



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Sibling Rewiring 2.0

Documentación Técnica



Presentado por Rubén Arasti Blanco
en la Universidad de Burgos — 9 de julio de 2024

Tutores: Dr. José Manuel Galán Ordax
y Dra. Virginia Ahedo García

Índice General

Índice General	1
Índice de Ilustraciones	3
Índice de Tablas	4
Apéndice A. Plan de Proyecto Software	5
A.1 Introducción	5
A.2 Planificación temporal.....	5
Pasos iniciales	5
Sprint 1 (24/04/2024 – 1/05/2024).....	5
Sprint 2 (1/05/2024 – 8/05/2024).....	6
Sprint 3 (8/05/2024 – 15/05/2024).....	7
Sprint 4 (15/05/2024 – 22/05/2024).....	7
Sprint 5 (22/05/2024 – 05/06/2024).....	7
Sprint 6 (05/06/2024 – 19/06/2024).....	8
Sprint 7 (05/06/2024 – 19/06/2024).....	9
Sprint 8 (27/06/2024 – 08/07/2024).....	10
A.3 Estudio de viabilidad.....	11
Viabilidad económica.....	11
Viabilidad legal	13
Apéndice B. Especificación de Requisitos	16
B.1. Introducción.....	16
B.2. Objetivos generales.....	16
B.3. Catálogo de requisitos.....	16
Requisitos funcionales	16
Requisitos no funcionales.....	17
B.4. Especificación de requisitos	13
Apéndice C. Especificación de diseño.....	17
C.1. Introducción	17

C.2. Diseño de datos	17
C.3. Diseño procedimental	18
C.4. Diseño arquitectónico	18
Apéndice D. Documentación técnica de programación	20
D.1. Introducción.....	20
D.2. Estructura de directorios	20
D.3. Manual del programador.....	22
Instalación y ejecución en Windows	22
Instalación y ejecución en Linux.....	23
Instalación y ejecución con Docker	24
D.5. Pruebas del sistema	26
Apéndice E. Documentación de usuario	29
E.1. Introducción.....	29
E.2. Requisitos de usuarios.....	29
E.3. Instalación.....	29
Importar máquina virtual	30
Ejecutar aplicación	30
Reenvío de puertos.....	30
E.4. Manual del usuario.....	31
Apéndice F. Anexo de sostenibilización curricular	38
F.1 Introducción.....	38
F.2 Objetivo número 3: Salud y Bienestar	38
F.2 Objetivo número 4: Educación de calidad	39
Bibliografía	40

Índice de Ilustraciones

Ilustración 1: Diagrama de casos de uso	13
Ilustración 2: Diagrama de secuencia	18
Ilustración 3 [7]: pantalla de Inicio	31
Ilustración 4 [7]: pantalla de Login	32
Ilustración 5 [7]: pantalla de Registro	33
Ilustración 6 [7]: pantalla de Inicio de sesión.....	33
Ilustración 7 [7]: pantalla de Selección del tipo de datos de entrada	34
Ilustración 8 [7]: pantalla de Generación de redes aleatorias	34
Ilustración 9 [7]: Subida de ficheros.....	35
Ilustración 10: pantalla de Selección de algoritmo	35
Ilustración 11: pantalla de Selección de opciones del algoritmo genético	36
Ilustración 12: pantalla de Selección de parámetros del algoritmo genético	36
Ilustración 13: Resultados del AG, descargas	37
Ilustración 14: Resultados del AG, gráficas	37

Índice de Tablas

Tabla 1: Costes de personal	11
Tabla 2: Costes de hardware	12
Tabla 3: Costes de software	12
Tabla 4: Costes varios	12
Tabla 5: Costes totales	13
Tabla 6: tabla de licencias.....	13
Tabla 7: Caso de uso 1	14
Tabla 8: Caso de uso 2	14
Tabla 9: Caso de uso 3	15
Tabla 10: Caso de uso 4	15
Tabla 11: Caso de uso 5	16

Apéndice A. Plan de Proyecto Software

A.1 Introducción

El plan se divide en dos secciones: planificación temporal y estudio de viabilidad económica y legal.

A.2 Planificación temporal

Con el objetivo de asegurar una ejecución ordenada y coherente de las diversas fases y tareas del proyecto, se ha decidido adoptar SCRUM como metodología ágil de trabajo basada en sprints. En este contexto, el proyecto se ha estructurado en sprints de aproximadamente una semana de duración cada uno.

Pasos iniciales

Durante los primeros meses del año, se avanzó muy lentamente en el proyecto y no se utilizó una metodología de trabajo. No obstante, se realizaron tareas relevantes para el desarrollo del proyecto que serán mencionadas en este apartado.

En la primera reunión se habló de los aspectos más relevantes acerca del proyecto. Estos son la importancia de desplegar la aplicación web, las metodologías que debía usar y la elección de las herramientas para llevar a cabo el proyecto.

En la segunda reunión, se informó a los tutores de qué herramientas iba a utilizar.

Tras varios intentos de ejecutar la anterior aplicación sin éxito, se acordó otra reunión en la que se proporcionó una máquina virtual de la anterior aplicación. Gracias a esto, se pudieron entender algunos pasos que no se especificaban en el anterior trabajo sobre las operaciones necesarias con MySQL y se consiguió ejecutar la anterior aplicación.

Sprint 1 (24/04/2024 – 1/05/2024)

En este sprint se han realizado tareas iniciales de investigación e instalación de algunos programas.

Las tareas programadas fueron las siguientes:

#1 Instalar Zotero.

#2 Entender la anterior aplicación. Esta tarea se dividía en leer la documentación de la anterior aplicación, entender el código de la anterior aplicación y documentar el problema a resolver.

#3 Investigación sobre los algoritmos genéticos multiobjetivo. Esta tarea se dividía en leer un artículo que me proporcionaron los tutores y documentar acerca del algoritmo.

La instalación de Zotero, su extensión para Chrome y un plugin para MS Word llevó aproximadamente media hora. Sumando otra media hora para entender su funcionamiento básico, esta tarea se completó en un total de 1 hora.

La lectura de la documentación de la anterior aplicación tomó 3 horas.

La comprensión y documentación del código de la anterior aplicación requirió 30 horas. En esta tarea, no sólo documenté el problema a resolver sino también los archivos principales del código y la resolución del problema.

No se logró iniciar la última tarea, por lo que se trasladó al siguiente sprint.

Retrospectiva del sprint. Se concluye que se sobreestimó el tiempo necesario para comprender el trabajo anterior. Además, no se registraron las estimaciones de tiempo ni los puntos de historia de cada tarea

Sprint 2 (1/05/2024 – 8/05/2024)

Este sprint se ha centrado en crear una primera implementación del algoritmo genético.

Las tareas programadas fueron las siguientes:

#5 Documentar el sprint 1. Se estimó un tiempo de 1 hora y se terminó en 45 minutos.

#3 Investigación sobre los algoritmos genéticos multiobjetivo. Se dividió en 2 subtareas: leer un artículo acerca de los algoritmos genéticos multiobjetivo y documentar acerca de este. Se estimó un tiempo de 12 horas en total. La lectura detallada del artículo tomó 6 horas. No se logró iniciar la documentación, por lo que esta tarea se trasladó al siguiente sprint.

#6 Primera implementación del algoritmo genético. Se dividió en 2 subtareas: implementar el código y crear la interfaz en la web. Se estimó un tiempo de 25 horas en total. Completar el código llevó 15 horas y crear la interfaz 3 horas. No obstante, tras una reunión con los tutores, se observó que el algoritmo implementado presentaba numerosos fallos, por lo que se decidió implementar uno nuevo para la siguiente semana. Aun así, el esqueleto principal del código está montado y la representación del genotipo es bueno, por lo que se puede reutilizar para la siguiente implementación.

#4 Corregir la función descargar archivo. Se estimó un tiempo de 3 horas. Esta tarea no se empezó por falta de tiempo invertido. Pasa al siguiente sprint.

Retrospectiva del sprint. Se sobreestimó el tiempo requerido para las tareas, ya que se completaron en menos tiempo del previsto. No se invirtió el tiempo necesario en el proyecto, lo que se reflejó en la cantidad de tareas sin completar.

Sprint 3 (8/05/2024 – 15/05/2024)

Este sprint se ha centrado en implementar una segunda versión del algoritmo genético.

Las tareas programadas fueron las siguientes:

#7 Documentar el sprint 2. Se estimó un tiempo de 30 minutos y se terminó en 40 minutos.

#3 Documentar acerca de los algoritmos multiobjetivo. Se estimó un tiempo de 6 horas. Tomó 3 horas documentar el frente de Pareto y la definición del problema a resolver.

#8 Segunda implementación al algoritmo genético. Se debía mostrar en una gráfica el frente de Pareto e implementar una interfaz para la web que permitiese escoger entre las distintas soluciones del frente. Se estimó un tiempo de 18 horas y se terminó en 20 horas.

#9 Implementar todos los fitness encontrados a la gráfica. Se estimó un tiempo de 1 hora y se completó en 50 minutos.

También se programaron para este sprint la función de creación de un archivo final y la de descarga del archivo pero no dio tiempo a empezarlas.

Retrospectiva del sprint. Los problemas ocasionados por el funcionamiento de la librería DEAP han retrasado otras tareas menos prioritarias.

Sprint 4 (15/05/2024 – 22/05/2024)

Este sprint se ha centrado en migrar el algoritmo genético de la librería deap a pymoo.

Las tareas programadas fueron las siguientes:

#12 Documentar el sprint 3. Se estimó un tiempo de 30 minutos y se completó en 40 minutos.

#13 Cambio de biblioteca de deap a pymoo. Se estimó un tiempo de 6 horas y se completó en 10 horas. No se logró replicar completamente el funcionamiento del algoritmo con la biblioteca DEAP.

También se programaron para este sprint la función de creación de un archivo final y la de descarga del archivo pero no se llegaron a comenzar.

Retrospectiva del sprint. Se invirtió poco tiempo en el proyecto durante este sprint, lo cual se reflejó en la cantidad de tareas completadas. Las tareas necesarias para el archivo final y para la descarga no son tan prioritarias como el buen funcionamiento del algoritmo, pero no lo he previsto al principio del sprint y he tenido que crear tareas a mitad de este.

Sprint 5 (22/05/2024 – 05/06/2024)

Este sprint se ha centrado en realizar correcciones sobre el algoritmo genético y añadir la funcionalidad de descarga de varios archivos a partir de una solución.

Las tareas programadas fueron las siguientes:

#14 Documentar el sprint 4. Se estimó un tiempo de 30 minutos y se completó en 10 minutos.

#16 Selección de parámetros para el algoritmo genético. Se estimó un tiempo de 20 horas y se completó en 15 horas. En esta tarea no sólo se seleccionaron los parámetros por defecto del algoritmo genético, sino que también se cambió el algoritmo para que no devolviese la misma solución siempre.

#11 Implementar una función para crear el archivo final. Se estimó un tiempo de 2 horas y se completó en 5 horas. Surgió un problema con la representación en imagen de NetworkX y los auto enlaces.

#18 Implementar descargas para el algoritmo genético. Se estimó un tiempo de 2 horas y se completó en el tiempo previsto. Se codificaron los archivos para su descarga a través de la página web.

También se programaron las tareas de documentar el algoritmo genético y la creación de test pero no llegué a empezarlas.

Retrospectiva del sprint. Este sprint se ha extendido una semana más porque al acabar la primera semana no se realizó ninguna tarea. Los nombres dados a las tareas no han sido del todo adecuados y eso ha provocado confusión sobre cómo abordarlas, retrasando su inicio.

Sprint 6 (05/06/2024 – 19/06/2024)

Este sprint se centró en mejorar la solución obtenida y los archivos resultado.

Las tareas programadas fueron las siguientes:

#Documentar sprint 5. Se estimó un tiempo de 1 hora y se completó en 30 minutos.

#23 Añadir la decisión de parámetros para el usuario. Se estimó un tiempo de 1 hora y se completó en 4 horas.

#24 Bug creación red aleatoria. Se estimó un tiempo de 1 hora y 30 minutos y se completó en 2 horas.

#20 Mejorar descargas para el algoritmo genético. Se estimó un tiempo de 7 horas y se invirtieron 4 horas pero no se completó. Se realizaron algunas mejoras como colorear los componentes en la imagen del grafo.

#21 Heurística para soluciones no factibles. Se estimó un tiempo de 5 horas y se completó en 11 horas. Además de crear la heurística para cambiar una solución no factible a una factible, se corrigió un problema en la creación de la solución. Esto hizo que llevara más tiempo del esperado.

También se programaron tareas de documentación, corrección de bugs e implementación de tests, pero no se llegaron a empezar.

Retrospectiva del sprint. En la mayoría de los sprints realizados hasta el momento, se han dejado tareas sin completar debido a una planificación inadecuada. Para mejorar este aspecto, es fundamental priorizar y programar las tareas más importantes, asegurándose de que puedan ser completadas dentro del período del sprint. Las tareas secundarias deben mantenerse en el producto backlog. Una vez que las tareas principales se hayan completado, las tareas secundarias pueden añadirse al sprint actual.

Sprint 7 (05/06/2024 – 19/06/2024)

Este sprint se centró sobre todo en la documentación del proyecto y en acabar el desarrollo de la aplicación.

Las tareas programadas fueron las siguientes:

#25 Documentar sprint 6. Se estimó un tiempo de 30 minutos y se completó en 45 minutos.

#26 Documentar introducción. Se estimó un tiempo de 3 horas y se completó en 6 horas.

#22 Documentar objetivos principales. Se estimó un tiempo de 3 horas y se completó en 3 horas.

#15 Documentación del algoritmo genético. Se estimó un tiempo de 4 horas y se completó en 40 horas aproximadamente. Esta tarea ha sido considerada como la documentación de los conceptos teóricos y por eso el error entre el tiempo estimado e invertido.

#20 Mejorar descargas para el algoritmo genético. Se estimó un tiempo de 3 horas y se completó en 12 horas. Se añadieron las siguientes mejoras, la leyenda en la imagen del grafo, la asignación de nombres a las soluciones y su corrección cuando son modificadas y la descarga de todos los archivos.

#27 Arreglar un bug con los pares de hermanos en la creación de la red aleatoria. Se estimó un tiempo de 1 hora y se completó en 8 horas. En esta tarea también se crearon archivos de ejemplo nuevos y se añadieron las revisiones a la memoria.

Retrospectiva del sprint. La tarea de documentar los objetivos teóricos fue mal definida y desglosada. Es importante que ninguna tarea individual exceda un total de 24 horas. Además, se observó una planificación deficiente en las tareas de diseño, ya que se introdujeron nuevas funcionalidades no planificadas durante la ejecución de las tareas, lo que complicó su gestión y desarrollo adecuado.

Sprint 8 (27/06/2024 – 08/07/2024)

El último sprint, se centró sobre todo en tareas de documentación, desplegar la aplicación web y la revisión de la calidad del código.

Las tareas programadas fueron las siguientes:

- #29 Documentar sprint 7. Se estimó un tiempo de 1 hora y se completó en 1 hora.
- #30 Desplegar la web. Se estimó un tiempo de 24 horas y se completó en 30 horas aproximadamente. Durante esta tarea se generaron archivos de configuración necesarios para la creación de un Docker, pero el tiempo invertido en estos se añadirá a la tarea #36.
- #38 Revisión de código mediante SonarCloud. Se estimó un tiempo de 24 horas y se completó en 12 horas. Durante esta tarea se invirtieron 5 horas en migrar el algoritmo a la librería pymoo para corregir un bug encontrado con la librería DEAP.
- #36 Creación de máquina virtual y Docker. Se estimó un tiempo de 24 horas y se completó en 15 horas. La creación del Docker tomó 10 horas teniendo en cuenta la corrección de sus fallos. Y la creación de la máquina virtual tomó 5 horas.
- #33 Documentar apéndice B especificación de requisitos. Se estimó un tiempo de 12 horas y se completó en 8 horas.
- #35 Documentar apéndice F anexo de sostenibilización curricular. Se estimó un tiempo de 5 horas y tomó 3 horas y media completarlo.
- #40 Documentar técnicas y herramientas. Se estimó un tiempo de 8 horas y se completó en 6 horas.
- #41 Documentar aspectos relevantes en el desarrollo del proyecto. Se estimó un tiempo de 8 horas y se completó en 9 horas.
- #42 Documentar trabajos relacionados. Se estimó un tiempo de 5 horas y se completó en 2 horas.
- #39 Documentar conclusiones y líneas de trabajo futuras. Se estimó un tiempo de 8 horas y se completó en 6 horas.
- #28 Documentar resumen. Se estimó un tiempo de 1 hora y se completó en 30 minutos.
- #34 Documentar apéndice C especificación de diseño. Se estimó un tiempo de 24 horas y se completó en 4 horas.
- #31 Documentar apéndice D documentación técnica de programación. Se estimó un tiempo de 12 horas y se completó en 4 horas.
- #32 Documentar apéndice A plan de proyecto software. Se estimó un tiempo de 8 horas y se completó en 6 horas.

Retrospectiva final. Durante los ocho sprints, el proyecto se centró en varias tareas clave como la instalación de software, la comprensión de una aplicación existente, y la implementación de algoritmos genéticos multiobjetivo. En los primeros sprints, se realizaron investigaciones y se instaló software esencial, como Zotero, y se comprendió el código de una aplicación anterior. Los sprints intermedios se dedicaron a la implementación y mejora de algoritmos genéticos, incluyendo la migración a nuevas bibliotecas y la creación de una interfaz web. Los últimos sprints se centraron en la documentación exhaustiva del proyecto, la corrección de errores y el despliegue de la aplicación. A lo largo del proyecto, se observó una tendencia a subestimar el tiempo necesario para completar tareas, lo que llevó a la necesidad de trasladar varias tareas a sprints posteriores y ajustar la planificación. En total, se invirtieron 320 horas en completar las tareas a lo largo de los sprints.

A.3 Estudio de viabilidad

Viabilidad económica

En el apartado siguiente se realiza un análisis del coste que supondría la realización de este proyecto dentro de un contexto laboral real. Este proyecto, aun habiéndose realizado a lo largo de 3 meses, se podría condensar en 2 de trabajo a tiempo completo, ya que la cantidad de trabajo en horas es de 320.

Costes de personal:

Para calcular los costes de personal, se consideran los siguientes elementos: el salario neto mensual, las deducciones fiscales (IRPF), y las contribuciones a la Seguridad Social. Estos factores se reflejan en la siguiente tabla:

Tabla 1: Costes de personal

Concepto	Coste
Salario neto mensual	1.782,72€
IRPF (17,22%) [1]	413,28€
Seguridad Social - Desempleo (Contrato duración determinada) (8,3%) [1]	199,2€
Seguridad Social - FOGASA (0,2%) [2]	4,8€
Salario bruto mensual (programador full stack [3])	2400€
Total de 2 meses	4800€

Costes de hardware:

El único coste de hardware viene del ordenador portátil utilizado, y se supone que este se amortiza a lo largo de 5 años y se utiliza durante 2 meses.

Tabla 2: Costes de hardware

Concepto	Coste	Amortizado
Ordenador portátil	700€	23,32€

Costes de software:

El sistema operativo viene preinstalado en el ordenador, la licencia de Microsoft 365 se puede obtener a distintos precios según la duración, programas y Nº de usuarios, pero la individual se puede adquirir por 7€ al mes.

Tabla 3: Costes de software

Concepto	Coste
Microsoft 365 Personal	14€

Costes varios:

Otros gastos incluyen consumo eléctrico, internet, el uso de la plataforma Heroku para el despliegue de la aplicación, y el alquiler de una oficina:

Tabla 4: Costes varios

Concepto	Coste
Consumo eléctrico	80€
Internet [4]	64€
Uso de Heroku	14€
Alquiler de oficina	360€
Total	518€

Costes totales:

La suma de todos los costes se detalla en la siguiente tabla:

Tabla 5: Costes totales

Concepto	Coste
Costes de personal	4800€
Costes de hardware	23,32€
Costes de software	14€
Costes varios	518€
Total	5355,32€

Beneficios:

El proyecto es de interés para las escuelas, así como para el bien de la salud pública en general, por lo tanto si se pretendiera obtener beneficios del desarrollo de esta aplicación, se podría cobrar a los centros por su uso, también se podría buscar financiación pública, ya sean nacionales o europeas, teniendo en cuenta que el uso de la aplicación se puede enmarcar en una investigación sobre la salud pública.

Viabilidad legal

En este apartado se mostrará una tabla con las herramientas utilizadas y sus licencias para luego decidir el tipo de licencia que tendrá el proyecto.

Tabla 6: tabla de licencias

Herramienta	Uso	Licencia
Docker	Plataforma de contenedorización	Apache 2.0
Docker Compose	Definir y ejecutar aplicaciones multi-contenedor	Apache 2.0
Draw.io	Herramienta de diagramación	Apache 2.0
Flask	Framework web para Python	BSD 3-Clause
Flask-Bootstrap	Integración de Bootstrap para Flask	BSD 3-Clause

Fuente EB Garamond	Fuente tipográfica	Open Font License (OFL)
Gedit	Editor de texto	GPL-2.0 o posterior
Gephi	Análisis y visualización de redes	GPL-3.0
Git	Sistema de control de versiones	GPL-2.0
GitHub	Plataforma de alojamiento para desarrollo de software	Propietaria (planes gratuitos disponibles)
Gunicorn	Servidor HTTP WSGI para Python	MIT
JawsDB MySQL	Base de datos MySQL como servicio	Propietaria
Matplotlib	Biblioteca de gráficos para Python	Licencia PSF (Python Software Foundation)
Microsoft Word	Software de procesamiento de textos	Propietaria
MySQL	Sistema de gestión de bases de datos relacional	GPL-2.0
Mysql-connector	Conector de base de datos MySQL para Python	GPL-2.0
NetworkX	Biblioteca de Python para redes complejas	BSD 3-Clause
Numpy	Biblioteca de computación numérica para Python	BSD 3-Clause
Pymoo	Framework de optimización multi-objetivo	MIT
Python	Lenguaje de programación	Licencia PSF
SonarCloud	Calidad continua y seguridad del código	Propietaria (planes gratuitos disponibles)
Trello	Herramienta de gestión de proyectos	Propietaria (planes gratuitos disponibles)
Visual Studio Code	Editor de código	MIT
Zotero	Software de gestión de referencias	AGPL-3.0

La licencia del MIT es compatible, además de ser la utilizada en el trabajo predecesor de este, sus permisos y obligaciones son:

Permisos

La licencia MIT permite el uso libre del software para cualquier propósito, incluyendo usos comerciales y personales. Los usuarios pueden distribuir copias del software, ya sea en su forma original o modificada, así como modificar el software para adaptarlo a sus necesidades. También se permite la fusión del software con otros proyectos, la sublicencia y la venta del software como parte de otro proyecto.

Obligaciones

Los usuarios deben incluir una copia de la licencia original del MIT en cualquier distribución del software, manteniendo las declaraciones de derechos de autor originales en todas las copias del software. Además, deben incluir el aviso de derechos de autor y la renuncia de garantía en todas las copias o partes sustanciales del software.

Apéndice B. Especificación de Requisitos

B.1. Introducción

Este apéndice se enfoca en la definición detallada de los requisitos de la aplicación, los cuales guiarán su diseño y desarrollo para que el producto final cumpla con los objetivos establecidos.

Debido a que este proyecto es una continuación de un proyecto anterior, se ampliará el diagrama de casos de uso del anterior proyecto, pero no se explicarán dichos casos ni se tendrán en cuenta los requisitos que no hayan sido añadidos en este proyecto, utilizando códigos de referencia nuevos.

B.2. Objetivos generales

- Desarrollar una aplicación web para optimizar la organización de los alumnos en un colegio y reducir el riesgo de contagio.
- Implementar un algoritmo genético como método de optimización multiobjetivo para aportar diversidad de soluciones al problema.
- Mostrar gráficamente la evaluación de distintas soluciones exploradas y resaltar el frente de Pareto.
- Aportar una solución intuitiva para el usuario y con información relevante para la toma de decisiones.

B.3. Catálogo de requisitos

En este apartado se mostrarán los requisitos funcionales y no funcionales.

Requisitos funcionales

RF-1. Selección de algoritmo. La aplicación debe permitir al usuario seleccionar entre el algoritmo genético y el recocido simulado.

RF-2. Configuración de Parámetros del Algoritmo Genético. La aplicación debe permitir al usuario configurar los parámetros del algoritmo genético, como el número de generaciones, el tamaño de la población, la probabilidad de cruce y mutación, y el operador de cruce. Además, debe ofrecer la opción de utilizar parámetros por defecto predefinidos.

RF-3. Mostrar soluciones. La aplicación debe mostrar las soluciones generadas por el algoritmo genético.

RF-3.1 Visualizar soluciones exploradas. La aplicación debe mostrar tres gráficas con la evaluación de las soluciones exploradas, resaltando las soluciones óptimas en otro color para facilitar su identificación.

RF-3.2 Mostrar botones de descarga. La aplicación debe mostrar botones de descarga para cada solución en una lista.

RF-4. Creación de archivos solución. La aplicación debe crear y gestionar archivos para las soluciones generadas.

RF-4.1 Corrección para soluciones no factibles. La aplicación debe convertir las soluciones no factibles en soluciones factibles antes de su almacenamiento o descarga.

RF-4.2 Creación de nombre para la solución. La aplicación debe asignar un nombre único y descriptivo a cada archivo solución.

RF-4.3 Creación de los archivos. La aplicación debe generar los archivos correspondientes a cada solución.

RF-5. Descarga de los archivos solución. La aplicación debe permitir al usuario descargar los archivos de solución generados.

RF-5.1 Descargar una sola solución. La aplicación debe permitir al usuario descargar una sola solución específica en un formato adecuado.

RF-5.2 Descargar todas las soluciones. La aplicación debe permitir al usuario descargar todas las soluciones generadas en un único archivo comprimido.

Requisitos no funcionales

RNF-1. Accesibilidad. Las soluciones de la aplicación deben estar diseñadas de manera que sean comprensibles y fáciles de usar para los encargados de administración del colegio.

RNF-2. Mantenibilidad. El código de la aplicación debe estar estructurado de forma modular y desacoplada, permitiendo que las modificaciones, mejoras y correcciones puedan realizarse de manera eficiente y con un mínimo de impacto en el resto del sistema. El código también debe tener una correcta documentación interna.

RNF-3. Escalabilidad. La arquitectura de la aplicación debe permitir el manejo de una mayor carga de trabajo sin una reestructuración significativa, garantizando que el sistema pueda evolucionar conforme a las necesidades futuras.

RNF-4. Usabilidad. La aplicación debe proporcionar una experiencia de usuario intuitiva, facilitando la navegación y el uso de sus funcionalidades mediante una interfaz clara y amigable. Además, la interfaz debe ser adaptable y responsive, garantizando una visualización óptima en dispositivos con distintos tamaños de pantalla.

RNF-5. Estabilidad. La aplicación debe ser capaz de ejecutar y gestionar múltiples hilos de procesamiento de manera continua durante un período máximo de 24 horas. Esto es esencial para que el algoritmo pueda realizar una exploración más amplia y profunda de las soluciones.

B.4. Especificación de requisitos

En este apartado se definirán los casos de uso junto al diagrama de casos de uso. Debido a que sólo hay un tipo de usuario, el actor en todos los casos será el usuario.

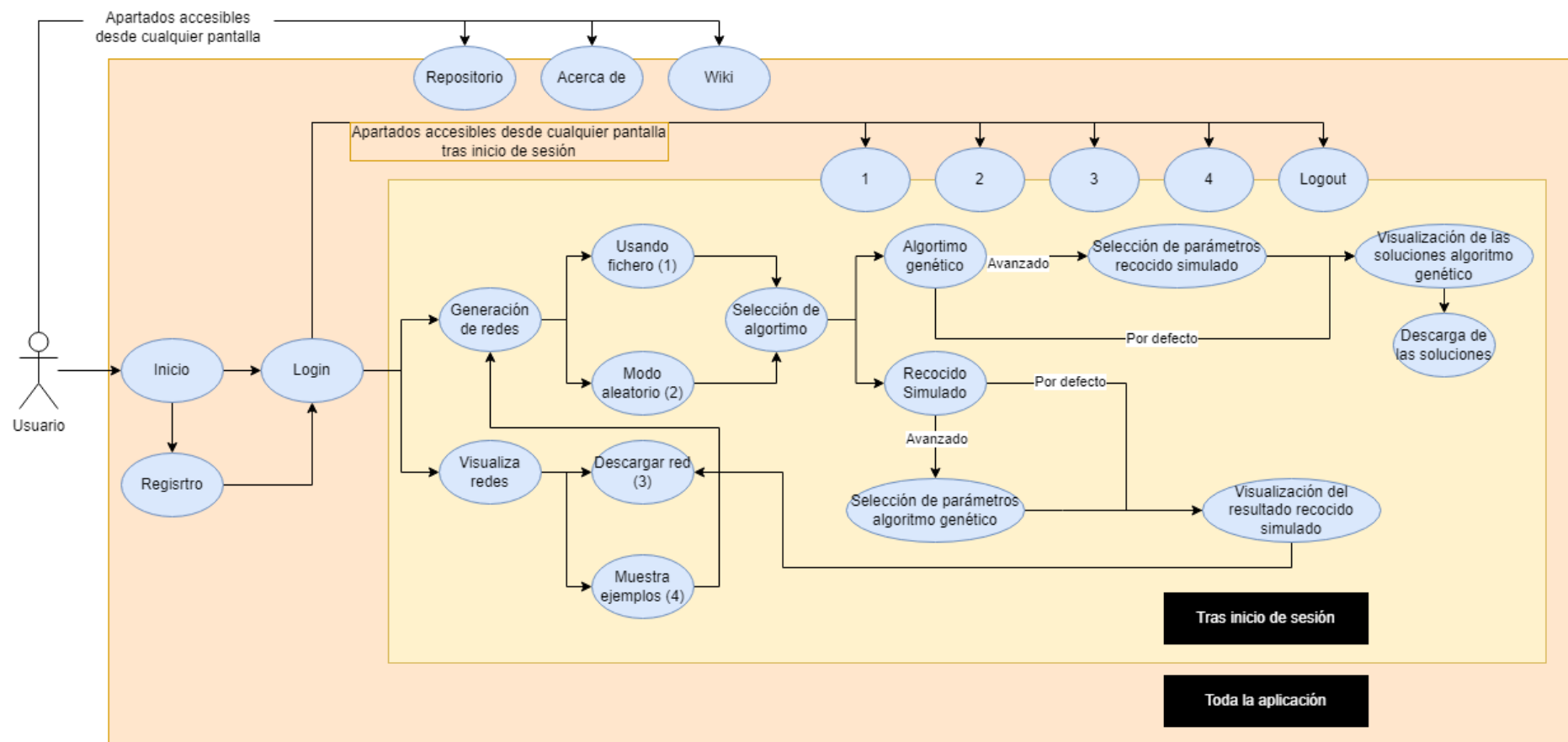


Ilustración 1: Diagrama de casos de uso

Tabla 7: Caso de uso 1

CU-1	Seleccionar Algoritmo	
Requisitos asociados	RF-1	
Descripción	Permitir al usuario elegir entre el algoritmo genético y el recocido simulado como método de Optimización.	
Precondición	Se deben haber introducido los datos de entrada del problema	
Acciones	Paso	Acción
	1	El usuario accede a la interfaz de selección de algoritmos.
	2	El usuario debe elegir una opción entre algoritmo genético y recocido simulado
	3	La aplicación registra la selección del usuario.
Postcondición	Se redirige a la pantalla de selección de parámetros dependiendo del algoritmo escogido	
Excepciones	Ninguna	

Tabla 8: Caso de uso 2

CU-2	Seleccionar parámetros genéticos	
Requisitos asociados	RF-2	
Descripción	Permitir al usuario elegir los parámetros del algoritmo genético	
Precondición	Se debe haber seleccionado el tipo de algoritmo	
Acciones	Paso	Acción
	1	Introducir el valor para el número de generaciones
	2	Introducir el valor para el tamaño de la población
	3	Introducir el valor para la probabilidad de cruce
	4	Introducir el valor para la probabilidad de mutación
	5	Elegir el operador de cruce
Postcondición	El algoritmo genético se ejecuta	
Excepciones	Número	Excepción
	1	El número de generaciones y el tamaño de la población deben ser enteros positivos
	2	Los valores de probabilidad deben estar entre 0 y 1

Tabla 9: Caso de uso 3

CU-3	Mostrar soluciones	
Requisitos asociados	RF-3	
Descripción	Se muestran 3 gráficas de las soluciones exploradas al usuario resaltando el frente de Pareto. Debajo de las gráficas se muestran las opciones de descarga para cada solución.	
Precondición	La ejecución del algoritmo genético debe haber finalizado.	
Acciones	Paso	Acción
	1	Se muestran las gráficas
	2	Se muestran los botones de descarga para las soluciones
Postcondición	Ninguna	
Excepciones	Ninguna	

Tabla 10: Caso de uso 4

CU-4	Descargar solución	
Requisitos asociados	RF-3.2, RF-4, RF-5.1	
Descripción	Permitir al usuario descargar una solución individual	
Precondición	La ejecución del algoritmo genético debe haber finalizado.	
Acciones	Paso	Acción
	1	Pulsar en el botón de descarga de una solución
	2	La aplicación convierte la solución a factible si es necesario
	4	Se pone nombre a la solución
	3	Los archivos solución son creados y comprimidos
Postcondición	Los archivos solución son enviados al usuario	
Excepciones	Número	Excepción
	1	El número de iteraciones para convertir la solución a factible supera el número de clases

Tabla 11: Caso de uso 5

CU-5		Descargar todas las soluciones	
Requisitos asociados	RF-3.2, RF-4, RF-5		
Descripción	Permitir al usuario descargar todas las soluciones		
Precondición	La ejecución del algoritmo genético debe haber finalizado.		
Acciones	Paso	Acción	
	1	Pulsar en el botón de descargar todas las soluciones	
	2	La aplicación convierte la siguiente solución a factible si es necesario	
	3	Se pone nombre a la solución actual	
	4	Los archivos de la solución actual son creados y guardados en una carpeta	
	5	Se repite desde la acción 2 con todas las soluciones y se comprimen las carpetas	
Postcondición	El conjunto de soluciones es enviado al usuario		
Excepciones	Número	Excepción	
	1	El número de iteraciones para convertir una solución a factible supera el número de clases	

Apéndice C. Especificación de diseño

C.1. Introducción

En este apartado se detallarán el diseño de datos, el diseño procedural y el diseño arquitectónico de la aplicación.

C.2. Diseño de datos

Debido a que no se ha implementado la base de datos del proyecto, y considerando que esta estructura está explicadas en la documentación del proyecto anterior, en este apartado nos centraremos en las variables añadidas al proyecto actual. Estas variables se encuentran definidas en el archivo `global_def.py`, que se utiliza para manejar datos de forma centralizada en las funciones relacionadas con el algoritmo genético.

Las variables introducidas en el proyecto son las siguientes:

classrooms. Lista que almacena la información de las diferentes clases de estudiantes. Cada elemento de la lista representa un grupo distinto. Sirve para conocer los valores que pueden tomar las variables de decisión y asignar un grupo a cada estudiante.

capacity. Variable que define la capacidad máxima permitida para cada grupo. Sirve para evaluar la viabilidad de las soluciones.

total_students. Variable que almacena el número total de estudiantes que se organizarán en grupos o clases. Se utiliza en el cálculo de la capacidad.

siblings_number. Variable que almacena el número de pares de hermanos en la red de estudiantes. Se utiliza para la creación de redes aleatorias.

initial_network. Grafo que representa la red inicial de estudiantes. Contiene un grafo que modela las interacciones entre estudiantes. Cada nodo tiene atributos para el id, nombre, curso, etapa y grupo de cada estudiante.

graph_eval_ini. Grafo que representa la red de clases necesaria para la evaluación de las soluciones. Cada nodo tiene atributos que incluyen el nombre, curso, etapa, grupo y una lista de estudiantes que pertenecen a esa clase específica. Esta variable sirve para almacenar el grafo inicial con las clases vacías y copiarlo en cada evaluación para rellenarlo de diferente forma.

siblings_dict. Diccionario que almacena las relaciones de hermandad entre los estudiantes. Este diccionario se deriva de la matriz de hermanos y utiliza los nombres de los estudiantes como claves, asignando a cada clave una lista de atributos que incluye la posición en el genotipo, etapa, curso, grupo y una lista de los nombres de sus hermanos.

C.3. Diseño procedimental

En este diagrama de secuencia se muestra el procedimiento general de la aplicación:

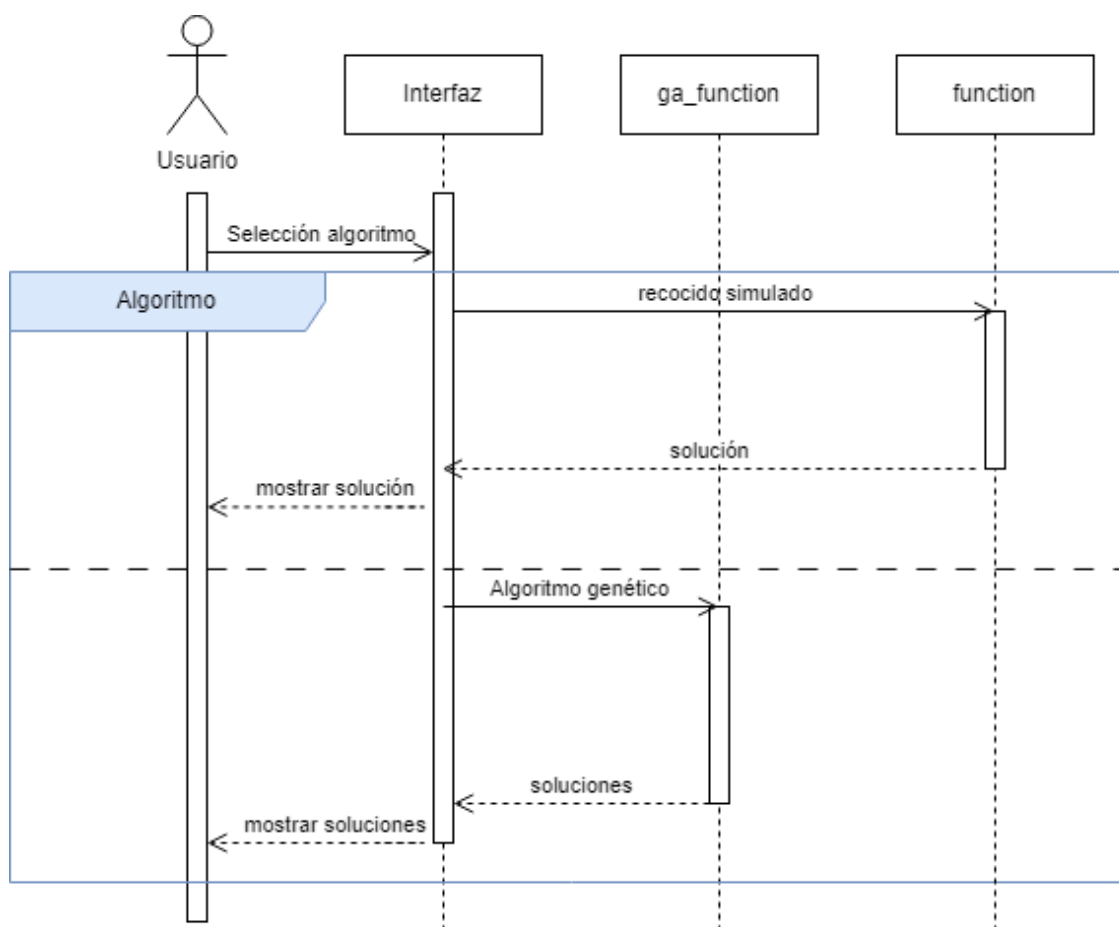


Ilustración 2: Diagrama de secuencia

C.4. Diseño arquitectónico

En el diseño arquitectónico se ha aplicado el patrón Modelo-Vista-Controlador (MVC), que es un patrón de diseño que separa la gestión de datos, la lógica de negocio y la presentación de la interfaz de usuario en tres componentes distintos.

En el contexto de este proyecto, el patrón MVC se aplica de la siguiente manera:

El Modelo. Se encarga de gestionar y procesar los datos relacionados con la red de estudiantes y las configuraciones de clases. Está compuesto por los ficheros y módulos que manejan la lógica de negocio y los datos. En concreto los archivos del algoritmo genético: `individual_evaluation.py`, `data_management.py`, `global_def.py` y `ga_function`.

La Vista. Representa la interfaz de usuario con la que interactúa el usuario final. Es responsable de mostrar la información generada por el Modelo y de recibir las entradas

del usuario. En este proyecto, la Vista se implementa a través de los ficheros HTML y CSS en las carpetas de templates y static.

El Controlador. Actúa como intermediario entre el Modelo y la Vista, gestionando la comunicación y las actualizaciones entre estos dos componentes. En este proyecto, el Controlador es responsable de manejar las solicitudes del usuario, actualizar el Modelo con nuevas entradas, y actualizar la Vista para reflejar los cambios. Esto se implementa principalmente a través del archivo `sire.py`, que contiene la lógica de Flask para manejar las rutas, recibir entradas del usuario y coordinar la interacción entre los datos y la presentación.

Apéndice D. Documentación técnica de programación

D.1. Introducción

En este apéndice se detalla la documentación técnica del proyecto, proporcionando una guía comprensiva para la comprensión y manipulación del código fuente. Se abordarán aspectos esenciales como la organización de los directorios del proyecto, la configuración y el entorno necesario para la compilación y ejecución del código, así como el procedimiento para realizar pruebas y verificar el correcto funcionamiento del sistema.

D.2. Estructura de directorios

En el proyecto se presentan varios directorios principales, cada uno con una función específica que contribuye al funcionamiento y organización del sistema. A continuación, se detalla la estructura de estos directorios y los archivos clave incluidos en cada uno de ellos:

/Documentación: contiene toda la documentación relacionada con el proyecto, incluyendo la memoria y los anexos.

/Documentación/Imagenes_documentacion: almacena todas las imágenes que se utilizan en la documentación del proyecto, tales como diagramas y capturas de pantalla.

/Environment/sire: este es el directorio principal donde se encuentra el código fuente de la aplicación, tanto el backend escrito en Python como los elementos necesarios para la interfaz de usuario.

/Environment/sire/Net_images: almacena las imágenes que se muestran en la aplicación como resultado del recocido simulado.

/Environment/sire/files_to_upload: contiene ejemplos de archivos que se pueden subir a la aplicación para su procesamiento.

/Environment/sire/static/css: incluye los archivos CSS que definen el estilo visual de los documentos HTML utilizados en la interfaz de usuario.

/Environment/sire/templates: contiene los archivos HTML que componen la interfaz de usuario de la aplicación.

/Environment/sire/templates/App_images: almacena las imágenes que se visualizan en las distintas pantallas de la aplicación.

/Environment/sire/database.py: archivo que gestiona la base de datos.

/Environment/sire/fileNetCreation.py: archivo encargado de procesar los archivos de entrada para generar la red de clases.

/Environment/sire/function.py: contiene la implementación del algoritmo de recocido simulado y las secuencias de enfriamiento necesarias.

/Environment/sire/procedure.sql: documento que contiene los pasos para la creación de usuarios en la base de datos en el proyecto original.

/Environment/sire/randomNetCreation.py: genera de forma aleatoria la red inicial de estudiantes y la tabla de hermanos.

/Environment/sire/sire.py: actúa como el servidor principal de la aplicación. Este archivo maneja las solicitudes HTTP y coordina la interacción entre la interfaz y los procesos de datos.

/Environment/sire/testRandomNet.py: contiene pruebas para verificar la integridad y consistencia de los archivos subidos con respecto a los archivos originales.

/Environment/sire/global_def.py: define las variables globales utilizadas en todo el algoritmo genético.

/Environment/sire/data_management.py: gestiona la entrada y salida de datos, así como la transformación de estos para su procesamiento en el sistema.

/Environment/sire/individual_evaluation.py: realiza operaciones relacionadas con los genotipos dentro del contexto del algoritmo genético.

/Environment/sire/ga_function.py: define la configuración y la lógica del algoritmo genético utilizado para la optimización en el proyecto.

/Environment/sire/procfile: define los comandos para la ejecución de la aplicación en un entorno de despliegue para Heroku, especificando cómo iniciar la aplicación.

/Environment/sire/dockerfile: contiene las instrucciones para construir una imagen de Docker para la aplicación, facilitando su despliegue en contenedores.

/Environment/sire/docker-compose.yml: define cómo se deben orquestar los contenedores Docker para la aplicación, especificando las configuraciones de red y los servicios necesarios.

/Environment/sire/requirements.txt: lista las dependencias y bibliotecas de Python necesarias para ejecutar el proyecto, permitiendo una fácil instalación del entorno.

/Environment/sire/app.py: archivo principal que inicia la aplicación Flask, configurando las rutas y controladores necesarios para la interacción con los usuarios.

/Environment/sire/config.py: contiene la configuración de la aplicación, incluyendo variables de entorno como la SECRET_KEY o el modo debug.

/LICENSE: incluye la licencia del proyecto, especificando los términos y condiciones para el uso, modificación y distribución del software.

/README.md: contiene el archivo de README del repositorio, ofreciendo una introducción general al proyecto, instrucciones para la instalación y otros detalles relevantes para los usuarios y desarrolladores.

D.3. Manual del programador

En esta sección se detallarán las instalaciones y configuraciones necesarias para la compilación y ejecución del código tanto en Windows como en Linux. Este manual está diseñado para ayudar a cualquier desarrollador que desee modificar el proyecto y ejecutar la aplicación localmente o en un entorno de despliegue.

Instalación y ejecución en Windows

En Windows, se permite la ejecución en local del servidor Flask para el desarrollo y pruebas. A continuación, se detallan los pasos necesarios para la instalación de todas las herramientas y dependencias requeridas:

Descargar e instalar Git:

<https://git-scm.com/download/win>

Descargar e instalar Python 3.10.12:

<https://www.python.org/downloads/release/python-31012/>

Descargar e instalar MySQL Community Server 8.0.20:

<https://dev.mysql.com/downloads/installer/>

En la instalación instalar los paquetes mysql-server y mysql-client.

Ahora necesitamos crear una base de datos en MySQL para que la aplicación pueda ejecutarse en local. Para hacer esto:

Ir a Inicio, buscar “MySQL command line client” y ejecutar la aplicación.

Abrir la consola de MySQL como usuario root:

```
sudo mysql -u root
```

Cambiar la autenticación del usuario root y establecer la contraseña a “root”:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH  
mysql_native_password BY 'root';
```

Esto se hace para que en database.py se pueda conectar a la base de datos mediante una contraseña.

Crear una nueva base de datos llamada sire:

```
CREATE DATABASE sire;
```

Salir de la consola de MySQL escribiendo:

```
EXIT;
```

Ahora se debe clonar el repositorio mediante Git. En una ventana de CMD, se debe ir al directorio donde se quiera clonarlo y escribir:

```
git clone https://github.com/rubenarasti/Sibling-Rewiring.git
```

Navegar a la carpeta del proyecto:

```
cd Sibling-Rewiring/Environment/sire
```

Crear y activar un entorno virtual:

```
python -m venv venv  
venv\Scripts\activate
```

Instalar las dependencias del proyecto:

```
pip install -r requirements.txt
```

Crear un archivo llamado .env y rellenarlo de la siguiente manera:

```
SECRET_KEY=claveSecreta  
IS_DOCKER=false
```

La clave secreta debe ser un valor que sólo conozcamos nosotros.

Ejecutar la aplicación con:

```
python app.py
```

Este comando lanza la aplicación utilizando python, que es adecuado para el desarrollo y pruebas, pero no debe usarse en producción.

Instalación y ejecución en Linux

En Linux (se ha utilizado Ubuntu 22.04), se permite la ejecución local con Flask y Gunicorn, además de la opción de ejecutar la aplicación en un contenedor Docker para facilitar el despliegue. A continuación, se detallan los pasos necesarios para la instalación y configuración:

Actualizar la lista de paquetes:

```
sudo apt-get update
```

Instalar Git:

```
apt-get install git
```

Instalar Python y herramientas necesarias:

```
sudo apt-get install python3 python3-venv python3-pip
```

Clonar el repositorio y moverse a la carpeta del proyecto:

```
git clone https://github.com/rubenarasti/Sibling-Rewiring.git
cd Sibling-Rewiring/Environment/sire
```

Configurar el entorno virtual de Python:

```
python3 -m venv environment
source environment/bin/activate
pip install -r requirements.txt
```

Instalar MySQL:

```
sudo apt install mysql-server mysql-client
```

Configurar MySQL:

```
sudo mysql -u root

ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'root';

CREATE DATABASE sire;

EXIT;
```

Escribir variables de entorno en el archivo .env:

```
echo -e "SECRET_KEY=claveSecreta\nIS_DOCKER=false" > .env
```

Ejecutar la aplicación con Python:

```
python app.py
```

Ejecutar la aplicación con Gunicorn:

```
gunicorn -b 0.0.0.0:5000 --timeout 1800 app:app
```

Instalación y ejecución con Docker

Para utilizar Docker es necesario que nos encontremos en un entorno Linux o que estemos en Windows con WSL instalado, aunque es mucho más recomendable la primera opción, ya que WSL suele causar problemas de rendimiento en Windows. [5]

Si ya se tiene instalado Docker y Docker Compose, y además se ha clonado el repositorio, tan sólo es necesario moverse a la carpeta Environment/sire dentro de los archivos del repositorio y ejecutar:

```
sudo docker-compose up --build
```

A continuación se detallan los pasos necesarios para su correcta instalación y ejecución:

Actualizar la lista de paquetes:

```
sudo apt-get update
```

Instalar paquetes necesarios:

```
sudo apt-get install -y apt-transport-https ca-certificates curl  
gnupg lsb-release
```

Añadir la clave GPG oficial de Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo  
gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Añadir el repositorio Docker APT:

```
echo "deb [arch=$(dpkg --print-architecture) signed-  
by=/usr/share/keyrings/docker-archive-keyring.gpg]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs)  
stable" | sudo tee /etc/apt/sources.list.d/docker.list >  
/dev/null
```

Instalar Docker CE, Docker CE CLI y containerd:

```
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

Instalar Docker Compose:

```
sudo curl -L  
"https://github.com/docker/compose/releases/download/$(curl -s  
https://api.github.com/repos/docker/compose/releases/latest |  
grep -Po '"tag_name": "\K.*?(?=")')/docker-compose-$(uname -s)-  
$(uname -m)" -o /usr/local/bin/docker-compose  
  
sudo chmod +x /usr/local/bin/docker-compose
```

Escribir variables de entorno en el archivo .env:

```
echo -e "SECRET_KEY=claveSecreta\nIS_DOCKER=true" > .env
```

Clonar el repositorio del proyecto:

```
git clone https://github.com/rubenarasti/Sibling-Rewiring.git
```

Navegar al directorio del proyecto:

```
cd Sibling-Rewiring/Environment/sire
```

Construir y ejecutar el contenedor Docker:

```
sudo docker-compose up --build
```

La imagen de Docker debería ejecutarse y debería estar disponible la página web en 0.0.0.0:5000.

D.5. Pruebas del sistema

En esta sección se detallarán las diferentes pruebas que se llevarán a cabo para comprobar el correcto funcionamiento de la aplicación. Todas las pruebas serán manuales debido a la ausencia de una batería de tests automáticos.

1. Selección de algoritmo

Descripción: Verificar que la interfaz permite al usuario seleccionar entre el algoritmo genético y el recocido simulado.

Pasos:

- a) Acceder a la interfaz de selección de algoritmos desde un navegador.
- b) Elegir una opción entre el algoritmo genético y el recocido simulado.
- c) Confirmar la selección.

Resultado esperado: La selección del usuario se registra correctamente y se redirige a la pantalla de selección de parámetros del algoritmo escogido.

2. Selección de parámetros genéticos

Descripción: Verificar que el usuario puede introducir y seleccionar los parámetros necesarios para la ejecución del algoritmo genético.

Pasos:

- a) Acceder a la interfaz de selección de parámetros genéticos tras seleccionar el algoritmo genético.
- b) Introducir el valor para el número de generaciones.
- c) Introducir el valor para el tamaño de la población.
- d) Introducir el valor para la probabilidad de cruce.
- e) Introducir el valor para la probabilidad de mutación.
- f) Elegir el operador de cruce.

Resultado esperado: Los parámetros se introducen correctamente y el algoritmo genético se ejecuta sin errores. Las validaciones aseguran que el número de generaciones y el tamaño de la población son enteros positivos y que las probabilidades están entre 0 y 1.

3. Mostrar soluciones

Descripción: Verificar que, tras la ejecución del algoritmo genético, se muestran correctamente las gráficas de las soluciones exploradas y las opciones de descarga.

Pasos:

- a) Ejecutar el algoritmo genético.
- b) Esperar a que la ejecución finalice.
- c) Verificar que se muestran tres gráficas de las soluciones exploradas resaltando el frente de Pareto.
- d) Verificar que debajo de las gráficas se muestran los botones de descarga para cada solución.

Resultado esperado: Las gráficas y los botones de descarga se muestran correctamente al usuario.

4. Descargar una solución individual

Descripción: Verificar que el usuario puede descargar una solución individual desde la interfaz.

Pasos:

- a) Ejecutar el algoritmo genético y esperar a que finalice.
- b) Pulsar en el botón de descarga de una solución.
- c) Verificar que la aplicación convierte la solución a factible si es necesario.
- d) Verificar que la solución se nombra correctamente.
- e) Verificar que los archivos solución no están vacíos.

Resultado esperado: Los archivos de la solución seleccionada son enviados correctamente al usuario. Si es necesario convertir la solución a factible, la solución recalcula el fitness y se añade la flag “modified” al nombre.

5. Descargar todas las soluciones

Descripción: Verificar que el usuario puede descargar todas las soluciones generadas por el algoritmo genético.

Pasos:

- a) Ejecutar el algoritmo genético y esperar a que finalice.
- b) Pulsar en el botón de descargar todas las soluciones.
- c) Verificar que la aplicación convierte cada solución a factible si es necesario.
- d) Verificar que cada solución se nombra correctamente.
- e) Verificar que los archivos de cada solución se crean y guardan en una carpeta.
- f) Repetir el proceso para todas las soluciones.
- g) Verificar que todas las carpetas con las soluciones se comprimen y se envían correctamente al usuario.

Resultado esperado: El conjunto de soluciones es enviado correctamente al usuario. Si es necesario convertir alguna solución a factible, se realiza sin superar el número de iteraciones permitido.

Apéndice E. Documentación de usuario

E.1. Introducción

En este apéndice se proporciona toda la información necesaria para que los usuarios puedan instalar, configurar y utilizar la aplicación de manera efectiva. Se describirán los requisitos que los usuarios deben cumplir, los pasos para la instalación del software y un manual detallado del usuario que incluye instrucciones para la operación de la aplicación y la resolución de problemas comunes.

El despliegue de la aplicación tiene limitaciones técnicas y ninguna consulta a la web puede durar más de 30 segundos. En ocasiones, al enviar una consulta, puede ocurrir un error interno del servidor (Internal Server Error) debido a cambios de hilo en el servidor. Esto puede resultar en una experiencia de usuario interrumpida. La solución a este problema es simplemente retroceder en el navegador y volver a intentar la consulta hasta que se procese correctamente. Si se muestra un error de Heroku, la solución es volver a introducir los datos de entrada.

Debido a estas limitaciones, se proponen dos opciones: el acceso a la aplicación desde el despliegue en Heroku y el acceso desde una máquina virtual que actúe como servidor y permita conectarse a esta.

E.2. Requisitos de usuarios

Los requisitos mínimos de la aplicación web son los siguientes:

- Conexión a internet.
- Un navegador web actualizado (Chrome, Edge, Mozilla Firefox o Safari).

Para la alternativa de utilizar la aplicación a través de una máquina virtual, se requiere además lo siguiente:

- VirtualBox instalado en el equipo. [6] Es posible que la máquina virtual funcione con otros servicios de virtualización, pero no se ha probado con ellos.
- La imagen de la máquina virtual (.ova) proporcionada, que contiene la aplicación preconfigurada y lista para ejecutarse como servidor.

E.3. Instalación

Para acceder a la aplicación no es necesario instalar nada, simplemente visitar el siguiente enlace: <https://sire-ddeac2c1782a.herokuapp.com/>

Importar máquina virtual

Si se desea utilizar la funcionalidad completa de la aplicación, será necesario instalarla en local. Este proceso está explicado en detalle en el apartado [D.3, Manual del programador](#). Alternativamente, se puede importar la máquina virtual en VirtualBox con todo ya instalado para ejecutar la aplicación. Para hacer esto último, siga estos pasos:

Importar la máquina virtual:

Abrir VirtualBox.

Ir al menú Archivo y seleccionar “Importar servicio virtualizado”.

En la ventana de importación, navegar hasta el archivo de la imagen de la máquina virtual descargada y seleccionarlo.

Seguir las instrucciones en pantalla para completar la importación.

Ejecutar la máquina virtual:

Una vez importada, seleccionar la máquina virtual en la lista de VirtualBox.

Hacer clic en el botón Iniciar para ejecutar la máquina virtual.

Ejecutar aplicación

Una vez se haya iniciado la máquina virtual, hacer clic derecho en el escritorio y abrir en una terminal y escribir:

```
./run.sh
```

Esto ejecutará la imagen de Docker. Si se abre el navegador en la dirección 0.0.0.0:5000, se podrá acceder a la aplicación.

Reenvío de puertos

Para acceder a la aplicación desde el sistema anfitrión, es necesario conocer la IP asignada a la máquina virtual.

Desde el centro de redes y recursos compartidos del panel de control de Windows, dentro de “Cambiar configuración del adaptador”, podemos ver qué adaptador está asignado a la máquina virtual y así comprobar su dirección ip.

Después debemos establecer una regla de reenvío de puertos para permitir la conexión desde fuera de la máquina virtual. El puerto invitado debe ser el mismo que abre la aplicación (puerto 5000). Si no existe, hay que agregar una nueva regla, configurando el puerto anfitrión a 5000 y el puerto invitado a 5000.

Ahora es posible conectarse a la aplicación desde el navegador del sistema anfitrión utilizando la IP de la máquina virtual seguida de :5000. Por ejemplo:

`http://<IP-de-la-máquina-virtual>:5000`

E.4. Manual del usuario

En este apartado se explicarán las pantallas clave para la ejecución de la funcionalidad añadida en esta iteración del proyecto. Algunas imágenes han sido reutilizadas del anterior proyecto [7] debido a que dichas pantallas no han cambiado y son esenciales para explicar el funcionamiento de la aplicación.

A continuación se mostrarán las pantallas principales y sus funcionalidades:

Inicio



Ilustración 3 [7]: pantalla de Inicio

Esta pantalla muestra el logo de la aplicación. En la parte superior, se encuentra la barra de navegación, que contiene un menú desplegable a la derecha del todo. En dicho menú, podemos dirigirnos a la pantalla de Inicio, al repositorio del proyecto, a la pantalla de Acerca de, a la wiki y a la pantalla Login. Para poder ejecutar el algoritmo genético, deberemos ir a la pantalla de Login.

Login

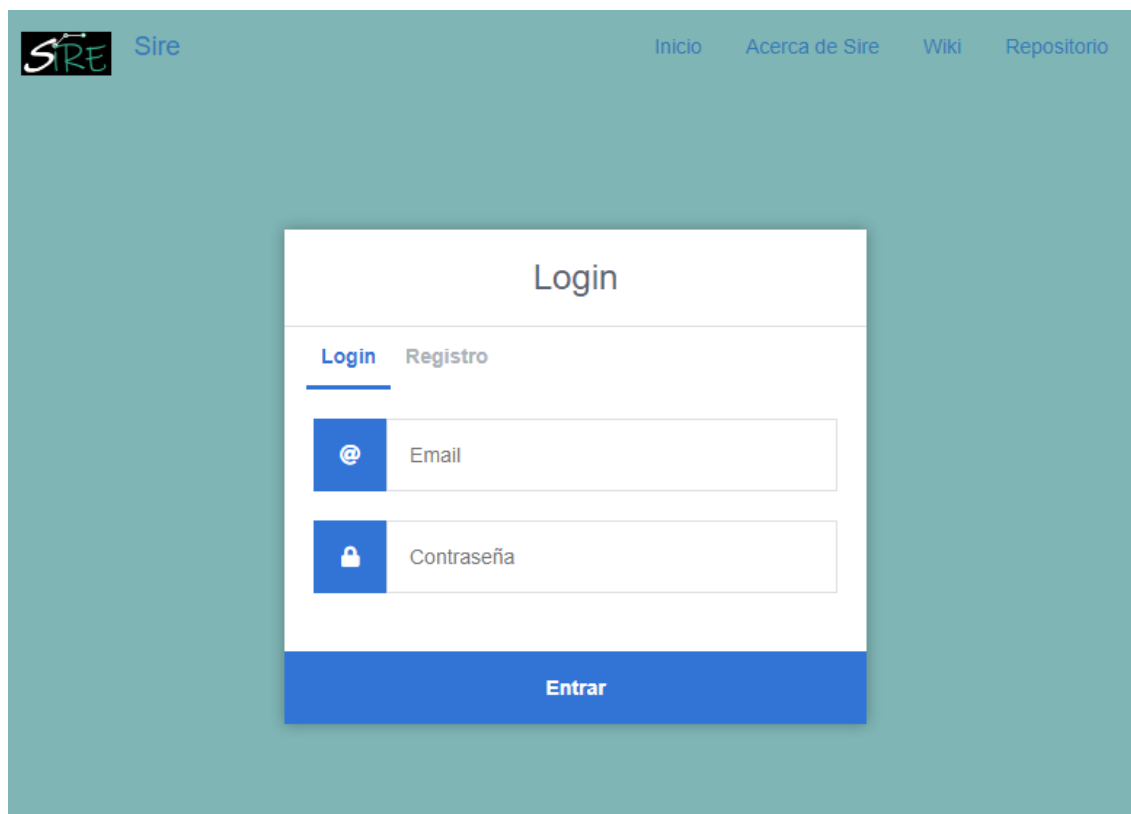


Ilustración 4 [7]: pantalla de Login

En esta pantalla se puede iniciar una sesión de usuario introduciendo el correo electrónico de este y la contraseña que se utilizó al registrarse. Una vez se inicie una sesión se redirigirá a la pantalla de inicio de sesión. Si no se ha registrado ningún usuario, se deberá ir a la pantalla de Registro pinchando en la pestaña “Registro” que aparece arriba del campo “Email”.

Registro

En esta pantalla hay 5 campos para completar la acción de registrar un usuario:

- Nombre. Introducir el nombre del usuario. Obligatorio.
- Apellido. Introducir el apellido del usuario. Obligatorio.
- Email. Introducir un email válido del usuario. Debe tener la forma [a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+. Obligatorio.
- Colegio. Introducir colegio del usuario. Opcional.
- Contraseña. Introducir la contraseña que desea el usuario. Obligatorio.

Después de rellenar los campos correctamente, pulsando la tecla Enter o el botón “Registrar” se registrará el usuario y se redirigirá a la pantalla de Login.

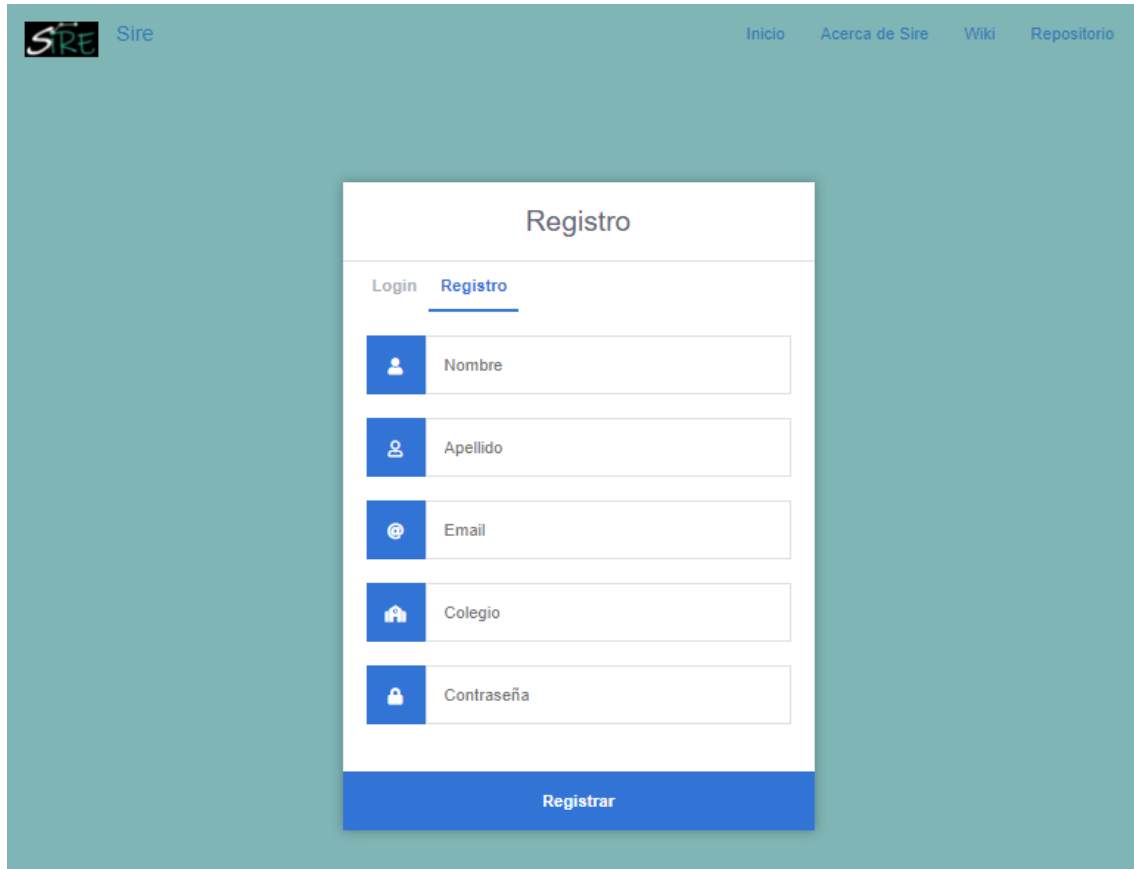


Ilustración 5 [7]: pantalla de Registro

Inicio de sesión

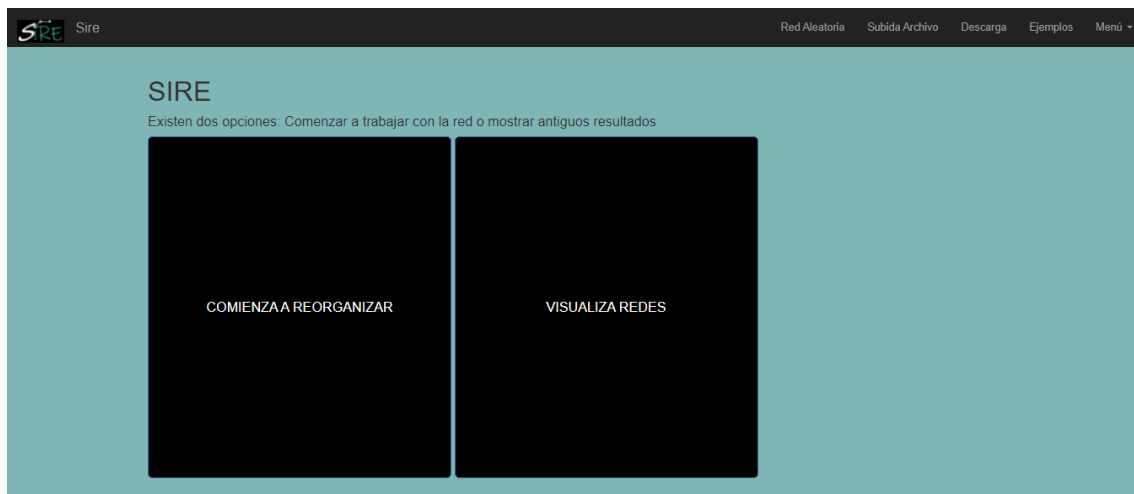


Ilustración 6 [7]: pantalla de Inicio de sesión

En esta pantalla hay 2 opciones que redirigen a distintas pantallas:

- Comienza a reorganizar: donde se elegirá el tipo de datos de entrada.
- Visualiza redes: donde se verán ejemplos de redes.

Selección del tipo de datos de entrada



Ilustración 7 [7]: pantalla de Selección del tipo de datos de entrada

En esta pantalla se decide si ir a la pantalla de Generación de redes aleatorias o a la pantalla de subida de ficheros.

Generación de redes aleatorias

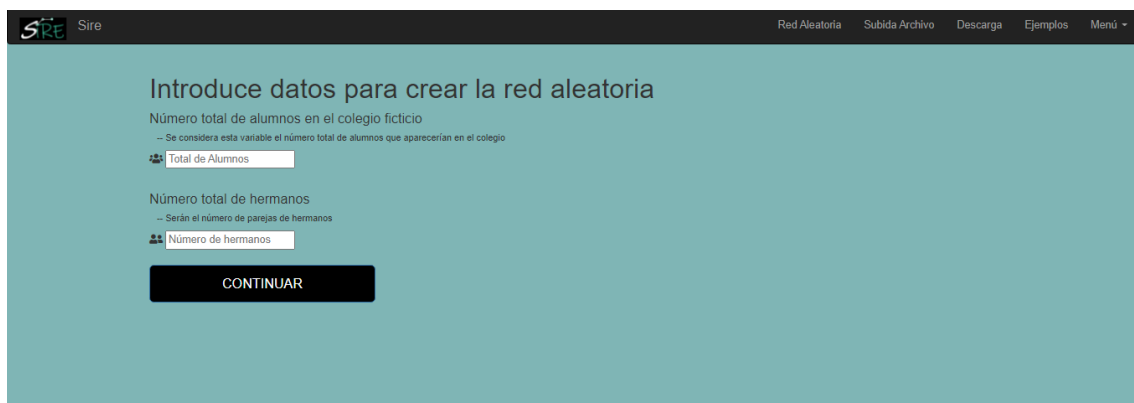


Ilustración 8 [7]: pantalla de Generación de redes aleatorias

En esta pantalla se introducirán valores enteros positivos para elegir el número total de alumnos y el número total de pares de hermanos en el colegio. Una vez se pulse a continuar con los campos rellenos adecuadamente se redirigirá a la pantalla de Selección de algoritmo.

Subida de ficheros

Selecciona el archivo para subir

Debe introducir un fichero '.graphml' que contenga la red inicial del colegio y un '.xlsx' que contenga la relación de hermanos.

Elegir archivos Ningún archivo seleccionado

Subir

Ilustración 9 [7]: Subida de ficheros

En esta pantalla se deberá pulsar el botón “Elegir archivos” para subir 2 archivos: uno con la red de clases en formato GEXF y otro con la tabla de hermanos en formato CSV. Al subir los archivos y pulsar en “Subir” se redirigirá a la pantalla de Selección de algoritmo.

Selección de algoritmo

SELECCIONE EL ALGORITMO DESEADO

ALGORITMO GENÉTICO

RECOCIDO SIMULADO

Ilustración 10: pantalla de Selección de algoritmo

En esta pantalla se elegirá el algoritmo para organizar a los estudiantes. Cada opción llevará a la pantalla de la Selección de opciones de parámetros del algoritmo respectivo.

Selección de opciones de parámetros del algoritmo genético

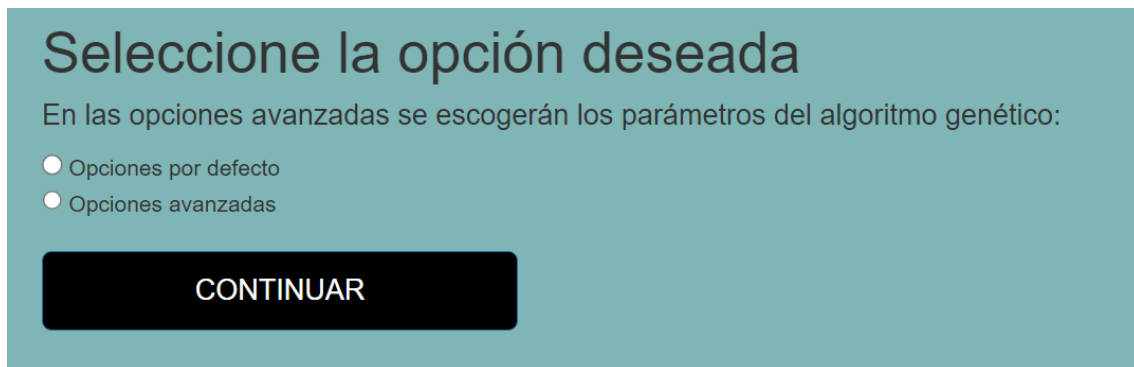


Ilustración 11: pantalla de Selección de opciones del algoritmo genético

En esta pantalla se seleccionará una opción para indicar si se quieren los parámetros por defecto o avanzados. Si se escogen las opciones por defecto, el algoritmo se ejecutará y se redirigirá a la pantalla de Mostrar soluciones; por el contrario, si se escogen las opciones avanzadas se redirigirá a la pantalla de Selección de parámetros del algoritmo genético.

Selección de parámetros del algoritmo genético

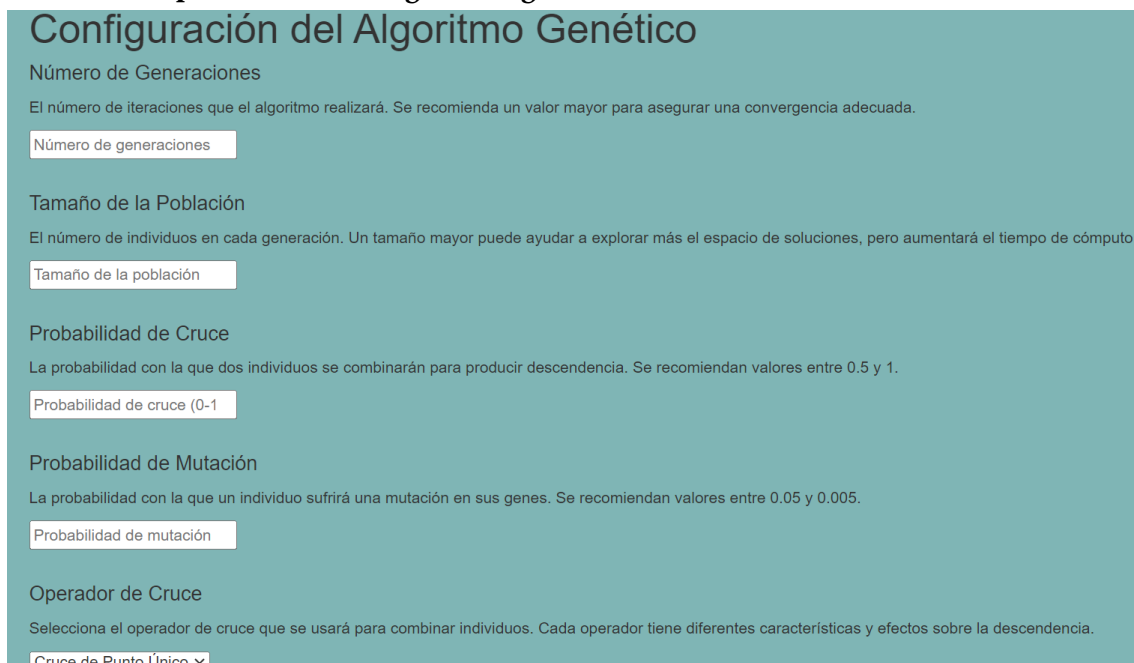


Ilustración 12: pantalla de Selección de parámetros del algoritmo genético

En esta pantalla se rellenan los campos necesarios para ejecutar el algoritmo genético como el número de generaciones, el tamaño de la población, las probabilidades de cruce y mutación y el operador de cruce. Las probabilidades deben ser valores entre 0 y 1 y no pueden tener más de tres decimales de precisión. El número de generaciones y el tamaño de la población deben ser enteros positivos. Una vez rellenos los campos

correctamente, al pulsar en el botón “CONTINUAR” al final de la página, se ejecutará el algoritmo genético y se redirigirá a la pantalla de Mostrar soluciones.

Mostrar soluciones

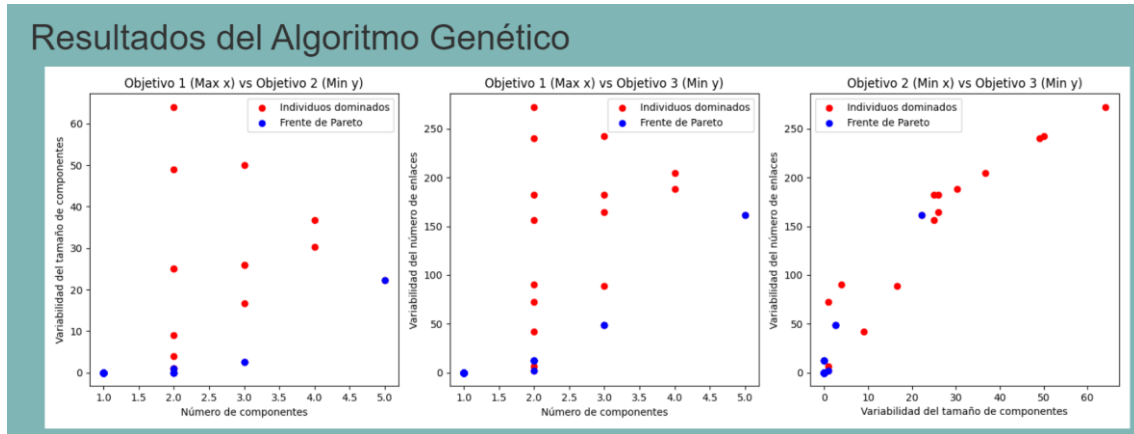


Ilustración 14: Resultados del AG, gráficas

Se encontraron 200 soluciones diferentes

[Descargar todas las soluciones](#)

Lista de Soluciones

Soluciones	Fitness (Nº Componentes, Variabilidad del Tamaño de los Componentes, Variabilidad del Nº de Enlaces)	Descargas
Solución 1	(5, 22.24, 161.44)	Descargar
Solución 2	(3, 2.67, 48.67)	Descargar
Solución 3	(3, 2.67, 48.67)	Descargar
Solución 4	(2, 1.0, 2.25)	Descargar
Solución 5	(2, 0.0, 12.25)	Descargar
Solución 6	(2, 0.0, 12.25)	Descargar
Solución 7	(2, 0.0, 12.25)	Descargar
Solución 8	(2, 0.0, 12.25)	Descargar

Ilustración 13: Resultados del AG, descargas

En esta pantalla se muestra la evaluación de las soluciones exploradas por el algoritmo genético (en rojo) y qué soluciones han sido las elegidas (en azul). Debajo de la gráfica, se muestra el número total de soluciones diferentes encontradas y un botón para descargar todas las soluciones. Si se quiere descargar sólo una solución en específico, se muestra una lista con las soluciones numeradas, su evaluación y un botón para descargar dicha solución.

Apéndice F. Anexo de sostenibilización curricular

F.1 Introducción

La Agenda 2030 de las Naciones Unidas establece 17 Objetivos de Desarrollo Sostenible (ODS) que buscan abordar los principales desafíos globales, incluyendo la salud y la educación. [8] Este proyecto, centrado en la organización de los alumnos para reducir el riesgo de contagio de enfermedades en entornos escolares, contribuye directamente a los ODS relacionados con la salud y el bienestar (ODS 3) y la educación de calidad (ODS 4).

F.2 Objetivo número 3: Salud y Bienestar

La organización de los alumnos en el colegio se realiza de manera estratégica para minimizar el riesgo de contagio de enfermedades infecciosas, cumpliendo así con el ODS 3 de la Agenda 2030, que se centra en garantizar una vida sana y promover el bienestar para todos en todas las edades. Este enfoque no solo se limita a una mera disposición física, sino que integra análisis avanzados de datos para una gestión proactiva y eficaz del riesgo.

En caso de un brote de enfermedad, las soluciones propuestas por la aplicación permiten identificar rápidamente las clases que están interconectadas y que, por lo tanto, pueden representar un riesgo de transmisión. Esto facilita la implementación de medidas de control específicas, tales como el confinamiento de ciertas clases o grupos de alumnos, evitando la necesidad de cerrar todo el colegio y permitiendo que el mayor número posible de estudiantes continúe su educación en un entorno seguro. Este enfoque selectivo y dirigido contribuye a la prevención de la propagación de enfermedades, minimizando el impacto en la comunidad educativa y en la salud pública en general.

Además, la aplicación proporciona una evaluación detallada del riesgo de contagio para cada conjunto de clases, facilitando la toma de decisiones informadas sobre la asignación de docentes, especialmente aquellos considerados pacientes de riesgo. Al asignar estos docentes a grupos con menor riesgo de contagio, se protege su salud y se asegura su participación continua en la educación, promoviendo así un entorno educativo seguro y saludable para todos.

Los datos permiten también a la administración educativa la coordinación efectiva con las autoridades de salud pública al proporcionar datos precisos y actualizados sobre la situación sanitaria del colegio. Esto facilita una respuesta más rápida y eficaz ante emergencias, ayudando a contener la propagación de enfermedades y a mantener un entorno seguro para todos los miembros de la comunidad escolar.

F.2 Objetivo número 4: Educación de calidad

Este proyecto también contribuye significativamente al ODS 4, que busca garantizar una educación inclusiva, equitativa y de calidad. Al gestionar de manera efectiva las conexiones entre clases, la aplicación minimiza las interrupciones en la educación causadas por enfermedades, asegurando que los estudiantes puedan continuar su aprendizaje sin pausas significativas. Esto es especialmente importante en contextos de emergencia sanitaria, como lo fue la pandemia de COVID-19, que ha tenido un impacto profundo en la educación a nivel global.

La pandemia de COVID-19 ha destacado la importancia de la educación presencial, que ofrece un entorno de aprendizaje más enriquecedor que la educación a distancia. Las clases presenciales facilitan la interacción social y el aprendizaje práctico, aspectos fundamentales para el desarrollo académico y personal de los estudiantes. Al permitir que las clases presenciales se lleven a cabo de manera segura, la aplicación contribuye a una educación de calidad, asegurando que los estudiantes puedan beneficiarse de una experiencia educativa completa y enriquecedora.

Al considerar la salud y el bienestar de todos los profesores, incluidos aquellos que son pacientes de riesgo, la aplicación garantiza que todos los docentes puedan seguir participando activamente en la educación. Esto fomenta un entorno educativo inclusivo, donde se valora y protege la diversidad de experiencias y enfoques pedagógicos, ya que cada docente, dependiendo de su edad y experiencia, aporta una perspectiva única y valiosa al proceso educativo.

Además, la aplicación puede servir como un recurso educativo en sí mismo, enseñando a los estudiantes sobre la importancia de la salud pública y la gestión del riesgo, proporcionando conocimientos valiosos que pueden aplicar a lo largo de sus vidas. Al involucrar a los estudiantes en la comprensión de cómo la aplicación organiza las clases para minimizar el riesgo de contagio, se les introduce a conceptos clave en epidemiología y salud pública.

Bibliografía

- [1] BBVA, «BBVA ESPAÑA». Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://www.bbva.es/personas/productos/cuentas/calculadora-sueldo-neto.html>
- [2] «Fondo de Garantía Salarial - Atención a la Ciudadanía - FAQ's». Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://www.mites.gob.es/fogasa/faqs.html>
- [3] «Salario para Desarrollador Full Stack en España - Salario Medio», Talent.com. Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://es.talent.com/salary>
- [4] Movistar, «Fibra óptica en casa sin permanencia. Movistar Internet en casa». Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://www.movistar.es/internet/>
- [5] J.Pomeyrol, «¿Mal rendimiento con WSL? Instala Linux, pero no desactives el antivirus - MuyLinux». Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://www.muylinux.com/2019/02/14/mal-rendimiento-con-wsl/>
- [6] «Downloads – Oracle VM VirtualBox». Accedido: 8 de julio de 2024. [En línea]. Disponible en: <https://www.virtualbox.org/wiki/Downloads>
- [7] «Mariaojruiz/Sibling-Rewiring: Proyecto de la universidad de Burgos tutelado por José Manuel Galán y Virginia Ahedo. Aplicación que realizará las modificaciones necesarias en las aulas para disminuir el número de contagios entre los alumnos al máximo.» Accedido: 20 de junio de 2024. [En línea]. Disponible en: <https://github.com/Mariaojruiz/Sibling-Rewiring>
- [8] M. J. Gamez, «Portada», Desarrollo Sostenible. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.un.org/sustainabledevelopment/es/>