Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 1
Primer cuatrimestre de 2021

Alumno:	LICERI MARTINEZ, Lucia M.
Número de padrón:	105964
Email:	lliceri@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de una agencia de control de Covid-19 en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Para desarrollar este Trabajo Practico se han asumido los siguientes modelos:

1. Ningun nombre recibido por AlgoVid deberá ser vacio. Esto se supone a partir de que AlgoVid utiliza los nombres asignados como forma de comunicación con el usuario. De ser inexistentes dificultaría la interacción y perjudicaría la claridad del código.

2. No debe haber dos Personas, Burbujas o Colegios con el mismo nombre. Como se enunció previamente, el nombre es la única herramienta para identificar de que objeto particular se esta hablando por lo que debe ser distinguible del resto de los objetos. Además se supone que se debe evitar agregar duplicados ya sea a las burbujas o a los colegios.

3. Supongo que puedo tener uno o más colegios. Si bien las pruebas dadas por la catedra no lo requieren, considero que mi modelo debe poder manejar más de un colegio sin inconvenientes.

3. Modelo de dominio

El diseño de mi trabajo practico consta de una clase AlgoVid, la cual con el fin de informar la situación de Personas, Burbujas o Colegios respecto al Covid-19, delega sus responsabilidades a otras clases. A través de la recepción de Strings, AlgoVid le informa al ususario diferentes situaciones.

La más importante, quizas, es la que determina si una persona puede o no circular. Para lograrlo, AlgoVid recibe el nombre de la persona, la busca en su coleccion ordenada de Personas y le delega la responsabilidad de verificar cual es su estado de circulación según las restricciones, que por default debería tener prohibida la circulación. Para modificar dicho estado se le solicita a AlgoVid que vacune a la persona o que le de permiso de personal escencial, para esto delega la responsabilidad a la persona. La persona delega la responsabilidad de diagnosticarse a Salud, quien en base a los sintomas resuelve un diagnostico que será parte de lo que determine si efectivamente puede o no circular. Por otro lado, persona verifica si es de riesgo, si tuvo contacto estrecho o si pertenece a una burbuja pinchada que son las condiciones que le prohibirían la circulación a pesar de estar vacunado o de ser escencial.

AlgoVid tambien puede informar si una Burbuja está o no pinchada, delegando la responsabilidad a la Burbuja. Esta delegará la responsabilidad de diagnosticar a las personas a cada una de ellas. De haber algún 'Positivo' se la considerará pinchada. A su vez AlgoVid informa si hay o no clases presenciales en los colegios, basandose en la cantidad de burbujas pinchadas. Si mas del 40 % de las burbujas se encuentran pinchadas, no hay clases presenciales en ese colegio.

4. Diagramas de clase

Dada la complejidad del modelo decidí presentar más de un diagrama. Se muestra un diagrama de como se relacionan las pricipales clases y debajo dos diagramas de detallan particularidades de la clase Persona y Salud.

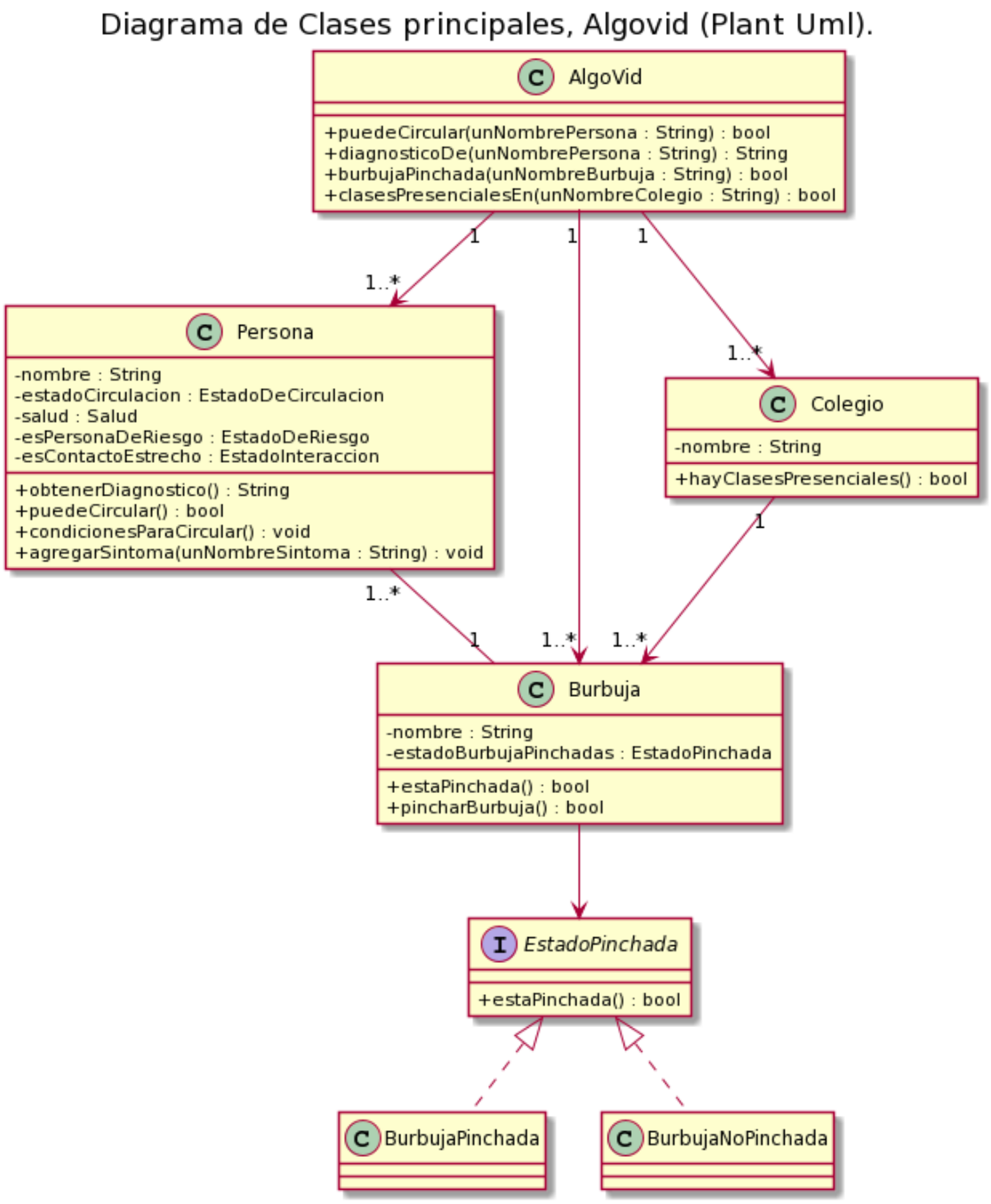

Figura 1: Diagrama de las principales clases.

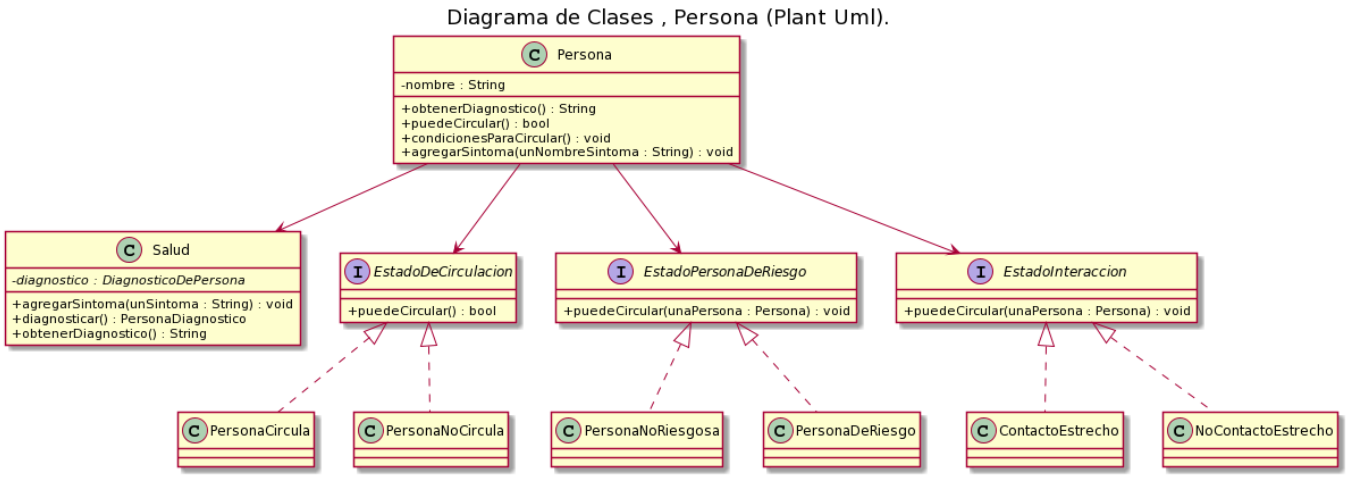

Figura 2: Diagrama de la clase Persona.

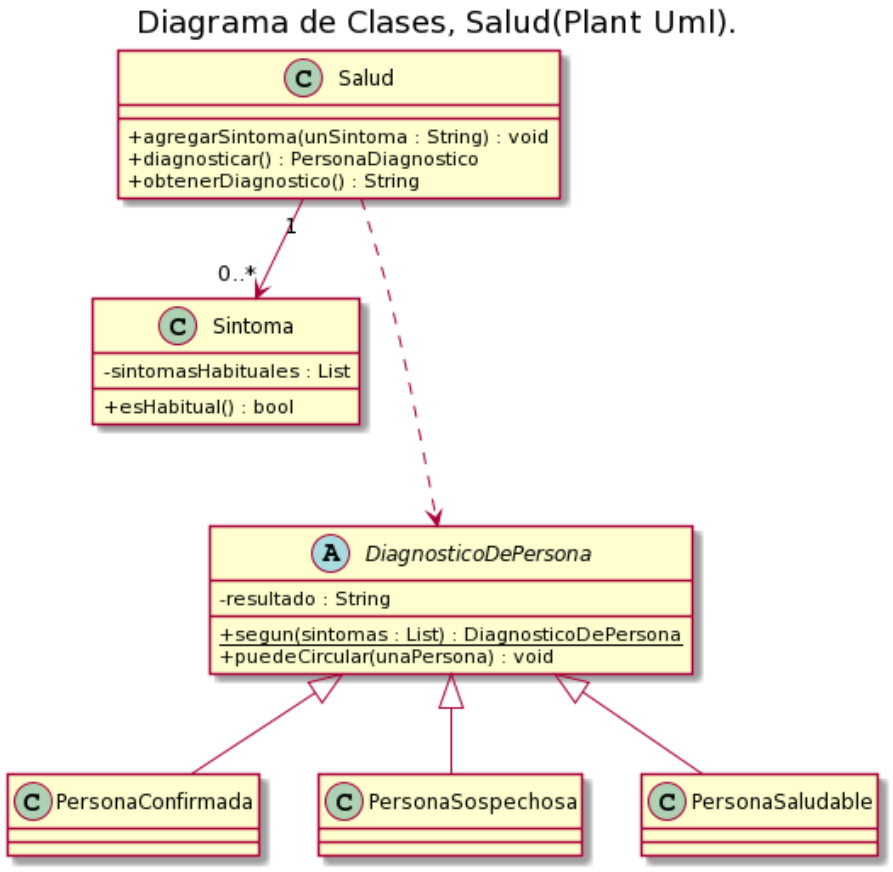

Figura 3: Diagrama de la clase Salud.

5. Detalles de implementación

5.1. Diagnostico como clase abstracta

Al notar que las clases PersonaSaludable, PersonaSospechosa y PersonaConfirmada respondian al mensaje puedeCircular de manera diferente decidí agruparlas dentro de la clase abstracta DiagnosticoDePersona, donde tambien definí otro mensaje que es común a todas, el getter de resultado, evitando así el código repetido.Tiene sentido pensar en esa relacion 'es un' para cada una de las clases hijas en cuestion ya que, por ejemplo, una PersonaSaludable **es un** DiagnosticoDePersona. Ademas esta clase abstracta cuenta con un metodo de clase el cual, a través de la lógica propuesta por las pruebas de la cátedra, devolverá a la clase hija correspondiente dependiendo de los sintomas de la persona. Esto le da al codigo mayor legibilidad y permite encapsular este comportamiento.

El método de clase que terminó de definir la decisión de optar por utilizar herencia por sobre una interfaz:

```
segun: sintomas
|cantidadSintomas diagnostico|
diagnostico := PersonaSaludable new.
cantidadSintomas := (sintomas size).
(cantidadSintomas ~= 0) ifTrue: [ diagnostico := PersonaSospechosa new ].
(cantidadSintomas = 3) ifTrue: [ sintomas detect: [ :unSintoma | (unSintoma esHabitual) ]
ifFound: [diagnostico := PersonaConfirmada new ]].
(cantidadSintomas >= 4) ifTrue: [ diagnostico := PersonaConfirmada new ].

^diagnostico.
```

5.2. Polimorfismo en los estados de Persona

Como el diagrama de la clase Persona demuestra, la clase utiliza las interfaces EstadoDeCirculacion, EstadoPersonaDeRiesgo y EstadoInteraccion. Esto fue implementado para darle una respuesta polimórfica al estado binario de la circulación de una persona(circula o no circula). Estas clases que implementan las interfaces previamente mencionadas responden el mensaje puedeCircular permitiendo o prohibiendo la circulación de Persona según corresponda. En el caso particular de EstadoDeCirculacion sus implementaciones responden con true o con false a puedeCircular. Esto le aporta mayor flexibilidad a la implementación cumpliendo con lo propuesto por el paradigma.

6. Excepciones

BurbujaRepetidaError Esta excepcion es arrojada al intentar agregar a un colegio una burbuja que ya existe dentro del mismo, evitando así duplicados que perjudiquen la obtención de resultados.

DosNombresIgualesError Esta excepcion fue creada con el fin de ser coherente a los supuestos. Agregar dos personas, burbujas o colegios a AlgoVid con el mismo nombre resultará en el arrojo de esta excepcion

NombreInexistenteError Este error aparece cuando se intenta obtener informacion de un objeto que se cree que está en alguna de las colecciones de AlgoVid y este no fue creado aún.

NombreVacioError Con el fin de cumplir con los supuestos, esta excepción es lanzada cuando el usuario intenta agregar ya sea una Persona, Buruja o un Colegio pasandole por parametro un String vacio, acción que perjudicaría el desarrollo del resto de las funcionalidades.

PersonaRepetidaError Esta excepción es lanzada cuando se pretende agregar a una Burbuja una Persona que ya existe en la misma.

SintomaRepetidoError Si el usuario intenta agregarle a una Persona un Sintoma que ya fue agregado previamente, se lanzará esta excepcion. Es de gran importancia evitar sintomas duplicados ya que la cantidad de sintomas agregados es clave para determinar el diagnostico correctamente.

7. Diagramas de secuencia

Los diagramas de secuencia buscan mostrarle al lector las partes más interesantes y complejas del codigo. Se hace especial enfasis en la forma de definir si una persona puede o no circular ya que es la parte más compleja del modelo.

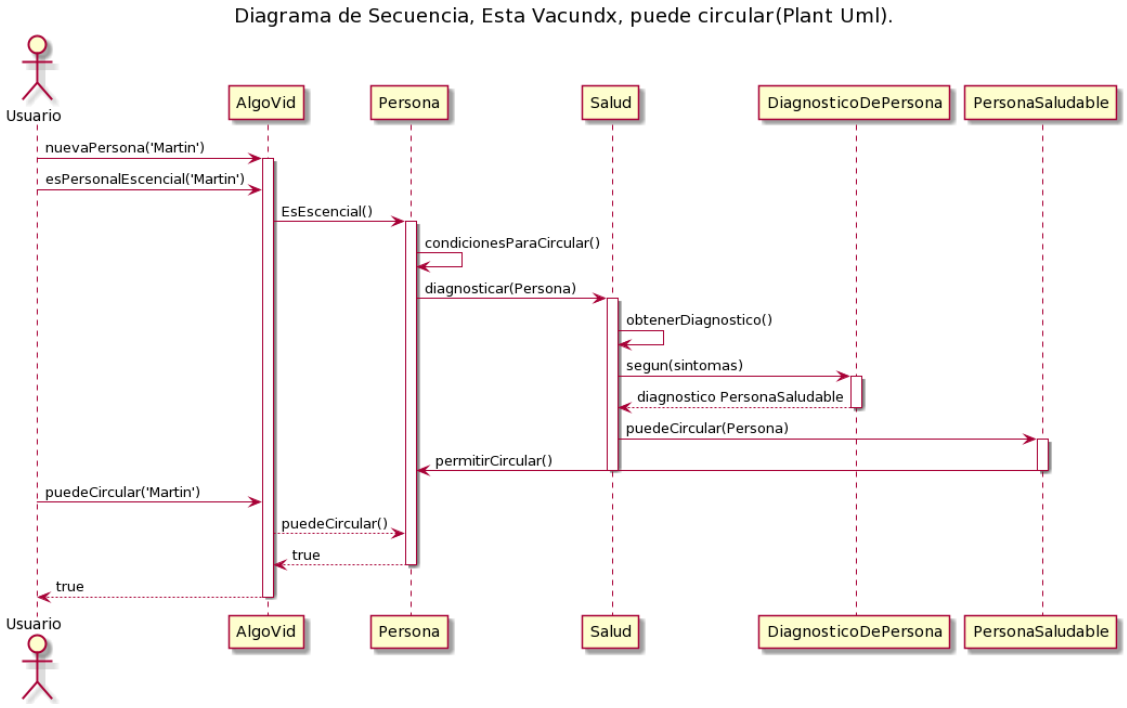

Figura 4: Una Persona vacunada puede circular.

Aqui se muestra claramente el proceso de vacunación. Este diagrama permite ver como es la interacción entre Salud y la clase abstracta DiagnosticoDePersona cuando el resultado es PersonaSaludable y como es esta la responsable de permitir la circulacion.

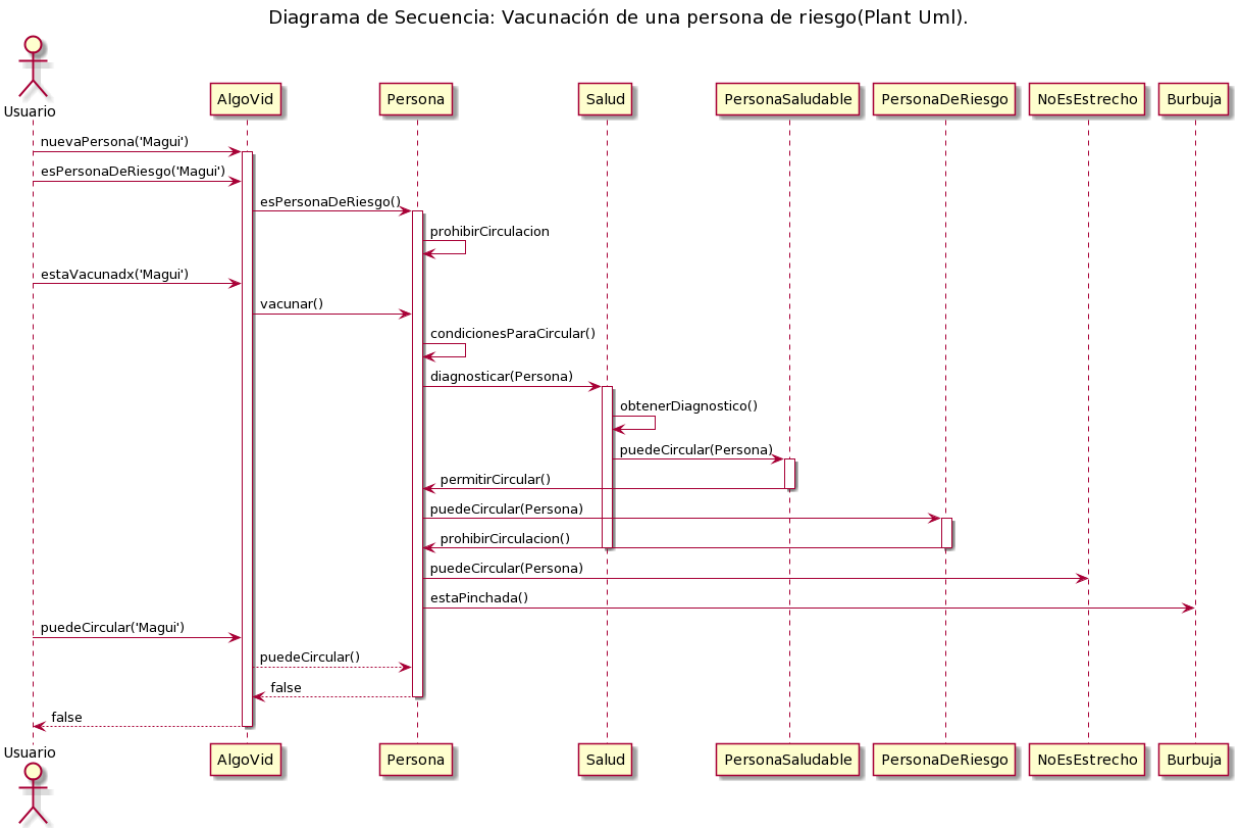

Figura 5: Vacunacion de una persona de riesgo, se le prohibe circular de todas formas.

Vacunar a una persona debería resultar en permitirle circular. En este diagrama queda demostrada la importancia de corroborar todas las condiciones antes de habilitar la circulación ya sea por que se vacuna a la persona o se le otorga permisos de personal escencial. Podemos ver como a pesar de que al ser una persona saludable a la que se le permite circular, al mandarle el mensaje puedeCircular a PersonaDeRiesgo se le vuelve a prohibir la circulación.

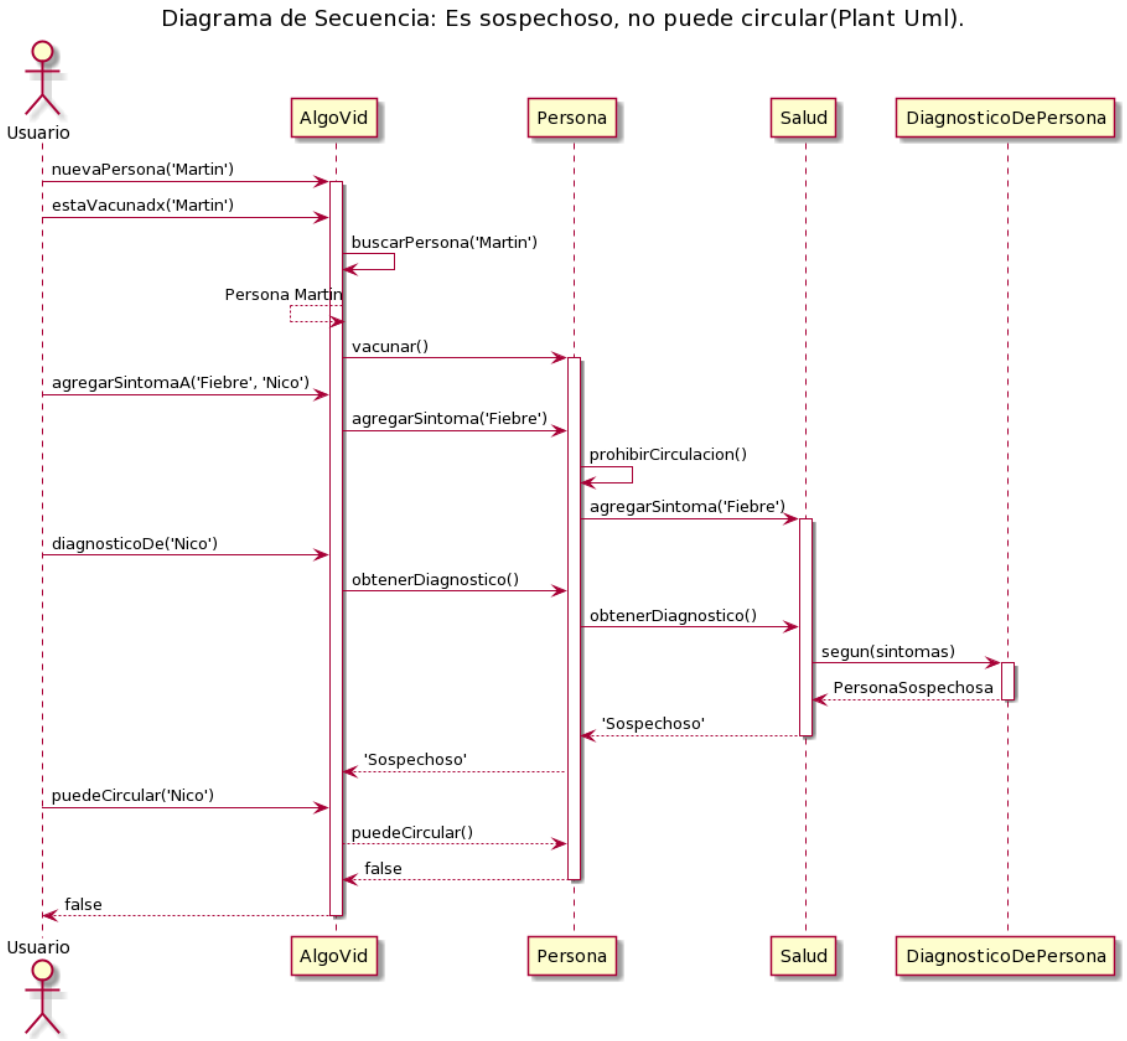

Figura 6: Se diagnostica a una persona como sospechosa.

Este diagrama permite ver la necesidad de una clase DiagnosticoDePersona que implemente un metodo de clase que devuelva el diagnostico correspondiente, se puede notar como esta recibe la colección de sintomas y devuelve una instancia de una de sus clases hijas según el diagnostico obtenido a partir de los sintomas pasados por parámentro. Además demuestra como es que se le prohíbe la circulación a una persona que previamente había sido habilitada

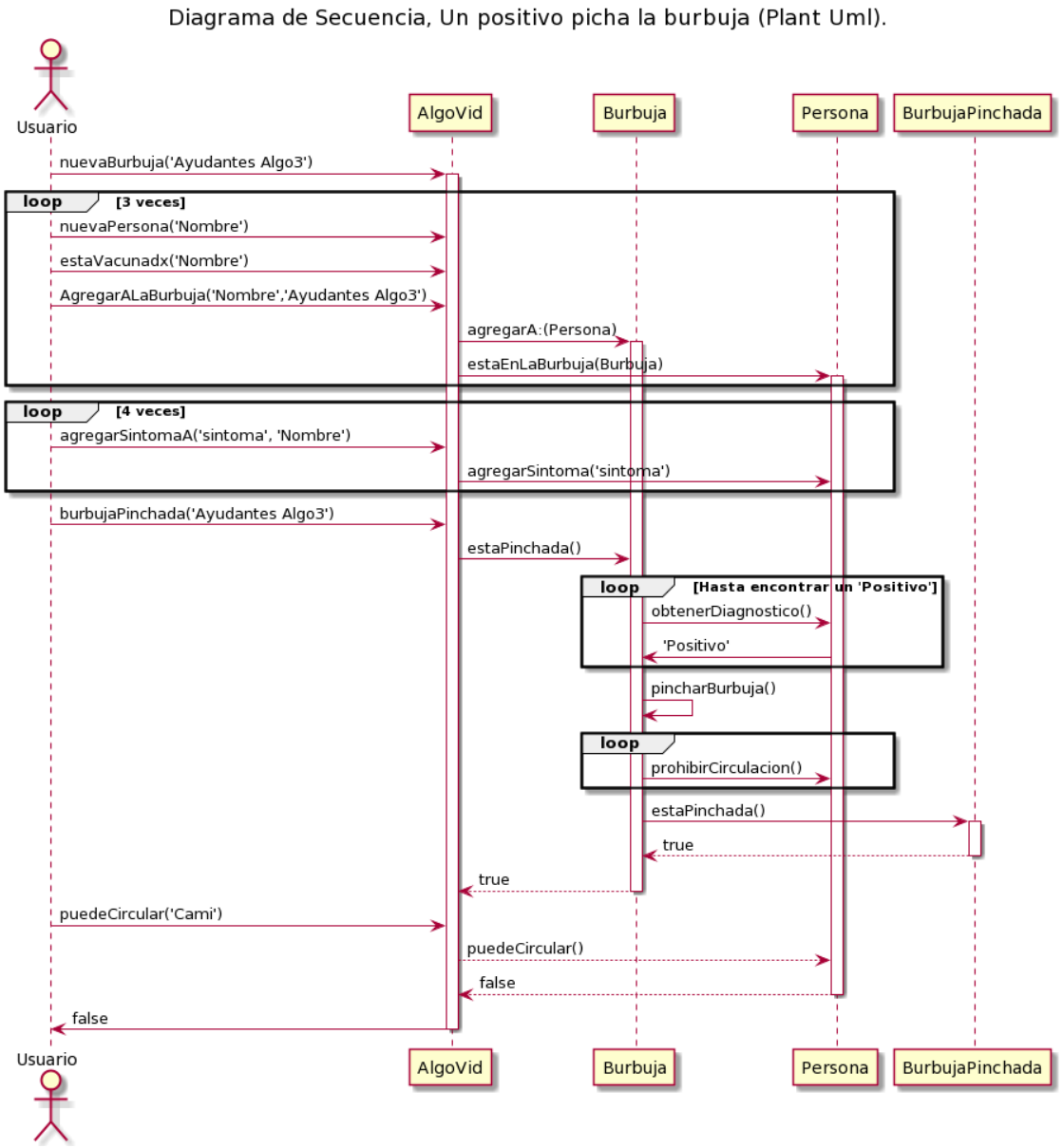

Figura 7: Una Persona con 4 sintomas da un diagnositco 'Positivo' y pincha la Burbuja.

Este diagrama de secuencia demuestra como es que al haber alguien con cuatro sintomas, se lo diagnostica 'Positivo' y la burbuja pasa a estar pinchada. Al verificar el diagnostico de cada una de las personas en la burbuja, al encontrar un positivo, se le prohíbe la circulación a todas las personas en la burbuja.

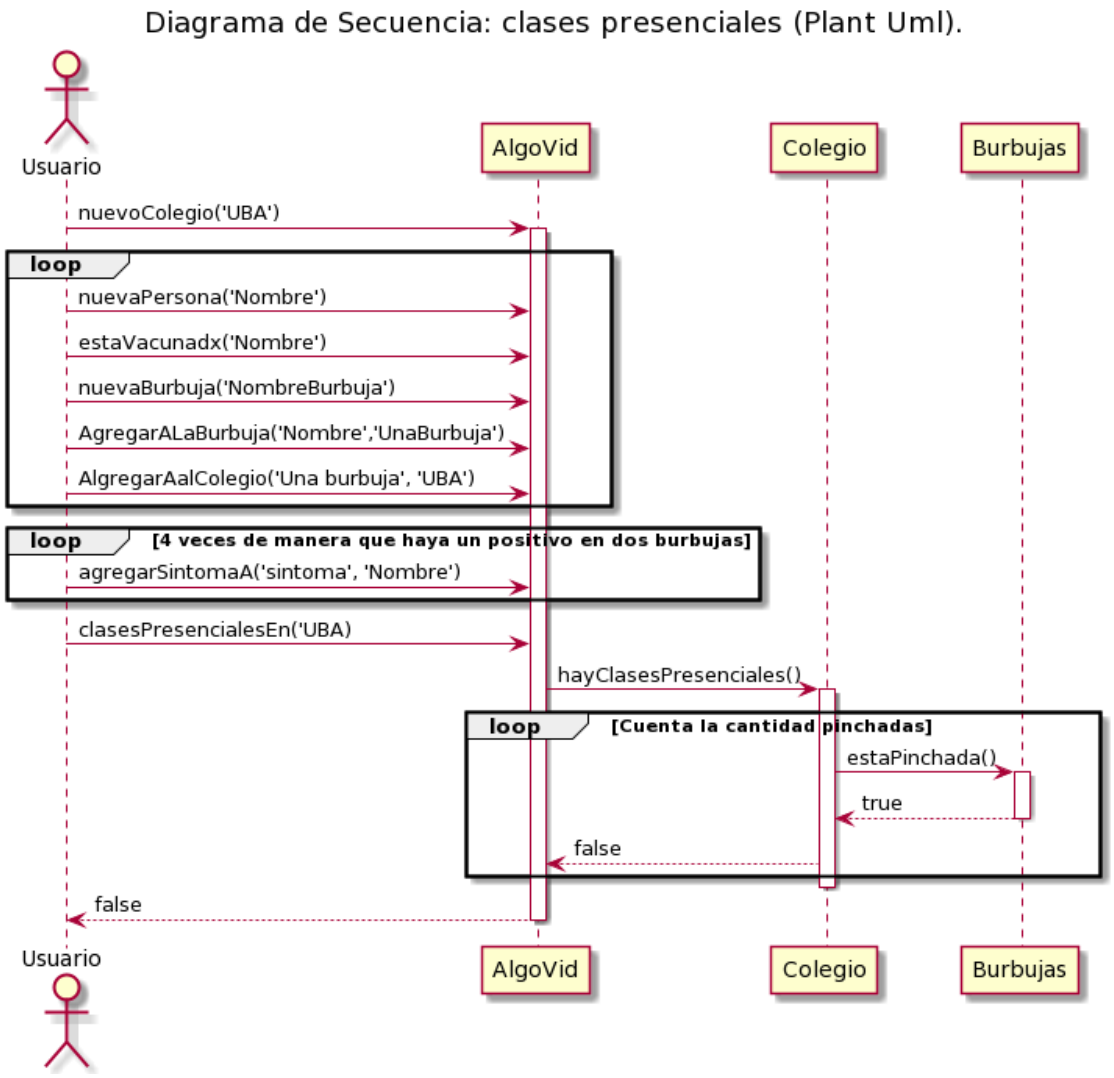

Figura 8: Mas del 40 % de las burbujas pinchadas, no hay clases presenciales.