# Contagio City (Modelo epidemiológico)



Alejandro González Flórez – 1701913677

Iván Junior Echeverri Gómez - 1701911851

**Sistemas Inteligentes I**

Luis Fernando Castillo Ossa

Manizales-Caldas

Contenido

[Contagio City (Modelo epidemiológico) 1](#_Toc200493550)

[1. Objetivo General: 3](#_Toc200493551)

[2. Componentes del Proyecto 4](#_Toc200493552)

[2.1. Descripción del Problema y Contexto 4](#_Toc200493553)

[2.2. Modelado del Sistema Multiagente 5](#_Toc200493554)

[2.3. Metodología de Modelado: 7](#_Toc200493555)

[2.4. Automatización del Sistema: 10](#_Toc200493556)

[2.5. Funcionamiento del sistema: 12](#_Toc200493557)

[2.6. Estimación de Escalabilidad: 15](#_Toc200493558)

## Objetivo General:

El objetivo principal del proyecto "Contagio City" es desarrollar una simulación computacional interactiva que represente el comportamiento de una enfermedad contagiosa al propagarse dentro de una población urbana. A través de un modelo basado en agentes, se busca visualizar de manera clara cómo distintos factores como la tasa de infección, el nivel de inmunidad, el aislamiento social y la vacunación influyen en la dinámica del contagio.

A causa de la pandemia de COVID del 2020, se evidenció de manera global la importancia de comprender cómo se propagan las enfermedades infecciosas en entornos urbanos y altamente conectados. La rápida transmisión del virus, sumada a la incertidumbre sobre sus efectos y la efectividad de las medidas de contención, generó una necesidad urgente de herramientas que ayudaran a visualizar y analizar diferentes escenarios de contagio.

A base de esto, las simulaciones por computadora se convirtieron en un recurso clave para investigadores, gobiernos y ciudadanos, ya que permitieron evaluar el impacto de decisiones como cuarentenas, restricciones de movilidad, campañas de vacunación y distanciamiento social.

Mediante Contagio City se pretende proporcionar una herramienta educativa que facilite la comprensión de conceptos epidemiológicos básicos como los modelos SIR (Susceptible-Infectado-Recuperado), así como fomentar el análisis de escenarios hipotéticos para la toma de decisiones en contextos de crisis sanitaria. El desarrollo del proyecto se realizará utilizando Python y la librería Mesa, lo que permitirá una implementación accesible, visual y fácilmente personalizable.

## Componentes del Proyecto

### Descripción del Problema y Contexto

La rápida propagación de enfermedades contagiosas en espacios urbanos representa un desafío significativo para los sistemas de salud pública y la gestión gubernamental. Cuando una enfermedad infecciosa, como ocurrió con el COVID-19 en 2020, irrumpe en una comunidad densamente poblada, se vuelve fundamental anticipar su comportamiento, evaluar el impacto de diferentes medidas sanitarias y promover una toma de decisiones informada. Sin embargo, este proceso resulta complejo debido a la gran cantidad de variables involucradas, como la movilidad de las personas, el nivel de inmunidad, el comportamiento individual, el acceso a servicios médicos, y las políticas de intervención (como confinamientos o vacunaciones).

Para abordar este problema, se optó por utilizar un enfoque multiagente, ya que ofrece una manera natural y poderosa de modelar sistemas complejos donde múltiples entidades autónomas interactúan entre sí. En una epidemia real, cada individuo actúa de forma diferente: algunos se aíslan voluntariamente, otros ignoran recomendaciones, algunos están vacunados o son inmunes, mientras que otros son altamente vulnerables o supercontagiadores. Estos comportamientos no pueden representarse de manera precisa mediante modelos matemáticos tradicionales (como las ecuaciones diferenciales simples del modelo SIR), ya que estos suelen asumir poblaciones homogéneas.

En el proyecto **"Contagio City"**, los agentes representan personas individuales que se mueven e interactúan dentro de un entorno urbano simplificado. Este entorno está compuesto por una cuadrícula (grid), que simula una ciudad con calles, espacios públicos y posibles áreas de confinamiento o atención médica. Cada celda del entorno puede representar una posición geográfica donde los agentes se desplazan, se agrupan o se aíslan.

### Modelado del Sistema Multiagente

En "Contagio City", los agentes actúan dentro de un entorno simulado que representa una ciudad simplificada organizada en forma de **una cuadrícula bidimensional (grid)**. Cada celda del grid representa una porción del espacio urbano, como una calle, una casa, un parque, un hospital o una zona pública de tránsito frecuente. El entorno simula un contexto de emergencia sanitaria causado por la aparición de una enfermedad contagiosa, donde se busca observar el comportamiento de la propagación bajo distintos escenarios sociales y de política pública.

Cada agente representa una persona individual con la capacidad de realizar diversas acciones durante la simulación, como:

* **Moverse** aleatoriamente o con una lógica definida dentro del entorno (si no está confinado).
* **Cambiar su estado de salud** de susceptible a infectado, de infectado a recuperado o inmunizado, dependiendo del curso de la enfermedad.
* **Evaluar riesgos** en su entorno (por ejemplo, evitar zonas con muchos infectados, si el agente tiene ese nivel de conciencia).
* **Participar en campañas de vacunación**, si están disponibles y el agente decide hacerlo.
* **Transmitir la enfermedad** si está infectado y entra en contacto con otros agentes susceptibles.
* **Respetar o ignorar** restricciones, dependiendo de su nivel de obediencia o conciencia sanitaria (atributo personalizable).

Estas acciones son ejecutadas en cada ciclo de simulación, lo que permite observar la evolución del sistema con el paso del tiempo.

Cada agente en el sistema toma decisiones autónomas, sin necesidad de intervención directa del usuario. Estas decisiones se basan en sus atributos internos y en las condiciones del entorno inmediato.

Los interactúan a través del entorno compartido. Es decir, la proximidad física en la cuadrícula funciona como el medio principal de interacción: si dos agentes coinciden en celdas cercanas, se evalúa la posibilidad de transmisión del virus. No se utiliza mensajería directa ni APIs en este modelo, ya que el objetivo es mantener una simulación sencilla y centrada en la dinámica epidemiológica. Sin embargo, los agentes sí "observan" el entorno, lo cual puede influir en sus decisiones.

### Metodología de Modelado:

Para el desarrollo de la simulación de epidemias, se ha adoptado una metodología de modelado basada en la identificación de agentes, sus roles, las tareas que realizan y los flujos de interacción entre ellos. Este enfoque práctico y modular permite una comprensión clara del sistema y una implementación eficiente, similar a los principios generales de GAIA o Prometheus, pero adaptado a la especificidad de los sistemas multi-agente (SMA).

#### Identificación de Agentes:

El agente principal en esta simulación es el Ciudadano. Cada ciudadano es una entidad individual con sus propios estados y comportamientos.

#### Roles y Estados de los Agentes:

Cada Ciudadano puede asumir uno de los siguientes roles (estados), que definen su comportamiento dentro de la simulación:

**Susceptible:** Individuo sano que puede contraer la enfermedad.

**Infectado:** Individuo que ha contraído la enfermedad y puede contagiar a otros.

**Recuperado:** Individuo que se ha recuperado de la enfermedad y es inmune por un período.

**Muerto:** Individuo que ha fallecido a causa de la enfermedad.

Además, se considera un atributo usa\_mascarilla para cada ciudadano, que influye en la probabilidad de contagio y es asignado aleatoriamente al inicio de la simulación.

#### Tareas de los Agentes:

Las tareas principales que realiza un agente Ciudadano en cada paso de tiempo (step) son:

Mover: Desplazarse a una celda vecina dentro de la cuadrícula. Los individuos Muertos no se mueven.

Infectar: Si un ciudadano es Susceptible, interactúa con sus vecinos para determinar si se contagia, teniendo en cuenta la tasa\_contagio del modelo, el uso de mascarillas y la virulencia de la variante activa.

Evolucionar el estado de infección: Si un ciudadano está Infectado, se actualiza su tiempo\_infeccion y, si se cumple la duracion\_infeccion o se alcanza una probabilidad de letalidad, cambia a Recuperado o Muerto respectivamente.

#### Flujos de Interacción:

Las interacciones clave en el modelo se dan entre los Ciudadanos a través de su posición en la cuadrícula:

Contagio: Los Ciudadanos Infectados interactúan con Ciudadanos Susceptibles en celdas adyacentes para transmitir la enfermedad. La probabilidad de contagio se ve modificada por el uso de mascarillas de los agentes involucrados y la variante del virus.

Movimiento: Los agentes se mueven de forma independiente, lo que facilita el encuentro y la interacción con otros agentes en diferentes partes de la cuadrícula.

#### Modelo del Entorno:

El EpidemiaModel actúa como el entorno y el coordinador de la simulación. Sus responsabilidades incluyen:

Inicialización: Crea la cuadrícula (MultiGrid), el planificador (RandomActivation), y los agentes Ciudadanos. Designa un "paciente cero" inicial.

Gestión de parámetros globales: Define y gestiona parámetros como tasa\_contagio, duracion\_infeccion, probabilidad\_uso\_mascarilla, y las características de las variantes del virus (contagio y letalidad).

Recolección de datos: Utiliza un DataCollector para registrar el número de Susceptibles, Infectados, Recuperados, Muertos, y el recuento de infectados por Variante X y Variante Y en cada paso.

Mutación del virus: Introduce un evento de mutación en un ciclo específico, cambiando la variante activa del virus de forma aleatoria, lo que afecta las dinámicas de contagio y letalidad.

Esta metodología basada en roles y tareas permite una clara separación de responsabilidades y facilita la extensión del modelo con nuevos comportamientos o tipos de agentes en el futuro.

### Automatización del Sistema:

La simulación de epidemias se implementa utilizando la librería Mesa en Python, que es una plataforma robusta para la creación de modelos basados en agentes. La automatización de la ejecución y visualización del sistema se logra a través de la integración de Mesa con su ModularServer.

Si bien se mencionan herramientas como n8n, Make o LangChain para la automatización general, para este proyecto específico de simulación basada en agentes con visualización en tiempo real, Mesa y su servidor incorporado son la elección más adecuada y eficiente.

#### Mesa (Python Library):

Mesa es una herramienta especializada en modelos basados en agentes (ABM) que proporciona una estructura clara para definir agentes, modelos, programadores de eventos y cuadrículas espaciales. Se ha elegido por la Facilidad para utilizar su API y su documentación, lo que facilita la implementación de modelos complejos de forma organizada.

La principal funcionalidad por la que se ha elegido Mesa es porqué Integra nativamente la recolección de datos y, crucialmente, la visualización interactiva a través de su ModularServer.

#### ModularServer (parte de Mesa):

Esta herramienta está diseñada específicamente para la visualización e interacción con modelos Mesa. Permite la configuración rápida de interfaces de usuario con sliders para parámetros modificables y gráficos para la visualización de datos en tiempo real y se integra perfectamente con el modelo de simulación, proporcionando una experiencia interactiva sin necesidad de herramientas de automatización externas para la ejecución básica.

#### Uso de LLM (Large Language Models):

En el contexto actual de esta simulación de epidemias, no se utilizan LLM para el funcionamiento intrínseco del modelo. El comportamiento de los agentes y la dinámica de la epidemia están definidos por reglas determinísticas y probabilísticas basadas en parámetros predefinidos.

Sin embargo, un LLM podría ser utilizado en fases posteriores para:

1. **Análisis de Resultados:**

Generar resúmenes o interpretaciones textuales de los datos recolectados por el DataCollector (por ejemplo, tendencias en el número de infectados, impacto de las mascarillas).

1. **Generación de Escenarios:**

Ayudar a idear nuevos escenarios de simulación o a ajustar parámetros de forma más inteligente.

1. **Documentación:**

Generar documentación automática del modelo o explicaciones del código.

Si se integrara un LLM para las tareas mencionadas, se podría considerar Gemini 2.0 Flash por su equilibrio entre capacidad y eficiencia, o un modelo de código abierto como Llama 3 (cuando esté disponible para uso práctico y licencias adecuadas) por su flexibilidad y transparencia si el despliegue es local. La elección dependería de si se requiere una API en la nube o un control total del modelo.

#### Diseño del Flujo de Trabajo Automatizado:

El flujo de trabajo automatizado con Mesa se organiza de la siguiente manera:

1. **Definición del Agente (Ciudadano):**

Se define la clase Ciudadano con sus atributos (estado, tiempo de infección, variante, uso de mascarilla) y métodos (mover, infectar, step). El método step encapsula el comportamiento del agente en cada iteración.

1. **Definición del Modelo (EpidemiaModel):**

Se inicializan los componentes del modelo: cuadrícula, planificador, agentes.

Se establecen los parámetros del modelo (tasa de contagio, duración de infección, probabilidad de mascarilla) y las propiedades de las variantes del virus.

Se configura el DataCollector para especificar qué datos se recopilarán en cada paso de la simulación. Se implementa el método step del modelo, que coordina la ejecución de los pasos de los agentes y maneja la mutación del virus.

1. **Configuración de la Visualización:**

Se define la función agent\_portrayal que asigna un color y forma a cada agente en función de su estado y si usa mascarilla.

Se configuran los módulos de visualización: CanvasGrid (para la representación de la cuadrícula) y ChartModule (para los gráficos de datos).

#### Lanzamiento del Servidor (ModularServer):

Se instancia ModularServer con el modelo (EpidemiaModel), los módulos de visualización (cuadrícula y gráfico), un título para la interfaz, y un diccionario de UserSettableParameters. Estos parámetros son los sliders que permiten al usuario modificar valores en tiempo real.

### Funcionamiento del sistema:

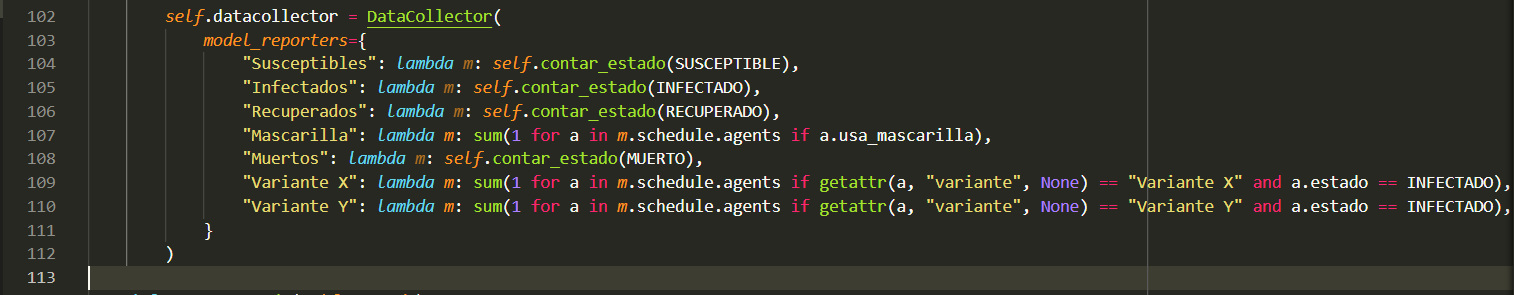
#### Inicialización del Servidor y Parámetros:

Este bloque de código muestra cómo se configura el servidor web (ModularServer) para la simulación. Define los parámetros que el usuario puede ajustar (cantidad de agentes, tasa de contagio, duración de infección, probabilidad de uso de mascarilla) a través de sliders en la interfaz web. El server.launch() inicia el servidor que ejecuta la simulación y presenta la interfaz visual.



#### Recolección de Datos (DataCollector):

Este fragmento dentro de la clase EpidemiaModel configura el DataCollector. Permite al sistema automatizar la recopilación de métricas clave en cada paso de la simulación, como el número de agentes en cada estado (susceptible, infectado, recuperado, muerto) y el conteo de infectados por variante. Estos datos son luego visualizados en el ChartModule.



* + 1. **Visualización de Agentes (agent\_portrayal y CanvasGrid):**

La función agent\_portrayal define cómo se representará visualmente cada agente en la cuadrícula. El CanvasGrid utiliza esta función para dibujar a los agentes, actualizando su apariencia en tiempo real a medida que sus estados cambian, lo que proporciona una evidencia visual directa del comportamiento del sistema.



* + 1. **Gráfico de Evolución (ChartModule):**

El ChartModule toma los datos recolectados por el DataCollector y los presenta en un gráfico de líneas en tiempo real. Esto automatiza la generación de informes visuales sobre la evolución de la epidemia, permitiendo al usuario observar las dinámicas de la población a lo largo del tiempo.

Texto

El contenido generado por IA puede ser incorrecto.

### Estimación de Escalabilidad:

El sistema de simulación de epidemias, basado en Mesa, está diseñado para modelar interacciones entre agentes en un entorno discreto. Su escalabilidad se ve influenciada principalmente por el número de agentes y la complejidad de sus interacciones, así como por la capacidad de la infraestructura subyacente.

El rendimiento del sistema dependerá de la capacidad de procesamiento (CPU) y la memoria (RAM) disponibles, ya que cada agente y sus interacciones consumen recursos. El cálculo de vecinos, la actualización de estados y el movimiento de agentes son operaciones que se escalan con el número de agentes. La visualización en tiempo real también añade una carga computacional, especialmente a medida que el número de agentes aumenta y la cuadrícula se vuelve más densa.

A continuación, se presenta una estimación de escalabilidad para diferentes escenarios, considerando el tiempo de respuesta (el tiempo que tarda un "paso" o step de la simulación en completarse, incluyendo la actualización de todos los agentes y la recolección de datos) y la infraestructura recomendada:

|  |  |  |  |
| --- | --- | --- | --- |
| Escenario | Nº de Agentes | Tiempo de Respuesta (por paso) | Infraestructura Sugerida |
| Mínimo | 3 | < 2 segundos | PC local (recursos básicos) |
| Intermedio | 10 | 2 - 4 segundos | Hosting básico (CPU 2-4 núcleos, 4-8 GB RAM) |
| Alta Carga | 50+ | 6 - 10 segundos | VPS / Nube (CPU 4-8+ núcleos, 8-16+ GB RAM) |

**Consideraciones sobre la Escalabilidad:**

* **Interacciones entre Agentes:** A medida que el número de agentes aumenta, el número de posibles interacciones entre ellos en cada paso de tiempo se incrementa cuadráticamente en un espacio denso. Sin embargo, en el modelo actual, las interacciones de contagio se limitan a los vecinos directos, lo que ayuda a mitigar este crecimiento exponencial.
* **Tamaño de la Cuadrícula:** Una cuadrícula más grande con el mismo número de agentes puede reducir la densidad y, por lo tanto, la frecuencia de las interacciones directas, lo que podría mejorar ligeramente el tiempo de respuesta, aunque la gestión de la cuadrícula sigue siendo una sobrecarga.
* **Recolección de Datos:** La recolección de datos en cada paso implica iterar sobre todos los agentes para contar sus estados. Para un número muy elevado de agentes, esto puede convertirse en un cuello de botella.
* **Visualización:** La renderización de un gran número de agentes en la interfaz web puede consumir recursos del navegador del cliente y, en algunos casos, del servidor si la visualización se actualiza con mucha frecuencia. Para escenarios de alta carga, se podría considerar desactivar o simplificar la visualización en tiempo real para priorizar la velocidad de la simulación.
* **Optimización del Código:** El rendimiento puede mejorarse a través de la optimización del código del agente y del modelo, reduciendo operaciones redundantes o utilizando estructuras de datos más eficientes. Mesa ya utiliza estructuras optimizadas como MultiGrid, pero las lógicas específicas de los agentes pueden ser ajustadas.
* **Paralelización:** Para simulaciones a gran escala con un número masivo de agentes (cientos de miles o millones), sería necesario explorar técnicas de paralelización o distribución de la carga de trabajo en múltiples núcleos o máquinas, lo cual va más allá del alcance de la implementación actual en Mesa que se ejecuta en un solo proceso.