UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN BIOINFORMÁTICA

Programación Avanzada
Proyecto - Unidad III

Objetivo

El objetivo es poder trabajar con los conceptos asociados al modelado de problemas mediante el paradigma de orientación a objetos considerando los saberes conceptuales y aplicarlos al contexto adecuado. Al término de este trabajo, se espera que el estudiante sea capaz de:

- Desarrollar una solución computacional que aplique el paradigma de programación orientada a objetos acorde a los objetivos y requerimientos de un problema específico.
- Aplicar el uso de una API de interfaz gráfica en el desarrollo de una solución computacional. (opcional)

Contexto

Las enfermedades infecciosas generan mucho entusiasmo en torno al modelado matemático. Un modelo matemático, y seguramente el más conocido, es el modelo SIR[1], que a pesar de ser demasiado simplificado para describir la realidad, el resultado de hecho "se comporta" como el de una pandemia real.

El modelo SIR, acrónimo de Susceptible-Infectado-Recuperado, es un modelo matemático ampliamente utilizado para estudiar y simular la propagación de enfermedades infecciosas en una población. En este modelo, se considera que la población se divide en tres categorías: susceptibles, que son individuos que aún no han sido infectados; infectados, que son aquellos que actualmente están infectados y pueden transmitir la enfermedad; y recuperados, que son aquellos que se han recuperado de la enfermedad y han adquirido inmunidad.

Las fórmulas del modelo SIR son:

Ecuación de Susceptibles:

$$\frac{dS}{dt} = -\frac{\beta \cdot S \cdot I}{N}$$

Ecuación de Infectados:

$$\frac{dI}{dt} = \beta \cdot S \cdot I - \gamma \cdot I$$

Ecuación de Recuperados:

$$\frac{dR}{dt} = \gamma \cdot I$$

En donde:

- d representa el diferencial, que describe cómo cambia una variable con respecto a otra.

[1] https://es.wikipedia.org/wiki/Modelo_SIR

- N representa la población total.
- S representa la cantidad de individuos susceptibles.
- I representa la cantidad de individuos infectados.
- R representa la cantidad de individuos recuperados.
- β es la tasa de transmisión.
- γ es la tasa de recuperación.

En este proyecto no profundizaremos ni defenderemos ningún modelo matemático existente, pero buscaremos simular una enfermedad infecciosa en una población de alguna forma. Esto permitirá a los estudiantes o interesados utilizar una herramienta de este tipo e investigar diferentes estrategias para gestionar una enfermedad, así como intentar predecir las características de la enfermedad en sí misma.

La tarea consiste en crear un simulador para observar cómo se comportaría una enfermedad infecciosa en una población, mediante la creación de un modelo simplificado de la realidad basado en el modelo SIR. Para lograr esto, debemos establecer que los escenarios de este simulador se ejecuten en **pasos (steps)**, que pueden representar una hora, un día, un mes, o cualquier unidad de tiempo seleccionada. La condición es que la elección siempre debe ser una unidad de tiempo. Así, la simulación avanza por pasos.

En este tipo de desarrollo es importante definir la **aleatoriedad**, pues deben existir algunas partes en el simulador que son aleatorias (puede determinar la propiedad μ de la distribución normal o elegir por supuesto otro tipo de distribución - véase Figure 1). Es importante porque como en la realidad misma hay eventos aleatorios.

Se definirá que **un ciudadano o persona** será susceptible de enfermarse y es probable que se enferme, o bien, que haya sido tratado (en caso de que el ciudadano infectado esté recuperado o muerto). Por ahora, utilizaremos la condición de que un ciudadano infectado y recuperado no puede volver a enfermarse; esta es la suposición básica en la mayoría de los modelos de enfermedades infecciosas.

Las personas, al vivir en comunidad, pueden socializar y, por ende, tener **contacto entre sí**. Al encontrarse físicamente, existe la posibilidad de contagiarse mutuamente. Los **contactos** entre cada par de objetos simulados (o ciudadanos) pueden ser de dos tipos: o no están conectados (no se ven físicamente) o sí lo están. En caso de estar conectados, hay dos opciones:

1. Son contactos estrechos, lo cual significa que forman parte del mismo grupo de ciudadanos que se encuentran en un mismo lugar (se puede pensar en una familia que comparte el mismo hogar).
2. No están conectados de forma regular, es decir, la conexión es aleatoria, lo que implica que cada objeto tiene una cierta probabilidad de establecer contacto con otros ciudadanos (al azar dentro de las posibilidades).

Clase definidas o clases base

A continuación se presentan clases previamente definidas con el fin de guiar al estudiante a un desarrollo acotado y controlado para el desarrollo. En otras palabras son requerimientos bases y funcionales del sistema.

La enfermedad

En nuestra simulación (aplicación), la población está luchando contra una enfermedad infecciosa. La enfermedad (clase) en nuestro simulador tendrá cuatro atributos esenciales:

- infección_probable: La probabilidad de que un ciudadano sano sea infectado por un ciudadano enfermo como resultado de una reunión física.
- promedio_pasos: el tiempo promedio (número de pasos) que la enfermedad es infecciosa antes de ser declarada "De alta" o muerta.
- enfermo: valor por defecto para el ciudadano no está infectado
- contador: valor para el número de pasos (tiempo).

El ciudadano o persona

Se requiere simular a un ciudadano en una comunidad simulada y los principales atributos que tendrá esta clase son:

- comunidad: para identificar a la comunidad a la que pertenece el ciudadano.
- _id: la identificación del ciudadano.
- nombre y apellido: Para identificar a la persona.
- familia: un identificador para una familia (puede ser el apellido) y el ciudadano esté vinculado a esta familia.
- enfermedad: la enfermedad que el ciudadano puede tener o no.
- estado: si es Verdadero, indica que el ciudadano está sano o muerto.

La comunidad

La comunidad tiene como objetivo fundamental describir a un grupo de personas y las conexiones entre ciudadanos.

Atributos esenciales:

- num_ciudadanos: el número de ciudadanos en la comunidad
- promedio_conexion_fisica: El promedio de conexión física que tiene un ciudadano dentro de la comunidad. La enfermedad se puede propagar solo por un ciudadano infectado haciendo una conexión física con otro ciudadano.
- enfermedad: la enfermedad misma.
- num_infectados: el número inicial de ciudadanos infectados en la comunidad.
- probabilidad_conexion_fisica: la probabilidad de que las conexiones físicas de un ciudadano en la comunidad sea un contacto estrecho.

Otras clases

Deberá existir una clase llamada **Simulador** que se encargará de inicializar y llevar el control de los pasos en la simulación, así como también una clase **main** que inicializará los valores de la aplicación.

Los valores de inicialización pueden ser algo como este ejemplo:

Listing 1: Ejemplo clase main.py

```
from enfermedad import Enfermedad
from comunidad import Comunidad
from simulador import Simulador

covid = Enfermedad(infeccion_probable=0.3,
                   promedio_pasos=18)
talca = Comunidad(num_ciudadanos=20000,
                  promedio_conexion_fisica=8,
                  enfermedad=covid,
                  num_infectados=10,
                  probabilidad_conexion_fisica=0.8)
sim = Simulador()
sim.set_comunidad(comunidad=talca)
sim.run(pasos=45)
```

Objetivos

- Crear una comunidad. Para este caso, se entregará un algoritmos para nombres y apellidos y ser almacenado en formato csv. Este archivo deberá ser leido, y luego, creados como personas parte de una comunidad.
- Definir el tamaño de las comunidades, crear grupos, ciudadanos y vincularlos a la comunidad.
- A partir del conjunto de ciudadanos creados, generar una muestra aleatoria y determinar a los ciudadanos contagiados en la comunidad.
- De los ciudadanos de la comunidad, determinar posibles contactos estrechos, muy en particular, para los ciudadanos ya contagiados.
- De los contactos estrechos, determinar nuevos contagiados si es que existiesen.
- Creados estos conjuntos generar iteraciones o pasos para visualizar como se propaga la enfermedad en la comunidad de acuerdo al contexto entregado. Esto incluye que se debe ingresar a nuevos posibles infectados como también considerar a los muertos o ya curados de la enfermedad.
- En cada paso de la simulación debe entregar una salida con la cantidad de personas infectadas como la cantidad de personas activas del total de la población.
- En cada iteración se deberá crear un archivo csv que almacene a lo menos la clase y los atributos de la clase ciudadano o persona. La razón de esto es, auditar en cada paso que sucedió con la comunidad y las personas.
- Diseñar una interfaz gráfica para el proyecto.
- Opcional - Graficar curva de contagios por iteración utilizando matplotlib.

Apoyo

Referencias prácticas y sencillas que podría considerar sobre el modelo SIR:

- https://culturacientifica.com/2020/08/24/el-modelo-sir-un-enfoque-matematico-de-la-propagacion-de-infecciones/

- https://www.dmsc.es/sir-el-modelo-dinamico-para-el-covid-19-que-inspira-a-las-matematicas-actuales/
- https://pybonacci.org/2020/09/16/modelo-sir-modelo-epidemiologico-con-python/
- https://medium.com/bioinformatic-snippets/el-modelo-epidemiol%C3%B3gico-sir-4f9570884b25

Condiciones

- Avance1: martes 25 y miércoles 26 de junio (10 %).
- Avance2: martes 2 y miércoles 3 de julio.(15 %).
- Entrega final: Antes del 10 de julio.(75 %)
- Se estima el tiempo de desarrollo en 16 bloques de clase (10 bloques presenciales y 6 autónomos)

Ademas:

- Proyecto individual.
- El lenguaje a utilizar es Python.
- Debe utilizar una API gráfica.
- El proyecto en general se debe expresar en conceptos de Orientación a Objetos.
- El código debe estar alojado en un repositorio para su revisión incluyendo solo los archivos .py.
- Será excluyente de aprobación:
  - Utilizar control de versiones (git) con un mínimo de 20 commit.
  - Para leer archivos csv usar pandas
  - Para tratar con listas o arreglos, utilizar numpy.

Anexo

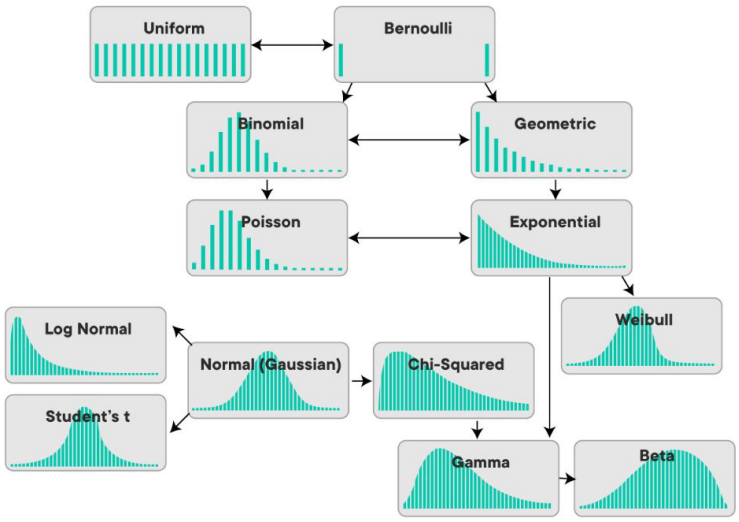

Figura 1: Distribuciones estadísticas

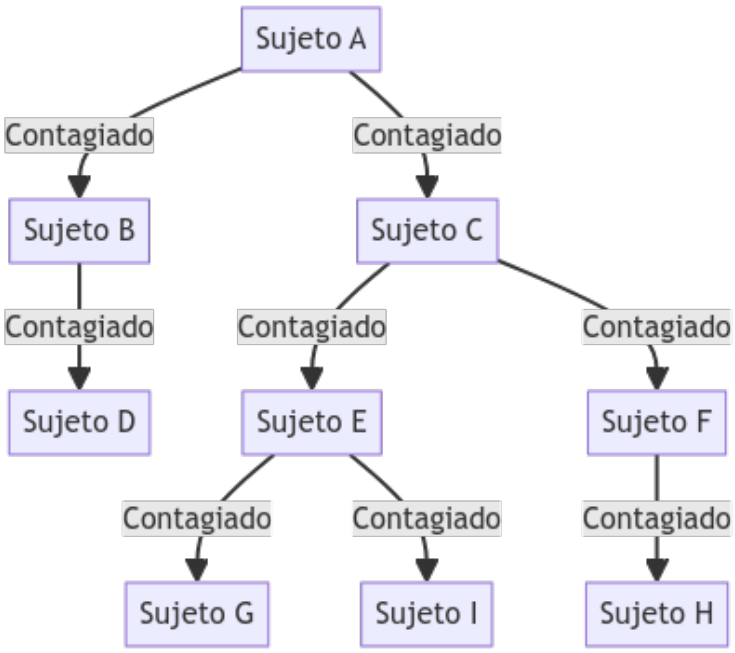

Figura 2: Ejemplo de distribución de contagios

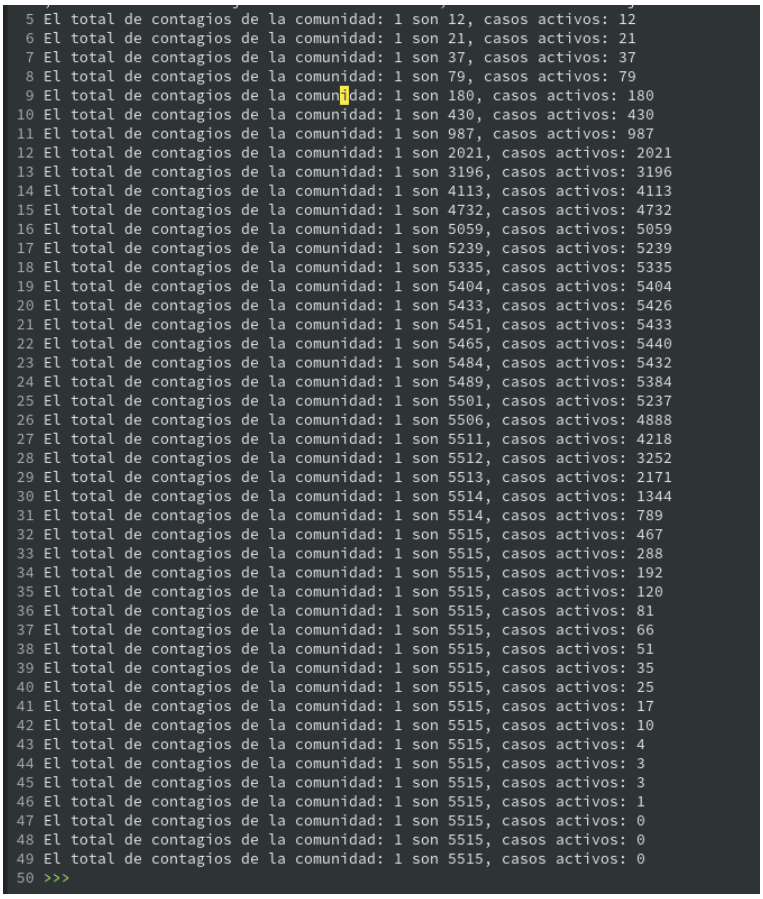

Figura 3: Ejemplo de ejecución

Este ejemplo, la ejecución inició con 12 contagios (linea 5) y que en cada paso ha ido incrementando hasta llegar a un máximo de 5515 contagios. A medida avanza el tiempo los casos activos van variando de acuerdo al parámetro de "promedio_de_pasos" de la clase Enfermedad.

Tenga presente que en muchos casos de acuerdo a la cantidad de pasos, no siempre se llega a un valor cero (0) en los casos activos.

El peak de casos activos se denota en la linea 24 y el número máximo de contagios se determina en la linea 32 de la imagen.

El total de la población (num_ciudadanos) es de 20 mil habitantes o 20 mil objetos de tipo Ciudadano.

Por último y al ser una simulación basada en probabilidad, es muy poco factible que existan dos ejecuciones con resultados iguales, esto a pesar de que los números de entrada en main.py siempre sean iguales.

Listing 2: Creación de nombres y apellidos de personas

```python
import csv
import random

# Lista de nombres y apellidos
nombres = [incluya muchos nombres]
apellidos = [incluya muchos apellidos]

# Generar 1000 nombres y apellidos aleatorios
nombres_aleatorios = random.choices(nombres, k=1000)
apellidos_aleatorios = random.choices(apellidos, k=1000)

# Crear lista de nombres y apellidos combinados
nombres_apellidos = list(zip(nombres_aleatorios, apellidos_aleatorios))

# Escribir los nombres y apellidos en un archivo CSV
with open("nombres_apellidos.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["nombre", "apellido"])
    writer.writerows(nombres_apellidos)

print("Archivo CSV generado con éxito.")
```

Rúbrica de Evaluación del Proyecto

Categoría	Puntos	Descripción
1. Planificación y Organización del proyecto (20 %)		
Planificación del Proyecto	10 %	▪ Excepcional (10-9 puntos): Se presentó un plan detallado del proyecto con hitos claros y fechas límite. La planificación muestra un entendimiento profundo de los requisitos. ▪ Bueno (8-7 puntos): Se presentó un plan del proyecto adecuado con hitos claros, pero con menos detalle. ▪ Satisfactorio (6-5 puntos): El plan del proyecto es básico y no cubre todos los hitos o fechas límite adecuadamente. ▪ Necesita Mejorar (4-0 puntos): No se presentó un plan de proyecto claro o este es inadecuado.
Organización del Código y Archivos	10 %	▪ Excepcional (10-9 puntos): El código y los archivos están organizados de manera clara y lógica, facilitando su comprensión y mantenimiento. ▪ Bueno (8-7 puntos): El código y los archivos están organizados adecuadamente pero con algunos aspectos que podrían mejorarse. ▪ Satisfactorio (6-5 puntos): La organización del código y archivos es básica y podría mejorarse significativamente. ▪ Necesita Mejorar (4-0 puntos): El código y los archivos están desorganizados o son difíciles de seguir.
2. Implementación Técnica del problema (50 %)		

Desarrollo del Modelo SIR	10 %	▪ Excepcional (15-14 puntos): El modelo SIR se desarrolló y aplicó con precisión, utilizando las ecuaciones diferenciales correctamente y adaptándolas al contexto del proyecto. ▪ Bueno (13-11 puntos): El modelo SIR se implementó adecuadamente con algunos errores menores. ▪ Satisfactorio (10-8 puntos): La implementación del modelo SIR tiene errores significativos o falta precisión. ▪ Necesita Mejorar (7-0 puntos): El modelo SIR no se implementó correctamente o se omitió.
Simulación y Algoritmos	30 %	▪ Excepcional (15-14 puntos): La simulación se implementó eficientemente con resultados precisos y se consideraron adecuadamente los aspectos de aleatoriedad. ▪ Bueno (13-11 puntos): La simulación funciona adecuadamente, pero podría optimizarse o presenta errores menores. ▪ Satisfactorio (10-8 puntos): La simulación tiene errores significativos que afectan los resultados. ▪ Necesita Mejorar (7-0 puntos): La simulación no funciona correctamente o no se completó.
Interfaz Gráfica y Visualización	10 %	▪ Excepcional (10-9 puntos): La interfaz gráfica es intuitiva, funcional y proporciona una visualización clara de los resultados. ▪ Bueno (8-7 puntos): La interfaz gráfica es funcional, pero podría ser más intuitiva o visualmente atractiva. ▪ Satisfactorio (6-5 puntos): La interfaz gráfica presenta problemas de usabilidad o diseño significativo. ▪ Necesita Mejorar (4-0 puntos): La interfaz gráfica no es funcional o no se desarrolló.
3. Comunicación (10 %)		

Presentación	10 %	▪ Excepcional (10-9 puntos): La presentación es clara, bien estructurada y cubre todos los aspectos importantes del proyecto de manera comprensible. ▪ Bueno (8-7 puntos): La presentación es adecuada, pero podría mejorar en claridad o estructura. ▪ Satisfactorio (6-5 puntos): La presentación es básica y necesita mejoras significativas. ▪ Necesita Mejorar (4-0 puntos): La presentación es confusa, incompleta o insuficiente.
4. Calidad y gestión del Código (20 %)		
Calidad del Código	10 %	▪ Excepcional (10-9 puntos): El código es limpio, eficiente y sigue las mejores prácticas de programación. Uso adecuado de pandas y numpy. ▪ Bueno (8-7 puntos): El código es adecuado pero podría mejorarse en términos de eficiencia o claridad. ▪ Satisfactorio (6-5 puntos): El código funciona pero es ineficiente o difícil de leer. ▪ Necesita Mejorar (4-0 puntos): El código es ineficiente, difícil de seguir o no cumple con los requisitos.
Uso de Control de Versiones	10 %	▪ Excepcional (10-9 puntos): Uso excelente de Git con más de 20 commits significativos. El historial de commits es claro y muestra un desarrollo continuo. ▪ Bueno (8-7 puntos): Uso adecuado de Git, pero con algunos aspectos que podrían mejorarse en términos de commits o estructura. ▪ Satisfactorio (6-5 puntos): Uso básico de Git, con menos de 20 commits o commits no significativos. ▪ Necesita Mejorar (4-0 puntos): No se utilizó Git adecuadamente o se omitió su uso.