# Glosario de términos

## Leyenda

Término: puede no tener valor.

Término: debe tener un valor.

## Environment

* Width (µm): *double*.
* Height (µm): *double*.
* Length (µm): *double*.

## General configuration

* numberOfProccesorsCreation: número de procesadores usados en la creación de los agentes, por ahora solamente se recomienda usar 1 (*integer*).
* numberOfProccesors: número de procesadores usados en la ejecución de la simulación. En caso de no tener valor se cogen todos los disponibles (*integer*).
* activateGUI: ejecuta el simulador con o sin interfaz (*boolean*).
* readFromCheckpoint: permite cargar un checkpoint guardado anteriormente. Es necesario mantener la misma configuración para que su resultado no varíe (*string con el path al fichero*).
* totalTries: número de intentos totales para introducir agentes si se da una posición ocupada (*integer*).
* *insertAllAgents: deprecado.*
* numberOfJobs: número de repeticiones que se van a realizar para una simulación sin interfaz (*integer*).
* simulationName: nombre de la simulación (*string*).
* numberOfSteps: número de steps que realizará el simulador para una simulación sin interfaz (*integer*).
* saveSimulationEvery: indica cada cuantos timesteps se deben guardar datos para una simulación sin interfaz (*integer*).
* writeResultsEvery: indica cada cuantos timesteps se debe generar un checkpoint para una simulación sin interfaz (*integer*).
* dirOutput: especifica un directorio donde escribir los datos de salida del simulador. Si está vacío se escriben en la ruta donde se ejecute el simulador (*string con path*).
* fileOutput: especifica un nombre de salida para los ficheros de resultados (*string*).
* emailTo: especifica cuentas de email, separadas por tabulación a donde mandar los resultados una vez la simulación termine (*email\temail*).

## Unity

* unityName: especifica el nombre de la unidad (*string*).
* unityMW (g/mol): especifica el peso molecular de la unidad. Si es vacío se calculará automáticamente (*double*).
* unityRadius (µm): especifica el radio de la unidad. Si es vacío se calculará automáticamente (*double*).
* unityDR0 (µm2/s): especifica la difusión de la unidad en el exterior (*double*).
* unityDR1 (µm2/s): especifica la difusión de la unidad en la membrana externa (*double*).
* unityDR2 (µm2/s): especifica la difusión de la unidad en el periplasma externo (*double*).
* unityDR3 (µm2/s): especifica la difusión de la unidad en el peptidoglicano (*double*).
* unityDR4 (µm2/s): especifica la difusión de la unidad en el periplasma interno (*double*).
* unityDR5 (µm2/s): especifica la difusión de la unidad en la membrana interna (*double*).
* unityDR6 (µm2/s): especifica la difusión de la unidad en el citoplasma (*double*).

## Cell/Layer

* totalCellsType: especifica el número total de tipos de células que aparecerán en la simulación (*integer*).
* cellNameX: especifica el nombre de la célula (capa más externa) (*string*).
* cellRadiusX (µm): especifica el radio de la célula (*double*).
* cellHeightX (µm): especifica la altura (h) de la célula (*double*).
* cellColorX: especifica el color de la célula (capa más externa) (*hexadecimal*).
* cellNumberX: especifica el número de células de este tipo que serán representadas (*integer*).
* cellFormX: especifica la forma de esta célula (*capsule, sphere, hemisphere*).
* cellLayersX: especifica el número de capas que tendrá esta célula (contando desde la externa que ya se ha definido) (*integer*).
* layerNameX\_Y: especifica el nombre de la capa (*string*).
* layerRadiusX\_Y (µm): especifica el radio de la capa (*double*).
* layerHeightX\_Y (µm): especifica la altura de la capa (*double*).
* layerColorX\_Y: especifica el color de la capa (*hexadecimal*).

## Transporters (después de Agents)

* totalTransportersType: especifica el tipo distinto de transportadores que se van a representar en la simulación (*integer*).
* transporterNameX: especifica el nombre del transportador (*string*).
* transporterCellNameX: especifica el tipo de célula al que estará asociado el transportador (*cellName*).
* transporterRadiusX (µm): especifica el radio del transportador (*double*).
* transporterDiffusionRateX (µm2/s): especifica el ratio de difusión que tendrán los agentes que se encuentren atravesando el transportador (*double*).
* transporterColorX: especifica el color del transportador (*hexadecimal*).
* transporterNumberX: especifica el número de transportadores que se representarán en la simulación (*integer*).
* transporterOuterLayerX: especifica en que capa a estar la entrada del transportador (*cell/layerName*).
* transporterInnerLayerX: especifica en que capa va a estar la salida del transportador (*cell/layerName*).
* transporterGetFromX: especifica en que capa va a obtener sus agentes (*exterior,* *cell/layerName*).
* transporterPutToX: especifica en que capa va a soltar sus agentes (*exterior,* *cell/layerName*).
* transporterTypeX: especifica el tipo de transportador (*uniporter, symporter, antiporter*).
* transporterInputsX: especifica que agentes va a poder obtener en su entrada (*agentName, null*).
* transporterOutputsX: especifica que agentes va a poder obtener en su salida (*agentName, null*).

## Agents (Moleculas)

* totalAgentsType: especifica el tipo de agentes distintos que se van a representar (*integer*).
* agentNameX: especifica el nombre del agente (*string*).
* agentMWX: especifica el peso molecular del agente. Si es vacío se calcula automáticamente (*double*).
* agentRadiusX: especifica el radio del agente. Si es vacío se calcula automáticamente (*double*).
* agentDR0\_X (µm2/s): especifica la difusión del agente en el exterior. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR1\_X (µm2/s): especifica la difusión del agente en la membrana externa. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR2\_X (µm2/s): especifica la difusión del agente en el periplasma externo. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR3\_X (µm2/s): especifica la difusión del agente en el peptidoglicano. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR4\_X (µm2/s): especifica la difusión del agente en el periplasma interno. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR5\_X (µm2/s): especifica la difusión del agente en la membrana interna. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentDR6\_X (µm2/s): especifica la difusión del agente en el citoplasma. Si es vacío se calcula automáticamente (no recomendado) (*double*).
* agentColorX: especifica el color del agente (*hexadecimal*).
* agentNumberX: especifica el número de agentes a representar (*integer*).
* agentMaxLayerX: especifica hasta que capa puede atravesar por difusión pasiva el agente (*exterior, cell/layerName*).
* agentMinLayerX: especifica de que capa no puede salir el agente (*exterior, cell/layerName*).
* agentCellLocalizationX: especifica si el agente se creará en una célula o en el exterior (*exterior, cellName*).
* agentLayerLocalizationX: especifica en que capa de la célula se creará el agente (*exterior, cell/layerName*).
* agentRadInflX: especifica el radio de influencia de este agente (*integer*).
* agentRadInflWithX: especifica contra quien puede usar su radio de influencia (*agentName*).

## Feeder

* totalFeederType: especifica el tipo de feeder que se van a representar (*integer*).
* feederTries: especifica el número de intentos del feeder para insertar nuevos agentes (*integer*).
* feederCreateX: especifica que agente ha de crear el feeder (*agentName*).
* feederMaxConcentrationX: especifica el número máximo de agentes a crear que va a poder haber en la simulación (*integer*).
* feederCreationTriggerByCellX: especifica si ha de crear un nuevo agente cuando se de este evento (ese agente llega a una capa de la célula indicada). Ejemplo: cellA,cytoplasm (*cellName,cell/layerName*).
* feederCreationTriggerByStepX: especifica si ha de crear un nuevo agente cada X timesteps (*integer*).
* feederCreationTriggerByReactionX: especifica si ha de crear un nuevo agente cuando se produce una reacción con ese agente (*boolean*).

## Events

### Kill events

* totalKillEvents: especifica el tipo de kill events a representar (*integer*).
* killInputX: especifica el agente que va a recibir este evento (*agentName*).
* killLayerX: especifica si el evento se va a activar cuando el agente entre en una capa (*cell/layerName*).
* killReboundWithX: especifica si el evento se va a activar cuando el agente rebote contra una capa, contra el ambiente o contra otro agente (*exterior, cell/layerName, agentName*)

### Reaction events

* totalReactionEvents: especifica el tipo de reaction events a representar (*integer*).
* reactionInputX: especifica los dos agentes que van a recibir este evento (*agentName\tagentName*).
* reactionOutputX: especifica los agentes que se van a crear al finalizar este evento, 1 o más agentes (*agentName\tagentName\t…*).
* reactionKMX: especifica el KM de este evento (*double*).
* reactionKCATX: especifica el KCAT de este evento (*integer*).

### Transform events

* totalTransformEvents: especifica el tipo de transforma events a representar (*integer*).
* transformInputX: especifica el agente que va a recibir este evento (*agentName*).
* transformOutputX: especifica el agente que va a surgir al finalizar este evento (*agentName*).
* transformLayerX: especifica si el evento se va a activar cuando el agente de entrada entre en una capa (*cell/layerName*).
* transformReboundWithX: especifica si el evento se va a activar cuando el agente de entrada rebote contra el exterior, una capa u otro agente (*exterior, cell/layerName, agentName*).