UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO DE INGENIERÍA DE LA SALUD

Modelo celular de mutabilidad y evolución de Organismos
Cellular model of organism mutability and evolution

Realizado por
Viktor Yosava Zinkovskyi

Tutorizado por
Silvia Mercado Sáenz* y
Laura Panizo Jaime+

Departamento
* Fisiología Humana, Histología Humana, Anatomía Patológica y Educación Física y Deportiva
+ Lenguajes y Ciencias de la Computación

MÁLAGA, Septiembre 2024

Resumen

Este Trabajo de Fin de Grado presenta el desarrollo e implementación de **Speciate**, un modelo probabilístico diseñado para simular la evolución de organismos unicelulares a organismos más complejos a través de mutaciones aleatorias. A la hora de representar la reproducción de organismos biológicos, los modelos matemáticos y teóricos a menudo deben enfrentarse a datos que aumentan exponencialmente y altos costos computacionales. El fin principal de **Speciate** es abordar la complejidad de este tipo de simulaciones equilibrando la eficiencia computacional con la complejidad de los organismos digitales.

En este proyecto se han unificado teorías evolutivas, incluyendo la teoría de la selección natural de Darwin, la hipótesis de Oparin sobre el origen de la vida y la hipótesis del Lucacene, que han dictado las características de los organismos de la simulación, que imitan en la medida de lo posible a los organismos más primitivos que han existido, y la manera en la que estos podían mutar.

Desde el punto de vista de la ingeniería del software, la arquitectura de **Speciate** sigue un patrón Modelo-Vista-Controlador, que ofrece una interfaz gráfica de usuario para configurar simulaciones y visualizar resultados. El usuario puede configurar los parámetros iniciales de las simulaciones, como el número de especies, el código genético de estas, las tasas de mutación y factores ambientales como la adquisición de energía y las frecuencias de extinción, que determinarán la dirección que tomen las simulaciones.

El proyecto concluye analizando los resultados obtenidos por medio de un caso de estudio en el que se contrastan las características de los organismos como la influencia de la mutabilidad, la capacidad de obtener energía y la movilidad sobre el éxito poblacional. Esta herramienta educativa pretende apoyar a profesores y alumnos en la comprensión de procesos evolutivos complejos.

Keywords: Evolución, mutación, simulación, organismo unicelular, modelo probabilístico.

Abstract

This Bachelor's Thesis presents the development and implementation of **Speciate**, a probabilistic model designed to simulate the evolution of unicellular organisms into more complex organisms through random mutations. When representing the reproduction of biological organisms, mathematical and theoretical models often have to handle exponentially increasing data and high computational costs. The main goal of **Speciate** is to address the complexity of these simulations by balancing computational efficiency with the complexity of digital organisms.

In this project, evolutionary theories have been unified, including Darwin's theory of natural selection, Oparin's hypothesis on the origin of life, and the Lucacene hypothesis, which have guided the characteristics of the simulated organisms. These organisms mimic, as closely as possible, the most primitive organisms that have existed and the way they could mutate.

From a software engineering perspective, **Speciate**'s architecture follows a Model-View-Controller pattern, offering a graphical user interface for configuring simulations and visualizing results. Users can set the initial parameters of the simulations, such as the number of species, their genetic code, mutation rates, and environmental factors like energy acquisition and extinction frequencies, which will determine the course of the simulations.

The project concludes by analyzing the results obtained through a case study, where the characteristics of the organisms—such as the influence of mutability, energy acquisition capability, and mobility on population success—are contrasted. This educational tool aims to support teachers and students in understanding complex evolutionary processes.

Keywords: Evolution, mutation, simulation, unicellular organism, probabilistic model.

Índice

1 Introducción

En esta sección se presenta una visión general del contexto y la motivación detrás del desarrollo de este TFG. Además, se da una breve introducción sobre la metodología utilizada para realizar el proyecto.

1.1. Motivación

El estudio de sistemas biológicos es complejo debido a la multitud de parámetros necesarios para formar modelos teóricos y matemáticos. Como resultado, los modelos tienden a simplificarse, enfocándose en parámetros específicos de interés (Matthias Ruth, 1997). Los modelos de evolución, son también sistemas biológicos, por lo que presentan estos mismos problemas: debido a que requieren simular y estudiar la reproducción de seres vivos, una máquina que ejecute las simulaciones debe enfrentarse a ambientes de crecimiento exponencial. Para que sean asequibles de ejecutar, las simulaciones deben minimizar el coste computacional, limitándose a estudiar parámetros específicos.

Existen pocos modelos que exploren la evolución de organismos y la influencia de la velocidad de reproducción y mutación en el comportamiento de las simulaciones, generalmente centran el estudio en una característica específica, como la naturaleza de las células cancerígenas (Atitey, 2022). Tierra o Avida son ejemplos de plataformas centradas en estos modelos generales (Maciej Komosinski, 2009). Estas plataformas tratan a los organismos simulados como organismos digitales, en contraste con las simulaciones numéricas (Matthias Ruth, 1997), permitiendo la simulación directa de evolución, a cambio de suponer un mayor coste computacional.

Por medio de una plataforma híbrida que trate a los organismos como poblaciones cuando el modelo lo permita y como individuos cuando sea necesario, se puede lograr la eficiencia computacional de los modelos numéricos a la par que la complejidad de los organismos di-

gitales. Por esto, la finalidad de **Speciate**, la herramienta desarrollada en este proyecto, es justamente analizar los aportes de las mutaciones y su influencia en la evolución por medio de un modelo híbrido.

1.2. Objetivos

En este Trabajo de Fin de Grado (TFG) se ha desarrollado un simulador probabilístico que modela la evolución de organismos unicelulares primitivos con características simples a organismos complejos por medio de mutaciones aleatorias.

El principal objetivo es diseñar e implementar un modelo que represente la reproducción y diferenciación de especies con parámetros variables definidos por el usuario. Este modelo formará parte de **Speciate**, que debe ofrecer al usuario la siguiente funcionalidad:

- Configurar los parámetros de la simulación (número de especies iniciales, número máximo de mutaciones, número inicial de organismos, código genético de especie/s inicial/es, frecuencia de extinciones, etc.) mediante una interfaz de usuario amigable.
- Simular poblaciones de organismos capaces de reproducirse y mutar.
- Modelar mutaciones que proporcionen características cuantitativas a las poblaciones (probabilidad de reproducción, mutación, obtención de energía, etc).

El simulador integrará diversas teorías sobre la evolución y permitirá evaluar la viabilidad de estas. El resultado del proyecto es una herramienta dedicada a fines didácticos y de investigación en el campo de la biología evolutiva.

Por último, mediante el uso de este simulador, se platean otro objetivo como es contrastar claramente el peso de las mutaciones que proporcionan reproductibilidad y dan lugar al incremento poblacional, frente al peso de las mutaciones que proporcionan control genético, evitando la mutabilidad desproporcionada, que desembocaría en poblaciones inestables.

1.3. Metodología y Organización del documento

Para llevar a cabo el desarrollo de este proyecto, se ha adoptado una metodología iterativa e incremental. Este enfoque permite la incorporación gradual de funcionalidad y la constante

validación de cada etapa, asegurando así la coherencia con los objetivos teóricos y prácticos del proyecto. El proceso iterativo se ha dividido en varias etapas que describimos a continuación.

Estudio del dominio y de las tecnologias

La primera etapa, consiste en estudiar los conceptos teóricos acerca de la evolución y mutación celular, modelos celulares y autómatas celulares. Además de teorías sobre células primitivas y sus funciones más básicas.

En esta etapa también se lleva a cabo la revisión de diferentes lenguajes y entornos de desarrollo software para determinar el que mejor se adapta a las necesidades de este TFG.

El resultado de esta etapa está recogido en la Sección 2.

Diseño y Desarrollo software

En esta etapa se lleva a cabo el diseño del simulador y su implementacion. Lógicamente, el primer paso es estudiar los requisitos del simulador. Como esta etapa es la de mayor complejidad y carga de trabajo, se han llevado a cabo varios ciclos o iteraciones, y en cada iteración se añadieron funcionalidades nuevas. Para su correcta implementación, el código debía pasar por multitud de pruebas y la constante corrección de errores.

El proceso y resultado de esta tarea está descrito en las Sección 3.

Prueba y evaluación

En esta última etapa se evalúa el correcto funcionamiento de todas las partes de la aplicación. Se comprueba si se han implementado los requisitos propuestos en las etapas anteriores, si el modelo se implementa de manera correcta y los algoritmos funcionan adecuadamente, si los resultados son lógicos y si la interfaz de usuario es funcional.

Las pruebas y la evaluación del programa se encuentran en la Sección 4.

Por último, este documento incluye una sección donde se presentan las conclusiones y líneas de trabajo futuras (Sección 5) y un apéndice con el manual de instalación y uso.

2
Preliminares

En esta sección se presenta una introducción a las teorías de la evolución biológica, que es el resultado del estudio preliminar que se ha llevado a cabo. Además de las herramientas de simulación que siguen fines comparables a los de este proyecto.

También se recogen las diferentes tecnologías que se han considerado para el elaboracion del TFG y se justifica la elección de algunas de ellas para la realización del proyecto.

Y finalmente se define el funcionamiento del proyecto, y se explican las bases teóricas de los mecanismos que se pretende simular.

2.1. Evolución biológica

El estudio de la evolución biológica ha sido fundamental para entender cómo la vida en la Tierra se ha desarrollado desde su aparición. Se trata de un proceso que engloba la transformación gradual de las especies a lo largo del tiempo y la interacción dinámica entre la variabilidad genética y los cambios ambientales.

Modelar este proceso, aspirando a imitar el inicio de la existencia de los organismos vivos de la forma más precisa posible, proporciona una herramienta poderosa para explorar dicha hipótesis y entender mejor los mecanismos que rigen la evolución de los especímenes.

En el resto de esta sección se describe brevemente las teorías más relevantes a tener en cuenta a la hora de modelar la evolución de seres vivos. En segundo lugar, se presentan algunos modelos preexistentes. Y finalmente, se exploran las herramientas que comunmente se pueden emplear en este ámbito y las que se utilizaron específicamente en este proyecto.

2.1.1. Teorías Evolutivas

Un gran peso de realizar un modelo recae sobre la implementación correcta de los conceptos teóricos de la vida real que se pretenden simular. Para que fuese posible realizar este

proyecto se ha recurrido a algunas de las teorías más influyentes a cerca de la evolución biológica y los seres vivos primitivos. Estas se exponen a continuación.

El Origen de las Especies

La teoría evolutiva de la selección natural propuesta por Charles Darwin en su obra: "El Origen de Las Especies" (Darwin, 1859) explora como los organismos evolucionan mediante cambios graduales impulsados por la competencia, los recursos limitados y la selección de características favorables que promueven la supervivencia, adaptación y reproducción. Es necesario recurrir a esta teoría a la hora de trabajar en modelos evolutivos.

El Origen de la Vida

En su obra "El Origen de la Vida", Aleksandr Oparin (1989), propuso una teoría sobre las características del medio en el que se formaron los primeros organismos. Oparin sugirió que la vida surgió en un ambiente primitivo de condiciones físicas y químicas específicas.

Para simular la evolución de los organismos es necesario recrear ciertas condiciones ambientales y genéticas relevantes en la evolución.

LUCA y LECA

El modelo del "Lucacene" explica cómo el último ancestro común universal o LUCA, que poseía un conjunto de genes compartidos por todos los seres vivos modernos, evolucionó en el Último Ancestro Común Eucariota o LECA (Mikhailovsky & Gordon, 2021).

Esta teoría se puede aplicar en el modelo para reducir el coste computacional y de memoria frente a otros modelos evolutivos. No es necesario tener en cuenta a todos los predecesores de LUCA o LECA, puesto que el antecesor común más primitivo de todos los seres vivos se separó de un grupo de organismos más simples. Esto se apoya en el hecho de que el código genético es universal para todos los seres vivos e incluso para los Acytotas, grupo que describe a los virus entre otros.

2.1.2. Modelos Matemáticos y Computacionales

El estudio de la evolución biológica ha llevado al desarrollo de numerosos modelos matemáticos y computacionales que permiten simular y analizar los procesos evolutivos. A con-

tinuación, se presenta una revisión del estado del arte de algunos modelos matemáticos y computacionales de la evolución de organismos.

DEBGOE

Los modelos de evolución en tiempo discreto son herramientas matemáticas que explican cómo cambian las poblaciones a lo largo de generaciones discretas. El "DEGBOE" (Atitey, 2022) es un ejemplo de modelo de evolución en tiempo discreto, evalúa la mutación genética en linajes celulares para la progresión de células cancerígenas a partir de técnicas matemáticas.

Se trata de un modelo muy eficiente en simular la mutación de linajes celulares. Es además un modelo muy especializado, haciendo que sea eficaz en el ámbito de las células cancerígenas, sin embargo, esto dificulta su aplicación en otros ámbitos.

Tierra y Avida

Los entornos Tierra y Avida son plataformas de simulación que modelan la evolución de organismos digitales. Tierra es un sistema artificial de vida que simula procesos evolutivos en un entorno de computación paralela. Avida, por otro lado, es un software de investigación que utiliza algoritmos evolutivos para estudiar la evolución (Maciej Komosinski, 2009).

Ambos programas permiten simular vida artificial en entornos controlados. Son capaces de simular directamente la evolución, a cambio de tener un mayor coste computacional debido a la naturaleza de los organismos simulados.

Modelado de Sistemas Biológicos Dinámicos

Bruce Hanon y Matthias Ruth (1997) hacen uso de conceptos matemáticos para implementar proporciones y fórmulas simples empleando STELLA, una herramienta de programación gráfica que facilita el procesamiento y análisis de modelos complejos de forma rápida y automática.

Este modelo trata a los organismos mediante simulaciones numéricas usando diagramas de bloques. Esto permite a las simulaciones ser muy eficientes, pero dificiles de escalar cuando son necesarios diagramas complejos, y menos flexibles que simulaciones basadas en programación.

2.2. Tecnologías y Herramientas

A continuación se exponen algunas de las tecnologías que se han considerado para la implementación del simulador de sistemas biológicos.

2.2.1. Herramientas de modelado de propósito general

Tras analizar el estado del arte, quedó claro que un tipo de recurso comúnmente empleado para recrear simulaciones son precisamente las herramientas de modelado, en especial, las de simulaciones de diagramas de bloques.

STELLA Architect: se trata de una herramienta de modelado diseñada para crear simulaciones de diagramas de bloques y presentaciones profesionales (isee systems, inc., 2024). Se ha tenido en cuenta debido a su uso en modelos similares, sin embargo, finalmente no se ha empleado por motivos explicados a continuación.

Simulink: es una herramienta similar a STELLA que permite crear diagramas de bloques complejos, siendo a su vez un entorno muy flexible, especialmente, gracias a que Simulink admite la opción de trabajar en conjunto con una plataforma de programación: MATLAB.

Estas dos herramientas, muy similares entre sí, se han empleado en proyectos similares para modelar y estudiar la evolución de organismos, tanto STELLA (Matthias Ruth, 1997) como Simulink (MathWorks, 2024b).

Finalmente, ambas herramientas se han descartado en favor de opciones que emplean lenguajes de programación, por la flexibilidad casi total de estos se alinea mejor con los objetivos del proyecto, ya que manejar diagramas de bloques habría complicado en gran medida la implementación.

2.2.2. Lenguajes de programación y entornos de desarrollo

Existen gran variedad de lenguajes de programación que nos permiten implementar un modelo y un simulador con la funcionalidad que se plantea en este proyecto. Entre los lenguajes que se han considerado hay que resaltar dos: MATLAB y Java.

MATLAB

MATLAB es una plataforma de programación de propósito general y cálculo numérico empleada comúnmente para analizar datos, desarrollar algoritmos y crear modelos (MathWorks, 2024a).

Planteando el modelo como un conjunto de matrices de contenido numérico fue una opción viable. Sin embargo, el coste de computación aumentaba rápidamente para las simulaciones que requerían matrices de gran tamaño, y éstas debían ser de dimensiones fijas. Por ello, se descartó este lenguaje en beneficio de otro capaz de trabajar con objetos.

Java

Java es un lenguaje de programación de propósito general orientado a objetos. Es un lenguaje de programación muy extendido en la que se crean muchos servicios y aplicaciones (Oracle, 2024), gracias, entre otras cosas, a la gran variedad de librerías y la documentación disponible.

Java es el lenguaje en el que se ha desarrollado tanto el simulador, como la interfaz de usuario gracias a su multitud de librerías, algunas de ellas desarrolladas por terceros. Por ejemplo, para la generación de gráfos dinámicos se ha utilizado GraphStream (Julien Baudy, Antoine Dutot, Yoann Pigné, Guilhelm Savin, 2024). Estos grafos permiten un representación gráfica de los resultados computados por el simulador.

Otro aspecto a destacar de Java, es que existen diferentes entornos de desarrollo para programar usando este lenguaje.

Eclipse es una plataforma de desarrollo integrada (IDE) que habilita la programación en Java, aunque también existen versiones para otros lenguajes de programación. Esta herramienta proporciona un entorno flexible con variedad de extensiones (Eclipse, 2024a). Por ejemplo, en este proyecto se recurre a la extensión Eclipse WindowBuilder, para el diseño de la interfaz gráfica de usuario con Java (Eclipse, 2024b).

2.2.3. Análisis de datos: Machine Learning

Existen multitud de programas y librerías que se dedican al análisis de datos empleando software de "machine learning". Una de las herramientas más completas de "machine learning"

con la que se aprende a trabajar en Ingeniería de la Salud es WEKA.

WEKA Data Platform ayuda a las organizaciones a almacenar, procesar y gestionar datos en la nube y en sus instalaciones para impulsar las cargas de trabajo de nueva generación (WEKA, 2024)·

Se puede utilizar esta herramienta para organizar los datos obtenidos a partir del modelo de Java, exponerlos de manera gráfica, buscar patrones en los datos de los modelos y sacar conclusiones.

2.3. Bases teóricas del proyecto

Para modelar la reproducción, mutación y muerte de los organismos se ha tenido que recurrir a multitud de simplificaciones y formas de interpretar el comportamiento de los organismos del mundo real por medio de números y algoritmos.

En este proyecto se ha optado por modelar organismos primitivos, para ello, ha sido necesario investigar el origen de los primeros seres vivos, el medio en el que prosperaron y sus características más relevantes. En primer lugar, los seres vivos no podían generarse de manera espontánea en un ambiente totalmente esterilizado, tal y como explica Oparin (1989, p. 2-3). Incluso los seres vivos unicelulares están formados por estructuras extremadamente complejas y los organismos más complejos evolucionaron gradualmente a partir de unos más simples (1989, p. 4-5).

2.3.1. Mecanismos de reproducción y mutación

La replicación es un mecanismo complejo que requirió un gran periodo de tiempo para llegar a producirse, pero una vez que los primeros organismos fueron capaces de replicarse, sus poblaciones pudieron crecer. En este proyecto se toma la replicación como una de las características más básicas de todos los organismos y será la forma principal de que aumente el número de nuevos individuos y poblaciones en la simulación después de la entrada inicial de organismos base. El modelo puede ser adaptado para entradas en forma de escalón, rampa o periódicas para simular ambientes no esterilizados que aporten organismos al modelo por diferentes medios.

Según la teoría del "Lucacene" (Mikhailovsky & Gordon, 2021), todos los seres vivos mo-

dernos descienden de un último ancestro común universal o LUCA. Es muy plausible que las primeras células sobreviviesen y mutasen mucho antes de que una en particular alcanzase la mutación necesaria para prosperar por encima de las demás, por lo que se ha priorizado adaptar el modelo para simular poblaciones que aparecen a partir de un pulso inicial.

El funcionamiento de la reproducción de los organismos es flexible e interpretable de diferentes maneras según los parámetros que se empleen. Pero al seguir la teoría del "Lucacene", se asume que se trabaja con organismos procariotas con ADN bicatenario. Así, el método de reproducción de los organismos simulados se basa en los mecanismos de replicación del ADN por medio de enzimas. **Speciate** almacena los códigos genéticos en forma de cadenas de mutaciones, en lugar de cadenas de pares de nucleótidos. Se optó por esta opción de diseño debido a que reduce el coste computacional a cambio de realismo, ya que es necesario almacenar mucha menos información por cada gen que aporta alguna característica útil. Se tiene en cuenta que la probabilidad de mutar se corresponde con la probabilidad que tiene un organismo, en el momento de la replicación del ADN, de desarrollar un gen que aporte alguna característica distinguible. Aunque el usuario puede introducir genes que no aporten característica alguna, y puede alterar la probabilidad de mutar de los organismos para crear simulaciones más fieles a la realidad.

La reproducción en sí está modelada por medio de la lectura del número de individuos aptos para reproducirse en cada población. En el último paso de una generación o paso discreto de tiempo, todos los seres aptos dan lugar a una copia suya que puede generarse con una mutación o no, según la probabilidad asociada a cada población. La reproducción se hace de manera que el individuo progenitor que se reproduce permanece en su población original sin mutar y da lugar a una célula hija. Esto imita la duplicación de una cadena simple de ARN. Es decir, los individuos solo se reproducen duplicándose, y no multiplicándose por otro número mayor, y no hay intercambio de material genético tras la duplicación.

2.3.2. Mecanismos de muerte

Para definir los métodos de muerte de los organismos, fue necesario analizar las posibles causas que tendrían para morir en el ambiente simulado. Los primeros organismos se enfrentaban a un ambiente agresivo, que podía causar la muerte de los organismos (Oparin, 1989, p. 5-6), aunque era necesario para fomentar la producción de compuestos orgánicos (1989, p. 20-

21). Esto fue modelado por medio del factor catástrofe, una función del modelo que se encarga de eliminar a todo un grupo de seres vivos que tienen en común una o varias mutaciones, simulando las extinciones debidas a la ausencia de un gen que aporte resistencia a la especie. Este factor incluye la muerte por toxicidad, que consiste en el aumento de compuestos químicos tóxicos para organismos sensibles a estos, bien por la ausencia de un gen que los proteja, o bien por la presencia de uno que los vuelva sensibles. También incluye la muerte por alteraciones en la temperatura del medio y otros factores similares que puedan resultar catastróficos para las distintas especies.

Otro mecanismo incluído fue la muerte por gasto excesivo de alimento (Oparin, 1989, p. 27). Al no obtener energía suficiente bien del medio o bien por procesos del organismo, este muere. Por medio del mecanismo de muerte por falta de alimento también se tiene en cuenta la muerte por lisis celular o lesión masiva, por la cual la célula es incapaz de regenerarse a tiempo, aunque este proceso puede a su vez incluirse en la categoría de muertes catastróficas: muertes accidentales (catástrofes físicas en todo el medio), muertes por plasmólisis (cambios en el pH catastróficos), etc. No todas las poblaciones de células decrecerán por falta de alimento, ya que una vez gastados los recursos, algunos organismos pueden vivir en estado de latencia (Guerrero & Berlanga, 2001, p. 19).

Otro mecanismo de muerte que se probó para el modelo consiste en un algoritmo que permite a los organismos atacarse los unos a los otros. No obstante, este es un mecanismo del que los seres más primitivos no disponían y que debían desarrollar al ser enfrentados a la escasez de recursos, por lo que no es básico de todos los seres (Guerrero & Berlanga, 2001, p. 17), y el desarrollo del algoritmo de esta causa de muerte no fue prioritario.

Ningún otro mecanismo de muerte fue incluido, los organismos del modelo tienen longevidad indefinida debido a que, para realizar la apoptosis por edad, son necesarios determinados genes (Kiraz et al., 2016), de los cuales estos organismos promitivos carecen.

3

Diseño e Implementación Speciate

Como se comentó en la Sección 1, el objetivo del proyecto es diseñar e implementar una herramienta dedicada a fines didácticos y de investigación en el campo de la biología evolutiva. Dicha herramienta, que hemos llamado **Speciate**, proporcionará una plataforma interactiva que permitirá a los usuarios simular y observar de forma práctica y visual el proceso de evolución en poblaciones de organismos.

La versatilidad y accesibilidad de la herramienta permitirá tanto a alumnos como a profesores contrastar hipótesis acerca de la influencia de las mutaciones en el desarrollo de las poblaciones.

En esta sección se presenta el diseño de este simulador y su implementación. Empezaremos con el análisis de requisitos de **Speciate**, continuaremos con la arquitectura general y el diseño de los componentes principales, y por último se describiran los aspectos más relevantes de la implementación.

3.1. Definición del proyecto

Se trata de un simulador probabilístico que modela la evolución de organismos unicelulares primitivos a organismos más complejos a través de mutaciones aleatorias.

El simulador permitirá a los usuarios definir y ajustar parámetros clave para observar cómo estos afectan la evolución de las especies en la simulación.

Internamente se almacenan las cantidades de distintos grupos de *Individuos* en *Poblaciones*, clasificados según su código genético (*Especie*) y la *Generación* y orden en los que el progenitor

de cada individuo mutó para dar lugar a su *Especie*. Durante cada *Generación*, porcentajes de *Individuos* aptos de cada *Población* interactúan con el medio, sobreviven y se reproducen. Al reproducirse, una fracción de los *Individuos* descendientes pueden mutar, resultando en *Especies* nuevas que coexisten con las ya presentes. La aptitud y los resultados a los que se somete cada *Individuo* dependen de sus mutaciones, determinadas por su *Especie*.

En la **Figura 1** se muestran dos posibilidades de cómo una *Población* inicial puede pasar de una *Generación* G0 a una siguiente *Generación* G1, dependiendo del valor de la *Probabilidad de Mutar* PM. Se observa que la *Población* inicial de *Individuos*, de cantidad $C = I$, indicada por el usuario, y *Especie* desconocida, también seleccionable por el usuario, puede no mutar y multiplicar el número de sus *Individuos* por la *Capacidad de Reproducirse* $R > 0$, de ser PM nula. O bien, si $PM > 0$, algunos *Individuos* descendientes de la *Población* inicial mutarían y formarían una *Especie* diferente, donde el único individuo que la forma puede compartir parte del código genético con la *Población* de su progenitor. Los colores diferentes de las *Poblaciones* representan códigos genéticos distintos.

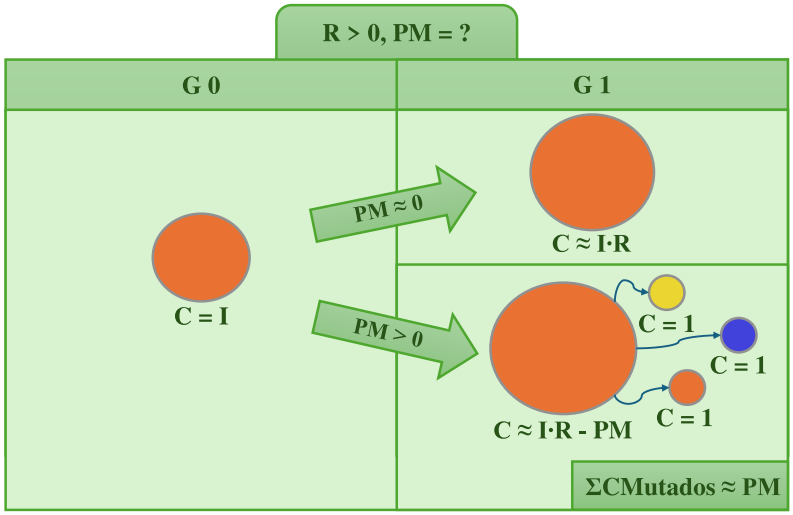

Figura 1: Ejemplo de paso de una generación de parámetros iniciales G0 a una generación G1

En la **Figura 2** se representa el desarrollo del modelo para múltiples pasos N entre las *Generaciones*.

- Se representa una *Población* inicial, por medio de un círculo naranja grande, que tiene una PM moderada. Esta *Población* muta en 3 poblaciones diferentes y a su vez que crece a un ritmo estable de casi R^N.

- La PM de la *Población* representada por medio de un círculo amarillo es casi nula, haciendo que su ritmo de crecimiento sea incluso mayor que la de la *Población* de la que desciende.

- El círculo azul representa un descendiente con una *Mutación* que aumenta la PM en gran medida, mitigando el crecimiento de la *Población*, al provocar que todos los descendientes se generen mutados y que, por tanto, formen parte de otras *Poblaciones*.

- El círculo naranja pequeño representa una *Población* que comparte *Especie* con la *Población* del progenitor a partir del cual mutó. Esto a priori puede parecer irrelevante, pero vuelve el modelo muy versátil, siendo una manera de representar las mutaciones que no aportan cambios al genoma, o permitiendo estudiar la evolución convergente (Cerca, 2023) para simulaciones más grandes. Además, reduce el coste de computación, ya que no es necesario comprobar si el código genético de un descendiente mutado coincide con el de alguna otra *Población* existente.

Cabe recalcar que este ejemplo ha sido construido con propósitos explicativos y representa el funcionamiento del modelo con pleno detalle.

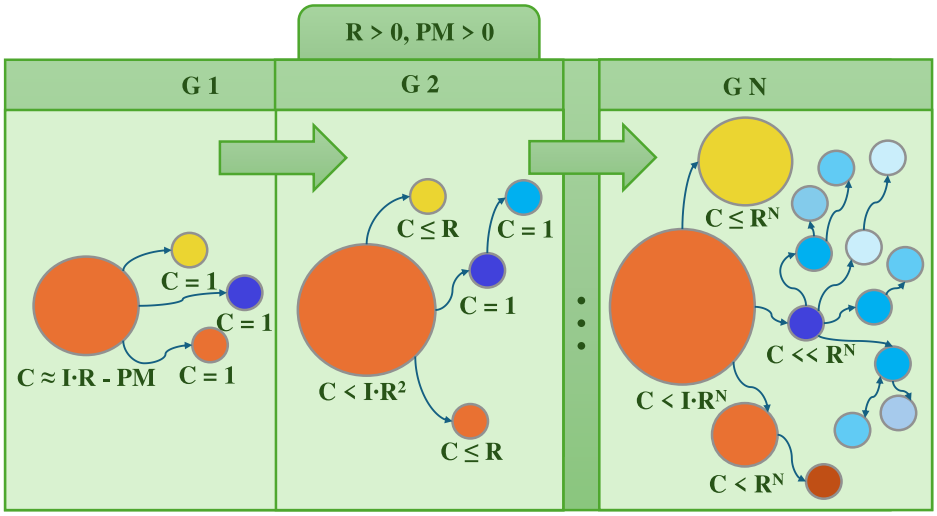

Figura 2: Ejemplo de paso de una generación G1 a una generación hipotética GN

3.2. Análisis de Requisitos

Antes de realizar el diseño e implementación de **Speciate**, se ha realizado un análisis de los requisitos que debe satisfacer para cumplir con los objetivos y que proporcione una experiencia de usuario satisfactoria. Los requisitos se han clasificado, como es habitual, en funcionales y no funcionales. Las Tablas 1, 2 y 3 muestran el códgo y descripción de los requisitos. En total, se han identificado 18 requisitos funcionales (con código RF.X) que se han dividido entre aquellos relacionados con las caraterísticas internas del simulador (Tabla 1), principalmente relacionadas con las poblaciones y las mutaciones, y los requisitos funcionales que en la interacción con el usuario (Tabla 2).

Código	Descripción
RF. 1	El sistema debe ser capaz de crear poblaciones con parámetros configurables.
RF. 2	El modelo debe ser capaz de almacenar múltiples poblaciones de organismos de diferentes especies.
RF. 3	El modelo debe simular la evolución de las poblaciones a través de varias generaciones.
RF. 4	Cada mutación debe o bien poder aumentar o reducir la tasa de una o más características de las poblaciones (probabilidad de mutar, coste de alimento y cantidad de población que interactúa por ciclo). O bien debe poder no apuntar ninguna característica extra.
RF. 5	Los tamaños de las poblaciones deben aumentar o reducirse en tamaño por medio de la reproducción o muerte de sus individuos.
RF. 6	Cada ciclo, un porcentaje de la población debe interactuar con el ambiente u otras poblaciones.
RF. 7	Los individuos no deben compartir su alimento almacenado con otros de su población a menos que las mutaciones lo permitan.
RF. 8	Solo se puede reproducir el porcentaje de población que disponga de alimento suficiente para reproducirse.
RF. 9	El sistema debe incluir la posibilidad de eventos catastróficos que afecten a la población de manera aleatoria.

Tabla 1: Requisitos funcionales del simulador

Código	Descripción
RF. 10	El usuario podrá simular distintas poblaciones de individuos.
RF. 11	El usuario podrá configurar el número, código genético y tamaño de las poblaciones iniciales.
RF. 12	El usuario podrá configurar la cantidad de alimento con la que empieza cada población inicial.
RF. 13	El usuario podrá determinar el número de posibles genes distintos de cada simulación.
RF. 14	El usuario podrá determinar el número de generaciones o ciclos durante los que se ejecute cada simulación.
RF. 15	El usuario podrá importar la tabla de probabilidades que determine todas las posibles mutaciones que las especies puedan tener.
RF. 16	El usuario podrá aleatorizar la tabla de probabilidades que determine las mutaciones.
RF. 17	El sistema debe imprimir los resultados de las simulaciones de forma gráfica.
RF. 18	El usuario tendrá la opción de cancelar la ejecución de una simulación.
RF. 19	El sistema debe ser capaz de simular conflictos entre especies por medio de organismos agresivos que se alimenten de otros, y organismos capaces de defenderse.
RF. 20	El sistema debe ser capaz de simular organismos simbióticos entre sí.
RF. 21	El sistema debe ser capaz de simular el paso de organismos procariotas a organismos eucariotas.

Tabla 2: Requisitos funcionales de la interfaz de usuario

Con respecto a los requisitos no funcionales (Tabla 3 con códigos RNF.X) se han identificado 6 y describen aspectos deseables de escalabilidad y usabilidad de la herramienta.

Código	Descripción
RNF. 1	La simulación debe ser capaz de manejar el mayor número de poblaciones y mutaciones posible.
RNF. 2	El tiempo de ejecución de una simulación debe ser lo más reducido posible.
RNF. 3	La interfaz debe ser intuitiva y fácil de usar.
RNF. 4	El sistema debe estar bien documentado en un manual de usuario.
RNF. 5	El efecto de los parámetros en los resultados de la simulación debe describirse, facilitando valores que resulten en simulaciones lógicas.
RNF. 6	La simulación debe ser precisa y reproducible, con una gestión adecuada de errores y mensajes claros para el usuario.

Tabla 3: Requisitos no funcionales del simulador

3.3. Arquitectura general

A continuación, se presenta la arquitectura de la herramienta **Speciate**, que se puede organizar según el patrón de arquitectura software Modelo-Vista-Controlador (Pinzón Núñez et al., 2019), utilizado comunmente para implementar sistemas con interfaces de usuario. Este patrón divide una aplicación interactiva en tres grandes componentes diferenciados:

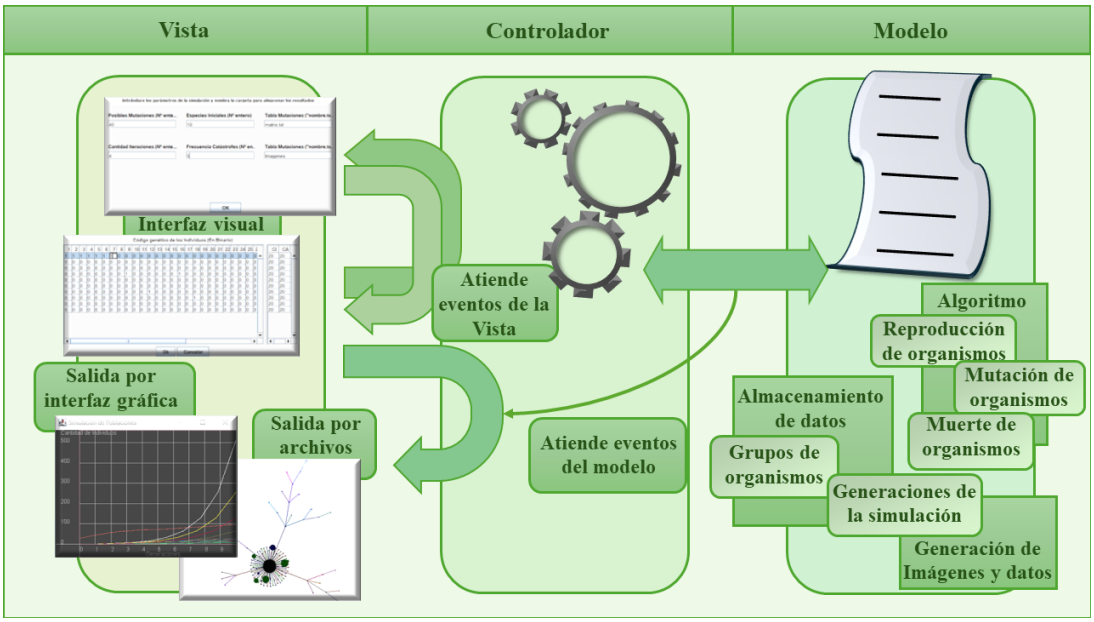

Figura 3: Patrón Modelo-Vista-Controlador

Modelo

El modelo contiene la funcionalidad central de la aplicación y los datos. El modelo de **Speciate** se encarga de calcular los resultados de las simulaciones, para ello almacena los datos de los organismos simulados y calcula cómo se desempeñan estos haciendo uso de operaciones y algoritmos. En nuestro caso, gestiona los siguientes componentes:

- Almacenamiento de datos. Para poder llevar a cabo las simulaciones de organismos, es necesario agrupar a estos en Poblaciones de organismos iguales que comparten un mismo genoma. Estas poblaciones, además, deben almacenarse en instancias de tiempo llamadas Generaciones, que compondrán también el espacio en el que interactúan.

- Algoritmo. Para calcular cómo se desempeñan los organismos, el modelo maneja una serie de operaciones en las cuales los organismos de cada Población pueden o bien reproducirse y aumentar el tamaño de su misma Población; o bien mutar y dar lugar a una nueva Población compuesta por un solo individuo; o bien morir, ya sea por interacción con el ambiente, otras poblaciones o falta de alimento. Todo ello por medio de un único algoritmo que calcula el paso de una Generación de Poblaciones a otra.

- Generación de imágenes y datos. El modelo es el encargado de utilizar toda la información que procesa para generar resultados. Esto lo hace en forma de imágenes que serán mostradas al usuario a través de ventanas o archivos almacenados, y además, el modelo forma salidas de datos de texto para que el usuario sea capaz de analizar las Generaciones de Poblaciones.

Vista

La vista del sistema proporciona los medios para que el usuario interactúe con este. La vista de **Speciate** consiste en dos ventanas de configuración de las simulaciones, y una serie de resultados, que se muestran una vez ejecutada la simulación.

En primer lugar, el usuario puede utilizar la interfaz para proporcionar los parámetros iniciales de la simulación y la carpeta en la que quiere guardar los resultados de esta. Estos datos determinan las características básicas de la generación que se va a ejecutar y la cantidad de

datos que se debe manejar. Se explican en más profundidad en el apartado de Implementación. Al iniciar la aplicación, los campos correspondientes a estos parámetros se encuentran predefinidos con valores por defecto, para ayudar a un usuario.

Tras modificar todos los datos necesarios, el usuario podrá editar con mayor detalle las entradas de la simulación, en el último paso antes de iniciarla.

Finalmente, la vista muestra las imágenes y los datos de resultados generados a través de la interfaz. Los resultados se presentan en tres formas principales: gráficos de líneas y grafos de puntos.

- Los gráficos de líneas representan la cantidad de organismos en cada Población a lo largo de las Generaciones o el tiempo, con el eje X mostrando el tiempo y el eje Y mostrando la cantidad.

- Los grafos de puntos ilustran el estado de las Poblaciones en una Generación o instante dado y sus relaciones de descendencia. En estos grafos, los puntos de mayor tamaño indican las poblaciones que han tenido éxito en reproducirse, mientras que los puntos de menor tamaño representan las poblaciones que no lograron crecer en el momento del tiempo correspondiente al grafo.

- La salida de texto guarda todos los datos relevantes de la última Generación que se calculó en la simulación. Estos pueden ser analizables, bien manualmente o bien por medio de herramientas de análisis de datos externas como Weka.

Controlador

El controlador actúa como intermediario entre la vista y el modelo.

Atiende a los eventos que surgen tanto por parte de la vista como por parte del modelo. Es decir, las acciones realizadas por el usuario a través de los distintos campos y botones de la interfaz, y los resultados generados por los algoritmos del modelo. Y utiliza esta información para permitir al usuario navegar a través de la interfaz visual, o poner en funcionamiento el simulador y para organizar los resultados generados, mostrándolos adecuadamente a través de los medios que les correspondan.

El controlador también gestiona que las operaciones pesadas del modelo trabajen en segundo plano para computar las simulaciones.

3.4. Casos de uso

En este punto presetamos un diagrama de casos de uso general a partir de los requisitos descritos anteriormente. Este diagrama describe el comportamiento del sistema desde el punto de vista del usuario según sus acciones y reacciones.

En el caso de **Speciate**, solo aparece un tipo de actor, el **Usuario**, que engloba a cualquier persona que interactúe con el sistema.

Al tratarse **Speciate** de una aplicación de escritorio cuyo fin principal es ejecutar simulaciones para generar resultados en la máquina del usuario, no requiere de bases de datos ni acceso a internet. Por tanto, solo es necesario un tipo de usuario con la capacidad de acceder a todas las funciones de la aplicación.

3.4.1. Especificación de casos de uso

En este apartado se detallan todos los casos de uso, mostrados en la Figura 4, que describe las formas de interactuar con **Speciate** que tiene disponibles el usuario.

Cada caso de uso viene especificado con un código CUX, siendo X el número del caso de uso; una definición; la precondición necesaria para poder acceder a CUX; las excepciones por las que el caso de uso podría no funcionar de forma normal; y la frecuencia, que puede ser Baja, Media o Alta dependiendo de lo común que sea para el usuario toparse con CUX. Alta se corresponde a cualquier uso normal del programa, Media a casos que no tienen por qué cumplirse en toda ejecución, y Baja a casos que no suelen ocurrir.

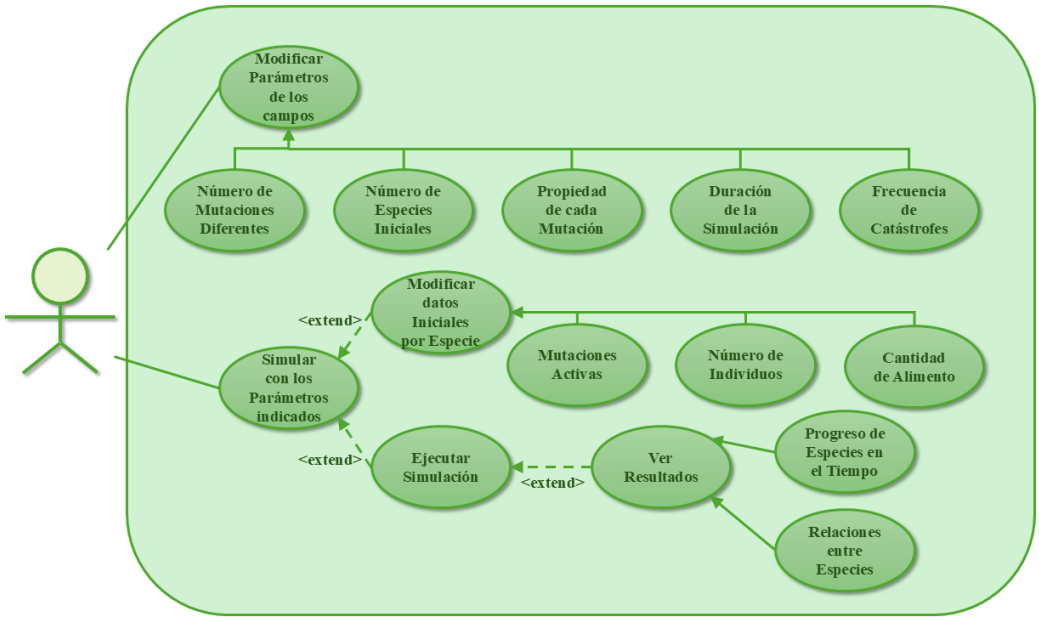

Figura 4: Casos de Uso

CU1	**Configurar simulación**
Definición	El sistema permitirá que el usuario pueda alterar el texto de los campos de los parámetros que se encuentran seleccionados por defecto en la pantalla de “Configuración Inicial” de la aplicación.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos tienen un formato incorrecto.
Frecuencia	Alta.

CU2	**Número de mutaciones diferentes**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “número de mutaciones diferentes”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de número entero.
Frecuencia	Media.

CU3	**Número de especies iniciales**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “número de especies iniciales”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de número entero.
Frecuencia	Media.

CU4	**Propiedad de cada mutación**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “propiedad de cada mutación”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de texto.
Frecuencia	Media.

CU5	**Duración de la simulación**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “duración de la simulación”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de número entero.
Frecuencia	Media.

CU6	**Frecuencia de catástrofes**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “frecuencia de catástrofes”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de número entero.
Frecuencia	Media.

CU7	**Nombre de carpeta para imágenes**
Definición	El sistema permitirá que el usuario pueda seleccionar y reescribir el texto del campo “nomrbe de carpeta para imágenes”.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no tienen formato de texto.
Frecuencia	Media.

CU8	**Simular con los parámetros indicados**
Definición	El sistema permitirá que el usuario pueda generar la siguiente pantalla de la interfaz a partir de los parámetros que se encuentren en los campos en ese momento.
Precondición	Acceder a la pantalla de “Configuración Inicial”.
Postcondición	El sistema mostrará la pantalla de “Valores por Especie”.
Excepciones	
Frecuencia	Alta.

CU9	**Modificar los datos iniciales por especie**
Definición	El sistema permitirá que el usuario pueda alterar el texto de los campos de los parámetros que se encuentran seleccionados por defecto en la pantalla de "Valores por Especie" de la aplicación.
Precondición	Acceder a la pantalla de "Valores por Especie".
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos tienen un formato incorrecto.
Frecuencia	Media.

CU10	**Mutaciones activas**
Definición	El sistema permitirá que el usuario pueda alterar el texto del campo "mutaciones activas", bien seleccionando y reescribiendo el texto de cada posición de la tabla, o bien haciendo click derecho para sumar 1 a la casilla sobre la que se encuentre el ratón, y click izquierdo para restar 1 a la casilla.
Precondición	Acceder a la pantalla de "Valores por Especie".
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no son un número entero de 16 bits.
Frecuencia	Media.

CU11	**Número de individuos**
Definición	El sistema permitirá que el usuario pueda reescribir el texto del campo "número de individuos" de cada especie.
Precondición	Acceder a la pantalla de "Valores por Especie".
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no son un número entero.
Frecuencia	Media.

CU12	**Cantidad de alimento**
Definición	El sistema permitirá que el usuario pueda reescribir el texto del campo “cantidad de alimento” de cada especie.
Precondición	Acceder a la pantalla de “Valores por Especie”.
Postcondición	Se actualizarán los datos de la pantalla.
Excepciones	Los datos introducidos no son un número entero.
Frecuencia	Media.

CU13	**Ver resultados**
Definición	El sistema permitirá al usuario acceder a todas las salidas intencionadas.
Precondición	Acceder a la pantalla de “Valores por Especie” y pulsar el botón “OK”.
Postcondición	Se generarán imágenes de grafos y un archivo de texto con los datos de la última generación. Además, se abrirá una nueva pestaña mostrando una gráfica del “Progreso de las especies”.
Excepciones	Algún dato introducido no tenía el formato correcto.
Frecuencia	Alta.

CU14	**Ver progreso de especies**
Definición	El sistema permitirá que el usuario pueda visualizar un gráfico de líneas representando el progreso de las especies a lo largo de las generaciones.
Precondición	Acceder a la pantalla “Progreso de especies”.
Postcondición	Se habrá generado una gráfica conteniendo mostrando los cambios en el número de individuos de cada población en el tiempo.
Excepciones	
Frecuencia	Alta.

CU15	Ver relaciones entre especies
Definición	El usuario podrá visualizar la secuencia de las diferenciaciones de las especies y sus relaciones parentales a través de una secuencia de imágenes almacenada.
Precondición	Acceder a la carpeta indicada a través del campo “nombre de carpeta para imágenes”.
Postcondición	Habrá una cantidad de “Cantidad Iteraciones” de imágenes .png en la carpeta indicada.
Excepciones	La simulación no ha terminado de ejecutarse.
Frecuencia	Alta.

CU16	Ver detalles de la última generación
Definición	El usuario podrá visualizar una lista de las poblaciones existentes en la última generación de la simulación a través de un archivo de texto en formato .arff.
Precondición	Acceder a la carpeta indicada a través del campo “nombre de carpeta para imágenes”.
Postcondición	Habrá un archivo compatible con Weka en la carpeta indicada.
Excepciones	La simulación no ha terminado de ejecutarse.
Frecuencia	Alta.

3.5. Implementación

En este apartado se explican los detalles a cerca de la implementación del proyecto. Se detallan los motivos por los que se ha seleccionado Java como lenguaje de programación y cómo se utilizó esta herramienta. Además, se describen las herramientas y los programas que fueron empleados para implementar el modelo y la arquitectura del sistema explicada en el apartado de Arquitectura general.

3.5.1. Lenguaje y Entorno de Desarrollo Integrado

Tal y como se anticipó en la Sección 2, para la implementación de **Speciate** se ha realizado en el lenguaje de programación Java, uno de los más conocidos y extendidos, que da acceso a multitud de librerías. Entre los motivos para seleccionar Java están, entre otros, que se trata de un lenguaje orientado a objetos que proporciona implementaciones eficientes de estructuras de datos que han sido de gran utilidad, como por ejemplo las listas enlazadas, el soporte a la concurrencia, la disponibilidad de marco de trabajo Java Swing y AWT para la implementación de interfaces gráficas.

Además, hay gran multitud de entornos de desarrollo para este lenguaje. En concreto, el entorno de desarrollo que se ha empleado para realizar este Trabajo de Fin de Grado ha sido Eclipse, el cual se ha utilizado durante el grado y que además proporciona herramientas y plugins como WindowBuilder, que permiten el diseño de la interfaz gráfica de forma muy intuitiva.

3.5.2. Implementación de la interfaz gráfica

La vista se compone de un total de 3 paneles principales, además de los archivos de resultados que se almacenan en una carpeta indicada por el usuario. El primer panel al que puede acceder el usuario es "Configuración Inicial", relacionado con el caso de uso "Configurar Simulación". Tras terminar con este caso de uso, podrá acceder al segundo panel, "Valores por Especie", correspondiente al caso de uso "Simular con los Parámetros Indicados". Una vez que la simulación se haya ejecutado, el usuario podrá visualizar los resultados a través de la tercera ventana, una gráfica que muestra el progreso de las especies. También podrá visualizar las relaciones de las especies tras cada generación, y los detalles de la última generación, ambos datos guardados en la carpeta indicada por el usuario previamente.

Pantallas interactivas Como se menciona en el aparato anterior, existe un plugin para el entorno de desarrollo Eclipse, llamado WindowBuilder, que permite crear diseños de interfaces gráficas Java Swing de forma sencilla usando *drag-and-drop* (es decir arrastrando componentes gráficos). Este plugin se ha utilizado para diseñar las dos pantallas: "Configuración Inicial" y "Valores por Especie".

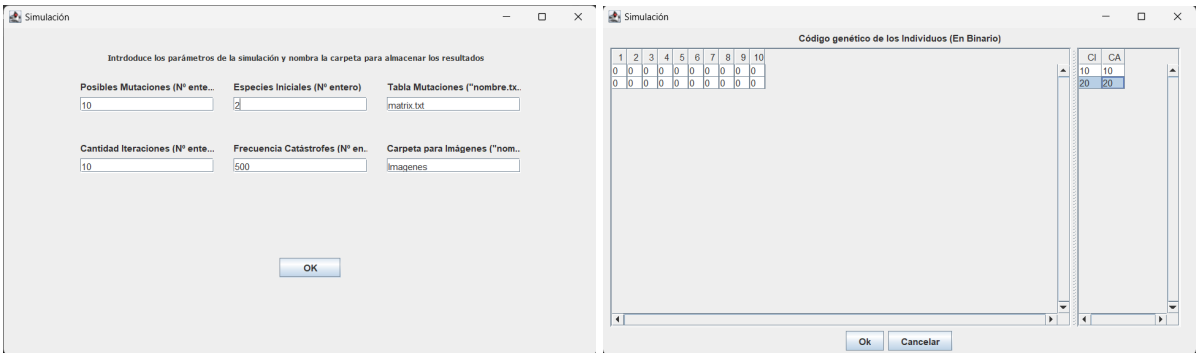

Figura 5: Pantallas de “Configuración Inicial” y “Valores por Especie”.

Grafos Para generar grafos y guardarlos como imágenes (en formato .png) con los resultados de las simulaciones se ha utilizado una librería externa llamada GraphStream (Julien Baudy, Antoine Dutot, Yoann Pigné, Guilhelm Savin, 2024).

Gracias a esta librería, se pueden generar grafos y observar las distintas generaciones de la simulación organizadas en nodos que representan poblaciones, conectadas por aristas según su parentesco. Cada nodo que aparece en estos grafos representa una población de individuos, cuantos más individuos formen parte de dicha población, mayor será el tamaño del nodo, hasta un máximo para mejorar la visualización y evitar en lo posible que se solapen entre nodos. Los aristas relacionarán estos nodos entre padres e hijos, de manera que las nuevas poblaciones generadas estarán unidas a través de una arista a la población a partir de la cual mutaron. Si el usuario introduce varias poblaciones iniciales, los grafos mostrarán tantos grupos independientes de nodos como poblaciones iniciales se hayan introducido. Un ejemplo de grafo de nodos se puede observar en la Imagen 13.

Gráficas de líneas Para generar gráficas de líneas, se ha utilizado Abstract Window Toolkit, una librería de Java. Esta librería, por medio de Graphics2D permite dibujar las gráficas necesarias.

Estas gráficas muestran tantas líneas como poblaciones distintas haya en la generación final del programa, cada línea representa la cantidad de individuos que forman parte de la población correspondiente en una generación concreta, se puede observar un ejemplo en la Imagen 13. Gracias a MouseAdapter, MouseEvent y MouseMotionListener, que manejan los eventos del ratón, el programa puede mostrar datos relevantes al usuario sobre cada población específica por medio de texto emergente.

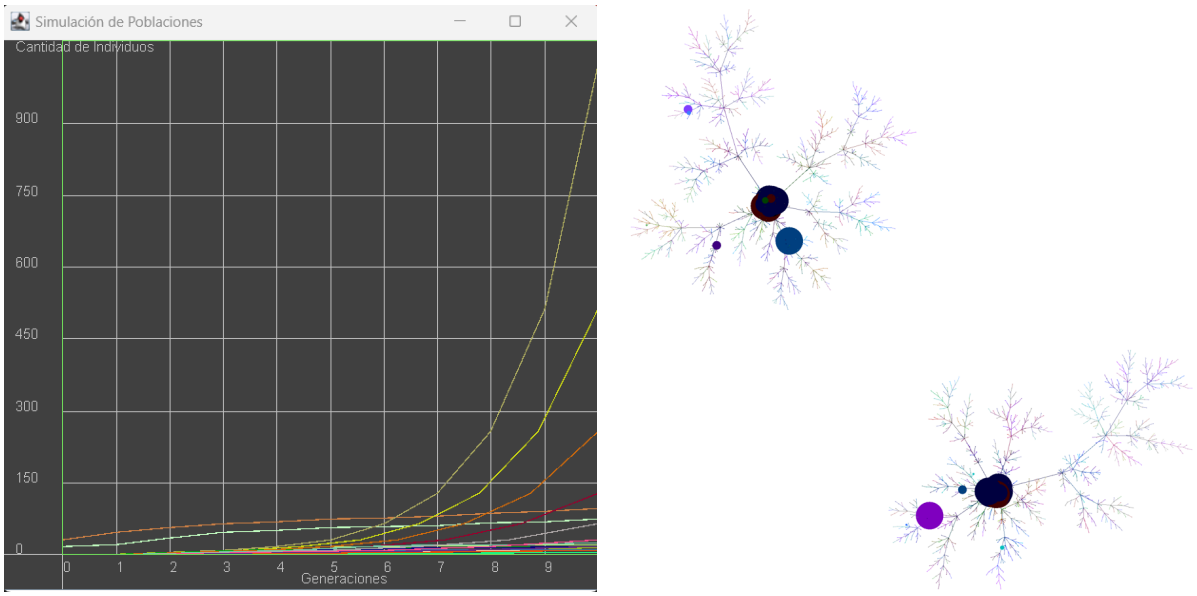

Figura 6: Resultados de gráfica de poblaciones por ventana y ejemplo de grafo.

Salida textual Para que los resultados de las simulaciones puedan ser analizados a fondo, **Speciate** también genera un archivo de texto (en formato .arff) que se almacena en la misma carpeta que las imágenes de los grafos. Este archivo contiene la información de las poblaciones contenidas en la última generación de la simulación, adaptada para ser compatible con el programa de análisis de datos Weka. Los datos y el formato de en el que estos se guardan pueden observarse en la Imagen 7, donde:

- Los atributos de "c0" a "c9" se corresponden con las mutaciones de cada población e indican si están activas o no.
- Los atributos "mutationProb", "refoodCap", "alimentCost" y "movementCap" (explicados en el último apartado de esta Sección) representan las 4 características que puede aportar la tabla de mutaciones a cada población, ya calculadas según las mutaciones activas.
- El atributo "generation" indica la generación en la que nació el primer individio de la población correspondiente.
- El atributo "size" indica el número de individuos perteneciente a cada población.
- El atributo "success" indica si la cantidad de individuos de la población correspondiente es mayor que 1.

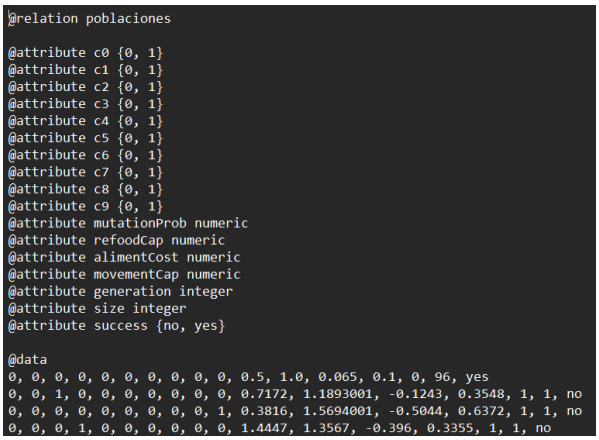

Figura 7: Resultados de la simulación en formato .arff

3.5.3. Controlador y Modelo

Como ya se ha comentado anteriormente, la aplicación **Speciate** tiene una interfaz gráfica para interactuar con el usuario, y también lleva a cabo tareas *pesadas* de simulación, entendidas como tareas que requieren tiempo de computo y de memoria. Para que la interfaz gráfica sea responsiva mientras se está llevando a cabo una simulación, se ha utilizado *programación concurrente* en la implementación. Es decir, la aplicación **Speciate** está compuesta por múltiples hebras que se ejecutan al mismo tiempo para realizar diferetes tareas.

Concurrencia La concurrencia en aplicaciones Java Swing es habitual, ya que el propio framework por defecto crea una hebra que ejecuta las tareas del Controlador del patrón Modelo-Vista-Controlador. En el caso de **Speciate** hemos hecho uso de otro tipo de hebras para realizar las simulaciones.

Por tanto, en la implementación de **Speciate** distinguimos tres tipos de hebras:

- Hebra Main: esta hebra se encarga de ejecutar el programa principal, y se encarga de crear los objetos necesarios del modelo, la vista y el controlador .

- Hebra EventDispatcher: La hebra manejadora de eventos se encarga de procesar los eventos que se producen en la aplicación, pueden ser eventos de la interfaz gráfica o eventos internos del programa. Al procesar los eventos, la hebra manejadora puede, entre otras cosas, actualizar la interfaz gráfica o crear otras hebras, denominadas trabajadoras, para que realicen tareas pesadas en tiempo.

- Hebras SwingWorker: Este tipo de hebras, denominadas trabajadoras, realizan las tareas que consumen más tiempo: el cómputo de las generaciones, las interacciones entre las distintas poblaciones, la cantidad de individuos que se reproducen, las poblaciones nuevas que son generadas, etc. Cuando una hebra trabajadora ha terminado su tarea, se producen una serie de eventos internos en la aplicación para que el controlador muestr los resultados de la simulación en la interfaz gráfica. Como se ha comentado en el punto anterior, estas hebras las crea la hebra EventDispatcher.

Pseudo-código Para determinar el funcionamiento del modelo de **Speciate**, se ha atendido a las descripciones de los mecanismos de reproducción, mutación y muerte, cuyas bases teóricas fueron explicadas en la Sección 2.

Las características de cada especie que determinan cómo ejecutan cada uno de los mecanismos de reproducción, mutación y muerte dependen de 4 características:

- **mutationProb**. Porcentaje de individuos de una población que mutarán cuando se reproduzcan.
- **refoodCap**. La capacidad que tiene cada Individuo de conseguir alimento, generalmente, al moverse durante una generación.
- **alimentCost**. El coste de alimento que supone para un Individuo moverse o reproducirse, se asume que ambas actividades cuestan la misma cantidad de alimento.
- **movementCap**. Porcentaje de individuos de una población que pueden moverse durante cada generación.

El valor de estas características viene determinado por el código genético de cada especie, es decir, se ven alteradas según las mutaciones que estén activas. En la Sección A se explica con mayor detalle cómo manejar cada una de estas características.

Cada vez que se ejecuta una simulación con ciertos valores, el controlador llama al método Generación del modelo a través del worker para transformar una generación de poblaciones en otra siguiendo los mecanismos correspondientes. Este proceso se debe realizar tantas veces como el usuario haya indicado que debe ejecutarse la simulación a través del parámetro "Cantidad Iteraciones" (Algorithm 1, línea 2).

Para ejecutar este proceso, se trabaja con las características explicadas, la cantidad de alimento y la cantidad de individuos de cada población. En primer lugar, se calcula la cantidad de individuos que se mueven a partir del total de la población y movementCap (Algorithm 1, línea 3). Estos individuos consiguen alimento según su refoodCap, pierden alimento en moverse según el alimentCost y mueren los que consumen un mayor número de alimento del que recuperan (esto incluye muerte por extravío en medios agresivos o autofagia) si se ha conseguido menos alimento del que se ha gastado (líneas 4-9). Este funcionamiento del algoritmo viene determinado por las cualidades probabilísticas del mismo, la capacidad de alimentarse se trata como un valor estadístico, cada Individuo tiene una probabilidad de tener éxito al buscar alimento, los que no tienen éxito, mueren. Al trabajar con la población completa en lugar de hacer un cálculo aplicando la probabilidad a cada Individuo, se consigue reducir el número de operaciones del algoritmo drásticamente.

Tras calcular cuánto alimento se ha conseguido y cuántos individuos siguen vivos, se calcula la cantidad que se reproduce, y de estos se calculan cuántos mutan. Los que no mutan se suman directamente a la cantidad de individuos de la población, ahorrando el cómputo de reproducir a cada individuo, pero para los que no mutan, la reproducción se debe calcular individualmente (Algorithm 1, líneas 10, 12-16). Esto aumenta el coste de cómputo para simulaciones concretas en las que las poblaciones iniciales tienen alta probabilidad de mutar.

Reproducirse también gasta alimento, por lo que se calcula cuánto se ha gastado, se almacenan los datos en la población correspondiente (Algorithm 1, línea 11).

Algorithm 1: Simulación de generaciones y poblaciones

Result: Resultado de las simulaciones de generaciones

```
for i ← 1 to CantIter do
    for j ← 1 to CantPob do
        CantIndMov ← (TotIndPob) × (movementCap);
        IndAliment ← (CantIndMov) × (refoodCap);
        IndHambr ← (CantIndMov) × (alimentCost);
        CantAliment ← (CantAliment) + (IndAliment) − (IndHambr);
        if IndHambr > IndAliment then
            TotIndPob ← (TotIndPob) + (IndAliment) − (IndHambr);
        end
        IndReprod ← (TotIndPob) × (CantAliment)/(alimentCost);
        poblacin.actualizarCantAliment((CantAliment) − (IndReprod) ×
         (alimentCost));
        IndMut ← (IndReprod) × (mutationProb);
        for k ← 1 to IndMut do
            Generacin.add(NuevoIndMut);
        end
        poblacin.actualizarTotIndPob((TotIndPob) + (IndReprod) − (IndMut));
        Generacin(j) ← Poblacion;
    end
    Result ← guardarGeneracin(Generacin);
    Result(i).generarGrafo();
    // Se guarda un Grafo de cada generación
end
generarGrafica(Result);
guardarTexto(Result(CantIter));
// Se guardan las demás salidas con la generación
   final
```

4

Pruebas y Resultados

Tal y como se menciona en la Sección 1, para realizar este proyecto se ha adoptado una metodología iterativa e incremental. Por tanto, las pruebas de la aplicación han sido realizadas continuamente durante todo el desarrollo desde las etapas más tempranas del proyecto según se completaban los requisitos descritos en la Tabla 1 de la Sección 3.

Primero se realizaron y comprobaron los requisitos que modelaban los objetos de la simulación (RF.1 y RF.2). Tras esto se desarrollaron los métodos y algoritmos que daban lugar a las simulaciones, estos algoritmos fueron puestos a prueba y los resultados fueron, en un principio, satisfactorios. En tercer lugar, se desarrolló la interfaz de usuario que permitía modificar los parámetros de la simulación de forma cómoda y se comprobó el manejo de errores de esta. Finalmente, se desarrolló la salida de resultados. Comprobando esta última parte, se detectaron errores en el funcionamiento del algoritmo, por lo que se corrigió y se probó que el programa funcionaba satisfactoriamente al completo.

A continuación se describen en más detalle todas las pruebas más relevantes o que han supuesto mayores dificultades durante la elaboración de **Speciate**.

4.1. Pruebas del simulador

En esta sección se describen las diferentes pruebas que se han realizado sobre el código y la interfaz de **Speciate**. En este proyecto, se han realizado pruebas generales sobre el software a medida que se desarrollaba el programa de manera incremental, y se ha comprobado el funcionamiento del programa final a través de *pruebas de caja negra* de forma manual. A continuación, se enumeran algunas de las pruebas realizadas sobre el código.

- Pruebas de objetos. Se ha comprobado que los objetos almacenaban correctamente los datos correspondientes.

- Pruebas de algoritmos. Se ha comprobado que los algoritmos de reproducción, muerte y mutación funcionan adecuadamente.

- Pruebas de números aleatorios. Se ha comprobado que los números aleatorios generados cumplen con los valores esperados.

- Pruebas de interfaz. Se ha comprobado que la interfaz de usuario se ejecuta y los datos introducibles son interpretados y manejados por el modelo.

- Pruebas de hebras. Se ha comprobado que la hebra inicial, la hebra trabajadora y la hebra manejadora de eventos se ejecutan en el orden correcto y manejan los datos que les corresponden.

- Pruebas de funciones generadoras de imágenes. Se ha comprobado que las funciones que generan resultados en forma de imágen empleen los datos adecuados en la hebra correspondiente.

Pruebas de caja negra Las pruebas de caja negra consisten en probar un sistema o programa informático sin tener acceso al código del mismo, por lo que se centra en verificar que la salida observada es el correcto de acuerdo a las entradas de cada prueba. En este proyecto, se ha comprobado que los resultados de estas pruebas fueron satisfactorios.

PR1	**Ejecutar aplicación**
Descripción	Comprobar que la aplicación se ejecuta en la máquina del usuario.
Caso de Prueba	La máquina del usuario tiene instalada la versión 8 de Java, actualización 441 o superior.
Salida esperada	Se inicia correctamente la aplicación y se muestra la pantalla de “Configuración Inicial”.

PR2	**Introducir parámetros**
Descripción	Comprobar que las modificaciones realizadas el usuario en los parámetros de la pantalla de “Configuración Inicial” se ven reflejados correctamente en la pantalla de “Valores por Especie”.
Caso de Prueba	Se introduce valores enteros en “Posibles Mutaciones”, “Especies Iniciales”, “Cantidad Iteraciones” y “Frecuencia Catástrofes” y valores String en “Tabla Mutaciones” y “Carpeta para Imágenes”. Y se pulsa el botón “OK”.
Salida esperada	Se inicia correctamente la aplicación y se muestra la pantalla de “Valores por Especie”. Debe observarse una tabla con el número de columnas determinado por la información introducida en el campo de “Posibles Mutaciones” y el número de filas determinado por “Especies Iniciales”.

PR3	**Cancelar edición de especies**
Descripción	Comprobar que el usuario puede volver de la pantalla “Valores por Especie” a la pantalla “Configuración Inicial” pulsando el botón “Cancelar”.
Caso de Prueba	El usuario pulsa el botón “Cancelar” desde la pantalla “Valores por Especie”.
Salida esperada	Se sustituye la pantalla de “Valores por Especie” por la pantalla “Configuración Inicial”.

PR4	Modificar pantalla de “Valores por Especie”
Descripción	Comprobar que el usuario puede modificar el texto de los campos de la pantalla “Valores por Especie”.
Caso de Prueba	El usuario hace doble click sobre cualquiera de las tablas de la pantalla “Valores por Especie”. O bien hace un único click izquierdo o derecho sobre cualquier posición de la tabla de mutaciones activas.
Salida esperada	El usuario puede editar las tablas por teclado a voluntad. Los valores de la celda de la tabla de mutaciones activas aumentan en 1 con el click derecho o disminuyen en 1 con el click izquierdo.

PR5	Salida de progreso de especies
Descripción	Comprobar que el usuario puede visualizar la gráfica de líneas que almacena el progreso de las especies a lo largo de las generaciones.
Caso de Prueba	El usuario pulsa el botón “OK” de la pantalla “Valores por Especie” teniendo todos los parámetros en el formato adecuado.
Salida esperada	Se abre una pestaña con un gráfico de líneas en el cual el eje Y se alarga hasta el parámetro indicado por el campo “Cantidad Iteraciones”.

PR6	Salida de relaciones entre especies
Descripción	Comprobar que se almacenan los grafos de las relaciones entre especies en la carpeta indicada por el usuario.
Caso de Prueba	El usuario pulsa el botón “OK” de la pantalla “Valores por Especie” teniendo todos los parámetros en el formato adecuado.
Salida esperada	Se guardan un número de “Cantidad Iteraciones” de grafos en la carpeta del programa nombrada por el usuario en el campo “Carpeta para Imágenes”.

PR7	Salida de archivo de texto
Descripción	Comprobar que se almacenan correctamente los datos de la última generación de la simulación en un archivo de texto en la carpeta indicada por el usuario.
Caso de Prueba	El usuario pulsa el botón "OK" de la pantalla "Valores por Especie" teniendo todos los parámetros en el formato adecuado.
Salida esperada	Se genera un archivo .arff que contiene la lista de poblaciones de la última generación de la simulación, junto con el número de la generación en el que fue originada cada especie, su código de especie, la cantidad de individuos que contiene y sus características. El archivo debe ser interpretable por Weka sin errores.

PR8	Compatibilidad de resultados con entornos de análisis
Descripción	Comprobar que el archivo de texto generado puede interpretarse correctamente por el entorno de análisis Weka.
Caso de Prueba	El usuario abre el archivo .arff generado de la salida de texto a través de Weka.
Salida esperada	Weka interpreta el archivo sin errores y se puede analizar por medio de algunas funciones del entorno, como el evaluador "CorrelationAttributeEval", que, gracias al método "Ranker" determina la correlación que tiene un atributo seleccionado con cada uno de los demás atributos del conjunto de datos.

4.2. Validación de los resultados de Speciate

Además de las pruebas ordinarias de aplicaciones sowtware, se han validado los resultados de las simulaciones de **Speciate** mediante el caso de estudio propuesto en los objetivos (Sección 1.2):

" Contrastar claramente el peso de las mutaciones que proporcionan reproductibilidad y dan lugar al incremento poblacional, frente al peso de las mutaciones que proporcionan control genético, evitando la mutabilidad desproporcionada, que desembocaría en poblaciones inestables."

En el Manual de Uso de la herramienta (Anexo A) se explica con más detalle los diferentes parámetros de entrada que necesita **Speciate** para poder ejecutar una simulación. En concreto, los valores de cada población se calculan a partir del sumatorio de los valores aportados por todas sus mutaciones activas y un valor mínimo base. Estos valores son *probabilidad de mutación* ($PMutBase$), *capacidad de conseguir alimento* ($CapAlimBase$), *coste de alimento* ($CostAlimBase$) y *capacidad de movimiento* ($CMovBase$), respectivamente, para cada fila de la tabla. Los valores base que se han utilizado en el caso de estudio son los siguientes: $PMutBase = 0{,}6$, $CapAlimBase = 0{,}3$, $CostAlimBase = 0{,}25$ y $CMovBase = 0{,}6$.

Para este caso de estudio se han utilizado los parámetros iniciales indicados en la Tabla 4.

Número de especies iniciales	1
Cantidad de iteraciones	20
Cantidad de individuos iniciales de la especie	20
Cantidad de alimento inicial de la especie	20
Tabla de mutaciones	Tabla 5
Mutaciones activas inicialmente	ninguna

Tabla 4: Parámetros iniciales del caso de estudio propuesto.

Los datos que se muestran la Tabla 5 codifican las posibles mutaciones posibles en la simulación. Estos números en formato Float corresponden con las probabilidades que aporta cada mutación activa a una especie. Las mutaciones se corresponden con las columnas y las filas corresponden a la probabilidad de mutar, capacidad de conseguir alimento, coste de alimento y capacidad de movimiento respectivamente. Todos estos datos vienen explicados a fondo en la Tabla 9 de la Sección A.

-0.5779	-0.1004	0.2172	-0.5999	-0.5897	-0.5709	0.0300	0.8286	-0.5517	-0.1184
0.1567	-0.1182	0.1893	0.1567	0.1437	-0.1242	0.1275	0.1082	-0.1567	0.1694
-0.1180	-0.3182	0.0893	0.1610	-0.1437	0.1242	-0.1275	-0.1082	-0.1367	-0.1394
0.3214	0.3933	-0.2548	0.2355	0.3824	-0.1978	0.3232	-0.0414	0.2770	-0.1372

Tabla 5: Datos en formato de tabla, mutaciones 0 a 9

4.2.1. Resultados de la simulación

Se ha realizado una simulación con los parametros de entrada anteriores y se han obtenido los resultados que se presentan a continuación. Por un lado, **Speciate** genera un conjunto de imágenes con los grafos de parentesco de las 10 generaciones que se han simulado. Estas imagenes se almacenan en la ruta indicada por el usuario. Las Figuras 8 y 9 muestran algunas de ellas.

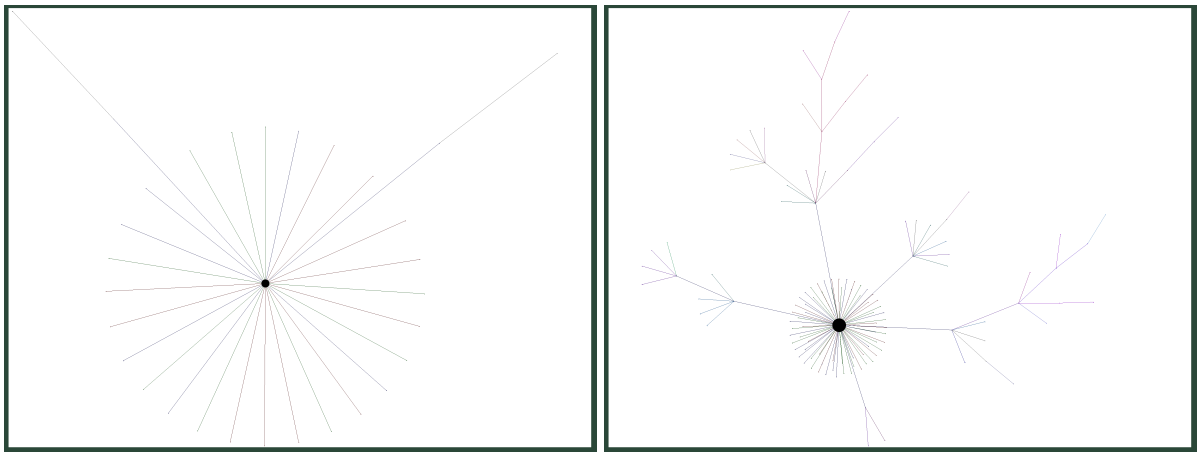

Figura 8: Generaciones 1 y 7 simuladas a partir de los datos iniciales indicados, grafos de parentesco.

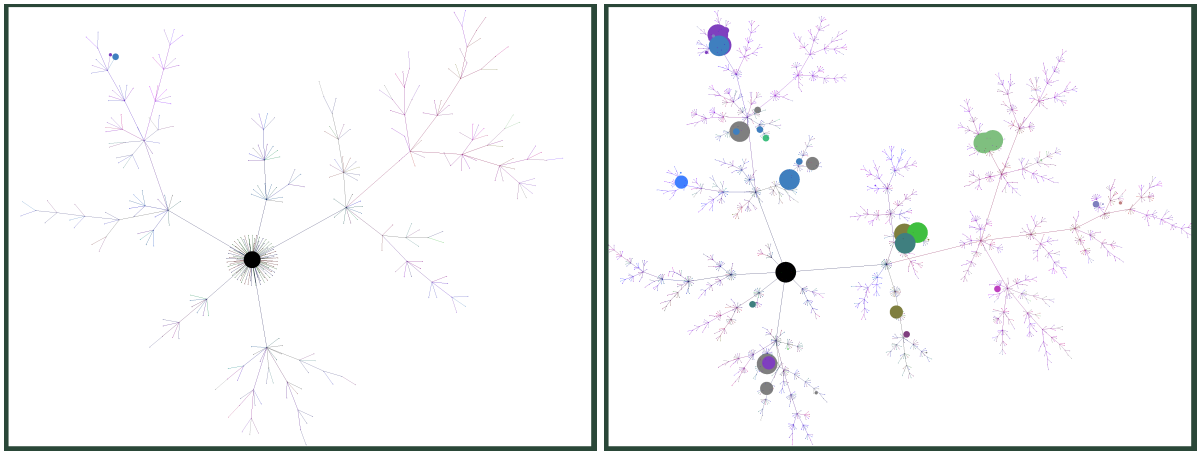

Figura 9: Generaciones 12 y 20 simuladas a partir de los datos iniciales indicados, grafos de parentesco.

Como podemos observar en las Imágenes 8 y 9, las poblaciones de mayor éxito, es decir, las que lograron un mayor tamaño, son aquellas que, a su vez, tienen un menor número de

ramificaciones saliendo de ellas, es decir, engendran menos individuos mutados, algo que les permite crecer.

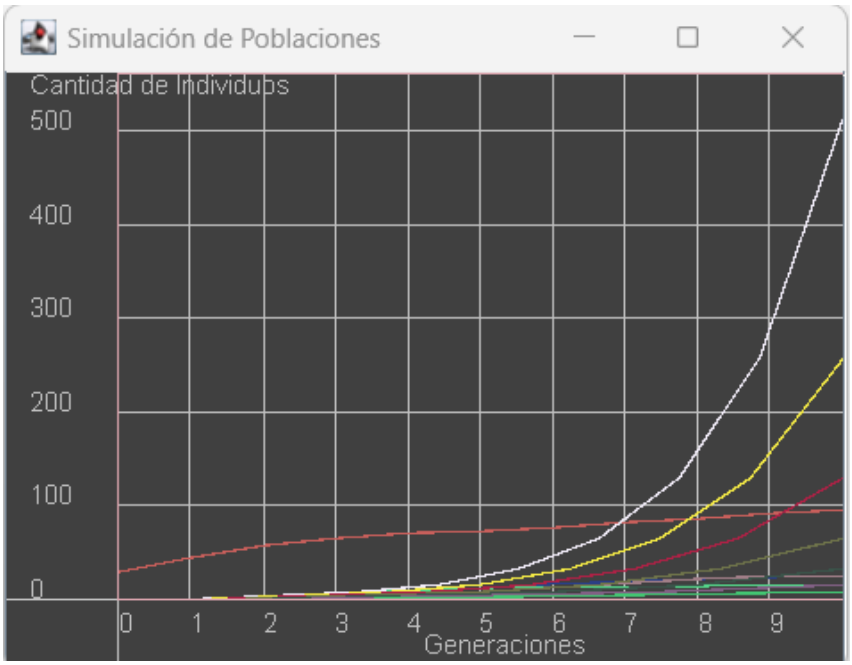

Figura 10: Progreso de las poblaciones a lo largo de 10 generaciones.

Por otro lado, **Speciate** también genera una gráfica 2D con el progreso de las poblaciones a lo largo de las diferentes generaciones, como la que se muestra en la Figura 10. Para este caso de estudio, vemos que algunas poblaciones progresaron especialmente bien, superando incluso a la población inicial, representada en este caso en color naranja, que aumenta su tamaño de manera logarítmica debido a que se rige únicamente por las características base.

Fijandonos en los datos que proporciona esta figura y en la salida textual del programa, vemos que estas poblaciones de mayor éxito comparten generalmente las mutaciones 3, 5, 6, 7, 8 y 9 las cuales, como se observa en la Tabla 5, son las mutaciones que aportan menor coste de alimento, mayor reproductibilidad, pero especialmente, menor mutabilidad.

Entre las poblaciones que no han tenido éxito y se han quedado con un número bajo de individuos, observamos especialmente poblaciones que comparten las mutaciones 2 y 10. Estos resultados son esperables, ya que la relación de coste de alimento frente a la capacidad de obtención de alimento es insuficiente para que la parte de la población con capacidad de moverse se reproduzca.

Pero además, se pueden observar múltiples poblaciones que comparten las mutaciones 1

y 4, que también quedaron un numero bajo de individuos, aún siendo generadas en etapas tempranas de la simulación y teniendo tiempo de reproducirse y aún teniendo bajo coste de alimento. Estas son las poblaciones que dieron lugar a la mayoría de las ramas observables en la Figura 9.

Podemos analizar la correlación que tienen las distintas características con el éxito de la especie gracias a los evaluadores de atributos de Weka. Con el evaluador "Principal Components" usando el método de búsqueda "Ranker", en el apartado de "Correlation matrix" se puede observar la Tabla 6, a partir de la cual podemos sacar las siguientes conclusiones:

- La probabilidad de mutación y el número de la generación en la que se originó una Población tienen una gran corelación inversamente proporcional al éxito de las Poblaciones (-0.49 y -0.81).
- El coste de alimento es también inversamente proporcional, pero la relación es más débil (-0.36).
- La capacidad de movimiento es directamente proporcional, con una relación también algo débil (0.37).
- El programa detecta una relación negativa en la relación de la capacidad de conseguir alimento. Al contrario que el resto de resultados del análisis, esto es antiintuitivo, no obstante, es la relación más débil (-0.24).

	mutationProb	refoodCap	alimentCost	movementCap	generation	success
mutationProb	1.00	-0.07	0.04	-0.49	0.66	-0.49
refoodCap	-0.07	1.00	0.15	0.09	0.21	-0.24
alimentCost	0.04	0.15	1.00	-0.45	0.22	-0.36
movementCap	-0.49	0.09	-0.45	1.00	-0.36	0.37
generation	0.66	0.21	0.22	-0.36	1.00	-0.81
success	-0.49	-0.24	-0.36	0.37	-0.81	1.00

Tabla 6: Matriz de correlación entre los atributos y el éxito (success).

Una posible explicación de estos datos es que los descendientes de las poblaciones con alta tasa de mutación, por lo general tendrán a su vez una tasa alta de mutar. Esto se ve representado

en las ramificaciones que salen a partir de otras ramificaciones. Pero esto también implica que alguna de estas poblaciones con alta tasa de mutación, puede a su vez tener un descendiente con alto control sobre mutaciones que de lugar a una población exitosa. Un ejemplo de esto se puede observar especialmente bien en la Figura 11 en forma de un nodo morado de gran tamaño alejado del nodo inicial y sus hijos imnediatos a través de varias generaciones.

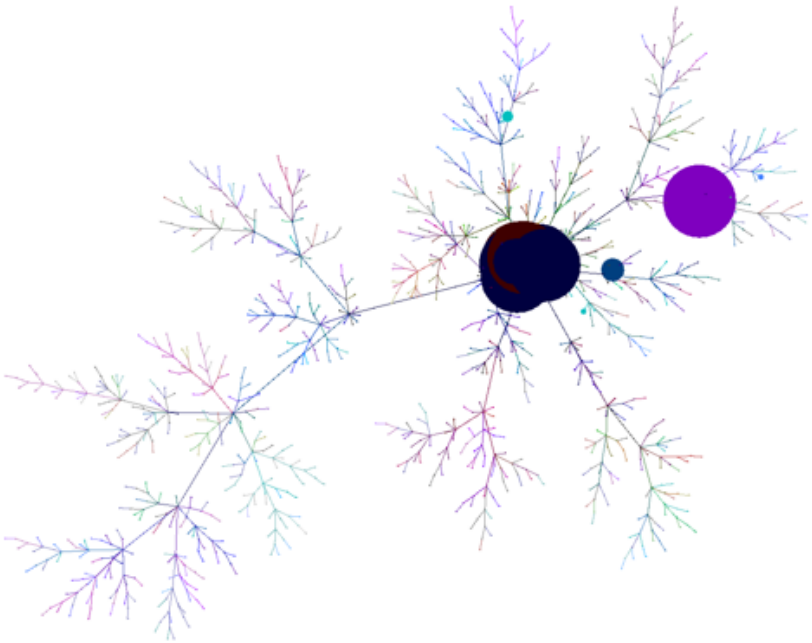

Figura 11: Generación 10 de otra simulación ejecutada usando los mismos parámetros.

Gracias a esto se puede concluir que, efectivamente, el control de mutaciones es importante para mantener estable el crecimiento de una población y, por tanto, el éxito de la especie.

Y finalmente, mientras que este caso de estudio no se centra en la muerte de los individuos y por tanto, no da resultados conclusivos al respecto de este mecanismo del programa, sí que es posible concluir que **Speciate** es capaz de simular la reproducción y mutación de organismos de manera coherente y permite llegar a conclusiones útiles.

5

Conclusiones y Trabajos futuros

Este Trabajo de Fin de Grado ha consistido en el diseño y la implementación de **Speciate**, una herramienta capaz de simular la mutabilidad y reproducción de organismos. A continuación se explican las metas del proceso de aprendizaje que fueron cumplidas a lo largo del desarrollo del proyecto.

- Estudio y evaluación de diferentes simuladores lenguajes de programación, librerías y entornos de desarrollo integrados como GraphStream y Graphics2D de Java o WindowBuilder de Eclipse.
- Estudio del paradigma de programación dirigida por eventos y la concurrencia propias de aplicaciones con interfaz gráfica y que realizan tareas pesadas como es la simulación.
- Estudio de aspectos de ingeniería del software como el patrón de diseño Modelo-Vista-Controlador, que permite organizar el código de manera que sea escalable.
- Estudio de conceptos teóricos sobre los organismos primitivos, su comportamiento y el ambiente en el que vivían.
- Estudio de conceptos teóricos acerca de la reproducción de los procariotas, sus funciones más básicas y la importancia de los plásmidos.
- Estudio de conocimientos de genómica y biología celular.
- Diseño, desarrollo y validación de resultados guiados por los requisitos marcados inicialmente, de los cuales se cumplió una gran parte.

Durante la elaboración del proyecto, gracias a la metodología iterativa e incremental, se han podido construir, en primer lugar, las partes más fundamentales de la aplicación, e ir agregando funcionalidad sobre estas. Gracias a dicha metodología, fue posible discernir los requisitos asequibles de los ambiciosos, y algunas ideas iniciales del proyecto fueron dejadas de lado.

5.1. Trabajos futuros

En esta sección se explican algunas adiciones que pueden ser agregadas a la aplicación resultante de este proyecto para mejorar su funcionalidad. Como se menciona anteriormente, algunos requisitos propuestos inicialmente exigían más tiempo del disponible para poder ser implementados, véase los requisitos funcionales RF19, RF20 y RF21, quedando pendientes como trabajo futuro. También se podrían agregar otras adiciones a la aplicación final que no fueron contempladas inicialmente.

5.1.1. Mejoras de la interfaz gráfica y el código general

La interfaz de usuario de **Speciate** permite alterar las condiciones iniciales de las simulaciones y las cualidades de las mutaciones que afectarán a toda la simulación. Sin embargo, para simplificar el funcionamiento de la interfaz, algunos elementos del algoritmo se mantienen por defecto, como la probabilidad de mutar inicial de una población sin mutaciones y la proporción que sigue el crecimiento de las poblaciones. Hay margen para mejorar la interfaz, permitiendo al usuario acceder a más parámetros del programa. Además, la entrada de organismos base siempre tiene forma de pulso, cuando esto se podría modificar, haciendo posibles entradas diferentes (polinómicas, sinusoidales, logarítmicas, etc).

En cuanto a las salidas del programa, algunos resultados requieren más tiempo y recursos para ser generados. Por ejemplo, almacenar los grafos de nodos de relaciones de parentesco es un proceso que requiere varios segundos por imagen para generaciones con muchas poblaciones, debido a que cada imagen debe generarse correctamente antes de ser almacenada. Esto provoca que la simulación al completo se pause durante la generación de cada imagen, y tarde más tiempo en finalizar. Una posible mejora sería implementar un parámetro de la interfaz mediante el cual el usuario pueda decidir si quiere o no generar estos grafos, o si prioriza

generar una gráfica de poblaciones con la mayor rapidez posible, algo especialmente útil para simulaciones con un alto número de iteraciones.

5.1.2. Diseño e implementación de nuevos algoritmos

El código de **Speciate** es adaptable para implementar multitud de funciones, como algoritmos que simulan simbiosis y conflictos entre las poblaciones. Las mutaciones que **Speciate** modela aportan cualidades medibles con valores numéricos a las distintas especies. Sería necesario agregar una función al programa que manejase mutaciones de cualidades especiales. Los métodos con los que se maneja esta función se encuentran en proceso de elaboración, pero precisan superar periodos de pruebas y mejoras, que debido a la falta de tiempo solo se encuentran en versiones preliminares futuras de la herramienta.

Speciate también se podría mejorar con la implementación de un algoritmo para alterar la cantidad de alimento en las simulaciones. El modelo está diseñado para que cada población tenga una reserva de alimento propia, esto facilita los cálculos y la implementación de los requisitos anteriores. Sin embargo, en caso de que se implementase un sistema de mutaciones de cualidades especiales, el modelo podría agrupar múltiples poblaciones que compartan una mutación de emplear un tipo de alimento, y se podrían alterar las reservas de cada una de estas poblaciones al mismo tiempo adecuadamente.

5.1.3. Validación de nuevos casos de estudio

Finalmente, como se ha visto en la Sección 4, se ha utilizado el simulador para contrastar el aporte de la capacidad de control de mutación, frente a la capacidad de reproducción sin más. En los resultados se observa que, desde el punto de vista evolutivo de la selección natural, es necesario un mínimo de control, siendo este una característica correspondiente a las especies más aptas.

Speciate podría aplicarse a otros casos de estudio, al ser capaz de simular características de las colonias de organismos celulares. Permite reproducir la evolución de unos organismos en otros, estudiar su adaptabilidad, estudiar el origen de la vida y las condiciones de vida primitivas, etc.

Referencias

Atitey, K. (2022). DEGBOE: Discrete time Evolution modeling of Gene mutation through Bayesian inference using qualitative Observation of mutation Events. *Journal of biomedical informatics*, *134*, 104197.

Cerca, J. (2023). Understanding natural selection and similarity: Convergent, parallel and repeated evolution. *Molecular Ecology*, *32*(20), 5451-5462.

Darwin, C. (1859). El origen de las especies. *Estados Unidos: Editorial Charles Darwing*.

Eclipse. (2024a). Eclipse [Accedido: 2024-06-21]. https://help.eclipse.org/2024-06/index.jsp

Eclipse. (2024b). Eclipse [Accedido: 2024-06-21]. https://eclipse.dev/windowbuilder/

Guerrero, R., & Berlanga, M. (2001). La "inmortalidad" procariótica y la tenacidad de la vida. *Actualidad Sem*, *32*, 16-23.

isee systems, inc. (2024). Stella Architect [Accedido: 2024-06-18]. https://www.iseesystems.com/store/products/stella-architect.aspx

Julien Baudy, Antoine Dutot, Yoann Pigné, Guilhelm Savin. (2024). GraphStream [Accedido: 2024-06-27]. https://graphstream-project.org/

Kiraz, Y., Adan, A., Kartal Yandim, M., & Baran, Y. (2016). Major apoptotic mechanisms and genes involved in apoptosis. *Tumor Biology*, *37*, 8471-8486.

Maciej Komosinski, A. A. (2009). *Artificial life models in software*. Springer Science & Business Media.

MathWorks. (2024a). MATLAB [Accedido: 2024-06-17]. https://es.mathworks.com/products/matlab.html

MathWorks. (2024b). SimBiology [Accedido: 2024-06-27].

Matthias Ruth, B. H. (1997). *Modeling dynamic biological systems*. Springer.

Mikhailovsky, G. E., & Gordon, R. (2021). LUCA to LECA, the Lucacene: A model for the gigayear delay from the first prokaryote to eukaryogenesis. *Biosystems*, *205*, 104415.

Oparin, A. I. (1989). *El origen de la vida*. Ediciones Akal. https://books.google.es/books?id=lNZPbLgd9OIC

Oracle. (2024). What is Java? [Accedido: 2024-06-18]. https://www.java.com/es/download/help/whatis_java.html

Pinzón Núñez, S., Rodríguez Guerrero, R., & Vanegas, C. A. (2019). *Java y el patrón Modelo-Vista-Controlador (mvc)*. Editorial Universidad Distrital Francisco José de Caldas.

WEKA. (2024). WEKA [Accedido: 2024-06-18]. https://www.weka.io/

Apéndice A
Manual de Instalación y Uso

En esta sección se describen los pasos que el usuario debe seguir antes de ejecutar una simulación con **Speciate**. Se explica por un lado la configuración previa que el usuario necesita en su máquina junto con la manera de instalar correctamente la aplicación. Y por otro lado se detallan los pasos a seguir una vez dentro de **Speciate** para utilizar la aplicación correctamente, cómo se manejan los parámetros y qué efectos tienen en los resultados.

A.1. Manual de Instalación

En este apartado se explican los requisitos previos y los pasos a seguir para ejecutar la aplicación de **Speciate** en una máquina del usuario.

La sencillez de la aplicación permite que la máquina del usuario requiera únicamente dos descargas.

- Versión correcta de Java. Para que la aplicación funcione, la máquina debe tener instalado Java, concretamente la versión: 8, actualización 441, que se puede descargar en el enlace siguiente:

  https://www.java.com/download/java8_update.jsp

- Speciate. También es necesaria la propia aplicación, el contenido de la cual se puede descargar desde la carpeta del siguiente repositorio de GitHub llamado "VikYosava\Speciate":

  https://github.com/VikYosava/Speciate

  A este enlace también se puede acceder navegando hasta el repositorio con ese mismo nombre a través del código QR de la Imágen 12.

  Para ejecutar **Speciate** es necesario descargar el contenido del repositorio en una carpeta de la máquina del usuario. Una vez completado este paso, al ejecutar el archivo

.jar runnable nombrado “Speciate.jar”, se encenderá la aplicación. Los resultados de las simulaciones que genere el usuario se almacenarán dentro de la carpeta en la que esté almacenado todo el contenido de la aplicación bajo el nombre que se indique. Es recomendable que el usuario cree un acceso directo a partir de “Speciate.jar” para mayor comodidad.

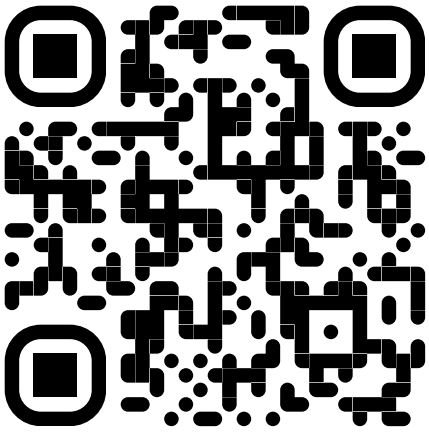

Figura 12: Código QR que dirige al GitHub de VikYosava

A.2. Manual de Uso

A continuación se explicarán en detalle las distintas pantallas y parámetros a los que tendrá acceso el usuario, qué influencia tienen en el simulador, qué formato deben tener y cómo utilizarlos.

A.2.1. Pantalla de “Configuración Inicial”

La primera pantalla que se abre al iniciar el programa, tal y como se explica en la Sección 3, es la pantalla de “Configuración Inicial”.

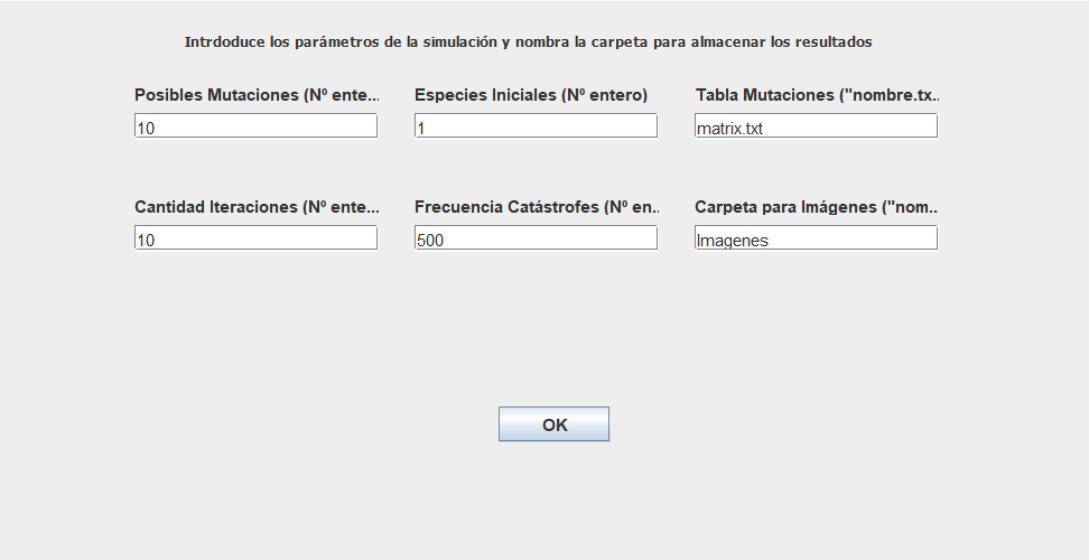

Figura 13: Pantalla de “Configuración Inicial”

En esta pantalla el usuario puede modificar los primeros parámetros necesarios para ejecutar una simulación. De no ser modificados, estos tendrían los valores por defecto que se observan en la Imágen 13.

Nombre del parámetro	Formato	Descripción y qué afecta
Posibles Mutaciones	Número entero	Tamaño máximo del código genético. Debe coincidir con el número de columnas del archivo que se introduzca a través de "Tabla Mutaciones". Determinará el número de columnas de la tabla que aparecerá en la pantalla de "Valores por Especie".
Especies Iniciales	Número entero	Número de especies que van a formar la generación 0. Determinará el número de filas de la tabla que aparecerá en la pantalla de "Valores por Especie".
Tabla Mutaciones	Texto seguido de ".txt"	Nombre del archivo .txt que contiene los valores de las características que aporta cada mutación. El contenido se explica más a fondo a continuación. Afecta a toda la simulación.
Cantidad Iteraciones	Número entero	Número de generaciones que se van a simular. Determina la cantidad de veces que se reproducen las distintas poblaciones, el tiempo discreto durante el que se ejecuta la simulación al completo.
Tabla Mutaciones	Texto seguido de ".txt"	Nombre del archivo .txt que contiene los valores de las características que aporta cada mutación. El contenido se explica más a fondo a continuación.
Frecuencia Catástrofes	Número entero	Valor que indica cada cuántas generaciones habrá una extinción masiva de todos los organismos que compartan un gen específico. Si es mayor que la cantidad de iteraciones, no habrá catástrofes.
Carpeta para Imágenes	Texto	Nombre de la carpeta en la que se almacenarán los resultados en forma de grafo de cada generación simulada, además de un archivo .arff conteniendo las especies de la última generación y sus características.

Tabla 7: Parámetros modificables en la pantalla de "Configuración Inicial"

Como se explica en la Tabla 7, es necesario un unico fichero de entrada, un documento de texto ".txt" cuyo nombre fichero debe indicarse correctamente en el parámetro de entrada "Tabla Mutaciones" de la pantalla "Configuración Inicial". Este fichero debe almacenarse dentro de la carpeta de la aplicación y debe seguir el siguiente formato de valores enteros, donde los números corresponden con los valores introducidos por pantalla que se explican en la tabla anterior:

-0.7779	-0.1004	0.2172	0.9447	-0.5897	-0.4709	0.4300	0.8286	-0.7517	-0.1184
0.3567	0.9182	0.1893	0.3567	0.3437	0.7242	1.6275	0.3082	0.3567	0.5694
-0.8880	-0.9182	-0.1893	-0.4610	0.3437	0.7242	-1.6275	-0.3082	-0.3567	-0.5694
0.7214	0.5933	0.2548	0.2355	0.9824	-0.0978	0.9232	-0.8414	0.7770	0.5372

Tabla 8: Ejemplo de datos en formato de tabla

Donde, como se explica en la Tabla 7, para que **Speciate** haga un uso correcto de la tabla introducida, el número de columnas debe corresponderse con el parámetro "Posibles Mutaciones" de la pantalla de "Configuración Inicial" y el número de filas debe ser 4.

Cada fila se corresponde respectivamente con uno de los siguientes valores de las mutaciones de la simulación:

Fila	Nombre	Tipo	Descripción
Primera	Probabilidad de mutación	Float	El sumatorio de esta fila más 0.6 da el porcentaje de que una especie mute al reproducirse o no.
Segunda	Capacidad de alimento	Float	El sumatorio de esta fila más 0.3 permite calcular la cantidad de alimento que obtiene cada uno de los seres vivos con capacidad de movimiento de una población durante una generación.
Tercera	Coste de alimento	Float	El sumatorio de esta fila más 0.25 determina la cantidad de alimento de la reserva que gasta cada individuo de una población en cada generación al moverse y al reproducirse.
Cuarta	Capacidad de movimiento	Float	El sumatorio de esta fila más 0.6 determina el porcentaje de una población que podrá buscar alimento e interactuar con otras especies.

Tabla 9: Valores de las capacidades numéricas que proporciona cada mutación editables a través de Tabla Mutaciones.

En caso de que el nombre del fichero de la tabla sea introducido incorrectamente, o de que la tabla no cumpla con el formato adecuado, el modelo ignora este parámetro y genera una tabla de mutaciones totalmente aleatoria, siguiendo un modelo probabilístico basado en campanas de Gauss.

Tras editar toda la información que el usuario desee, podrá acceder a la siguiente pantalla de la aplicación pulsando el botón “Ok”.

A.2.2. Pantalla de “Valores por Especie”

Después de la pantalla de “Configuración Inicial”, el usuario puede acceder a la pantalla de “Valores por Especie”. En caso de querer volver a la pantalla de “Configuración Inicial”, puede hacerlo a través del botón “Cancelar”, mediante el cual accederá a dicha pantalla reiniciada, con los cuadros de texto conteniendo los valores por defecto de la aplicación.

En la pantalla de “Valores por Especie”, el usuario puede observar 2 tablas:

- Tabla de mutaciones activas, que contiene un número de columnas igual al número de Posibles Mutaciones introducidas en la pantalla anterior, y un número de filas igual al número de Especies Iniciales, como se puede observar en el ejemplo de la Imágen 14, donde el número es de 40 Posibles Mutaciones por 100 Especies Iniciales.

- Tabla de Cantidad de Individuos (CI) y Cantidad de Alimento (CA). Esta tabla tendrá un número de filas igual a la tabla de mutaciones activas y solo dos columnas correspondientes a la CI y la CA iniciales de cada población inicial.

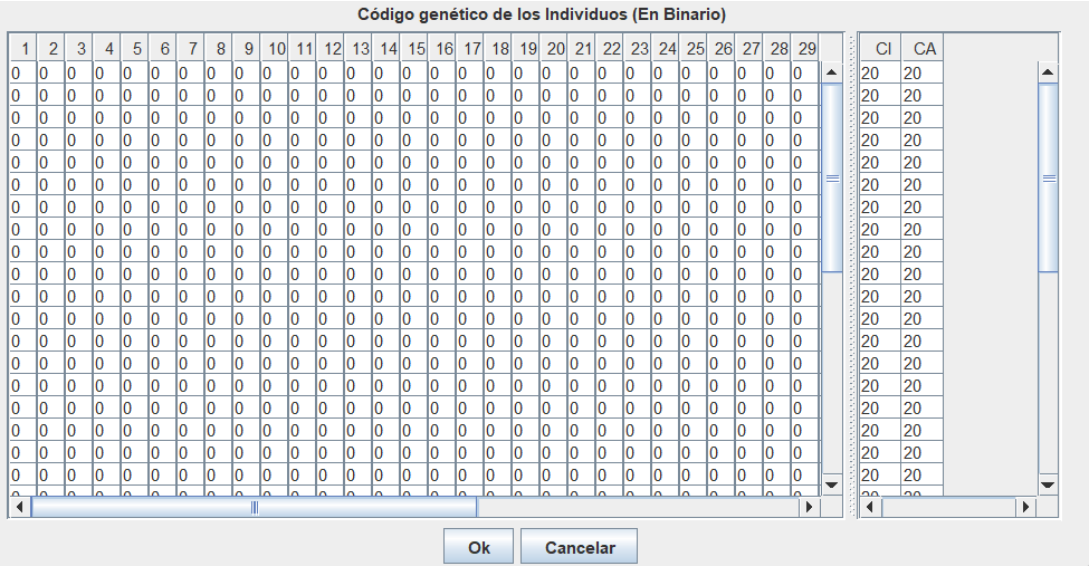

Figura 14: Pantalla de "Valores por Especie".

La tabla de mutaciones activas determina las mutaciones de la tabla introducida por Tabla Mutaciones, de la pantalla anterior, que estarán activas en las especies iniciales de la simulación. Se pueden modificar las mutaciones activas haciendo click izquierdo para sumar 1 y click derecho para restar 1 sobre una casilla de la tabla, como se observa en la Tabla 15. También se puede modificar dando doble click y escribiendo. Las simulaciones están planteadas para funcionar con números binarios, pero si el usuario lo desea, para mayor flexibilidad, se puede introducir cualquier número entero positivo o negativo. De esta manera, el programa calcula que la especie inicial tuviese N veces la mutación correspondiente, y las cualidades se acumulan. Las especies descendientes no serán capaces de desarrollar mutaciones que se rijan por otro sistema que no sea el binario.

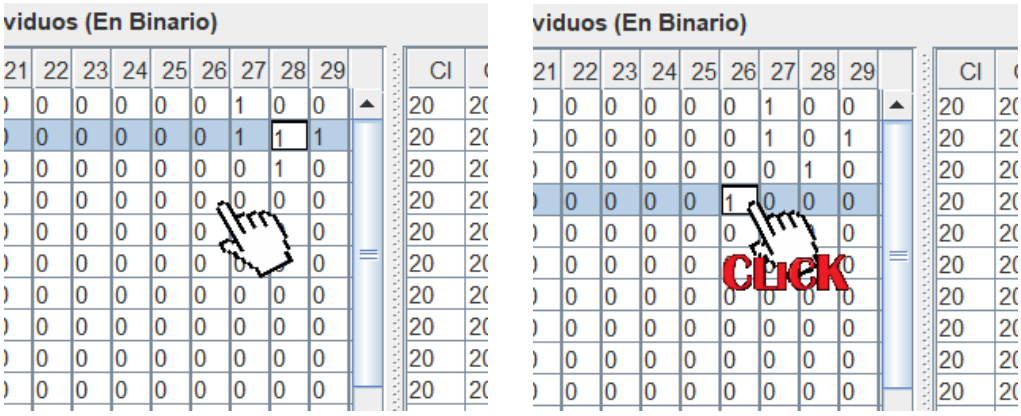

Figura 15: Ejemplo de edición de tabla de mutaciones activas

La tabla de CI y CA se puede editar haciendo doble click sobre la casilla que se pretenda modificar y escribiendo los valores por teclado, como se observa en el ejemplo de la Tabla 16. Los valores introducidos deben ser números naturales.

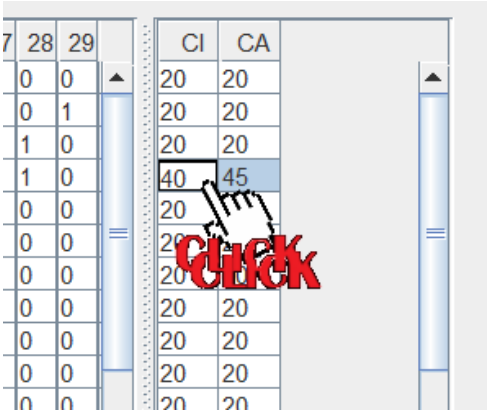

Figura 16: Ejemplo de edición de tabla de CI y CA

Tras modificar todos los datos según sus preferencias, el usuario puede pulsar el botón "Ok", ejecutando la simulación y generando los resultados.

A.2.3. Resultados

Tras ejecutar la simulación, se generarán 3 tipos de salidas diferentes:

- Pantalla de gráfico de progreso de poblaciones. En esta pantalla se muestra el progreso de cada población generada a lo largo de la simulación en forma de gráfico bidimensional de líneas, donde cada una de las líneas es una población. El eje Y indica el número

de individuos que forman cada población en un momento discreto determinado, el eje X indica dicho momento del tiempo o generación. Esta pantalla aparecerá en una ventana nueva generada, podemos observar un ejemplo en la Imágen 10 de la Sección 4. Colocando el ratón sobre una de las líneas, se puede ver las posiciones de la matriz introducida por Tabla Mutaciones (archivo .txt) que se encuentran activas en la especie correspondiente a la línea señalada, además de la generación en la que la población fue generada.

- Grafos de relaciones entre especies. En la carpeta indicada en Carpeta para Imágenes se almacena en formato .png un grafo por cada generación que fue simulada. En estos grafos el usuario puede comprobar el crecimiento de las poblaciones y su ramificación y conexiones entre especies engendradas y engendradoras, como se observa en las Imágenes 8 y 9 de la Sección 4.

- Archivo .arff. Un archivo de texto organizado conteniendo la información a cerca del número de individuos, el código genético y las características de cada población generada durante la última generación. Estos datos se encuentran adaptados para ser analizables con el entorno de análisis Weka y se almacenará en la misma carpeta que los grafos de relaciones entre especies.