



Campus Guadalajara

# Evidencia 2. Revisión 2.

**Autores:**

Clay Rodolfo Gutiérrez Herrera - **A01220835**

Luis Alfredo Carmona Martínez - **A01707658**

Luis Fernando Díaz Hernández - **A01639435**

Emilio Hernández Flores - **A01637284**

Dylan Pereyra López - **A01540618**

Alejandro Gutiérrez Zamudio - **A00227610**

**Profesores:**

Iván Axel Dounce Nava

Javier Félix Rendón

Jesús Israel Hernández

Eduardo Morales Vargas

**Modelación de Sistemas Multiagentes con gráficas computacionales - Grupo 302**

18 de Noviembre del 2025

|   |           |
|---|-----------|
| <b>Conformación del equipo de trabajo</b>                               | <b>6</b>  |
| 1. Clay Gutiérrez   | 6         |
| Fortalezas  | 6         |
| Áreas de oportunidad  | 6         |
| Expectativas  | 6         |
| 2. Alfredo Carmona  | 7         |
| Fortalezas  | 7         |
| Áreas de oportunidad  | 7         |
| Expectativas  | 7         |
| 3. Luis Díaz  | 8         |
| Fortalezas  | 8         |
| Áreas de oportunidad  | 8         |
| Expectativas  | 8         |
| 4. Emilio Hernández   | 8         |
| Fortalezas  | 8         |
| Áreas de oportunidad  | 9         |
| Expectativas  | 9         |
| 5. Dylan Pereyra  | 9         |
| Fortalezas  | 9         |
| Áreas de oportunidad  | 9         |
| Expectativas  | 10        |
| 6. Alejandro Gutiérrez  | 10        |
| Fortalezas  | 10        |
| Áreas de oportunidad  | 10        |
| Expectativas  | 10        |
| Expectativas grupales   | 10        |
| Compromisos grupales  | 11        |
| <b>Herramientas colaborativas:</b>                                      | <b>12</b> |
| Github  | 12        |
| Discord (Invitación para los profesores a unirse)                       | 12        |
| <b>Descripción del reto</b>   | <b>12</b> |
| <b>Propuesta general de solución</b>                                    | <b>13</b> |
| <b>Flujo completo del sistema</b>                                       | <b>14</b> |
| 0. Modelo del invernadero (base común)                                  | 14        |
| 1. Planificación inicial (Patólogo)                                     | 15        |
| 2. Misión del ScoutBot  | 15        |
| 3. Descarga en la Base (ScoutBot → Patólogo)                            | 16        |
| 4. Análisis Etapa 1 (Patólogo – screening masivo)                       | 17        |
| 5. Activación del PickBot Enfermero (Etapa 2 – verificación focalizada) | 18        |
| 6. Análisis Etapa 2 (Patólogo – confirmación de diagnóstico)            | 18        |
| 7. (Opcional) Activación del PickBot de sanos                           | 19        |
| 8. Ciclo  | 19        |
| <b>Agentes</b>  | <b>19</b> |

|   |           |
|---|-----------|
| 1. Agente Patólogo Digital (PathologistAgent)                       | 19        |
| 1.1. Identificación del agente y relaciones                         | 19        |
| 1.1.1. Responsabilidades clave:                                     | 20        |
| 1.1.2. Relaciones con otros agentes y elementos:                    | 21        |
| 1.2. Tipo de arquitectura del Patólogo                              | 23        |
| 1.3. Componentes arquitectónicos del Patólogo                       | 24        |
| 1.3.1. Capa Reactiva (componente del agente híbrido)                | 24        |
| 1.3.2. Capa Deliberativa (BDI) del Patólogo                         | 29        |
| 1.3.3. Integración Híbrida  | 33        |
| 2. Agente ScoutBot (Agente Explorador)                              | 34        |
| 2.1. Identificación del agente y relaciones                         | 34        |
| 2.1.1. Responsabilidades clave                                      | 34        |
| 2.1.2. Relaciones con otros agentes y elementos                     | 36        |
| 2.2. Tipo de arquitectura del ScoutBot                              | 37        |
| 2.3. Componentes arquitectónicos del ScoutBot                       | 37        |
| 2.3.1. Capa de Percepción   | 37        |
| 2.3.2. Capa de Control Reactivo                                     | 38        |
| 2.3.3. Capa de Acción   | 39        |
| 2.3.4. Características clave del diseño reactivo                    | 39        |
| 3. PickBot Enfermero (Agente Recolector Especializado)              | 40        |
| 3.1. Identificación del agente y relaciones                         | 40        |
| 3.1.1. Responsabilidades clave                                      | 40        |
| 3.1.2. Relaciones con otros agentes y elementos                     | 41        |
| 3.2. Tipo de arquitectura del PickBot Enfermero                     | 43        |
| 3.3. Componentes arquitectónicos del PickBot Enfermero              | 43        |
| 3.3.1. Capa de Percepción   | 43        |
| 3.3.2. Capa de Control Reactivo                                     | 44        |
| 3.3.3. Capa de Acción   | 45        |
| 3.3.4. Características clave del diseño reactivo                    | 46        |
| 4. Agente PickBot de Sanos (Agente Recolector de Producción Segura) | 46        |
| 4.1. Identificación del agente y relaciones                         | 46        |
| 4.1.1. Responsabilidades clave                                      | 47        |
| 4.1.2. Relaciones con otros agentes y elementos                     | 47        |
| 4.2. Tipo de arquitectura del PickBot de Sanos                      | 49        |
| 4.3. Componentes arquitectónicos del PickBot de Sanos               | 49        |
| 4.3.1. Capa de Percepción   | 49        |
| 4.3.2. Capa de Control Reactivo                                     | 50        |
| 4.3.3. Capa de Acción   | 51        |
| 4.3.4. Características clave del diseño reactivo                    | 52        |
| <b>Diagramas</b>  | <b>53</b> |
| Globales  | 53        |
| Dominio del Invernadero   | 53        |
| Dominio del Invernadero   | 54        |

|  |            |
|--|------------|
| Observaciones, Muestras y Reportes                                     | 55         |
| Agente Patólogo  | 63         |
| Diagrama de clase  | 63         |
| AIP  | 65         |
| Subsistemas  | 66         |
| Estados  | 69         |
| Actividad  | 71         |
| Agente Scoutbot  | 78         |
| Diagrama de clase  | 78         |
| AIP  | 87         |
| Subsistemas  | 93         |
| Estados  | 95         |
| Actividad  | 98         |
| Agente PickBot Sanos   | 101        |
| Diagrama de clase  | 101        |
| AIP  | 120        |
| Subsistemas  | 121        |
| Estados  | 123        |
| Actividad  | 124        |
| Agente PickBot Enfermero   | 127        |
| Diagrama de clase  | 127        |
| AIP  | 145        |
| Subsistemas  | 152        |
| Estados  | 154        |
| Actividad  | 155        |
| <b>Descripción del entorno</b>   | <b>158</b> |
| <b>Plan de trabajo</b>   | <b>160</b> |
| 1. Organización general del plan                                       | 160        |
| 2. Plan de trabajo por semanas   | 160        |
| Semana 1 – Definición y base colaborativa                              | 160        |
| Objetivos principales:   | 160        |
| Actividades:   | 160        |
| Responsables (sugerido):   | 161        |
| Intervalo de esfuerzo estimado (por persona)                           | 161        |
| Semana 2 – Modelo del invernadero y agentes básicos (Primera revisión) | 161        |
| Objetivos principales:   | 161        |
| Actividades planeadas para la primera revisión:                        | 161        |
| Responsables y esfuerzo estimado:                                      | 162        |
| Resultado esperado al final de Semana 2 (primera revisión):            | 162        |
| Semana 3 – Integración del Patólogo y flujo ScoutBot → Patólogo        | 162        |
| Objetivos principales:   | 163        |
| Actividades:   | 163        |
| Responsables:  | 163        |

|   |            |
|---|------------|
| Actividades pendientes al cierre de Semana 3 (esperado):        | 164        |
| Semana 4 – Screening, PickBots y pipeline completo              | 164        |
| Objetivos principales:  | 164        |
| Actividades:  | 164        |
| Responsables:   | 165        |
| Actividades pendientes al cierre de Semana 4 (esperado):        | 165        |
| Semana 5 – Refinamiento, validación y presentación final        | 165        |
| Objetivos principales:  | 165        |
| Actividades:  | 165        |
| Responsables:   | 166        |
| <b>Aprendizaje adquirido</b>                                    | <b>167</b> |
| 1. Coherencia global en un sistema multiagente                  | 167        |
| 2. Valor de separar dominio, agentes y protocolos               | 167        |
| 3. Profundización en el diseño de arquitecturas de agentes      | 167        |
| 4. Comprensión práctica de los AIPs (protocolos de interacción) | 168        |
| 5. Trazabilidad entre requisitos, agentes y comportamientos     | 168        |
| 6. Disciplina de diseño incremental y modular                   | 169        |

# Conformación del equipo de trabajo

## 1. Clay Gutiérrez

### Fortalezas

- Capacidad para diseñar arquitecturas de sistemas complejos.
- Pensamiento estructurado y orientación a la documentación clara y consistente.
- Experiencia previa usando herramientas modernas (GitHub, control de versiones, herramientas colaborativas).
- Facilidad para conectar el problema técnico con el contexto real (operación en campo, restricciones de hardware, comunicación).
- Compromiso alto con la calidad de las entregas y seguimiento de planes de trabajo.

### Áreas de oportunidad

- Tendencia a abarcar demasiado alcance en paralelo; requiere acotar entregables para asegurar versiones funcionales tempranas.
- Necesidad de delegar de forma más explícita para equilibrar carga de trabajo dentro del equipo.
- Mantener tiempos realistas entre detalle técnico y requerimientos académicos del bloque.

### Expectativas

- Consolidar conocimientos en sistemas multiagentes aplicados a problemas reales (detección temprana de enfermedades en agricultura).
- Profundizar en el diseño de arquitecturas híbridas (reactivo + deliberativo/BDI).
- Mejorar la integración entre simulación (Unity), control centralizado y modelos de decisión basados en datos.

## **2. Alfredo Carmona**

### **Fortalezas**

- Habilidad para la comunicación efectiva y la escucha activa, facilitando la colaboración, el intercambio de ideas y la construcción de soluciones en consenso.
- Pensamiento estructurado y alta capacidad de organización, implementando una planificación detallada antes de la ejecución para minimizar imprevistos y optimizar el flujo de trabajo.
- Enfoque pragmático orientado a resultados, priorizando la funcionalidad inicial de las soluciones para después refinarlas y optimizarlas de forma iterativa.
- Capacidad para integrar activamente la retroalimentación y las diversas perspectivas del equipo, fomentando un ambiente colaborativo para construir soluciones consensuadas.

### **Áreas de oportunidad**

- Optimizar el proceso de síntesis de ideas para estructurar y priorizar planes de acción con mayor agilidad.
- Implementar técnicas de priorización y gestión de carga de trabajo para mitigar la confusión y reducir la incidencia del trabajo bajo presión.
- Desarrollar metodologías para la estimación de tiempos y alcance, buscando un balance más preciso entre la ambición del proyecto y los plazos reales.

### **Expectativas**

- Adquirir un dominio práctico del entorno de Unity, con un enfoque específico en la manipulación de cámaras y la implementación de mecánicas de movimiento.
- Explorar y analizar aplicaciones prácticas y casos de uso de los sistemas de agentes en contextos del mundo real.
- Desarrollar una comprensión conceptual sólida de los agentes inteligentes, incluyendo sus modelos de interacción y convivencia.

### **3. Luis Díaz**

#### **Fortalezas**

- Habilidad para expresar ideas técnicas de forma clara, ordenada y comprensible, facilitando la colaboración y la transferencia de conocimiento.
- Compromiso continuo con la mejora de mis conocimientos.
- Participación activa en proyectos colaborativos, asumiendo roles de coordinación y fomentando un ambiente de trabajo digno.
- Priorizar la funcionalidad antes de la optimización y aplicar metodologías sistemáticas en el desarrollo de soluciones.

#### **Áreas de oportunidad**

- Fortalecer la administración de tiempos y prioridades para equilibrar la profundidad técnica con la eficiencia en la entrega de resultados.
- Reforzar la confianza en el propio criterio profesional al momento de tomar decisiones.
- Saber delegar responsabilidades.

#### **Expectativas**

- Aprender los fundamentos del manejo de las cámaras y demás elementos técnicos de Unity.
- Entender e integrar correctamente los conceptos relacionados a los multiagentes y saber aplicarlos en situaciones de la vida real.
- Identificar la mejor arquitectura posible para el diseño al que se quiere llegar.

### **4. Emilio Hernández**

#### **Fortalezas**

- Experiencia en el uso de herramientas colaborativas para el trabajo de diseño de software.
- Experiencia plena en el uso de entornos de desarrollo y simulación 3D como Unity y el uso de scripts y gameobjects para crear entornos creíbles.

- Experiencia anterior con el manejo de topográfica, texturizado, renderizado, UV mapping y animación en entornos de simulacion 3D como Blender.
- Pensamiento estructurado y logico para la resolución de problemas computacionales.

### **Áreas de oportunidad**

- Fortalecer técnicas de manejo de tiempo y división de labor.
- Mejorar hacia una actitud mucho más independiente y proactiva

### **Expectativas**

- Profundizar mi percepción del uso de agentes para la resolución de problemas de alta complejidad / alto nivel de mantenimiento.
- Convertirme proficiente en la automatización de metodologías de resolución autosuficientes.
- Realizar trabajos profundos enfocadas a las estructuras de resolución usando sistemas de multiagentes.

## **5. Dylan Pereyra**

### **Fortalezas**

- Habilidad de poder escuchar a las personas, no solamente escucharlas si no puedo comprenderlas y entender sus emociones
- Pensamiento crítico, para analizar la información objetivamente, evaluar argumentos y resolver problemas de manera lógica
- Tengo experiencia en Unity así como en blender, unity puedo comprender las máquinas de estado y puedo modelar ciertos objetos en blender para usarlos en Unity
- Tengo un poco de experiencia liderando

### **Áreas de oportunidad**

- Trabajo, pero trabajo mucho mejor bajo presión
- Experiencia nula con sistemas multiagentes, pero capaz de entenderlos en un periodo corto de tiempo

## **Expectativas**

- Refinar mis conocimientos en Unity ya que no he trabajado en Unity desde hace tiempo
- Poder comprender a profundidad los conceptos de sistemas multiagentes e implementarlos en el proyecto que se realizará
- Profundizar en las diferentes arquitecturas que se nos presentarán

## **6. Alejandro Gutiérrez**

### **Fortalezas**

- Me gusta el orientado a objetos
- Entiendo lógica de otros rápido
- Experiencia previa de Unity

### **Áreas de oportunidad**

- Mi planeación de tiempo no es muy buena
- No soy muy bueno siendo líder

### **Expectativas**

- Que el reto sea entretenido de desarrollar
- Lograr un buen trabajo en equipo entre todos

### **Expectativas grupales**

- Desarrollar una propuesta formal sólida, coherente y técnicamente defendible para el reto de detección temprana de infección mediante un sistema multiagente.
- Lograr un prototipo funcional en Unity que represente:
  - El invernadero como grafo de segmentos.
  - El comportamiento básico de ScoutBots, PickBot Enfermero y Patólogo Digital, opcional PickBot de sanos.
- Mantener una documentación clara, consistente y alineada con los lineamientos del bloque.

- Repartir responsabilidades de manera equitativa, aprovechando las fortalezas de cada integrante.

## Compromisos grupales

El equipo asume los siguientes compromisos:

- Compromiso de comunicación:
  - Definir y utilizar de forma constante un canal oficial (Discord y repositorio en GitHub).
  - Reportar avances, bloqueos y necesidades de apoyo de forma oportuna.
- Compromiso de organización:
  - Mantener un plan de trabajo actualizado con actividades, responsables y fechas.
  - Respetar acuerdos internos y tiempos límite fijados por el equipo y por el curso.
- Compromiso técnico:
  - Asegurar que todas las decisiones de diseño (agentes, arquitecturas, flujos) se documenten.
  - Mantener la consistencia entre modelo conceptual, simulación en Unity y documentación escrita.
- Compromiso de calidad:
  - Revisar en equipo los entregables antes de su envío.
  - Cuidar coherencia gráfica (tipografía, colores, estilo, diagramas) en todos los documentos.
- Compromiso de aprendizaje:
  - Aprovechar el proyecto como espacio para experimentar con ideas de sistemas multiagentes, no solo para cumplir.
  - Compartir conocimiento entre integrantes para reducir brechas y fortalecer al equipo como unidad.

# Herramientas colaborativas:

## Github

<https://github.com/Clay2605293/MASTomatoes>

## Discord (Invitación para los profesores a unirse)

<https://discord.gg/pvtdnZsYBV>

## Descripción del reto

En la producción de tomate en México, la presencia del Tomato brown rugose fruit virus (ToBRFV) representa una amenaza crítica para la productividad y la seguridad alimentaria. De acuerdo a la información recibida en clase, actualmente, la detección de posibles infecciones depende de inspecciones visuales realizadas por trabajadores en campo. Cuando un recolector identifica síntomas sospechosos, informa al capataz, quien a su vez escala el caso a un especialista o patólogo vegetal. Este proceso es lento, sujeto a error humano y altamente reactivo: cuando se confirma la infección, con frecuencia ya es necesario retirar plantas o secciones completas del invernadero, generando pérdidas económicas significativas.

El reto consiste en diseñar una solución que permita acelerar y sistematizar la detección temprana de posibles infecciones, reduciendo la dependencia exclusiva de la inspección manual y del juicio tardío de expertos humanos. La solución debe ser capaz de:

- Recorrer el invernadero de forma estructurada.
- Capturar información relevante del estado de las plantas y frutos.
- Detectar posibles síntomas de enfermedad con mayor cobertura y trazabilidad.
- Separar los flujos de manejo de frutos sanos, sospechosos y potencialmente infectados.

- Disminuir el tiempo entre la observación inicial y la toma de decisiones informadas.

Todo esto se abordará en un entorno simulado, utilizando un invernadero modelado digitalmente para experimentar con estrategias multiagente sin requerir hardware físico ni datos reales en esta etapa.

## Propuesta general de solución

La propuesta se basa en el desarrollo de un sistema multiagente dentro de un ambiente digital, donde un invernadero simulado en Unity se representa como un grafo de segmentos de cultivo. Sobre este entorno se coordinan distintos tipos de agentes con responsabilidades claras y complementarias.

El núcleo del sistema es el Patólogo Digital (PathologistAgent), un agente híbrido que concentra toda la inteligencia del sistema. Este agente:

- Conoce el modelo completo del invernadero.
- Planifica y asigna misiones de inspección a agentes móviles.
- Procesa las imágenes recibidas.
- Evalúa el estado sanitario de plantas y frutos.
- Coordina la recolección diferenciada de producción sana y muestras sospechosas.
- Genera alertas cuando identifica plantas potencialmente infectadas.

Para la exploración se utilizan ScoutBots, agentes móviles reactivos que ejecutan misiones simples definidas por el Patólogo Digital. Cada ScoutBot recorre segmentos específicos del invernadero, captura imágenes de plantas y frutos según parámetros establecidos, y al finalizar regresa a la base central para descargar la información. Estos agentes no realizan diagnóstico: funcionan como sensores móviles de bajo costo computacional.

La propuesta incluye además dos tipos de agentes recolectores especializados:

- PickBot Enfermero: agente encargado de acudir a plantas con frutos sospechosos, recolectar muestras y llevarlas a la zona de análisis del Patólogo Digital para una evaluación más detallada.
- PickBot de Sanos (opcional): agente dedicado exclusivamente a la recolección de producción en zonas confirmadas como seguras, manteniendo separado el flujo de frutos sanos del manejo de muestras sospechosas.

Con esta arquitectura, el sistema simulado permite:

- Probar estrategias de planificación, cobertura y coordinación multiagente.
- Centralizar el uso de modelos de visión y diagnóstico en un único agente inteligente.
- Representar de forma clara el ciclo completo: inspección → sospecha → verificación → acción.
- Sentar bases conceptuales que, en una etapa futura, podrían integrarse con sensores reales, robots físicos o esquemas de Digital Shadow / Digital Twin.

Esta propuesta formal define el marco conceptual sobre el cual el equipo construirá la simulación, la lógica de los agentes y el plan de trabajo del bloque.

## Flujo completo del sistema

### 0. Modelo del invernadero (base común)

- El invernadero se representa como un grafo de segmentos:
  - Nodos = segmentos de fila / puntos clave (F1\_A, F1\_B, F2\_A, etc.) + Base central.
  - Aristas = pasillos transitables entre esos puntos.
- Cada segmento tiene asociada una lista de plantas, y cada planta puede tener varios frutos.

Este grafo lo conocen todos. El patólogo para calcular las rutas y los demás para recalcular si hay algún obstáculo.

## 1. Planificación inicial (Patólogo)

### 1. El Patólogo:

- Detecta cuántos ScoutBots están disponibles.
- Divide los segmentos del invernadero en subconjuntos balanceados (zonas).
- Asigna a cada ScoutBot una lista ordenada de segmentos a visitar.

### 2. Define también:

- Parámetros de captura (cuántas imágenes por planta / por segmento).
- Reglas básicas de misión (qué hacer si hay bloqueo, cuándo regresar).

## 2. Misión del ScoutBot

Cada ScoutBot ejecuta una misión simple, casi totalmente reactiva:

### 1. Sale de la Base → va al primer segmento asignado.

### 2. En cada segmento:

- Recorre la fila siguiendo un patrón fijo (por ejemplo lineal).
- Para cada planta relevante:

#### ■ Toma 1–3 fotos:

- Vista general de planta.
- Vista donde se observen frutos.
- (Opcional) otra vista si el layout lo requiere.

#### ■ Guarda:

- Imágenes en memoria local.

- Metadata:
  - ID de segmento
  - ID/planta
  - Timestamp.

3. Si detecta un obstáculo físico (maquinaria, personas, etc.) que impide seguir:

- Aplica una regla simple:
  - Marca ese segmento como “no inspeccionado”.
  - Pasa al siguiente segmento de su lista.
- No recalcula rutas globales complejas, solo:
  - “No pude aquí, sigo con mi lista”.

4. Cuando termina su lista (más los reintentos posibles):

- Regresa a la Base central.

Durante la misión normal no manda fotos una por una por la red; solo acumula.

### **3. Descarga en la Base (ScoutBot → Patólogo)**

Al llegar a la Base:

1. El ScoutBot se acopla (dock).
2. Transfiere en lote:
  - Todas las imágenes capturadas.
  - Bitácora de qué segmentos fueron:
    - Inspeccionados con éxito.
    - Saltados por bloqueo.
3. El Patólogo:

- Integra estos datos en su modelo del invernadero.
- Sabe:
  - Qué fue revisado.
  - Qué sigue pendiente.
  - Dónde hay datos para analizar.

#### **4. Análisis Etapa 1 (Patólogo – screening masivo)**

Con las imágenes descargadas:

1. El Patólogo corre su módulo de visión computarizada/diagnóstico (simulado):
  - Detecta frutos en cada imagen.
  - Para cada fruto identificado:
    - Clasifica como:
      - SANO
      - SOSPECHOSO
      - CRÍTICO si se ve grave.
2. Actualiza el mapa de estado:
  - Plantas con solo frutos sanos → zona segura.
  - Plantas con frutos sospechosos y críticos → zona en observación.
3. Genera una lista de objetivos sospechosos:
  - Coordenadas aproximadas
  - Nivel de sospecha
  - Prioridad.

Si detecta segmentos no inspeccionados (por obstáculos o fallos):

- Puede planear una nueva misión con ScoutBots en una iteración posterior (esto ya es decisión del Patólogo que definiremos en su BDI).

## **5. Activación del PickBot Enfermero (Etapa 2 – verificación focalizada)**

Con la lista de frutos sospechosos:

1. El Patólogo ordena al PickBot Enfermero:
  - Visitar esas plantas específicas (usando el grafo para generar rutas eficientes).
2. El PickBot Enfermero:
  - Llega a la planta sospechosa.
  - Simula:
    - Recolección del fruto sospechoso
  - Lleva el fruto a la zona de análisis del Patólogo (en la Base).

## **6. Análisis Etapa 2 (Patólogo – confirmación de diagnóstico)**

El Patólogo realiza una segunda pasada, con una cámara que toma imágenes ultra detalladas:

1. Con el fruto físico:
  - Aplica reglas/modelo más estricto.
2. Decide:
  - Enfermo confirmado:
    - Marca la planta (y potencialmente su entorno) como infectada.
    - Genera alerta a humano (capataz/agricultor).
  - Descartado / baja sospecha:
    - Planta queda como sana o en observación.

3. Actualiza su mapa global:
  - Esto ya es parte central de sus creencias.

## 7. (Opcional) Activación del PickBot de sanos

Con la información consolidada:

1. El Patólogo puede:
  - Mandar el PickBot de sanos a cosechar zonas confirmadas como seguras.
2. Esto refuerza:
  - Separación de flujos: un robot no contamina al otro.
  - Uso eficiente de la información generada por el análisis central.

## 8. Ciclo

- El proceso se repite en ciclos:
  - Nuevas misiones de ScoutBots para nuevas fechas/lotes.
  - Actualización de creencias.
- Siempre con la lógica:
  - ScoutBots = movimiento + captura.
  - Patólogo = análisis, decisión, coordinación.
  - PickBots = ejecución física específica según orden del Patólogo.

# Agentes

## 1. Agente Patólogo Digital (PathologistAgent)

### 1.1. Identificación del agente y relaciones

- **Nombre del agente:** Patólogo Digital (PathologistAgent)

- **Tipo:** Agente central de decisión, coordinación y diagnóstico.
- **Rol en el sistema:** Es el cerebro global del invernadero simulado.

#### **1.1.1. Responsabilidades clave:**

1. Conocer el modelo digital del invernadero:
  - Representado como grafo (nodos = segmentos de filas con plantas asociadas; aristas = pasillos/conexiones).
2. Detectar y gestionar recursos:
  - Saber cuántos ScoutBots están disponibles y su estado.
  - Saber si existen PickBots (sanos) y PickBot Enfermero.
3. Asignar trabajo:
  - Dividir el invernadero en filas y asignarlas a cada ScoutBot.
4. Integrar información:
  - Recibir reportes de cobertura (qué filas fueron inspeccionadas).
  - Recibir reportes de detección: plantas/frutos sanos, sospechosos, críticos.
5. Tomar decisiones sobre acción:
  - Decidir cuándo activar PickBots sanos para recolectar producción segura.
  - Decidir cuándo activar el PickBot Enfermero para recolectar muestras sospechosas.
  - Evaluar las muestras sospechosas (diagnóstico simulado).
  - Marcar plantas como:
    - Sana confirmada
    - Sospechosa en observación

- Enferma confirmada (disparando alerta a humano).

#### **1.1.2. Relaciones con otros agentes y elementos:**

- **Con ScoutBots (Agentes Exploradores)**

- Envía:
  - Misiones iniciales:
    - Listas de segmentos (nodos del grafo) a visitar.
    - Parámetros de captura (número de fotos, etc.).
    - Reglas básicas (qué hacer si hay bloqueo, cuándo regresar).
- Recibe (al regreso a Base):
  - Lote de imágenes capturadas.
  - Bitácora de segmentos inspeccionados y no inspeccionados.
- Relación conceptual:
  - El Patólogo usa a los ScoutBots como sensores móviles del entorno. Los ScoutBots no interpretan; solo capturan y reportan.

- **Con PickBot Enfermero (agente recolector especializado)**

- Envía:
  - Lista de plantas/frutos sospechosos a verificar.
  - Rutas óptimas (en términos de segmentos o puntos objetivo).
- Recibe:
  - Confirmación de recolección de cada fruto sospechoso.
  - Disponibilidad actual del PickBot.
- Relación conceptual:

- El Patólogo decide qué frutos ameritan inspección especializada y usa al PickBot Enfermero como herramienta para traer muestras físicas al módulo de análisis detallado.

- **Con PickBot de sanos (agente opcional)**

- Envía:
  - Tareas de cosecha en zonas confirmadas como seguras.
- Recibe:
  - Confirmación de tareas completadas.
- Relación conceptual:
  - Separa la cadena “segura” de la “sospechosa/enferma”, reduciendo riesgo de contaminación cruzada.

- **Con el Humano (operador/agricultor)**

- Envía:
  - Alertas cuando se confirma infección.
  - Reportes de zonas seguras/no seguras.
- Recibe (opcional como centro de configuración):
  - Parámetros de política (umbrales, prioridades).
  - Confirmaciones de acciones reales (fuera de la simulación).
- Relación conceptual:
  - No se modela como agente; es un actor externo.

- **Con el medio de comunicación**

- El canal no es un agente, es un elemento del sistema.
- El Patólogo define protocolos de mensajes, los transmite y los recibe.

## 1.2. Tipo de arquitectura del Patólogo

El Patólogo Digital se define como un agente híbrido.

Justificación:

- Tiene una capa reactiva:
  - Responde de forma inmediata a eventos:
    - Llegada de un ScoutBot con datos.
    - Llegada de muestras del PickBot Enfermero.
    - Detección de inconsistencias (segmentos no inspeccionados).
  - Aplica reglas simples de actualización de estado.
- Tiene una capa deliberativa (BDI):
  - Mantiene un modelo interno del invernadero.
  - Formula metas (detectar temprano, minimizar riesgo, maximizar rendimiento sano).
  - Selecciona y mantiene planes (intenciones) para:
    - Cobertura del invernadero.
    - Gestión de sospechosos.
    - Uso de recursos (ScoutBots/PickBots).
    - Generación de alertas.
- Combina ambas:
  - La capa reactiva detecta eventos.
  - La capa deliberativa actualiza creencias y ajusta intenciones según esos eventos.

## **1.3. Componentes arquitectónicos del Patólogo**

### **1.3.1. Capa Reactiva (componente del agente híbrido)**

La capa reactiva se encarga de respuestas inmediatas, basadas en reglas simples:

#### **Capa de Percepción**

La capa de percepción del Patólogo Digital se encarga de recibir y estructurar los eventos provenientes del entorno y de los demás agentes. No toma decisiones estratégicas; su función es transformar entradas en información utilizable por las capas superiores.

Fuentes de percepción:

- ScoutBots
  - SCOUT\_MISSION\_COMPLETED(scout\_id, datos\_misión)
    - Incluye:
      - Lista de segmentos inspeccionados.
      - Lista de segmentos no inspeccionados.
      - Referencias a imágenes capturadas por segmento/planta.
- PickBot Enfermero
  - SAMPLE\_DELIVERED(fruto\_id, planta\_id, segmento\_id)
    - Indica que una muestra sospechosa ha sido entregada en la Base.
- PickBot de sanos (opcional)
  - HARVEST\_COMPLETED(zona\_id)
    - Indica finalización de recolección en una zona segura.
- Estado de recursos

- PICKBOT\_AVAILABLE(tipo)
- SCOUT\_AVAILABLE(scout\_id)
- Alertas internas
  - ANOMALY\_IN\_DATA
    - Disparada cuando se detectan inconsistencias en la información recibida (segmentos sin datos, resultados contradictorios, etc.).

Funciones principales de la capa de percepción:

- Validar la estructura de los mensajes recibidos.
- Asociar cada evento a:
  - Segmentos del grafo.
  - Plantas y frutos correspondientes.
  - Agente emisor.
- Entregar estos eventos de forma normalizada a la Capa de Control Reactivo para su procesamiento inmediato.
- Registrar trazas mínimas para depuración y trazabilidad.

### **Capa de Control Reactivo**

La capa de control reactivo implementa las reglas de estímulo–respuesta del Patólogo Digital. Opera sobre los eventos recibidos por la capa de percepción y ejecuta acciones inmediatas sobre el estado interno, sin realizar planificación compleja.

Reglas principales:

- Regla 1: Finalización de misión de ScoutBot
  - Evento: SCOUT\_MISSION\_COMPLETED(scout\_id, datos\_misión)
  - Acciones:

- Marcar al scout\_id como disponible.
  - Registrar segmentos marcados como inspeccionados.
  - Registrar segmentos no inspeccionados como PENDIENTE\_REVISIÓN o NO\_INSPECCIONADO.
  - Activar el disparador para el Análisis Etapa 1 (screening masivo) sobre las imágenes asociadas.
- Regla 2: Actualización de estado de segmento
  - Evento: SEGMENT\_STATUS\_UPDATE(segmento\_id, estado)
  - Acciones:
    - Actualizar en el modelo interno el estado del segmento:
      - INSPECCIONADO, NO\_INSPECCIONADO, BLOQUEADO, PENDIENTE\_REVISIÓN.
- Regla 3: Entrega de muestras sospechosas
  - Evento: SAMPLE\_DELIVERED(fruto\_id, planta\_id, segmento\_id)
  - Acciones:
    - Registrar que el fruto sospechoso está disponible para análisis detallado.
    - Activar el disparador para el Análisis Etapa 2 asociado a esa muestra.
    - Mantener al PickBot Enfermero en estado “en evaluación” o “disponible”, según la configuración.
- Regla 4: Disponibilidad de PickBots
  - Evento: PICKBOT\_AVAILABLE(tipo)
  - Acciones:

- Actualizar la disponibilidad del PickBot correspondiente.
- Dejar preparado este estado para que la capa deliberativa pueda asignar nuevas tareas si existen intenciones pendientes.
- Regla 5: Detección de anomalías en datos
  - Evento: ANOMALY\_IN\_DATA
  - Acciones:
    - Marcar segmentos o zonas afectadas como PENDIENTE\_REVISIÓN.
    - Registrar la necesidad de una futura misión de reinspección.

Características:

- No selecciona estrategias globales; solo mantiene el estado inmediato coherente.
- Actúa como puente entre la percepción y la lógica BDI:
  - Actualiza creencias básicas.
  - Dispara procesos deliberativos cuando corresponde (sin decidir su contenido).

## **Capa de Acción**

La capa de acción del Patólogo Digital agrupa las operaciones concretas que el agente ejecuta como resultado directo de sus reglas reactivas. Estas acciones se reflejan sobre su estado interno y sobre los demás agentes del sistema.

Acciones principales:

- UPDATE\_SEGMENT\_STATUS(segmento\_id, estado)
  - Actualiza el estado de inspección del segmento en el modelo interno.
- REGISTER\_SCOUT\_AVAILABILITY(scout\_id)

- Marca al ScoutBot como disponible para futuras misiones.
- STORE\_MISSION\_DATA(scout\_id, datos\_misión)
  - Asocia imágenes y bitácoras recibidas con segmentos, plantas y frutos.
- TRIGGER\_STAGE1\_ANALYSIS(lote\_imágenes)
  - Inicia el proceso de Análisis Etapa 1 sobre las imágenes descargadas.
  - Esta acción conecta la capa reactiva con la capa deliberativa, que definirá cómo interpretar los resultados.
- REGISTER\_SAMPLE(fruto\_id, planta\_id, segmento\_id)
  - Registra la llegada de una muestra sospechosa para análisis detallado.
- TRIGGER\_STAGE2\_ANALYSIS(fruto\_id, planta\_id)
  - Inicia el proceso de Análisis Etapa 2 sobre una muestra específica.
  - Prepara la información para que la capa deliberativa actualice diagnósticos y tome decisiones.
- UPDATE\_PICKBOT\_STATUS(tipo, estado)
  - Actualiza el estado de los PickBots (disponible, en misión, en espera).
- FLAG REVIEW ZONE(zona\_id)
  - Marca una zona como candidata para reinspección en próximas misiones, cuando se detectan inconsistencias.

Características:

- Todas las acciones son atómicas y de bajo costo computacional.
- No implican por sí mismas decisiones estratégicas; habilitan información y condiciones para que la capa deliberativa (BDI) formule y ajuste intenciones.

- Garantizan que el estado interno del Patólogo esté siempre sincronizado con los eventos del entorno y de los demás agentes.

### **1.3.2. Capa Deliberativa (BDI) del Patólogo**

En la parte superior del agente híbrido, el Patólogo funciona como un agente BDI (Beliefs–Desires–Intentions).

#### **Creencias (Beliefs)**

El Patólogo mantiene un modelo interno que incluye:

1. Modelo del invernadero:
  - Grafo  $G(V, E)$ :
    - $V$ : Base central, segmentos de filas.
    - $E$ : pasillos transitables.
  - Para cada segmento:
    - Lista de plantas.
    - Para cada planta:
      - IDs de frutos asociados.
2. Estado de inspección:
  - Por segmento:
    - NO\_INSPECCIONADO, INSPECCIONADO, BLOQUEADO, PENDIENTE\_REVISIÓN.
  - Por planta/fruto (screening):
    - DESCONOCIDO, SANO, SOSPECHOSO, CRÍTICO.
3. Recursos:
  - ScoutBots:

- Disponibles / en misión / fallando.
- Última zona asignada.
- PickBot Enfermero:
  - Disponibilidad, posición (a nivel de segmento).
- PickBot de sanos (opcional):
  - Disponibilidad, carga de trabajo.

#### 4. Resultados de análisis:

- Etapa 1 (screening masivo):
  - Lista de frutos sospechosos por planta/segmento.
- Etapa 2 (confirmación):
  - Lista de frutos analizados en detalle.
  - Plantas marcadas como:
    - ENFERMO\_CONFIRMADO
    - EN\_OBSERVACIÓN
    - DESCARTADO.

#### 5. Políticas/umbrales:

- Probabilidad mínima para marcar como sospechoso.
- Probabilidad mínima para marcar como enfermo confirmado.
- Reglas para decidir cuándo generar alerta a humano.
- Reglas para replanear inspecciones en segmentos bloqueados/no inspeccionados.

### Deseos (Desires)

Objetivos de alto nivel que guían el comportamiento del Patólogo:

1. D1: Garantizar cobertura completa o suficiente del invernadero.
  - Minimizar segmentos sin inspeccionar.
2. D2: Detectar tempranamente posibles infecciones.
  - Identificar frutos/plants sospechosos lo antes posible.
3. D3: Reducir falsos negativos críticos.
  - Evitar dejar pasar una planta infectada como sana.
4. D4: Minimizar impacto productivo.
  - Solo activar medidas fuertes (alerta/aislamiento) cuando haya evidencia suficiente.
  - Aprovechar los datos para identificar zonas seguras para cosecha.
5. D5: Optimizar uso de recursos.
  - No desperdiciar ScoutBots ni PickBots.
  - Reducir recorridos innecesarios.

### **Intenciones (Intentions)**

Las intenciones son planes concretos que el Patólogo adopta basándose en sus creencias y deseos.

Principales intenciones:

1. I1: Planificación de misiones de ScoutBots
  - Plan:
    - Dividir los segmentos del grafo en zonas balanceadas.
    - Generar listas ordenadas de segmentos para cada ScoutBot.
    - Asignar misiones desde la Base.
  - Se mantiene mientras haya áreas por inspeccionar.

## 2. I2: Análisis Etapa 1 (screening masivo)

- Plan:
  - Procesar todas las imágenes recibidas en lote.
  - Detectar frutos y clasificarlos (SANO/SOSPECHOSO/CRÍTICO).
  - Actualizar mapa de salud.
  - Construir lista de objetivos sospechosos.
- Se activa tras cada descarga de ScoutBots.

## 3. I3: Gestión de sospechosos con PickBot Enfermero

- Plan:
  - Tomar la lista de frutos sospechosos.
  - Ordenarlos por prioridad (nivel de sospecha, cercanía, densidad).
  - Asignar al PickBot Enfermero una ruta eficiente.
  - Esperar muestras y lanzar Etapa 2.
- Iterativo y condicionado por disponibilidad del PickBot.

## 4. I4: Análisis Etapa 2 (confirmación)

- Plan:
  - Evaluar frutos traídos por el PickBot Enfermero con cámara ultra detallada.
  - Si se confirma infección:
    - Marcar planta y zona como infectadas.
    - Generar alerta al humano.

- Si se descarta:

- Actualizar estado a “sano” o “en observación”.

- Busca cumplir D2 y D3.

## 5. I5: Reinspección / Replanificación

- Plan:

- Detectar segmentos NO\_INSPECCIONADO o BLOQUEADO.

- Preparar nuevas misiones para próximos ciclos.

- Mantiene la cobertura (D1).

## 6. I6: Activación del PickBot de sanos (opcional)

- Plan:

- Identificar zonas marcadas como seguras.

- Asignar rutas de cosecha a PickBot sanos.

- Alineado con D4 y D5.

Cada intención puede ser reevaluada si cambian las creencias:

- Si aparece un nuevo brote grave, la prioridad puede pasar a confirmación rápida y alertas.

### 1.3.3. Integración Híbrida

Resumiendo cómo se combinan componentes:

#### 1. La capa reactiva:

- Escucha eventos (llegan datos, llegan muestras, cambios de disponibilidad).
  - Actualiza creencias inmediatas.
  - Dispara o ajusta planes.

## 2. La capa deliberativa BDI:

- Sobre esas creencias:
  - Decide qué intención activar o modificar:
    - ¿Nuevo ciclo de scouts?
    - ¿Envío del enfermero?
    - ¿Generar alerta?
- Ejerce control de alto nivel sobre ScoutBots y PickBots.

En resumen, el Patólogo queda definido como:

- Agente híbrido (reactivo + BDI).
- Centro de coordinación.
- Punto de integración de toda la información del invernadero.
- Responsable de las decisiones críticas del sistema.

## 2. Agente ScoutBot (Agente Explorador)

### 2.1. Identificación del agente y relaciones

- **Nombre del agente:** ScoutBot (Agente Explorador)
- **Tipo:** Agente móvil de exploración y captura de información.
- **Rol en el sistema:** Actúa como sensor móvil del invernadero; recorre segmentos asignados, captura imágenes de plantas y frutos, y reporta la información al Patólogo Digital al finalizar su misión.

#### 2.1.1. Responsabilidades clave

- Ejecutar misiones definidas por el Patólogo Digital:
  - Recibir una lista ordenada de segmentos (nodos del grafo) a inspeccionar.

- Desplazarse físicamente (en la simulación) a través de esos segmentos.
- Capturar información visual:
  - En cada planta relevante dentro del segmento:
    - Tomar de 1 a 3 imágenes:
      - Vista general de la planta.
      - Vista donde se observen frutos.
      - (Opcional) vista adicional según configuración definida por el Patólogo.
  - Asociar a cada imagen:
    - ID de segmento.
    - ID de planta.
    - Timestamp.
- Manejo local de obstáculos:
  - Si detecta un obstáculo físico que impide inspeccionar un segmento:
    - Marca dicho segmento como “no inspeccionado”.
    - Continúa con el siguiente segmento de su lista.
- Gestión de datos:
  - Almacenar temporalmente en memoria local:
    - Imágenes capturadas.
    - Bitácora de segmentos inspeccionados y no inspeccionados.
  - Regresar a la Base central al finalizar la misión.
  - Transferir en lote toda la información al Patólogo Digital.

### **2.1.2. Relaciones con otros agentes y elementos**

- **Con el Patólogo Digital (PathologistAgent)**

- Recibe:
  - Misiones iniciales:
    - Lista de segmentos a recorrer.
    - Parámetros de captura de imágenes.
    - Reglas de comportamiento ante bloqueo.
  - Puntos de retorno a Base.
- Envía (al regresar a Base):
  - Conjunto de imágenes capturadas.
  - Bitácora de misión:
    - Segmentos inspeccionados con éxito.
    - Segmentos marcados como no inspeccionados por bloqueo.
- Relación conceptual:
  - El ScoutBot es un ejecutor reactivo de la estrategia definida por el Patólogo; no interpreta la información, solo la captura y la entrega.

- **Con PickBot Enfermero y PickBot de sanos**

- No existe coordinación directa.
- La interacción es indirecta, mediada por el Patólogo Digital, quien decide las acciones de los PickBots en función de los datos proporcionados por los ScoutBots.

- **Con el medio de comunicación y la Base central**

- Utiliza la Base como punto de sincronización:
  - Inicio de misión.
  - Fin de misión y descarga de datos.

## **2.2. Tipo de arquitectura del ScoutBot**

El ScoutBot se define como un agente reactivo.

Justificación:

- No mantiene un modelo interno complejo del invernadero.
- No realiza diagnóstico ni planificación global.
- Responde a estímulos y reglas simples:
  - Seguir la lista de segmentos recibida.
  - Capturar imágenes según parámetros fijados.
  - Saltar segmentos bloqueados.
  - Regresar a Base al completar su misión.
- Toda la “inteligencia pesada” (análisis, priorización, diagnóstico, replanificación global) reside en el Patólogo Digital.

## **2.3. Componentes arquitectónicos del ScoutBot**

### **2.3.1. Capa de Percepción**

- Sensores simulados (a nivel conceptual):
  - Posición dentro del invernadero (conocer el segmento actual).
  - Detección de presencia de planta en la posición indicada.
  - Detección de obstáculo en el camino (maquinaria, personas, bloqueo temporal).
  - Confirmación de llegada a Base central.

- Entrada de configuración:
  - Misión recibida desde el Patólogo:
    - Lista ordenada de segmentos.
    - Parámetros de captura (número de fotos por planta).

### **2.3.2. Capa de Control Reactivo**

Basada en reglas simples de estímulo-respuesta:

- Regla 1: Inicio de misión
  - Si se recibe una misión y el ScoutBot está en Base:
    - Cargar lista de segmentos.
    - Cambiar estado a “en misión”.
    - Ir al primer segmento asignado.
- Regla 2: Llegada a segmento asignado
  - Si se llega a un segmento:
    - Recorrer la fila con un patrón predefinido.
    - Para cada planta marcada como relevante:
      - Capturar las imágenes configuradas.
      - Guardar imagen + metadata en memoria local.
- Regla 3: Obstáculo detectado en segmento
  - Si durante el recorrido de un segmento se detecta un obstáculo que impide la inspección adecuada:
    - Marcar el segmento como “no inspeccionado”.
    - Registrar el evento en la bitácora interna.
    - Continuar con el siguiente segmento de la lista.

- No recalcular rutas globales; sólo seguir el orden definido.
- Regla 4: Fin de lista de segmentos
  - Si no hay más segmentos pendientes en la misión:
    - Cambiar estado a “regreso a Base”.
    - Navegar desde la posición actual hasta la Base central.
- Regla 5: Llegada a Base
  - Si llega a la Base y está en estado “regreso a Base”:
    - Iniciar transferencia de datos al Patólogo:
      - Imágenes capturadas.
      - Bitácora de segmentos inspeccionados/no inspeccionados.
    - Cambiar estado a “disponible”.
    - Esperar nueva misión.

### **2.3.3. Capa de Acción**

- Acciones principales:
  - MOVE\_TO(segmento): desplazarse al segmento indicado.
  - SCAN\_PLANT(planta\_id): capturar imágenes según parámetros.
  - MARK\_UNINSPECTED(segmento): marcar un segmento como no inspeccionado por bloqueo.
  - RETURN\_TO\_BASE(): regresar a la Base central.
  - UPLOAD\_DATA(): enviar al Patólogo las imágenes y la bitácora.

### **2.3.4. Características clave del diseño reactivo**

1. No realiza inferencias sobre salud de plantas o frutos.

2. No gestiona prioridades globales ni redistribuye segmentos.
3. No coordina misiones con otros ScoutBots directamente.
4. Su comportamiento completo se puede describir como:
  - a. "Seguir misión → Capturar → Registrar → Regresar → Descargar".

En conjunto, los ScoutBots forman una capa distribuida de percepción simple, controlada y explotada inteligentemente por el agente Patólogo Digital.

### **3. PickBot Enfermero (Agente Recolector Especializado)**

#### **3.1. Identificación del agente y relaciones**

**Nombre del agente:** PickBot Enfermero

**Tipo:** Agente móvil recolector especializado para muestras sospechosas.

**Rol en el sistema:** Ejecutar misiones de recolección focalizada de frutos sospechosos identificados por el Patólogo Digital, trasladándolos a la zona de análisis para su evaluación detallada.

##### **3.1.1. Responsabilidades clave**

- Ejecutar órdenes emitidas por el Patólogo Digital:
  - Recibir una lista de objetivos sospechosos:
    - Plantas y frutos específicos.
    - Ubicaciones expresadas en términos de segmentos del grafo.
  - Desplazarse hasta dichos objetivos siguiendo rutas definidas o derivadas del modelo del invernadero.
- Recolección especializada:
  - En cada planta objetivo:
    - Localizar el fruto sospechoso indicado.
    - Simular la recolección del fruto sospechoso.

- Asegurar su transporte controlado hacia la Base central para evitar contaminación de otras zonas.
- Entrega y confirmación:
  - Depositar los frutos sospechosos en la zona de análisis del Patólogo Digital.
  - Reportar la lista de objetivos atendidos y aquellos que no pudieron ser recolectados (por bloqueo u otra causa).
- Gestión básica de estado:
  - Indicar al Patólogo:
    - Inicio de misión.
    - Finalización de misión.
    - Disponibilidad para nuevas tareas.

### **3.1.2. Relaciones con otros agentes y elementos**

- Con el Patólogo Digital (**PathologistAgent**)
  - Recibe:
    - Misiones de recolección:
      - Lista de frutos sospechosos (planta, segmento, identificador).
      - Orden sugerido o ruta (o puntos a visitar sobre el grafo).
    - Instrucciones operativas específicas (por ejemplo: prioridad de ciertos casos críticos).
  - Envía:
    - Confirmaciones de recolección:
      - Fruto\_id / planta\_id recolectado.

- Segmentos atendidos.
- Notificaciones de imposibilidad de acceso:
  - Bloqueos o problemas encontrados.
- Señales de “misión completada” y “disponible”.
- Relación conceptual:
  - El PickBot Enfermero es un ejecutor especializado y reactivo de las decisiones del Patólogo, enfocado en la etapa de verificación y diagnóstico de alta precisión.
- **Con ScoutBots**
  - No existe comunicación directa.
  - La relación es indirecta:
    - Los ScoutBots generan los datos de sospecha.
    - El Patólogo utiliza esos datos para asignar misiones al PickBot Enfermero.
- **Con PickBot de sanos**
  - No existe coordinación directa.
  - Ambos actúan bajo las órdenes del Patólogo para mantener separadas las cadenas de manejo de frutos sanos y sospechosos.
- **Con la Base central**
  - Utiliza la Base como:
    - Punto de partida y regreso.
    - Punto de entrega de muestras sospechosas para análisis detallado.

- La comunicación y transferencia de información se realiza a través de los mecanismos definidos por el Patólogo y el sistema.

### **3.2. Tipo de arquitectura del PickBot Enfermero**

El PickBot Enfermero se define como un agente reactivo.

Justificación:

- No mantiene un modelo completo del invernadero.
- No realiza diagnóstico ni toma decisiones estratégicas.
- Su comportamiento se basa en seguir misiones, reaccionar a objetivos e informar resultados.
- Aplica reglas simples:
  - Ir al objetivo indicado.
  - Recolectar fruto sospechoso.
  - Regresar a Base y entregar.
  - Reportar disponibilidad o incidencias.

### **3.3. Componentes arquitectónicos del PickBot Enfermero**

#### **3.3.1. Capa de Percepción**

- Sensores simulados (conceptuales):
  - Posición actual dentro del invernadero.
  - Detección de llegada a un segmento objetivo.
  - Detección de la planta objetivo dentro del segmento.
  - Confirmación de interacción con el fruto sospechoso (recolección).
  - Detección de obstáculos que impidan el acceso a un objetivo.
  - Confirmación de llegada a la Base central.

- Entrada de configuración:
  - Lista de objetivos sospechosos enviada por el Patólogo:
    - Segmento, planta, fruto\_id.
  - Parámetros de manipulación segura definidos por el sistema.

### **3.3.2. Capa de Control Reactivo**

Conjunto de reglas de estímulo–respuesta que definen su comportamiento:

- Regla 1: Inicio de misión
  - Si el PickBot Enfermero está en Base y recibe una lista de objetivos:
    - Cargar lista de objetivos.
    - Cambiar estado a “en misión”.
    - Ir al primer objetivo de la lista.
- Regla 2: Llegada a segmento objetivo
  - Si se alcanza el segmento asociado a un objetivo:
    - Localizar la planta indicada.
    - Posicionarse junto al fruto sospechoso.
- Regla 3: Recolección de fruto sospechoso
  - Si el fruto sospechoso es accesible:
    - Simular PICK\_FRUIT(fruto\_id).
    - Marcar objetivo como “recolectado”.
    - Almacenar el fruto en el contenedor seguro del PickBot.
  - Si el acceso está bloqueado:
    - Marcar objetivo como “no recolectado por bloqueo”.

- Registrar el incidente en la bitácora interna.
- Regla 4: Siguiente objetivo
  - Si existen objetivos pendientes:
    - Seleccionar el siguiente objetivo de la lista.
    - Desplazarse hacia su segmento correspondiente.
  - Si no hay más objetivos pendientes:
    - Cambiar estado a “regreso a Base”.
    - Ejecutar RETURN\_TO\_BASE().
- Regla 5: Llegada a Base
  - Si llega a la Base con estado “regreso a Base”:
    - Ejecutar DELIVER\_SAMPLES():
      - Entregar todos los frutos sospechosos recolectados al área de análisis del Patólogo.
    - Enviar reporte:
      - Lista de objetivos recolectados.
      - Lista de objetivos no atendidos y motivos.
    - Cambiar estado a “disponible”.
    - Esperar nuevas instrucciones.

### **3.3.3. Capa de Acción**

Acciones principales del PickBot Enfermero:

- MOVE\_TO(segmento): desplazarse hacia el segmento objetivo.
- APPROACH\_PLANT(planta\_id): posicionarse junto a la planta indicada.
- PICK\_FRUIT(fruto\_id): simular la recolección del fruto sospechoso.

- MARK\_UNREACHABLE(objetivo\_id): registrar que el objetivo no pudo ser atendido.
- RETURN\_TO\_BASE(): regresar a la Base central al finalizar la misión.
- DELIVER\_SAMPLES(): transferir las muestras sospechosas al módulo de análisis del Patólogo.
- REPORT\_STATUS(): informar al Patólogo el resultado de la misión y la disponibilidad actual.

### **3.3.4. Características clave del diseño reactivo**

- Operación basada en misiones definidas externamente por el Patólogo Digital.
- Lógica centrada en cumplir objetivos específicos sin modificar la estrategia global.
- No ejecuta análisis de salud, ni reorganiza prioridades de manera autónoma.
- Su función principal es cerrar el ciclo de verificación: transportar información física (frutos sospechosos) desde el campo hasta el punto de diagnóstico detallado.

Con este diseño, el PickBot Enfermero complementa el trabajo de los ScoutBots y del Patólogo Digital, asegurando una segunda etapa de revisión precisa y controlada para la detección del Tomato brown rugose fruit virus dentro del entorno simulado.

## **4. Agente PickBot de Sanos (Agente Recolector de Producción Segura)**

### **4.1. Identificación del agente y relaciones**

**Nombre del agente:** PickBot de Sanos

**Tipo:** Agente móvil recolector de frutos/planta en zonas confirmadas como seguras.

**Rol en el sistema:** Ejecutar misiones de cosecha en áreas clasificadas como no infectadas por el Patólogo Digital, manteniendo separado el flujo de producción sana del flujo de muestras sospechosas o infectadas.

#### **4.1.1. Responsabilidades clave**

- Ejecutar órdenes emitidas por el Patólogo Digital:
  - Recibir listas de segmentos, plantas o zonas marcadas como seguras.
  - Desplazarse a dichas zonas siguiendo rutas basadas en el grafo del invernadero.
- Recolección de producción segura:
  - En cada planta o zona segura asignada:
    - Simular la recolección de frutos o la cosecha correspondiente.
    - Garantizar que solo se manipule producción previamente confirmada como sana.
- Entrega y confirmación:
  - Transportar la producción sana al punto designado (Base central/área de acopio).
  - Reportar al Patólogo Digital:
    - Zonas atendidas.
    - Cantidad o estado de la cosecha (a nivel conceptual).
- Gestión básica de estado:
  - Indicar inicio de misión, finalización y disponibilidad para nuevas tareas.

#### **4.1.2. Relaciones con otros agentes y elementos**

- Con el Patólogo Digital (**PathologistAgent**)

- Recibe:
    - Misiones de cosecha:
      - Listas de segmentos o plantas marcadas como seguras.
      - Orden o prioridades sugeridas.
    - Instrucciones operativas:
      - Ventanas de tiempo, restricciones, etc. (si se modelan).
  - Envía:
    - Confirmaciones de recolección en zonas seguras.
    - Notificación de zonas inaccesibles por bloqueo u otra causa.
    - Indicaciones de finalización de misión y estado de disponibilidad.
  - Relación conceptual:
    - El PickBot de Sanos es un ejecutor directo de la estrategia de aprovechamiento productivo definida por el Patólogo, sin intervenir en diagnóstico ni toma de decisiones.
- **Con PickBot Enfermero**
    - No existe comunicación directa.
    - La separación física y funcional entre ambos agentes reduce la probabilidad de contaminación cruzada:
      - PickBot de Sanos opera solo en áreas seguras.
      - PickBot Enfermero manipula únicamente frutos sospechosos.
  - **Con ScoutBots**
    - Relación indirecta:

- Las decisiones sobre qué zonas son seguras se basan en la información capturada por los ScoutBots y procesada por el Patólogo.
- **Con la Base central**
  - Utiliza la Base como:
    - Punto de inicio de misión.
    - Punto de entrega de producción sana.
    - Punto de sincronización con el Patólogo Digital.

#### **4.2. Tipo de arquitectura del PickBot de Sanos**

El PickBot de Sanos se define como un agente reactivo.

Justificación:

- No mantiene un modelo completo del invernadero.
- No realiza análisis de estado sanitario.
- Su comportamiento está guiado por:
  - Misiones concretas emitidas por el Patólogo.
  - Reglas simples de navegación y recolección.
- La lógica estratégica (qué cosechar, cuándo, en qué orden global) reside completamente en el Patólogo Digital.

#### **4.3. Componentes arquitectónicos del PickBot de Sanos**

##### **4.3.1. Capa de Percepción**

- Sensores simulados:
  - Posición actual dentro del invernadero.
  - Detección de llegada a un segmento o planta objetivo.

- Confirmación de recolección de frutos/planta.
- Detección de obstáculos que impidan el acceso a una zona segura.
- Confirmación de llegada a la Base o punto de acopio.
- Entrada de configuración:
  - Lista de zonas/segmentos/planta seguras enviada por el Patólogo.
  - Parámetros de operación:
    - Forma de recolección.
    - Cantidad máxima por misión (si se modela).

#### **4.3.2. Capa de Control Reactivo**

Reglas de estímulo–respuesta que definen su comportamiento:

- Regla 1: Inicio de misión
  - Si el PickBot de Sanos está en Base y recibe una lista de zonas seguras:
    - Cargar la lista de objetivos.
    - Cambiar estado a “en misión”.
    - Ir al primer segmento o planta de la lista.
- Regla 2: Llegada a zona segura
  - Si se alcanza un segmento o planta marcada como segura:
    - Ejecutar HARVEST\_SAFE():
      - Simular la recolección de frutos/planta definidos como sanos.
      - Registrar la zona como atendida.
- Regla 3: Obstáculo en zona segura

- Si la zona segura no es accesible por bloqueo:
  - Marcar objetivo como “no atendido por bloqueo”.
  - Registrar el incidente en la bitácora.
  - Continuar con el siguiente objetivo.
- Regla 4: Siguiente objetivo
  - Si existen objetivos pendientes en la misión:
    - Seleccionar el siguiente objetivo de la lista.
    - Desplazarse al segmento/planta correspondiente.
  - Si no hay más objetivos:
    - Cambiar estado a “regreso a Base”.
    - Ejecutar RETURN\_TO\_BASE().
- Regla 5: Llegada a Base
  - Si llega a la Base con estado “regreso a Base”:
    - Ejecutar DELIVER\_HARVEST():
      - Entregar la producción sana recogida.
    - Enviar reporte al Patólogo Digital:
      - Zonas atendidas.
      - Zonas no atendidas y motivos.
    - Cambiar estado a “disponible”.

#### **4.3.3. Capa de Acción**

Acciones principales del PickBot de Sanos:

- MOVE\_TO(segmento): desplazarse al segmento objetivo seguro.

- APPROACH\_PLANT(planta\_id): posicionarse junto a la planta seleccionada.
- HARVEST\_SAFE(): simular recolección de frutos/planta en zona segura.
- MARK\_UNREACHABLE(objetivo\_id): registrar zona segura no atendida por bloqueo.
- RETURN\_TO\_BASE(): regresar al punto de acopio/Base.
- DELIVER\_HARVEST(): entregar la producción sana recolectada.
- REPORT\_STATUS(): informar estado de misión y disponibilidad.

#### **4.3.4. Características clave del diseño reactivo**

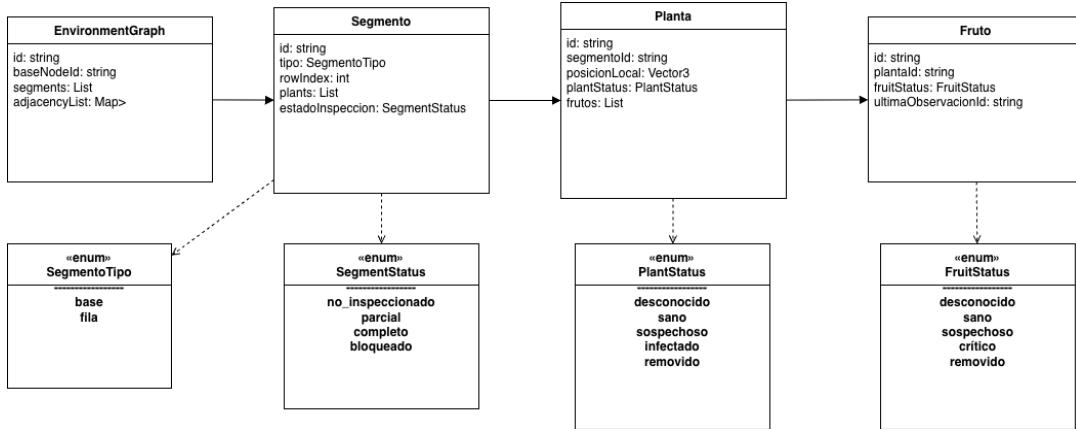
- Opera exclusivamente sobre información validada por el Patólogo Digital.
- No modifica rutas globales ni redefine qué es seguro; solo ejecuta.
- Ayuda a materializar el objetivo de maximizar el aprovechamiento de producción sana, manteniendo una separación clara entre:
  - Flujos de cosecha segura.
  - Flujos de diagnóstico y manejo de sospechosos.

Con este diseño, el PickBot de Sanos complementa al Patólogo y al PickBot Enfermero, cerrando el ciclo operativo de detección, verificación y aprovechamiento seguro dentro del invernadero simulado.

# Diagramas

## Globales

### Dominio del Invernadero

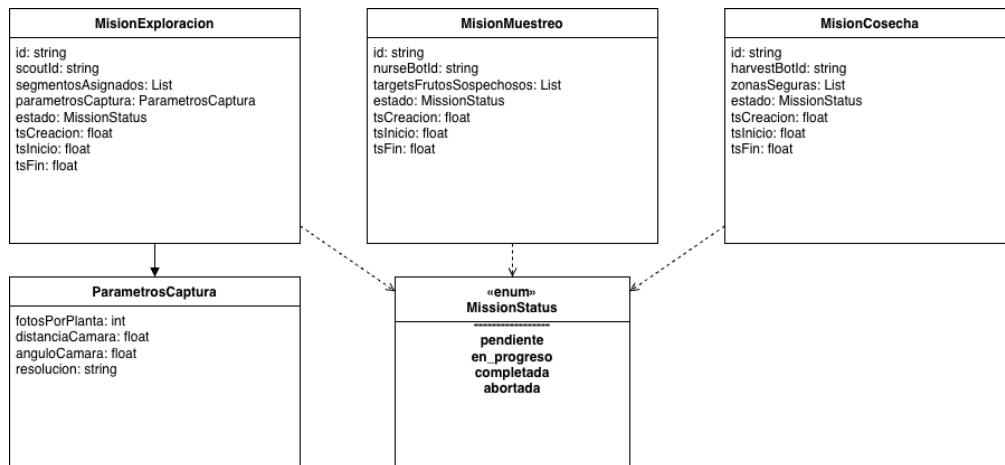


Este diagrama modela la estructura física y el estado sanitario del invernadero, que es la base sobre la que todos los agentes razonan.

- **EnvironmentGraph** representa el mapa navegable del invernadero:
  - Tiene una lista de `Segmento` y una `adjacencyList` que define qué segmentos son vecinos, útil para planificación de rutas y para que los agentes recalculen caminos cuando hay obstáculos.
- **Segmento** es una unidad navegable (tramo de fila o base central):
  - Sabe si es base o fila (`SegmentoTipo`) y su `rowIndex` en el invernadero.
  - Lleva una colección de `Planta` y un `estadolInspeccion` (`SegmentStatus`: `no_inspeccionado`, `parcial`, `completo`, `bloqueado`), que el Patólogo usa para decidir dónde mandar próximos Scouts.
- **Planta** representa cada planta de tomate:
  - Pertenece a un `Segmento`, tiene una posición local 3D (`posicionLocal`).
  - Tiene un `plantStatus` (`PlantStatus`: `desconocido`, `sano`, `sospechoso`, `infectado`, `removido`) y una lista de `Fruto`.
- **Fruto** es cada fruto individual:
  - Pertenece a una `Planta`.

- Tiene un fruitStatus y referencia a la ultimaObservacionId, que permite vincularlo con las imágenes/observaciones más recientes.
- Los enums (SegmentStatus, PlantStatus, FruitStatus, SegmentoTipo) definen el vocabulario común para marcar estados y tipos, que luego usan todos los agentes (Pathologist, ScoutBot, PickBots) en sus creencias y decisiones.

## Dominio del Misiones



Este diagrama modela el trabajo que el Patólogo asigna a los agentes y cómo se representa el ciclo de vida de esas misiones.

- MisionExploracion

Representa una misión que un ScoutBot ejecuta:

- Tiene el scoutId asignado, la lista de segmentosAsignados y sus parametrosCaptura.
- Usa MissionStatus para seguir si la misión está pendiente, en progreso, completada o abortada.
- Lleva timestamps (tsCreacion, tsInicio, tsFin).

- MisionMuestreo

Es la misión del PickBotEnfermero:

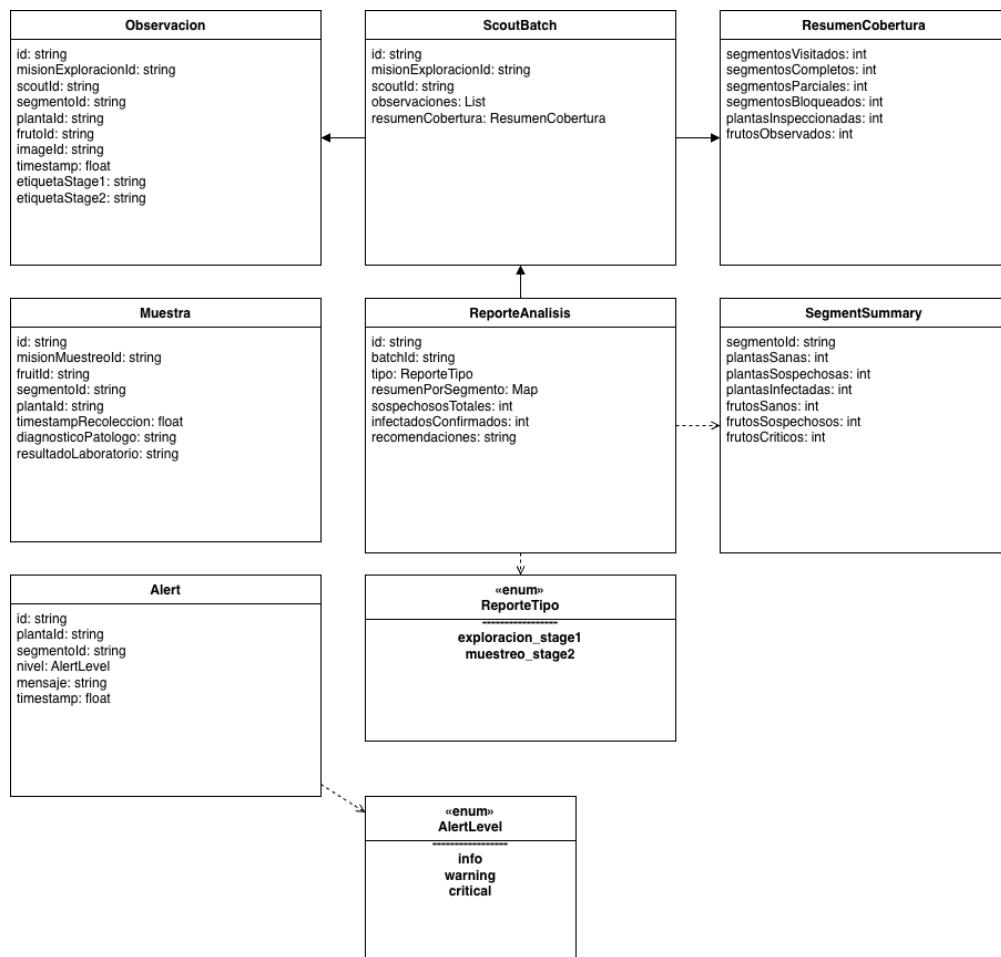
- Contiene nurseBotId y la lista targetsFrutosSospechosos.

- MisionCosecha

Es la misión del PickBotSanos:

- Tiene harvestBotId y zonasSeguras (segmentos o plantas que se consideran seguras para cosecha).
- De nuevo, control de estado con MissionStatus y tiempos.
- ParametrosCaptura  
Es un value-object asociado a MisionExploracion:
  - Define cómo debe tomar las fotos el Scout: fotosPorPlanta, distanciaCamara, anguloCamara, resolucion.
  - Permite que el Patólogo cambie la “configuración de exploración” sin cambiar la lógica del Scout.
- MissionStatus («enum»)  
Proporciona un vocabulario estándar para el estado de cualquier misión:
  - pendiente, en\_progreso, completada, abortada.
  - Esto se refleja en las creencias del Patólogo.

## Observaciones, Muestras y Reportes



Este diagrama describe todo lo que el sistema ve, agrupa y concluye a partir del trabajo de los agentes. Es el dominio de la “evidencia” y del “diagnóstico”.

## Observación y lotes del Scout

- Observacion

Representa una evidencia puntual capturada por un Scout:

- Se liga a una MisionExploracion y a un scoutId.
- También referencia el contexto físico: segmentId, plantaId y opcionalmente frutoId.
- imageId apunta al recurso de imagen almacenado (archivo, textura, etc.).
- timestamp indica cuándo fue tomada.
- etiquetaStage1 y etiquetaStage2 permiten guardar el resultado de los análisis del Patólogo (por ejemplo, “planta\_sospechosa”, “fruto\_critico”).

- ScoutBatch

Es el lote que viaja del Scout al Patólogo:

- Agrupa todas las observaciones generadas en una misión de exploración (misionExploracionId, scoutId).
- Incluye un resumenCobertura que condensa métricas útiles para planificación futura: qué tanto se cubrió, qué quedó bloqueado, etc.

- ResumenCobertura

Es un value object de métricas:

- segmentosVisitados, segmentosCompletos, segmentosParciales, segmentosBloqueados.

- Conteos de plantasInspeccionadas y frutosObservados.
- Esto permite al Patólogo evaluar qué tan efectiva fue la misión y dónde hay huecos.

## Muestras físicas

- Muestra

Representa una muestra física recolectada por el PickBot Enfermero:

- Está ligada a una misionMuestreold (definida en el diagrama 1.2).
- Sabe de qué segmentold, plantald y opcionalmente fruitld proviene.
- Tiene timestampRecolección, útil para rastrear tiempos entre detección, recolección y diagnóstico.
- diagnosticoPatologo guarda el juicio del Patólogo sobre esa muestra.
- resultadoLaboratorio podría usarse si simulan una segunda capa de confirmación externa.

## Reportes de análisis y resúmenes por segmento

- ReporteAnalysis

Es la salida consolidada del Patólogo sobre un lote de datos:

- batchId indica qué ScoutBatch se analizó (para etapa 1) o qué conjunto de muestras (para etapa 2, si lo reutilizan).
- tipo: ReporteTipo distingue entre:
  - exploracion\_stage1: resultados del screening masivo con imágenes.
  - muestreo\_stage2: resultados del análisis de muestras físicas.
- resumenPorSegmento es un mapa segmentold → SegmentSummary.

- sospechososTotales e infectadosConfirmados dan métricas globales.
- recomendaciones puede contener texto con acciones sugeridas (aislar segmentos, re-inspecciones, etc.).
- SegmentSummary

Resume el estado sanitario por segmento después del análisis:

- plantasSanas, plantasSospechosas, plantasInfectadas.
- frutosSanos, frutosSospechosos, frutosCriticos.
- Esto alimenta tanto decisiones del Patólogo como posibles dashboards para el usuario humano.

## Alertas al humano

- Alert

Modela las alertas que el Patólogo envía a la persona humana:

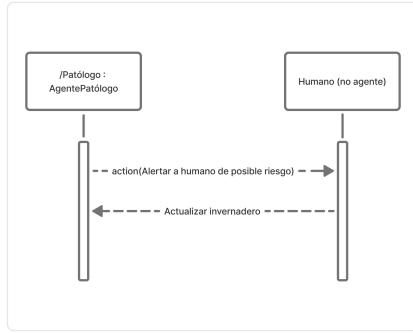
- Apunta a una planta y segmento concretos (plantald, segmentold).
- nivel: AlertLevel indica la gravedad (info, warning, critical).
- mensaje y timestamp permiten registrar qué se comunicó y cuándo.
- Se alinea con el AIP AIP\_AlertaHumano que tendrás en el diagrama global de interacciones.

- ReporteTipo («enum»)

- Distingue la naturaleza del reporte: exploracion\_stage1 vs muestreo\_stage2.

- AlertLevel («enum»)

- Vocabulario estándar para la severidad de una alerta: info, warning, critical.



El AIP\_AlertaHumano define la interacción entre el AgentePatólogo y un usuario humano (no agente) cuando se detecta una condición de riesgo en el invernadero. Su propósito es garantizar que los peligros detectados por los agentes —ya sean riesgos de enfermedad, bloqueos, fallos en sistemas o anomalías— sean comunicados oportunamente a un humano responsable, y que este pueda confirmar la recepción y solicitar la actualización del estado del invernadero.

El AIP se divide conceptualmente en:

- **Fase de detección de riesgo** → donde el AgentePatólogo identifica una situación anómala.
- **Fase de alerta al humano** → donde el Patólogo envía un aviso explícito de riesgo.
- **Fase de reacción humana** → donde el humano valida la alerta y solicita actualización.
- **Mensajes de acción** → expresan interacciones dirigidas a un usuario humano, no a otro agente.

## 1. Detección de riesgo (activación interna del Patólogo)

El protocolo inicia cuando el AgentePatólogo detecta una condición que no puede ser ignorada, típicamente derivada de:

- análisis de muestras,
- datos de ScoutBots, PickBots o sensores,
- inconsistencias en mapas de calor del invernadero,
- cambios bruscos en estados de segmentos (riesgo de plaga, enfermedad o falla estructural).

Esta fase no está representada como mensaje ACL, sino como activación interna del agente que dispara el siguiente paso.

## 2. Alerta al humano (acción → usuario no agente)

El Patólogo envía una alerta dirigida a un operador humano.

Como el destinatario no es un agente FIPA ACL, el mensaje se expresa como **acción**:

action(Alerta a humano de posible riesgo) → Humano (no agente)

Este mensaje incluye:

- la descripción del riesgo detectado,
- los segmentos involucrados,
- la posible causa (planta enferma, bloqueo, fallo mecánico, etc.),
- el nivel de severidad,
- recomendaciones inmediatas (revisar, aislar, intervenir).

La naturaleza del mensaje es unimodal:

**el Patólogo alerta, el humano recibe.**

No se requiere negociación o aceptación.

### 3. Respuesta humana: solicitud de actualización

Tras recibir la alerta, el humano puede solicitar al Patólogo una actualización del estado del invernadero para entender mejor el problema, confirmar riesgos o modificar instrucciones operativas.

Esto se modela en el AIP como:

Actualizar invernadero ← Humano (no agente)

La respuesta puede representar:

- confirmación de que el humano recibió la alerta,
- una petición de estado actualizado,
- una instrucción para pausar tareas,
- una orden para generar un reporte ampliado,
- o una solicitud para reconfigurar misiones de agentes.

El Patólogo procesa esta reacción humana y actualiza su modelo:

- reevalúa segmentos a partir del grafo,
- revisa estados recientes de los agentes,
- reenfila misiones si es necesario,
- publica estados renovados en su base interna.

#### 4. Cierre del protocolo

Una vez que el Patólogo responde a la solicitud del humano (actualizando el invernadero o generando el reporte solicitado), el protocolo concluye.

No existe un mensaje formal de “finalización” porque el humano no opera bajo ACL; sin embargo, dentro del MAS se considera:

- alerta emitida,
- reacción registrada,
- actualización ejecutada,
- sistema listo para reanudar operaciones.

#### 5. Lista de mensajes del AIP\_AlertaHumano()

Salida del Patólogo

- action(Alerta a humano de posible riesgo)

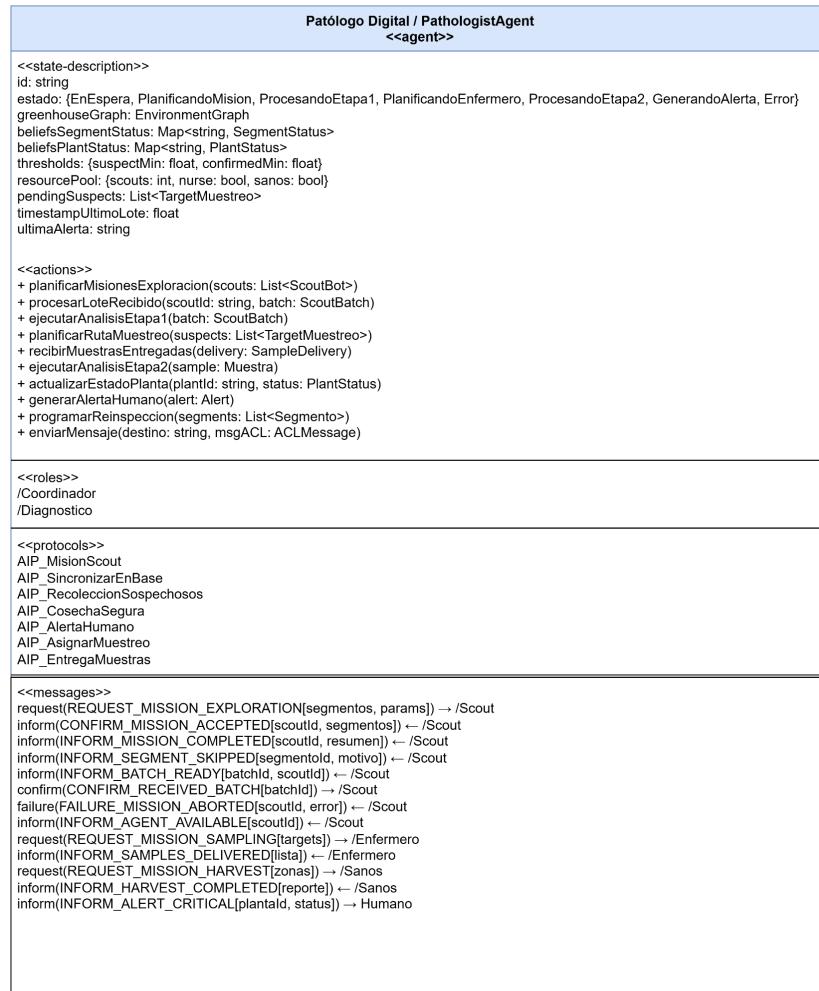
Entrada hacia el Patólogo

- Actualizar invernadero

*(mensaje no ACL, originado por un humano, representado como evento externo)*

# Agente Patólogo

## Diagrama de clase



Este diagrama de clase de agente modela la arquitectura interna del Patólogo Digital, definiéndolo como el núcleo inteligente del sistema encargado de la coordinación estratégica y el diagnóstico fitosanitario.

El esquema formaliza cómo el agente construye su conocimiento del entorno mediante creencias y variables de estado, detalla las operaciones lógicas que ejecuta para procesar datos visuales y asignar tareas, y establece la interfaz de

comunicación estandarizada necesaria para interactuar con los agentes operativos (ScoutBot y PickBots) dentro del invernadero.

### 1. Estado interno

El estado interno del PickBot de Sanos se define por las siguientes variables clave: id (identificador único del agente); estado (máquina de estados de alto nivel que incluye Idle, EjecutandoMision, Cosechando, ContenedorLleno y RegresandoBase); greenhouseGraph (referencia al grafo de segmentos para navegación); zonasAsignadas (la lista ordenada de áreas seguras a cosechar asignadas por el Patólogo); cargaActual (kilos o volumen actual de la cosecha recolectada); y capacidadMaxima (el límite físico de la carga que puede transportar).

### 2. Acciones

Las acciones del PickBot de Sanos se enfocan estrictamente en la ejecución de la cosecha: recibirOrdenCosecha() (carga la lista de segmentos seguros), navegarAZona() (desplazamiento al segmento objetivo), cosecharPlanta() (simula la recolección de frutos sanos), verificarContenedor() (comprueba si la capacidad máxima fue alcanzada), reportarZonaBloqueada() (registra segmentos no atendidos por obstáculos), regresarABase() (inicia la navegación de retorno), descargarCosecha() (transfiere la producción recolectada), y enviarMensaje() (utiliza el módulo de comunicación).

### 3. Roles

El agente desempeña tres roles clave dentro del sistema multiagente: Cosechador (la responsabilidad principal de recolección de producción), Ejecutor (cumplir las

órdenes del Patólogo sin deliberación propia), y Transportista (encargado del traslado físico de la producción segura desde el campo hasta la Base de acopio)

#### 4. Protocolos

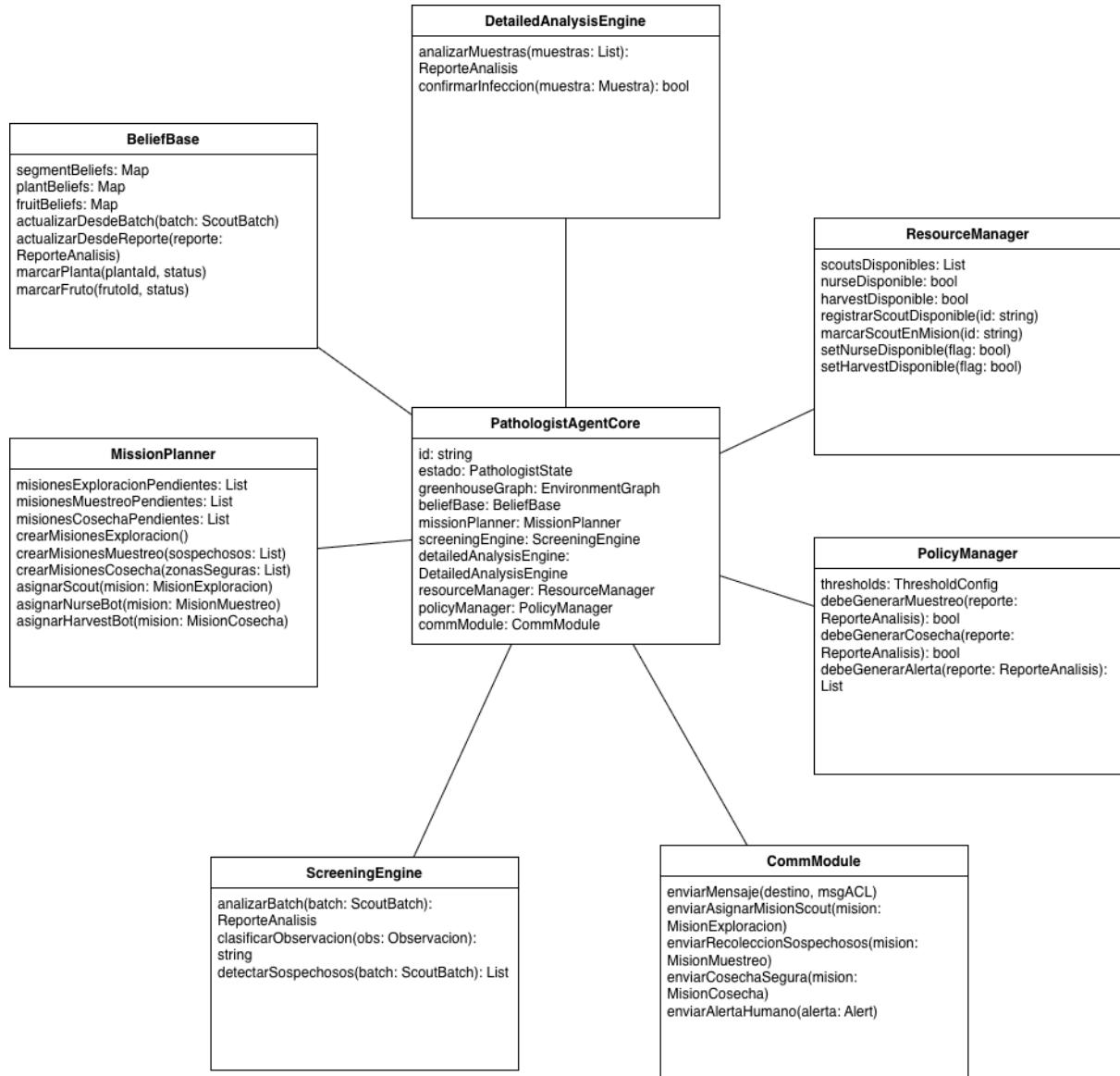
El PickBot de Sanos participa en tres protocolos de interacción principales que definen su ciclo de trabajo: AIP\_CosechaSegura (protocolo para recibir y aceptar la solicitud de misión), AIP\_EntregaCosechaSana (protocolo para reportar la finalización de la misión y la entrega de la producción), y AIP\_SincronizarBase (protocolo estándar de reporte de estado al llegar a la Base).

5. El vocabulario de comunicación incluye los siguientes mensajes: request(HARVEST\_SAFE\_ZONES) (recibido del Patólogo para iniciar la misión); agree / refuse (respuesta para aceptar o rechazar la misión); inform(ZONE\_UNREACHABLE) (notificación si una zona segura no pudo ser atendida); inform(HARVEST\_DONE) (reporte de que la cosecha ha finalizado y se ha descargado); y inform(AVAILABLE) (señalización de que el agente está listo para recibir una nueva orden)

#### AIP

Al ser el agente central, interactúa con todos los agentes. Se añaden los AIP en cada uno de los siguientes agentes.

## Subsistemas



Este diagrama muestra la arquitectura interna del Patólogo como un conjunto de módulos especializados, todos orquestados por PathologistAgentCore.

- PathologistAgentCore

Es el “cerebro central” del Patólogo.

- Mantiene el estado del agente y una referencia al EnvironmentGraph.

- Orquestra a los demás módulos: les pasa batches, reportes, misiones y recibe sus resultados.
- Expone la lógica de alto nivel: procesar batches, actualizar creencias, planear exploración, muestreo y cosecha, y evaluar alertas.
- BeliefBase

Representa la base de creencias del Patólogo sobre el invernadero.

- Guarda los estados de segmentos, plantas y frutos (SegmentStatus, PlantStatus, FruitStatus).
- Actualiza estos estados a partir de ScoutBatch y ReporteAnalisis.
- Proporciona operaciones para marcar plantas o frutos como sospechosos, infectados, etc.

- MissionPlanner

Es el módulo que diseña y administra las misiones.

- Mantiene listas de misiones pendientes de exploración, muestreo y cosecha.
- Decide qué misiones crear y a qué agente asignarlas (scout, nurseBot, harvestBot).
- Se apoya en la BeliefBase, el ResourceManager y las políticas para priorizar.

- ScreeningEngine

Implementa la etapa 1 de análisis (screening masivo de imágenes).

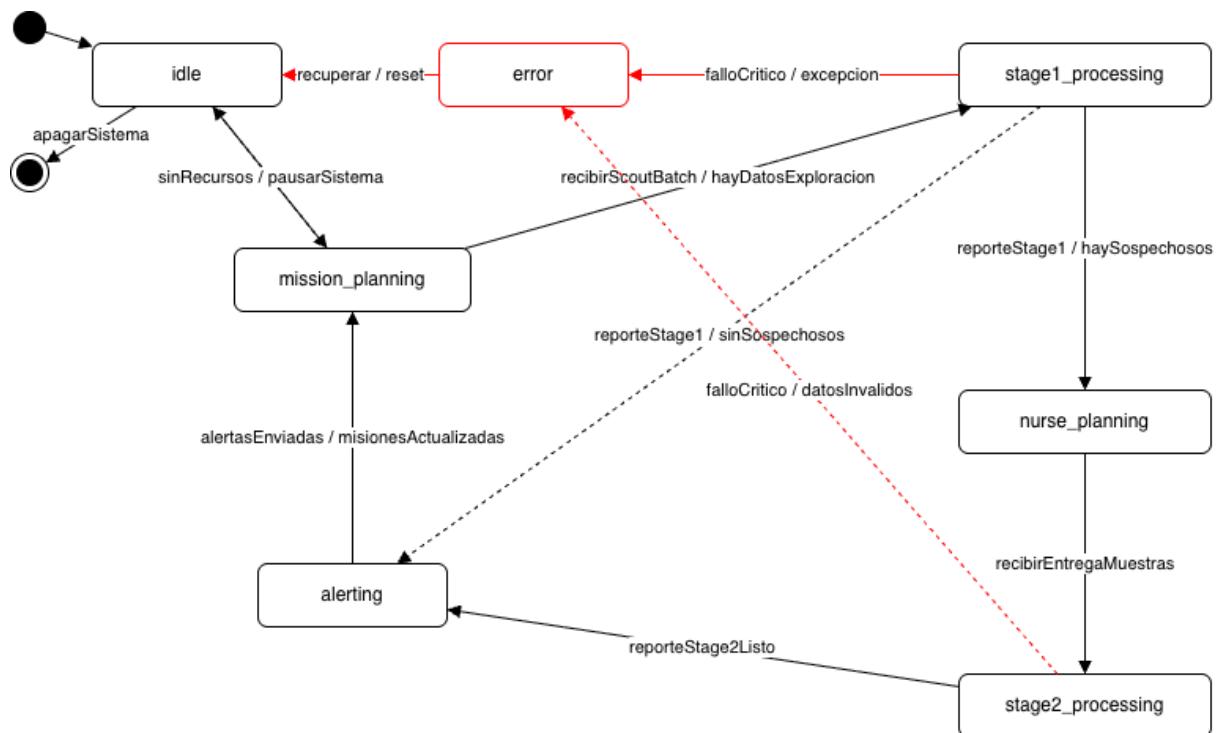
- Toma un ScoutBatch y genera un ReporteAnalisis de exploración.
- Clasifica observaciones individuales (por ejemplo “planta\_sospechosa”).

- Detecta una lista de candidatos sospechosos para misiones de muestreo.
- DetailedAnalysisEngine
  - Implementa la etapa 2 de análisis (muestras físicas).
    - Trabaja con listas de Muestra.
    - Produce reportes detallados de muestreo (ReporteAnalisis de stage 2).
    - Confirma si una muestra indica infección o no.
- ResourceManager
  - Administra los recursos disponibles:
    - Lista de scoutsDisponibles y disponibilidad de nurse y harvest.
    - Expone métodos para registrar disponibilidad y marcar agentes en misión.
    - Sirve de insumo al MissionPlanner para no planear más de lo que el sistema puede ejecutar.
- PolicyManager
  - Encapsula las políticas y umbrales de decisión.
    - Contiene los thresholds de sospecha, confirmación, etc.
    - Decide cuándo se debe generar muestreo, cosecha o alerta a partir de un ReporteAnalisis.
    - Permite cambiar comportamiento “estratégico” sin tocar la lógica de bajo nivel.
- CommModule
  - Es la capa de comunicación ACL del Patólogo.
    - Implementa el envío de mensajes para los AIPs: asignar misiones a Scouts, Enfermero, Harvest, y enviar alertas al humano.

- Aísla los detalles de formato de mensaje y protocolo, para que el core solo diga “envía esta misión” o “envía esta alerta”.

En conjunto, el diagrama deja claro que el Patólogo no es un monolito, sino un agente BDI modular: creencias (BeliefBase), generación de planes (MissionPlanner + políticas) y ejecución/comunicación (CommModule).

## Estados



Este diagrama muestra el ciclo de vida lógico del Patólogo como agente BDI:

- **idle**

El sistema está encendido pero sin monitoreo activo, o en pausa.

- **mission\_planning**

El Patólogo analiza su base de creencias y recursos para planear nuevas misiones de exploración para los ScoutBots:

- Decide qué segmentos necesitan inspección.
  - Asigna misiones a Scouts disponibles.
- stage1\_processing
    - El Patólogo recibe uno o varios ScoutBatch y ejecuta el screening masivo de imágenes:
      - Llama a ScreeningEngine.
      - Genera un ReporteAnalisis de stage 1.
      - Actualiza la BeliefBase con plantas/segmentos sospechosos o sanos.
- nurse\_planning
    - Si el reporte de stage 1 indica suficientes sospechosos:
      - Selecciona los frutos/plants candidatos.
      - Crea MisionMuestreo para el PickBot Enfermero.
      - Asigna y envía misiones de muestreo.
- stage2\_processing
    - El Patólogo recibe las Muestras recolectadas y hace el análisis detallado:
      - Llama a DetailedAnalysisEngine.
      - Confirma infecciones.
      - Actualiza creencias con estados “infectado”/“descartado”.
- alerting
    - Con la información consolidada (stage 1 + stage 2):
      - Genera alertas a humanos si es necesario.
      - Identifica zonas seguras y críticas.
      - Puede crear misiones de cosecha (MisionCosecha) para PickBots de sanos.
      - Tras alertar y actualizar misiones, regresa a mission\_planning.

- error

Estado de fallo crítico (datos inválidos, error grave).

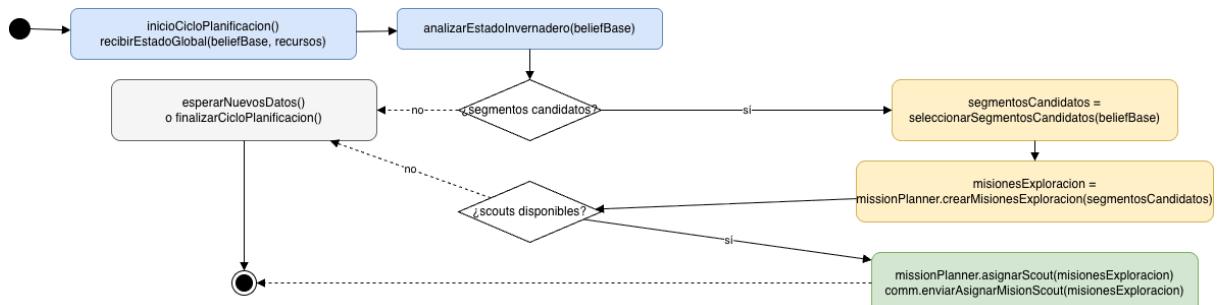
Desde aquí puede:

- Intentar recuperación y volver a idle.
- Registrar el fallo.

Transiciones clave:

- idle → mission\_planning: arranque de monitoreo o nuevo ciclo.
- mission\_planning → stage1\_processing: comienzan a llegar ScoutBatch de exploración.
- stage1\_processing → nurse\_planning: hay suficientes sospechosos.
- stage1\_processing → alerting: no hay sospechosos, pero sí actualización de estado global.
- nurse\_planning → stage2\_processing: llegan muestras físicas.
- stage2\_processing → alerting: se concluye el stage 2.
- alerting → mission\_planning: se cierra el ciclo con misiones, alertas y posibles cosechas.

## Actividad



Esta actividad modela lo que hace el Patólogo en el modo mission\_planning, cuando decide qué segmentos explorar, cómo agruparlos en misiones y cómo asignarlos a los ScoutBots.

Flujo principal:

1. Recibir estado global / inicio de ciclo

El Patólogo entra en modo de planificación con:

- BeliefBase actualizada (segmentos inspeccionados, sospechosos, sin datos, etc.).
- Información de recursos (ResourceManager: scouts disponibles).

2. Analizar creencias

Actividad: analizarEstadoInvernadero(beliefBase)

- Identifica segmentos sin inspección reciente.
- Revisa segmentos con dudas o calidad de datos baja.
- Marca candidatos para nueva exploración.

3. Decisión: ¿hay segmentos candidatos?

- Si no hay nada urgente: pasa a una actividad de “esperar nuevos datos” o termina el ciclo de planificación.
- Si sí, continúa.

4. Seleccionar segmentos candidatos

|  |                     |   |
|--|---------------------|---|
| Actividad:                                 | segmentosCandidatos | = |
| seleccionarSegmentosCandidatos(beliefBase) |                     |   |

Se filtra una lista de segmentos prioritarios (por cobertura, tiempo desde última inspección, etc.).

## 5. Agrupar en misiones

Actividad: misionesExploracion =  
 missionPlanner.crearMisionesExploracion(segmentosCandidatos)

- Agrupa segmentos contiguos en misiones razonables.
- Cada misión tiene lista de segmentos + parámetros de captura.

## 6. Decisión: ¿hay Scouts disponibles?

Se consulta ResourceManager:

- Si no, la planificación se queda “en espera” (sin mandar misiones).
- Si sí, se asignan.

## 7. Asignar y enviar misiones

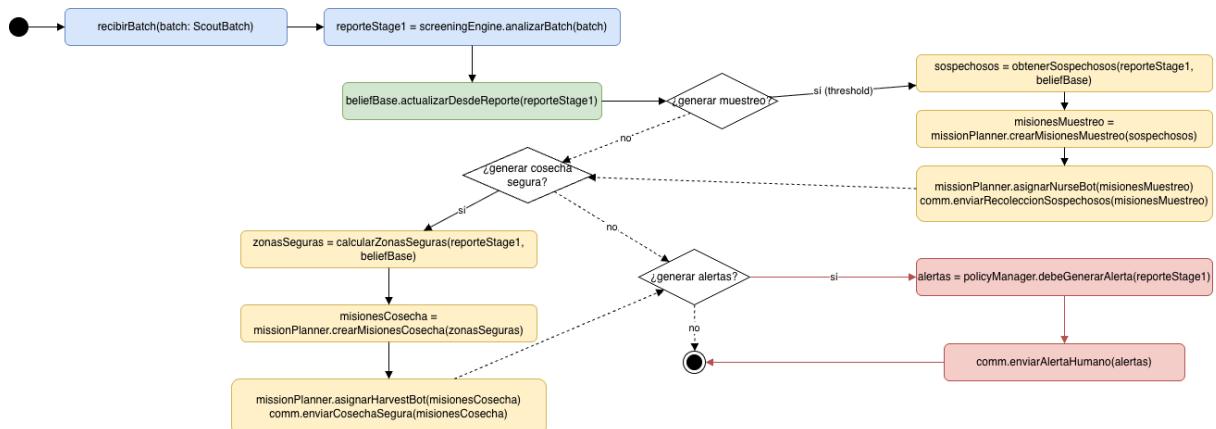
Actividad:

- missionPlanner.asignarScout(misionesExploracion)
- comm.enviarAsignarMisionScout(misionesExploracion)

Se eligen Scouts específicos y se envían los mensajes ACL correspondientes.

## 8. Final del ciclo de planificación

El Patólogo termina este ciclo; esperará reportes de los Scouts o disparará otra ronda más adelante.



Este flujo describe lo que hace el Patólogo cuando llega un ScoutBatch:

1. recibirBatch(batch)

El Patólogo recibe el lote de observaciones del Scout (AIP de reporte).

2. screeningEngine.analizarBatch(batch)

Ejecuta el análisis de etapa 1:

- Clasifica observaciones.
- Resume cobertura por segmento.
- Produce reporteStage1.

3. beliefBase.actualizarDesdeReporte(reporteStage1)

Actualiza las creencias:

- Marca plantas/segmentos como sanos, sospechosos, etc.
- Actualiza conteos por segmento.

4. Decisión: ¿generar muestreo?

Con PolicyManager y el reporte:

- Si el nivel de sospecha supera umbral → sí.
- Si no, se salta el muestreo.

5. Si SÍ muestreo:

- sospechosos = obtenerSospechosos(...)

Se lista qué plantas/frutos deben tener muestra física.

- misionesMuestreo

=

missionPlanner.crearMisionesMuestreo(sospechosos)

Se agrupan en misiones para el NurseBot.

- missionPlanner.asignarNurseBot(...)

y

comm.enviarRecoleccionSospechosos(...)

Se asignan y envían las misiones via AIP correspondiente.

## 6. Decisión: ¿generar cosecha segura?

Con base en el mismo reporte y creencias:

- Si hay zonas claramente sanas → se pueden explotar ahora.
- Si no, se omite para este batch.

## 7. Si Sí cosecha:

- zonasSeguras = calcularZonasSeguras(...)

- misionesCosecha

=

missionPlanner.crearMisionesCosecha(zonasSeguras)

- missionPlanner.asignarHarvestBot(...)

y

comm.enviarCosechaSegura(...)

## 8. Decisión: ¿generar alertas humanas?

Se evalúa si hay:

- Infectados confirmados.
- Riesgo alto en segmentos específicos.

## 9. Si Sí alertas:

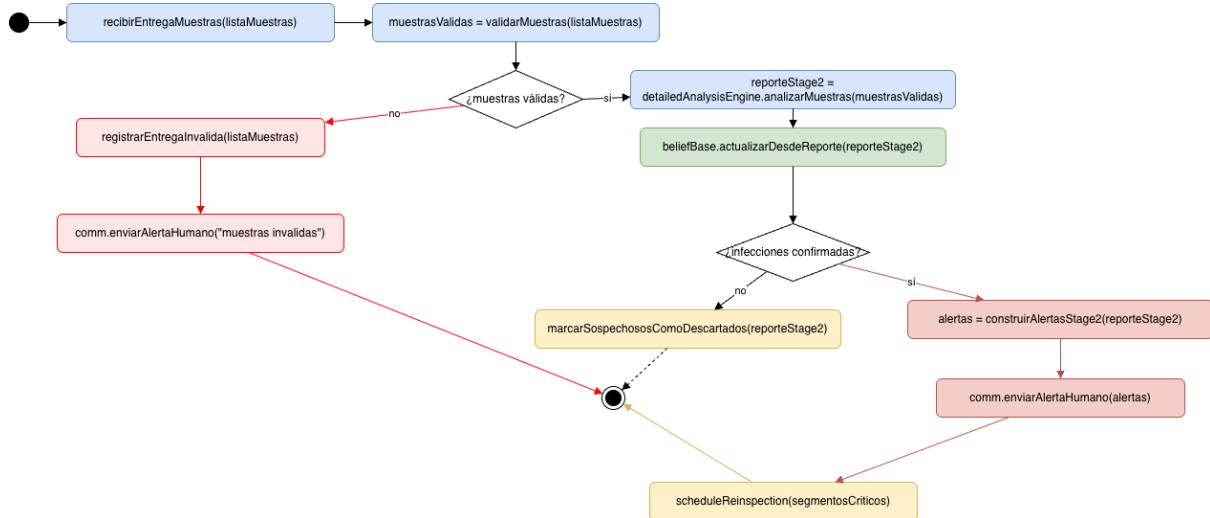
- alertas = policyManager.debeGenerarAlerta(reporteStage1)

- comm.enviarAlertaHumano(alertas)

## 10. Fin de flujo

El Patólogo termina de procesar ese batch y queda listo para:

- Seguir recibiendo más batches.
- Entrar a stage 2 cuando lleguen muestras.
- Actualizar su estado a alerting / mission\_planning según corresponda.



Esta actividad describe el flujo del Patólogo cuando recibe muestras físicas del PickBot Enfermero (etapa 2 del diagnóstico).

Flujo principal:

1. Recibir entrega de muestras

Actividad: recibirEntregaMuestras(listaMuestras)

Llega la lista de Muestra (AIP EntregaMuestras).

2. Validar muestras

Actividad: muestrasValidas = validarMuestras(listaMuestras)

- Se descartan duplicadas, dañadas, incompletas.
- Se pueden registrar incidencias.

3. Decisión: ¿hay muestras válidas?

- Si no:
  - Se registra la entrega como problemática.
  - Opcionalmente se genera una alerta humana.
  - Termina el flujo.
- Si sí, continúa.

#### 4. Analizar muestras – Stage 2

Actividad: reporteStage2 =  
detailedAnalysisEngine.analizarMuestras(muestrasValidas)

- Análisis más caro (biología, IA pesada).
- Marca muestras como infectadas / sanas / dudosas.

#### 5. Actualizar creencias

Actividad: beliefBase.actualizarDesdeReporte(reporteStage2)

- Se actualizan los estados de plantas y frutos asociados a las muestras.
- Se consolidan segmentos con infección confirmada.

#### 6. Decisión: ¿hay infecciones confirmadas?

- Si no:
  - Se marcan sospechosos previos como descartados.
  - Se pueden programar reinspecciones suaves.
- Si sí:
  - Se construyen alertas específicas para humanos.
  - Se programa reinspección y/o destruir/aislar zonas.

#### 7. Acciones según resultado

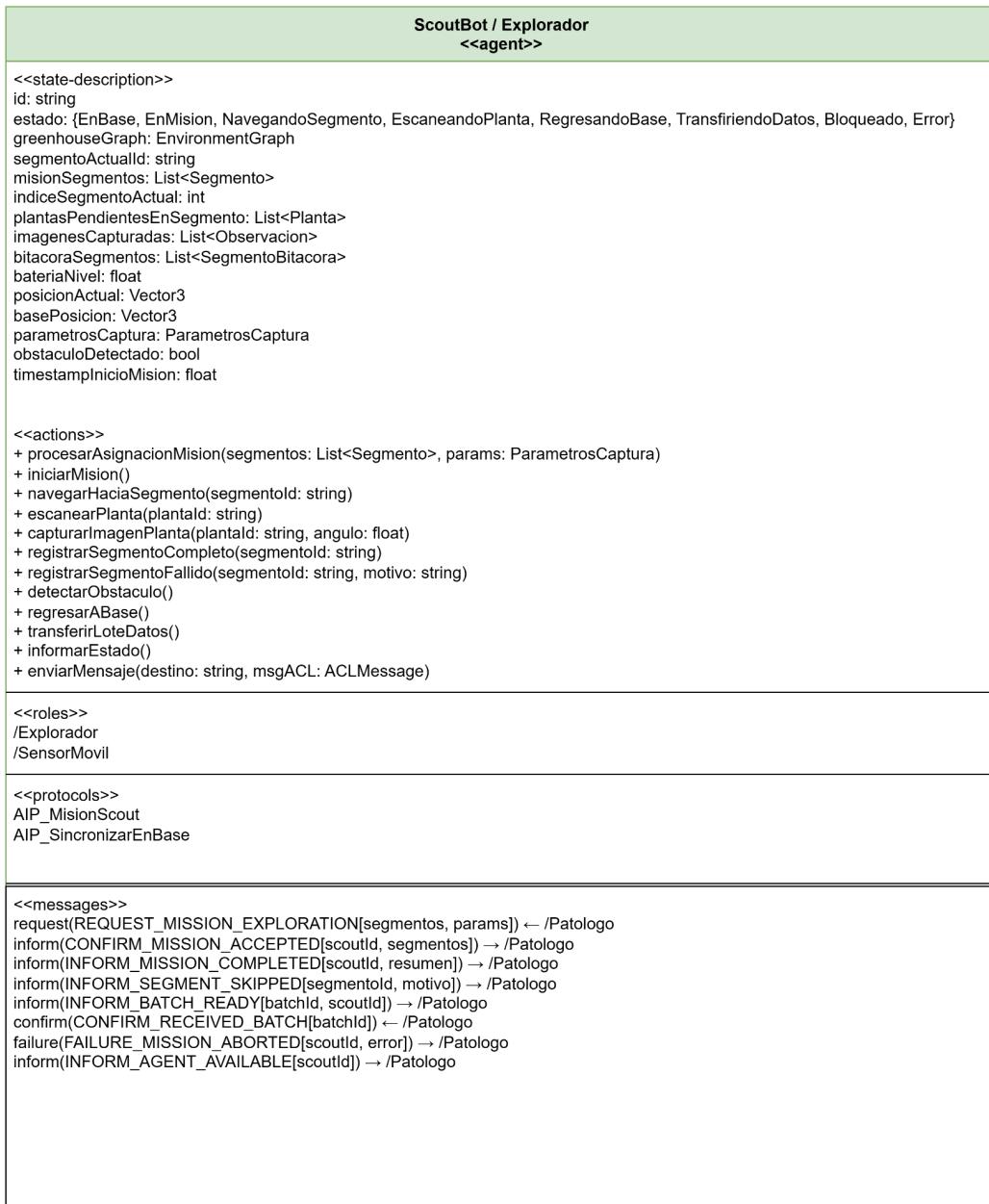
- marcarSospechososComoDescartados(reporteStage2) (cuando no hay infección).
- alertas = construirAlertasStage2(reporteStage2)  
y comm.enviarAlertaHumano(alertas) (cuando sí hay infección).
- scheduleReinspection(segmentosCríticos).

#### 8. Fin del flujo

El Patólogo termina de procesar esa entrega y actualiza su estado general.

# Agente Scoutbot

## Diagrama de clase



Es un agente autónomo cuyo rol es actuar como explorador del invernadero: recibe misiones de exploración, navega por los segmentos asignados, escanea plantas, capture imágenes para diagnóstico y regresa a base para transferir los datos al Patólogo Digital.

El diagrama se divide en:

- <<state-description>> → qué sabe / qué recuerda (estado interno).
- <<actions>> → qué puede hacer (operaciones de alto nivel).
- <<roles>> → papel que juega dentro de la organización multiagente.
- <<protocols>> → en qué AIPs participa.
- <<messages>> → mensajes ACL típicos que envía/recibe.

### 1. Bloque <<state-description>> (estado interno)

- id
  - Identificador único del ScoutBot (por ejemplo, "Scout\_01").
  - Se usa en bitácoras, logs y mensajes ACL para distinguirlo de otros robots.
- estado

Máquina de estados de alto nivel:

- EnBase: el bot está en la base, sin misión activa.
- EnMision: ha aceptado una misión y está procesándola.
- NavegandoSegmento: se está moviendo hacia un segmento objetivo.
- EscaneandoPlanta: está capturando imágenes/observaciones de una planta.
- RegresandoBase: ha terminado la misión (o la abortó) y vuelve a base.
- TransfiriendoDatos: está enviando el lote de datos/imágenes al Patólogo.
- Bloqueado: encontró un obstáculo o ruta inalcanzable y espera recálculo.

- Error: ocurrió un fallo crítico que detuvo la misión.
- greenhouseGraph : EnvironmentGraph
  - Referencia al grafo que modela el invernadero (pasillos, intersecciones, segmentos).
  - Se usa para planear rutas, detectar bloqueos y recalcular caminos alternos.
- segmentoActualId : string
  - Identificador del segmento donde está trabajando en este momento (p. ej. "RowA\_03\_Left").
  - Permite ubicar el foco actual de exploración.
- misionSegmentos : List
  - Lista ordenada de segmentos que forman la misión de exploración.
  - Proviene directamente de la asignación del Patólogo.
- indiceSegmentoActual : int
  - Índice dentro de misionSegmentos que indica qué segmento se está inspeccionando.
  - Se incrementa cuando se marca un segmento como completo o fallido.
- plantasPendientesEnSegmento : List
  - Plantas del segmentoActualId que aún no han sido escaneadas.
  - Se va vaciando conforme el bot procesa cada planta.
- imagenesCapturadas : List
  - Colección de observaciones generadas durante la misión (imágenes + metadatos).
  - Sirve de base para crear el lote de datos que se transferirá al Patólogo.

- bitacoraSegmentos : List
  - Registro de cómo terminó cada segmento: completado, fallido, omitido, tiempos, nº de imágenes, etc.
  - Se usa para armar el resumen en INFORM\_MISSION\_COMPLETE.
- bateriaNivel : float
  - Nivel de batería actual del ScoutBot (0–100 o normalizado).
  - Permite decidir si se acepta una nueva misión o si debe abortar y regresar a base.
- posicionActual : Vector3
  - Posición 3D actual del robot en el mundo (Unity).
  - Se utiliza para navegación y para reportes de estado al Patólogo.
- basePosicion : Vector3
  - Ubicación de la base / docking station donde el ScoutBot inicia y termina misiones.
- parametrosCaptura : ParametrosCaptura
  - Conjunto de parámetros que controlan cómo se capturan las imágenes:
  - número de fotos por planta, distancia, ángulo, resolución, etc.
- obstaculoDetectado : bool
  - Indica si se detectó un obstáculo en la ruta actual.
  - Cuando es true, normalmente dispara el subsistema de recálculo de ruta.
- timestampInicioMision : float
  - Marca de tiempo del inicio efectivo de la misión.
  - Permite medir la duración de la misión y detectar timeouts.

## 2. Bloque <>actions<> (comportamiento)

Estas son las operaciones de alto nivel que definen lo que puede hacer el ScoutBot:

- procesarAsignacionMision(segmentos, params)
  - Procesa el request(REQUEST\_MISSION\_EXPLORATION[segmentos, params]) del Patólogo.
  - Guarda los segmentos en misionSegmentos, los parámetros en parametrosCaptura y prepara el estado para iniciar la misión.
- iniciarMision()
  - Inicializa contadores, establece indiceSegmentoActual = 0, captura timestampInicioMision y hace la transición de EnBase a EnMision.
- navegarHaciaSegmento(segmentoid)
  - Planea y sigue una ruta desde posicionActual hasta el segmentoid usando greenhouseGraph.
  - Actualiza el estado a NavegandoSegmento y vigila la presencia de obstáculos.
- escanearPlanta(plantaid)
  - Coordina el proceso de inspección de una planta: posiciona el robot, captura imágenes y actualiza plantasPendientesEnSegmento.
- capturarImagenPlanta(plantaid, angulo)
  - Realiza la captura concreta de una imagen para una planta y ángulo dados.
  - Genera una Observacion que se añade a imagenesCapturadas.
- registrarSegmentoCompleto(segmentoid)

- Marca el segmento como correctamente explorado en bitacoraSegmentos.
- Actualiza indiceSegmentoActual para avanzar al siguiente segmento de la misión.
- registrarSegmentoFallido(segmentoid, motivo)
  - Registra en la bitácora que el segmento no pudo explorarse (bloqueo, error de sensores, etc.), con la causa.
  - Se usa luego para construir el mensaje INFORM\_SEGMENT\_SKIPPED.
- detectarObstaculo()
  - Analiza los sensores de proximidad / mapa para determinar si hay un bloqueo en ruta.
  - En caso afirmativo, pone obstaculoDetectado = true y puede disparar el recálculo.
- regresarABase()
  - Calcula una ruta desde la posicionActual hasta basePosicion y navega de vuelta.
  - Se usa al terminar la misión o al abortarla.
- transferirLoteDatos()
  - Empaquea imagenesCapturadas y bitacoraSegmentos en un lote identificado por batchId y lo envía al Patólogo.
  - Normalmente precede a INFORM\_BATCH\_READY y CONFIRM\_RECEIVED\_BATCH.
- informarEstado()
  - Centraliza la lógica para reportar progreso y estado al Patólogo:

- estado actual, segmento, batería, nº de imágenes, etc.
  - Se mapea a mensajes como INFORM\_MISSION\_STATUS o INFORM\_AGENT\_STATUS.
- enviarMensaje(destino, msgACL)
  - Operación genérica para mandar mensajes ACL a otros agentes (principalmente al Patólogo).
  - Permite reutilizar una misma infraestructura de comunicación desde las demás acciones.

### 3. Bloque <>roles<>

Indica los papeles organizacionales que puede desempeñar el ScoutBot:

- /Explorador
  - Rol principal: recorrer segmentos del invernadero para obtener evidencia visual del estado de las plantas.
- /SensorMovil
  - Resalta su función como plataforma móvil de sensores (cámara, posiblemente otros).
  - Alimenta de datos a los procesos de diagnóstico y planificación de otros agentes.

### 4. Bloque <>protocols<>

Enumera los AIPs en los que participa el ScoutBot:

- AIP\_MisionScout

- Protocolo de misión de exploración: el Patólogo asigna segmentos, el ScoutBot acepta o rechaza, explora, maneja segmentos omitidos y reporta
 

|                          |   |
|--------------------------|---|
| INFORM_MISSION_COMPLETE  | o |
| FAILURE_MISSION_ABORTED. |   |
- AIP\_SincronizarEnBase
  - Protocolo de sincronización al final de la jornada: el ScoutBot reporta su estado, recibe CONFIRM\_SYNC\_COMPLETED y envía INFORM\_AGENT\_AVAILABLE cuando está listo para una nueva misión.

## 5. Bloque <>messages<> (comunicación ACL)

Lista de mensajes típicos que intercambia con el Patólogo:

Entrada principal

- request(REQUEST\_MISSION\_EXPLORATION[segmentos, params]) ← /Patologo

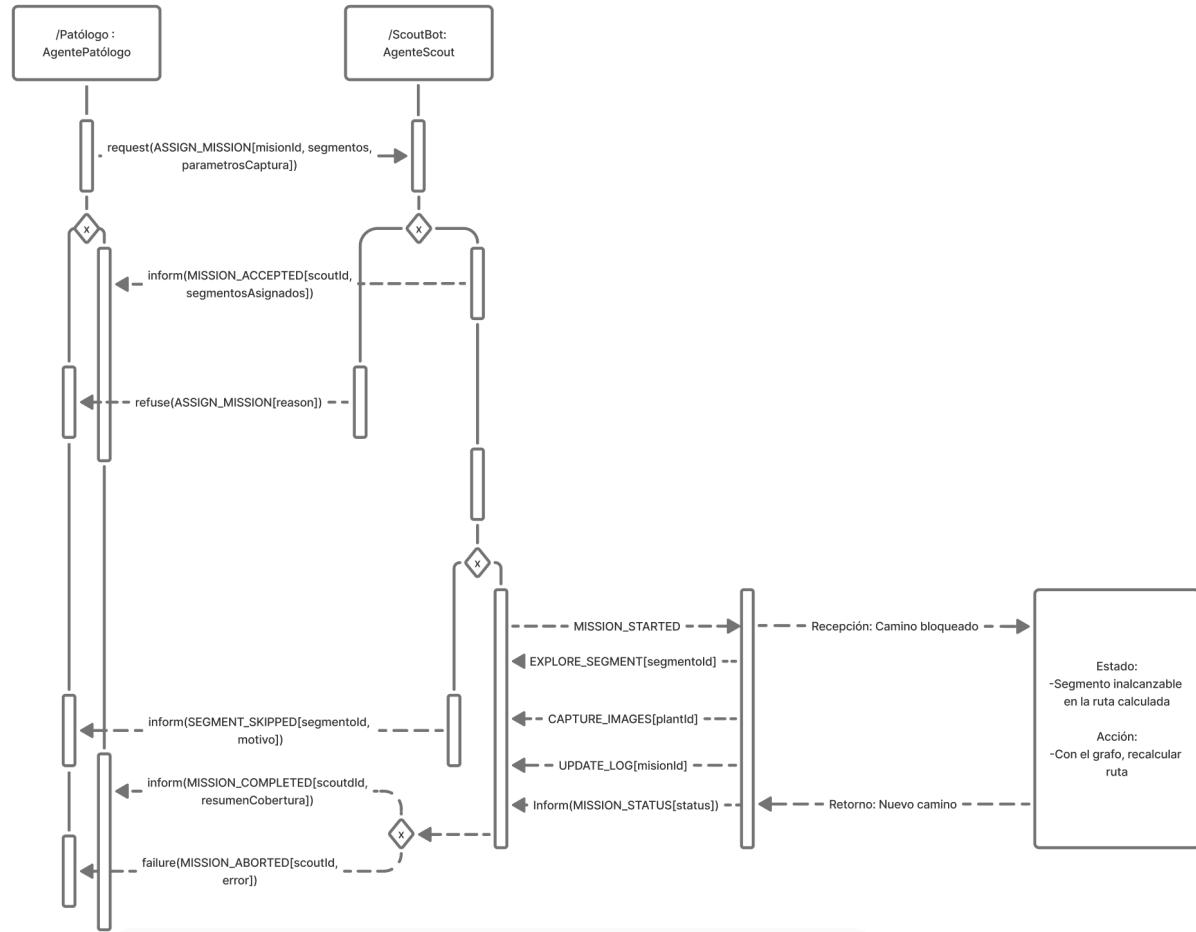
El Patólogo asigna una nueva misión de exploración.

Salidas del ScoutBot

- inform(CONFIRM\_MISSION\_ACCEPTED[scoutId, segmentos]) → /Patologo  
Confirma aceptación de la misión y los segmentos a explorar.
- inform(INFORM\_MISSION\_COMPLETE[scoutId, resumen]) → /Patologo  
Resume la misión: segmentos cubiertos, tiempos, nº de imágenes, etc.
- inform(INFORM\_SEGMENT\_SKIPPED[segmentId, motivo]) → /Patologo  
Informa que un segmento se omitió y explica por qué.

- `inform(INFORM_BATCH_READY[batchId, scoutId]) → /Patologo`  
Indica que hay un lote de datos listo para ser consumido.
- `confirm(CONFIRM_RECEIVED_BATCH[batchId]) ← /Patologo`  
Confirmación de que el Patólogo recibió el lote (mensaje de entrada para el ScoutBot).
- `failure(FAILURE_MISSION_ABORTED[scoutId, error]) → /Patologo`  
Informa que la misión se abortó por un error crítico.
- `inform(INFORM_AGENT_AVAILABLE[scoutId]) → /Patologo`  
Señala que el ScoutBot terminó, está en base y listo para una nueva misión.

## AIP



El AIP se divide conceptualmente en:

- Fase de asignación de misión → donde el Patólogo delega el trabajo.
- Fase de ejecución y exploración → donde el ScoutBot recorre segmentos, captura imágenes y actualiza bitácoras.
- Subsistema de recálculo de ruta → que se activa cuando un segmento no es alcanzable.
- Fase de cierre → donde el ScoutBot reporta el resultado (completado o abortado) y notifica disponibilidad.

- Mensajes ACL → representan el flujo formal de comunicación.

### 1. Asignación de misión (request → accept/refuse)

```
request(ASSIGN_MISSION[missionId,    segmentos,    parametrosCaptura]) ←
/Patólogo
```

El AIP inicia cuando el Patólogo envía al ScoutBot una misión que incluye:

- missionId: identificador de misión.
- segmentos: lista ordenada de segmentos que deben explorarse.
- parametrosCaptura: reglas de captura (número de imágenes, ángulos, distancia mínima).

Esto coloca al ScoutBot en fase de decisión.

### (XOR) Decisión del ScoutBot

En este punto, el ScoutBot evalúa si puede aceptar la misión:

- nivel de batería suficiente,
- sensores funcionales,
- estado libre (idle),
- ruta inicial calculable.

Solo ocurre una de las siguientes ramas:

```
inform(MISSION_ACCEPTED[scoutId, segmentosAsignados]) → /Patólogo
```

El ScoutBot acepta la misión, guarda los segmentos asignados e inicia su fase de preparación de navegación.

`refuse(ASSIGN_MISSION[reason]) → /Patólogo`

El ScoutBot rechaza la misión por alguna razón (batería baja, conflicto de misión, falla en sensores).

Con esto, el protocolo termina para esta instancia.

## 2. Ejecución de misión y exploración del invernadero

Una vez aceptada la misión, el ScoutBot entra en el ciclo de exploración.

`MISSION_STARTED`

Marcador interno que indica el inicio de la inspección.

`EXPLORE_SEGMENT[segmentId]`

El ScoutBot navega hacia el segmento objetivo siguiendo su grafo interno del invernadero.

`CAPTURE_IMAGES[plantId]`

Acción repetida para cada planta del segmento:

- captura imágenes según `parametrosCaptura`,
- guarda metadatos (posición, planta, timestamp),
- acumula contenido para su reporte.

`UPDATE_LOG[missionId]`

Actualiza la bitácora interna:

- plantas inspeccionadas,

- imágenes capturadas,
- segmentos procesados,
- timestamps de avance.

### 3. Subsistema de recálculo de ruta (manejo de bloqueos)

Durante EXPLORE\_SEGMENT, el ScoutBot puede detectar que un segmento de la ruta actual es inalcanzable. Esto activa el subsistema mostrado en el AIP.

Detonante: camino bloqueado

El ScoutBot detecta:

- pasillo obstruido,
- segmento inaccesible,
- pérdida de conectividad en la ruta planificada.

Estado (del subsistema)

- Segmento inalcanzable en la ruta calculada.

Acción (del subsistema)

- Con el grafo, recalcular ruta.

El subsistema utiliza el grafo greenhouseGraph para generar una nueva ruta alternativa.

Retorno: nuevo camino

El resultado se devuelve al ScoutBot, que actualiza su ruta y retoma su ciclo de exploración.

`inform(SEGMENT_SKIPPED[segmentId, motivo]) → /Patólogo`

Si incluso después del recálculo un segmento sigue siendo inalcanzable, el ScoutBot lo reporta.

Esto permite al Patólogo saber qué zonas quedaron sin evidencia.

#### 4. Cierre de misión (resultado final)

Al terminar la exploración, el AIP llega a un segundo XOR que separa las salidas posibles:

Opción A — Misión completada

`inform(MISSION_COMPLETED[scoutId, resumenCobertura]) → /Patólogo`

El ScoutBot resume:

- segmentos inspeccionados,
- segmentos omitidos,
- conteo y metadatos de imágenes,
- tiempos y progreso total.

Opción B — Misión abortada

`failure(MISSION_ABORTED[scoutId, error]) → /Patólogo`

Usada cuando ocurre:

- falla de hardware,
- batería crítica,
- error interno que impide continuar.

Solo una de las dos rutas se ejecuta por misión.

## 5. Sincronización y liberación del agente

Tras el resultado final, el protocolo se cierra con dos mensajes:

confirm(RECEIVED\_BATCH[batchId]) ← /Patólogo

El Patólogo confirma que recibió y procesó los datos del ScoutBot.

inform(AVAILABLE[scoutId]) → /Patólogo

El ScoutBot vuelve a estado idle y queda disponible para una nueva misión.

## 6. Lista de mensajes ACL del AIP del ScoutBot

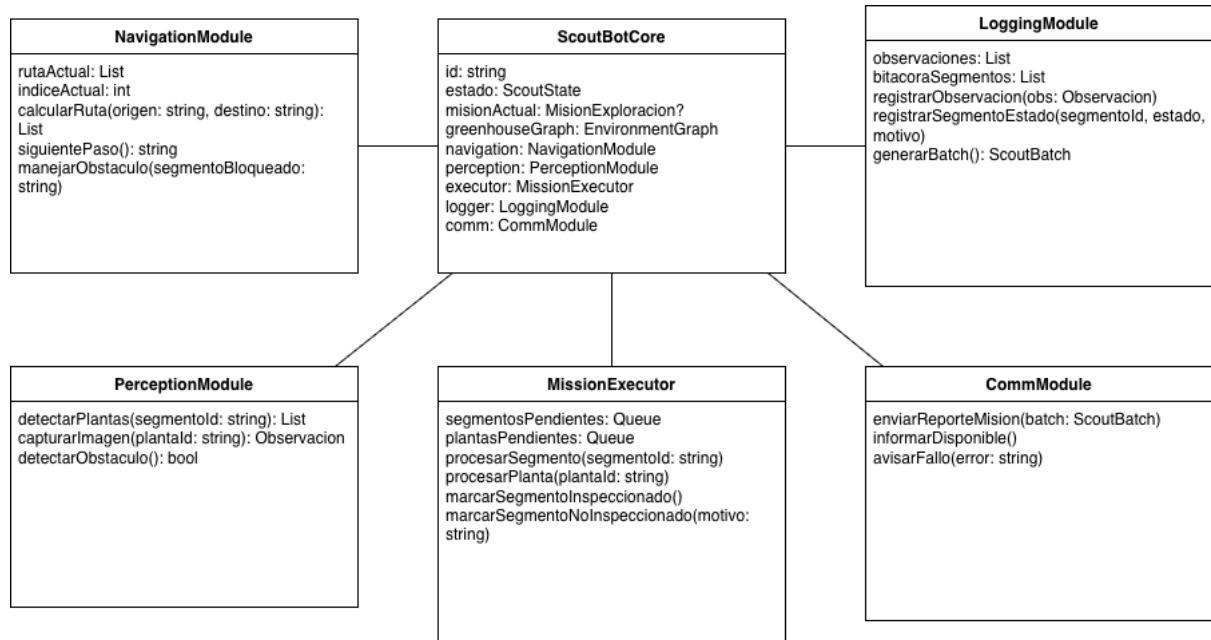
Entradas del ScoutBot

- request(ASSIGN\_MISSION[missionId, segmentos, parametrosCaptura])
- confirm(RECEIVED\_BATCH[batchId])

Salidas del ScoutBot

- inform(MISSION\_ACCEPTED[scoutId, segmentosAsignados])
- refuse(ASSIGN\_MISSION[reason])
- inform(SEGMENT\_SKIPPED[segmentId, motivo])
- inform(MISSION\_COMPLETED[scoutId, resumenCobertura])
- failure(MISSION\_ABORTED[scoutId, error])
- inform(AVAILABLE[scoutId])

## Subsistemas



Este diagrama modela al Scout como un agente reactivo pero estructurado, centrado en navegar, percibir, ejecutar la misión y reportar.

- **ScoutBotCore**

Núcleo del agente explorador.

- Mantiene el estado, la misionActual y referencia al EnvironmentGraph.
- Coordina al módulo de navegación, percepción, ejecución de misión, logging y comunicación.
- Implementa el ciclo principal (recibirMision, iniciarMision, tick, finalizarMision).

- **NavigationModule**

Se encarga de calcular y seguir rutas en el grafo.

- Guarda la rutaActual y el indiceActual.
- Calcula rutas entre segmentos (utilizando el EnvironmentGraph).

- Responde al manejo de obstáculos recalculando rutas o marcando segmentos bloqueados.
- PerceptionModule

Encapsula la percepción del entorno.

  - Detecta plantas en el segmento actual.
  - Captura imágenes de cada planta y devuelve Observacion.
  - Puede detectar obstáculos físicos que obligan a actualizar la ruta o informar al Patólogo.
- MissionExecutor

Implementa la lógica paso a paso de la misión.

  - Mantiene colas de segmentosPendientes y plantasPendientes.
  - Decide qué segmento/planta procesar a continuación.
  - Marca segmentos como inspeccionados o no inspeccionados (con motivo).
- LoggingModule

Gestiona la bitácora de la misión.

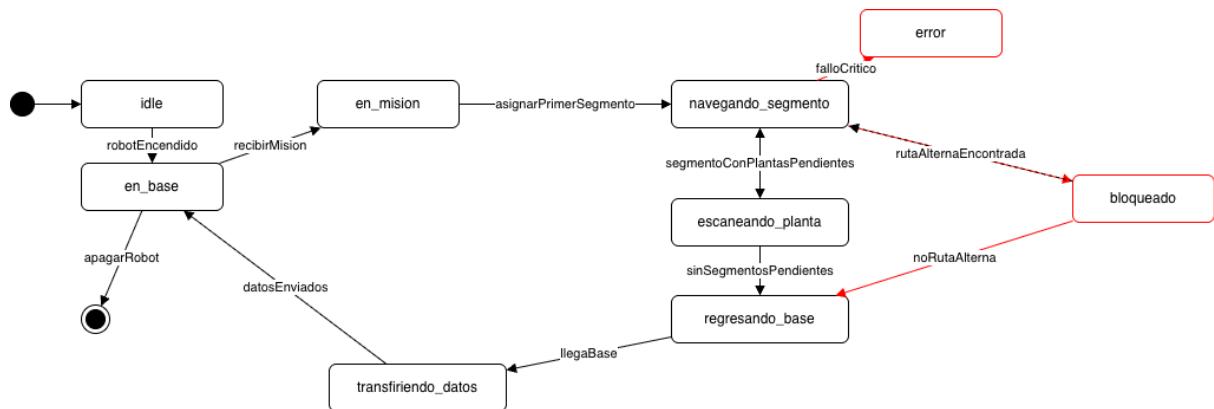
  - Guarda todas las observaciones tomadas.
  - Lleva una bitacoraSegmentos con el estado de cada uno.
  - Genera el ScoutBatch que se enviará al Patólogo al terminar la misión.
- CommModule

Capa de comunicación del Scout.

  - Envía el reporte de misión (ScoutBatch) al Patólogo.
  - Informa que está disponible de nuevo.
  - Notifica fallos graves (avisarFallo) si la misión se aborta.

Este diagrama deja claro que el ScoutBot está construido alrededor de 4 capacidades clave: navegar, percibir, ejecutar la misión y reportar, todas orquestadas por un core simple.

## Estados



Este diagrama de estado describe el ciclo de vida completo del ScoutBot durante una misión de exploración, visto como una máquina de estados.

Estados principales:

- `idle`

Robot apagado / fuera de servicio. No participa en la simulación activa.

- `en_base`

Robot encendido, en la base, sin misión activa pero listo para recibir una.

Es el “estado estacionario” entre misiones.

- `en_mision`

Misión aceptada y cargada. El Scout ya sabe qué segmentos debe inspeccionar, pero aún está en fase de organización inicial (por ejemplo, preparando colas de segmentos).

- navegando\_segmento

El robot se está moviendo hacia el segmento objetivo (o entre segmentos) usando el grafo de invernadero.

Aquí se usan las rutas calculadas por el módulo de navegación y se pueden detectar obstáculos.

- escaneando\_planta

El Scout está tomando fotos/observaciones en el segmento actual.

Recorre las plantas del segmento, captura imágenes y las envía al módulo de logging.

- regresando\_base

Una vez que ya no hay segmentos pendientes (o la misión debe terminar), el Scout vuelve a la base con la misión completada o incompleta (según los casos).

- transfiriendo\_datos

El Scout entrega al Patólogo el ScoutBatch con todas las observaciones de la misión.

Aquí se completa el AIP de reporte de misión.

- bloqueado

Estado excepcional donde el Scout detecta que no puede continuar por un obstáculo:

- Puede intentar recalcular ruta.
- Si no hay ruta alterna, marca el segmento como no inspeccionado y termina la misión.

- error

Estado de fallo crítico (por ejemplo, fallo de hardware, error grave de

navegación).

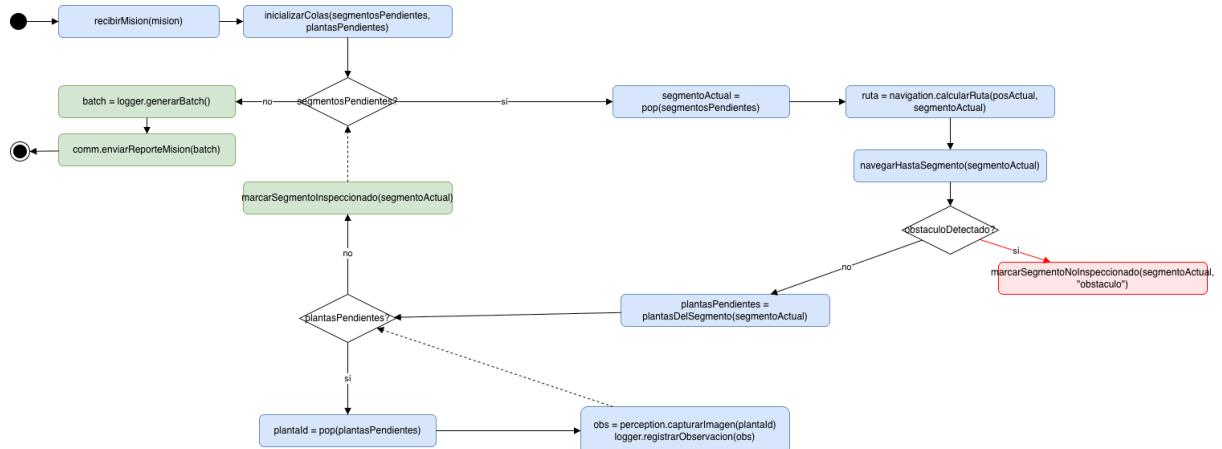
Normalmente implica abortar misión y notificar al Patólogo.

Transiciones clave:

- init → idle → en\_base: arranque del robot.
- en\_base → en\_mision: el Patólogo asigna una misión (AIP AsignarMisionScout).
- en\_mision → navegando\_segmento: el ejecutor selecciona el primer segmento y pide ruta.
- navegando\_segmento ↔ escaneando\_planta: alternancia entre llegar al segmento y escanear plantas.
- navegando\_segmento → bloqueado: al detectar obstáculo.
- bloqueado → navegando\_segmento si hay ruta alterna; o bloqueado → regresando\_base si no.
- escaneando\_planta → regresando\_base: cuando ya no hay segmentos pendientes.
- regresando\_base → transfiriendo\_datos → en\_base: cierre de misión y vuelta a estado listo.
- en\_base → final: apagado del robot.
- Desde estados operativos → error: en caso de fallo crítico.

Este diagrama sirve para mostrar de manera compacta cómo se comporta el agente completo a nivel macro, y cómo se relacionan las fases de movimiento, exploración y reporte.

## Actividad



Este diagrama de actividad se enfoca en el flujo interno del subsistema MissionExecutor del ScoutBot: qué pasos sigue para ejecutar una misión una vez que la misión ya fue asignada.

Flujo principal:

1. recibirMision(mision)

El MissionExecutor recibe la misión desde el ScoutBotCore (lista de segmentos a inspeccionar).

2. Inicializar colas

Actividad:

`inicializarColas(segmentosPendientes, plantasPendientes)`

- Se llenan las estructuras `segmentosPendientes` con los segmentos asignados.
- `plantasPendientes` se deja vacía al inicio.

3. Decisión: ¿hay segmentos pendientes?

- sí: se toma el siguiente segmento.

- o no: se da por terminada la misión y se genera el batch de datos.

#### 4. Seleccionar segmento y calcular ruta

- o Actividad: segmentoActual = pop(segmentosPendientes)
- o Actividad: ruta = navigation.calcularRuta(posActual, segmentoActual)

Aquí se coordina con el módulo de navegación para planear el movimiento.

#### 5. Navegar hacia el segmento

Actividad: navegarHastaSegmento(segmentoActual)

El robot se mueve físicamente en el grafo de invernadero hasta llegar al segmento objetivo.

#### 6. Decisión: ¿obstáculo detectado?

- o Si sí:
  - Actividad: marcarSegmentoNoInspeccionado(segmentoActual, "obstaculo")
  - Regresa al paso 3 (siguiente segmento).
- o Si no:
  - Continúa al escaneo de plantas.

#### 7. Cargar plantas del segmento

Actividad: plantasPendientes = plantasDelSegmento(segmentoActual)

Se llena la cola de plantas del segmento actual.

#### 8. Decisión: ¿hay plantas pendientes?

- o sí:
  - Actividad: plantald = pop(plantasPendientes)
  - Actividad: obs = perception.capturarImagen(plantald)
  - y logger.registrarObservacion(obs)

- Despues de registrar la observación, regresa al mismo punto de decisión de plantas (loop).

- no:
  - El segmento se considera cubierto.

#### 9. Marcar segmento inspeccionado

Actividad: marcarSegmentoInspeccionado(segmentoActual)

Confirma que ese segmento quedó completado y vuelve al paso 3 (próximo segmento).

#### 10. Generar lote y enviar reporte

Cuando ya no hay segmentos:

- Actividad: batch = logger.generarBatch()
- Actividad: comm.enviarReporteMision(batch)
- El flujo termina (final de la actividad).

Qué deja claro este diagrama:

- Cómo MissionExecutor coordina:
  - Navegación (NavigationModule),
  - Percepción (PerceptionModule),
  - Registro (LoggingModule),
  - Comunicación (CommModule).
- Dónde se manejan:
  - Casos normales (todos los segmentos y plantas recorridas),
  - Casos excepcionales (segmentos no inspeccionados por obstáculo).
- El patrón de “bucle doble”:
  - Bucle externo sobre segmentos,

- Bucle interno sobre plantas de cada segmento.

Este diagrama es una vista algorítmica del comportamiento del subsistema y complementa al diagrama de estado.

## Agente PickBot Sanos

### Diagrama de clase

| PickBot Sanos / HealthyPickBot<br>->agent>   |
|--|
| <<state-description>><br>id: string<br>estado: {EnEspera, RecibiendoMision, Navegando, Cosechando, Lleno, RegresandoABase, Descargando, Error}<br>harvestZones: List<Segmento><br>currentLoad: float<br>maxCapacity: float<br>status: string<br>greenhouseGraph: EnvironmentGraph<br>segmentoActualId: string<br>plantasPendientesEnSegmento: List<Planta><br>frutosCosechados: List<FrutoID><br>posicionActual: Vector3<br>basePosicion: Vector3<br>zonaDescargaPosicion: Vector3<br>rutaActual: List<Segmento><br>bateriaNivel: float<br>timestampUltimoReporte: float |
| <<actions>><br>+ procesarAsignacionCosecha(zonas: List<Segmento>)<br>+ calcularRuta()<br>+ cosecharFrutosSanos(plantaid: string)<br>+ descargarCosecha()<br>+ regresarABase()<br>+ informarEstado()<br>+ enviarMensaje(destino: string, msgACL: ACLMessage)  |
| <<roles>><br>/PickBotSanos<br>/CosechaSegura   |
| <<protocols>><br>AIP_CosechaSegura<br>AIP_SincronizarEnBase  |
| <<messages>><br>request(REQUEST_MISSION_HARVEST[zonas]) ← /Patologo<br>inform(CONFIRM_MISSION_ACCEPTED[sanosId, zonas]) → /Patologo<br>inform(INFORM_HARVEST_PROGRESS[sanosId, segmentId, avance]) → /Patologo<br>inform(INFORM_SEGMENT_UNHARVESTED[segmentId, motivo]) → /Patologo<br>inform(INFORM_HARVEST_COMPLETED[reporte]) → /Patologo<br>inform(INFORM_AGENT_AVAILABLE[sanosId]) → /Patologo<br>failure(FAILURE_MISSION_ABORTED[sanosId, error]) → /Patologo  |

Es un agente autónomo cuyo rol es actuar como cosechador de frutos sanos en el invernadero: recibe misiones de cosecha en zonas seguras, navega por los segmentos asignados, cosecha los frutos y los lleva a la zona de descarga/base.

El diagrama se divide en:

- <<state-description>> → qué sabe / qué recuerda (estado interno).
- <<actions>> → qué puede hacer (operaciones de alto nivel).
- <<roles>> → papel que juega dentro de la organización multiagente.
- <<protocols>> → en qué AIPs participa.
- <<messages>> → mensajes ACL típicos que envía/recibe.

## 1. Bloque <<state-description>> (estado interno)

- id
  - Identificador único del PickBot de sanos (por ejemplo, "Harvester\_01").
  - estado

Máquina de estados de alto nivel:

  - Idle: en espera de una nueva misión de cosecha.
  - RecibiendoMision: procesando el mensaje de asignación de zonas seguras.
  - Navegando: moviéndose hacia un segmento o planta objetivo.
  - Cosechando: ejecutando la acción de cosecha en una planta.
  - Lleno: la carga actual alcanzó la capacidad máxima (maxCapacity).
  - RegresandoABase: volviendo a base o zona de descarga.
  - Descargando: transfiriendo físicamente los frutos a la zona de descarga.
  - Error: algún fallo crítico detuvo la misión.
  - harvestZones
    - Lista de zonas o segmentos seguros que el Patólogo asignó para cosechar, típicamente derivada del resultado de diagnóstico (plantas marcadas como sanas).
  - currentLoad

- Cantidad de carga actual (por ejemplo, kg de frutos o unidades cosechadas). Se incrementa al cosechar y se revisa para saber si el bot está “Lleno”.
- maxCapacity
  - Capacidad máxima de carga del PickBot en una misión (por ejemplo, 15.0 kg).
  - Cuando  $\text{currentLoad} \geq \text{maxCapacity}$ , el bot debe dejar de cosechar y pasar a RegresandoABase.
- status
  - Texto corto de estado (“Cosechando segmento S3”, “Descargando en zona segura”, etc.) útil para logging o para una UI de monitoreo.
- greenhouseGraph
  - Referencia al grafo que modela los pasillos, segmentos y conexiones del invernadero.
  - Se usa para planear rutas y construir rutaActual.
- segmentoActualId
  - Identificador del segmento en el que está trabajando actualmente (por ejemplo, "RowA\_03\_Left").
- plantasPendientesEnSegmento
  - Lista de plantas sanas en el segmento actual que aún faltan por cosechar. Se va vaciando conforme el bot cosecha.
- frutosCosechados
  - Registro de los frutos que ya fueron cosechados (puede guardar frutoid, plantaid, segmentoid, etc.).
  - Sirve para formar el reporte final HARVEST\_DONE.

- posicionActual, basePosicion (Vector3)
  - posicionActual: posición 3D del bot en el mundo virtual (Unity).
  - basePosicion: posición 3D de la base o docking station donde inicia/termina misión.
- zonaDescargaPosicion (Vector3)
  - Ubicación física de la zona de descarga (banda, contenedor, área de empaquetado, etc.) donde entrega los frutos cosechados.
- rutaActual
  - Secuencia de nodos/waypoints que el bot está siguiendo: puede ser desde base → segmento, entre segmentos, o segmento → zona de descarga.
- bateriaNivel
  - Porcentaje (0–100) o valor normalizado de la batería.
  - Si baja de cierto umbral, puede provocar que el bot cierre misión antes de tiempo (HARVEST\_ABORTED) y regrese a base.
- timestampUltimoReporte
  - Timestamp del último reporte enviado al Patólogo.
  - Permite manejar timeouts (“hace mucho que no sabemos nada de este bot”).

### 3. Bloque <>actions<> (comportamiento)

Estas son las operaciones de alto nivel que definen el comportamiento del PickBot de sanos:

- procesarAsignacionCosecha(listaZonas)

- Procesa el request(HARVEST\_SAFE\_ZONES[...]) que viene del Patólogo.
- Guarda las zonas en harvestZones y cambia el estado de Idle → RecibiendoMision → Navegando.
- calcularRuta()

Usa greenhouseGraph para planear la rutaActual:

- Desde base a la primera zona de cosecha.
- Entre zonas dentro de la lista harvestZones.
- De la última zona a zonaDescargaPosicion o basePosicion.
- cosecharFrutosSanos(plantId)

Representa la acción de cosecha en una planta sana:

- Marca esa planta como atendida dentro de plantasPendientesEnSegmento.
- Agrega los frutos correspondientes a frutosCosechados.
- Actualiza currentLoad.
- Si se alcanza maxCapacity, cambia el estado a Lleno y prepara el regreso.
- descargarCosecha()

Simula la descarga física de frutos en zonaDescargaPosicion:

- Vacía o marca como descargados los elementos de frutosCosechados.
- Resetea currentLoad a 0.
- Cambia a un estado apropiado, por ejemplo Descargando → Idle o Descargando → RecibiendoMision.
- regresarABase()

- Cambia el estado a RegresandoABase, calcula una ruta desde donde se encuentre el bot hasta basePosicion o zonaDescargaPosicion (según el diseño), y navega hasta ahí.
- informarEstado()

Centraliza la lógica de reporte hacia el Patólogo:

- Progreso de cosecha por segmento.
- Estado actual (estado).
- Carga, batería, etc.
- enviarMensaje(destino, msgACL)
  - Método genérico para enviar mensajes ACL a otros agentes (en la práctica, principalmente al Patólogo Digital).

#### 4. Bloque <>roles>>

Indica en qué papeles organizacionales participa:

- /PickBotSanos → tipo de agente, especializado en cosecha de frutos sanos.
- /CosechaSegura → rol específico dentro del sistema, responsable de explotar las zonas confirmadas como seguras después de que el Patólogo y los Scout/Nurse Bots han descartado enfermedad.

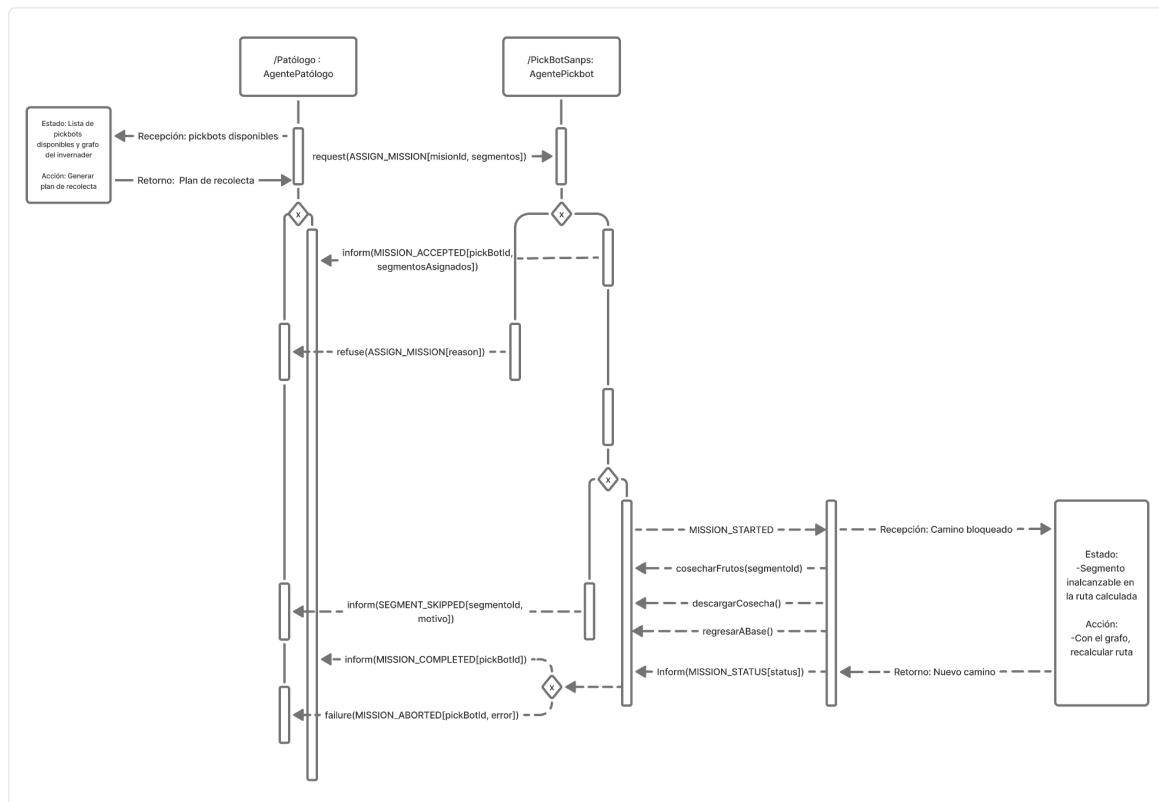
Este rol se activa típicamente en la etapa final del flujo: “todo lo que está marcado como sano puede enviarse al mercado”.

#### 5. Bloque <>protocols>>

Aquí se enumeran los AIPs (Agent Interaction Protocols) en los que participa el PickBot de sanos:

## AIP

- AIP\_CosechaSegura ()
  - Protocolo en el que el Patólogo envía la orden de cosecha de zonas seguras a uno o varios PickBots Sanos.
  - Incluye la negociación/aceptación de la misión (HARVEST\_ACCEPTED).



El AIP\_CosechaSegura coordina la interacción entre el AgentePatólogo y los PickBots Sanos encargados de recolectar frutos en segmentos confirmados como seguros. El protocolo describe cómo el Patólogo asigna una misión de cosecha, cómo los PickBots aceptan o rechazan la tarea, cómo ejecutan la recolección

siguiendo un plan, cómo manejan segmentos inaccesibles mediante un subsistema de recálculo, y cómo reportan el resultado final de la misión.

El AIP se divide conceptualmente en:

- Fase de generación del plan de recolección → donde el Patólogo consulta la disponibilidad de PickBots y genera el plan de cosecha.
- Fase de asignación de misión → donde el Patólogo envía la misión de cosecha al PickBot Sano.
- Fase de ejecución de cosecha → donde el PickBot navega por los segmentos, cosecha frutas, descarga la cosecha y reporta progreso.
- Subsistema de recálculo de ruta → mecanismo que se activa si un segmento asignado resulta inalcanzable.
- Fase de cierre → donde el PickBot reporta misión completada, abortada o segmentos omitidos.
- Mensajes ACL → representan el flujo formal de comunicación entre agentes.

### 1. Generación de plan de recolección (estado interno del Patólogo)

Antes de iniciar el protocolo, el Patólogo:

- recopila la lista de PickBots disponibles,
- analiza el grafo del invernadero,
- identifica los segmentos seguros para cosecha,
- determina qué PickBot puede atender qué segmento.

El cuadro del diagrama refleja esta fase:

Estado: Lista de pickbots disponibles y grafo del invernadero

Acción: Generar plan de recolecta

Esta preparación antecede al envío del mensaje ASSIGN\_MISSION.

## 2. Asignación de misión (request → accept/refuse)

request(ASSIGN\_MISSION[missionId, segmentos]) ← /Patólogo

El AIP inicia cuando el Patólogo envía una misión de cosecha al PickBot Sano, indicando:

- missionId: identificador de la misión de cosecha.
- segmentos: lista de segmentos de plantas sanas que deben cosecharse.

Esto coloca al PickBot en fase de evaluación.

(XOR) Decisión del PickBot Sano

El PickBot revisa:

- su nivel de batería,
- si está libre (estado = Idle),
- si puede desplazarse a los primeros segmentos,
- si su carga actual permite comenzar una misión.

Solo una rama ocurre:

inform(MISSION\_ACCEPTED[pickBotId, segmentosAsignados]) → /Patólogo

El PickBot acepta la misión, guarda los segmentos asignados y prepara su navegación.

`refuse(ASSIGN_MISSION[reason]) → /Patólogo`

El PickBot rechaza la misión por motivos como batería baja, carga actual saturada o falla en sistema de navegación.

El protocolo termina para esa instancia.

### 3. Ejecución de misión y recolección (navegación → cosecha → descarga)

Tras aceptar la misión, el PickBot entra en fase operativa.

`MISSION_STARTED`

Marca el inicio del proceso de cosecha.

#### 1) cosecharFrutos(segmentId)

El PickBot:

- navega hacia el segmento asignado,
- identifica plantas sanas,
- cosecha frutos,
- incrementa currentLoad,
- actualiza el registro de frutos cosechados.

Si  $\text{currentLoad} \geq \text{maxCapacity}$ , detiene cosecha y pasa a descargar.

#### 2) descargarCosecha()

El PickBot se dirige a base o zona designada y descarga los frutos cosechados.

Esta acción reinicia su capacidad de carga.

### 3) regresarABase()

Una vez finalizada la misión:

- el PickBot regresa a base o docking station,
- actualiza su posición,
- cierra la misión internamente.

### 4) Inform(MISSION\_STATUS[status])

Durante la misión, el PickBot reporta su avance al Patólogo:

- segmento actual,
- carga actual,
- progreso de recolecta,
- fallos detectados.

## 4. Subsistema de recálculo de ruta (manejo de bloqueos)

Durante su navegación o recolección, el PickBot puede encontrar un segmento inalcanzable (pasillo bloqueado, planta inaccesible, nodo dañado). El diagrama incluye el subsistema de recálculo igual que en el ScoutBot.

Detonante:

Recepción de camino bloqueado

Se activa si el PickBot detecta que no puede llegar al segmento o ruta asignada.

Estado del subsistema

- Segmento inalcanzable en la ruta calculada.

Acción del subsistema

- Con el grafo, recalcular ruta.

El subsistema reconstruye una nueva ruta usando el grafo del invernadero.

Retorno:

Nuevo camino

El PickBot retoma su misión siguiendo el camino alternativo.

`inform(SEGMENT_SKIPPED[segmentId, motivo]) → /Patólogo`

Si el segmento sigue siendo inaccesible después del recálculo, el PickBot lo marca como omitido y lo reporta al Patólogo.

## 5. Cierre de misión (resultado final)

Un último XOR del AIP separa las dos rutas de cierre posibles:

Opción A — Misión completada

`inform(MISSION_COMPLETED[pickBotId]) → /Patólogo`

El PickBot finaliza la recolección y descarga de frutos correctamente.

Opción B — Misión abortada

`failure(MISSION_ABORTED[pickBotId, error]) → /Patólogo`

Utilizado cuando:

- ocurre una falla crítica del hardware,

- batería insuficiente que no permite continuar,
- daño en motores o actuadores,
- inconsistencia interna en el plan de misión.

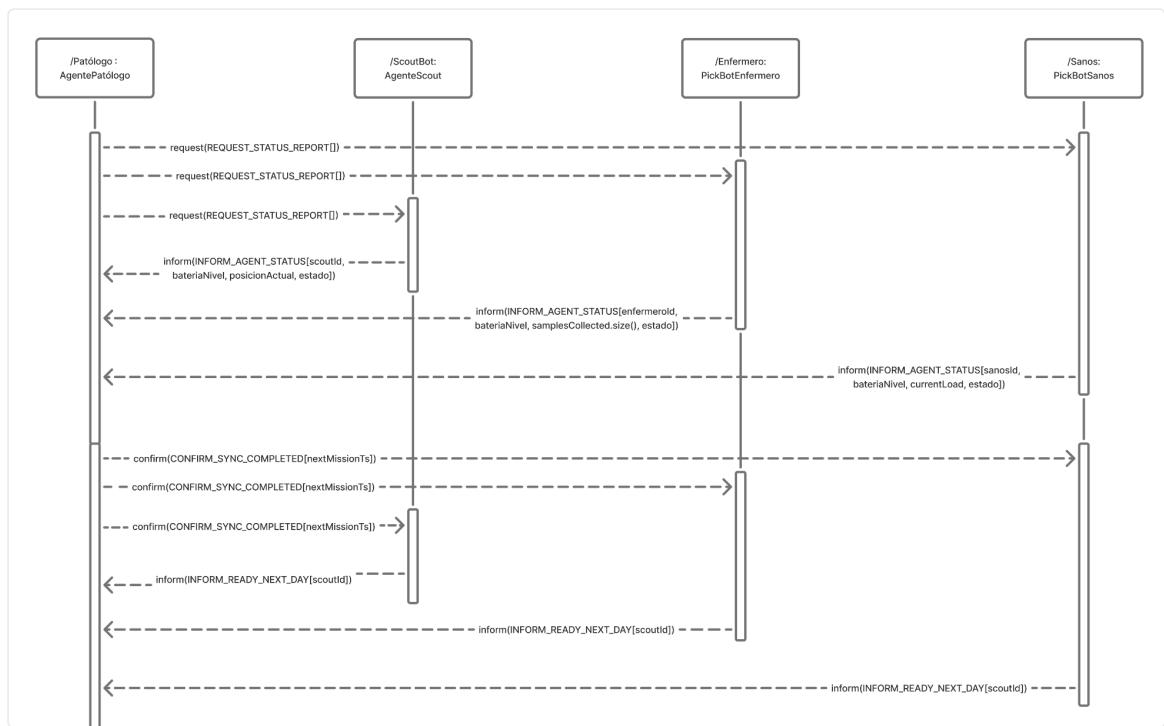
## 6. Lista de mensajes ACL del AIP\_CosechaSegura

### Mensajes de entrada hacia el PickBot

- request(ASSIGN\_MISSION[missionId, segmentos])

### Mensajes de salida del PickBot

- inform(MISSION\_ACCEPTED[pickBotId, segmentosAsignados])
- refuse(ASSIGN\_MISSION[reason])
- inform(MISSION\_STATUS[status])
- inform(SEGMENT\_SKIPPED[segmentoid, motivo])
- inform(MISSION\_COMPLETED[pickBotId])
- failure(MISSION\_ABORTED[pickBotId, error])



- AIP\_SincronizarEnBase ()
  - Protocolo dedicado a la entrega de la cosecha y reporte final (HARVEST\_DONE), además de notificar disponibilidad (HARVESTER\_AVAILABLE).

El AIP\_SincronizarEnBase() coordina el proceso de sincronización de estado entre el AgentePatólogo y los diferentes robots operativos del invernadero (ScoutBots, PickBots Enfermeros y PickBots Sanos). Su objetivo es recopilar el estado actualizado de cada agente al final de una jornada, validar su preparación para la siguiente misión y confirmar que todos han completado la sincronización correspondiente.

Este AIP se divide conceptualmente en:

- Fase de solicitud de estado → donde el Patólogo inicia la sincronización enviando solicitudes de reporte.
- Fase de reporte de agentes → donde cada robot responde con su estado actual (batería, posición o métricas relevantes).
- Fase de confirmación → donde el Patólogo analiza los reportes y envía confirmaciones de sincronización completa.
- Fase de preparación para el siguiente día → donde los agentes informan que están listos para una nueva jornada.

- Mensajes ACL → representan el flujo formal de comunicación.

## 1. Solicitud de estado (request → reportes de agentes)

request(REQUEST\_STATUS\_REPORT[]) ← /Patólogo

El AIP inicia cuando el Patólogo envía una solicitud de estado a cada tipo de agente:

- ScoutBots,
- PickBots Enfermeros,
- PickBots Sanos.

Este mensaje indica que se está por iniciar la fase de sincronización global del sistema.

El objetivo del Patólogo es obtener un snapshot del estado operativo de todos los robots antes de asignar nuevas misiones en la siguiente jornada.

## 2. Reporte de estado de agentes (inform → estado del agente)

Una vez recibida la solicitud del Patólogo, cada agente responde con su estado actual.

inform(INFORM\_AGENT\_STATUS[scoutId, bateriaNivel, posicionActual, estado]) →  
/Patólogo

Los ScoutBots envían:

- nivel de batería,
- posición dentro del invernadero,
- estado actual (idle, en\_base, cargando, etc.).

```
inform(INFORM_AGENT_STATUS[enfermeroid,  
samplesCollected.size(), estado]) → /Patólogo  
  
bateriaNivel,
```

Los PickBots Enfermeros envían:

- batería,
- cantidad de muestras recolectadas,
- estado.

```
inform(INFORM_AGENT_STATUS[sanosId, bateriaNivel, currentLoad, estado]) →  
/Patólogo
```

Los PickBots Sanos reportan:

- batería,
- carga actual,
- estado.

El Patólogo usa esta información para confirmar que los tres tipos de agentes han completado sus actividades previas y están listos para transición de jornada.

### 3. Confirmación de sincronización (confirm → validación del Patólogo)

```
confirm(CONFIRM_SYNC_COMPLETED[nextMissionTs]) ← /Patólogo
```

Una vez recopilados todos los reportes, el Patólogo verifica:

- que ningún agente presente fallas,
- que las baterías estén dentro de rangos tolerables,
- que no existan tareas pendientes,

- que el sistema está listo para avanzar al siguiente ciclo operativo.

Posteriormente, envía una confirmación de sincronización completada a todos los agentes participantes:

- ScoutBots
- PickBots Enfermeros
- PickBots Sanos

El parámetro nextMissionTs especifica el timestamp o ciclo en el que comenzará la siguiente jornada de misiones.

Este mensaje es la señal oficial de que la fase de sincronización ha terminado.

#### 4. Preparación para la siguiente jornada

Una vez recibida la confirmación de sincronización, cada agente transita a estado de reposo y notifica al Patólogo que está listo para iniciar un nuevo ciclo operativo.

`inform(INFORM_READY_NEXT_DAY[scoutId]) → /Patólogo`

Este mensaje indica que el ScoutBot:

- ya procesó la confirmación de sincronización,
- reseteó sus variables internas relevantes,
- completó cualquier procedimiento de apagado suave,
- está disponible para asignación de misiones el siguiente día.

El mismo flujo ocurre para:

- PickBotEnfermero

- PickBotSanos

aunque en el diagrama solo se visualiza detallado para el ScoutBot.

## 5. Lista de mensajes ACL del AIP\_SincronizarEnBase()

Mensajes de entrada hacia los agentes

- request(REQUEST\_STATUS\_REPORT[])
- confirm(CONFIRM\_SYNC\_COMPLETED[nextMissionTs])

Mensajes de salida desde los agentes

- inform(INFORM\_AGENT\_STATUS[scoutId, bateriaNivel, posicionActual, estado])
- inform(INFORM\_AGENT\_STATUS[enfermeroid, bateriaNivel, samplesCollected.size(), estado])
- inform(INFORM\_AGENT\_STATUS[sanosId, bateriaNivel, currentLoad, estado])
- inform(INFORM\_READY\_NEXT\_DAY[scoutId])
- inform(INFORM\_READY\_NEXT\_DAY[enfermeroid])
- inform(INFORM\_READY\_NEXT\_DAY[sanosId])

## 6. Bloque <>messages<> (comunicación ACL)

Muestran los mensajes típicos que intercambia con el Patólogo Digital:

- Mensaje de entrada principal:
  - request(HARVEST\_SAFE\_ZONES[harvestZones]) ← /Patologo

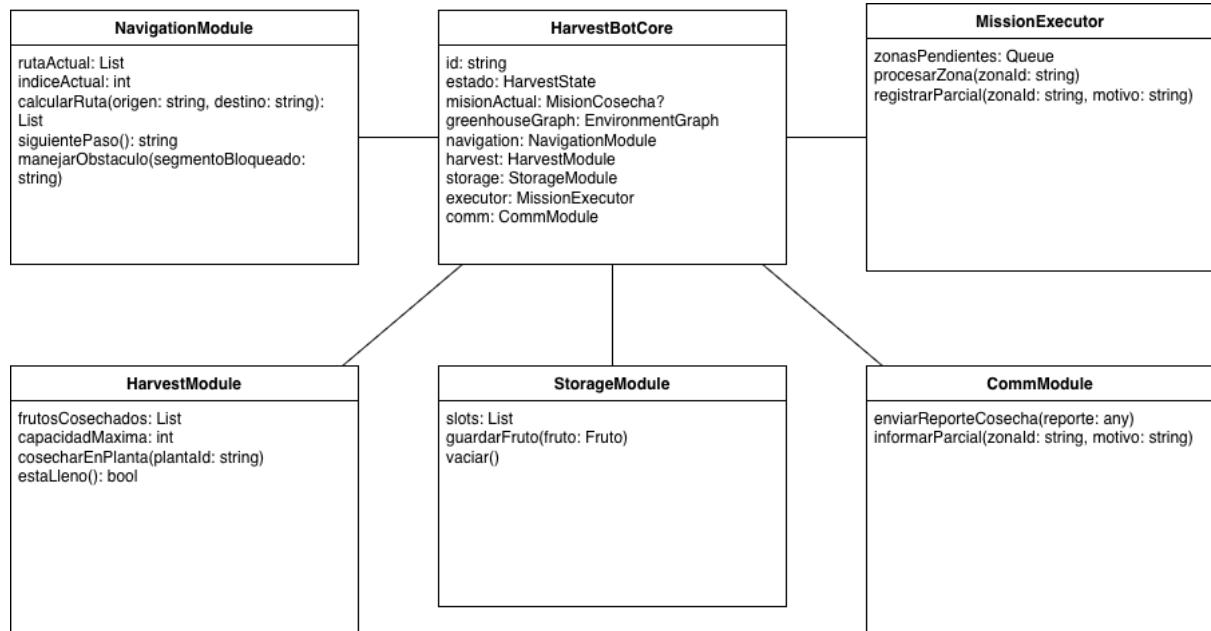
El Patólogo envía la lista de zonas seguras a cosechar.

- Mensajes de salida:
  - inform(HARVEST\_ACCEPTED[harvesterId, harvestZones]) → /Patologo

Confirma que el PickBot acepta la misión y las zonas asignadas.

- inform(HARVEST\_PROGRESS[harvesterId, segmentId, avance]) → /Patologo
  - Reporta progreso de cosecha en un segmento específico (avance podría ser porcentaje, nº de plantas atendidas, etc.).
- inform(SEGMENT\_UNHARVESTED[segmentId, motivo]) → /Patologo
  - Indica que un segmento no pudo cosecharse (obstáculo, riesgo detectado, baja batería, etc.).
- inform(HARVEST\_DONE[reporte]) → /Patologo
  - Resume la cosecha completada: qué zonas se atendieron, cuántos frutos se cosecharon, tiempos, etc.
- inform(HARVESTER\_AVAILABLE[harvesterId]) → /Patologo
  - Indica que el PickBot ha terminado y está disponible para una nueva misión.
- failure(HARVEST\_ABORTED[harvesterId, error]) → /Patologo
  - Reporta que la misión fue abortada por un error (fallo mecánico, batería crítica, bloqueo, etc.).

## Subsistemas



Este diagrama modela el PickBot dedicado a cosechar frutos sanos en zonas seguras, con una estructura muy paralela al enfermero pero orientada a cosecha.

- **HarvestBotCore**

Núcleo del agente de cosecha.

- Mantiene estado, misionActual (MisionCosecha) y el EnvironmentGraph.
- Coordina navegación, cosecha, almacenamiento, ejecución de misión y comunicación.

- **NavigationModule**

Igual que en los otros bots:

- Planea rutas entre zonasSeguras (segmentos o plantas) asignadas en la misión.
- Gestiona desvíos por obstáculos.

- HarvestModule

Módulo específico de cosecha de frutos sanos.

- Gestiona la lista de frutosCosechados y la capacidadMaxima.
- Implementa cosecharEnPlanta(plantId) y estaLleno().

- StorageModule

Almacena los frutos cosechados.

- Similar al del NurseBot, pero con Fruto en lugar de Muestra.
- Permite vaciar el contenedor una vez entregada la cosecha.

- MissionExecutor

Lleva la misión a cabo zona por zona.

- Mantiene zonasPendientes (segmentos o plantas).
- Decide qué zona procesar, llama a HarvestModule y registra parciales o fallos.

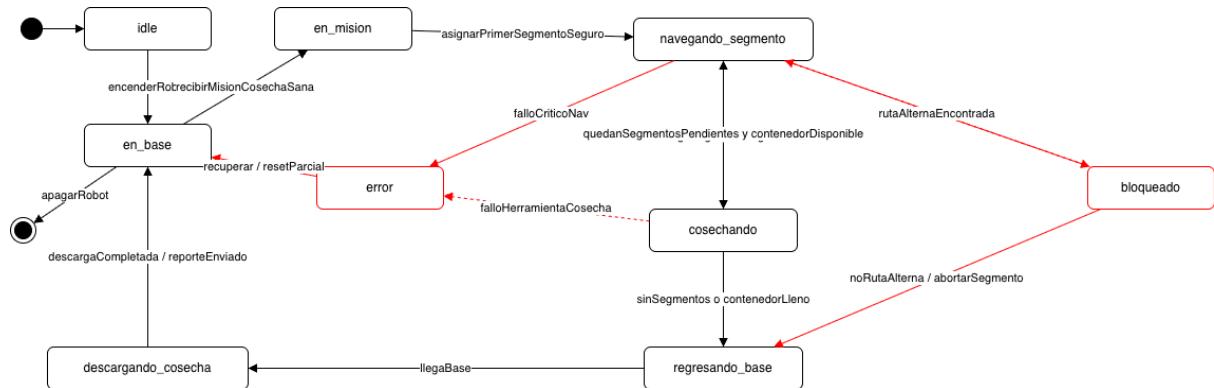
- CommModule

Capa de comunicación específica para cosecha.

- Envía el reporteCosecha al Patólogo (o un resumen de lo cosechado).
- Informa zonas atendidas parcialmente o no atendidas con sus motivos.

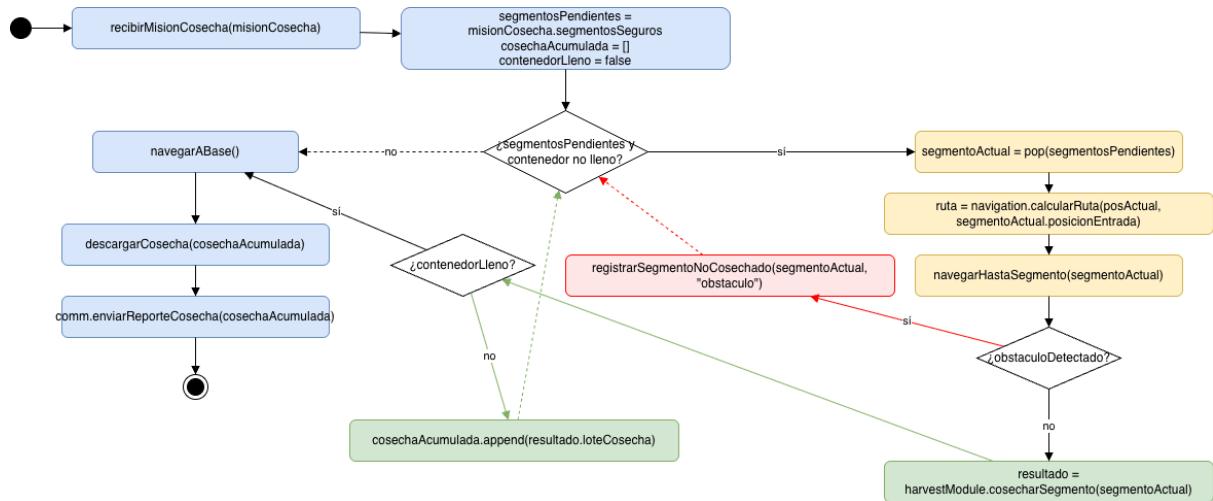
El patrón de este diagrama refuerza que los PickBots comparten una arquitectura común (core + navegación + storage + executor + comm), pero se especializan en el módulo de tarea (SampleCollector vs HarvestModule) y en el tipo de objeto que manejan (Muestra vs Fruto).

## Estados



- Modela el ciclo completo del PickBot de cosecha segura:
  - Se enciende (`idle` → `en_base`), recibe misiones de zonas seguras, entra a `en_mision`.
  - Alterna entre `navegando_segmento` y `cosechando` los segmentos proporcionados por el Patólogo.
  - Maneja obstáculos con estado `bloqueado`.
  - Cuando termina segmentos o se llena el contenedor, regresa a base (`regresando_base`) y se pone en `descargando_cosecha`.
  - Tras descargar y enviar reporte de cosecha, vuelve a `en_base` o se apaga.
- Define también cómo se entra/sale de error por problemas de navegación o herramientas de cosecha.

## Actividad



Esta actividad describe el flujo de ejecución de una misión de cosecha segura:

### 1. Recibir misión de cosecha

`recibirMisionCosecha(misionCosecha).`

### 2. Inicializar estructuras

- `segmentosPendientes = misionCosecha.segmentosSeguros`
- `contenedorLleno = false`
- `cosechaAcumulada = []`

### 3. Decisión: ¿hay segmentos y contenedor no lleno?

- Si sí:
  - Seleccionar segmentoActual.
  - Calcular ruta y navegar hacia él.
  - Manejar obstáculos: marcar segmentoNoCosechado si no hay ruta.
  - Si llega, llamar a `harvestModule.cosecharSegmento(segmentoActual)`.

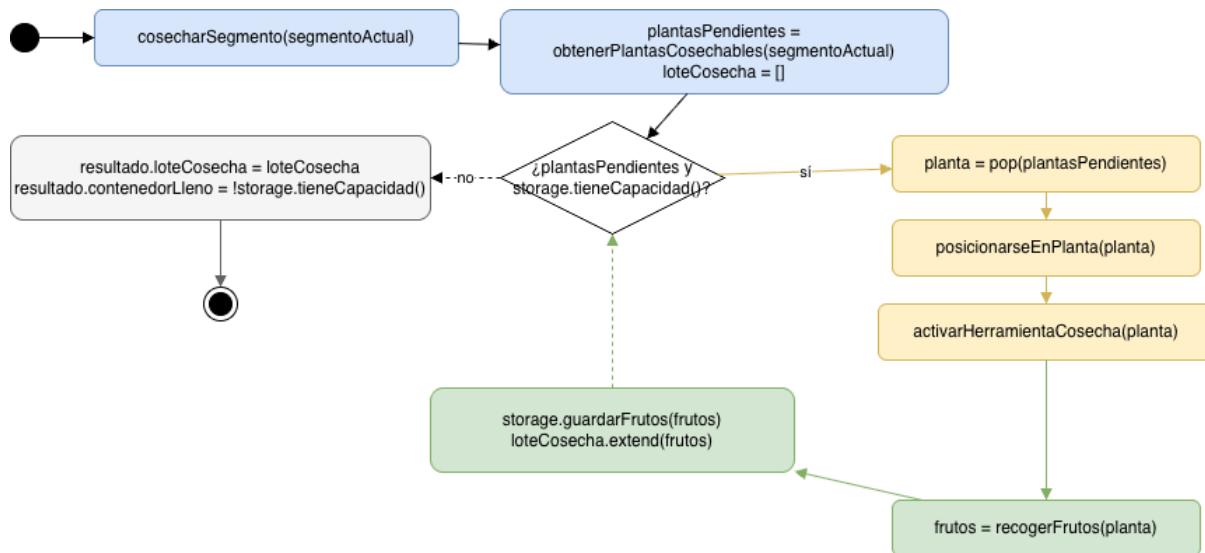
- Si no:
  - Se navega a base para descargar.

#### 4. Resultado de cosecharSegmento

- Devuelve:
  - loteCosecha (frutos/cajas)
  - contenedorLleno actualizado
- Se agrega loteCosecha a cosechaAcumulada.

#### 5. Regresar a base y descargar

- navegarABase()
- descargarCosecha(cosechaAcumulada)
- comm.enviarReporteCosecha(cosechaAcumulada).



Esta actividad describe el flujo interno del módulo de cosecha para un segmento:

#### 1. Recibir segmento

`cosecharSegmento(segmentoActual).`

2. Cargar plantas y/o frutos del segmento

plantasPendientes = obtenerPlantasCosechables(segmentoActual).

3. Loop plantas mientras haya capacidad

- Decisión: ¿hay plantas pendientes y capacidad en contenedor?

- Si sí:

- Seleccionar planta.
    - Posicionarse entre plantas.
    - Activar herramienta de corte/agarre.
    - Tomar frutos (uno o varios).
    - Guardar en contenedor y actualizar contadores.

- Si no:

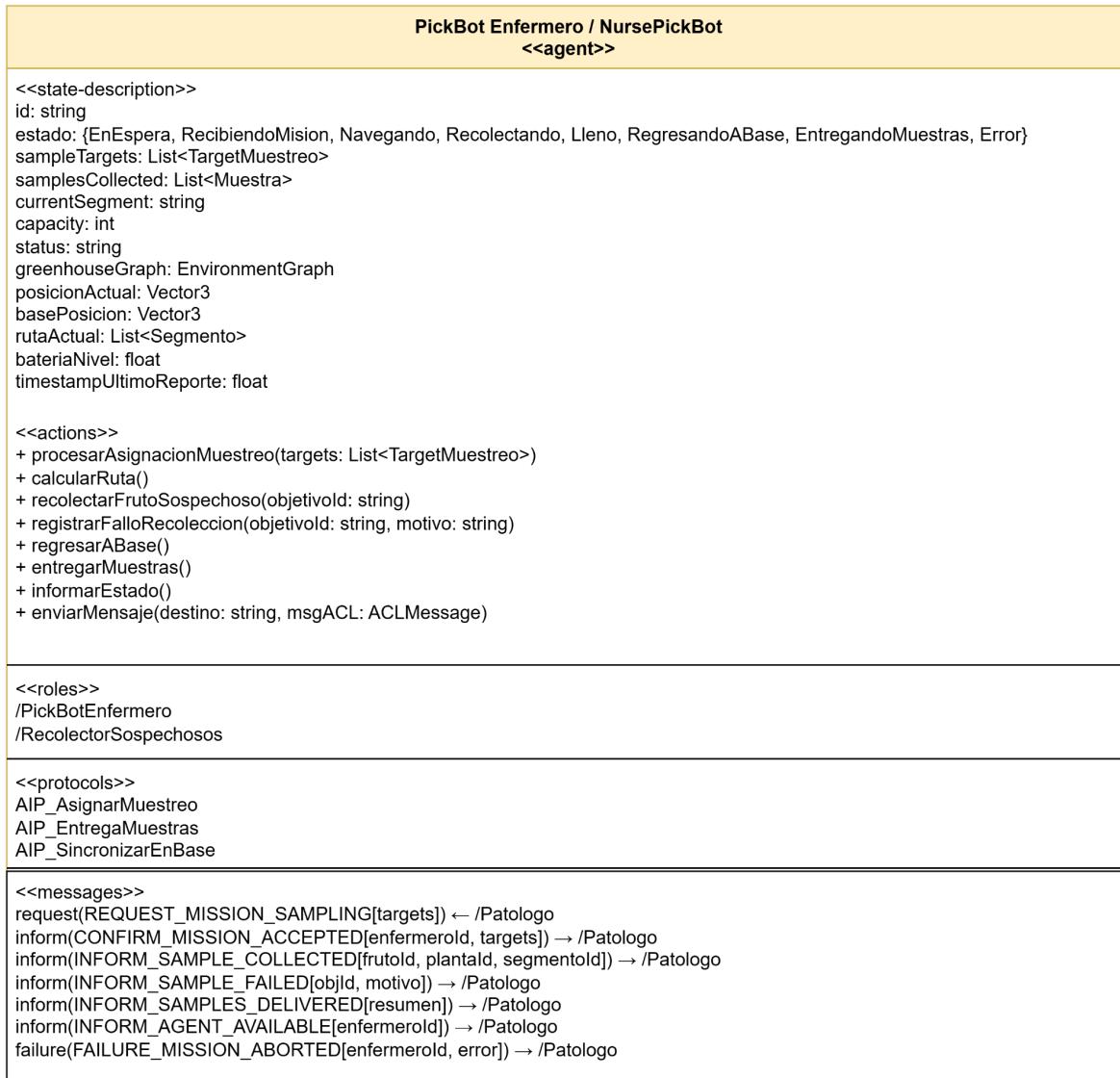
- Terminar segmento.

4. Devolver resultado

resultado.loteCosecha y resultado.contenedorLleno.

## Agente PickBot Enfermero

### Diagrama de clase



Es un agente autónomo cuyo rol es actuar como “enfermero” en el invernadero: recibe misiones de muestreo, navega hasta los frutos sospechosos, toma muestras y las entrega en base.

El diagrama se divide en:

1. <<state-description>> → qué sabe / qué recuerda (estado interno).
2. <<actions>> → qué puede hacer (operaciones de alto nivel).

3. <>roles>> → papel que juega dentro de la organización multiagente.
  4. <>protocols>> → en qué AIPs participa.
  5. <>messages>> → mensajes ACL típicos que envía/recibe.
1. Bloque <>state-description>> (estado interno)
- **id**  
Identificador único del PickBot (por ejemplo, “Nurse\_01”).
  - **estado**  
Máquina de estados de alto nivel:
    - Idle: en espera de nueva misión.
    - RecibiendoMision: procesando el mensaje de asignación.
    - Navegando: moviéndose hacia una planta/fruto objetivo.
    - Recolectando: ejecutando la acción de tomar muestra.
    - Lleno: ya no puede recolectar más (capacidad llena).
    - RegresandoABase: volviendo a la base para entregar.
    - EntregandoMuestras: transfiriendo físicamente las muestras.
    - Error: algún fallo crítico.
  - **sampleTargets**
    - Lista de objetivos de muestreo que el Patólogo asignó. Cada elemento podría ser un struct con plantaId, frutoId, segmentoId.
  - **samplesCollected**
    - Registro de las muestras que ya fueron tomadas. Sirve tanto para trazabilidad como para armar el resumen que se enviará al Patólogo.
  - **currentSegment**
    - Indica en qué segmento del invernadero está trabajando actualmente.
  - **capacity**
    - Capacidad máxima de muestras que puede cargar en una misión (por ejemplo, 20 frutos).
  - **status**
    - Texto corto de estado (“En ruta a segmento X”, “Entregando muestras”, etc.) para logging o interfaz.
  - **greenhouseGraph**

- Referencia al grafo de pasillos/segmentos del invernadero. Se usa para planear rutas (`rutaActual`).
- `posicionActual, basePosicion (Vector3)`
  - Posiciones 3D del bot y de la base/docking station dentro del mundo virtual.
- `rutaActual`
  - Secuencia de nodos/waypoints que está siguiendo en este momento.
- `bateriaNivel`
  - Porcentaje o valor normalizado de batería; puede detonar decisiones como “abort mission & regresar”.
- `timestampUltimoReporte`
  - Timestamp del último reporte enviado al Patólogo (útil para timeouts o heartbeats).

### 3. Bloque <>actions<> (comportamiento)

Estas son las operaciones de alto nivel que definen su comportamiento:

- `procesarAsignacionMuestreo(listaObjetivos)`
  - Procesa el `request (ASSIGN_SAMPLING[ . . . ])` del Patólogo, guarda los objetivos en `sampleTargets` y cambia el estado de `Idle` → `RecibiendoMision` → `Navegando`.
- `calcularRuta()`
  - Usa `greenhouseGraph` para planear la ruta (`rutaActual`) hacia el siguiente objetivo o hacia la base.
- `recolectarFrutoSospechoso(objetivoId)`
  - Se asume que el bot ya está frente a la planta/fruto. Actualiza `samplesCollected`, reduce espacio disponible (`capacity efectiva`) y avanza el estado (por ejemplo, `Recolectando` → `Navegando al siguiente`).
- `registrarFalloRecolección(objetivoId, motivo)`

- Marca un objetivo como no atendido (obstáculo, fruto ausente, etc.) y luego informará al Patólogo mediante un mensaje tipo SAMPLE\_FAILED.
- regresarABase()
  - Cambia el estado a RegresandoABase, recalcula ruta hacia basePosicion y navega.
- entregarMuestras()
  - Simula la entrega física al módulo de análisis. Después limpia samplesCollected o marca cuáles ya fueron entregadas.
- informarEstado()
  - Centraliza la lógica de comunicación de estado (puede mandar progreso, heartbeat, etc.).
- enviarMensaje(destino, msgACL)
  - Método genérico para mandar mensajes ACL a otros agentes.

#### 4. Bloque <<roles>>

Indica en qué papeles organizacionales participa:

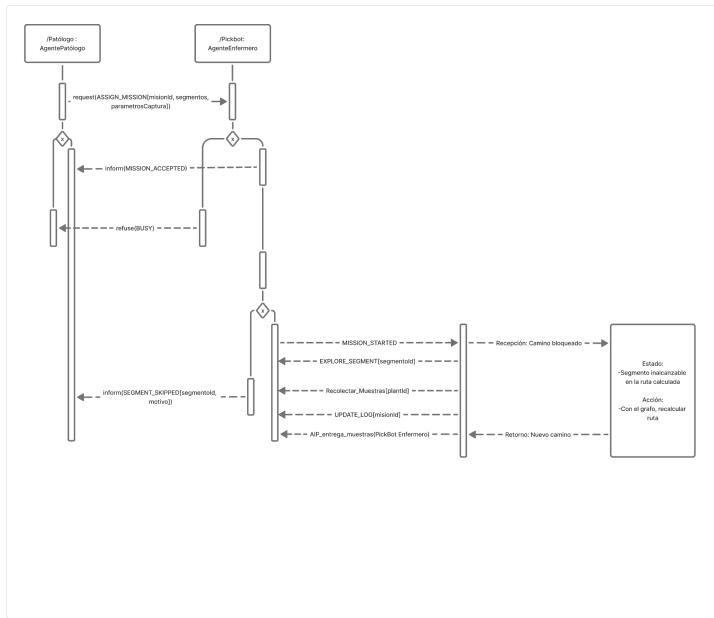
- /PickBotEnfermero → tipo de agente (enfermero).
- /RecolectorSospechosos → rol específico de recolección de frutos sospechosos detectados en la etapa 1 por los ScoutBots + Patólogo.

#### 5. Bloque <<protocols>>

##### AIP

Aquí se enumeran los AIPs en los que participa:

- AIP\_AsignarMuestreo ()
  - Protocolo donde el Patólogo asigna los objetivos de muestreo al PickBot Enfermero.



El AIP\_AsignarMuestreo coordina la interacción entre el AgentePatólogo y el PickBot Enfermero (NurseBot), encargado de recolectar muestras de plantas sospechosas dentro del invernadero. El protocolo define cómo el Patólogo asigna una misión de muestreo, cómo el PickBot Enfermero evalúa si puede asumirla, cómo ejecuta la exploración y recolección de muestras, cómo maneja rutas bloqueadas mediante un subsistema de recálculo y cómo reporta segmentos omitidos o concluye la misión.

El AIP se divide conceptualmente en:

- Fase de asignación de muestreo → el Patólogo delega la misión al PickBot Enfermero.
- Fase de decisión del agente → donde el PickBot puede aceptar o rechazar la misión.
- Fase de exploración y recolección de muestras → donde el PickBot navega, recolecta muestras y actualiza su bitácora.

- Subsistema de recálculo de ruta → mecanismo para manejar segmentos inaccesibles.
- Fase de cierre → donde el PickBot reporta segmentos bloqueados y entrega las muestras recolectadas.
- Mensajes ACL → representan formalmente el intercambio de información entre agentes.

## 1. Asignación de misión (request → accept/refuse)

request(ASSIGN\_MISSION[missionId,     segmentos,     parametrosCaptura]) ←  
 /Patólogo

El AIP inicia cuando el AgentePatólogo envía una misión de muestreo al PickBot Enfermero. El mensaje contiene:

- missionId: identificador único de la misión.
- segmentos: lista de segmentos donde deben recolectarse muestras.
- parametrosCaptura: configuraciones asociadas a la recolección (cantidad de muestras, profundidad, estandarización).

El PickBot Enfermero pasa a estado de evaluación de la misión.

## 2. Decisión del PickBot (XOR)

Una decisión interna del agente determina si puede aceptar la misión.

Evaluá:

- nivel de batería,
- disponibilidad (idle o no ocupado),

- capacidad restante de muestras,
- accesibilidad inicial del primer segmento.

Solo una de las siguientes ramas ocurre:

inform(MISSION\_ACCEPTED) → /Patólogo

El PickBot acepta la misión y carga los segmentos asignados, preparando su navegación.

refuse(BUSY) → /Patólogo

El PickBot rechaza la misión porque está ocupado o sus condiciones actuales no le permiten iniciar muestreo.

El AIP termina para esta instancia.

### 3. Ejecución del muestreo y avance de misión

Una vez aceptada la misión, el PickBot Enfermero entra en fase operativa.

MISSION\_STARTED

Indica el comienzo formal del proceso de muestreo.

EXPLORE\_SEGMENT[segmentId]

El PickBot navega hacia el segmento asignado para comenzar la recolección.

Recolectar\_Muestras[plantId]

Para cada planta sospechosa dentro del segmento:

- identifica el área a muestrear,
- toma la muestra,
- la registra en su memoria interna,
- actualiza la lista local de muestras recolectadas.

Este paso se repite para todas las plantas relevantes.

`UPDATE_LOG[missionId]`

El PickBot actualiza su bitácora con:

- muestras recolectadas,
- segmentos procesados,
- timestamps,
- ubicación actual.

El avance queda disponible para referencia posterior en el AIP de entrega.

#### 4. Subsistema de recálculo de ruta (manejo de bloqueos)

El PickBot Enfermero, al igual que otros agentes, puede encontrar un segmento inaccesible. El diagrama incluye un subsistema dedicado a ello.

Detonante:

Recepción de camino bloqueado

Esto ocurre cuando el PickBot detecta:

- pasillo u obstáculo que bloquea el acceso a un segmento,
- nodo del grafo desconectado,

- camino inválido respecto a su trayectoria.

Estado del subsistema

- Segmento inalcanzable en la ruta calculada.

Acción del subsistema

- Con el grafo, recalcular ruta.

El subsistema utiliza el grafo del invernadero para construir una nueva trayectoria accesible.

Retorno:

Nuevo camino

La ruta recalculada se devuelve al PickBot, que continúa la misión con esta nueva trayectoria.

## 5. Reporte de segmentos omitidos

Si un segmento permanece inaccesible incluso después del recálculo, el PickBot lo reporta:

inform(SEGMENT\_SKIPPED[segmentoid, motivo]) → /Patólogo

Esto permite al Patólogo registrar qué áreas no pudieron muestrearse y por qué.

## 6. Fase de cierre (entrega o aborto de misión)

Tras procesar todos los segmentos alcanzables, el AIP llega al cierre. El diagrama muestra que, en lugar de un mensaje explícito de “MISSION\_COMPLETED”, la misión continúa inmediatamente al AIP de entrega de muestras.

#### AIP\_entrega\_muestras(PickBot Enfermero)

Una vez concluido el muestreo, el PickBot inicia el protocolo para entregar las muestras recolectadas.

Este AIP forma parte del flujo general y se dispara automáticamente al finalizar todos los segmentos accesibles.

En caso de fallo crítico, el PickBot enviaría una notificación de terminación anormal (no mostrada en tu diagrama, pero coherente con el diseño general del sistema).

#### 7. Lista de mensajes ACL del AIP\_AsignarMuestreo

Mensajes de entrada hacia el PickBot Enfermero

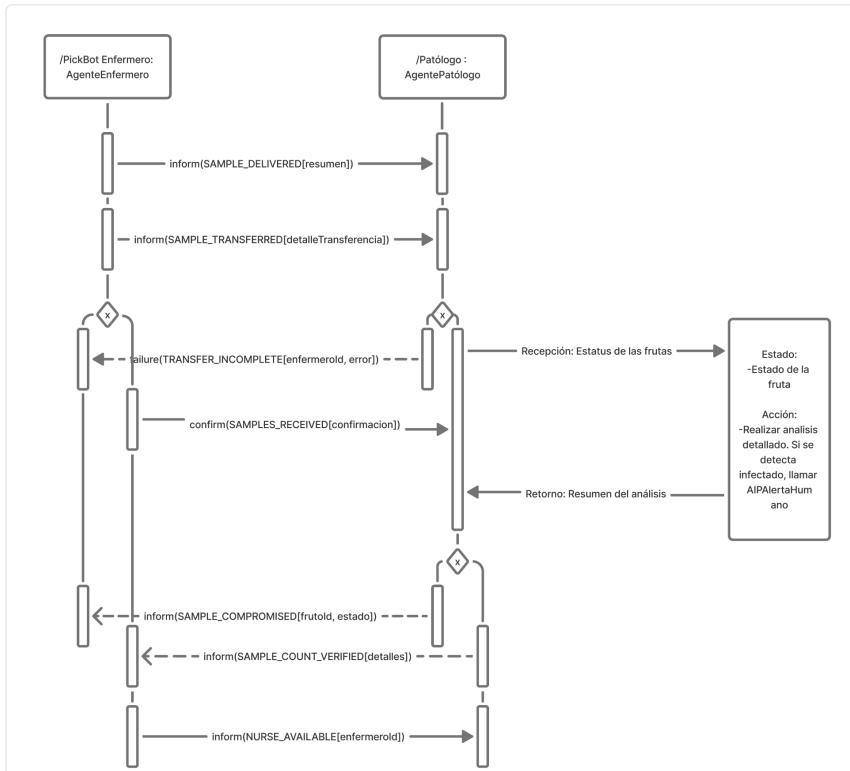
- request(ASSIGN\_MISSION[missionId, segmentos, parametrosCaptura])

Mensajes de salida del PickBot Enfermero

- inform(MISSION\_ACCEPTED)
- refuse(BUSY)
- inform(SEGMENT\_SKIPPED[segmentId, motivo])
- (y posteriormente, dentro del siguiente protocolo)

AIP\_entrega\_muestras(PickBot Enfermero)

- AIP\_EntregaMuestras ()



El AIP\_EntregaMuestras coordina la interacción entre el AgentePatólogo y el PickBot Enfermero una vez que este ha concluido la fase de recolección de muestras en segmentos sospechosos del invernadero. El propósito del protocolo es asegurar que las muestras obtenidas sean entregadas correctamente al Patólogo, registrar los metadatos de recolección y dejar al PickBot Enfermero listo para nuevas misiones posteriores.

El AIP se divide conceptualmente en:

- Fase de inicio de entrega → donde el PickBot transiciona desde su misión de recolección hacia la etapa de entrega.
- Fase de transferencia de muestras → donde el PickBot entrega los datos y evidencia recolectada al AgentePatólogo.

- Fase de confirmación → donde el Patólogo valida que la entrega está completa.
- Fase de liberación del agente → donde el PickBot Enfermero confirma que está disponible para nuevas misiones.
- Mensajes ACL → representan el flujo formal de comunicación del protocolo.

## 1. Inicio de la entrega de muestras

Al finalizar la misión de muestreo dentro del AIP\_AsignarMuestreo, el PickBot Enfermero invoca este protocolo.

En el diagrama previo, esta transición aparece como:

AIP\_entrega\_muestras(PickBotEnfermero)

Esto indica que el PickBot Enfermero ha:

- concluido la exploración,
- recolectado las muestras solicitadas,
- actualizado su bitácora interna (UPDATE\_LOG),
- decidido transitionar a la fase de entrega.

En este momento, se activa el AIP\_EntregaMuestras().

## 2. Transferencia de muestras al Patólogo

El PickBot Enfermero procede a ubicarse en un punto de entrega (base o estación de muestreo) y comienza la transmisión de los datos.

Dependiendo del modelo de comunicación del sistema, este bloque puede incluir:

- detalle de muestras recolectadas,
- segmentos inspeccionados,
- timestamps,
- IDs de plantas analizadas,
- evidencia en forma de lista o paquete de datos.

El AIP expresa este proceso como:

```
inform(DELIVER_SAMPLES_REPORT[pickBotId], muestraRecolectadas,
segmentosProcesados] → /Patólogo
```

Este mensaje asegura que el Patólogo recibe toda la información necesaria para análisis posteriores (por ejemplo, diagnóstico de enfermedades).

### 3. Confirmación del Patólogo (validación de entrega)

Una vez que el Patólogo recibe el reporte de muestras, valida:

- que la cantidad de muestras coincida con lo esperado,
- que no haya duplicados o inconsistencias,
- que se hayan registrado correctamente los segmentos visitados,
- que el PickBot no haya omitido datos importantes.

Tras esta verificación, envía la confirmación correspondiente:

```
confirm(CONFIRM_SAMPLES RECEIVED[pickBotId]) ← /Patólogo
```

Esto indica que la entrega ha sido registrada y aceptada en el sistema.

### 4. Liberación del PickBot Enfermero (fin del protocolo)

Una vez recibida la confirmación, el PickBot Enfermero actualiza su estado y libera la misión.

Finalmente informa al Patólogo que está listo para una nueva asignación:

```
inform(AVAILABLE[pickBotId]) → /Patólogo
```

Con esto el agente regresa a un estado Idle o EnBase, permaneciendo a disposición del sistema para futuras misiones de muestreo.

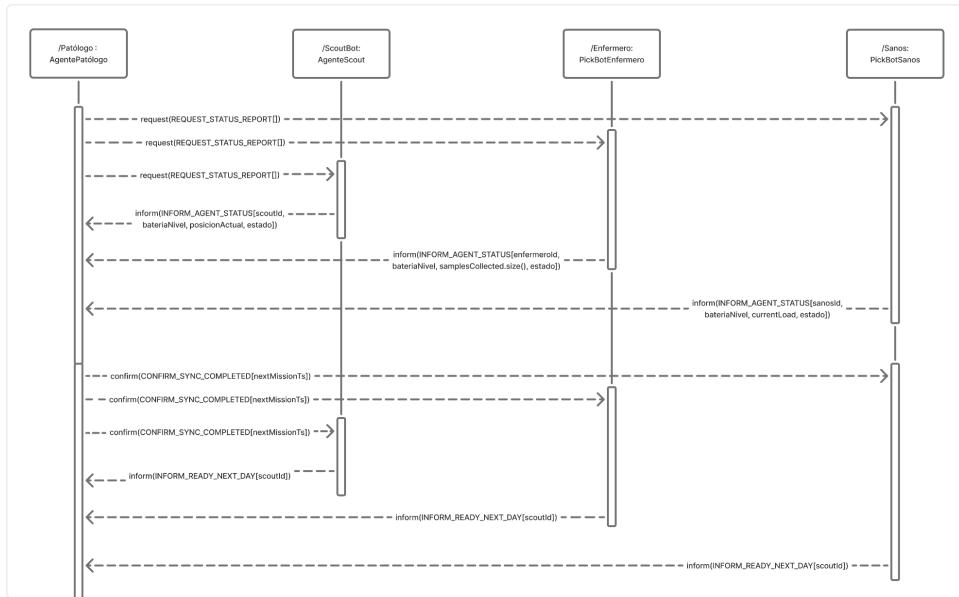
## 5. Lista de mensajes ACL del AIP\_EntregaMuestras()

Mensajes de salida del PickBot Enfermero

- `inform(DELIVER_SAMPLES_REPORT[pickBotId,       muestrasRecolectadas,  
         segmentosProcesados])`
- `inform(AVAILABLE[pickBotId])`

Mensajes de entrada hacia el PickBot Enfermero

- `confirm(CONFIRM_SAMPLES RECEIVED[pickBotId])`
  - Protocolo orientado a la entrega de muestras y confirmación de recepción.
- `AIP_SincronizarEnBase ()`
  - Protocolo donde se cargan los datos al patólogo y se actualiza el estatus de la disponibilidad del agente enfermero.



El AIP\_SincronizarEnBase coordina la sincronización nocturna entre el AgentePatólogo y todos los agentes operativos del invernadero —ScoutBots, PickBots Enfermeros y PickBots Sanos. Su objetivo es recopilar sus estados finales de jornada, validar que no existan tareas pendientes, confirmar la integridad de sus datos internos y asegurar que cada agente esté listo para operar en la siguiente misión del día siguiente.

El protocolo incluye:

- Fase de solicitud de estado → el Patólogo convoca a los agentes a reportar su estatus.
  - Fase de reporte de agentes → cada robot informa sus condiciones finales.
  - Fase de confirmación de sincronización → el Patólogo valida que todos hayan completado su reporte.
  - Fase de preparación para el siguiente día → los agentes informan que están listos para iniciar nuevas misiones.

- Mensajes ACL → expresan formalmente el flujo de interacción del protocolo.

## 1. Solicitud de reporte de estado (request → respuesta)

El AIP inicia cuando el Patólogo solicita explícitamente el estatus de cada agente activo:

```
request(REQUEST_STATUS_REPORT[]) ← /Patólogo
```

Este mensaje se envía de manera independiente a:

- ScoutBot(s),
- PickBotEnfermero(s),
- PickBotSanos(s).

El objetivo es iniciar un proceso de auditoría interna al finalizar la jornada operativa.

Los agentes deben responder con su estado actual de forma inmediata.

## 2. Reporte de estado de los agentes (inform → estatus individual)

Cada agente reporta su estado completo al Patólogo.

Las respuestas incluyen información crucial para evaluar si están listos para el día siguiente.

```
inform(INFORM_AGENT_STATUS[scoutId, bateriaNivel, posicionActual, estado]) →  
/Patólogo
```

El ScoutBot reporta:

- nivel de batería,

- posición 3D en el invernadero,
- estado (idle, en\_base, transfiriendo\_datos, etc.).

inform(INFORM\_AGENT\_STATUS[enfermeroid, bateriaNivel, samplesCollected.size(), estado]) → /Patólogo

El PickBot Enfermero reporta:

- su batería,
- número de muestras recolectadas,
- estado actual (idle, en\_base, regresando, etc.).

inform(INFORM\_AGENT\_STATUS[sanosId, bateriaNivel, currentLoad, estado]) → /Patólogo

El PickBot Sano reporta:

- batería,
- carga actual de frutos,
- estado operativo.

Estos reportes permiten al Patólogo validar que:

- no existan muestras pendientes,
- no existan frutos sin descargar,
- no existan segmentos sin finalizar,
- todos los robots regresaron a base o se encuentran en un estado seguro.

### 3. Confirmación de sincronización (confirm → validación del Patólogo)

Una vez recibidos todos los reportes de todos los agentes, el Patólogo valida la consistencia de la información:

- niveles de batería aceptables,
- posición correcta en el invernadero,
- cargas descargadas,
- cero tareas en ejecución.

Cuando la validación se completa, envía:

```
confirm(CONFIRM_SYNC_COMPLETED[nextMissionTs]) ← /Patólogo
```

Este mensaje se envía a cada agente, indicando que:

- el proceso de sincronización ha finalizado correctamente,
- no se detectaron errores,
- se establece el timestamp para la siguiente misión del día siguiente.

El parámetro nextMissionTs representa oficialmente el momento en que comenzará la siguiente ronda de misiones del sistema multiagente.

#### 4. Agentes confirman disponibilidad para el siguiente día

Tras recibir la confirmación, cada agente actualiza su estado interno y se declara listo para operar al día siguiente.

```
inform(INFORM_READY_NEXT_DAY[scoutId]) → /Patólogo
```

Un ScoutBot informa que:

- completó la sincronización,

- reseteoó contadores necesarios,
- está en base,
- está disponible para nuevas tareas del día siguiente.

inform(INFORM\_READY\_NEXT\_DAY[enfermeroid]) → /Patólogo

El PickBot Enfermero notifica disponibilidad completa tras:

- validar sus muestras,
- cerrar bitácoras,
- limpiar buffers internos.

inform(INFORM\_READY\_NEXT\_DAY[sanosId]) → /Patólogo

El PickBot de Sanos finaliza su sincronización:

- carga descargada,
- cola de cosecha vacía,
- estado “idle”.

Con estos mensajes se cierra formalmente el protocolo.

## 5. Lista de mensajes ACL del AIP\_SincronizarEnBase()

Mensajes de entrada hacia los agentes

- request(REQUEST\_STATUS\_REPORT[])
- confirm(CONFIRM\_SYNC\_COMPLETED[nextMissionTs])

Mensajes de salida desde los agentes

- inform(INFORM\_AGENT\_STATUS[scoutId, bateriaNivel, posicionActual, estado])
- inform(INFORM\_AGENT\_STATUS[enfermeroid, bateriaNivel, samplesCollected.size(), estado])
- inform(INFORM\_AGENT\_STATUS[sanosId, bateriaNivel, currentLoad, estado])
- inform(INFORM\_READY\_NEXT\_DAY[scoutId])
- inform(INFORM\_READY\_NEXT\_DAY[enfermeroid])
- inform(INFORM\_READY\_NEXT\_DAY[sanosId])

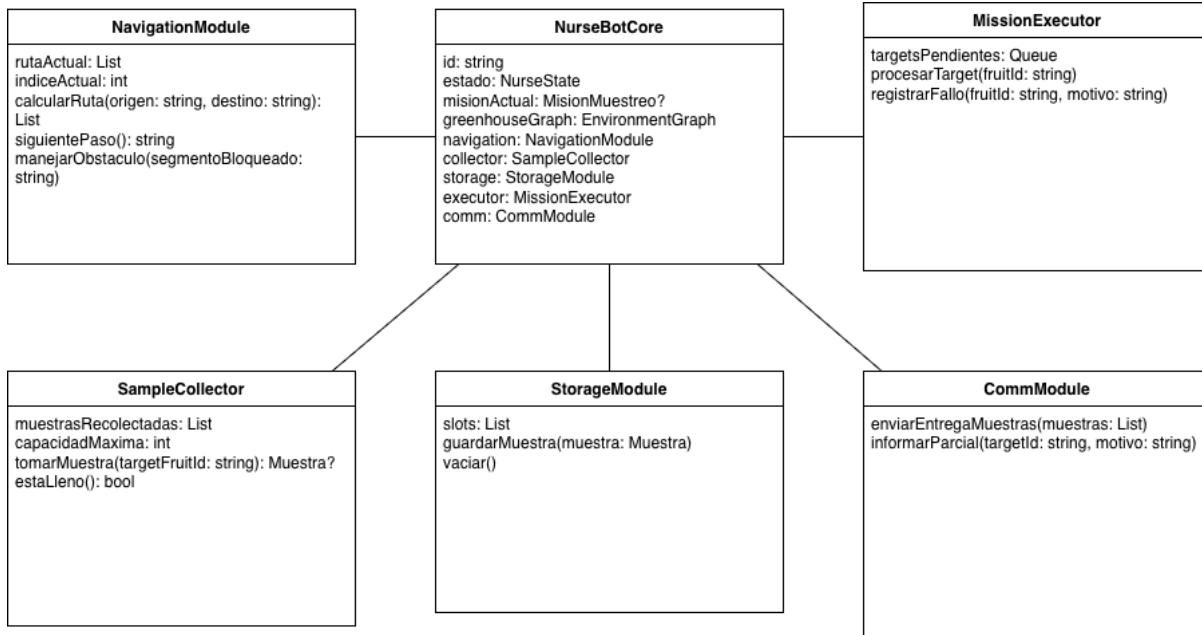
## 6. Bloque <>messages>> (comunicación ACL)

Muestran los mensajes típicos que intercambia con el Patólogo:

- Mensaje de entrada:
  - request(ASSIGN\_SAMPLING[...]) ← /Patologo  
El Patólogo envía la misión de muestreo.
- Mensajes de salida:
  - inform(MISSION\_ACCEPTED[...]) → confirma aceptación de la misión.
  - inform(SAMPLE\_COLLECTED[...]) → reporta cada muestra tomada.
  - inform(SAMPLE\_FAILED[...]) → reporta objetivos no atendidos y su motivo.
  - inform(SAMPLE\_DELIVERED[...]) → indica que las muestras llegaron a la base.
  - inform(NURSE\_AVAILABLE[...]) → indica que el bot está de nuevo disponible.

- failure(MISSION\_ABORTED[...]) → reporta abortos de misión por error.

## Subsistemas



Aquí se muestra el PickBot especializado en muestreo de frutos sospechosos, con énfasis en recolección y manejo de muestras.

- NurseBotCore

Núcleo del agente enfermero.

- Mantiene el estado, la misionActual (MisionMuestreo) y el EnvironmentGraph.
- Coordina navegación, recolección, almacenamiento, ejecución de misión y comunicación.

- NavigationModule

Igual patrón que en el Scout:

- Maneja rutas hacia los frutos/plantas objetivo de la misión.
- Responde a obstáculos recalculando rutas.

- SampleCollector

Módulo específico de toma de muestras.

- Administra la lista de muestras Recolectadas y la capacidadMaxima.
- Implementa la lógica para tomarMuestra(targetFruitId) y para saber si ya está lleno (estaLleno()).

- StorageModule

Gestiona el almacenamiento interno de muestras.

- Guarda cada Muestra en slots.
- Permite vaciar el contenedor cuando se entregan al Patólogo.

- MissionExecutor

Controla la visita a cada fruto objetivo de la misión.

- Maneja la cola de targetsPendientes (los fruitId).
- Invoca SampleCollector para recolectar y registra fallos si un target no puede ser atendido (por ejemplo, planta inaccesible).

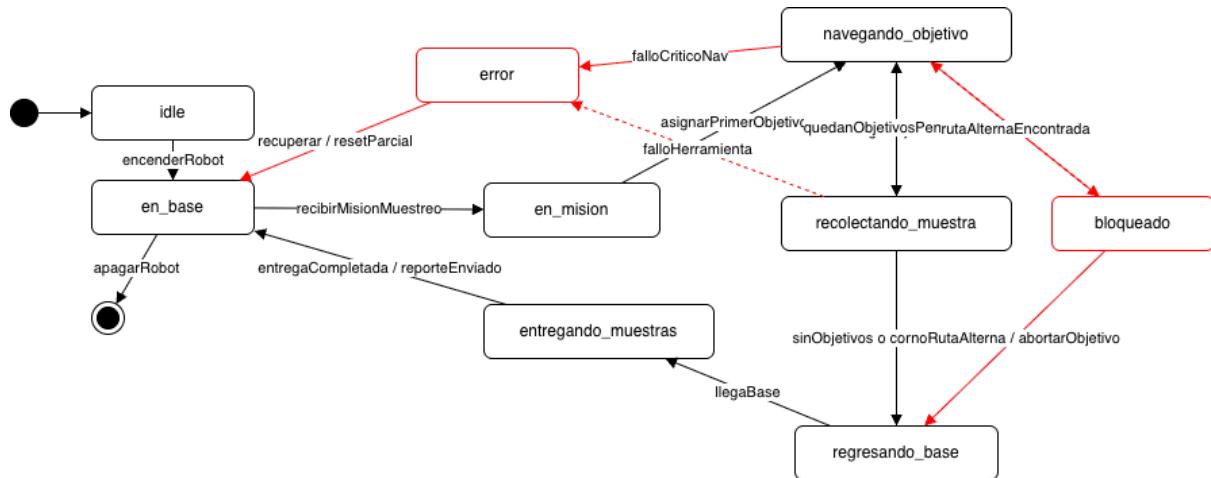
- CommModule

Gestiona la comunicación sobre el muestreo.

- Envía la entregaMuestras al Patólogo con la lista de muestras recolectadas.
- Informa casos parciales o imposibles (informarParcial con motivo).

Este diagrama muestra que el NurseBot es un agente reactivo con fuerte enfoque en gestión de inventario de muestras, reutilizando navegación y comunicación similares a las del Scout pero con lógica distinta de tarea.

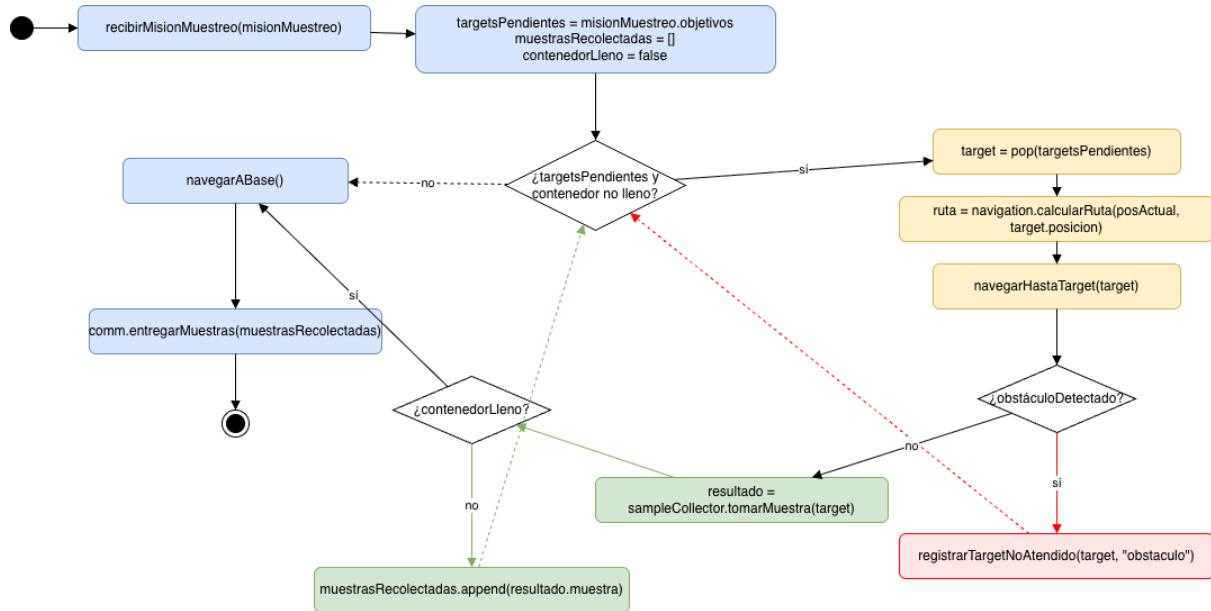
## Estados



Muestra el ciclo de vida de un PickBot Enfermero durante misiones de muestreo:

- Desde que está en idle hasta que se enciende (en\_base).
- Recibe misión (en\_mision).
- Alterna entre desplazarse a cada objetivo (navegando\_objetivo) y ejecutar muestreo (recolectando\_muestra).
- Maneja obstáculos (bloqueado) con intento de ruta alterna o abortar objetivo.
- Al terminar objetivos o llenarse el contenedor, pasa a regresando\_base y luego entregando\_muestras.
- Tras entregar, vuelve a en\_base esperando otra misión o se apaga (estado final).
- También define transiciones a error cuando hay fallos críticos en navegación o herramientas.

## Actividad



Flujo típico del subsistema MissionExecutor en NurseBot:

### 1. Recibir misión de muestreo

`recibirMisionMuestreo(mision)`

La misión incluye una lista de objetivos (frutos/plants) y quizá prioridades.

### 2. Inicializar estructuras internas

`targetsPendientes = mision.objetivos`

`muestrasRecolectadas = []`

`contenedorLleno = false`

### 3. Decisión: ¿hay targets pendientes y contenedor no lleno?

- Si sí:

- Seleccionar siguiente target.
- Calcular ruta.
- Navegar al target.
- Ver si hay obstáculo.

- Si hay obstáculo sin ruta alterna: marcar target como noAtendido y continuar con el siguiente.
- Si no hay obstáculo: intentar muestreo.
- Si no:
  - Pasar a regresar a base y entregar muestras.

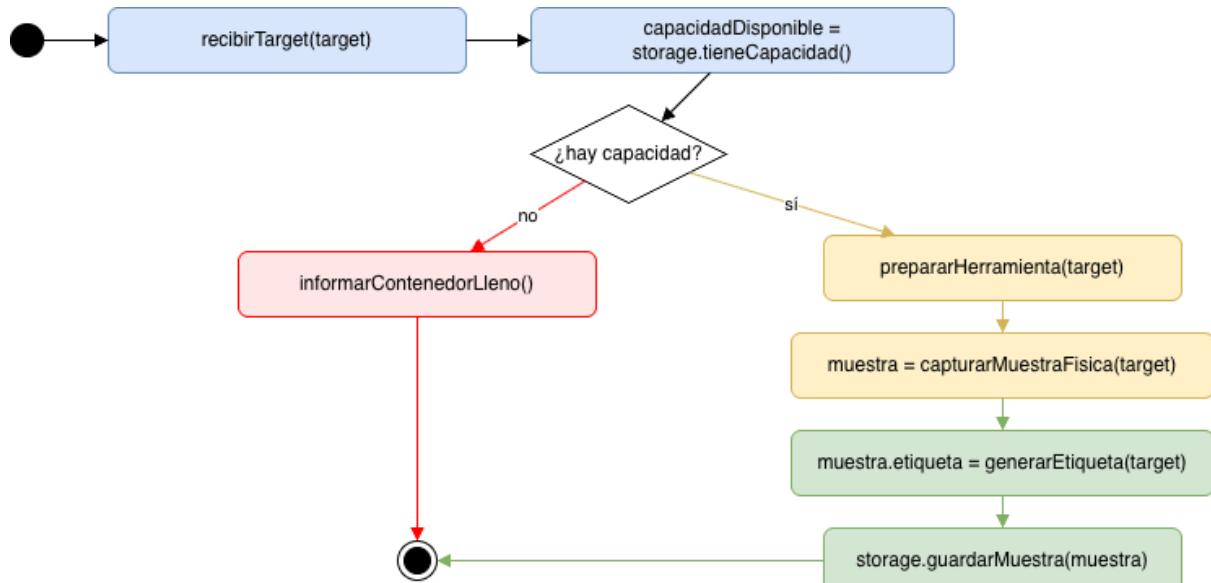
#### 4. Recolectar muestra

Llamar a SampleCollector.tomarMuestra(target):

- Si no hay capacidad: marcar contenedorLleno = true y terminar loop.
- Si se toma con éxito: guardar en Storage y agregarlas a muestrasRecolectadas.

#### 5. Regresar a base y entregar

- Navegar hasta base.
- Entregar muestrasRecolectadas al Patólogo mediante el CommModule.



Esta actividad se centra en el subsistema SampleCollector + Storage para un único objetivo:

1. Recibir target

Llega el target con fruitId / plantId.

2. Comprobar capacidad del contenedor

Si está lleno:

- No se toma muestra.
- Se avisa al MissionExecutor que el contenedor está lleno.

3. Preparar herramienta de muestreo

Actividad: prepararHerramienta(target) (acercar brazo, ajustar ángulo, etc.).

4. Tomar muestra física

Actividad: muestra = capturarMuestraFisica(target).

5. Etiquetar y registrar

- muestra.etiqueta = generarEtiqueta(target)
- storage.guardarMuestra(muestra).

6. Actualizar contador / estado

Se incrementa número de muestras, se devuelve éxito al MissionExecutor.

## Descripción del entorno

El mundo virtual se visualiza como el interior de un gran invernadero de tomates, visto en 3D desde una cámara en tercera persona y cámaras “de sistema” ubicadas en puntos clave. La escena está organizada en una cuadrícula de pasillos rectos y perpendiculares; entre cada par de pasillos hay una cama de cultivo de tierra que se divide lógicamente en dos secciones: columna A (accesible desde el

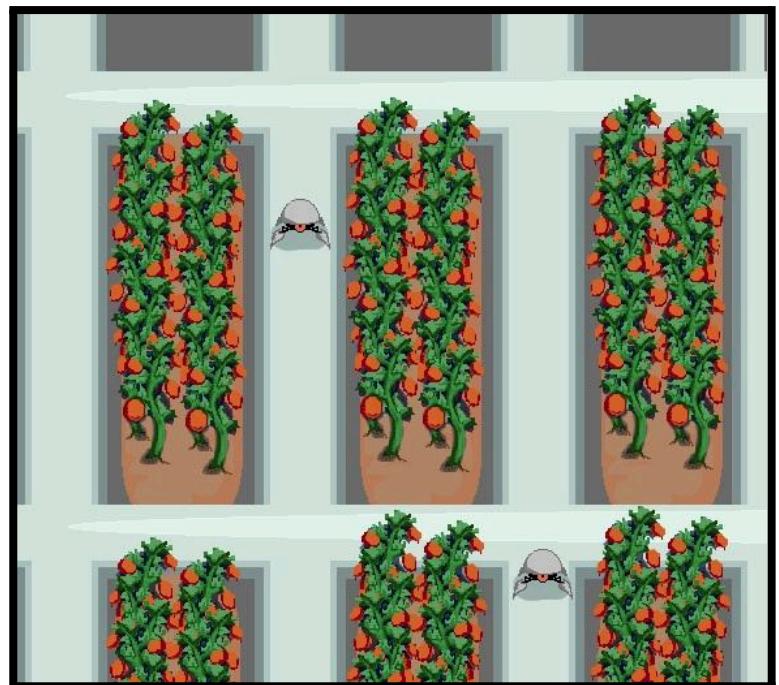
pasillo izquierdo) y columna B (accesible desde el pasillo derecho). El techo es alto y translúcido, con estructuras metálicas y paneles tipo invernadero que dejan pasar una luz suave; el piso es limpio, de materiales claros y marcas que guían el movimiento de los robots.

Los elementos virtuales clave son: las camas de cultivo con tierra y plantas de tomate, los frutos en diferentes estados (sanos, sospechosos, críticos) visibles a simple vista, los pasillos que conectan los segmentos del grafo, la zona de entrada del invernadero y, junto a ella, la Base central. En la Base se concentra el “cerebro” del sistema (el Patólogo Digital) representado como una estación fija de control: racks de servidores, pantallas y paneles de monitoreo donde se visualiza el mapa del invernadero y el estado de cada segmento. En esa misma zona se ubican las docking stations de los robots, donde los ScoutBots y PickBots aparecen acoplados al inicio y al final de cada misión.

Los modelos de agentes se inspiran en robots compactos y limpios:

- **ScoutBots**: pequeños vehículos móviles que recorren los pasillos; con un sistema de piernas y un “ojo” frontal tipo para las cámaras de inspección.
- **PickBot Enfermero**: variante más robusta, con un mecanismo frontal o modo de movimiento que simula la toma de frutos sospechosos y su depósito en un contenedor protegido.
- **PickBot de Sanos**: similar al PickBot Enfermero, pero con contenedor diferenciado para frutos sanos, para reforzar visualmente la separación de flujos.

Además de los agentes, se modelan las plantas con suficiente detalle para distinguir hojas y racimos de tomates, las cajas o contenedores en la Base donde se depositan muestras y cosecha, y elementos secundarios (tuberías, estructuras metálicas, luminarias) que refuerzan la sensación de un invernadero automatizado y futurista, manteniendo siempre una lectura clara de pasillos, filas A/B y puntos de interés para el comportamiento de los agentes.



## Plan de trabajo

### 1. Organización general del plan

Roles:

- **Integrante 1** – Clay Gutiérrez: Coordinación general, diseño de arquitectura multiagente, documentación.
- **Integrante 2** - Emilio Hernandez: Líder técnico Unity (escena 3D, navegación básica).
- **Integrante 3** - Alfredo Carmona: Lógica de agentes en C# (ScoutBots).
- **Integrante 4** - Alejandro Gutiérrez: Lógica de agentes en C# (PickBots).

- **Integrante 5** - Dylan Pereyra: Soporte en Patólogo Digital (BDI, flujos, documentación técnica).
- **Integrante 6** - Luis Díaz: Pruebas, métricas, pulido visual y consistencia gráfica.

## 2. Plan de trabajo por semanas

### Semana 1 – Definición y base colaborativa

#### Objetivos principales:

- Formalizar la comprensión del reto y del contexto (ToBRFV, necesidad de detección temprana).
- Definir agentes, roles y arquitecturas (Patólogo híbrido, ScoutBots, PickBots).
- Establecer herramientas colaborativas.

#### Actividades:

1. Definición final de la propuesta conceptual:
  - Reto, agentes, relaciones, arquitecturas.
2. Redacción del documento base:
  - Secciones: descripción del reto, agentes, arquitecturas, flujo general.
3. Creación de herramientas colaborativas:
  - Repositorio en GitHub para documentación y código.
  - Canal oficial de comunicación (Discord).
4. Definir roles internos y responsables por componente (Unity, agentes, docs).

#### Responsables (sugerido):

- Integrante 1: coordinación del documento y arquitectura multiagente.
- Integrante 2: configuración inicial de proyecto Unity.

- Integrante 6: creación y organización del repositorio + lineamientos de commits.
- Todo el equipo: acordar canal de comunicación y reglas de trabajo.

**Intervalo de esfuerzo estimado (por persona)**

4–6 horas en la semana.

**Semana 2 – Modelo del invernadero y agentes básicos (Primera revisión)**

**Objetivos principales:**

- Contar con la base visual y lógica mínima del sistema.
- Tener lista la propuesta formal documentada para la primera revisión.

**Actividades planeadas para la primera revisión:**

1. Modelo del invernadero como grafo de segmentos:
  - Definir nodos (segmentos + Base) y aristas (pasillos).
  - Representación conceptual en el documento + boceto en Unity.
2. Prototipo en Unity:
  - Escena con invernadero simplificado.
  - Un ScoutBot con movimiento básico siguiendo segmentos predefinidos.
3. Definición formal de agentes:
  - Patólogo Digital (híbrido).
  - ScoutBots, PickBot Enfermero, PickBot de sanos (reactivos).
  - Componentes arquitectónicos documentados (como ya se definió).
4. Actualización del plan de trabajo en el documento.

**Responsables y esfuerzo estimado:**

- Modelo de grafo + documentación:
  - Integrante 1 + Integrante 5 – 3–4 horas.
- Escena base Unity + movimiento simple ScoutBot:
  - Integrante 2 + Integrante 3 – 5–7 horas.
- Revisión y pulido de documento para primera revisión:
  - Integrante 1 + Integrante 6 – 3–4 horas.

**Resultado esperado al final de Semana 2 (primera revisión):**

- Documento formal coherente y completo en:
  - Descripción del reto.
  - Agentes y arquitecturas.
- Prototipo inicial:
  - Invernadero básico.
  - ScoutBot recorriendo segmentos asignados.

**Semana 3 – Integración del Patólogo y flujo ScoutBot → Patólogo****Objetivos principales:**

- Implementar el ciclo de misión del ScoutBot.
- Integrar el Patólogo como coordinador lógico (sin toda la complejidad aún).

**Actividades:**

1. Implementar lógica de misión del ScoutBot:
  - Recibir lista de segmentos.
  - Recorrer, simular captura de imágenes, registrar bitácora.
  - Regresar a Base.

2. Implementar módulo básico del Patólogo:
  - Asignar misiones a ScoutBots.
  - Recibir datos al regreso (simulados).
  - Actualizar estado de segmentos (inspeccionado / no inspeccionado).
3. Documentar el flujo completo:
  - Desde planificación inicial hasta descarga de datos.

**Responsables:**

- ScoutBot misión completa:
  - Integrante 3 – 5–7 horas.
- Patólogo (coordinación básica + recepción de datos):
  - Integrante 1 + Integrante 5 – 5–7 horas.
- Pruebas y documentación:
  - Integrante 6 – 3–4 horas.

**Actividades pendientes al cierre de Semana 3 (esperado):**

- Implementar screening masivo (clasificación SANO/SOSPECHOSO/CRÍTICO simulada).
- Implementar PickBot Enfermero y PickBot de sanos.

**Semana 4 – Screening, PickBots y pipeline completo**

**Objetivos principales:**

- Completar el flujo técnico de detección en dos etapas.
- Integrar PickBot Enfermero y PickBot de sanos en la simulación.

**Actividades:**

1. Análisis Etapa 1 en el Patólogo:

- Simular procesamiento de “imágenes”:
  - Asignar estados SANO/SOSPECHOSO/CRÍTICO.
- Generar lista de objetivos sospechosos.

2. Implementar PickBot Enfermero:

- Recibir lista de objetivos.
- Recorrer segmentos, “recolectar” frutos sospechosos.
- Regresar a Base y notificar.

3. Implementar PickBot de sanos (opcional pero recomendado):

- Recibir zonas seguras.
- Simular cosecha en esas zonas.

4. Integrar Etapa 2:

- Patólogo recibe muestras del PickBot Enfermero.
- Simula análisis detallado.
- Marca plantas como enfermas confirmadas o descartadas.
- Genera alertas.

**Responsables:**

- Análisis Etapa 1 + Etapa 2 en Patólogo:
  - Integrante 1 + Integrante 5 – 5–7 horas.
- Implementación PickBot Enfermero:
  - Integrante 4 – 5–7 horas.
- Implementación PickBot de sanos:
  - Integrante 2 + Integrante 4 – 4–6 horas.

- Integración final y pruebas:
  - Integrante 6 + todo el equipo – 4–6 horas.

**Actividades pendientes al cierre de Semana 4 (esperado):**

- Refinar comportamiento.
- Mejorar visualización, métricas y presentación final.

**Semana 5 – Refinamiento, validación y presentación final**

**Objetivos principales:**

- Pulir simulación, documentación y coherencia gráfica.
- Preparar la presentación final del reto.

**Actividades:**

1. Refinamiento técnico:

- Ajustar comportamientos de agentes.
- Revisar consistencia entre arquitectura teórica y comportamiento en Unity.

2. Validación del flujo:

- Verificar:
  - Misiones de ScoutBots.
  - Screening del Patólogo.
  - Acciones del PickBot Enfermero y PickBot de sanos.

3. Documentación final:

- Actualizar documento con:
  - Plan de trabajo final.
  - Aprendizaje adquirido.

- Capturas/diagramas del sistema.

#### 4. Preparación de presentación:

- Diapositivas.
- Guión de explicación del flujo y decisiones de diseño.

#### **Responsables:**

- Refinamiento técnico:
  - Integrante 2, 3, 4 – 6–8 horas.
- Documentación final:
  - Integrante 1 + Integrante 5 + Integrante 6 – 5–7 horas.
- Presentación:
  - Todo el equipo.

## Aprendizaje adquirido

Al corte de esta propuesta formal, después de completar el conjunto de diagramas (dominio, agentes, protocolos de interacción y comportamiento), el equipo identifica los siguientes aprendizajes:

### **1. Coherencia global en un sistema multiagente**

Al construir todos los diagramas, el equipo entendió mejor la importancia de mantener **un vocabulario de dominio unificado** (EnvironmentGraph, Segmento, Planta, Fruto, Misiones, Observaciones, Muestras, Reportes) que se refleja de forma consistente en:

- Diagramas de clases de dominio.
- Diagramas de clases de agentes.
- Diagramas de interacción (AIPs).

Esto permite trazar claramente cómo cada agente “ve” el invernadero y cómo sus acciones modifican ese estado común.

## 2. Valor de separar dominio, agentes y protocolos

El ejercicio de generar **diagramas globales de dominio**, **diagramas específicos de agente** y **AIPs** ayudó a distinguir:

- Qué pertenece al **mundo físico/simulado** (invernadero y su estado sanitario).
- Qué pertenece a la **arquitectura interna de los agentes** (estado, acciones, roles, protocolos).
- Qué pertenece a la **capa de comunicación** (mensajes ACL y secuencias de interacción).

Esto clarificó las responsabilidades de cada elemento y redujo ambigüedades en el diseño.

## 3. Profundización en el diseño de arquitecturas de agentes

Al detallar estados, acciones, roles, protocolos y comportamientos:

- Se consolidó la diferencia entre **agentes reactivos** (ScoutBots y PickBots) y **el agente híbrido BDI** (Patólogo Digital).
- Se hizo explícita la separación entre **percepción, decisión y ejecución** dentro de los subsistemas del Patólogo (BeliefBase, MissionPlanner, ScreeningEngine, etc.).
- Se entendió mejor qué componentes pueden reutilizarse entre agentes (p.ej., módulos de navegación y comunicación).

## 4. Comprensión práctica de los AIPs (protocolos de interacción)

Al modelar los AIPs (AsignarMisiónScout, ReporteMisiónScout, RecolecciónSospechosos, CosechaSegura, AlertaHumano), el equipo:

- Visualizó cómo se encadenan los mensajes ACL entre roles y cómo se documentan los casos de éxito y error.
- Identificó los puntos críticos donde se requiere confirmación, manejo de fallas y sincronización de estado (por ejemplo, recepción de lotes de imágenes y entrega de muestras).

- Vio la utilidad de los AIPs como “contratos” de comunicación que luego se pueden implementar en la simulación.

## 5. Trazabilidad entre requisitos, agentes y comportamientos

El proceso de diagramar permitió mapear con mayor claridad:

- Requisitos del problema (detección temprana, priorización de zonas, manejo de sospechosos) →
- Misiones definidas (Exploración, Muestreo, Cosecha) →
- Comportamientos concretos en diagramas de actividad/estado para cada agente.

Esto facilitó ver qué partes del problema ya están cubiertas en el diseño y qué aspectos podrían incorporarse en iteraciones futuras.

## 6. Disciplina de diseño incremental y modular

1. Al ir construyendo diagramas por capas (dominio → agentes → protocolos → comportamiento), el equipo reforzó la importancia de:
  - Diseñar primero un **modelo conceptual claro** antes de entrar a la implementación.
  - Mantener los diagramas lo suficientemente **modulares** para poder actualizar un agente o un protocolo sin tener que rehacer todo el modelo.
  - Pensar en la **evolución futura** del sistema (por ejemplo, agregar nuevos tipos de agentes o nuevas políticas del Patólogo) sin romper la estructura actual.

Este bloque de trabajo sobre modelado y diagramas se utilizará como base; en las siguientes entregas el equipo actualizará el plan y el apartado de aprendizaje adquirido conforme avance la implementación en Unity, la integración de los agentes y las pruebas del sistema en el entorno simulado.