



Movilidad Urbana

Entrega Final

Integrantes

Elí Salomón Martínez Hernández

Arturo Alfaro González

Rodrigo Aldahir Rosete Flores

Isaac Jacinto Ruiz

Profesores

José Daniel Azofeifa Ugalde

Oscar Francisco Fuentes Casarrubias

Modelación de sistemas multiagentes con gráficas computacionales

Índice

Índice	2
Diagramas de clase	3
Protocolos de interacción	4
Plan de Trabajo	4
Implementación completa de los agentes	5
Implementación completa de la interfaz gráfica de la simulación	6
Proceso de instalación, configuración y ejecución de la simulación	8
Instalación	8
Paso 1:	8
Crear cuenta en IBM Cloud: https://cloud.ibm.com/registration	8
Paso 2:	9
Descargar e instalar IBM Cloud CLI: https://github.com/IBM-Cloud/ibm-cloud-cli-release/releases/	9
Paso 3:	9
Descargar e instalar Unity: https://unity3d.com/es/get-unity/download	9
Paso 4:	10
Descargar unitypackage y el modelo de multiagentes: https://drive.google.com/drive/folders/170_cwPJ1crFpFbH9aZCrIVYfdwHj8Z5p?usp=share_link	10
Paso 1:	10
Entrar a IBM Cloud y seleccionar nueva app de Cloud Foundry en Python: https://cloud.ibm.com/cloudfoundry/overview	10
Paso 2:	11
En localización elegir Dallas (us-south), darle nombre a la app y dar clic en crear.	11
Paso 3:	12
En el CLI iniciar sesión con el comando <code>ibmcloud login</code> e introducir mail y contraseña de IBM Cloud y seleccionar la región: <code>us-south</code> .	12
Paso 4:	12
Ejecutar el comando <code>ibmcloud cf install</code> .	12
Paso 5:	12
Configurar API endpoint con el comando <code>ibmcloud api</code> https://api.ng.bluemix.net .	12
Paso 6:	12
Configurar target con el comando <code>ibmcloud target -cf</code> .	13
Paso 7:	13
Ejecutar el comando <code>ibmcloud cf apps</code> .	13
Paso 8:	13
Editar el archivo <code>manifest.yml</code> y cambiar name por el servidor creado en la nube.	13
Paso 9:	13
Ubicarse en la carpeta donde está el zip descargado anteriormente y ejecuta el comando <code>ibmcloud cf push</code> y verificar que el resultado diga estado en ejecución.	13
Paso 1:	13
Abrir Unity y crear un nuevo proyecto en 3D.	14
Paso 2:	14
Assets > Import Package > Custom Package > seleccionar archivo unitypackage descargado y dar clic en import.	14
Paso 3:	14
Cargar escena principal.	14
Paso 4:	14
En el archivo <code>webClient.cs</code> cambiar string url por el url de IBM Cloud.	14
Paso 5:	14
Correr la simulación en Unity.	14
Análisis de la solución desarrollada	15
Proceso de aprendizaje	16
Elí:	16

Rodrigo:

16

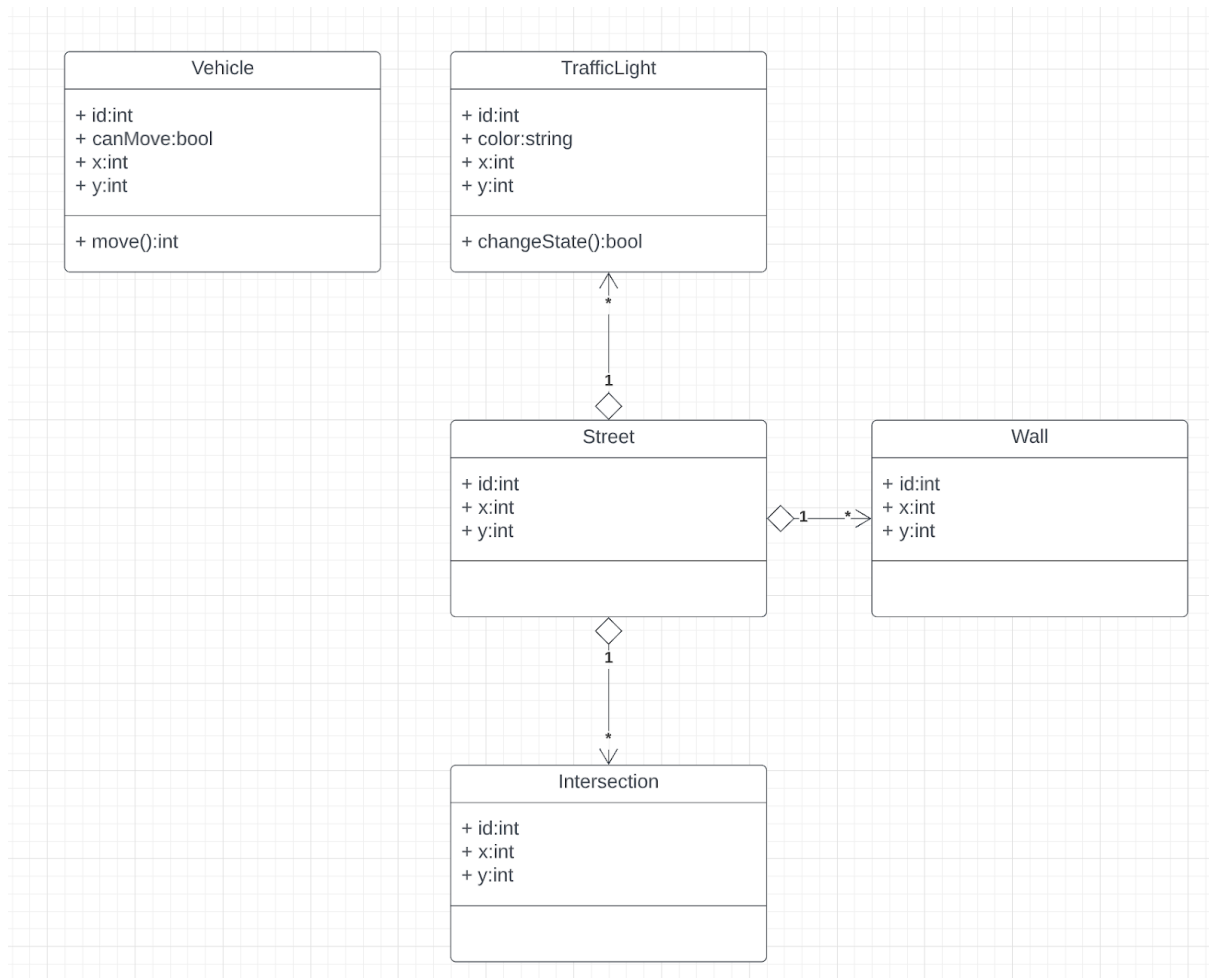
Arturo:

16

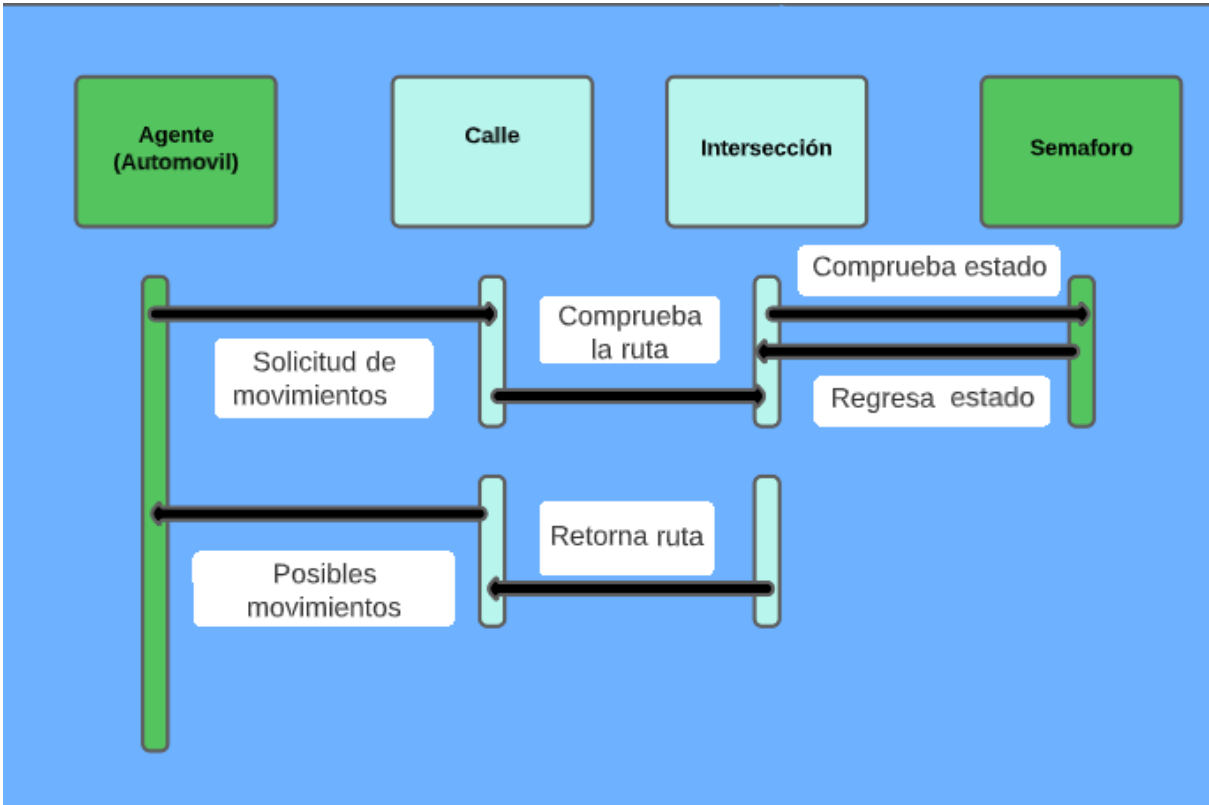
Isaac:

16

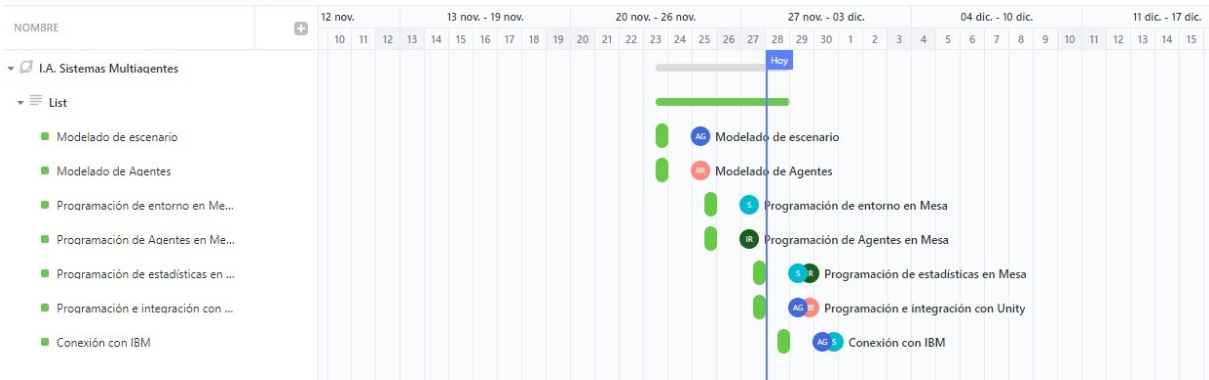
Diagramas de clase



Protocolos de interacción



Plan de Trabajo



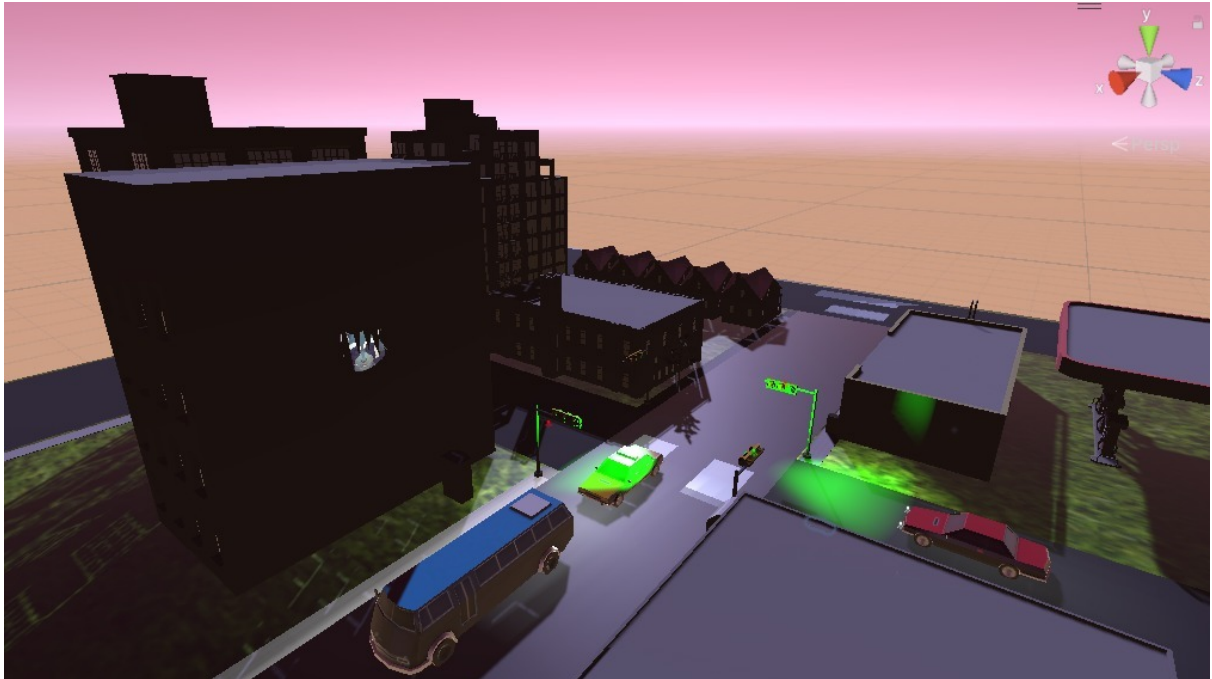
Implementación completa de los agentes

```
class SideWalk(mesa.Agent):  
    def __init__(self, x, y):  
        super().__init__(x, y)
```

```
class TrafficLight(mesa.Agent):  
    # Agent that represents a traffic light  
  
    def __init__(self, unique_id, pos, model):  
        super().__init__(unique_id, model)  
        # Set the position of the traffic light  
        self.pos_x = pos[0]  
        self.pox_y = pos[1]  
  
        # Set the state of the traffic light  
        self.state = self.set_state()  
  
        # Set the timer of the traffic light  
        self.timer = 20
```

```
class Vehicle(mesa.Agent):  
    # Agent that represents a vehicle  
  
    def __init__(self, unique_id, pos, direction, percentagepesero, model):  
        super().__init__(unique_id, model)  
        # Set the position of the vehicle  
        self.pos_x = pos[0]  
        self.pos_y = pos[1]  
  
        # Set the direction of the vehicle  
        self.direction_x = direction[0]  
        self.direction_y = direction[1]  
  
        # Set the speed of the vehicle  
        # self.speed = 1  
        self.speed = random.randint(1, 4)
```

Implementación completa de la interfaz gráfica de la simulación



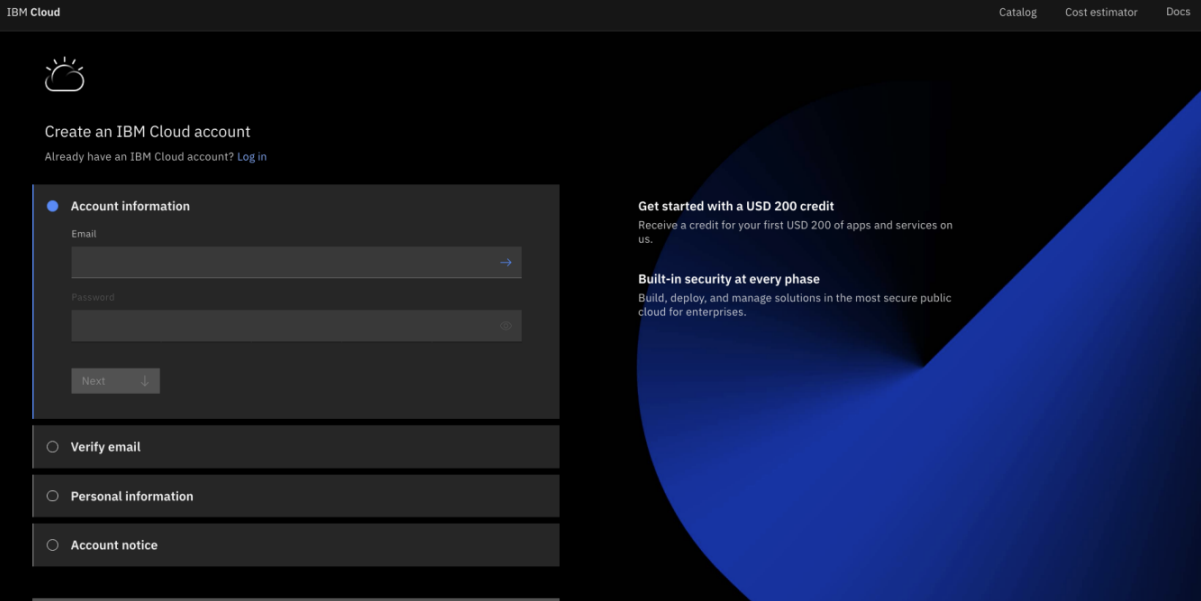


Proceso de instalación, configuración y ejecución de la simulación

Instalación

Paso 1:

Crear cuenta en IBM Cloud: <https://cloud.ibm.com/registration>



The screenshot shows the IBM Cloud registration page. At the top, there's a navigation bar with 'IBM Cloud' on the left and 'Catalog', 'Cost estimator', and 'Docs' on the right. Below the navigation bar is a header section with the IBM Cloud logo and the text 'Create an IBM Cloud account' and 'Already have an IBM Cloud account? Log in'. The main content area is divided into two columns. The left column contains a form for 'Account information' with fields for 'Email' and 'Password', and a 'Next' button. Below the form are three radio buttons for 'Verify email', 'Personal information', and 'Account notice'. The right column contains two promotional messages: 'Get started with a USD 200 credit' and 'Built-in security at every phase'.

IBM Cloud

Catalog Cost estimator Docs

Create an IBM Cloud account
Already have an IBM Cloud account? Log in

Account information

Email

Password

Next

☐ Verify email

☐ Personal information

☐ Account notice

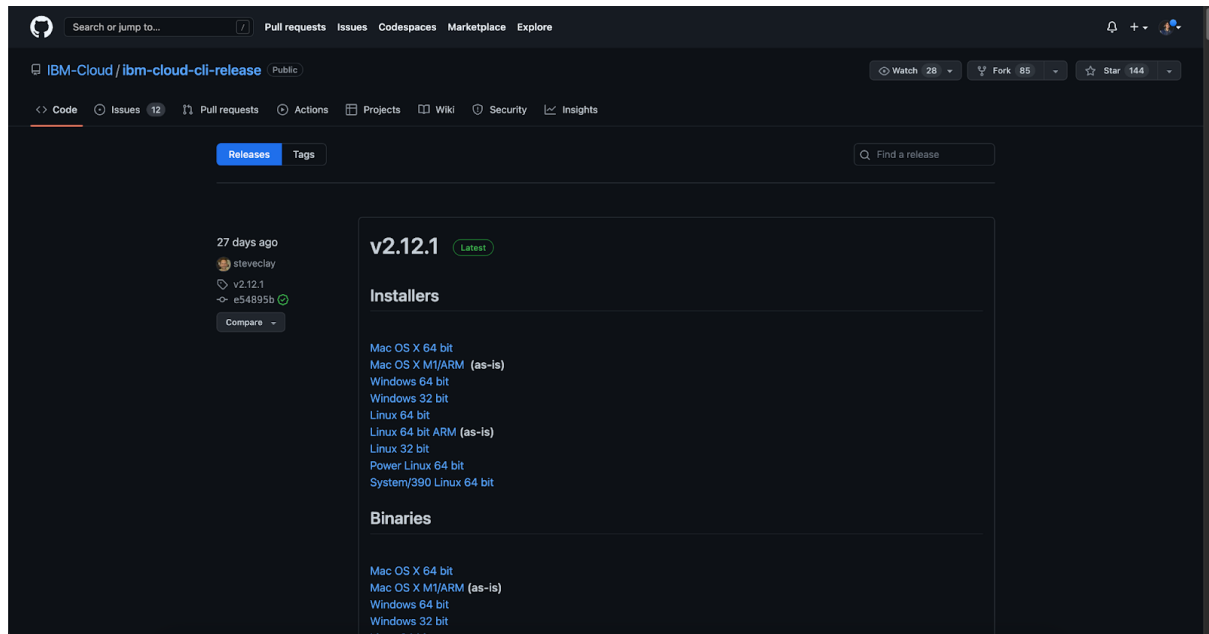
Get started with a USD 200 credit
Receive a credit for your first USD 200 of apps and services on us.

Built-in security at every phase
Build, deploy, and manage solutions in the most secure public cloud for enterprises.

Paso 2:

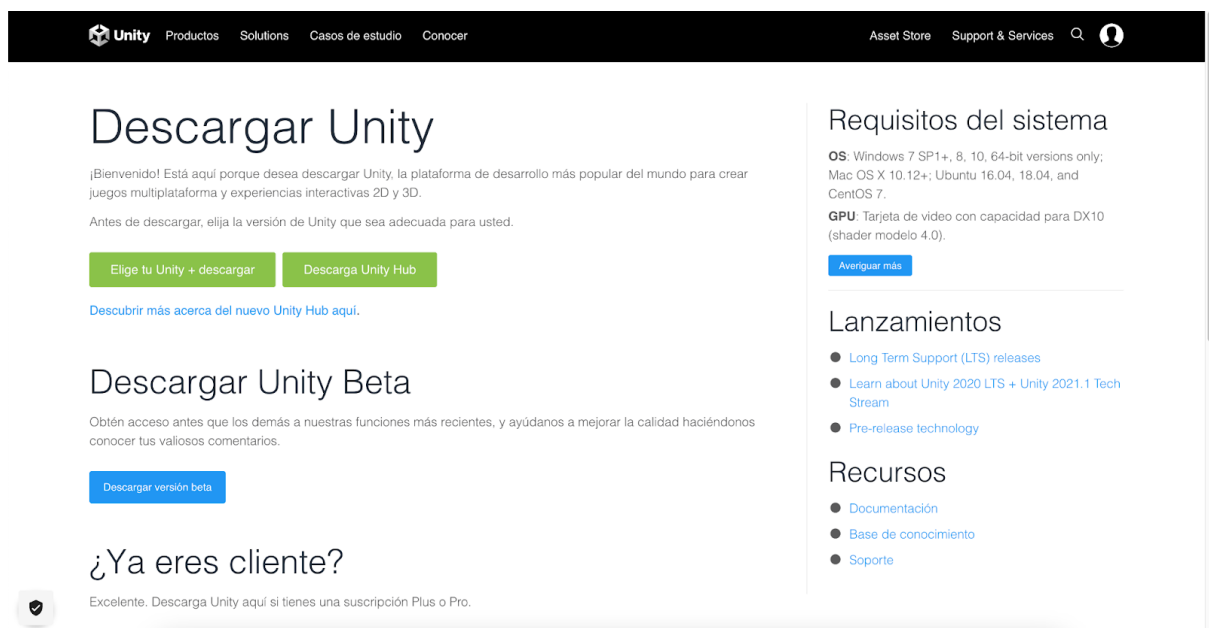
Descargar e instalar IBM Cloud CLI:

<https://github.com/IBM-Cloud/ibm-cloud-cli-release/releases/>



Paso 3:

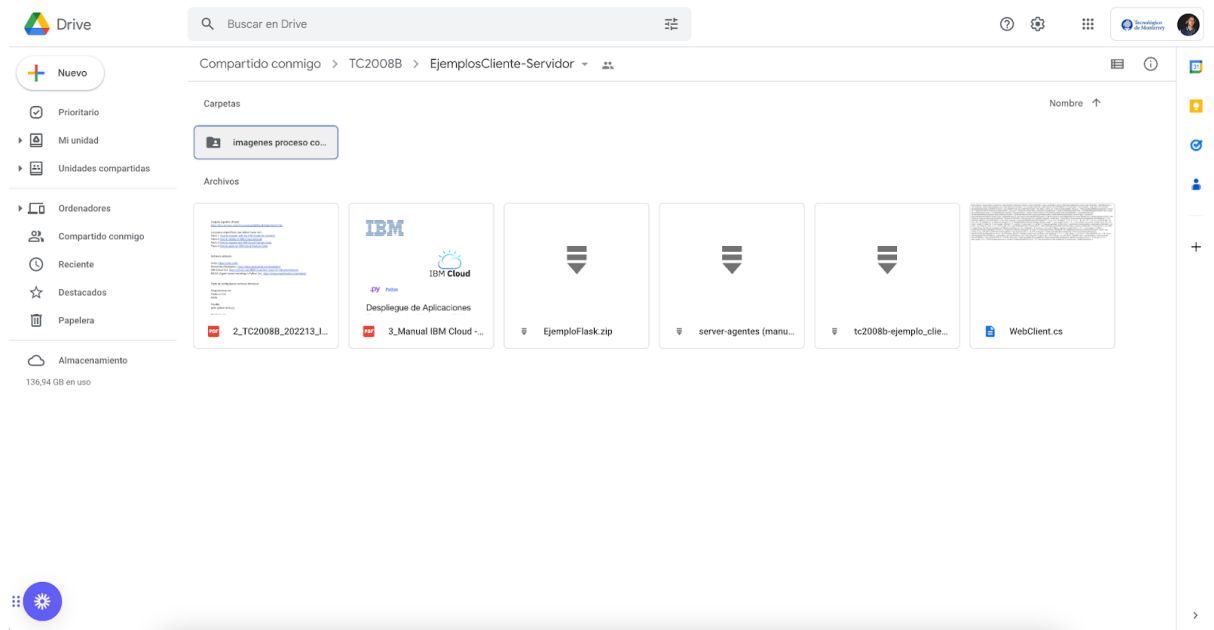
Descargar e instalar Unity: <https://unity3d.com/es/get-unity/download>



Paso 4:

Descargar unitypackage y el modelo de multiagentes:

https://drive.google.com/drive/folders/170_cwPJ1crFpFbH9aZCrIVYfdwHj8Z5p?usp=share_link

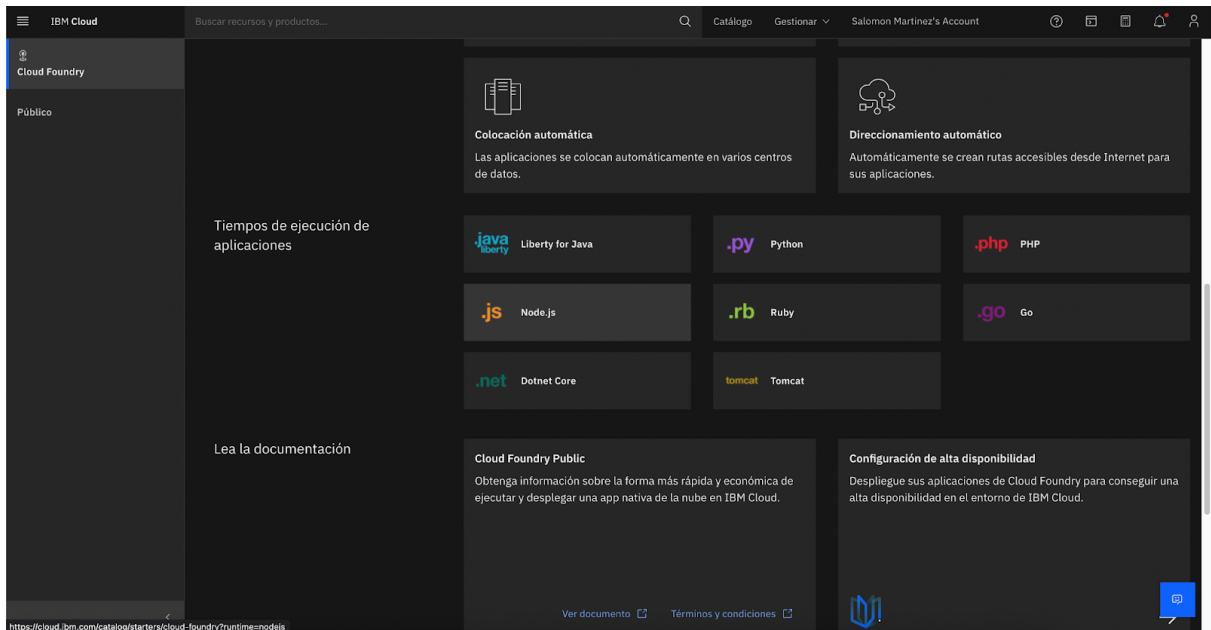


Configuración

Paso 1:

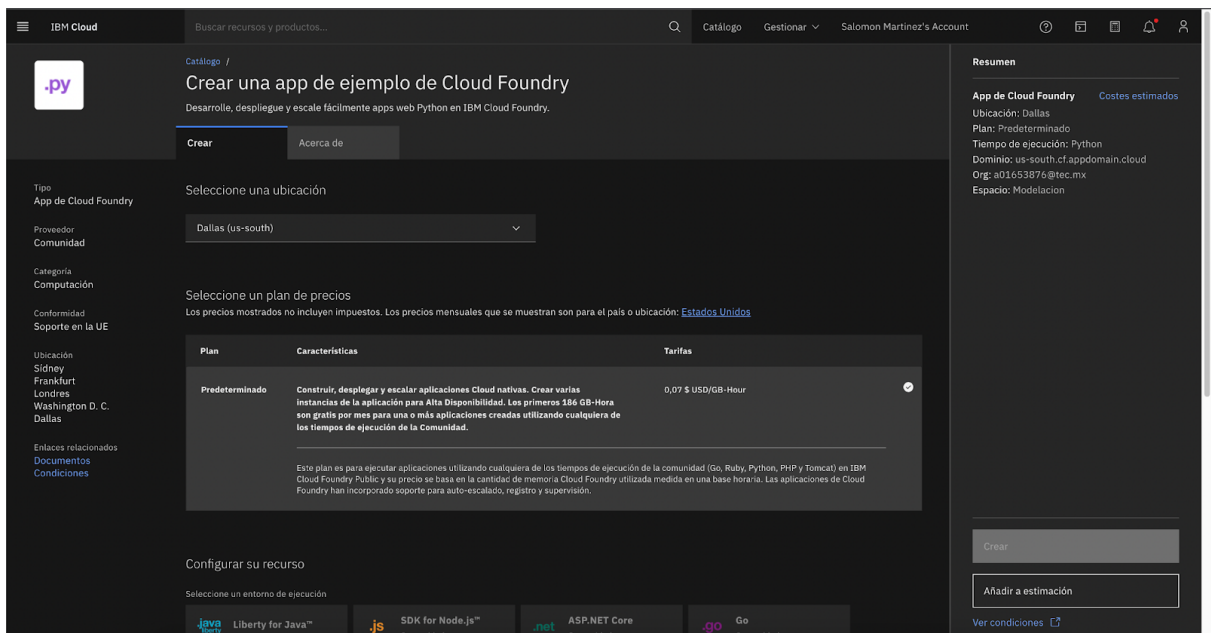
Entrar a IBM Cloud y seleccionar nueva app de Cloud Foundry en Python:

<https://cloud.ibm.com/cloudfoundry/overview>



Paso 2:

En localización elegir Dallas (us-south), darle nombre a la app y dar clic en crear.



Paso 3:

En el CLI iniciar sesión con el comando `ibmcloud login` e introducir mail y contraseña de IBM Cloud y seleccionar la región: `us-south`.

```
a01653876@cloudshell:~$ ibmcloud login
API endpoint: https://cloud.ibm.com
Region: us-south

Email> a01653876@tec.mx

Password>
Authenticating...
OK

Targeted account Salomon Martinez's Account (7a943ae888e74f6fb461ec8969359317)
```

Paso 4:

Ejecutar el comando `ibmcloud cf install`.

```
a01653876@cloudshell:~$ ibmcloud cf install
Installed Cloud Foundry CLI version is '6.53.0'. Do you want to install and use latest version? [y/N] > y
[[IBM MESSAGE NOT FOUND]]
Attempting to download Cloud Foundry CLI...
8.56 MiB / 8.56 MiB |=====| 100.00% 0s
<no values> bytes downloaded
Saved in /tmp/ic/cloudshell-37b9c353-1246-4805-9284-0f37324e91b8-1-78df747bjn74n-1/.bluemix/tmp/cf_902959836/cf-cli_6.53.0_linux_x86-64.tgz
Installing Cloud Foundry CLI...
OK
Cloud Foundry CLI is successfully installed
```

Paso 5:

Configurar API endpoint con el comando `ibmcloud api` <https://api.ng.bluemix.net>.

```
a01653876@cloudshell:~$ ibmcloud api https://api.ng.bluemix.net
Setting api endpoint...
API endpoint https://api.ng.bluemix.net is going to be deprecated. Use https://cloud.ibm.com.
OK

API endpoint: https://cloud.ibm.com
Not logged in. Use 'ibmcloud login' to log in.
```

Paso 6:

Configurar target con el comando `ibmcloud target --cf`.

```
a01653876@cloudshell:~$ ibmcloud target --cf
!!i18N_MESSAGE_NOT_FOUND!!
Targeted Cloud Foundry (https://api.us-south.cf.cloud.ibm.com)

Targeted org a01653876@tec.mx

Targeted space Modelacion

API endpoint:      https://cloud.ibm.com
Region:           us-south
User:             a01653876@tec.mx
Account:          Salomon Martinez's Account (7a943ae888e74f6fb461ec8969359317)
Resource group:    No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'
CF API endpoint:   https://api.us-south.cf.cloud.ibm.com (API version: 2.189.0)
Org:              a01653876@tec.mx
Space:            Modelacion
```

Paso 7:

Ejecutar el comando `ibmcloud cf apps`.

```
a01653876@cloudshell:~$ ibmcloud cf apps
Invoking 'cf apps'...

Getting apps in org a01653876@tec.mx / space Modelacion as a01653876@tec.mx...
OK

name           requested state  instances  memory  disk  urls
a01653876      started          1/1        128M    1G    a01653876.us-south.cf.appdomain.cloud
```

Paso 8:

Editar el archivo `manifest.yml` y cambiar name por el servidor creado en la nube.

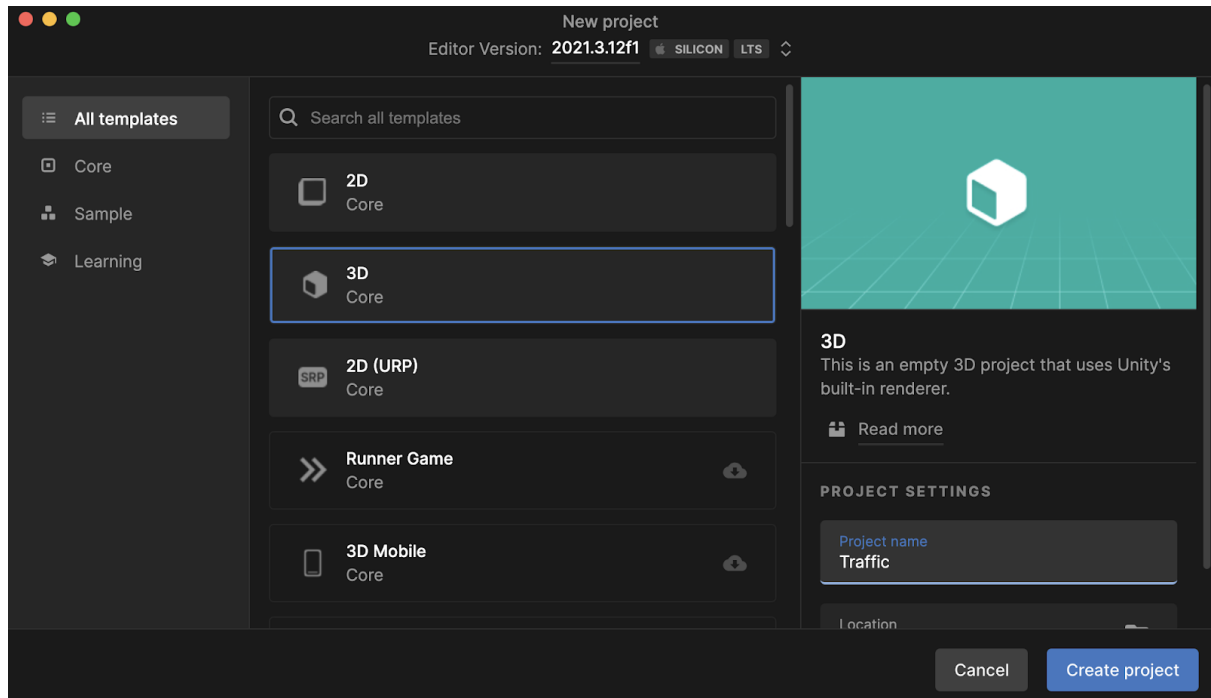
Paso 9:

Ubicarse en la carpeta donde está el zip descargado anteriormente y ejecuta el comando `ibmcloud cf push` y verificar que el resultado diga estado en ejecución.

Ejecución

Paso 1:

Abrir Unity y crear un nuevo proyecto en 3D.



Paso 2:

Assets > Import Package > Custom Package > seleccionar archivo unitypackage descargado y dar clic en import.

Paso 3:

Cargar escena principal.

Paso 4:

En el archivo webClient.cs cambiar string url por el url de IBM Cloud.

Paso 5:

Correr la simulación en Unity.

Análisis de la solución desarrollada

Para la solución de la situación problema se utilizaron diferentes herramientas, para el modelado 3D se utilizó la aplicación Unity, en esta aplicación incluimos los carros, los edificios, semáforos, las calles y todo aquello que se muestra en nuestro entregable que ayude a la visualización del ejemplo de nuestros agentes. De igual forma contamos con la herramienta de IBM Cloud la cual nos ayudó a generar un servidor web con la intención de recibir y mandar información por medio de formatos JSON. Por último utilizamos python e importamos una librería, para poder realizar nuestra simulación en 2D, este fue el primer paso para poder realizar nuestra simulación y así tener de primera instancia una noción de cómo iba interactuar nuestra simulación en Unity.

Los agentes que utilizamos para esta entrega fueron 3, los cuales son 4 carros de diferente diseño cada uno, 4 semáforos y las baquetas correspondientes al diseño de nuestro cruce, estos agentes fueron elegidos debido a que son los principales objetos que necesitamos para poder crear una simulación lo más real posible, pues el funcionamiento de nuestra propuesta consta en darle prioridad a un vehículo que tenga mayor velocidad que otro, esto poniendo el ejemplo de un microbús, el cual es uno de los vehículos principales que producen accidentes por ir a una gran velocidad, por tal motivo cuando pasa el cruce, los semáforos opuestos se ponen de otro color, en este caso puede ser naranja o rojo, para que los otros coches estén advertidos de que un vehículo viene a gran velocidad y necesita pasar, el objetivo de esto es evitar accidentes y dar el acceso a automóviles que requieran cruzar de manera inmediata.

El diseño gráfico presentado fue seleccionado debido a que necesitábamos un diseño donde los coches pudieran interactuar de manera cíclica por lo que nuestro diseño de las calles es de manera cuadrática, así los automóviles pueden dar vuelta en cada esquina y seguir moviéndose sobre el mapa hasta volver a llegar al cruce de las calles de igual forma respetando los semáforos de acuerdo a nuestras condicionales que pusimos en el script. También nuestro ambiente es nocturno, esto más que nada para que las luces de los automóviles y de los semáforos sean más visibles, por último pusimos edificios y casas para hacer más realista la simulación de una ciudad.

Ventajas	Desventajas
Reduce el tráfico	Podría tener problemas con personas que no respeten las reglas de tránsito
Disminuyen el número de accidentes provocados por los microbuses	Le da prioridad a un sentido del cruce y si hay gran cantidad de automóviles que vaya rápido, el tiempo de la otra calle será mayor.

Las modificaciones que podemos realizar en nuestras propuestas, son poner un tiempo límite para que precisamente no haya embotellamiento de vehículos en una solo sentido del cruce

Proceso de aprendizaje

Elí

Durante el desarrollo de esta situación problema, aplicamos diferentes conocimientos que hemos ido aprendiendo a lo largo de la carrera. A su vez aprendimos también cosas nuevas y de gran importancia para poder integrar todo lo que sabemos en un solo proyecto. Desde la generación de código en python usando las librerías de mesa, pandas y numpy; hasta la modelación 3d en unity y el despliegue de la aplicación en el servidor de IBM Cloud.

Asimismo logramos tener un acercamiento con la inteligencia artificial y su desarrollo dentro de un ambiente con sus distintos agentes interactuando. Al ser todo esto algo nuevo para mí, creía que sería algo complejo y complicado de comprender. Sin embargo en un principio pareció así hasta que con ayuda de la práctica y los ejercicios, pude ir comprendiendo mejor. De igual manera la modelación fue algo que me resultó un poco complicado al no tener una idea clara del funcionamiento de la interfaz de unity. Pero al final fue más sencillo de lo que esperaba y creo que logramos un resultado bueno para nuestro modelado.

Rodrigo

Dentro de esta entrega hubieron diferentes conocimientos adquiridos, por ejemplo en el caso de Unity, aprendimos a modelar automóviles, robots, edificios, casas, incluso simular un cruce de calle. De igual forma aprendimos a utilizar colliders, skybox, generar nuevos materiales para la colocación de colores y texturas. Tuvimos una introducción básica de probuilder, el cual tiene diferentes herramientas para conseguir un modelado más estético y poder realizar los modelados de la primera y segunda evidencia. Por otro lado a lo largo del curso obtuvimos conocimientos para poder modelar nuestra situación problema en 2D, esto con ayuda de python y una librería llamada "mesa", con esto desarrollamos una simulación donde el agente interactúa con el ambiente. Por último tuvimos la oportunidad de crear un servidor en IBM cloud, el cual nos ayudó a enviar y recibir información con ayuda de formatos JSON, esto me pareció muy interesante pues es la primera vez que trabajo con IBM.

Arturo

Al inicio del semestre, al escuchar que la clase iba estar relacionada con agentes, esperaba aprender lo básico en cuanto a Inteligencia Artificial, agentes y su interacción con el ambiente. Durante todo el bloque, aprendimos a utilizar estos elementos para solucionar problemas, e también implementamos algoritmos que nos ayudarían a optimizar nuestra solución. Gracias a esto pudimos aprender las bases de una área que me parece bastante interesante, sin embargo debido al tiempo, solo pudimos ver los temas generales, aunque también eso me pareció correcto. Ya que al ser temas algo complejos sería algo complicado querer cubrirlos en tan poco tiempo. Al final la simulación quedó como esperábamos y aprendí bastante de todos los elementos que hacen el funcionamiento de una simulación por agentes y su ambiente.

Isaac

La modelación de sistemas multiagentes es un tema completamente nuevo para mi, por lo que al principio me pareció que iba a ser un tema sumamente interesante y complejo. Aprender la diferencia entre un agente y el ambiente fue una tarea fácil de comprender, sin embargo, me di cuenta de que la cantidad de respuestas o comportamientos que puede tener el agente en base a lo que ocurre a su alrededor es exponencialmente mayor. Además, es importante conocer que existen diferentes tipos de agentes, y estos se catalogan de acuerdo al funcionamiento que uno desea en ellos. Por otro lado, me pareció muy interesante el hecho de haber tenido la oportunidad de montar un servidor en IBM Cloud para ver el funcionamiento de los archivos python desde la nube y no solo de manera local (Local Host).