ESCUELA SUPERIOR POLITÉCNICA AGROPECUARIA DE MANABÍ MANUEL FÉLIX LÓPEZ

CARRERA DE COMPUTACIÓN

INFORME DE TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

MECANISMO: SISTEMATIZACIÓN DE EXPERIENCIAS PRÁCTICAS DE INVESTIGACIÓN Y/O INTERVENCIÓN

TEMA:

ALGORITMOS GÉNETICOS PARALELIZADOS

AUTORES:

JESÚS ALBERTO GARCÍA MERA

ANDERSON JOEL ZAMBRANO MOREIRA

TUTOR:

MGST. VÍCTOR JOEL PINARGORTE BRAVO

CALCETA, OCTUBRE DE 2022

DECLARACIÓN DE AUTORÍA

Yo **ANDERSON JOEL ZAMBRANO MOREIRA**, con cédula de ciudadanía **0804360030**; y **JESÚS ALBERTO GARCÍA MERA**, con cédula de ciudadanía **1312114166**, declaramos bajo juramento que el Trabajo de Integración Curricular titulado: **ALGORITMOS GENÉTICOS PARALELIZADOS** es de nuestra autoría, que no ha sido previamente presentado para ningún grado o calificación profesional, y que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, concedemos a favor de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos, conservando a mi favor todos los derechos patrimoniales de autor sobre la obra, en conformidad con el Artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación.

ANDERSON J. ZAMBRANO MOREIRA
CC: 0804360030

JESÚS A. GARCÍA MERA
CC: 1312114166

AUTORIZACIÓN DE PUBLICACIÓN

ANDERSON JOEL ZAMBRANO MOREIRA, con cédula de ciudadanía **0804360030** y **JESÚS ALBERTO GARCÍA MERA**, con cédula de ciudadanía **1312114166**, autorizamos a la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, la publicación en la biblioteca de la institución del Trabajo de Integración Curricular titulado: **ALGORITMOS GÉNETICOS PARALELIZADOS,** cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y total autoría.

ANDERSON J. ZAMBRANO MOREIRA
CC: 0804360030

JESÚS A. GARCÍA MERA
CC: 1312114166

CERTIFICACIÓN DEL TUTOR

VÍCTOR JOEL PINARGOTE BRAVO, certifica haber tutelado el Trabajo de titulación: **ALGORITMOS GÉNETICOS PARALELIZADOS**, que ha sido desarrollado por **JESÚS ALBERTO GARCÍA MERA Y ANDERSON JOEL ZAMBRANO MOREIRA** previo a la obtención del título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

VÍCTOR JOEL PINARGOTE BRAVO

CC: 1310867930

TUTOR

APROBACIÓN DEL TRIBUNAL

Los suscritos integrantes del Tribunal correspondiente, declaramos que hemos **APROBADO** el Trabajo de Integración Curricular titulado: **ALGORITMOS GÉNETICOS PARALELIZADOS** que ha sido desarrollado por **JESÚS ALBERTO GARCÍA MERA** y **ANDERSON JOEL MOREIRA ZAMBRANO** previo a la obtención del título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

ING. LUIS C. CEDEÑO VARALEZO

CC: 1306246651

PRESIDENTE DEL TRIBUNAL

ING. ALFONSO T. LOOR VERA

CC: 1311655938

MIEMBRO DEL TRIBUNAL

ING. ÁNGEL A. VÉLEZ MERO

CC: 1308648565

MIEMBRO DEL TRIBUNAL

AGRADECIMIENTO

En primer lugar, expresamos nuestro agradecimiento infinito a Dios por permitirnos haber culminado los estudios con salud.

De igual forma, a cada uno de nuestros familiares, amigos que, sin duda alguna, fueron un pilar fundamental para poder alcanzar nuestra meta de ser ingenieros.

Así mismo al gran cuerpo de docente que con su ardua labor logran forjar estudiantes profesionales de alta calidad. Agradecemos a nuestro. Tutor Ing. Víctor Pinargote por brindarnos su apoyo y guiarnos en cada una de las etapas del proceso de titulación.

A la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López que me dio la oportunidad de crecer como ser humano a través de una educación superior de calidad con compromiso ético y social, y en la cual he forjado mis conocimientos profesionales día a día.

DEDICATORIA

Esta tesis se la dedico a Dios, quien ha sido mi guía en cada momento, concediéndome fortaleza, sabiduría, perseverancia y amor incondicional en este largo caminar.

A mis amigos y compañeros por la confianza y apoyo moral, por su paciencia y conocimiento que me brindaron desde el inicio de mi etapa de formación profesional, los llevaré siempre presentes.

A todo el grupo de profesores que me forjó como profesional, son unas excelente personas y profesionales, gracias por todo su apoyo profesional.

A mis padres y familiares por inculcarme valores que me han formado como persona; gracias por su esfuerzo, cariño y apoyo emocional.

A mi hija y mi madre que son mi fortaleza, mi motor principal para salir adelante, gracias por la paciencia que me tuvieron cada día, son uno de mis motivos para cumplir las metas que me propongo y así darle el mejor ejemplo como persona y profesional.

A mi esposa, por su apoyo incondicional, por siempre estar ahí acompañándome en los buenos y malos momentos, gracias por ser parte de mi vida y querer siempre lo mejor para nuestra familia.

Por último, dedico este trabajo de titulación a unas personas maravillosas el Ingeniero Galo García y la Ingeniera Kaviria Flores de Válgaz que desde siempre quisieron que me formara como profesional, gracias infinitas por apoyarme en cada momento que lo necesité y en el cielo a mi abuelito Luis García que desde el allá me acompaña en todo momento, siempre serán indispensables en mi vida; los llevo siempre en mi corazón.

JESÚS ALBERTO GARCÍA MERA

DEDICATORIA

Este proyecto se lo dedico a Dios por guiarme siempre en la vida, por bridarme salud y fuerzas para nunca rendirme.

A mis padres que son mi mayor motivación y mi mayor alegría, se los dedico por siempre confiar en mí y haberme dado la oportunidad de estudiar, por su gran sacrificio y su incondicional apoyo constante que me brindan siempre, sin ellos mis metas no serían posibles.

A mi hermana, novia, tíos y primos por siempre estar cerca de mí y de mi familia, por bridarme su apoyo moral, aconsejarme y siempre inculcándome buenos valores.

A mis compañeros por la confianza y apoyo moral, por su paciencia y conocimiento que también me brindaron desde el inicio de mi etapa de formación profesional.

Por último, dedico este trabajo de titulación a mis amigos de toda la vida, con los que compartimos lejos de casa alegrías, tristeza y que luego de tanto cada uno está cobrando los frutos del esfuerzo.

ANDERSON JOEL ZAMBRANO MOREIRA

CONTENIDO GENERAL

CARATULA.....i
DECLARACIÓN DE AUTORÍA ii
AUTORIZACIÓN DE PUBLICACIÓN iii
CERTIFICACIÓN DEL TUTOR iv
APROBACIÓN DEL TRIBUNAL.....v
AGRADECIMIENTO vi
DEDICATORIA vii
DEDICATORIA viii
CONTENIDO GENERAL.....ix
CONTENIDO DE CUADROS xiii
CONTENIDO DE FIGURAS xiii
RESUMEN xiv
PALABRAS CLAVE xiv
ABSTRACT.....xv
KEY WORDS xv
CAPÍTULO I. ANTECEDENTES 1
1.1. DESCRIPCIÓN DE LA INSTITUCIÓN 1
1.2. DESCRIPCIÓN DE LA INTERVENCIÓN 2
1.3. OBJETIVOS.....4
1.3.1. OBJETIVO GENERAL.....4
1.3.2. OBJETIVOS ESPECÍFICOS.....4
CAPÍTULO I. 5
CAPÍTULO II. DESARROLLO METODOLÓGICO.....5
2.1. FASE 1: REALIZAR EL LEVANTAMIENTO DE REQUERIMIENTOS E INFORMACIÓN NECESARIA PARA LA IMPLEMENTACIÓN Y PARALELIZACIÓN DEL ALGORITMO (PLANIFICACIÓN) 5
2.2. FASE 2: DISEÑAR UN PROTOTIPO DEL ALGORITMO GENÉTICO EN PYTHON Y UNA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS (DISEÑO).....6
2.2.1. DISEÑO DEL ALGORITMO GENÉTICO 6
2.2.1.1. POBLACIÓN INICIAL 7
2.2.1.2. FUNCIÓN DE APTITUD 8
2.2.1.3. SELECCIÓN.....8
2.2.1.3.1. SELECCIÓN POR TORNEO 8

2.2.1.4. CRUCE.....9
2.2.1.5. MUTACIÓN 9
2.2.1.6. REEMPLAZO 9
2.2.2. DISEÑO DE LA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS.....9
2.3. FASE 3: CODIFICACIÓN.....10
2.3.1. CONSTRUIR EL ALGORITMO GENÉTICO CON LOS MÓDULOS Y PROCESOS QUE PERMITAN PARALELIZACIÓN. 10
2.3.2. DESARROLLAR MÓDULO WEB PARA LA EVALUACIÓN DE LAS INSTANCIAS. 10
2.4. FASE 4: PRUEBAS.....11
2.4.1. EVALUAR EL RENDIMIENTO DEL ALGORITMO PARALELIZADO. 11
CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA.....12
3.1. FASE 1: REALIZAR EL LEVANTAMIENTO DE REQUERIMIENTOS E INFORMACIÓN NECESARIA PARA LA IMPLEMENTACIÓN Y PARALELIZACIÓN DEL ALGORITMO (PLANIFICACIÓN) 12
3.2. FASE 2: DISEÑAR UN PROTOTIPO DEL ALGORITMO GENÉTICO EN PYTHON Y UNA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS (DISEÑO).....13
3.2.1. DISEÑAR UN ALGORITMO GENÉTICO EN PYTHON 13
3.2.2. DISEÑO DE LA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS.....15
3.3. FASE 3: CODIFICACIÓN.....16
3.3.1 CONSTRUIR EL ALGORITMO GENÉTICO 16
3.3.1.1. FUNCIÓN POBLACIÓN INICIAL 16
3.3.1.2. FUNCIÓN DE APTITUD 17
3.3.1.3. FUNCIÓN DE SELECCIÓN.....17
3.3.1.4. FUNCIÓN DE CRUCE.....18
3.3.1.5. FUNCIÓN DE MUTACIÓN 21
3.3.1.6. FUNCIÓN DE REEMPLAZO 21
3.3.2. IDENTIFICAR LOS MÓDULOS Y PROCESOS QUE PERMITAN PARALELIZACIÓN 22
3.3.3. DESARROLLAR MÓDULO WEB PARA LA EVALUACIÓN DE LAS INSTANCIAS. 23
3.4. FASE 4: PRUEBAS.....24

3.4.1. EVALUAR EL RENDIMIENTO DEL ALGORITMO PARALELIZADO. 24
CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES.....26
4.1. CONCLUSIONES 26
4.2. RECOMENDACIONES 26
BIBLIOGRAFÍA 28
ANEXOS 30
1. INTRODUCCIÓN 42
1.1. IDENTIFICACIÓN DEL SISTEMA 42
1.2. OBJETIVO 42
1.3. ALCANCE 42
1.4. PERSONAL INVOLUCRADO 43
1.5. NOTACIONES Y DEFINICIONES 43
1.5.1. NOTACIONES 43
1.5.2. DEFINICIONES 44
1.6. REFERENCIAS 44
2. DESCRIPCIÓN GENERAL 44
2.1. PERSPECTIVAS DEL PRODUCTO 45
2.2. FUNCIONES DEL PRODUCTO 45
2.3. CARACTERÍSTICAS DE USUARIO 45
2.4. RESTRICCIONES 46
2.5. SUPOSICIONES Y DEPENDENCIAS 46
2.6. REQUISITOS FUTUROS 46
DIAGRAMAS 47
2.6.1. DIAGRAMA DE CASO DE USO 47
2.6.2. DIAGRAMA DE CLASES DEL SISTEMA 48
2.6.3. DIAGRAMA DE BASE DE DATOS DEL SISTEMA 49
3. REQUERIMIENTOS ESPECÍFICOS 49
3.1. REQUERIMIENTOS DE INTERFACES EXTERNAS 49
3.1.1. INTERFACES DE USUARIO 49
3.1.2. INTERFACES CON EL HARDWARE 50
3.1.3. INTERFACES SOFTWARE 50
3.2. REQUERIMIENTOS FUNCIONALES 51
3.2.1. REGISTROS DE USUARIOS. 51

3.2.2. VERIFICAR AUTENTICACIÓN DE USUARIOS. 51
3.2.3. INGRESO DE PRODUCTOS 52
3.2.4. INGRESO DE PRECIOS 53
3.2.5. INGRESO DE INSTANCIAS 54
3.3. REQUERIMIENTOS NO FUNCIONALES 54
3.3.1. REQUERIMIENTOS DE RENDIMIENTO. 54
3.3.2. REQUISITO DE DISEÑO 54
3.3.3. ATRIBUTOS DEL SISTEMA.....55

CONTENIDO DE CUADROS

CUADRO 3.1: REPORTE DE LOS DATOS OBTENIDOS 12
CUADRO 3.2: HISTORIAS DE USUARIO 15
CUADRO 3.3: CÓDIGO DE GENERAR POBLACIÓN INICIAL 16
CUADRO 3.4: CÓDIGO DE LA FUNCIÓN FITNESS O DE APTITUD 17
CUADRO 3.5: CÓDIGO DE LA FUNCIÓN SELECCIÓN 18
CUADRO 3.6: CÓDIGO DE LA FUNCIÓN DE CRUCE 19
CUADRO 3.7: CÓDIGO DE LA FUNCIÓN COMPROBAR_HIJO 20
CUADRO 3.8: CÓDIGO DE LA FUNCIÓN MEJOR_LOTE 20
CUADRO 3.9: CÓDIGO DE LA FUNCIÓN DE MUTACIÓN 21
CUADRO 3.10: CÓDIGO DE LA FUNCIÓN DE REEMPLAZO 21
CUADRO 3.11: DECLARACIÓN DE CUDA 22
CUADRO 3.12: TRANSFERENCIA DE TENSORES DE CPU A GPU 22
CUADRO 3.13: FUNCIÓN FITNESS CON GPU 23

CONTENIDO DE FIGURAS

FIGURA 2.1: FASES DE LA METODOLOGÍA XP 5
FIGURA 2.2: ESTRUCTURA DE UN ALGORITMO GENÉTICO 7
FIGURA 3.1: FLUJO DEL ALGORITMO GENÉTICO 14
FIGURA 3.3: DIAGRAMA DE CASO DE USO 16
FIGURA 3.4: MÉTODO DE SELECCIÓN POR TORNEO 17
FIGURA 3.5: DEMOSTRACIÓN DE CRUCE DE TRES PUNTOS 18
FIGURA 3.6: DEMOSTRACIÓN DE INDIVIDUOS CRUZADOS 19
FIGURA 3.7: MÓDULO PARA CREAR INSTANCIAS 24

RESUMEN

La presente investigación tuvo como objetivo optimizar mediante técnicas de computación paralela, el tiempo de ejecución de un algoritmo genético aplicado a la planificación de la programación agrícola, se utilizaron herramientas como Google Colab, Python 3.8, la librería Torch, en la fase de codificación, no fue posible paralelizar todos los módulos, por lo que se decidió paralelizar los más complejos en cuanto a número de operaciones, en la fase de resultados se obtuvieron excelentes tiempos de ejecución, donde se destaca una mejoría del 200%, esto para soluciones mayores a 50 genes (Productos Agrícolas) frente al algoritmo no paralelizado. Se implementó un módulo web para generar nuevas instancias, mismo que recibió datos como: número de ciclos para la cosecha, área disponible, fecha de inicio del cultivo, población (número de soluciones), número de productos (número de productos que se desean formen parte de la solución), y este genera como resultado una población de soluciones con los mayores márgenes de beneficios.

PALABRAS CLAVE

Optimización agrícola, algoritmos paralelos, algoritmos genéticos, planificación agrícola.

ABSTRACT

The objective of this research was to optimize the execution time of a genetic algorithm applied to the planning of agricultural programming using parallel computing techniques, tools such as Google Colab, Python 3.8, Torch library, where applied, it was not possible to parallelize all the modules in the coding phase, so the most complex in terms of number of operations had to be parallelized, among the results excellent improvements were obtained with times greater than 200% for solutions greater than 50 genes (Agricultural Products) compared to the none-parallelized algorithm. A web module was implemented to generate new instances, which received parameters such as: number of cycles for harvest, available area, date of start of cultivation, population (number of final solutions), number of products (number of products desired to be part of the solution), and this results in a population of solutions with the highest profit margins.

KEY WORDS

Agricultural optimization, parallel algorithms, genetic algorithms, agricultural planning

CAPÍTULO I. ANTECEDENTES

1.1. DESCRIPCIÓN DE LA INSTITUCIÓN

La Escuela Superior Politécnica Agropecuaria de Manabí “Manuel Félix López” se encuentra localizada en la ciudad de Calceta perteneciente al cantón Bolívar, provincia de Manabí, fue fundada en abril de 1999 con el propósito de participar junto a otras entidades, en el auge y avance de la provincia de Manabí y del país, a través de la enseñanza universitaria, la investigación científica y el emprendimiento. El contexto rural y socioeconómico manabita, con un alto potencial productivo, resultó determinante para la elección de las carreras, la ESPAM MFL inicia con las carreras de Agroindustrias, Ingeniería Agrícola y Medicina Veterinaria. Luego mediante un estudio de mercado se crea la carrera de Computación, Administración de Empresas, Administración Pública y a partir del año 2007 los estudiantes tienen una nueva opción: Turismo. (Carreño et al., 2016)

La Carrera de Computación es una de las carreras ofertadas en la ESPAM MFL cuya misión es la “formación de Profesionales íntegros que conjuguen ciencia, tecnología y valores en su accionar, comprometidos con la comunidad en el manejo adecuado de programas y herramientas computacionales de última generación”; y su visión “ser referentes en la formación de profesionales de prestigio en el desarrollo de aplicaciones informática y soluciones de hardware” (*ESPAM MFL*, 2021). En esta carrera existen tres unidades, una de las cuales es la Unidad de Docencia, Investigación y Vinculación (UDIV), quien es la encargada de ofrecer una formación sólida, teórica, metodológica y práctica a los estudiantes de la ESPAM MFL en el análisis de problemas relacionados con los procesos del computador, auditoría y redes. (UNIDAD DE INFRAESTRUCTURA DE LA CARRERA DE COMPUTACION DE LA ESPAM MFL, 2020)

La ESPAM MFL es una institución de educación superior que pretende institucionalizar y sistematizar la investigación en la institución, de manera interactiva y multidisciplinaria para propiciar la creación, adaptación, generación y transferencia de tecnologías, en la búsqueda de alternativas viables e

innovadoras para la solución de problemas prioritarios y productivos de la región y el país.

La ESPAM MFL aporta a la investigación a través de sus diferentes grupos de investigación, cuyos proyectos se deben enmarcar en sus respectivas líneas de investigación, la presente propuesta es parte de un proyecto investigativo presentado por el grupo de investigación SISCOM de la Carrera de Computación, que tributa a la línea de investigación: Soluciones computacionales para el sector agroproductivo y de servicios.

1.2. DESCRIPCIÓN DE LA INTERVENCIÓN

Los algoritmos genéticos (AGs) son métodos de búsqueda basados en los principios de selección natural y la genética. Han sido amplificados con éxito a múltiples problemas científicos y de ingeniería. En muchas aplicaciones prácticas los AGs encuentran una buena solución en tiempo razonable. (Hidalgo et al., 2006)

Existen varios estudios para mejorar el rendimiento de los AGs y una de las alternativas más interesantes es el uso de implementaciones paralelas. El éxito de los AGs paralelos reside en la reducción de tiempo requerido para encontrar una solución aceptable en problemas de alta complejidad, donde los AGs podrían llegar a tardar 3 o 6 veces más para dar una solución(Martínez Vargas et al., 2016)

A pesar de su simplicidad operacional los AGs paralelos están controlados por múltiples parámetros que afectan a la eficiencia y a la calidad de las soluciones encontradas. Fijar estos parámetros correctamente, buscando un equilibrio entre ellos, es fundamental para obtener buenas soluciones de forma más rápida, intentando no desaprovechar recursos de computación. Se introduce el término “paralelización” como la implementación de un algoritmo en paralelo.

La inclusión de técnicas paralelas en la definición de algoritmos ha sido muy importante en los recientes diseños de este tipo de búsqueda y optimización. De esta forma, la especial adecuación de estos algoritmos para trabajar sobre

problemas de elevada complejidad puede verse mejorada si el diseño y la implementación son paralelos. (Martínez Vargas et al., 2016)

En el presente trabajo de integración curricular se desarrolló un algoritmo genético desde cero, para luego identificar las fases o procesos susceptibles a paralelización, con el objetivo de mejorar tiempo y costo computacional, también se desarrolló un módulo web que permite evaluar posibles soluciones heurísticas y configurar escenarios para encontrar soluciones potenciales. Por tal motivo se utilizó una estrategia ágil para el desarrollo del presente proyecto.

El presente trabajo aporta una experiencia a la línea de investigación de la Carrera de Computación de la ESPAM MFL (Soluciones computacionales para el sector agro productivo y de servicios), concretamente a los componentes 3 (Desarrollo del Algoritmo Genético) y 4 (Paralelización del Algoritmo) del proyecto de investigación institucional titulado: Modelo de optimización para la programación de la producción agrícola basado en Algoritmos Genéticos, el mismo que es auspiciado por la Carrera de Computación de la ESPAM se encuentra en ejecución. El presente proyecto aporta directamente a su línea de investigación generando fuentes de conocimientos y alternativas de implementación en la programación de los cultivos agrícolas para optimizar diferentes variables como la producción, beneficio, etc. y en el ámbito informático se optimizaron los tiempos de cómputo utilizando técnicas de paralización en algoritmos Genéticos, obteniendo así mejores resultados en menor tiempo y con ello ayudar a resolver problemas de alta complejidad computacional que no se suelen abordar, puesto que se puede tardar semanas o incluso meses en obtener buenos resultados con las estrategias existentes en la actualidad.

Como beneficiarios del proyecto institucional se identifica al sector agroproductivo de la zona 4 del país, donde se implementarán las soluciones obtenidas por el algoritmo en las zonas destinadas al cultivo de la ESPAM MFL (esto dentro del proyecto institucional), por tal motivo, se identifica como cliente final el agricultor.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Implementar el algoritmo genético basado en (GPU/TPU) para alcanzar máximos globales optimizando el tiempo de cómputo.

1.3.2. OBJETIVOS ESPECÍFICOS

- Realizar el levantamiento de requerimientos e información necesaria para la implementación y paralelización del algoritmo.
- Diseñar un prototipo del algoritmo genético en Python y una interfaz Web para la visualización de resultados.
- Construir el algoritmo genético con los módulos y procesos que permitan paralelización.
- Desarrollar módulo web para la evaluación de las instancias.
- Evaluar el rendimiento del algoritmo paralelizado.

CAPÍTULO II. DESARROLLO METODOLÓGICO

En el desarrollo de la paralelización del algoritmo genético para optimizar la programación de la producción agrícola se plantearon cinco objetivos específicos que se adaptan a la metodología XP la cual cuenta con fases como: planificación, diseño, codificación y pruebas.

Figura 0.1: Fases de la metodología XP

2.1. FASE 1: REALIZAR EL LEVANTAMIENTO DE REQUERIMIENTOS E INFORMACIÓN NECESARIA PARA LA IMPLEMENTACIÓN Y PARALELIZACIÓN DEL ALGORITMO (PLANIFICACIÓN)

Para el desarrollo de esta fase fue necesario reunirse con el director y codirector del proyecto institucional antes mencionado, donde se definieron los componentes de la intervención; desarrollo de un algoritmo genético para la programación agrícola (componente 3), mismo que tiene como entrada: productos, ciclos, área, fecha de inicio. Módulo web donde se almacenan los parámetros y se ejecutan las instancias del algoritmo genético paralelizado (componente 4).

En esta fase se realizó el levantamiento de información mediante las páginas oficiales de las siguientes instituciones nacionales: Instituto Nacional de Investigaciones Agropecuarias (INIAP) y Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP) esta última por medio del Sistema de Información Pública Agropecuaria del Ecuador (SIPA).

El SIPA es el Sistema de Información Pública Agropecuaria del Ecuador, es un servicio integrado de información estadística y geográfica, que sirve como insumo para la toma de decisiones del sector agropecuario, además, permite generar conocimiento científico-académico en el área y exponer la situación del país en el ámbito agropecuario (Precios Mayoristas, n.d.).

En las páginas antes mencionadas se realizó la recolección de datos en base a los objetivos del proyecto y se recolectó información que ayude a construir un conjunto de datos identificando ciertos criterios como son: productos transitorios, precios, rendimiento de los productos y periodos de maduración de cada producto. Cada parámetro se analizó, se organizó de acuerdo a lo requerido y se realizó la preparación del conjunto de datos que servirá para la construcción del algoritmo genético.

Según el INEC (2020) Manabí lidera las provincias con mayor superficie de labor agropecuaria, cuenta con cultivos permanentes de mayor producción y cultivos transitorios de mayor producción, entre los cultivos permanentes están el plátano y la palma africana mientras que en los transitorios están el maíz duro seco y el arroz en cáscara.

2.2. FASE 2: DISEÑAR UN PROTOTIPO DEL ALGORITMO GENÉTICO EN PYTHON Y UNA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS (DISEÑO)

2.2.1. DISEÑO DEL ALGORITMO GENÉTICO

En esta fase se diseñó cada uno de los diferentes componentes o etapas del algoritmo genético que según Martínez (2020) la primera etapa de cualquier algoritmo genético consiste en establecer la relación entre el fenotipo y el

genotipo. Es decir, hay que decidir cómo se va a codificar el espacio de búsqueda, lo cual resulta fundamental pues afecta a todo el desarrollo del algoritmo.

Un algoritmo genético está formado por una población inicial donde cada individuo está codificado en un cromosoma, donde cada cromosoma tiene una cadena de genes, estos a su vez pueden estar codificados por números o letras en diferentes posiciones. En la **Figura 0.2:** Estructura de un algoritmo genético podemos visualizar el esquema de un algoritmo genético.

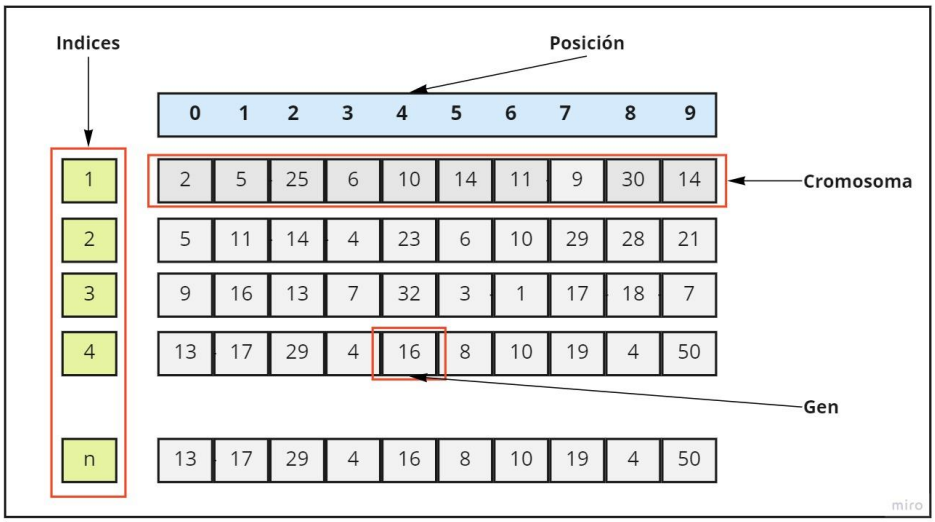


Figura 0.2: Estructura de un algoritmo genético

Fuente: Los autores

Los Algoritmos Genéticos cuentan con el grupo de elementos básicos de todo individuo en su camino trascendencia, como son el nacimiento, la reproducción (cruza), mutación y selección de individuos, lo que lleva a los individuos más capacitados a la siguiente generación y se extingue a los menos competentes. (Hernández Domínguez, 2010)

Cada una de las etapas del algoritmo genético se describirán a continuación:

2.2.1.1. POBLACIÓN INICIAL

Esta fase es un conjunto de individuos que está construida por posibles soluciones, este conjunto de individuos es generado de manera aleatoria.

2.2.1.2. FUNCIÓN DE APTITUD

La función de coste o de aptitud está relacionada con el valor de la función para cada uno de los individuos, en nuestro caso de querer maximizar, cuanto mayor sea el valor de la función mayor será su fitness.

Erazo (2022) considera que esta función también nos permite evaluar todas las posibles soluciones en un espacio de búsqueda determinado, la función de coste es una de las partes principales de los algoritmos de programación genética.

2.2.1.3. SELECCIÓN

Durante la primera fase del algoritmo, se seleccionan n individuos de la población inicial P0, los cuales serán los candidatos a ser padres de la siguiente generación, posteriormente se cruza el material genético de los padres seleccionados, produciendo con esto, algunos descendientes los cuales una vez mutados constituirán parcial o totalmente la siguiente población de individuos. (Arias & Londoño, 2009)

Existen diversas técnicas de selección en los algoritmos genéticos, para problema presentado es necesario seleccionar alguna que demande un bajo costo computacional, puesto que el algoritmo se ejecutará desde una interfaz web y se requieren tiempos de ejecución no muy prolongados, en virtud de lo antes mencionado se escogió la selección por torneo ya que a diferencia de la selección por probabilidad, esta tiene una menor complejidad computacional y por tanto se adapta a las necesidades de nuestro proyecto, las características de esta selección se describen a continuación:

2.2.1.3.1. SELECCIÓN POR TORNEO

En la selección por torneo se escogen de forma aleatoria un número determinados de individuos (padres) entre los cuales el que tenga el valor de aptitud más alto se reproduce. Valencia (1997) nos dice que “este método de selección no se basa en valores esperados y no requiere por lo tanto de un algoritmo de muestreo”.

2.2.1.4. CRUCE

Consiste en mezclar el material genético de dos cromosomas que serán progenitores y que han sido seleccionados (utilizando cualquier técnica). Por lo general, los cromosomas son aleatoriamente divididos y mezclados, por lo que ciertos genes de los descendientes provienen de un progenitor, mientras otros provienen del otro, es decir, los descendientes tendrán en su genotipo información de ambos progenitores. (Espitia & Mendoza, 2021)

En nuestra implementación utilizamos el cruce por tres puntos ya que para el tamaño de nuestros individuos es la forma más óptima para generar nuevos descendientes con un buen valor de aptitud.

2.2.1.5. MUTACIÓN

La función de mutación transforma en nuestro caso el individuo (hijo) donde se escogerán aleatoriamente dos posiciones de los genes del cromosoma, para así intercambiarlas por otras. Molina et al., (2014) nos dicen que “hay muchas variantes para realizar una mutación, pero usualmente esta involucra uno o varios cambios sobre uno o varios rasgos de cada individuo”

2.2.1.6. REEMPLAZO

Una parte importante de los algoritmos genéticos es la formación de la siguiente generación, al final de cada interacción en los métodos de selección y cruce la población ha aumentado ya que según Corral Sastre (2018) “Esto debe ser evitado para que no se dispare el número de individuos y aumente de forma innecesaria el tiempo de cómputo del algoritmo” por lo cual con esta fase de reemplazo reduciremos la población al tamaño establecido.

2.2.2. DISEÑO DE LA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS

En esta fase también se diseñó el módulo de la interfaz web mediante los requisitos necesarios, mismos que son parte del algoritmo; es decir se diseñó una web que tiene como entrada datos ingresados por el usuario, los cuales

serán recibidos por el algoritmo para luego visualizar un resultado, y a su vez se realizó en un diagrama de caso de uso, el cual es un sistema de secuencia de acciones que un lleva un sistema a cabo de dar un resultado de valor observable.

Según Pérez A. (2011) **"**solo se efectúa el diseño necesario para cumplir con los requerimientos actuales, es decir, no se abordan requerimientos futuros". Por lo que se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto y la complejidad innecesaria y el código extra debe ser removido.

2.3. FASE 3: CODIFICACIÓN

2.3.1. CONSTRUIR EL ALGORITMO GENÉTICO CON LOS MÓDULOS Y PROCESOS QUE PERMITAN PARALELIZACIÓN.

La fase de desarrollo de XP se realiza en parejas, esta es estandarizada por el equipo de trabajo, además, se realizan liberaciones frecuentes de versiones. También se realizan las pruebas funcionales, donde se evalúa lo planteado si fue implementado correctamente. (Jiménez Builes et al., 2019)

Se codificó el algoritmo genético de acuerdo a las etapas descritas en la fase de diseño, la codificación se hizo en el lenguaje de programación Python utilizando Google Colab, también identificamos las etapas que permiten paralelización y se utilizó la librería Torch para paralelizar mediante GPU.

2.3.2. DESARROLLAR MÓDULO WEB PARA LA EVALUACIÓN DE LAS INSTANCIAS.

Debido a la construcción del algoritmo en Python, para el desarrollo del módulo web se desarrolló en el framework de Django, según Vidal-Silva et al., (2021) Django "representa un marco de trabajo para el desarrollo rápido de sistemas de información web con Python".

El análisis de la información de los datos recolectados nos permite crear las historias de usuario. En XP la gestión de requisitos es "extremadamente" simple,

el cliente escribe y prioriza las historias de usuario que expresan requisitos funcionales y no funcionales del sistema. (Sánchez et al., 2003)

La creación del módulo web conlleva a la respectiva y correcta documentación para lo cual se lo hizo por el medio de la especificación de requerimientos (ERS). Los ERS son una actividad primordial en el desarrollo de sistemas de software que orientan en todo momento las acciones del equipo de trabajo; la manera de realizarla varía de acuerdo al proyecto y a las prácticas adoptadas por los desarrolladores. (Ramón et al., 2012)

2.4. FASE 4: PRUEBAS

2.4.1. EVALUAR EL RENDIMIENTO DEL ALGORITMO PARALELIZADO.

La programación extrema anima a probar constantemente tanto como sea posible, ya que esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. (Gutiérrez et al., n.d.)

CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA

3.1. FASE 1: REALIZAR EL LEVANTAMIENTO DE REQUERIMIENTOS E INFORMACIÓN NECESARIA PARA LA IMPLEMENTACIÓN Y PARALELIZACIÓN DEL ALGORITMO (PLANIFICACIÓN)

Como punto de partida para el desarrollo del algoritmo genético se realizó la recolección de datos mediante las páginas Webs del INIAP Y MAGAP (Anexo 1) en las páginas del INIAP se recolectó los datos de los productos transitorios, los periodos de maduración y el rendimiento y en el MAGAP los diferentes precios de cada semana de los productos transitorios a lo largo del año 2020 y 2021. En el siguiente **Cuadro 0.1** se referencia los parámetros a considerar para la investigación y para la creación del conjunto de datos.

Cuadro 0.1: Reporte de los datos obtenidos

	Nombre	Tiempo de Madurez Promedio (Semanas)	Rendimiento Kg/m2	Precio ($/Kg) Promedio (2020/2021)
1	Zapallo	15	1,62	0,22
2	Arveja	52	0,7	0,86
3	Cebolla colorada	13	3,4	0,41
4	Cebolla Junta	56	3,95	0,67
5	Cilantro	20	1,14	0,47
6	Cebada	11	0,024	0,51
7	Maíz	9	0,6	1,74
8	Frejol	9	0,33	0,72
9	Habichuela	11	1,1	0,58
10	Lechuga	15	0,8	0,28
11	Perejil	14	1,75	0,29
12	Pimiento	9	1,38	0,87
13	Tomate	54	2	0,46

14	Banano	19	4	0,25
15	Naranjilla	44	0,9	1,04
16	Papaya	13	20	0,51
17	Piña	5	2,58	0,46
18	Apio	78	2	0,16
19	Papa	56	1,4	0,34
20	Plátano	18	5,5	0,28
21	Yuca	64	0,7	0,30
22	Maracuyá	41	1,1	0,49
23	Maní	23	0,65	2,08
24	Brócoli	13	1,6	0,42
25	Sandía	15	2,3	0,28
26	Melón	12	3,2	0,78
27	Pepino	10	1,32	0,28
28	Zanahoria	11	4,5	0,24
29	Ajo	17	1,47	2,17
30	Col	18	1,29	0,18
31	Lenteja	15	0,12	1,18
32	Haba	14	0,8	0,32

3.2. FASE 2: DISEÑAR UN PROTOTIPO DEL ALGORITMO GENÉTICO EN PYTHON Y UNA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS (DISEÑO)

3.2.1. DISEÑAR UN ALGORITMO GENÉTICO EN PYTHON

Para el diseño de nuestro algoritmo genético se definió cada uno de los procesos con los que contará para optimizar el tiempo de la programación en la producción agrícola. El proceso que ejecuta el algoritmo genético, desde su inicio, hasta su finalización se aprecia en la **Figura 0.1**.

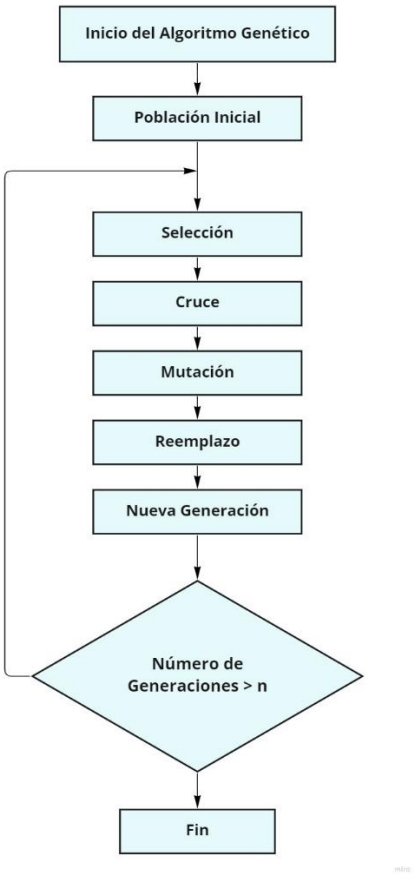


Figura 0.1: Flujo del algoritmo genético

Fuente: Los autores

Dicho flujo de procesos del algoritmo genético cuenta con las siguientes etapas:

- Crea la población Inicial, el tamaño depende de los parámetros ingresados.
- Una vez que tengamos la población Inicial se hace una selección por torneo, se escogen un número aleatorio de individuos y los dos de mejor fitness serán tomados como padres para el cruce.
- Cuando se hayan seleccionado los dos padres se hace un cruce de tres puntos para así obtener los hijos de la nueva generación.
- A continuación, cada hijo será mutado con la condición de mutación.
- Luego de selección y cruce la población es muy grande se hace el reemplazo por el factor elitista.

3.2.2. DISEÑO DE LA INTERFAZ WEB PARA LA VISUALIZACIÓN DE RESULTADOS

Como punto de partida el análisis de la información obtenida y los requisitos funcionales y no funcionales se elaboró las historias de usuario de la forma más detallada de las cuales se definieron por prioridades como se muestra en el siguiente cuadro.

Cuadro 0.2: Historias de usuario

N°	Historias de Usuario	Prioridad
1	Registro de usuario.	Alta/Esencial
2	Validación de cuenta.	Alta/Esencial
3	Información de perfil.	Baja/Opcional
4	Ingreso de precios	Alta/Esencial
5	Ingreso de parámetros	Alta/Esencial
6	Visualización de resultados	Alta/Esencial

Las historias de usuario fueron tomadas como guía para diseñar el diagrama de casos de uso, el cual es una referencia que tendrán los usuarios con el módulo Web. En la **Figura 0.2** se representa de manera general el diagrama de caso de uso. También se realizó el diagrama de clases y el diagrama de modelo entidad-relación que se encuentran en Anexo 3 de especificación de requerimientos de software (ERS).

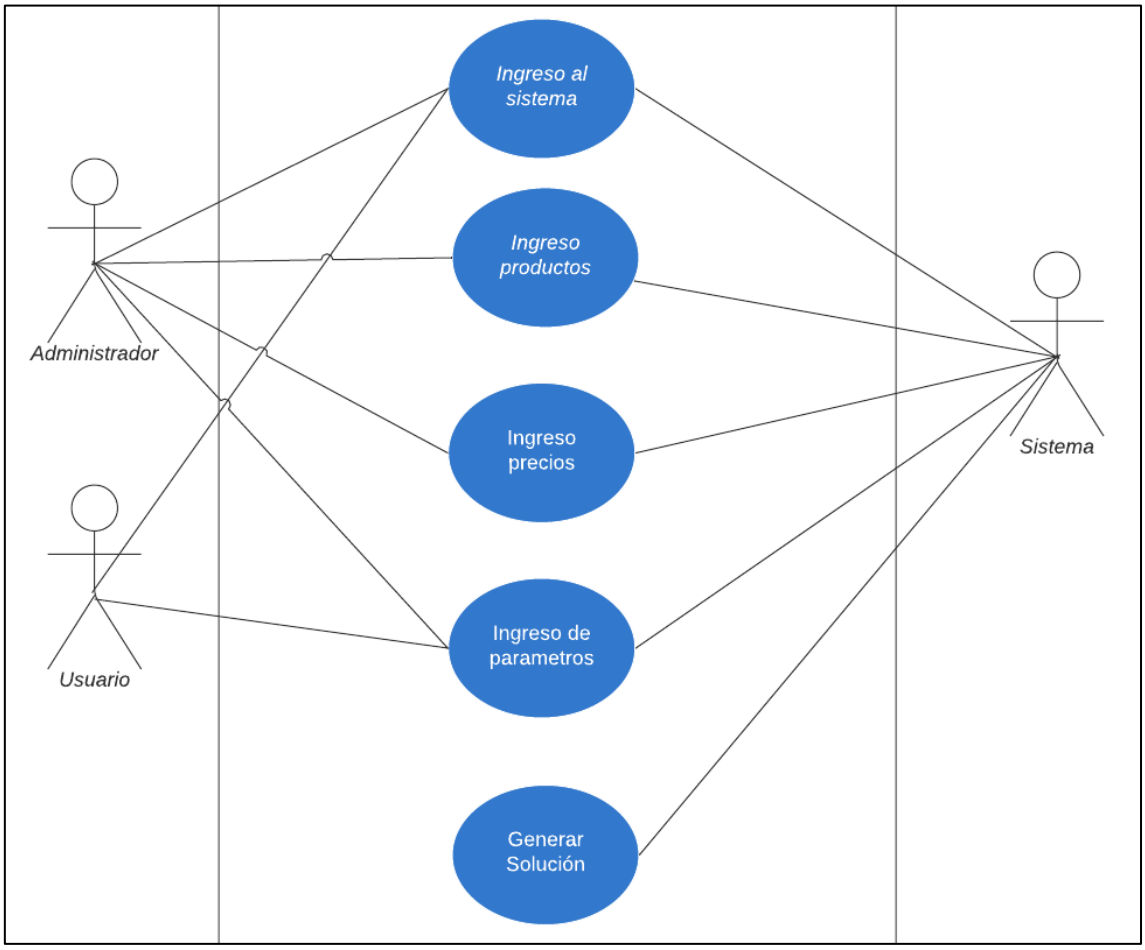


Figura 0.2: Diagrama de caso de uso

Fuente: Los autores

3.3. FASE 3: CODIFICACIÓN

3.3.1 CONSTRUIR EL ALGORITMO GENÉTICO

3.3.1.1. FUNCIÓN POBLACIÓN INICIAL

Se construye con **n** cantidad de individuos (cromosomas), mientras que cada individuo está constituido con los índices de cada producto y con números aleatorios de lotes en base a la restricción del límite de área (genes). Esta población inicial se generará de manera aleatoria, el tamaño de la población y de los individuos varía de acuerdo a los parámetros ingresados.

Cuadro 0.3: Código de generar población inicial

```
def individuo(Ni_ciclos_int, n):
```

```
  lst = np.array(Ni_ciclos_int)
  result=np.where(lst!=0)[0].tolist()
  individuo=np.random.choice(result, n, False)
  return individuo

def dist_lotes(n, total):
  dividers = sorted(random.sample(range(50, total,50), n - 1))
  return [a - b for a, b in zip(dividers + [total], [0] + dividers)]

solucion_inicial=[]
for i in range(IND):
  solucion_inicial.append(np.array((individuo(Ni_ciclos_int,
n).tolist(), dist_lotes(n, area))))
```

3.3.1.2. FUNCIÓN DE APTITUD

Una vez construida la población inicial a cada individuo se le asigna un valor de aptitud que está basado en su rendimiento por kg/m2 y su precio por kg.

Cuadro 0.4: Código de la función fitness o de aptitud

```
def fitness(individuo, rendimiento, precio):
  fo=0
  for i,j in zip(individuo[0], individuo[1]):
    fo+=rendimiento[i]*precio[i]*j
  return fo
```

3.3.1.3. FUNCIÓN DE SELECCIÓN

Hacemos una selección por torneo como se lo muestra en la **Figura 0.3**, en el cual se escoge de un numero aleatorio de n individuos que serán los padres para las nuevas generaciones y entre ese número aleatorio de padres retornará el que tenga el mayor valor de aptitud para cruzarlo en la fase siguiente como lo demostramos en el código del **Cuadro 0.5**.

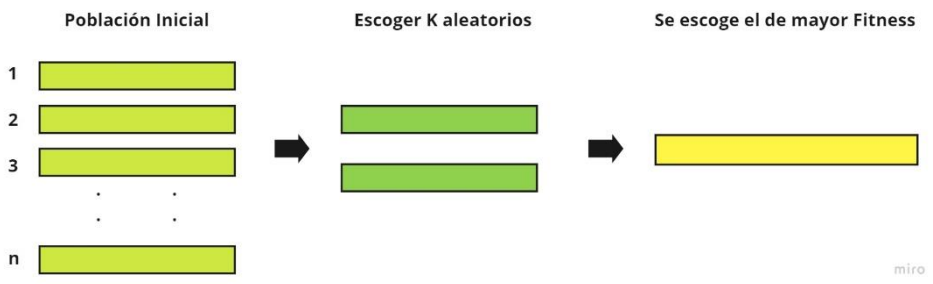


Figura 0.3: Método de selección por torneo

Fuente: Los autores

Cuadro 0.5: Código de la función Selección

```
def seleccionTorneo(poblacionInicial, porcentaje):
    nTorneo = round((len(poblacionInicial) * porcentaje)/100)
    size_poblacion = len(poblacionInicial)
    parents = np.random.choice(size_poblacion, nTorneo, False)
    fo = 0
    index = 0
    for i in parents:
      _fo = fitness(poblacionInicial[i], rendimiento_kg_m2_prod,
precio_kg_t_suma)
      if _fo > fo:
        fo = _fo
        index = i
    return poblacionInicial[index]
```

3.3.1.4. FUNCIÓN DE CRUCE

Teniendo seleccionado dos padres se hace el cruce, en nuestro caso se hace un cruce de tres puntos como se muestra en la **Figura 0.4** y se intercambian los valores como en la **Figura 0.5**.

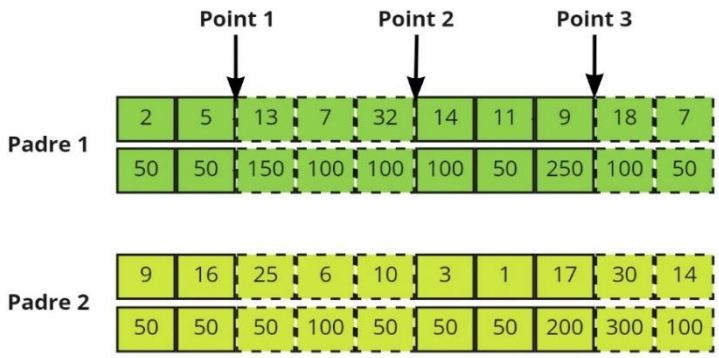


Figura 0.4: Demostración de cruce de tres puntos

Fuente: Los autores

Hijo 1	2	5	25	6	10	14	11	9	30	14
	50	50	50	100	50	100	50	250	300	100
Hijo 2	9	16	13	7	32	3	1	17	18	7
	50	50	150	100	100	50	50	200	100	50

Figura 0.5: Demostración de individuos cruzados

Fuente: Los autores

Una vez que se hace el cruce procedemos a comprobar los hijos como en el **Cuadro 0.7** y a mutarlos, para así retornar los hijos cruzados para la nueva generación como lo muestra el código en el **Cuadro 0.6**.

Cuadro 0.6: Código de la función de cruce

```
def three_point_cross_over(padres, disponibles, mut_fact, k=3):
    cortes=len(padres[0])//4
    pc = [cortes,2*cortes,3*cortes]

    hijo1 =  mutacion(comprobar_hijo
            ([padres[0][0][:pc[0]].tolist() +
              padres[1][0][pc[0]:pc[1]].tolist() +
              padres[0][0][pc[1]:pc[2]].tolist() +
              padres[1][0][pc[2]:].tolist()
              ,padres[0][1][:pc[0]].tolist() +
              padres[1][1][pc[0]:pc[1]].tolist() +
              padres[0][1][pc[1]:pc[2]].tolist() +
              padres[1][1][pc[2]:].tolist()]
              ,prod_disponibles)
              ,disponibles
              ,mut_fact)
    hijo2 =  mutacion(comprobar_hijo
            ([padres[1][0][:pc[0]].tolist() +
              padres[0][0][pc[0]:pc[1]].tolist() +
              padres[1][0][pc[1]:pc[2]].tolist() +
              padres[0][0][pc[2]:].tolist()
              ,padres[1][1][:pc[0]].tolist() +
              padres[0][1][pc[0]:pc[1]].tolist() +
              padres[1][1][pc[1]:pc[2]].tolist() +
              padres[0][1][pc[2]:].tolist()]
              ,prod_disponibles)
              ,disponibles
              ,mut_fact)

    return np.array(hijo1), np.array(hijo2)
```

El objetivo de esta función es comprobar si al momento de hacer la función de cruce uno de los genes se repite en el cromosoma, en el caso de repetirse se elimina y se lo cambia por un nuevo gen y también comprobamos si luego del cruce el número de número de lotes se excedió del límite definido por el usuario esto lo comprobamos en la función mejor_lote como mostramos en el **Cuadro 0.8**.

Cuadro 0.7: Código de la función comprobar_hijo

```
def comprobar_hijo(hijo,prod_disponibles):
    Nodos = prod_disponibles.copy()
    not_in_solution = list(set(Nodos) - set(hijo[0]))
    #Recorremos todos los genes y cuando lo encontremos
    for i in range(len(hijo[0]) ):
        if hijo[0][i] in hijo[0][:i]:
            #Cambiamos el gen y lo eliminamos de la lista
not_in_solution
            hijo[0][i] = not_in_solution.pop(0)

    hijo[1]=mejor_lote(hijo, 100, n, area, rendimiento_kg_m2_prod,
precio_kg_t_suma)
    return hijo
```

Comprobamos si el lote excedió el valor ingresado y generamos un número de aleatorios de lotes, el de mayor fitness lo intercambiamos con el que ha excedido.

Cuadro 0.8: Código de la función mejor_lote

```
def mejor_lote(hijo1, k, n, area, rendimiento_kg_m2_prod,
precio_kg_t_suma):
  fo=0
  mejor_lote=[]
  for j in range(k):
    lotes=dist_lotes(n, area)
    _fo=fitness([hijo1[0],lotes], rendimiento_kg_m2_prod,
precio_kg_t_suma)
    if _fo>fo:
      fo=_fo
      mejor_lote=lotes

  return mejor_lote
```

3.3.1.5. FUNCIÓN DE MUTACIÓN

Cuando se termina de cruzar los padres y obtenemos los nuevos hijos nuestro siguiente paso es mutarlos, esta función de mutación escogerá aleatoriamente dos genes del nuevo hijo y los intercambiará por nuevos genes, esta mutación se hará de acuerdo al factor de mutación.

La función de mutación recibe al hijo, a los productos disponibles y el factor de mutación, si el valor generado aleatorio es menor o igual que el factor de mutación entonces escogerá dos posiciones y las intercambiará por nuevos valores.

Cuadro 0.9: Código de la función de mutación

```
def mutacion(hijo, disponibles, mut_fact):
  if random.random() <= mut_fact:
    gen1 = random.choice(range(len(hijo[0])))
    not_in_solution=list(set(disponibles)-set(hijo[0]))
    if len(not_in_solution)>0:
      temp=random.choice(not_in_solution)
      hijo[0][gen1]=temp
    return hijo
  else:
    return hijo[::]
```

3.3.1.6. FUNCIÓN DE REEMPLAZO

Debido a las funciones anteriores de cruce y selección la población inicial es mayor, por los nuevos descendientes, está función reemplazará y escogerá la mejor población. Primero ordenará toda la población acumulada de mayor a menor dependiendo de fitness o valor de aptitud, para luego escoger a los individuos por el porcentaje del valor elitista; y el resto de la población restante se escogerá de manera aleatoria.

Cuadro 0.10: Código de la función de reemplazo

```
def reemplazo(poblacion, IND, fact_elit):
  #Se ordena la población según el fitness(tamaño del recorrido) en
una lista de elementos [distancia, solucion]
  poblacion_ordenada = sorted(
                       [[fitness(individuo, rendimiento_kg_m2_prod,
precio_kg_t_suma), individuo]
                        for individuo in poblacion_final ],
```

```
                        key= lambda x:x[0], reverse= True)

    #Devolvemos elitismo % y el resto se eligen aleatoriamente
    return [x[1] for x in poblacion_ordenada][:int(N*fact_elit)]  + \
    random.sample([x[1] for x in poblacion_ordenada][int(N*fact_elit):]
 , int(N*(1-fact_elit)))
```

3.3.2. IDENTIFICAR LOS MÓDULOS Y PROCESOS QUE PERMITAN PARALELIZACIÓN

No todos los procesos son disponibles o permiten la paralelización, cabe mencionar que la GPU y la CPU tienen diferentes ventajas razón por la cual se usan para diferentes tareas, la GPU tiene un mayor rendimiento cuando los datos a procesar son muchos.

Para utilizar la GPU mediante la Librería Torch debe tener instalado CUDA, y declaramos que utilizaremos la GPU como lo muestra en el **Cuadro 0.11**.

Cuadro 0.11: Declaración de CUDA

```
device = torch.device('cuda:0')
```

Como Torch trabaja con tensores debemos convertir cada array a tensor, y como cada tensor se crea en la CPU lo mandamos a la GPU como lo muestra en el siguiente cuadro.

Cuadro 0.12: Transferencia de tensores de Cpu a Gpu

```
T_poblacion_inicial = torch.tensor(np.array(solucion_inicial))
T_rendimiento_kg_m2_prod = torch.tensor(rendimiento_kg_m2_prod)
T_precio_kg_t_suma = torch.tensor(precio_kg_t_suma)

T_poblacion_inicial = T_poblacion_inicial.to(device)
T_rendimiento_kg_m2_prod = T_rendimiento_kg_m2_prod.to(device)
T_precio_kg_t_suma = T_precio_kg_t_suma.to(device)
```

Una vez que los tensores estén en la GPU los podremos utilizar para optimizar y paralelizar diferentes funciones matemáticas, como ejemplo tenemos el siguiente código, donde la función “torch.sum()” hace todo el procedimiento con los módulos de la GPU.

Cuadro 0.13: Función Fitness con Gpu

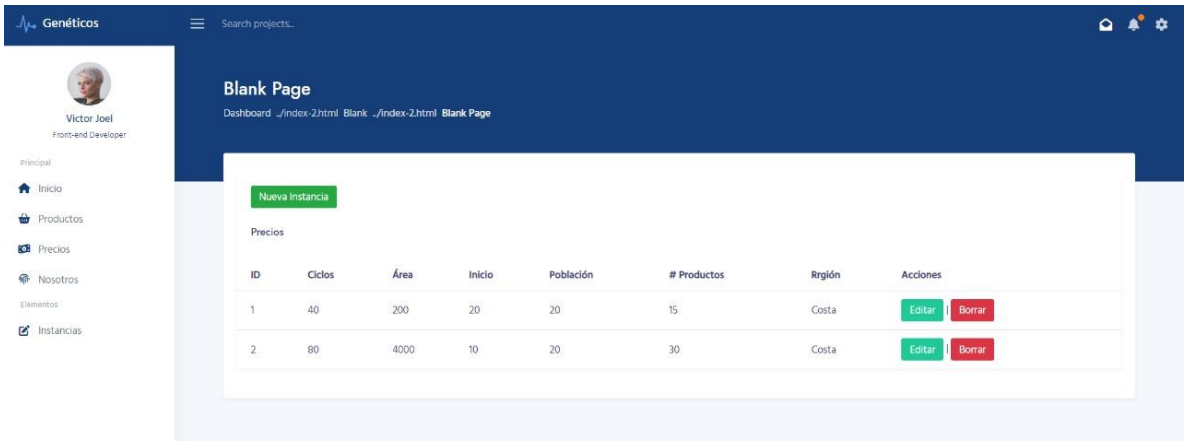
```
def fitness(individuo, rendimiento, precio):
  return torch.sum(rendimiento[individuo[0]] * precio[individuo[0]]
* individuo[1])
```

Según el tamaño de la operación y la CPU/GPU del sistema, la aceleración de esta operación puede ser >50x. Como las operaciones de sum o matmul son muy comunes, ya podemos ver el gran beneficio de acelerar una operación en una GPU.

3.3.3. DESARROLLAR MÓDULO WEB PARA LA EVALUACIÓN DE LAS INSTANCIAS.

Se desarrolló el módulo Web para la respectiva evaluación de las instancias, pero también se crearon vistas que sirven almacenar los parámetros requeridos en la generación de la solución las vistas, son Productos, Precios e Instancias las cuales se pueden visualizar en el Anexo2. Aquí en esta figura podemos ver los parámetros que se pide para la generación de la solución la cual será enviada por el correo registrado.

Figura 0.6: Módulo para crear Instancias



3.4. FASE 4: PRUEBAS

3.4.1. EVALUAR EL RENDIMIENTO DEL ALGORITMO PARALELIZADO.

Una vez que terminamos la fase de codificación con el objetivo de verificar el funcionamiento de cada una de las etapas se inició con la ejecución de las pruebas unitarias y carga de instancias.

En esta fase de pruebas se realizaron 4 experimentos donde se evalúa el rendimiento del algoritmo sin paralelizar frente al paralelizado, teniendo en cuenta los siguientes parámetros: **#Genes** que corresponde al número de genes de cada instancia, **Implementación** que corresponde al algoritmo paralelizado y al no paralelizado, **Función Fitness** que corresponde al tiempo de ejecución promedio de llamadas a la función de calidad en 10.000 loops, **Función Selección** correspondiente al tiempo de ejecución promedio de la fase de selección en 10.000 loops, **Función Cruce** correspondiente al tiempo de ejecución promedio de llamadas a la fase de cruce en 10.000 loops, como se muestra en la siguiente tabla:

Tabla 1: Resultados del rendimiento del algoritmo.

# Genes	Implementación	Función Fitness	Función Selección	Función Cruce
30	Paralelizado	33.8 ms	450 us	4.1 s

	Sin paralelizar	197 ms	1.2 ms	18 s
50	Paralelizado	24.8 ms	812 us	16 s
	Sin paralelizar	190 ms	1.34 ms	53.2 s
75	Paralelizado	52.6 ms	981 us	47 s
	Sin paralelizar	197 ms	1.62 ms	1.67 s
100	Paralelizado	56 ms	1.3 ms	1.2 m
	Sin paralelizar	113 ms	2.23 ms	2.37 m

Podemos comprobar que existe una mejora en rango de diferencia de entre 200% a 400%. Y mientras más se extienda el número mejor será, cabe recalcar que en las otras funciones que no se tomaron en cuenta no eran posible ya que no había mejora y el trabajo de la CPU era mejor a la de la GPU.

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Los autores del presente trabajo de integración curricular, concluyen que:

- Las páginas web consultadas para la recolección de datos permitieron identificar los parámetros, los requisitos funcionales y no funcionales que fueron importantes en cada una de las fases del proyecto.
- Se desarrolló un algoritmo basado en el modelo genético básico, mismo que permitió obtener buenos resultados en tiempo razonable.
- Entre las tecnologías y herramientas de paralelización se trabajó con la librería Torch, puesto que, a diferencia de otras librerías, esta permite empezar a trabajar de forma casi inmediata, sin necesidad de instalar aplicaciones externas y difíciles de configurar, como lo es el caso de CUDA.
- El módulo web se trabajó bajo la metodología XP, la misma que permitió diferenciar los procesos y completar cada una de las tareas durante el desarrollo del algoritmo y del módulo web.
- En las fases de paralización se obtuvieron resultados muy satisfactorios donde se evidencian mejoras en tiempo de ejecución frente al algoritmo no paralelizado, donde se concluye que el algoritmo paralelizado cumple con las condiciones necesarias para ser integrado en el módulo web, puesto que en ninguno de los experimentos se evidencian tiempos prolongados.

4.2. RECOMENDACIONES

- Se aconseja incluir más productos al dataset, puesto la eficiencia de un algoritmo paralelizado es directamente proporcional a la complejidad de su entrada, se evidencian mejoras en la implementación, pero serían aún más significativas si se trabaja con un dataset más grande.
- Se recomienda utilizar una buena metodología ágil que ayude a estructurar y planificar por medio de las fases cada uno de los procesos, ya que así se garantizará un buen proyecto.

- Teniendo en cuenta que existen diversas técnicas y herramientas para la paralelización en GPU, se recomienda siempre leer la documentación.
- La potencia de un algoritmo genético radica en la simpleza de sus fases; es decir; una implementación simple y limpia de cada fase permite alcanzar mejores resultados, puesto que se ejecutan más generaciones, lo que implica explorar un mayor espacio de solución.

BIBLIOGRAFÍA

Arias, M., & Londoño, J. (2009). *Algoritmos genéticos: una solución alternativa para optimizar el modelo de inventario*. http://200.12.180.26/handle/10784/125

Carreño, L., Isabel, M., Regalado, L., & Font, M. (2016). *Modelo Educativo*.

Corral Sastre, A. (2018). Desarrollo y Evaluación de Algoritmos Genóticos Multiobjetivo. Aplicación al problema del viajante. *Escuela Técnica Superior de Ingenieria Informética Universidad Politécnica de Valencia*, 2017–2018.

Erazo, I. (2022). Diseño y aplicación de los algoritmos genéticos para la sintonización de un controlador PID. *Polo Del Conocimiento*, *7*(1), 766–781. https://doi.org/10.23857/pc.v7i1.3509

ESPAM MFL. (2021). http://www.espam.edu.ec/web/oferta/grado/computacion.aspx

Espitia, J., & Mendoza, L. (2021). *Metodología basada en un algoritmo genético para programar la producción de una empresa del sector textil*. *número 4*.

Gutiérrez, J. J., Escalona, M. J., Mejías, M., & Torres, J. (n.d.). *Pruebas del sistema en programación extrema*.

Hernández Domínguez, J. L. (2010). Algoritmo Genético Multi-objetivo para el descubrimiento de secuencias reguladoras. In *INAOE*. INAOE.

Hidalgo, J., Portero, M. A., Nogueras, J., & Salvador, A. (2006). *Distribución equilibrada del esfuerzo de cómputo en Algoritmos Genéticos Paralelos.* 1–113.

INEC. (2020). *Instituto Nacional de Estadística y Censos. Encuesta de Superficie y Producción Agropecuaria Continua 2020*. https://www.ecuadorencifras.gob.ec/estadisticas-agropecuarias-2/

Jiménez Builes, J. A., Ramírez Bedoya, D. L., & Branch Bedoya, J. W. (2019). Metodología de desarrollo de software para plataformas educativas

robóticas usando ROS-XP. *Revista Politécnica*, *15*(30), 55–69. https://doi.org/10.33571/rpolitec.v15n30a6

Martínez, C. (2020). *Heuristicas basadas en algoritmos genéticos*.

Martínez Vargas, E., Rivera Martínez, M., Marcial Castillo, L. R., & Sandoval Solis, L. (2016). Implementación paralela de un algoritmo genético para el problema del agente viajero usando OpenMP. *Research in Computing Science*, *128*(1), 9–19. https://doi.org/10.13053/rcs-128-1-1

Molina, D. A., Pandolfi, D. R., & Villagra, N. A. (2014). Aplicación y evaluación de diferentes algoritmos genéticos canónicos en el diseño eficiente de redes de radio frecuencia en comunicaciones inalámbricas. *Informes Científicos Técnicos - UNPA*, *5*(3), 135–161. https://doi.org/10.22305/ict-unpa.v5i3.88

Pérez A., O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM. *Inventum*, *6*(10), 64–78. https://doi.org/10.26620/uniminuto.inventum.6.10.2011.64-78

Precios Mayoristas. (n.d.). Retrieved June 7, 2022, from http://sipa.agricultura.gob.ec/index.php/precios-mayoristas

Ramón, V., Díaz, R., & Fernandez, A. (2012). La especificación de requerimientos de software desde la perspectiva de un nuevo paradigma. *CienciaUAT*, *6*(3), 56–59.

Sánchez, E., Letelier, P., & Canós, J. (2003). *Mejorando la gestión de historias de usuario en Extreme Programming*. *May 2014*.

UNIDAD DE INFRAESTRUCTURA DE LA CARRERA DE COMPUTACION DE LA ESPAM MFL. (2020). *Documento de aprobación*.

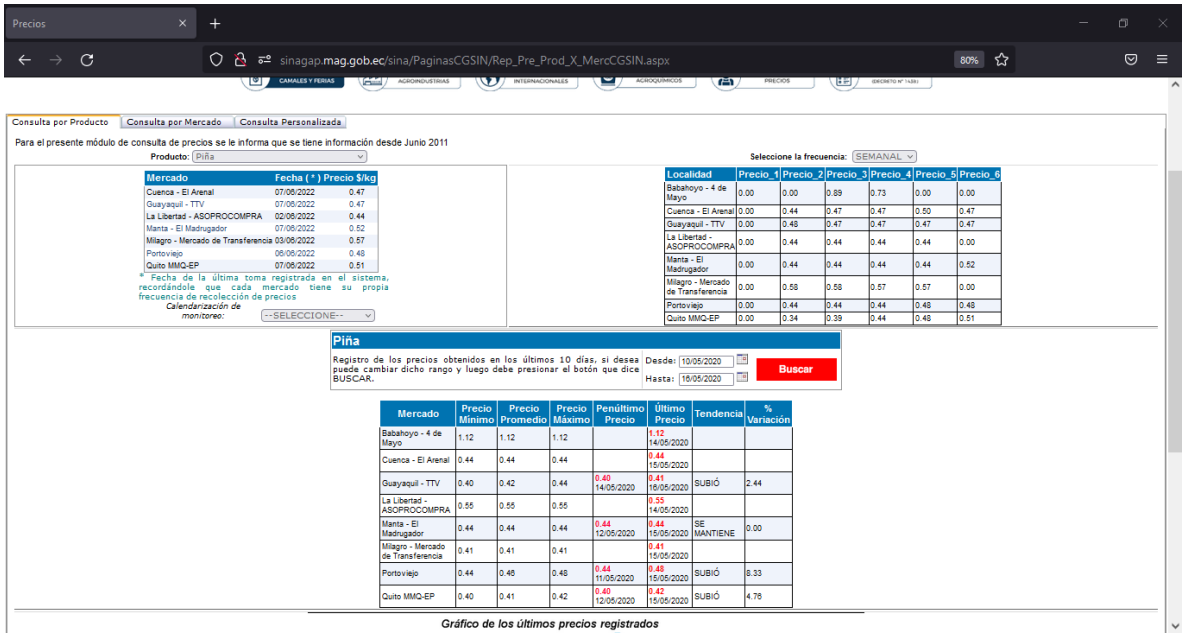
Valencia, P. E. (1997). Optimización mediante algoritmos genéticos. *Anales Del Instituto de Ingenieros de Chile*, *Agosto*, 83–92.

Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formacion Universitaria*, *14*(5), 85–94.

ANEXOS

ANEXO 1

RECOLECCIÓN DE DATOS POR MEDIO DE LA PÁGINA WEB DEL SIPA.



Consulta por Producto | Consulta por Mercado | Consulta Personalizada

Para el presente módulo de consulta de precios se le informa que se tiene información desde Junio 2011

Producto: Piña

Mercado	Fecha (*)	Precio $/kg
Cuenca - El Arenal	07/06/2022	0.47
Guayaquil - TTV	07/06/2022	0.47
La Libertad - ASOPROCOMPRA	02/06/2022	0.44
Manta - El Madrugador	07/06/2022	0.52
Milagro - Mercado de Transferencia	03/06/2022	0.57
Portoviejo	06/05/2022	0.48
Quito MMQ-EP	07/06/2022	0.51

* Fecha de la última toma registrada en el sistema, recordándole que cada mercado tiene su propia frecuencia de recolección de precios

Calendarización de monitoreo: --SELECCIONE--

Seleccione la frecuencia: SEMANAL

Localidad	Precio_1	Precio_2	Precio_3	Precio_4	Precio_5	Precio_6
Babahoyo - 4 de Mayo	0.00	0.00	0.89	0.73	0.00	0.00
Cuenca - El Arenal	0.00	0.44	0.47	0.47	0.50	0.47
Guayaquil - TTV	0.00	0.48	0.47	0.47	0.47	0.47
La Libertad - ASOPROCOMPRA	0.00	0.44	0.44	0.44	0.44	0.00
Manta - El Madrugador	0.00	0.44	0.44	0.44	0.44	0.52
Milagro - Mercado de Transferencia	0.00	0.58	0.58	0.57	0.57	0.00
Portoviejo	0.00	0.44	0.44	0.44	0.48	0.48
Quito MMQ-EP	0.00	0.34	0.39	0.44	0.48	0.51

Piña

Registro de los precios obtenidos en los últimos 10 días, si desea puede cambiar dicho rango y luego debe presionar el botón que dice BUSCAR.

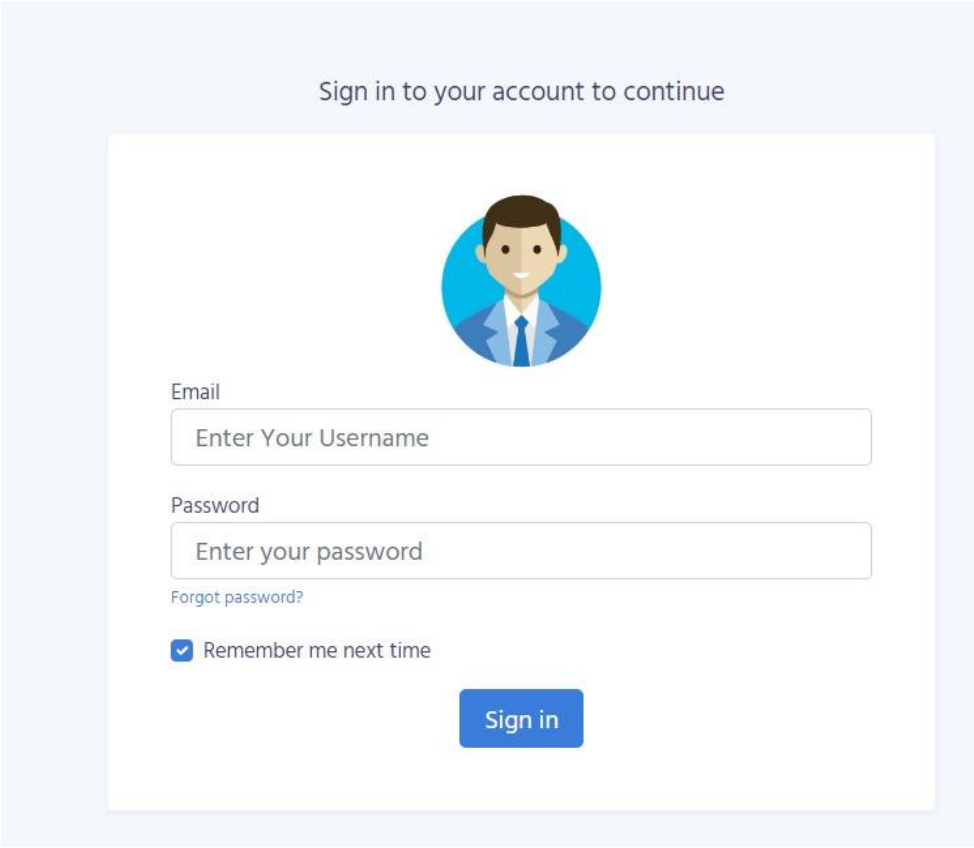
Desde: 10/05/2020 Hasta: 16/05/2020 Buscar

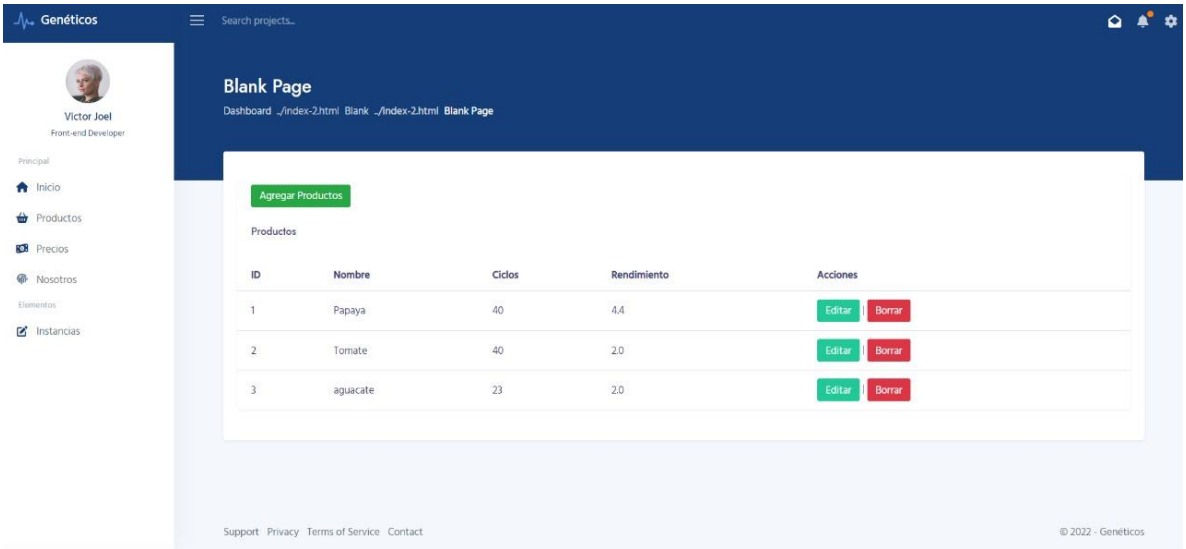
Mercado	Precio Mínimo	Precio Promedio	Precio Máximo	Penúltimo Precio	Último Precio	Tendencia	% Variación
Babahoyo - 4 de Mayo	1.12	1.12	1.12		1.12 14/05/2020		
Cuenca - El Arenal	0.44	0.44	0.44		0.44 15/05/2020		
Guayaquil - TTV	0.40	0.42	0.44	0.40 14/05/2020	0.41 15/05/2020	SUBIÓ	2.44
La Libertad - ASOPROCOMPRA	0.55	0.55	0.55		0.55 14/05/2020		
Manta - El Madrugador	0.44	0.44	0.44	0.44 12/05/2020	0.44 15/05/2020	SE MANTIENE	0.00
Milagro - Mercado de Transferencia	0.41	0.41	0.41		0.41 15/05/2020		
Portoviejo	0.44	0.46	0.48	0.44 11/05/2020	0.48 15/05/2020	SUBIÓ	8.33
Quito MMQ-EP	0.40	0.41	0.42	0.40 12/05/2020	0.42 15/05/2020	SUBIÓ	4.76

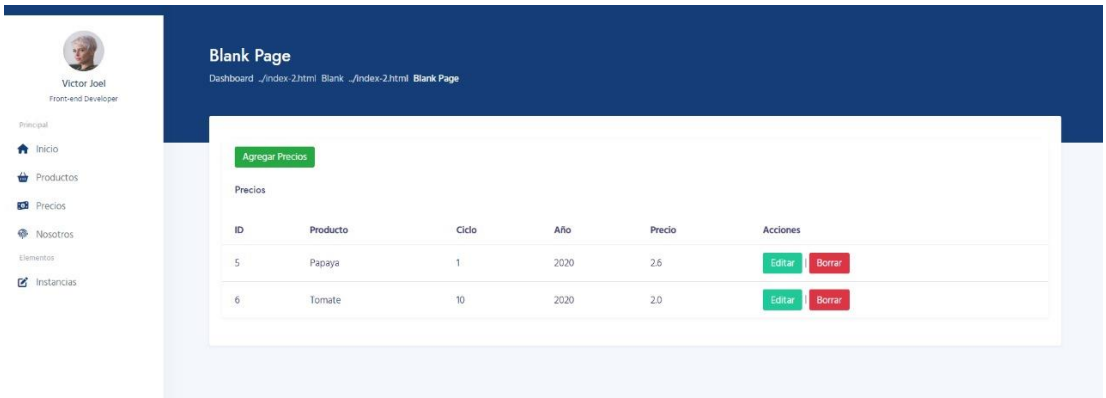
Gráfico de los últimos precios registrados

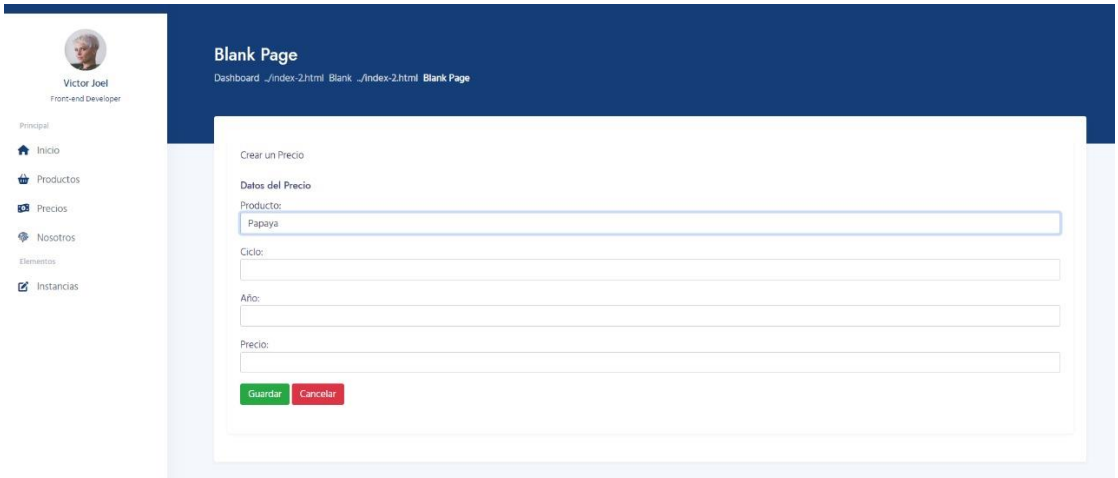
ANEXO 2

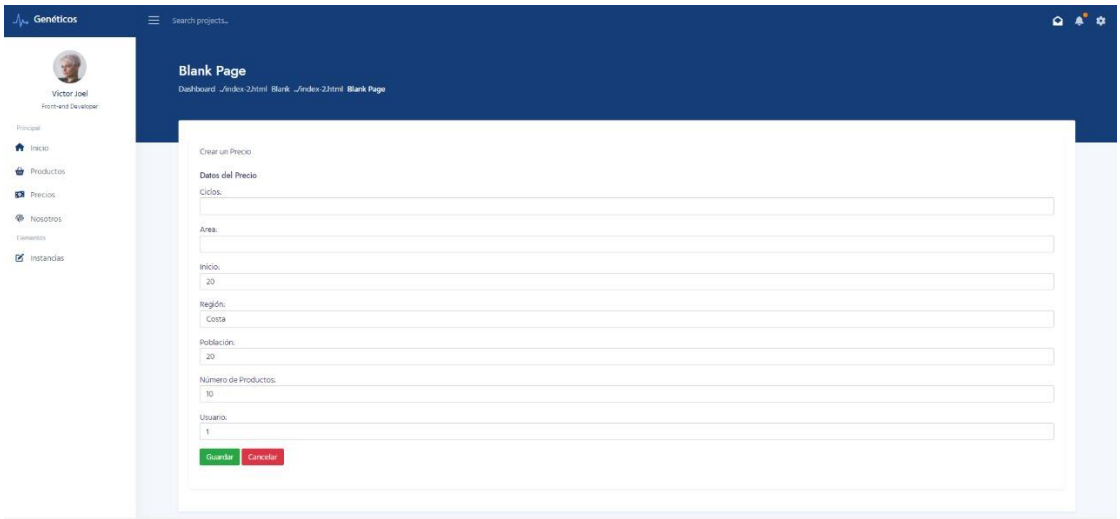
CAPTURA DEL MÓDULO WEB PARA LA VISUALIZACIÓN DE DATOS


Sign in to your account to continue
Email
Enter Your Username
Password
Enter your password
Forgot password?
Remember me next time
Sign in


Genéticos
Search projects...
Victor Joel
Front-end Developer
Principal
Inicio
Productos
Precios
Nosotros
Elementos
Instancias
Blank Page
Dashboard ../index-2.html Blank ../index-2.html Blank Page
Agregar Productos
Productos
ID Nombre Ciclos Rendimiento Acciones
1 Papaya 40 4.4 Editar | Borrar
2 Tomate 40 2.0 Editar | Borrar
3 aguacate 23 2.0 Editar | Borrar
Support Privacy Terms of Service Contact
© 2022 - Genéticos


Victor Joel
Front-end Developer
Principal
Inicio
Productos
Precios
Nosotros
Elementos
Instancias
Blank Page
Dashboard ../index-2.html Blank ../index-2.html Blank Page
Agregar Precios
Precios
ID Producto Ciclo Año Precio Acciones
5 Papaya 1 2020 2.6 Editar | Borrar
6 Tomate 10 2020 2.0 Editar | Borrar


Victor Joel
Front-end Developer
Principal
Inicio
Productos
Precios
Nosotros
Elementos
Instancias
Blank Page
Dashboard ../index-2.html Blank ../index-2.html Blank Page
Crear un Precio
Datos del Precio
Producto:
Papaya
Ciclo:
Año:
Precio:
Guardar
Cancelar


Genéticos
Search projects...
Victor Joel
Front-end Developer
Principal
Inicio
Productos
Precios
Nosotros
Elementos
Instancias
Blank Page
Dashboard ../index-2.html Blank ../index-2.html Blank Page
Crear un Precio
Datos del Precio
Ciclos:
Area:
Inicio:
20
Región:
Costa
Población:
20
Número de Productos:
10
Usuario:
1
Guardar
Cancelar

ANEXO 3

ESPECIFICACIÓN DE REQUISITOS SEGÚN EL ESTANDAR IEEE 830

UNIDAD DE DESARROLLO COMPUTACIONAL

FORMATO PARA LA ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

OBJETIVO

Registrar los procesos y características definidas que debe cumplir el software, de tal forma estos requisitos puedan ser verificados y validados objetivamente.

PROCESOS OPERATIVOS: Planificación del proyecto de software

UNIDAD DE DESARROLLO COMPUTACIONAL

Módulo web para la evaluación de las instancias

Anderson Joel Zambrano Moreira

Jesús Alberto García Mera

Versión 1.0

2022/05/09

Contenido

1. INTRODUCCIÓN 42
1.1. IDENTIFICACIÓN DEL SISTEMA 42
1.2. OBJETIVO 42
1.3. ALCANCE 42
1.4. PERSONAL INVOLUCRADO 43
1.5. NOTACIONES Y DEFINICIONES 43
1.5.1. NOTACIONES 43
1.5.2. DEFINICIONES 44
1.6. REFERENCIAS 44
2. DESCRIPCIÓN GENERAL 44
2.1. PERSPECTIVAS DEL PRODUCTO 45
2.2. FUNCIONES DEL PRODUCTO 45
2.3. CARACTERÍSTICAS DE USUARIO 45
2.4. RESTRICCIONES 46
2.5. SUPOSICIONES Y DEPENDENCIAS 46
2.6. REQUISITOS FUTUROS 46
DIAGRAMAS 47
2.6.1. DIAGRAMA DE CASO DE USO 47
2.6.2. DIAGRAMA DE CLASES DEL SISTEMA 48
2.6.3. DIAGRAMA DE BASE DE DATOS DEL SISTEMA 49
3. REQUERIMIENTOS ESPECÍFICOS 49

3.1. REQUERIMIENTOS DE INTERFACES EXTERNAS 49
3.1.1. INTERFACES DE USUARIO 49
3.1.2. INTERFACES CON EL HARDWARE 50
3.1.3. INTERFACES SOFTWARE 50
3.2. REQUERIMIENTOS FUNCIONALES 51
3.2.1. REGISTROS DE USUARIOS. 51
3.2.2. VERIFICAR AUTENTICACIÓN DE USUARIOS. 51
3.2.3. INGRESO DE PRODUCTOS 52
3.2.4. INGRESO DE PRECIOS 53
3.2.5. INGRESO DE PARAMETROS 54
3.2.6. VISUALIZACIÓN DE RESULTADOS**¡Error! Marcador no definido.**
3.3. REQUERIMIENTOS NO FUNCIONALES 54
3.3.1. REQUERIMIENTOS DE RENDIMIENTO. 54
3.3.2. REQUISITO DE DISEÑO 54
3.3.3. ATRIBUTOS DEL SISTEMA 55

1. INTRODUCCIÓN

El presente documento describe la Especificación de Requerimientos de Software (ERS), del sistema web para generar y visualizar resultados del algoritmo genético para optimizar la programación agrícola, la que se constituye como un manual durante el desarrollo y posterior implementación del aplicativo. Además, en este se describe cada uno de los requerimientos obtenidos a través del análisis e investigación realizados, las características del sistema, lo que puede y no puede realizar, además se define los requerimientos tecnológicos necesarios para el buen funcionamiento del sistema.

Este ERS podrá ser utilizado como descripción, para obtener información sobre la administración, funcionamiento y mantenimiento, también contendrá información relevante como guía para cualquier otro desarrollador que necesite realizar mejoras o modificaciones del sistema.

1.1. IDENTIFICACIÓN DEL SISTEMA

El módulo servirá para generar y mostrar la visualización del algoritmo genético. El sistema web incluirá varios módulos con los cuales se podrán administrar las apps tanto como su contenido, los archivos, los roles de usuario, los usuarios, gestionar productos y precios.

1.2. OBJETIVO

Desarrollar módulo web para la evaluación de las instancias.

1.3. ALCANCE

El sistema web permitirá administrar y almacenar los productos, rendimientos, precios que servirán para construir la solución.

- **Objetivos específicos del Sistema**
    - Recopilar la información necesaria para el sistema.
    - Diseñar la arquitectura de software.

- Desarrollar el sistema web basado en la metodología ágil Xp.
- Ejecutar las pruebas de software.
- Implementar el sistema web.

1.4. PERSONAL INVOLUCRADO

Nombre	Anderson Joel Zambrano Moreira
Rol	Programador
Categoría Profesional	Estudiante Universitario
Información de contacto	anderson.zambrano@espam.edu.ec

Nombre	Jesús Alberto García Mera
Rol	Programador
Categoría Profesional	Estudiante Universitario
Información de contacto	jesus.garcia@espam.edu.ec

Nombre	Víctor Joel Pinargote Bravo
Rol	Tutor
Categoría Profesional	Ingeniero
Información de contacto	vpinargote@espam.edu.ec

1.5. NOTACIONES Y DEFINICIONES

1.5.1. NOTACIONES

ERS: Especificación de requerimientos de software.

1.5.2. DEFINICIONES

- **Usuarios**

Aquella persona que hace uso de un servicio.

- **Framework**

Es el esquema o estructura que se establece y que se aprovecha para desarrollar y organizar un software determinado.

1.6. REFERENCIAS

IEEE (Institute of Electrical and Electronics Engineers), 2009. IEEE Recommended Practice for Software Requirements Specifications Standard IEEE-830-1998. New York, USA.

2. DESCRIPCIÓN GENERAL

El siguiente ERS muestra información sobre los requisitos del sistema a desarrollar, de una manera general, sin describir de manera profunda el sistema, lo que permitirá obtener un gran entendimiento por parte del cliente y desarrollador.

Sin embargo, se detallan los Requerimientos Específicos del sistema web panel de control de aplicaciones móviles, para el diseño del sistema con cada uno de los requerimientos que presenta el cliente, en este caso la Unidad de Desarrollo Computacional de la ESPAM MFL.

El ERS está dividido en tres temas generales:

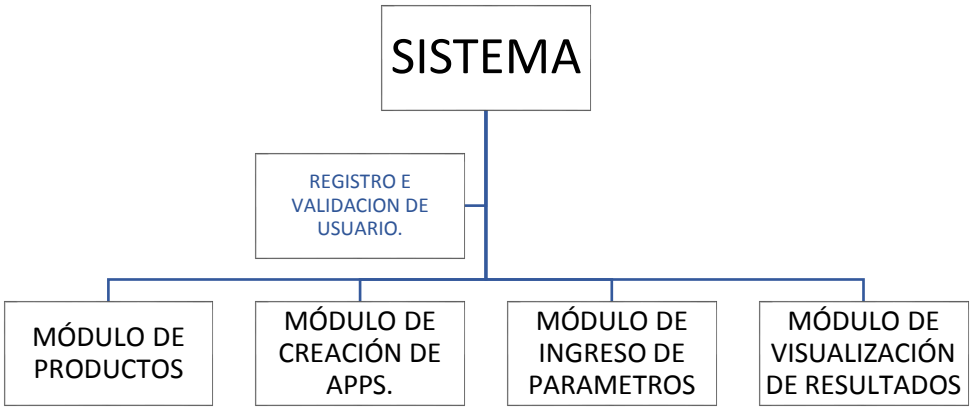
- Introducción

- Descripción General
- Requerimientos Específicos.

2.1. PERSPECTIVAS DEL PRODUCTO

Está orientado a la reserva y administración de datos, junto con la visualización de resultados.

2.2. FUNCIONES DEL PRODUCTO



2.3. CARACTERÍSTICAS DE USUARIO

El acceso al sistema está restringido por diferentes módulos mediante una verificación de su perfil:

TIPO DE USUARIO	**ADMINISTRADOR**
DESCRIPCIÓN	Tiene acceso a los diferentes módulos del sistema, se encarga de la configuración del sitio.

TIPO DE USUARIO	**USUARIO**
DESCRIPCIÓN	Tiene acceso a los diferentes módulos del sistema.

2.4. RESTRICCIONES

Entre las limitaciones que se tienen en la aplicación son las siguientes:

- El sistema será desarrollado con el framework de desarrollo web de código abierto Django.
- El motor de la base de datos relacional a utilizar será **MySQL**.
- Como fuente de diseño para el módulo web se utilizará una plantilla de **Booststrap 4.**
- La metodología para el desarrollo se basará en las mejores características de las metodologías **ágiles** como **XP**.

2.5. SUPOSICIONES Y DEPENDENCIAS

El sistema debe adaptarse a cada uno de los navegadores web disponibles actualmente, para que así ningún cambio o actualización en ellos puedan afectar en el diseño o uso de elementos vinculados al mismo.

2.6. REQUISITOS FUTUROS

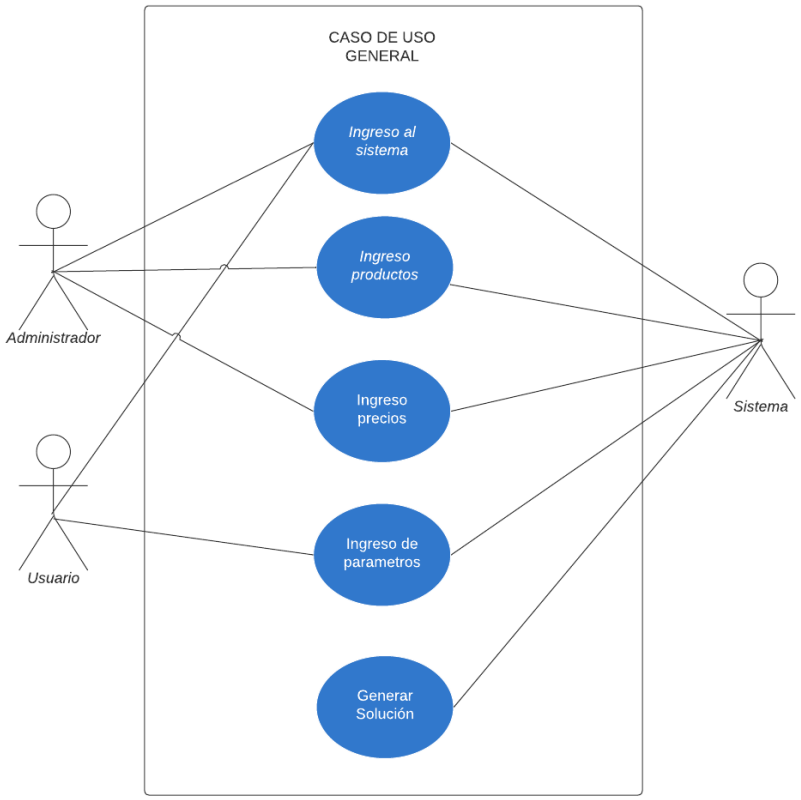
Los requisitos planteados pueden ser posibles mejoras, que luego de estudio y análisis pueden generar cambios en el sistema:

- Mejoras en la plantilla del sistema general.
- Implementación de nuevos mecanismos de seguridad en el ingreso del sistema.
- Optimización de módulos que posee el sistema.

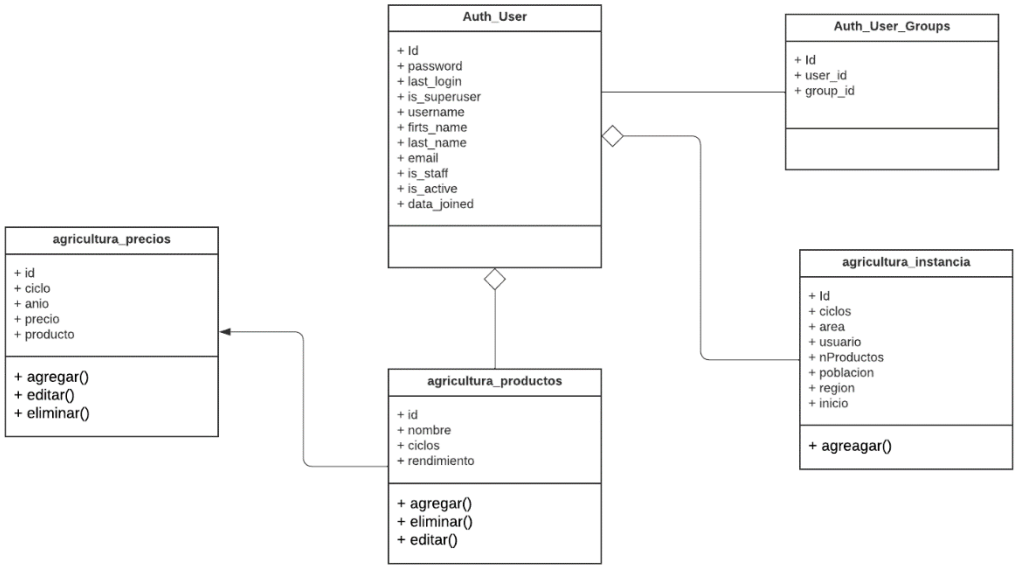
- Inclusión de nuevos módulos acorde a las necesidades futuras.
- Mejoras en la interacción del usuario con el sistema.

DIAGRAMAS

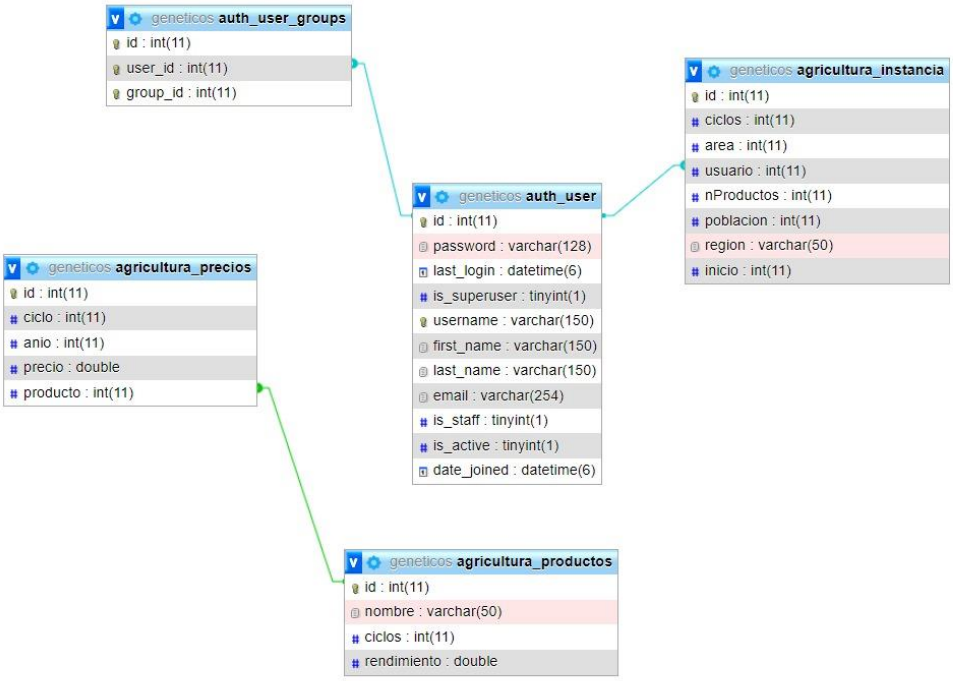
2.6.1. DIAGRAMA DE CASO DE USO



2.6.2. DIAGRAMA DE CLASES DEL SISTEMA



2.6.3. DIAGRAMA DE BASE DE DATOS DEL SISTEMA



3. REQUERIMIENTOS ESPECÍFICOS

Con estos requerimientos se permitirá la comprobación que los procesos desarrollados con la metodología ágil XP, y a su vez se tomará como referencia para la comprensión del diseño que este contenga.

3.1. REQUERIMIENTOS DE INTERFACES EXTERNAS

3.1.1. INTERFACES DE USUARIO

En el aplicativo se implementarán diferentes elementos para manipular la información de las apps.

- Ingreso al sistema mediante en base la información que se le solicita al registrar usuarios.

- Botones para gestionar diferentes procesos, como editar, guardar y eliminar funciones propias del módulo.
- Generación de reportes
- Menús desplegables
- Mensajes informativos
- Mensajes de error

3.1.2. INTERFACES CON EL HARDWARE

Será necesario disponer de computadores y servidores en perfecto estado con las siguientes características mínimas.

3.1.2.1. Tecnología mínima que debe disponer el servidor.

Las características mínimas que debe de tener el servidor para que pueda soportar las herramientas y permita funcionar el módulo son los siguientes:

- Procesador Intel mínimo Core i3, AMD.
- Memoria RAM de 64 Gb.
- Disco Duro de 500 Gb.
- Tarjeta de red integrada 10/100/1000 Ethernet.
- Monitor, mouse, teclado.
- Conexión a internet.

3.1.3. INTERFACES SOFTWARE

Tendrá la capacidad de que el usuario pueda acceder desde cualquier computador con sistemas operativos como Windows, macOS o GNU/Linux y navegadores web como Microsoft Edge, Mozilla Firefox, Google Chrome.

3.2. REQUERIMIENTOS FUNCIONALES

3.2.1. REGISTROS DE USUARIOS.

Módulo de ingreso al sistema

Código de requisito	**RF001**
Nombre de requisito	Registro de usuario.
Tipo	Requisitos de producto ● Requisitos de proyecto ○
Fuente del requisito	Roles del sistema
Prioridad del requisito	Alta/Esencial ● Media/Deseado ○ Baja/Opcional ○
Caso de uso asociado	CU001
Historia de usuario	Paso 01: Registrar usuarios.
DESCRIPCIÓN	El sistema debe permitir el registro para obtener las credenciales del ingreso al sistema.
PROCESO	El sistema pedirá los correspondientes datos de usuario para poder hacer el registro los cuales son: nombres, apellidos, correo y contraseña. Luego el sistema enviará un correo para verificar y registrarlos en la base de datos.
ENTRADAS	Nombre de usuario, nombre, apellido, correo y contraseña.
SALIDAS	• Mensaje de error en el caso de no haber llenado algún campo. • Mensaje de error en caso de que las contraseñas no coincidan. • Ingreso exitoso.
RESTRICCIONES	Ninguna

3.2.2. VERIFICAR AUTENTICACIÓN DE USUARIOS.

Módulo de configuración del sistema

Código de requisito	**RF002**
Nombre de requisito	Validación de cuenta.
Tipo	Requisitos de producto Requisitos de proyecto ○
Fuente del requisito	Roles del sistema

Prioridad del requisito	Alta/Esencial ● Media/Deseado ○ Baja/Opcional ○
Caso de uso asociado	CU002
Historia de usuario	Paso 01: Ingreso al sistema
DESCRIPCIÓN	El sistema debe permitir el ingreso del nombre de usuario y contraseña.
PROCESO	El sistema pedirá los correspondientes datos de usuario para ingresar al sistema.
ENTRADAS	Nombre de usuario y contraseña
SALIDAS	• Mensaje de error en el caso de no haber llenado algún campo. • Mensaje de error en casos de ingresar incorrectamente los datos es decir que el formato de los datos no sea el correcto. • Validación exitosa.
RESTRICCIONES	Ninguna

3.2.3. INGRESO DE PRODUCTOS

Módulo de configuración del sistema

Código de requisito	**RF003**
Nombre de requisito	Información de perfil.
Tipo	Requisitos de producto ● Requisitos de proyecto ○
Fuente del requisito	Roles del sistema
Prioridad del requisito	Alta/Esencial ○ Media/Deseado ○ Baja/Opcional ●
Caso de uso asociado	CU003
Historia de usuario	Paso 01: Ingreso a al sistema como administrador Paso 02: Ingreso a la opción productos
DESCRIPCIÓN	El administrador tendrá la opción de agregar, editar y eliminar los productos junto con sus características

PROCESO	El sistema mostrará un formulario donde se pedirán los datos y características del producto y guardarlos en la base de datos.
ENTRADAS	Nombre del producto, rendimiento del producto y etapas de maduración.
SALIDAS	• Productos ingresados
RESTRICCIONES	No aplica.

3.2.4. INGRESO DE PRECIOS

Módulo de configuración del sistema

Código de requisito	**RF004**
Nombre de requisito	Ingreso de precios
Tipo	Requisitos de producto ● Requisitos de proyecto ○
Fuente del requisito	Roles del PCAM
Prioridad del requisito	Alta/Esencial ● Media/Deseado ○ Baja/Opcional ○
Caso de uso asociado	CU004
Historia de usuario	Paso 01: Ingreso al sistema como administrador Paso 02: Ingreso a la opción precios.
DESCRIPCIÓN	El administrador tendrá la opción de agregar los precios a cada producto.
PROCESO	El administrador deberá de ingresar los precios de cada uno de los productos y guardarlos en la base de datos.
ENTRADAS	Precios
SALIDAS	• Ingreso de precios
RESTRICCIONES	Ninguna

3.2.5. INGRESO DE INSTANCIAS

Módulo de configuración del sistema

Código de requisito	**RF004**
Nombre de requisito	Ingreso de instancias
Tipo	Requisitos de producto ● Requisitos de proyecto ○
Fuente del requisito	Roles del PCAM
Prioridad del requisito	Alta/Esencial ● Media/Deseado ○ Baja/Opcional ○
Caso de uso asociado	CU004
Historia de usuario	Paso 01: Ingreso al sistema. Paso 02: Ingreso a la opción instancias.
DESCRIPCIÓN	El usuario tendrá la opción de agregar nuevas instancias
PROCESO	El usuario deberá agregar cada una de las instancias.
ENTRADAS	Ciclos, área, número de productos, fecha inicio.
SALIDAS	• Visualización de resultados
RESTRICCIONES	Ninguna

3.3. REQUERIMIENTOS NO FUNCIONALES

3.3.1. REQUERIMIENTOS DE RENDIMIENTO.

Garantizar que el diseño de las consultas u otro proceso que no afecte el desempeño de la base de datos.

3.3.2. REQUISITO DE DISEÑO

La aplicación web debe tener un diseño Responsivo, a fin de garantizar la adecuada visualización.

3.3.3. ATRIBUTOS DEL SISTEMA

3.3.3.1. Requerimientos de desarrollo

Se usará para el desarrollo web el framework de programación Django y la base de datos relacional MySQL en conjunto con una plantilla de Bootstrap 4 y aplicando la metodología XP.

3.3.3.2. Seguridad

Para poder ingresar al sistema web es necesario contar con un usuario y una contraseña que se encuentren almacenados en la base de datos, de esta manera se controla que solo los usuarios autorizados podrán manipular su información y realizar sus respectivas tareas garantizando la seguridad por medio de métodos de encriptación de datos y confiabilidad de la información.

3.3.3.3. Mantenimiento

Se le dará futuras mejoras en el aspecto de la plantilla y relacionado a la seguridad, optimización del sistema a partir de estudios y pruebas realizas.