Universidad de La Habana
Facultad de Matemática y Computación

Crystal: Herramienta computacional para la resolución de modelos epidemiológicos definidos por ecuaciones diferenciales ordinarias.

Autor:
Daniela Rodríguez Cepero.

Tutores:

DraC. Aymée de los Ángeles Marrero Severo.
MSc. Wilfredo Morales Lezca.

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

2023

github.com/Daroce1012/Tesis

Esta tesis está dedicada con gran amor a toda mi familia, en especial a mis padres Yanelis Cepero Peña y Márbel Rodríguez Cárdenas, por su incondicional apoyo y comprensión durante todo este tiempo. Sin ustedes, este logro no habría sido posible.

Agradecimientos

Quisiera agradecer a todas aquellas personas que han sido fundamentales en el camino hacia la culminación de mi tesis. En primer lugar me gustaría darle las gracias a mis padres porque, creyeron en mí, siempre estuvieron dándome fuerza en los momentos más difíciles y porque el orgullo que sienten por mí, fue lo que me hizo llegar hasta el final. Gracias mamá por cada una de tus comidas que me levantaron el ánimo y me transmitieron amor, gracias papá por tus exigencias para con mis estudios. Gracias a ambos por luchar por mí y darme todo lo que estuviera a sus alcances para que no me faltara nada. Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta.

A mi hermano, que junto a nuestros padres han sido mi apoyo y fuente de inspiración para este trabajo.

A mis abuelos: mami y papi, a mami por sus consejos, por siempre consentirme, por ayudarme siempre que pudo y por todo el amor que me da; a papi por ser mi recadero, por su cariño y su apoyo. A ambos por la excelente madre que me dieron.

A mi abuela Cary por siempre orar por mí, a mi abuelo Luis por cuidarme desde el cielo. A ambos por darme un padre maravilloso.

A mi tutora Aymée de los Ángeles Marrero Severo, quien siempre estuvo para mí, a pesar de todas las dificultades por las que pasó. Por responder mis mensajes a cualquier hora y rectificar mis errores. A mi tutor Wilfredo Morales Lezca por sus consejos y ayuda a lo largo de la carrera. Quiero darle las gracias a ambos, por su orientación, sabiduría y paciencia. Agradezco profundamente su tiempo, su dedicación y su experiencia en mi formación académica. Sin ustedes no hubiera podido llegar hasta aquí.

Además, quiero agradecer a Daniela quien a pesar de no tener tiempo sacó espacio para mí, y a la profesora Joanna Campbell que generosamente compartió su tiempo y conocimiento conmigo. Sin su colaboración, este trabajo no hubiera sido posible.

A los valiosos profesores de la facultad que impartieron con amor sus conocimientos, y con exigencia, me enseñaron a ser un mejor profesional.

A mis suegros, Marisela García y Alberto Machín por hacer más cómodo el recorrido y brindarme su apoyo.

A mi novio Bryan Machín, por su amor, paciencia y comprensión. Ha sido mi roca y apoyo en momentos de incertidumbre y estrés. Le agradezco por su amor incondicional, su confianza en mí y por haberme ayudado a mantener la motivación y el enfoque durante todo el proceso.

Gracias a Dios por permitirme culminar con éxito mi tan anhelada carrera, darme buena salud y fortaleza en todo momento, y por sobre todo, escucharme.

Por último, quiero expresar mi gratitud a todas las personas que, de una forma u otra, han contribuido a mi formación académica y personal. A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

Opinión del tutor

El trabajo de diploma **Crystal: Herramienta computacional para la resolución de modelos epidemiológicos definidos por ecuaciones diferenciales ordinarias**, presentado por la estudiante **Daniela Rodríguez Cepero**, para optar por el título de licenciada en Ciencia de la Computación, se corresponde con una necesidad de un grupo de profesores de la facultad de Matemática y Computación que trabajamos la modelación, solución y análisis de problemas aplicados a las ciencias de la vida.

Después de años de trabajo y de algunos intentos para contar con una herramienta computacional que automatizara el tratamiento inherente a modelos matemáticos en epidemiología, el curso pasado nos acercamos a una herramienta que cuenta con una interfaz visual amigable con el usuario, pero que sólo considera la resolución y estimación de parámetros para un pequeño grupo de modelos prototipos.

Por esa razón, nos propusimos como objetivo fundamental de esta tesis el diseño e implementación de una herramienta computacional, para la resolución, análisis y graficación de cualquier modelo epidemiológico poblacional definido por ecuaciones diferenciales ordinarias.

Cuando me entrevisté con Daniela para conversar sobre el tema de diploma, se motivó de inmediato y me propuso el diseño de un lenguaje de dominio específico (DSL), que le permitiría al usuario introducir una amplia clase de modelos matemáticos poblacionales, para su resolución, la graficación de resultados y el manejo de los mismos.

Por supuesto que acepté, aunque no tenía idea de lo que me estaba proponiendo. La colaboración con el cotutor nos iluminó un poco el camino.

A pesar del poco tiempo con que ha contado la diplomante en este atípico último semestre de su carrera, considero que como versión inicial y luego de las sugerencias y señalamientos de la licenciada Daniela González, y sobre todo de la profesora Joanna Campbell, consideramos que la interface está bien concebida, cumple los objetivos

que nos propusimos y brinda un manual de usuario para quienes tienen conocimientos básicos de este tipo de problemas.

El código se encuentra estructurado y organizado según lo que dictan las buenas prácticas y los estándares actuales en cuanto a diseño y organización. No obstante, el diseño es perfectible, algo comprensible para una primera versión y esperamos que continuemos trabajando juntas para incorporar el problema de estimación de parámetros, esencial para nuestras investigaciones.

En más de una ocasión, al emitir mi opinión como tutora he plasmado que los autores han trabajado de forma independiente y en este caso, no me queda más remedio que repetirme, pues no puedo terminar este informe sin expresar que debido a mis muy limitados conocimientos de programación, toda la propuesta es de total creación de la diplomante que tuvo todo el peso de la formulación de la interface.

Desde el punto de vista matemático y teórico, realizó un estudio que le permitió comprender los principales modelos con que deseábamos trabajar y las técnicas de solución de los sistemas de ecuaciones diferenciales ordinarias.

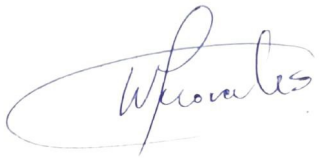
Considero que el trabajo presentado cumple con creces los requerimientos para ser defendido como tesis de licenciatura, con una evaluación excelente.

Hay mucho por perfeccionar para que esta herramienta cumpla con mayores expectativas y por eso invito a Daniela a que mantengamos el mismo intercambio ameno y fructífero para dar continuidad al mismo.

Le deseo grandes éxitos en su vida profesional y la felicito de corazón.

La Habana, Diciembre 12 de 2023

DraC. Aymée de los Ángeles Marrero Severo



MSc. Wilfredo Morales Lezca

Resumen

Durante años, los profesores del Grupo de Modelación Biomatemática de la facultad de Matemática y Computación de la Universidad de La Habana, han obtenido importantes resultados en el trabajo con modelos epidemiológicos poblacionales y la estimación y ajuste de sus parámetros, aplicados a diferentes enfermedades infecto-contagiosas y no trasmisibles. Esfuerzos y resultados que se han visto particularmente visibilizados durante la pandemia de la COVID-19. Urge por tanto, contar con una herramienta computacional amigable que permita resolver muchos de los problemas inherentes a la modelación y la solución, que aporte a una mejor comprensión y control de factores de riesgos.

El objetivo fundamental de este trabajo es el diseño e implementación de una herramienta computacional, para la resolución, análisis y graficación de los modelos epidemiológicos, tipo poblacionales. Para ello se diseñó e implementará un software que permite resolver modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias aplicados a las ciencias de la vida, específicamente a la epidemiología. Para el desarrollo del mismo se diseñó un lenguaje de dominio específico, conocido por las siglas en inglés (DSL) que permite al usuario introducir una amplia clase de modelos matemáticos poblacionales. Dichos modelos se solucionan por métodos numéricos para sistemas de ecuaciones diferenciales. Además, esta herramienta permitire la graficación y análisis de los resultados.

Palabras Claves: Modelos matemáticos poblacionales, Ecuaciones Diferenciales Ordinarias (EDO), Lenguaje de dominio específico, Epidemiología.

Abstract

For years, the professors of the Biomathematical Modeling Group of the Faculty of Mathematics and Computer Science of the University of Havana have obtained important results in the work with population epidemiological models and the estimation and adjustment of their parameters, applied to different infectious-contagious diseases and also for non-transmissible illness. These efforts and results have been particularly visible during the COVID-19 pandemic. Therefore, it is urgent to have a friendly computational tool that allows solving many of the problems inherent to modelinga nd solution, which contributes to a better understanding and control of risk factors.

The fundamental objective of this work is the design and implementation of a computational tool for the resolution, analysis and graphing of epidemiological models, population type. For it Software was designed and implemented that allows solving population mathematical models defined by systems of ordinary differential equations applied to the life sciences, specifically to epidemiology. For its development, a domain-specific language was designed, known by its acronym in English (DSL), which allows the user to introduce a wide class of population mathematical models. These models are solved by numerical methods for systems of differential equations. In addition, this tool will allow the graphing and analysis of the results.

Keywords: Population Mathematical Models, Ordinary Differential Equations (ODE), Domain Specific Language, Epidemiology.

Índice general

Índice de figuras

Capítulo 1

Introducción

Los modelos matemáticos nos permiten representar de manera precisa y cuantitativa el comportamiento de sistemas complejos en la realidad. Esto es especialmente relevante en el campo de la epidemiología poblacional, donde es crucial comprender y predecir la propagación de enfermedades infecciosas para poder implementar estrategias efectivas de control y prevención [6].

Un modelo matemático es siempre una idealización de la realidad puesto que la relación entre los modelos matemáticos y la realidad radica en su capacidad para capturar las interacciones y dinámicas que ocurren en la población, teniendo en cuenta factores como la transmisión de la enfermedad, la inmunidad adquirida, las tasas de infección y recuperación, entre otros. Esto permite a los investigadores y expertos en salud pública tener una comprensión más profunda del impacto de las enfermedades infecciosas y evaluar diferentes escenarios y estrategias de intervención.

Los modelos matemáticos pueden ser expresados en forma de ecuaciones diferenciales, ecuaciones de reacción-difusión, modelos estocásticos, entre otros. Estos modelos son útiles para realizar simulaciones numéricas y analizar diferentes escenarios, lo que permite a los investigadores obtener información sobre el comportamiento del sistema en cuestión y tomar decisiones informadas [26], [38].

De todas las formas existentes, la modelación mediante ecuaciones diferenciales ordinarias desde el punto de vista matemático es una herramienta poderosa y de las mejores para simular la dinámica de enfermedades infecciosas en una población y es ampliamente utilizada en la investigación epidemiológica para comprender y predecir la propagación de enfermedades [3], [17].

Contar con una herramienta computacional que permita llevar la realidad a un ambiente que pueda ser analizado resulta una ventaja en la sociedad actual, debido al gran número de epidemias a las que se enfrenta el mundo constantemente. De ahí que contar con un software resulta una necesidad, con vistas a una posible solución o mejora de la situación existente.

Durante años, los profesores del Grupo de Modelación Biomatemática de la facultad de Matemática y Computación de la Universidad de La Habana, han obtenido importantes resultados en el trabajo con modelos epidemiológicos poblacionales y la estimación y ajuste de sus parámetros, aplicados a diferentes enfermedades infecto-contagiosas y no trasmisibles. Esfuerzos y resultados que se han visto particularmente visibilizados durante la pandemia de la COVID-19 [20], [30], [29], [21], [1].

Como se explicó anteriormente se han hecho varios intentos de implementación, para contar con un software o herramienta computacional que permita la resolución y tratamiento de esta clase de modelos, imprescindible para las aplicaciones. Hasta el presente se han dado pasos en este sentido pero aún muy limitados.

Sería interesante hacerse la siguiente Pregunta Científica:

Por qué es importante y necesario contar con un software que automatice la labor de los investigadores del Grupo de Biomatemática de la facultad en su trabajo con los modelos epidemiológicos? Existirá alguna herramienta para esto o podrá desarrollarse una con todos los requerimientos deseados?

Es conocido que la modelación matemática aplicada a la dinámica de transmisión de enfermedades exige el manejo de un volumen importante de resultados, con vistas a su análisis y utilización en la predicción y control de las mismas.

Por tanto, podemos plantearnos el siguiente Problema Científico:

El diseño e implementación de un software con interfaz amigable, que permita el tratamiento de los problemas inherentes a la modelación biomatemática, la solución de los modelos, la graficación de resultados, la estimación de los principales parámetros que caracterizan la dinámica de transmisión, el estudio de sensibilidad, el diseño de experimentos entre otros muchos aspectos.

Este trabajo marca el inicio de tan abarcadores propósitos y cumplirá los objetivos que se describen a continuación.

1.1. Objetivos

El objetivo principal de esta tesis es diseñar e implementar una herramienta computacional nombrada Crystal, que contenga una interfaz visual amigable con el usuario para la resolución, análisis y graficación de modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias aplicados a las ciencias de la vida, específicamente a la epidemiología.

Para la confección de la aplicación se proponen los siguientes objetivos parciales:

- Realizar una investigación para determinar si existe algún software que resuelva el problema planteado con el objetivo de realizar un análisis crítico y comparativo.
- Desarrollar un lenguaje de dominio específico, conocido por las siglas en inglés (DSL) que permita introducir una amplia clase de modelos matemáticos poblacionales.
- Implementar la funcionalidad de resolver numéricamente modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias.
- Permitir la graficación de los resultados obtenidos al resolver los modelos definidos por el usuario.
- Diseñar e implementar una aplicación web que contenga una interfaz de usuario agradable.
- Aplicar pruebas de funcionalidad para validar el correcto funcionamiento del software.

1.2. Estructura

La tesis está dividida en varios capítulos que abordan las diferentes fases del proceso de desarrollo del software, el diseño, la implementación, la ejecución y el análisis de los resultados.

En este **Capítulo 1** se ofrece la Introducción que familiariza al lector con la motivación e importancia de la problemática a resolver. Además se presentan los objetivos y la estructura del mismo.

En el **Capítulo 2** se describen los modelos epidemiológicos poblacionales definidos por sistemas de EDOs y se muestran algunos softwares precedentes.

En el **Capítulo 3** se explica la elección de algunas herramientas, software y lenguajes utilizados en la confección del software.

En el **Capítulo 4** se expone el diagrama de uso, la arquitectura y el patrón de diseño empleado en el software, así como algunos detalles de implementación y ejemplos de códigos importantes en la lógica del software.

En el **Capítulo 5** se muestran algunos resultados de la ejecución de la aplicación y un resumen del manual de usuario, para que el lector entienda como usarla.

Con el **Capítulo 6** se exponen las conclusiones, posibles líneas de investigaciones futuras que se pudieran desarrollar y las recomendaciones.

Capítulo 2

Estado del Arte

Los modelos epidemiológicos poblacionales definidos por Ecuaciones Diferenciales Ordinarias describen el comportamiento epidemiológico en la actualidad, de ahí la necesidad de conocer sobre ellos. En este capítulo se presentan los modelos epidemiológicos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias, esencialmente los llamados modelos clásicos. Además, se enumeran y explican algunos softwares precedentes analizados con objetivos y características afines.

2.1. Modelos Epidemiológicos Poblacionales definidos por Ecuaciones Diferenciales Ordinarias

La modelación mediante ecuaciones diferenciales ordinarias (EDO) es una técnica matemática comúnmente utilizada en epidemiología poblacional para simular la propagación de enfermedades infecciosas. Las EDO son ecuaciones que describen cómo una cantidad cambia en función del tiempo, y son especialmente útiles para modelar sistemas dinámicos como la dinámica de transmisión de enfermedades [20].

En el contexto de la epidemiología poblacional, las EDO se utilizan para representar cómo cambian las tasas de infección, recuperación y mortalidad a lo largo del tiempo, teniendo en cuenta la interacción entre individuos susceptibles, infectados y recuperados. Estas ecuaciones pueden ser derivadas a partir de modelos matemáticos que describen la dinámica de la enfermedad, que se resuelven numéricamente para predecir la evolución de la enfermedad en la población [21], [1].

La modelación mediante EDO permite a los epidemiólogos simular diferentes escenarios y evaluar el impacto de intervenciones como la vacunación, el tratamiento o el distanciamiento social en la propagación de la enfermedad. Además, estas ecuaciones pueden ser ajustadas utilizando datos reales para calibrar el modelo y hacer predicciones más precisas sobre la evolución de un brote epidémico [30].

En resumen, la modelación mediante ecuaciones diferenciales ordinarias es una herramienta poderosa para simular la dinámica de enfermedades infecciosas en una población, y es ampliamente utilizada en la investigación epidemiológica para comprender y predecir la propagación de enfermedades [29].

Los modelos matemáticos pueden clasificarse según sus características en diversos grupos, entre estos [41]:

- Empíricos o teóricos: los primeros se basan en relaciones estadísticamente significativas entre variables y son válidos solo para el contexto espacio-temporal en el que se calibraron; mientras que los segundos, en las leyes físicas y biológicas que rigen los procesos.
- Estáticos o dinámicos: se refieren a la forma en que se manipula el tiempo. Los modelos estáticos dan un resultado agregado para todo el período de tiempo considerado. Los modelos dinámicos devuelven las series temporales de las variables consideradas a lo largo del período de estudio.
- Deterministas o estocásticos: los primeros, entre los que se encuentran los modelos compartimentales, son aquellos que trabajan con condiciones y datos conocidos, y se pueden controlar los factores que intervienen en el estudio, por lo que no se contempla la existencia de incertidumbre o el azar; y en el segundo se incluyen generadores de procesos aleatorios y producen salidas diferentes para el mismo conjunto de datos de entrada.

2.2. Modelos compartimentales

Los modelos compartimentales constituyen una técnica utilizada para simplificar la modelización matemática de las enfermedades infecciosas. En estos modelos, la población se divide en compartimentos, asumiendo que cada individuo en un mismo compartimento tiene la misma relación con la enfermedad [31].

Dentro de los modelos compartimentales se encuentran los llamados modelos clásicos que dividen la población en subpoblaciones principales. Los más utilizados son aquellos que caracterizan a la población en Susceptibles (S), Infectados o Infecciosos

(I) y Recuperados o Removidos (R) y que se denotan por las siglas SI, SIS, SIR. Estos modelos ayudan a los epidemiólogos a comprender cómo se propaga una enfermedad en una población y a evaluar estrategias de control y prevención [19].

2.2.1. SI

El **modelo SI** es uno de los modelos compartimentales clásicos y se utiliza para describir la propagación de enfermedades infecciosas de las que no se recupera una población. En este modelo, hay dos subpoblaciones:

- **susceptibles**: Representa el número de personas susceptibles a la enfermedad. Estas personas aún no han sido infectadas pero pueden contraer la enfermedad si están expuestas al patógeno.
- **infectados**: Representa el número de personas infectadas. Una vez que alguien se infecta, permanece en este compartimento durante toda la duración de la enfermedad.

El modelo SI es adecuado para enfermedades sin recuperación o inmunidad, como algunas enfermedades de transmisión sexual, debido a que los individuos se mueven directamente de la clase de **susceptibles** a la de **infectados** y no tiene un compartimento de recuperados o inmunes lo que implica que las personas infectadas no se recuperan ni desarrollan inmunidad [31].

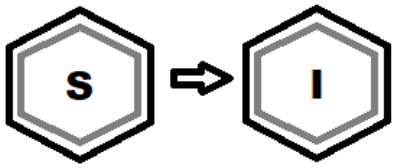


Figura 2.1: Diagrama de Flujo del modelo SI

Sistema de Ecuaciones Diferenciales

El modelo SI se describe mediante **ecuaciones diferenciales** y se utiliza para comprender cómo se propaga una enfermedad en una población. Aunque es simple, proporciona una base importante para modelos más complejos como el SIR y el SEIR.

$$\begin{cases} \dfrac{dS}{dt} = & -\beta S(t)I(t), \quad S(0) = S_0 > 0 \\ \dfrac{dI}{dt} = & \beta S(t)I(t), \quad I(0) = I_0 > 0 \end{cases}$$

- S, I : número de individuos susceptibles e infectados, respectivamente.
- β : tasa de contagio (probabilidad por individuo y unidad de tiempo de contraer la enfermedad).

2.2.2. SIS

En este modelo, hay dos compartimentos:

- **susceptibles**: Representa el número de personas susceptibles a la enfermedad. Estas personas aún no han sido infectadas pero pueden contraer la enfermedad si están expuestas al patógeno.
- **infectados**: Representa el número de personas infectadas. Una vez que alguien se infecta, permanece en este compartimento durante toda la duración de la enfermedad.

En el modelo SIS los individuos se trasladan de la clase de **susceptibles** a la de **infectados** y, al recuperarse, vuelven a ser **susceptibles**, por tanto no hay inmunidad permanente. Este es útil para enfermedades con recuperación pero sin inmunidad duradera, como la gripe.

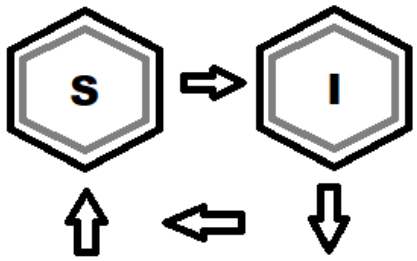


Figura 2.2: Diagrama de Flujo del modelo SIS

Sistema de Ecuaciones Diferenciales

$$\begin{cases} \dfrac{dS}{dt} = & -\beta S(t)I(t) + \gamma I(t), \quad S(0) = S_0 > 0 \\ \dfrac{dI}{dt} = & \beta S(t)I(t) - \gamma I(t), \quad I(0) = I_0 > 0 \end{cases}$$

- S, I: número de individuos susceptibles e infectados, respectivamente.
- β: tasa de contagio (probabilidad por individuo y unidad de tiempo de contraer la enfermedad).
- γ: tasa de recuperación (intensidad de recuperación de los infectados).

2.2.3. SIR

El modelo SIR es un modelo compartimental utilizado en epidemiología para describir la propagación de enfermedades infecciosas en una población. A diferencia del modelo SI, el SIR incluye una subpoblación adicional:

- **Susceptibles**: Representa el número de personas susceptibles a la enfermedad. Estas personas aún no han sido infectadas pero pueden contraer la enfermedad si están expuestas al patógeno.

- **Infectados**: Representa el número de personas infectadas. Una vez que alguien se infecta, permanece en este compartimento durante toda la duración de la enfermedad.
- **Recuperados**: Representa el número de personas que se han recuperado o son inmunes después de la infección. En el modelo SIR, las personas se mueven desde el compartimento I al R una vez que se recuperan.

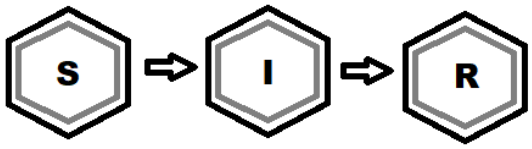


Figura 2.3: Diagrama de Flujo del modelo SIR

Sistema de Ecuaciones Diferenciales

El modelo SIR se describe mediante ecuaciones diferenciales y se utiliza para comprender cómo se propaga una enfermedad en una población. Aunque más complejo que el modelo SI, el SIR proporciona una representación más realista de la dinámica de la infección.

$$\begin{cases} \frac{dS}{dt} = & -\beta S(t)I(t), \quad S(0) = S_0 = N - I_0 > 0 \\ \frac{dI}{dt} = & \beta S(t)I(t) - \gamma I(t), \quad I(0) = I_0 > 0 \\ \frac{dR}{dt} = & \gamma I(t), \quad R(0) = 0 \end{cases}$$

- S, I, R : número de individuos susceptibles, infectados y recuperados, respectivamente.
- β : tasa de contagio (probabilidad por individuo y unidad de tiempo de contraer la enfermedad).
- γ : tasa de recuperación (intensidad de recuperación de los infectados).
- N : número de individuos totales, tiene dimensión número de individuos.

2.3. Softwares Precedentes

Como parte de la investigación se realizó un estudio sobre las aplicaciones existentes que tratan la resolución de modelos epidemiológicos tanto dentro como fuera del país. Los resultados fueron escasos, se encontró un número pequeño de softwares relacionados con esta problemática, la mayoría no son para modelos epidemiológicos y los que sí lo son, no resuelven numéricamente cualquier modelo, sino que están orientados a algunos modelos en específico. Otros solo permiten el análisis de algunos modelos mediante simulaciones. Para la resolución y graficación de modelos epidemiológicos poblacionales solo se tiene conocimiento de Eagle que es un software precedente a este y que se explicará más adelante. Algunas de las principales aplicaciones encontradas se muestran a continuación.

2.3.1. Simulación y Análisis de Modelos Poblacionales Matriciales

Es un software desarrollado por la Universidad Nacional de Córdoba que permite el estudio de modelos poblacionales matriciales permitiendo su construcción, análisis e interpretación. El software cuenta con un lenguaje de programación (llamado PML) desarrollado para permitir la simulación y el análisis de los modelos matriciales. Mediante la simulación y rutinas numéricas analizan diferentes casos poblacionales. Esta implementación permite interpretar los resultados obtenidos por medio de gráficos de líneas, barras e imágenes de sensibilidad [13].

2.3.2. Maplesoft

Maple es un software poderoso para desarrollar y analizar modelos SIR. Permite explorar, visualizar y resolver prácticamente cualquier problema matemático. Pero tanto sus cursos para aprender a utilizarlo como la licencia del software son de pago. Por lo que se tiene poca información sobre este.

2.3.3. PET

Es un software desarrollado por la Facultad de Matemática y Computación para automatizar la experimentación numérica en la resolución de problemas de estimación de parámetros en modelos epidemiológicos. En PET se tiene un grupo de carpetas donde se deben copiar todos los datos experimentales (mediciones), los archivos que definen el modelo matemático (ecuaciones diferenciales), así como los algoritmos con los que se quiere trabajar y sus archivos descriptores. No tiene una interfaz visual

donde se le pueda introducir directamente el modelo epidemiológico y que éste sea resuelto.

2.3.4. Eagle

Es un software desarrollado por la Lic. Airelys Collazo Perez en su tesis de licenciatura, tutoreada por la Dra. Aymée Marrero Severo, perteneciente a la Facultad de Matemática y Computación. Presenta una interfaz visual para la resolución de modelos epidemiológicos prototipos (SI, SIS, SIR), además de permitir la estimación de parámetros y su graficación. Para la estimación de parámetros cuenta con métodos clásicos y metaheurísticas. Presenta una interfaz cómoda para el usuario, pero el software no está diseñado para asumir dinámicamente modelos insertados por el usuario, solo admite los modelos clásicos predefinidos en el software: SI, SIS, SIR.

La consulta realizada y el análisis anterior permiten concluir que el software Simulación y Análisis de Modelos Poblacionales Matriciales se limita solo a modelos matriciales y no se tiene acceso al código fuente. PET no cuenta con una interfaz visual para introducir un modelo epidemiológico, ni con algoritmos de solución; sino que necesita de conocimiento experto matemático, por parte del usuario, para introducir el algoritmo a utilizar. Maplesoft es un software de pago al que no se tiene acceso. Por último está Eagle que cuenta con una interfaz visual, pero no está diseñado para asumir dinámicamente modelos insertados por el usuario, solo admite los modelos clásicos predefinidos: SI, SIS, SIR. Por tanto es clara la necesidad de construir una aplicación web desde cero que cumpla con los objetivos planteados en el capítulo anterior.

Capítulo 3

Propuesta

En este capítulo se enfatizará en el marco teórico-computacional de la solución propuesta, abordando las funciones que pueden ejecutar los usuarios y la estructura del software. Se explicará la arquitectura, patrón de diseño y patrón de visualización del mismo, así como las tecnologías utilizadas.

3.1. Diagrama de casos de uso

A continuación se presentan las diferentes acciones que puede ejecutar un usuario mediante un diagrama de casos de uso [1].

En la aplicación solo existe un rol de usuario **Usuario Anónimo**, este es un usuario que puede acceder a la aplicación sin introducir ningún dato previo al sistema, teniendo disponibles todas las funcionalidades de la misma.

Todo usuario de Crystal puede:

- Crear un modelo matemático.
- Resolver numéricamente un modelo epidemiológico previamente definido, teniendo la posibilidad, además, de escoger diferentes métodos de solución.
- Obtener los resultados en los diferentes instantes de tiempo.
- Graficar los resultados de un modelo epidemiológico previamente definido y visualizar el mismo.

[1] Un diagrama de casos de uso es una representación gráfica de los requisitos funcionales de un sistema, los actores que se comunican con el sistema y las relaciones entre ellos. Este diagrama es parte del Lenguaje Unificado de Modelado, conocido por sus siglas en inglés (UML). Se utiliza para describir las interacciones entre los usuarios y un sistema.

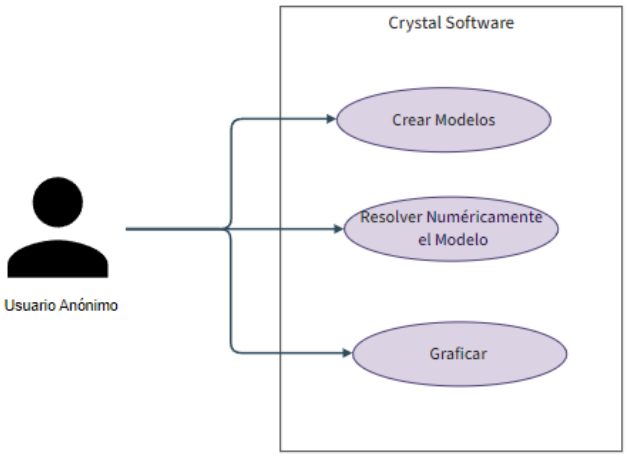


Figura 3.1: Diagrama de casos de uso

3.2. Arquitectura de software

Se denomina Arquitectura de software a la estructura y organización de un sistema de software, incluyendo sus componentes, relaciones y principios de diseño. Define cómo las diferentes partes del software interactúan entre sí y con el hardware; y cómo se distribuyen las responsabilidades y tareas dentro del sistema. Es fundamental para garantizar la calidad, mantenibilidad y escalabilidad de un sistema de software.

La arquitectura en capas es un modelo de diseño de software, cuya base es la separación de las diferentes funcionalidades del sistema en capas o niveles. Cada capa se encarga de un conjunto de tareas específicas y se comunica con las capas adyacentes mediante interfaces bien definidas. Los elementos clave de la arquitectura en capas son [9], [32]:

- Capas: Cada capa es un conjunto de componentes o módulos de software que realizan una función específica dentro del sistema. Las capas se organizan jerárquicamente, desde la más baja (encargada de los detalles de implementación) hasta la superior (que proporciona la interfaz de usuario y la lógica de presentación).
- Interfaces: Las interfaces definen cómo las capas interactúan entre sí. Proporcionan una forma estandarizada de comunicación. Por ejemplo, una interfaz puede definir operaciones como lectura/escritura de datos o ejecución de funciones.

En una arquitectura de capas, todas las capas se colocan de forma horizontal, donde una capa solo puede depender de otra que esté por debajo de ella, nunca por

encima. En una forma estricta de la arquitectura, solo se puede acceder a la capa que está exactamente debajo. Si se usa un acercamiento más flexible, una capa puede acceder a todas las capas por debajo de ella [7].

La arquitectura en capas ofrece varias ventajas importantes en el diseño de sistemas de software [23], [36]:

1. Modularidad: La separación en capas permite que cada componente se enfoque en una tarea específica. Esto facilita la reutilización de código, ya que los módulos pueden ser intercambiados o mejorados sin afectar otras partes del sistema.

2. Facilidad de mantenimiento: Al tener capas claramente definidas, es más sencillo realizar actualizaciones o correcciones en el sistema. Si un error se encuentra en una capa, no es necesario modificar todo el sistema, sino solo la capa afectada.

3. Escalabilidad: La arquitectura en capas permite escalar partes específicas del sistema según las necesidades. Por ejemplo, si se requiere mayor capacidad de procesamiento en la capa de acceso a datos, se puede escalar esa capa sin afectar otras áreas.

4. Abstracción: Cada capa oculta los detalles de implementación a las capas superiores. Esto proporciona una abstracción que facilita la comprensión y colaboración entre equipos de desarrollo.

5. Flexibilidad: Las capas pueden ser reorganizadas o reemplazadas sin afectar la funcionalidad general del sistema. Esto permite adaptarse a cambios en los requisitos o tecnologías emergentes.

La arquitectura en capas permite una mejor organización, modularidad, mantenimiento y evolución a largo plazo del sistema.

3.2.1. Arquitectura de Crystal

El software Crystal implementa una arquitectura de 3 capas.

En la figura 3.2 se puede observar la separación de las capas y la dependencia entre ellas.

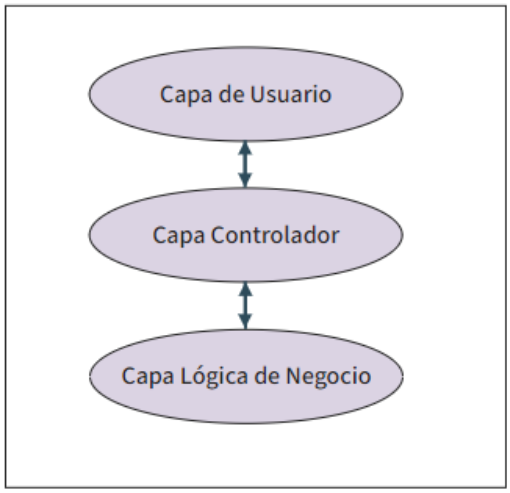


Figura 3.2: Arquitectura de Crystal

La Capa Lógica de Negocio es la capa base. No tiene dependencia externa con otras capas. Incluye el análisis sintáctico y lexicográfico del DSL para poder interpretar lo introducido por el usuario en la interfaz visual y ejecutar a partir de ahí acciones de negocio, como la resolución de un modelo. Para la implementación de esta capa se utilizó el lenguaje de Python.

La Capa Controlador depende de la capa Lógica de Negocio. Es visto como un controlador pues recibe una solicitud y da una respuesta, no es más que la interfaz de comunicación donde se conectan el Usuario (la persona) con el negocio (la capa de lógica de negocio). En esta capa se utilizó el framework FastApi.

La Capa Usuario depende de las capas anteriores. Contiene la implementación de la interfaz visual y por tanto posee los componentes y servicios encargados de enviar peticiones al backend y recibir la respuesta a dichas peticiones. En esta capa se utilizó el lenguaje de programación TypeScript con el framework Angular.

3.3. Frameworks, Lenguajes de Programación y Herramientas

Para el desarrollo del software se realizó una investigación con el objetivo de determinar las tecnologías adecuadas para su confección. Se estudiaron los lenguajes de programación, herramientas y frameworks más utilizados en la actualidad, así como las ventajas que brindan.

La aplicación web está dividida en dos grandes bloques, el backend y el frontend. Para el deasarrollo del backend se seleccionó como lenguaje de programación a Python porque tiene una funcionalidad que ningún otro tiene y que es de suma importancia para la problemática a tratar. Esta funcionalidad y demás ventajas de Python se explican en los epígrafes 3.3.1 y 3.3.5. Se utilizó el lenguaje de TypeScript como lenguaje para la confección de la interfaz visual.

Los frameworks fueron otras herramientas que se hizo necesaria en el desarrollo del proyecto. Para la interfaz visual se analizaron algunos de los más utilizados entre estos: Vue, React y Angular, finalmente fue seleccionado Angular, este tema se abordará en el epígrafe 3.4. En el backend se utilizó otro framework como conector con el frontend de la aplicación, en este caso fueron analizados algunos framework de python como Django, FastApi y Flask, pero se escogió a FastApi, esta elección se argumentará en el epígrafe 3.3.3.

3.3.1. Python

Python es un lenguaje de propósito general que se puede utilizar en diversos campos. Es ideal para el desarrollo de software y admite programación estructurada, funcional y orientada a objetos. Se caracteriza por tener una sintaxis clara y concisa, lo que lo convierte en un lenguaje relativamente fácil de utilizar [27].

Creado a finales de los años ochenta [40] por Guido van Rossum en Stichting Mathematisch Centrum (CWI), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba [35]. Su nombre proviene de la afición de su creador por los humoristas británicos Monty Python [34].

Actualmente es uno de los lenguajes de programación más populares en el mundo, es así que ocupa el puesto número uno en el índice de popularidad de TIOBE.

Nov 2023	Nov 2022	Change	Programming Language	Ratings
1	1		Python	14.16%
2	2		C	11.77%
3	4	▲	C++	10.36%
4	3	▼	Java	8.35%
5	5		C#	7.65%
6	7	▲	JavaScript	3.21%
7	10	▲	PHP	2.30%
8	6	▼	Visual Basic	2.10%
9	9		SQL	1.88%
10	8	▼	Assembly language	1.35%

Figura 3.3: Índice de Popularidad de TIOBE

3.3.2. TypeScript

Anders Hejlsberg, diseñador de *C#* y creador de Delphi y Turbo Pascal, ha trabajado en el desarrollo de TypeScript que es un lenguaje de programación fuertemente tipado [2] que se basa en JavaScript[3], lo que le brinda mejores herramientas a cualquier escala. Es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno). Además al extender la sintaxis de JavaScript, cualquier código existente debería funcionar sin problemas. En resumen, TypeScript es una herramienta poderosa para desarrolladores que buscan mejorar la calidad y mantenibilidad de sus proyectos JavaScript mediante el uso de tipos estáticos y otras características avanzadas [8], [10].

[2]Lenguaje que no permite violaciones de los tipos de datos, por ejemplo no se puede sumar directamente una variable numérica b = 3 con una variable que contenga una cadena a = "hola".

[3]Hace referencia al nombre de un Lenguaje de Programación [22], [15], [11], [2], [43].

3.3.3. Python como plataforma de desarrollo

Python es un lenguaje de programación versátil y ampliamente utilizado. Aunque no existe un **mejor** lenguaje universal y se pudieron haber utilizado otros lenguajaes de programación como C#, Java o C++. Hubo ventajas que resultaron determinantes en su elección como lenguaje de desarrollo. Algunas de ellas son [28], [25]:

1. Bibliotecas: Python tiene una amplia colección de bibliotecas integradas. Su biblioteca estándar es extensa, y los usuarios pueden encontrar bibliotecas adicionales en el índice de paquetes de Python (PyPI). Cuenta con **bibliotecas especializadas en la resolución numérica de sistemas de ecuaciones diferenciales**. Además tiene una forma fácil y sencilla de **graficar** los resultados obtenidos mediante la utilización de otras bibliotecas que posee. Gracias a esto se extiende a casi cualquier rama de las ciencias.

2. Es una plataforma en la cual se tiene experiencia trabajando en el grupo de investigación, por lo tanto, es un ambiente cómodo para el desarrollo.

Además de estas dos ventajas que fueron las razones de su elección, Python tiene otras ventajas que son importantes y también se tuvieron en cuenta como:

- Lenguaje de alto nivel: Es más fácil de usar que los lenguajes de bajo nivel. Su sintaxis es similar al inglés, lo que facilita la lectura, escritura y aprendizaje.

- Orden y Legibilidad: Se destaca por su sintaxis clara y legible, lo que facilita la escritura y comprensión del código. Sus módulos están bien organizados.

- Multiplataforma: Python se ejecuta en diversos sistemas operativos, lo que lo hace ideal para aplicaciones multiplataforma.

- Comunidad activa y Frameworks: Cuenta con una gran comunidad activa y solidaria de desarrolladores. Además posee una gran variedad frameworks.

Dentro de los principales frameworks de Python se encuentran:

Django

Es un framework web de alto nivel que permite el desarrollo rápido de aplicaciones web seguras y mantenibles. Django se encarga de gran parte de las complicaciones del desarrollo web [14]. Es completo puesto que provee casi todo lo que los desarrolladores quisieran; lo que significa que todo funciona a la perfección, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación. Es versátil, puede ser (y ha sido) usado para construir casi cualquier tipo de aplicación web, desde sistemas

administradores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente y puede entregar contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc). Ayuda a los desarrolladores a evitar varios errores comunes de seguridad al proporcionar un framework que ha sido diseñado para "hacer lo correcto" y proteger automáticamente el sitio web. Por ejemplo, Django proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando errores como almacenar información de sesión en cookies (en lugar de eso, las cookies solo contienen una clave y los datos se almacenan en la base de datos) o almacenar directamente las contraseñas en un hash seguro. Posee una Arquitectura Modelo-Vista-Controlador (MVC), lo que facilita la organización y separación de la lógica de la aplicación en componentes claros. Tiene soporte para múltiples bases de datos lo que permite elegir entre diferentes motores de base de datos, según sean las necesidades.

Flask

Es un microframework de Python que permite crear aplicaciones web con pocas líneas de código. No utiliza bibliotecas externas para su desarrollo, lo que lo hace ligero y flexible. Aunque es "micro", no se refiere a la capacidad de crear solo aplicaciones pequeñas, sino a su enfoque minimalista y extensible. Incluye un sistema integrado de pruebas unitarias que acelera la depuración y fomenta un desarrollo robusto. Permite agregar funcionalidad adicional a la aplicación mediante extensiones de terceros. Flask utiliza el motor de plantillas Jinja2 para renderizar[4] HTML dinámico. Esto permite crear contenido personalizado en las vistas.

FastApi

Es un framework web moderno, rápido (de alto rendimiento) para crear API con Python 3.8+, basado en sugerencias de tipo estándar de Python. Está basado y es totalmente compatible con los estándares abiertos para APIs: OpenAPI (conocido previamente como Swagger) y JSON Schema. Crea código listo para producción, con documentación automática interactiva. Posee un alto rendimiento, a la par con NodeJS y Go (gracias a Starlette y Pydantic). Es uno de los frameworks de Python más rápidos puesto que incrementa la velocidad de desarrollo entre 200 % y 300 % [18].

[4]El renderizado es el proceso de finalización de una imagen digital o un modelo 3D mediante software informático

Backend: Python y FastApi

Luego del estudio anterior se seleccionó a FastApi como framework para el desarrollo del backend de la aplicación, a pesar de que se hubiera podido utilizar cualquiera de ellos, debido a que:

- Se tiene una experiencia satisfactoria en el desarrollo de aplicaciones web con este framework.
- La aplicación no requiere de un servicio de gestión de bases de datos y no cuenta con una arquitectura de capas compleja. Por tanto no es necesario un framework complejo como Django.
- Incrementa la velocidad de desarrollo.
- Es fácil de usar y aprender.
- Gran soporte en los editores con auto completado en todas partes. Gasta menos tiempo debugging [5].
- Minimiza la duplicación de código debido a las múltiples funcionalidades con cada declaración de parámetros.

3.3.4. TypeScript como lenguaje para el desarrollo de la interfaz visual

TypeScript es un lenguaje de programación desarrollado y mantenido por Microsoft. A continuación, se presentan algunas de sus características clave:

1. La sintaxis es relativamente sencilla de aprender comparada con otros lenguajes de programación.
2. Hay incontables recursos para aprender. Sitios como StackOverflow y Github experimentan una creciente cantidad de proyectos que usan este lenguaje.
3. Cuenta con múltiples opciones de efectos visuales, lo que lo hace útil para desarrollar páginas dinámicas y aplicaciones web.
4. Es compatible con los dispositivos más modernos, incluyendo IPhone y móviles Android.

[5] Debugging es el proceso de depurar o eliminar errores de un programa de computadora.

5. Tipado Estático: TypeScript introduce tipos estáticos, lo que permite atrapar errores en tiempo de compilación. Cada variable tiene su propio tipo de dato, lo que mejora la calidad del código y reduce errores en tiempo de ejecución.

6. Compatibilidad con JavaScript: TypeScript es una extensión de JavaScript, por lo que cualquier código JavaScript existente puede ser gradualmente migrado a TypeScript sin problemas. Esto facilita la adopción y la integración en proyectos existentes.

7. Soporte para ES6+: TypeScript soporta características modernas de ECMAScript, como módulos, clases, arrow functions, async/await, etc. Esto permite escribir código más actualizado y legible.

8. Herramientas de Desarrollo: El compilador de TypeScript proporciona información detallada sobre errores y advertencias, lo que facilita la depuración y mejora la productividad. Además, ofrece funciones como autocompletado, inferencia de tipos y refactorización.

9. Ficheros de Definición: TypeScript permite definir tipos para librerías JavaScript existentes mediante ficheros de definición. Esto facilita la integración con bibliotecas de terceros y mejora la interoperabilidad.

10. Escalabilidad: TypeScript es ideal para proyectos grandes y complejos. Su sistema de tipos y características avanzadas ayudan a mantener la escalabilidad y la estructura del código.

En resumen, TypeScript combina las ventajas de un lenguaje tipado con la flexibilidad y compatibilidad de JavaScript, lo que lo convierte en una excelente elección para proyectos de desarrollo web y aplicaciones en general [39], [24]. Por esas razones fue seleccionado como lenguaje para el desarrollo del frontend.

Los frameworks son herramientas inteligentes que posibilitan un desarrollo rápido de las aplicaciones sin comprometer la usabilidad de estas. Otra de las ventajas de TypeScript es la variedad de frameworks que posee, estos se han popularizado en los últimos años debido a la experiencia web dinámica e interactiva que ofrecen. Tres de los frameworks más populares y utilizados por la comunidad de programadores son:

React

React es una biblioteca JavaScript de código abierto creada por el equipo de la compañía Facebook (Meta), junto a una gran comunidad de desarrolladores independientes. Desde su lanzamiento en 2013, se ha convertido en una de las tecnologías

para frontend más usadas, ya que permite construir interfaces de usuario dinámicas y escalables [12].

La función principal de React es desarrollar páginas web de manera gratuita y sencilla gracias a sus componentes reutilizables. Estos permiten usar un mismo elemento en varias partes del sitio o incluso en otros sitios sin necesidad de volver a escribir todo el código [12].

Angular

Angular fue creado por Google en 2009. Es un framework completo que ofrece una amplia gama de características para el desarrollo de aplicaciones web. Proporciona una estructura sólida y herramientas para manejar rutas, formularios, autenticación y más. Su nombre proviene de los paréntesis angulares (<>) utilizados en HTML [4], [5].

La vinculación bidireccional es la característica innovadora más intuitiva que permite la actualización en tiempo real de un sitio web desde diferentes dispositivos. Además, cuenta con el mayor apoyo comunitario activo en todo el mundo [4].

Vue

Vue (también conocido como Vue o VueJS) es un framework de JavaScript progresivo y fácil de aprender. Su enfoque en la creación de componentes y su integración fluida con otras bibliotecas lo convierten en una excelente opción para proyectos pequeños y grandes [42].

Vue es una excelente opción para desarrolladores frontend, ya que combina simplicidad con potencia, y es capaz de manejar desde pequeños componentes hasta sofisticadas Single-Page Applications[6] cuando se utiliza en combinación con herramientas modernas y librerías de apoyo [42] .

Frontend: TypeScript y Angular

Para el desarrollo del frontend se eligió al framework **Angular** debido a que [4]:

1. La arquitectura de Angular enlaza TypeScript y HTML, el código de ambos ya está sincronizado. Por lo tanto, el framework ahorra mucho tiempo a los desarrolladores.

[6]Una single page application (SPA) o aplicación de una sola página, es un tipo de aplicación web que carga una única página inicial desde el servidor y luego actualiza dinámicamente su contenido, mediante las interacciones del usuario, sin necesidad de cargar páginas adicionales del servidor [16]

2. El framework admite pruebas unitarias y de integración.

3. Su base de usuarios sigue creciendo y tiene una gran cantidad de documentación en profundidad que se actualiza constantemente.

4. Puede ejecutarse en la mayoría de los navegadores web. No solo en computadoras de escritorio, sino también en dispositivos móviles.

5. Permite a los usuarios manipular elementos de una página web sin necesidad de secuencias de comandos complejas. Además, mejora la visualización de páginas con gran cantidad de datos.

6. Angular es adecuado tanto para aplicaciones móviles como de escritorio.

7. Genera código altamente optimizado, lo que garantiza una carga rápida de aplicaciones. También ofrece una división de código automática para cargar solo lo necesario.

3.3.5. Bibliotecas

En esta sección se dará una breve explicación de las principales bibliotecas de Python que fueron usadas, con el objetivo de permitir acciones de la lógica del negocio como: resolver numéricamente sistemas de EDOs y la graficación de los mismos.

- **scipy.integrate.solve_ivp** Es un módulo que forma parte de la biblioteca scipy y resuelve un problema de valor inicial para un sistema de EDOs. Integra numéricamente un sistema de ecuaciones diferenciales ordinarias dado un valor inicial. Posee diferentes métodos de solución como son RK45[7], RK23[8], DOP853[9], Radau[10], BDF[11] y LSODA[12] [33].

- **matplotlib** Es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python de una manera fácil. Es muy buena para graficar soluciones. Produce figuras de calidad en una variedad de formatos

[7]Método explícito de Runge-Kutta de orden 5(4). El error se controla asumiendo la precisión del método de cuarto orden, pero se toman medidas utilizando la fórmula precisa de quinto orden.

[8]Método explícito de Runge-Kutta de orden 3(2). El error se controla asumiendo la precisión del método de segundo orden, pero se toman medidas utilizando la fórmula precisa de tercer orden.

[9]Método explícito de Runge-Kutta de orden 8.

[10]Método implícito de Runge-Kutta de la familia de orden Radau IIA 5. El error se controla con una fórmula incrustada precisa de tercer orden.

[11]Método implícito de orden variable de varios pasos (1 a 5) basado en una fórmula de diferenciación hacia atrás para la aproximación derivada.

[12]Método Adams/BDF con detección y cambio automáticos de rigidez.

impresos y entornos interactivos en todas las plataformas. Matplotlib se puede usar en scripts15 y shells16 de Python, servidores de aplicaciones web y varias herramientas de interfaz gráfica de usuario [37].

A pesar de que Python brinda la posibilidad de la resolución numérica de sistemas de EDOs, es necesario la confección de un puente entre los usuarios y el lenguaje de programación, con el objetivo de facilitarles su uso y comprensión. En el capítulo siguiente se abordará la confección del DSL[13] del software, que servirá de puente.

[13]Un lenguaje específico de dominio (DSL) es un lenguaje de programación que se especializa en resolver problemas en un dominio particular. A diferencia de los lenguajes de programación generales, que son ampliamente aplicables en muchos dominios, los DSLs están diseñados para resolver problemas específicos en un dominio particular.

Capítulo 4

Detalles de Implementación

Para permitir la modelación y graficación de EDOs de una forma más simple y amigable para los usuarios, se decidió confeccionar un DSL. Este capítulo tratará sobre el desarrollo del mismo. Se explicará la gramática del lenguaje y su árbol de sintaxis abstracta (AST). Además se dejarán en claro algunos detalles importantes a tener en cuenta ante la modificación de su gramática, ya sea para ampliar sus funcionalidades o restringirlas.

4.1. El Lenguaje de Crystal

Para el desarrollo del lenguaje se implementó un DSL dinámico cuya sintaxis es parecida a la de Python. El lenguaje permite crear modelos epidemiológicos, resolverlos y graficar su resultado; además de la definición de variables y procedimientos como operaciones aritméticas.

4.1.1. Definiciones Básicas

$< commentario >$: #...

$< num >$: $[0-9]$

$< string >$: "..."

$< boolean >$: $True|Fasle$

$< id >$: $[A-Za-z]$

$< punto >$: .

$< o\_paréntisis >$: (

$< c\_paréntisis >$:)

$< o\_key >$: {

$< c\_key >$: }

$< comma >$: ,

$< semi >$: ;

Operadores

$< asignación >$: =

$< suma >$: +

$< resta >$: −

$< multiplicación >$: ∗

$< división >$: /

Operadores de Comparación

$< leq >$: <=

$< geq >$: >=

$< equal >$: ==

$< not >$: !=

$< less >$: [<]

$< greater >$: [>]

Operadores Lógicos

$< and >$: *and*

$< or >$: *or*

4.1.2. Programa y Comandos

Para la ejecución de un programa se lee línea a línea de código y cada línea está compuesta por una serie de comandos o declaraciones que indican pasos o acciones a ejecutar.

$< program > \rightarrow < stat - list >.$

$< stat - list > \rightarrow < stat > ; \mid < stat > < stat - list > ;$

Algunas de estas acciones pueden ser la declaración de una variable o un modelo, la llamada a ejecución de una función determinada o una simple intrucción de mostrar un resultado.

$< stat > \rightarrow < def - var > \mid < def - model > \mid < func - call > \mid < print - stat > \mid < def - return > \mid < solve - model > \mid < def - plot >$

Ejemplo:
$a = 5;$
print $a;$

Este programa tiene dos declaraciones una que se encarga de definir una variable con valor 5 y otra que se encarga de mostrar en la salida el valor de la variable antes definida.

4.1.3. Declaraciones

Cada declaración tiene su propia estructura que la define. Para mostrar un resultado en específico se utiliza el comando print.

$< print - stat > \rightarrow$ **print** $< expr >$

Ejemplo:
print 3;

Esta línea se encarga de mostrar en la salida el número 3 en el momento exacto que se procesa la línea.

Para la declaración de los parámetros y variables del sistema se utiliza:

$< def - var > \rightarrow < id > = < expr >$

Ejemplo:
$beta = 0{,}42;$
$betan = 0 - beta;$

Esta declaración se encarga de definir la variables beta y betan con un valor de 0.42 y -0.42 respetivamente.

Para la definición de los modelos se utilizan las declaraciones $< def - model >$ y $< def - return >$. Esta última se utiliza con el objetivo de obtener los valores de las variables que caracterizan el sistema de ecuaciones diferenciales, una vez ejecutado el procedimiento que las define.

$< def - model > \rightarrow$ **model** $< id >$ ($< arg - list >$) { $< stat - list >$ }

$< def - return > \rightarrow$ **return** $< expr - list >$

Ejemplo:
$model\ name\_model(s,i)\{$
$ds = betan * s * i;$
$di = beta * s * i;$
$return\ ds, di;\}$

El ejemplo anterior muestra la definición del modelo SI en el lenguaje de Crystal. Si queremos resolver numéricamente un modelo utilizamos este comando:

$< solve - model > \rightarrow < id >$.**solve** ($< expr - list >$)

Ejemplo:
$name\_model.solve("RK45", to, tf, so, io);$

Esta línea se encarga de resolver numéricamente el modelo llamado *name_model* definido anteriormente. Es importante resaltar que solo se pueden usar variables que estén previamente definidas, es decir, to, tf, so y si, deben estar definidas con anterioridad para poder ser usadas, en este ejemplo se omite ese paso. También es necesario aclarar que no es obligatorio poner variables, podrían ponerse los valores directamente, pero si es importante no violar el orden de los argumentos necesarios en esta declaración. El primer argumento sería el tipo de método que se desea utilizar para la resolución numérica, luego le sigue el tiempo inicial, seguido por el tiempo final y a continuación, los valores iniciales de las variables, en el mismo orden en que se definen en el modelo que se desea resolver.

Ejemplo:
$name\_model.solve("RK45", 0, 100, 10, 5);$

Si queremos graficar el modelo utilizamos el comando:
$< def - plot > \rightarrow$ **plot** ($< expr - list >$)

Ejemplo:
$result = name\_model.solve("RK45", to, tf, so, io);$
$plot("name\_model", result, to, tf, "S", "I");$

El comando plot se encarga de graficar y guardar en formato de imagen, el modelo. El primer parámetro representa el nombre con el que se desea guardar la imagen del modelo, el que le sigue, es el valor de la resolución numérica del mismo, los dos siguientes son el período de tiempo, es decir, tiempo inicial y final, respectivamente, y

para finalizar en el mismo orden que se definen las variables del modelo, los símbolos con los que se desea representar a cada una de estas.

Las siguientes declaraciones se utilizan para representar una serie de parámetros o argumentos que requiere el modelo.

$< arg-list > \rightarrow < id > \mid < id > \ , < arg-list >$

$< expr-list > \rightarrow < expr > \mid < expr > \ , < expr-list >$

4.1.4. Expresiones

$< op > \rightarrow + \mid - \mid <= \mid >= \mid = \mid < \mid > \mid and \mid or$

$< expr > \rightarrow < solve-model > \mid < term > \mid < expr > < op > < term >$

4.1.5. Términos

$< term > \rightarrow < factor > \mid < term > + < factor > \mid < term > - < factor >$

4.1.6. Factores

$< factor > \rightarrow < atom > \mid (< expr >)$

4.1.7. Átomos

$< atom > \rightarrow < string > \mid < boolean > \mid < num > \mid < id >$

4.2. AST

Para construir el AST se utiliza el método **construction ast** que recibe un árbol de derivación, las operaciones que se realizaron (Shift o Reduce) y el conjunto de tokens[1]. Este método devuelve un árbol cuya raíz es el nodo Program. A partir de ahí se recorren todos los nodos chequeando la semántica de las expresiones, y luego los de tipo **DeclarationNode** se ejecutan y los **ExpressionNode** se evalúan logrando así llevar el código al lenguaje Python.

[1]Hace referencia a palabras especiales, ej: print, (, }.

La base de los nodos del AST está sustentada a partir de dos tipos de nodos: **DeclarationNode** y **ExpressionNode**. Como se puede apreciar los nodos que implican una declaración son los que ejecutan acciones dentro del lenguaje sobre las expresiones. Se define por cada sentencia deseada en el lenguaje, una clase que hereda un comportamiento específico. Entre ellas tenemos la declaración de variables y el llamado de funciones.

Como nodo raíz del árbol de sintaxis abstracta tenemos **ProgramNode** que se encarga de ejecutar el recorrido en el árbol chequeando la semántica de las sentencias y ejecutando las declaraciones.

Con respecto al chequeo semántico de las sentencias se constata la pertenencia de variables definidas o no en un ámbito determinado.

El lenguaje almacena las funciones y variables que se definen en un scope denominado **Context**, que cuenta con herramientas tales como su pertenencia o no, así como su definición. Dicho scope contiene la propiedad **children context** que facilita la definición de entes internos para determinadas funcionalidades. Resultando satisfactorio el chequeo semántico, el programa ejecuta la secuencia de definiciones.

Capítulo 5

Pruebas de funcionalidad y Resultados

En este capítulo se presentan un conjunto de pruebas de correctitud, para demostrar el cumplimiento de los objetivos planteados y el funcionamiento correcto del software. También se explicará cómo utilizar la herramienta y la sintaxis del lenguaje que usarán los usuarios en el mismo.

Las pruebas se realizaron en una laptop con las siguientes características:

- **Sistema operativo:** Windows 10 Home, 64 bits.
- **Procesador:** AMD A12-9720P RADEON R7, 12 COMPUTE CORES (4CPUs), 2.7GHz.
- **Memoria RAM:** 16 GB.
- **Disco Duro:** 1TB SSD.

5.1. Manual de Usuario

Al ejecutar la aplicación web, la primera vista que aparece es la página de inicio. La imagen a continuación muestra la visualización de la misma.

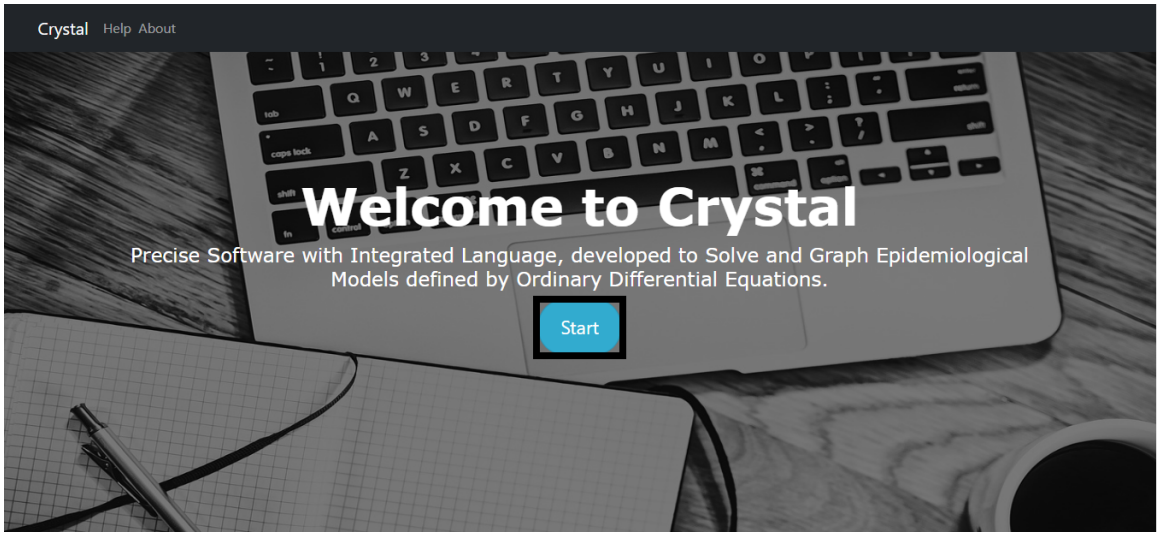


Figura 5.1: Página de Bienvenida.

Como se puede observar en la fig 5.1, la página contiene cuatro botones principales. Entre estos, el botón **Start**, que redirige al editor de código.

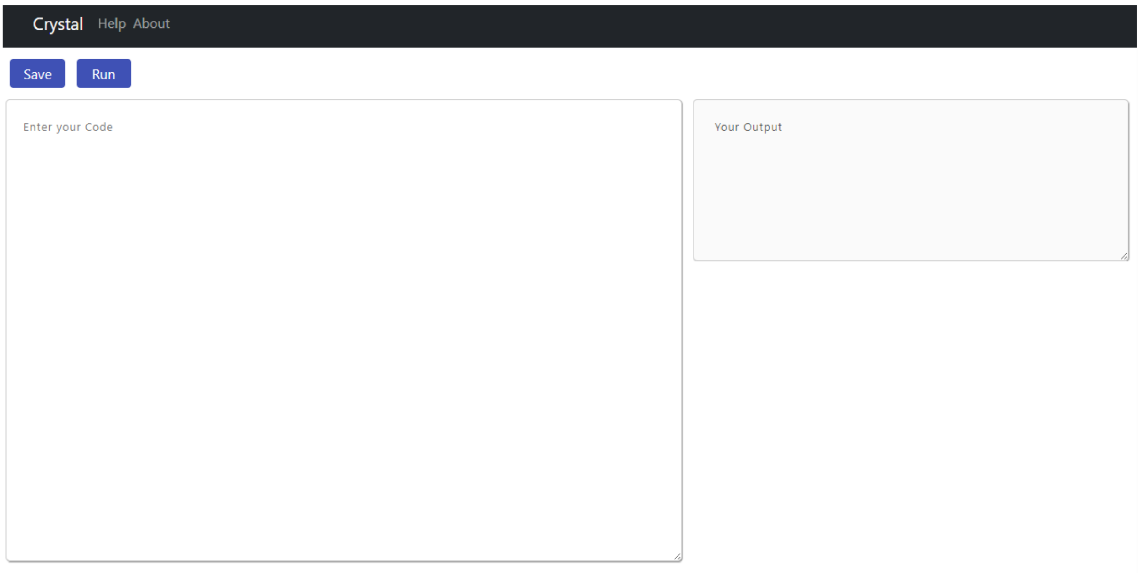


Figura 5.2: Página del Editor.

En la página editor de código es donde se introduce la sintaxis del lenguaje, que nos permite resolver modelos epidemiológicos y graficar los resultados. Esta tiene dos botones y tres componentes principales.

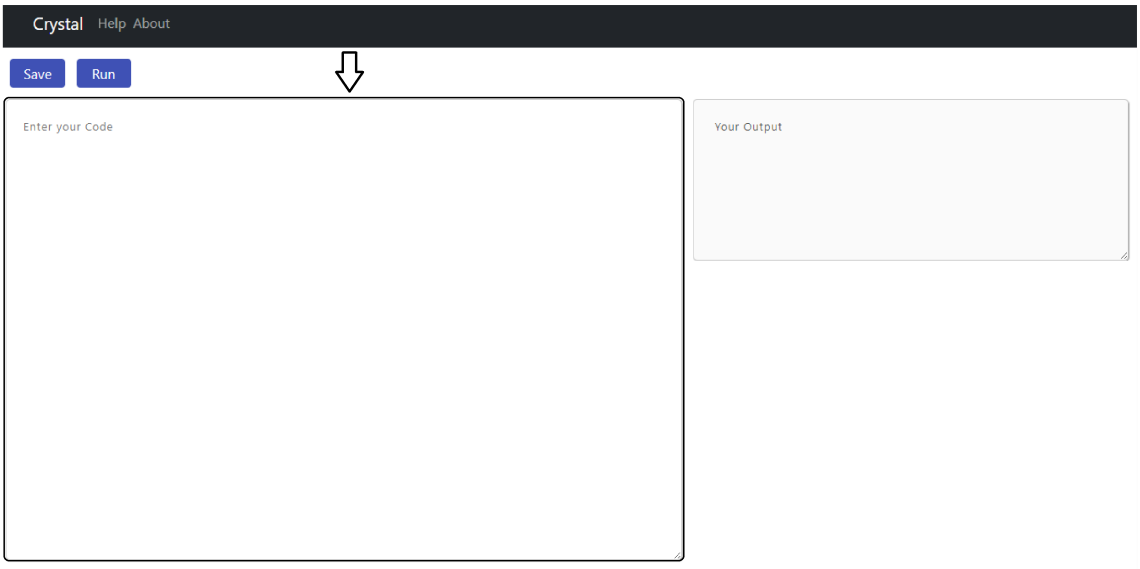


Figura 5.3: Input.

En la figura 5.3 aparece marcado uno de estos componente, denominado Input pues en él, es donde se introducen los comandos que permiten definir los modelos, así como su resolución y graficación. Como ejemplo tenemos la imagen siguiente.

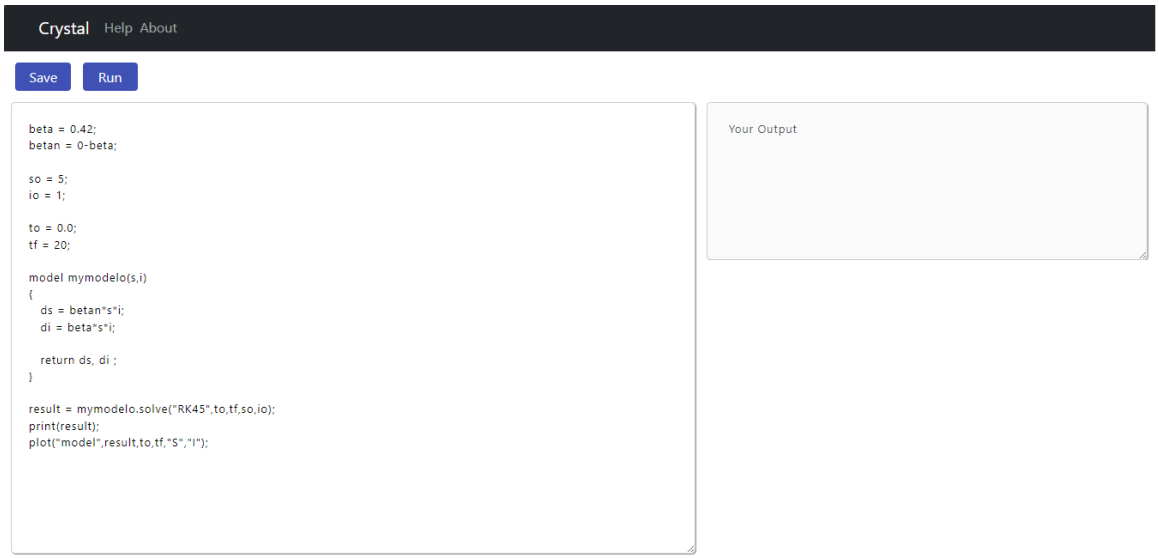


Figura 5.4: Ejemplo de Código.

Después de la introducción de los comandos que requiere la herramienta para resolver nuestro problema, se tienen dos opciones, una de estas es, guardar el código introducido para ser modificado con posterioridad. Esta función se puede realizar mediante el botón **save**.

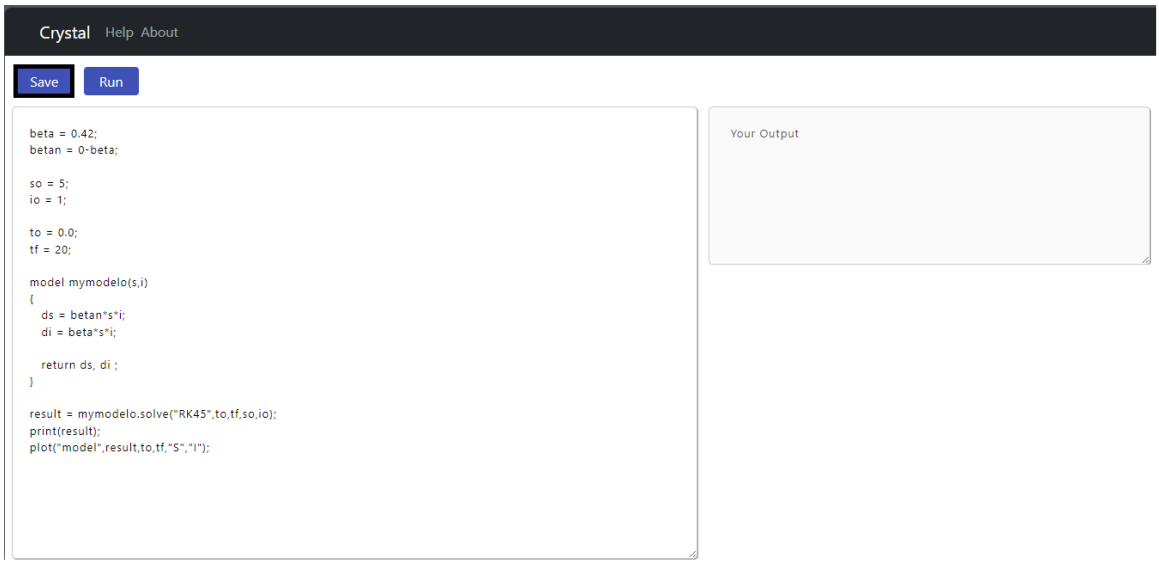


Figura 5.5: Botón Save.

La otra opción garantiza la ejecución del código introducido, con el objetivo de resolver o vizualizar el modelo. Esta función se puede ejecutar mediante el botón **run**, que se encuentra a la derecha del botón **save**.

Si algún comando de los introducidos, como **solve** devuelve un resultado, este se mostrará en el componente output, luego de presionar el botón **run** . Si en lugar de un valor la salida es una imagen como en el caso del comando **plot**, se mostrará debajo del componente output. A continuación se muestra un ejemplo.

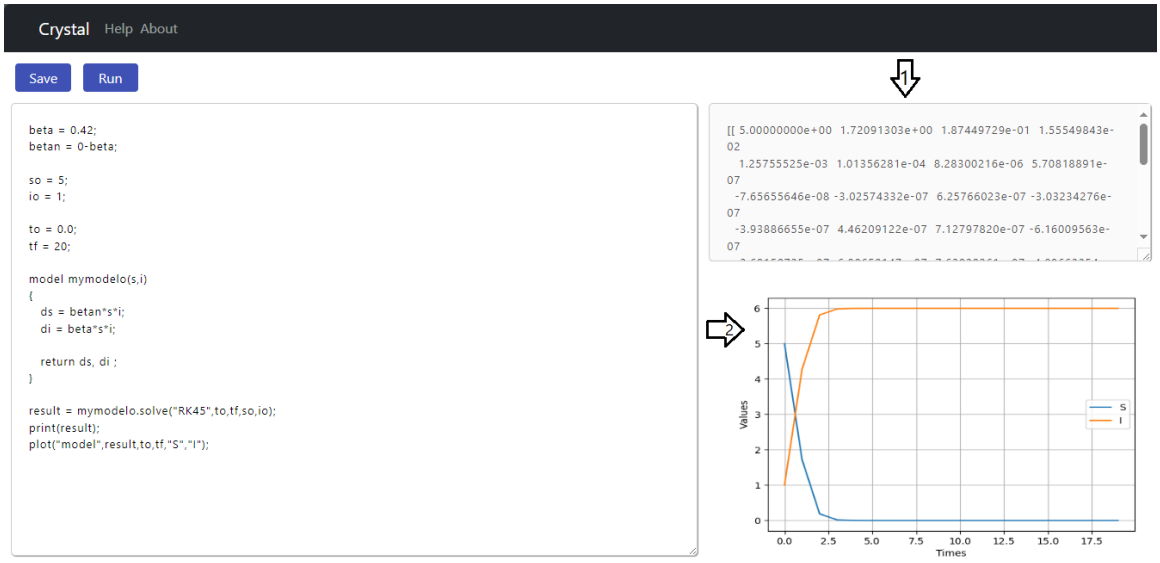


Figura 5.6: Output.

Para garantizar el correcto funcionamiento de la aplicación se debe seguir un orden en las secuencias de comandos que se introducen. Para definir un modelo los pasos correctos serían:

1. Definir los parámetros.
2. Definir los valores iniciales de las variables.
3. Definir el período de tiempo.
4. Definir el modelo.

Los pasos 2 y 3 no son obligatorios, esos valores se pueden introducir directamente en el comando de resolución. Para más información sobre este comando leer el epígrafe 4.1.3.

Para la definición del modelo se utiliza el comando **model** explicado en el epígrafe 4.1.3. Es necesario aclarar que el orden en que se declaran las variables en el argumento, debe ser el mismo en que se devuelven las derivadas de este, es decir, el mismo orden con que se escriben seguido del comando **return**.

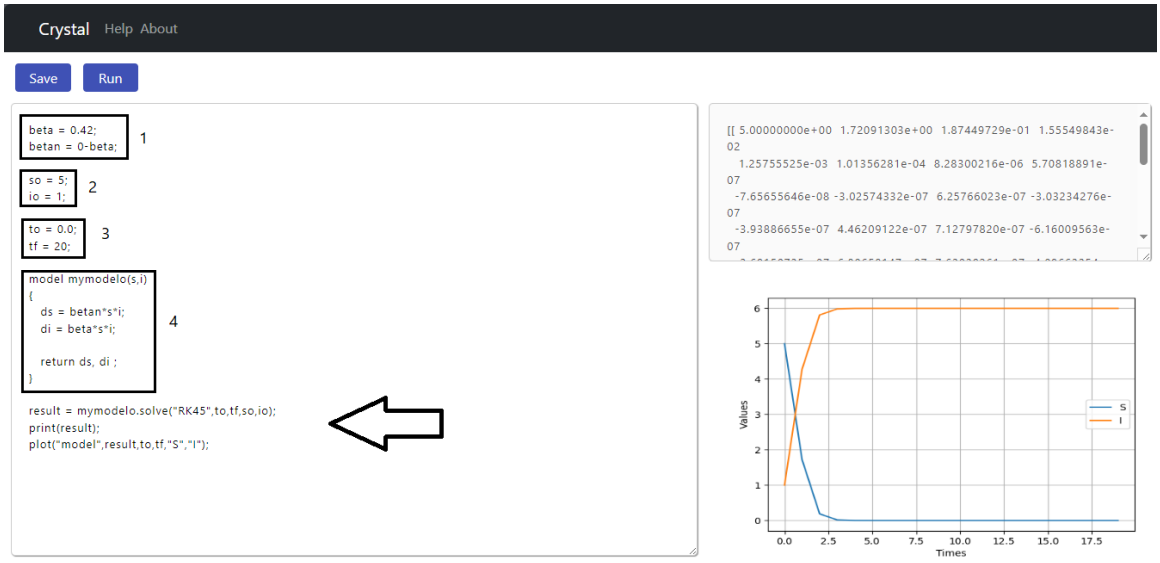


Figura 5.7: Pasos para modelar.

Por útimo, estarán los comandos que se encargan de resolver y graficar. Para resolver el modelo se utiliza el comando **solve**, si queremos que se muestre el resultado utilizamos el comando print y para obtener el valor de la solución utilizamos una variable, en el ejemplo anterior la variable que se utilizó para esto fue result. Cada uno de estos comandos se explican en el epígrafe 4.1.3.

Si desde cualquiera de las vistas de la aplicación web se desea volver a la página principal, solo es necesario presionar el botón **Crystal**.

Figura 5.8: Botón Crystal.

Y si se desea obtener información sobre la aplicación web, solo es necesario presionar el botón **About**.

Figura 5.9: Botón About.

Si se desconoce cómo usar el software se debe presionar el botón **Help** y se descargará un archivo .pdf que contiene el Manual de Usuario.

Figura 5.10: Botón Help.

5.2. Resultados

A modo de ilustración se muestran los resultados para dos modelos. El que se muestra en la figura 5.11 es un modelo clásico de tipo SIR considerando muerte natural y por enfermedad. En la 5.12 se presentan los resultados para un modelo que simula la dinámica de transmisión de la Covid-19 considerando asintomáticos y expuestos, de la autoría de investigadores del grupo de Biomatemática de la facultad.

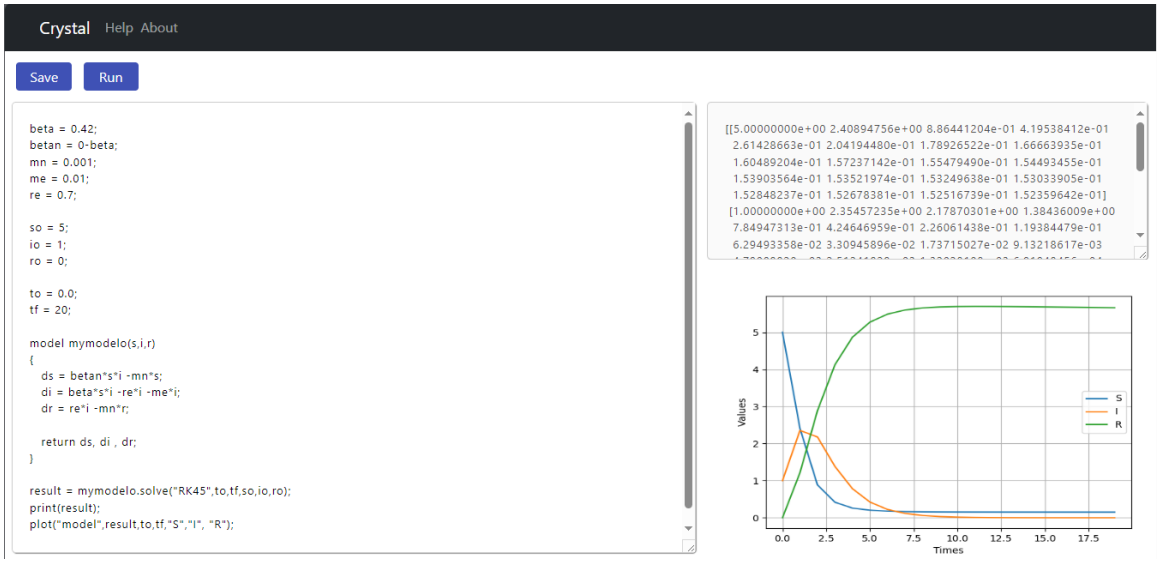


Figura 5.11: Modelo SIR

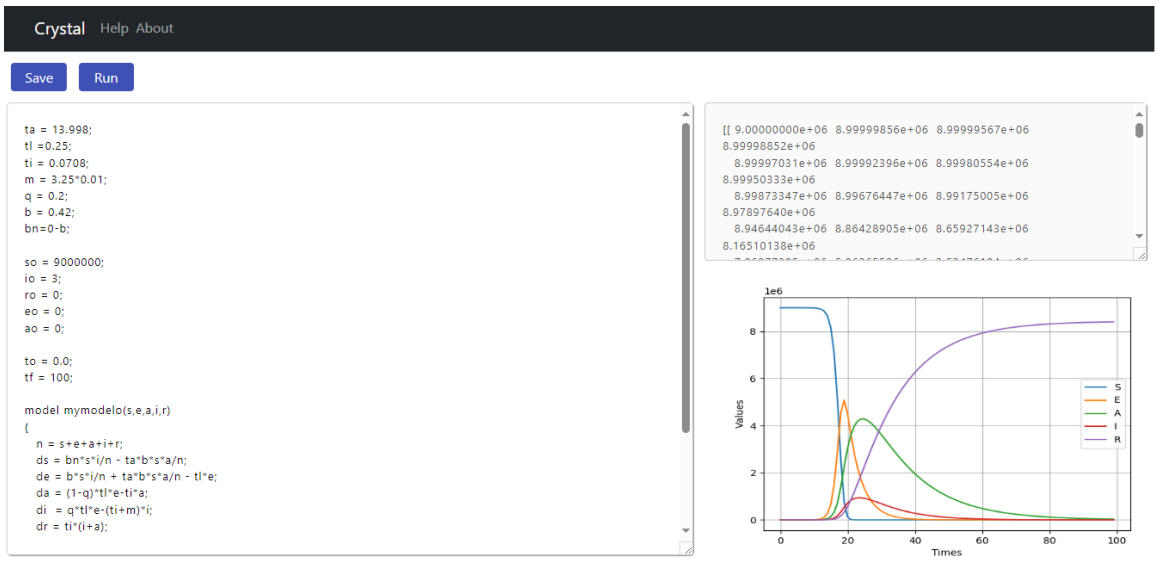


Figura 5.12: Modelo Covid-19

A la izquierda de ambas figuras se muestra el modelo, tal y como el usuario lo introduce al software, así como las condiciones iniciales para las variables y los valores seleccionados para los parámetros y el intervalo de tiempo. En las líneas finales aparece el método numérico que se seleccionó y el comando para la graficación de los resultados.

A la derecha, aparecen los resultados obtenidos y el gráfico del modelo.

En la leyenda de las gráficas se presentan las curvas de los Susceptibles (“S”), Infectados (“I”) y Recuperados (“R”), para el primer modelo, a los que se incorporan en el segundo modelo, los de Expuestos (“E”) y Asintomáticos (“A”). Dichos resultados permiten un análisis visual, en los que, como se espera, la subpoblación de Susceptibles tiende a disminuir y la de Recuperados a crecer a medida que transita la epidemia, mientras los Infectados, Asintomáticos y Expuestos alcanzan su punto máximo (pico de la epidemia) a partir del cual decrecen, lo que significa control de la misma.

Conclusiones

Se logró diseñar e implementar una herramienta computacional denominada Crystal, que contiene una interfaz visual amigable con el usuario para la resolución, análisis y graficación de modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias aplicados a la epidemiología, objetivo fundamental de este trabajo.

Se cumplieron los objetivos parciales, teniendo en cuenta que:

- A partir de un estudio del estado del arte que permitió determinar que no se cuenta con un software con las características deseadas.
- Se desarrolló un lenguaje de dominio específico (DSL), que permite introducir una amplia clase de modelos matemáticos poblacionales, implementando la funcionalidad de resolver numéricamente modelos matemáticos poblacionales definidos por sistemas de ecuaciones diferenciales ordinarias.

Además, se garantiza la graficación de los resultados obtenidos al resolver los modelos definidos por el usuario. Las pruebas de funcionalidad permitieron validar el correcto funcionamiento de la aplicación web.

Este primer resultado, permite su extensión para que funcione como un repositorio de modelos con sus potencialidades de solución, manejo de datos y graficación donde se almacenarán importantes resultados del trabajo del Grupo de Biomatemática.

El propio desarrollo de las herramientas y las investigaciones realizadas potencian líneas para continuar el trabajo. Todos los resultados se muestran de manera amigable haciendo uso de tablas y gráficas. La aplicación está apta para ser utilizada por cualquier investigador, sin necesidad de que tenga conocimientos de programación.

Recomendaciones

Durante el desarrollo del software y las investigaciones llevadas a cabo se evidenciaron líneas de investigaciones futuras que se pudieran desarrollar, entre las más importantes están:

- Mejorar la corrección de errores antes de la ejecución del software.
- Agregar otros métodos de solución numérica, ampliando así las vías de solución.
- Añadir algoritmos que permitan resolver el problema de la estimación de parámetros.
- Integrar una base de datos que permita guardar los resultados y la reutilización de los mismos.
- Incorporar seguridad a la aplicación, implementando un módulo de autenticación.
- Mejorar el diseño gráfico de la interfaz visual, teniendo en cuenta que es la primera versión.

Bibliografía

[1] Isidro Alfredo Abelló Ugalde, Raúl Guinovart Díaz y Wilfredo Morales Lezca. «El modelo SIR básico y políticas antiepidémicas de salud pública para la COVID-19 en Cuba». En: *Revista Cubana de Salud Pública* 46.suppl 1 (2020), e2597 (vid. págs. 2, 5).

[2] Khan Academy. *HTML/JS: hacer páginas web interactivas*. [Online; accessed 25-November-2023]. URL: `https://es.khanacademy.org/computing/computer-programming/html-css-js` (vid. pág. 18).

[3] Shair Ahmad y Antonio Ambrosetti. *A textbook on ordinary differential equations*. Vol. 88. Springer, 2015 (vid. pág. 1).

[4] Citio Web Oficial de Angular. *Angular*. [Online; accessed 26-November-2023]. URL: `https://angular.io/` (vid. pág. 23).

[5] Smart UI for Angular. *The Best Angular UI Components*. [Online; accessed 26-November-2023]. URL: `https://www.syncfusion.com/angular/components` (vid. pág. 23).

[6] Nicolas Bacaër. *Mathématiques et épidémies*. Cassini, 2021 (vid. pág. 1).

[7] Len Bass, Paul Clements y Rick Kazman. *Software Architecture in Practice*. 3rd. Addison-Wesley Professional, 2012. ISBN: 0321815734 (vid. pág. 15).

[8] KeepCoding Bootcamps. *¿Qué es TypeScript?* [Online; accessed 25-November-2023]. URL: `https://keepcoding.io/blog/typescript/` (vid. pág. 18).

[9] Tortuga Code. *Arquitectura en capas: qué es y cuándo usarla*. [Online; accessed 29-November-2023]. URL: `https://www.tortugacode.com/arquitectura-en-capas-que-es-y-cuando-usarla/` (vid. pág. 14).

[10] DEV Community. *¿Qué es TypeScript? Tipos, Clases e Interfaces*. [Online; accessed 25-November-2023]. URL: `https://dev.to/edisonsanchez/que-es-typescript-1jmb` (vid. pág. 18).

[11] Hotel Marketing Consulting. *¿Qué es JavaScript y cómo funciona?* [Online; accessed 25-November-2023]. URL: `https://soyhorizonte.com/blog/que-es-javascript-y-como-funciona/` (vid. pág. 18).

[12] Maria Coppola. *¿Qué es React y para qué sirve?* [Online; accessed 26-November-2023]. URL: https://blog.hubspot.es/website/que-es-react (vid. pág. 23).

[13] RJ Corin. «Simulación y Análisis de Modelos Poblacionales Matriciales». En: () (vid. pág. 11).

[14] Citio Web Oficial de Django. *Django*. [Online; accessed 24-November-2023]. URL: https://www.djangoproject.com/ (vid. pág. 19).

[15] Tutoriales Dongee. *¿Cómo se relaciona HTML, CSS y Javascript?* [Online; accessed 25-November-2023]. URL: https://www.dongee.com/tutoriales/como-se-relaciona-html-css-y-javascript/ (vid. pág. 18).

[16] Martín Durán. *Qué es una single page application, cómo funciona y ejemplo.* [Online; accessed 24-November-2023]. 2023. URL: https://blog.hubspot.es/website/que-es-single-page-application (vid. pág. 23).

[17] Charles Henry Edwards y David E Penney. *Differential equations and boundary value problems: computing and modeling*. Pearson Educación, 2000 (vid. pág. 1).

[18] Citio Web Oficial de FastApi. *FastApi*. [Online; accessed 24-November-2023]. URL: https://fastapi.tiangolo.com/es/ (vid. pág. 20).

[19] Laura García Rovira et al. «Modelos matemáticos compartimentales en epidemiología». En: (2017) (vid. pág. 7).

[20] R Guinovart-Diaz et al. «Un modelo matemático explica la necesidad de la protección para vencer a la COVID-19». En: *INFODIR (published online)* (2020) (vid. págs. 2, 5).

[21] D Guinovart-Sanjuán et al. «Multi-population analysis of the Cuban SARS-CoV-2 epidemic transmission before and during the vaccination process». En: *Physics of Fluids* 33.10 (2021) (vid. págs. 2, 5).

[22] Blog de HubSpot. *Qué es JavaScript, para qué sirve y cómo funciona*. [Online; accessed 25-November-2023]. URL: https://blog.hubspot.es/website/que-es-javascript (vid. pág. 18).

[23] Blog de HubSpot. *Qué es la arquitectura en capas, ventajas y ejemplos*. [Online; accessed 29-November-2023]. URL: https://blog.hubspot.es/website/que-es-arquitectura-en-capas (vid. pág. 15).

[24] Arquitectura Java. *10 Características que me gustan de TypeScript*. [Online; accessed 25-November-2023]. URL: https://www.arquitecturajava.com/10-cosas-que-me-gustan-de-typescript/ (vid. pág. 22).

[25] Redacción KeepCoding. *Ventajas y Desventajas de Python.* [Online; accessed 24-November-2023]. 2022. URL: https://keepcoding.io/blog/ventajas-y-desventajas-de-python/ (vid. pág. 19).

[26] Andrew King, John Billingham y Stephen Otto. «Differential equations: Linear, nonlinear, ordinary and partial». En: (2003) (vid. pág. 1).

[27] Jonathan Llamas. *Python.* [Online; accessed 24-November-2023]. 2023. URL: https://economipedia.com/definiciones/python.html (vid. pág. 17).

[28] Alejandra Milena Machado. *Por qué aprender Phyton y cuáles son sus ventajas.* [Online; accessed 24-November-2023]. 2020. URL: https://www.pragma.com.co/academia/lecciones/descubre-por-que-aprender-phyton-y-cuales-son-sus-ventajas (vid. pág. 19).

[29] D Menció y G Bayolo. «et A. Marrero. 2020..Evolución de la CoVid19 a partit de un modelo SAIRV con tasa de transmisión variable ante percepción de riesgo, cuarentena y hospitalización. Caso Cuba."» En: *Revista Ciencias Matemáticas* 34.1 (), págs. 67-72 (vid. págs. 2, 6).

[30] Daniel Menció Padrón, Gabriela Bayolo Soler y Aymée Marrero Severo. «Análisis de Modelo Matemático con percepción de riesgo para la CoVid19. Resultados para Cuba». En: *Revista Cubana de Informática Médica* 12.2 (2020) (vid. págs. 2, 6).

[31] Eustasio del Barrio y Cristina Rueda Pedro C. Álvarez-Esteban. *Modelos compartimentales: un estudio comparativo de la evolución de la pandemia de COVID19 en China y Castilla y León.* [Online; accessed 22-November-2023]. 2020. URL: http://www.imuva.uva.es/covid19/COVID19-AnalisisComparadoChinaCyL.html (vid. págs. 6, 7).

[32] Reactive Programming. *Arquitectura en Capas.* [Online; accessed 29-November-2023]. URL: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas (vid. pág. 14).

[33] Documentación de Python. *scipy.integrate.solve_ivp.* [Online; accessed 28-November-2023]. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve%5C_ivp.html (vid. pág. 24).

[34] Documentación de Python. *Whetting Your Appetite.* [Online; accessed 24-November-2023]. 2023. URL: https://docs.python.org/3/tutorial/appetite.html (vid. pág. 17).

[35] Documentación de Python. *Why was Python created in the first place?.* [Online; accessed 24-November-2023]. 2023. URL: https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place (vid. pág. 17).

[36] *Qué es la arquitectura en capas: descubre sus ventajas y ejemplos*. [Online; accessed 29-November-2023]. URL: https://latamtech-sac.com/que-es-la-arquitectura-en-capas-descubre-sus-ventajas-y-ejemplos/ (vid. pág. 15).

[37] The Matplotlib development team. *matplotlib*. [Online; accessed 28-November-2023]. 2023. URL: https://matplotlib.org (vid. pág. 25).

[38] Gerald Teschl. *Ordinary differential equations and dynamical systems*. Vol. 140. American Mathematical Soc., 2012 (vid. pág. 1).

[39] UNIR. *¿Qué es TypeScript? Usos y claves para aprenderlo*. [Online; accessed 25-November-2023]. URL: https://www.unir.net/ingenieria/revista/que-es-typescript/ (vid. pág. 22).

[40] Bill Venners. *The Making of Python A Conversation with Guido van Rossum, Part I*. [Online; accessed 24-November-2023]. 2003. URL: https://www.artima.com/articles/the-making-of-pythonl (vid. pág. 17).

[41] María Vidal Ledo et al. «Modelos matemáticos para el control epidemiológico». En: *Educación Médica Superior* 34.2 (2020) (vid. pág. 6).

[42] Citio Web Oficial de VueJs. *VueJs*. [Online; accessed 26-November-2023]. URL: https://es.vuejs.org/v2/guide/ (vid. pág. 23).

[43] Aprende desarrollo web.MDN. *Construye tu propia función*. [Online; accessed 25-November-2023]. URL: https://developer.mozilla.org/es/docs/Learn/JavaScript/Building_blocks/Build_your_own_function (vid. pág. 18).