Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Cátedra Suárez
1ºC 2021

Alumno:	DIEGUEZ, Manuel
Número de padrón:	106146
Email:	jdieguez@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 permisos de circulación y presencialidad escolar en el contexto actual por COVID-19 en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

- Una persona solo puede pertenecer a una burbuja: si bien una persona en la realidad pertenece a varias burbujas, el TDD no señala que en el dominio del problema una persona pertenezca a dos burbujas.
- Un sintoma no puede añadirse dos veces.
- Un contacto estrecho se tiene en cuenta si el diagnóstico de otra persona es sospechoso o positivo.
- Una burbuja no cuenta con un estado que represente si esta pinchada o no, sino que se le realiza un diagnóstico al instante.
- Se llevará el mismo procedimiento en casos de covid sospechoso y positivo.
- Un contacto estrecho es instantáneo, mas no permanente: lo que significa que si una persona tuvo contacto estrecho con una persona de diagnóstico negativo, va a poder circular aunque después la otra persona de sospechoso o positivo.

3. Diagramas de clase

El modelo del programa está basado sobre una clase AlgoVid, de la cuál el usuario dispondrá una instancia para efectuar cambios dentro del entorno que quiere crear.

```
AlgoVid
    colegios : OrderedCollection
    burbujas : OrderedCollection
    personas : OrderedCollection
```

A partir de ella puede crear, modificar y organizar instancias de entidades como personas, burbujas y colegios. Para que tal clase funcione sin lógica propia, es menester el diseño de clases designadas a interpretar los mensajes que recibe AlgoVid.

Estas entidades contenidas dentro de AlgoVid pueden enlazarse entre sí, puesto que un colegio puede contener burbujas, una burbuja puede contener personas y finalmente una persona puede contener una burbuja (Supuesto).

Adicionalmente se crea una clase Diagnóstico que será usada por AlgoVid para determinar el comportamiento de sus mensajes.

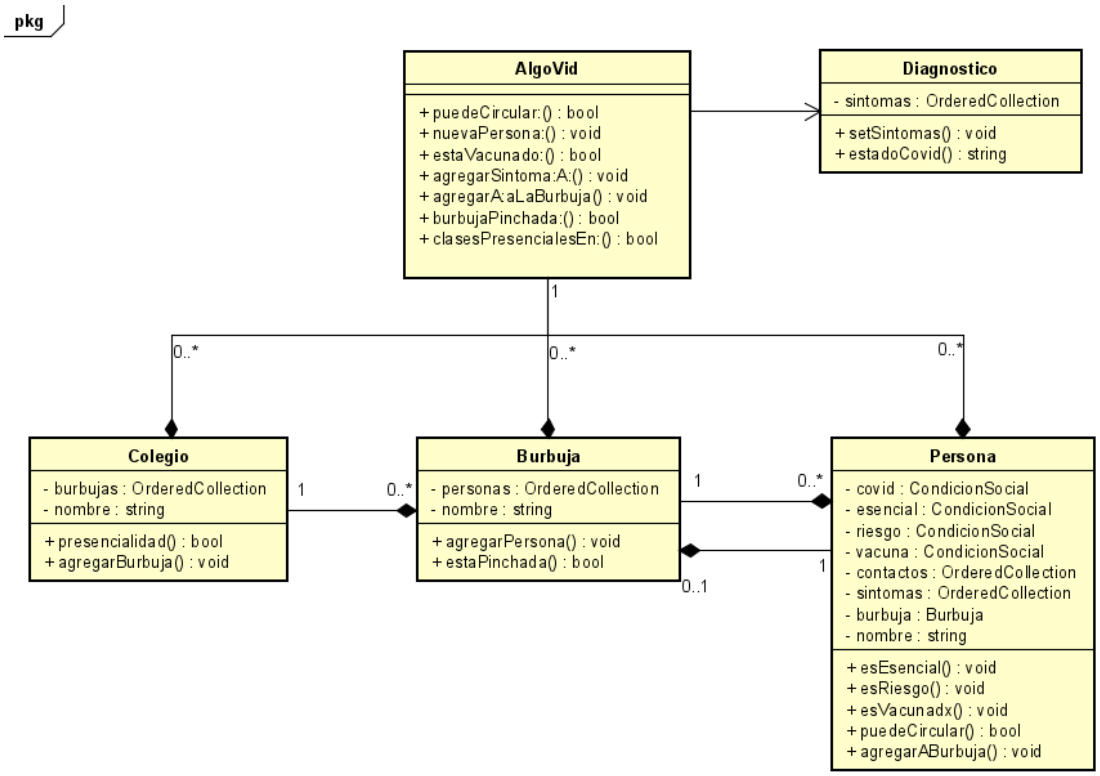

Figura 1: Diagrama de clases general.

La clase Persona es un pilar fundamental dentro del funcionamiento del programa. Esta compuesta por diferentes atributos cuyo valor varía en subClases de la interfaz CondicionSocial. Puesto a que una persona va a verse catalogada dentro de distintas condiciones, esta solución resulta ventajosa debido a su gran abstracción, puesto a que dentro del dominio del problema, las condiciones sociales son binarias, por ende no es necesaria la creación e interpretación de cada condicion de forma individual.

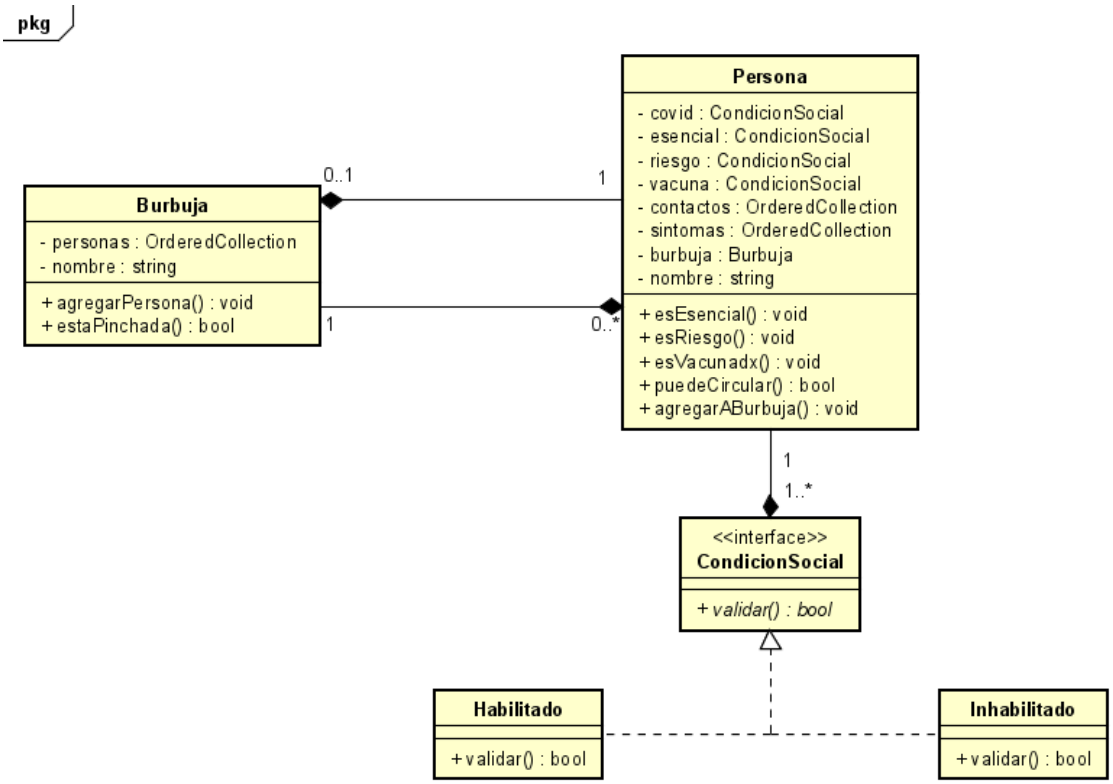

Figura 2: Diagrama de clase Persona.

4. Detalles de implementación

4.1. Delegaciones

Las principales delegaciones, como ya se mencionó anteriormente en el informe, es entre la clase AlgoVid y las clases Persona, Burbuja y Colegio. Esto nos garantiza que dentro de la primera no haya ningún tipo de lógica/código, sino que los problemas se resuelven dentro de los métodos que implemente cada entidad, asegurándonos el resultado esperado.

Adicionalmente se implementa una interfaz para clases puramente metodológicas (no poseen atributos) como lo son las clases Habilitado e Inhabilitado, que heredan de CondicionSocial la firma de su único método "validar() : bool".

```
Object subclass: #CondicionSocial

validar
^self subclassResponsibility

    CondicionSocial subclass: #Habilitado
    CondicionSocial subclass: #Inhabilitado
```

5. Excepciones

PersonaNoEncontradaError Error generado por intentar buscar una instancia de Persona inexistente dentro de una colección de personas.

BurbujaNoEncontradaError Error generado por intentar buscar una instancia de Burbuja inexistente dentro de una colección de burbujas.

ColegioNoEncontradoError Error generado por intentar buscar una instancia de Colegio inexistente dentro de una colección de colegios.

SintomaExistenteError Error generado por intentar agregar un sintoma existente dentro de una coleccion de sintomas de una persona.

6. Diagramas de secuencia

En este apartado se verán los principales diagramas de secuencia del programa, generados a partir de el envío de mensajes la instancia de AlgoVid poseída.

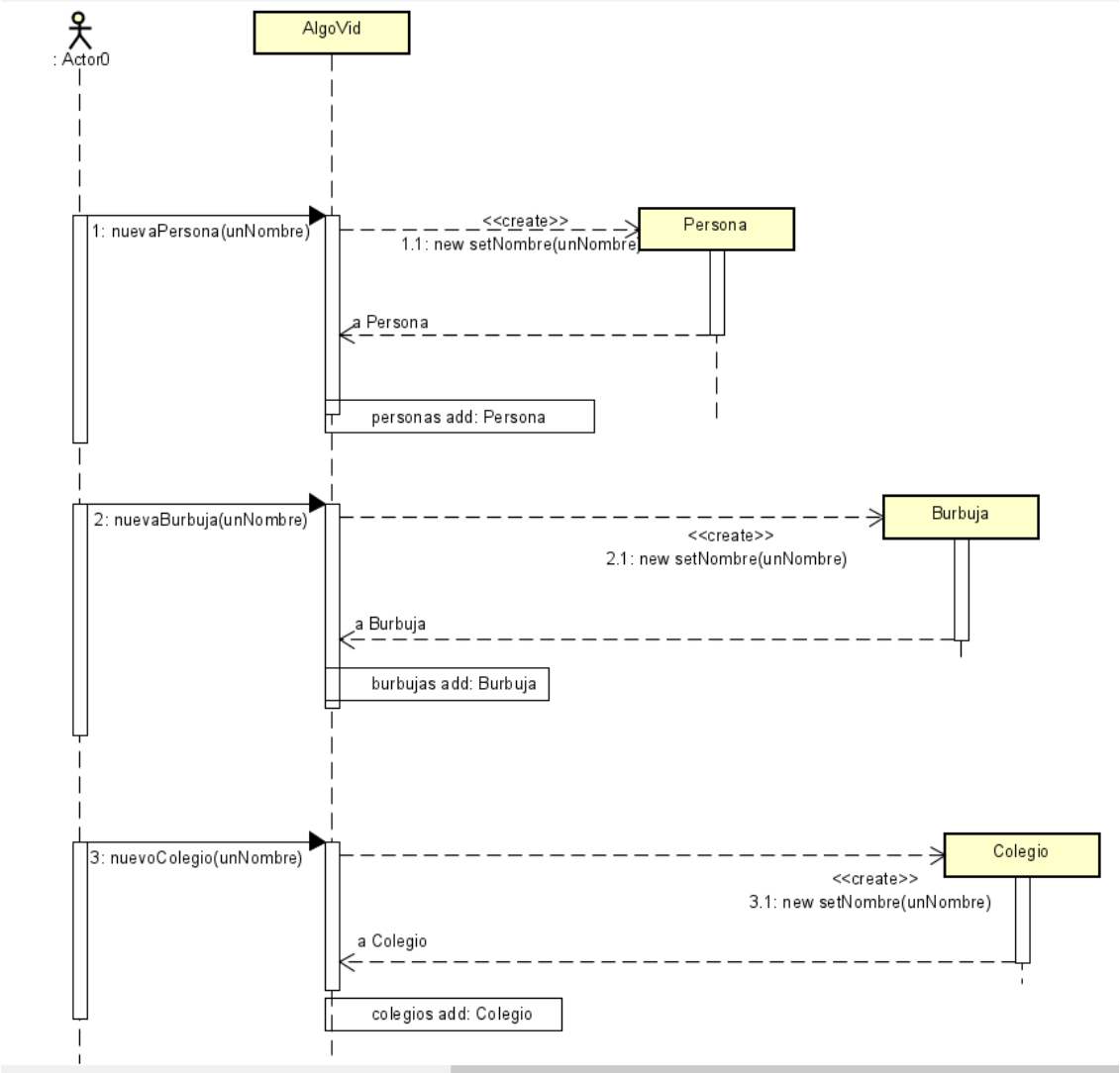

Figura 3: Nueva entidad.

En este diagrama podemos apreciar una típica operación del programa, donde a través de una instancia de Algovid (algovid) se pide a crear una nueva persona, burbuja o colegio. La secuencia comienza al enviarle a una instancia de AlgoVid uno de los siguientes mensajes: nuevaPersona(unNombre), nuevaBurbuja(unNombre) o nuevoColegio(unNombre).

Finalmente, AlgoVid al trabajar con colecciones ordenadas simplemente debería:

```
coleccion add: a Ente
"siendo:
    colecciones: (personas/burbujas/colegios)
    Ente: (a Persona/a Burbuja/a Colegio)"
```

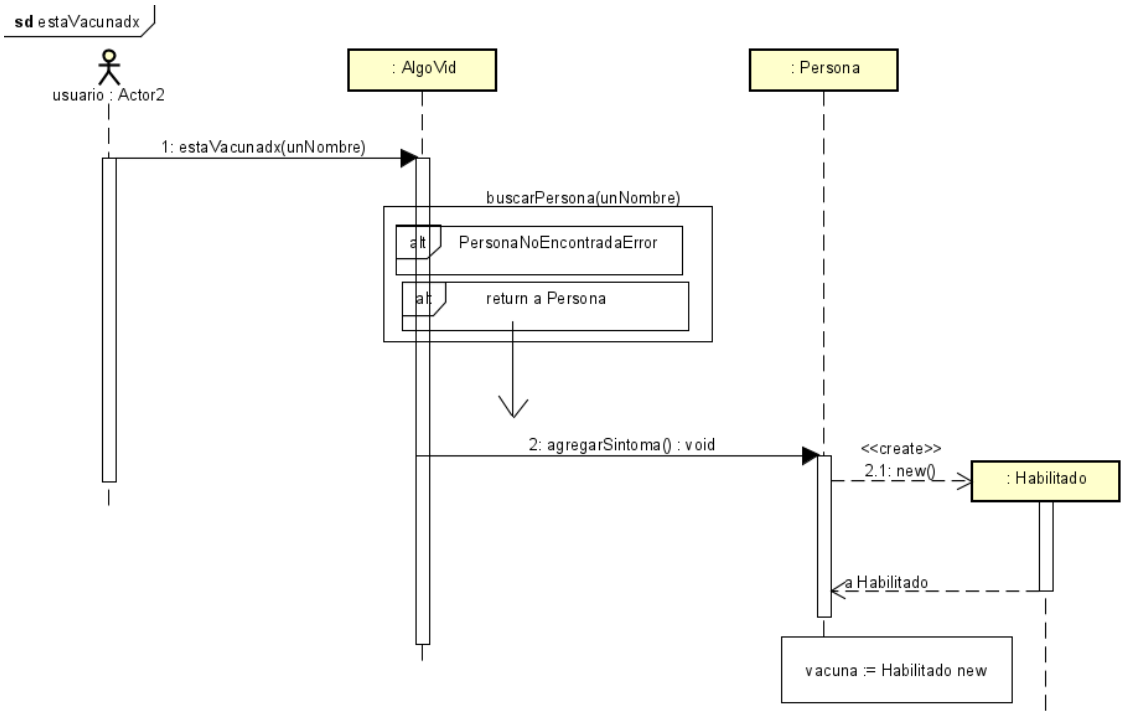

Figura 4: Se vacuna a una persona.

Este diagrama es idéntico para otras condiciones sociales como ser personal esencial y una persona de riesgo. La secuencia comienza al enviarle a una instancia de AlgoVid el mensaje estaVacunadx(unNombre). Notese que la ejecución del programa puede interrumpirse si se da un error de tipo PersonaNoEncontradaError.

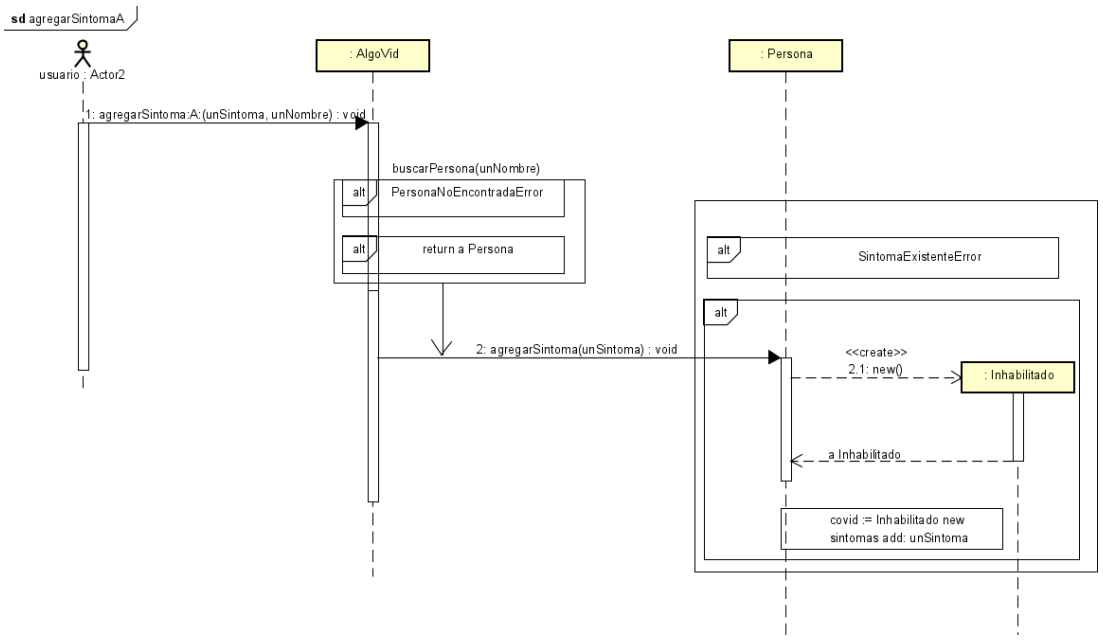

Figura 5: Se le agrega un sintoma a una persona.

La secuencia comienza al enviarle a una instancia de AlgoVid el mensaje agregarSintomaA(unSintoma, unNombre). Notese que la ejecución del programa puede interrumpirse si se da un error de tipo PersonaNoEncontradaError. En caso de que se encuentre, si el sintoma existe dentro de la colección de sintomas de la persona, se devuelve un error de tipo SintomaExistenteError, caso contrario, el string es añadido a la coleccion de sintomas y la condición social Covid de la persona pasa a ser positiva.

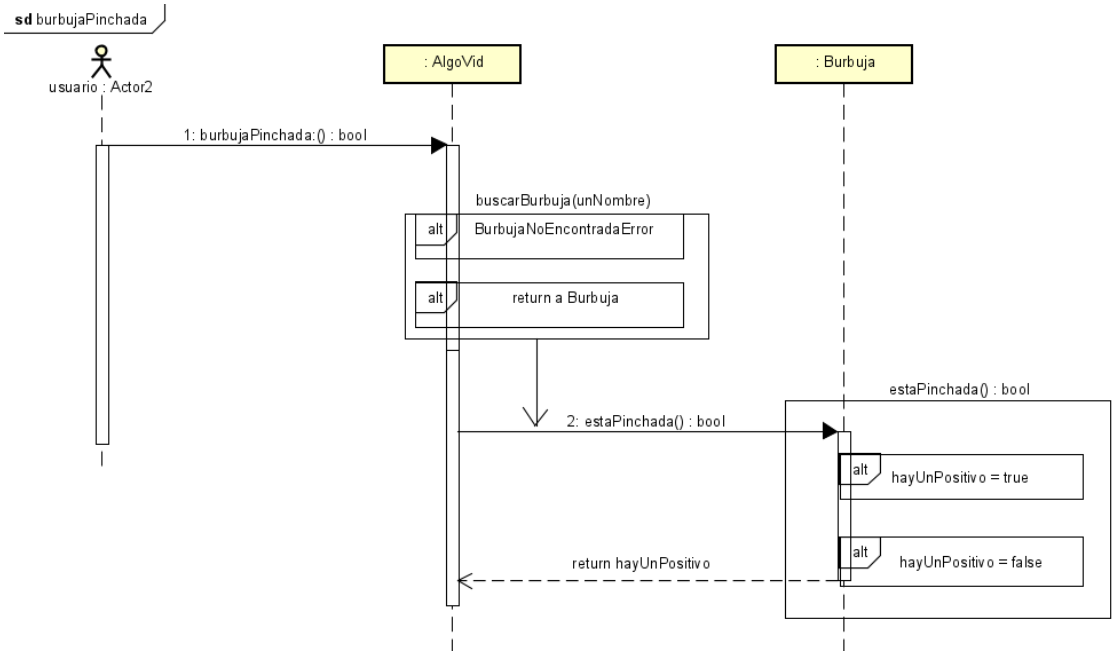

Figura 6: Se consulta si una burbuja está pinchada.

Podemos apreciar el funcionamiento para verificar si una burbuja está pinchada o no al momento de enviar su nombre a AlgoVid. Puede ser que el input sea un nombre inexistente en la coleccion de burbujas, y cortarse el programa con un error de tipo BurbujaNoEncontradaError. Caso contrario, se recorre la colección de personas de burbuja y se colecta los diagnósticos de las personas que contiene. El mensjae estaPinchada() devuelve si dentro de estos diagnosticos se encuentra un caso positivo.

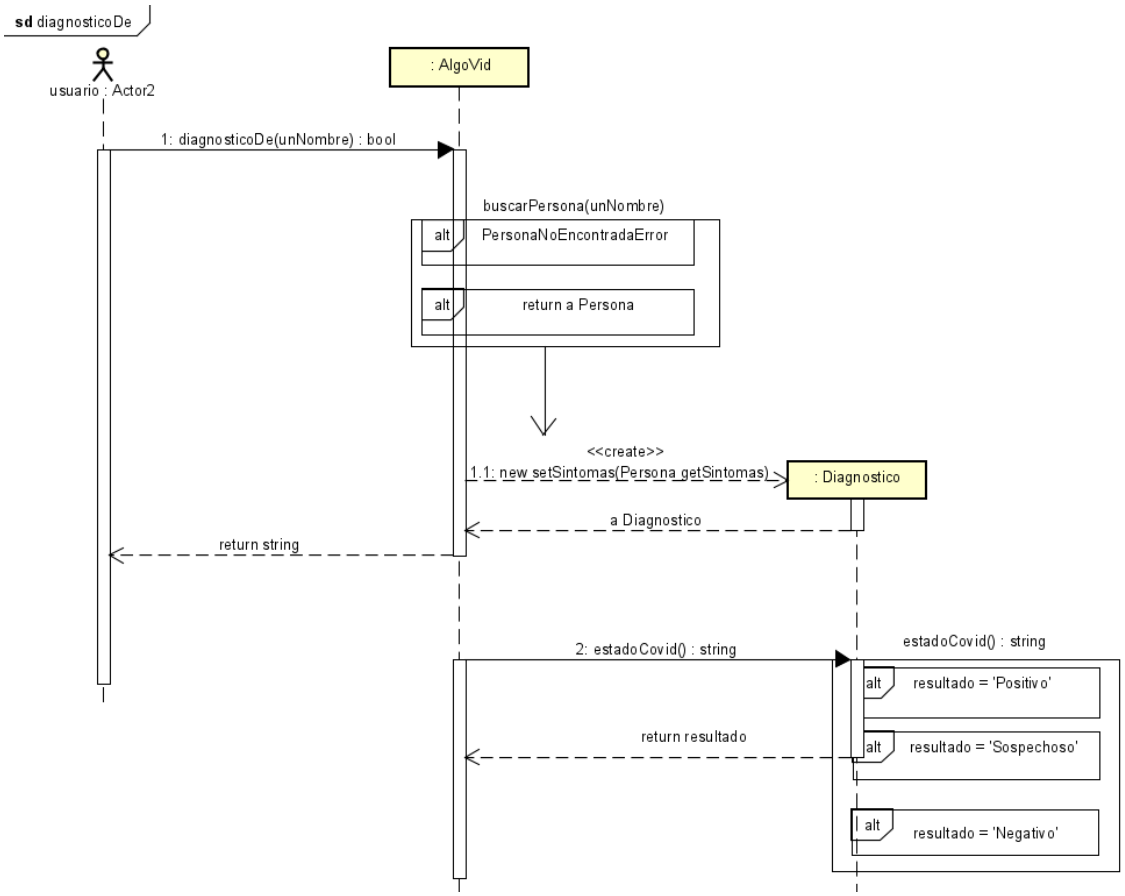

Figura 7: Se consulta el diagnóstico de una persona.

En este diagrama podemos observar cómo se opera para conseguir el diagnóstico médico de una persona dándole su nombre a Algovid. Puede ser que el input sea un nombre inexistente en la coleccion de personas, y cortarse el programa con un error de tipo PersonaNoEncontradaError. Caso contrario, se crea una instancia de Diagnostico con los sintomas de la persona y acto seguido se le envia el mensaje estadoCovid, que devuelve un string cuyo valor varía en 'Positivo' 'Sospechoso' y 'Negativo'.

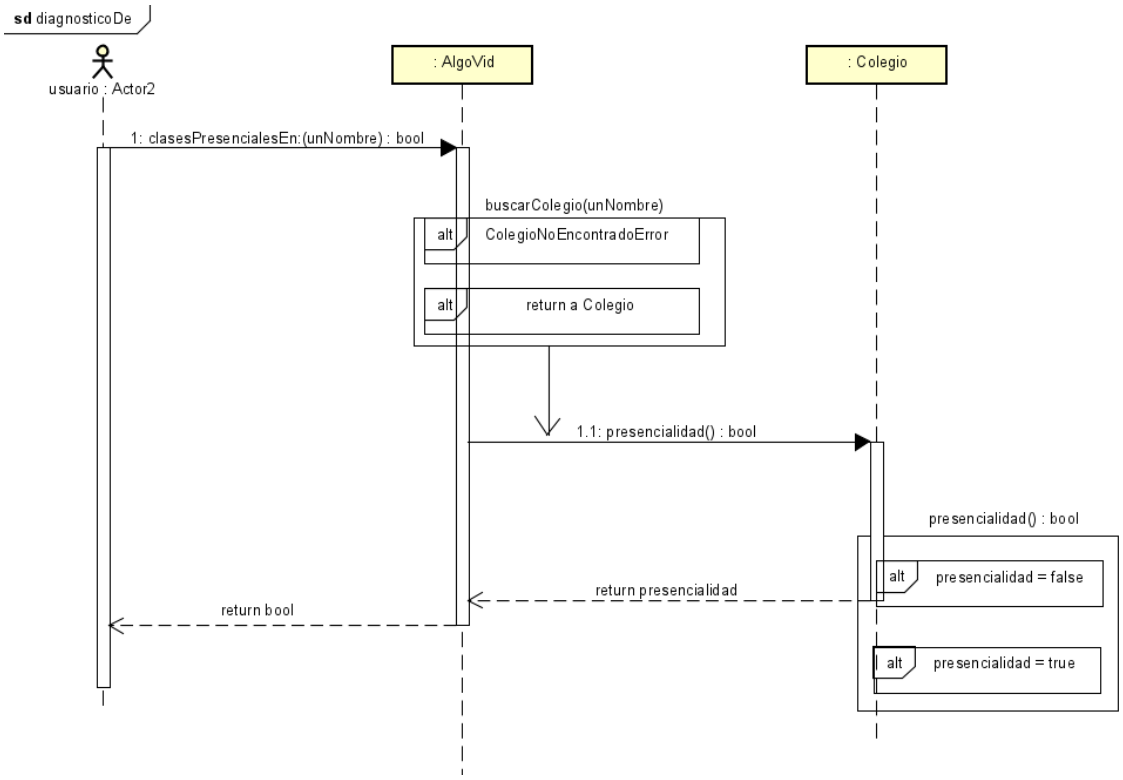

Figura 8: Se consulta si hay clases presenciales en un colegio.

Se ve cómo se opera para saber si un colegio esta habilitado a tener clases presenciales dándole su nombre a Algovid. Puede ser que el input sea un nombre inexistente en la coleccion de colegios, y cortarse el programa con un error de tipo ColegioNoEncontradoError. Caso contrario, se le envía el mensaje presencialidad() a la instancia de Colegio encontrada, y retorna su valor booleano.

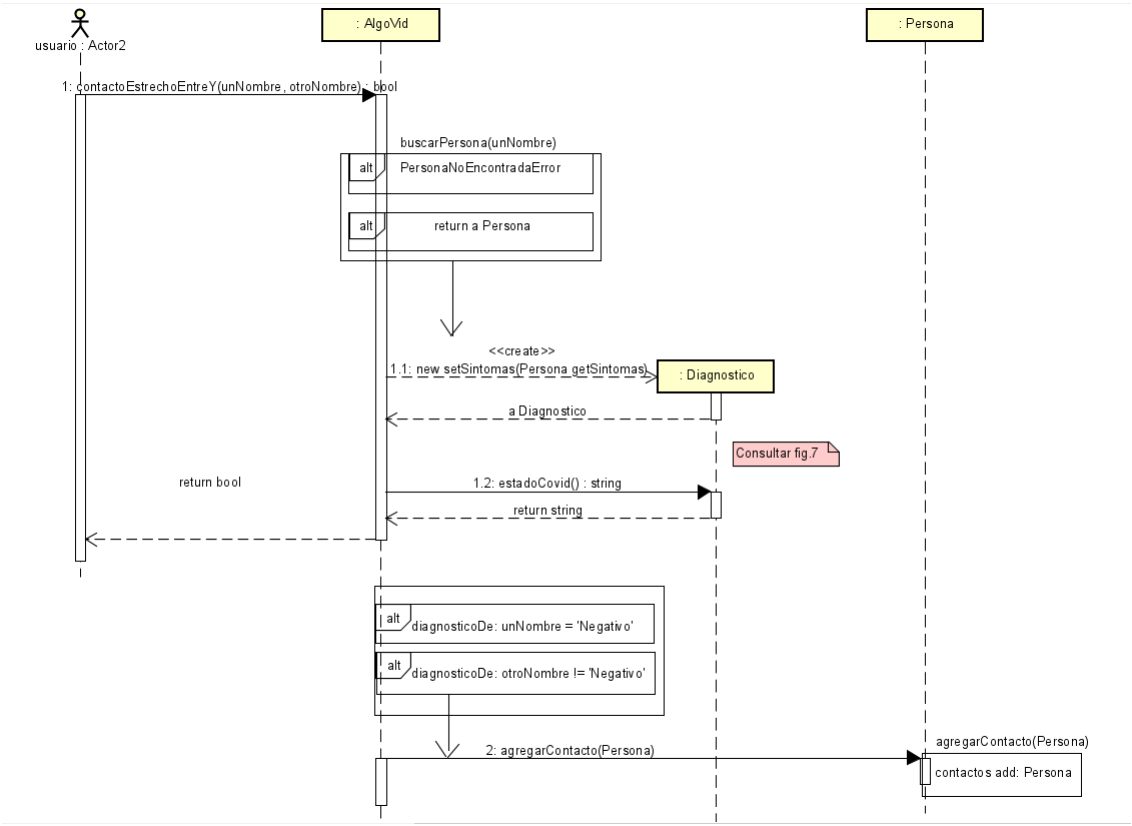

Figura 9: Se procesa el contacto estrecho entre dos personas.

Podemos observar cómo se trata el contacto estrecho entre diferentes instancias de Personas, pasando sus nombres a Algovid. Puede ser que uno de los input sea un nombre inexistente en la coleccion de personas, y cortarse el programa con un error de tipo PersonaNoEncontradaError. Caso contrario, se crea un tipo de dato Diagnostico con los atributos correspondientes a las dos personas, y si el diagnostico de una persona resulta ser 'Positivo' o 'Sospechoso',se agrega esa persona a la coleccion de contactos estrechos de la otra persona.