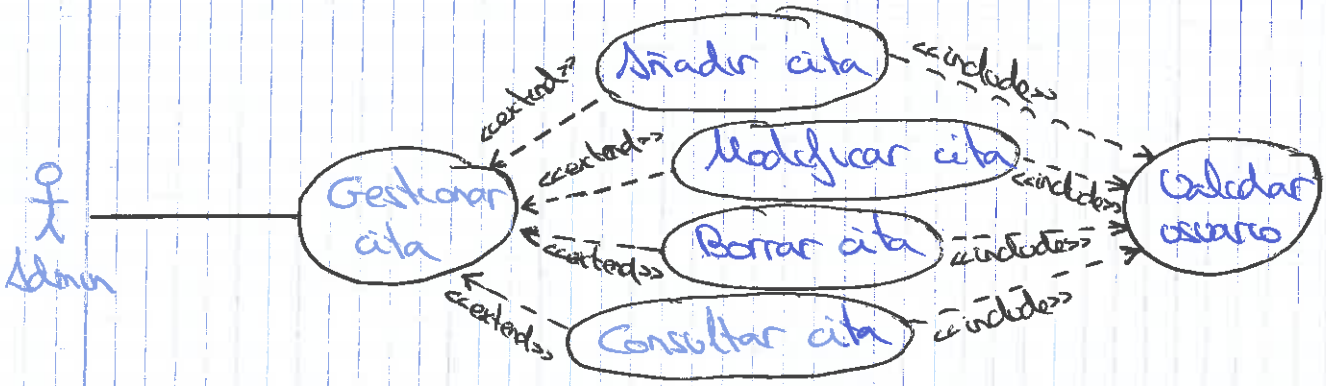
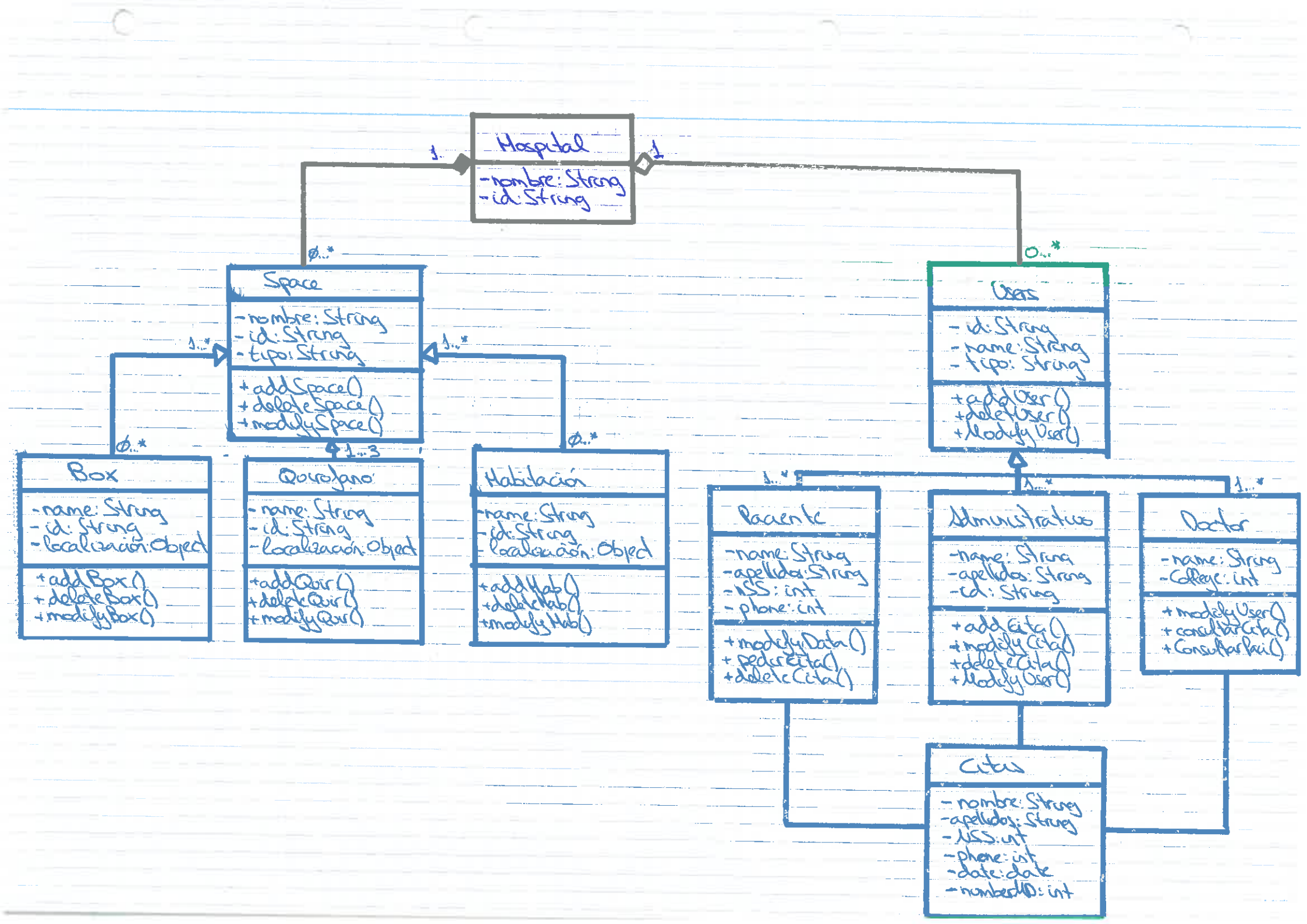


Diagrama de casos de uso de cita telefónica.




Hospital
-nombre: String
-id: String
1
1
Space
-nombre: String
-id: String
-tipo: String
+addSpace()
+deleteSpace()
+modifySpace()
0..*
1..*
1..*
Box
-name: String
-id: String
-localización: Object
+addBox()
+deleteBox()
+modifyBox()
0..*
Quirofano
-name: String
-id: String
-localización: Object
+addQuir()
+deleteQuir()
+modifyQuir()
1..3
Habitación
-name: String
-id: String
-localización: Object
+addHab()
+deleteHab()
+modifyHab()
0..*
User
-id: String
-name: String
-tipo: String
+addUser()
+deleteUser()
+modifyUser()
0..*
Paciente
-name: String
-apellidos: String
-NSS: int
-phone: int
+modifyData()
+pedirCita()
+deleteCita()
1..*
Administrativo
-name: String
-apellidos: String
-id: String
+addCita()
+modifyCita()
+deleteCita()
+modifyUser()
1..*
Doctor
-name: String
-College: int
+modifyUser()
+consultarCita()
+ConsultarPaci()
1..*
Cita
-nombre: String
-apellidos: String
-NSS: int
-phone: int
-date: date
-numberID: int