Instituto Politécnico Nacional
Escuela Superior de Cómputo

Avance de Proyecto

Compihinador

Ingeniería de software
Profesora: Martha Rosa Cordero Lopez
Grupo: 3CM3
Integrantes:
Adrian González Pardo
Melani Betsabee Valdez Esquivel
Ángel Edmundo Hernández Rivera
Mayer Abraham Pérez González
Semestre: 20/02

Índice

1. Introducción

El nombre para el proyecto sera: Compihinador.

1.1. Objetivo

El aprendizaje de lenguajes de programación actualmente tiene muchas vertientes en las cuales se puede partir desde lenguajes con una curva de aprendizaje muy sencilla y fluida, hasta lenguajes en los que su curva de aprendizaje implican el uso de algunas notaciones matemáticas o algunas notaciones que impliquen una alta abstracción del lenguaje.

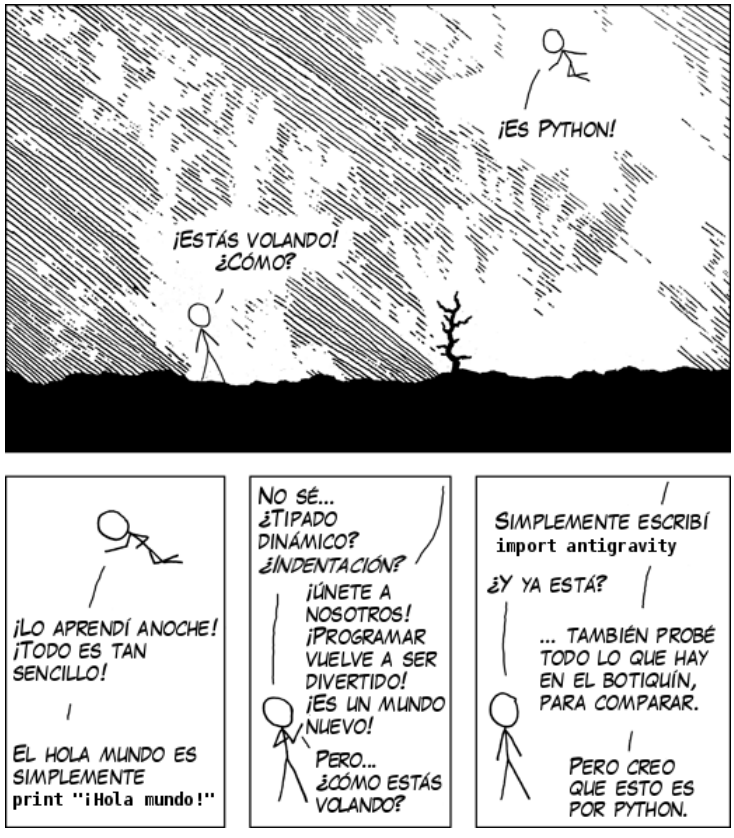


Figura 1: Python siendo mayormente usado como lenguaje de programación, siendo un lenguaje mayormente interpretado

En algunos casos el uso de lenguajes que son interpretados, generan un consumo significativo de energía y esto en volumenes enormes de información genera problemas en la industria e incluso en las pruebas de performance.

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Figura 2: Tabla de resultados de uso de energía, tiempo y peso en disco de cada lenguaje de programación a los que se realizaron pruebas

Documento de: How Do Energy, Time, and Memory Relate?

Con base a los datos mostrados en la tabla anterior, se tiene una idea de la diferencia que existe en el consumo de energía, tiempo y memoria al usar un determinado lenguaje de programación. Lo que se contempla para este proyecto es tomar como referentes estos datos para realizar una optimización en cada uno de los parámetros establecidos en la tabla, por lo tanto, se planea que al ejecutar el compilador se puedan obtener valores de costo de energía, tiempo y memoria menores en el uso de un determinado lenguaje de programación con referencia a los que se indican anteriormente.

1.2. Descripción del Proyecto

El compilador realizara uso de módulos de herramientas que nos puede proporcionar el software especializado para la creación y descripción de compiladores, en los cuales se buscara realizar una aproximación de que el compilador y el uso de memoria no cree un consumo excedente de energía, memoria o tiempo al generar el código objeto.

1.3. Funciones Principales

El compilador es realizado con el fin de poder tener una mejor cercanía con usuarios principiantes los cuales tiene noción de algunos lenguajes de programación como son *Python*, *Ruby*,

Perl, *etc.*
Si bien estos lenguajes son considerados lenguajes de alto nivel, se pretende realizar un compilador que realice algunas operaciones en las que puedan tener un acercamiento a estos tipos de lenguajes y que más tarde puedan mudar de un lenguaje a otro.
Algunas de las funciones que se consideran son:

- Apoyar el aprendizaje del usuario de algún determinado lenguaje de programación.
- Capacidad de compilar codigos escritos en diferentes lenguajes de programación.
- Optimización de recursos. (Energía, tiempo y memoria)

1.4. Justificación

Entender mejor el funcionamiento de los compiladores y su comportamiento a un bajo nivel para poder optimizar y gestionar mejor los recursos a la hora de desarrollar un software de aplicación.

1.5. Herramientas

El proyecto a desarrollar contiene algunos puntos a resaltar de los cuales son Hardware y Software en el que se realizara el desarrollo del producto y de su documentación.

1.5.1. Hardware

- El hardware en el que sera desarrollado el proyecto trabajara bajo una arquitectura x86_64, debido a que es el actual estándar de desarrollo.
- El equipo deseablemente contara con al menos 4GB de RAM.
- Procesador dualcore o más cores superior a 1GHz.

1.5.2. Software

- El producto pretende crear un pseudolenguaje que se compila y pretende tener una mejor cercanía con el programador principiante.
- El Sistema Operativo en el que se desarrollara y sera la plataforma en la que se ejecutara el proyecto sera en sistema Linux, debido a que las herramientas de trabajo como el analizador léxico, sintactico y semantico pueden encontrarse en paqueterias libres que proporciona Linux.
- Se pretende hacer uso de herramientas que proporciona la terminal como el uso de alias y simbolic links para que las llamadas del compilador sean más sencillas en su uso.
- Tambien se en la instalación de la paqueteria del compilador se añadiran los manuales pertinentes para que el usuario pueda trabajar correctamente con el producto.

1.6. Innovación

La innovación del producto de software es el hacer uso de estandares que nos pueden proporcionar el estandar de POSIX, y algunas otras buenas practicas de la programación de modulos de software.
Por otro lado tambien es necesario el hacer uso de algoritmos que permitan optimizar el uso de energía que genera la creación y ejecución de código en su entorno de programación.
Por lo tanto, si bien es un tiempo corto para la realización de todas las actualizaciones y desarrollo, es bueno planificar el hecho de que pueda existir una aproximación a prototipos del compilador.

1.7. Complejidad

De acuerdo a la idea de proyecto a realizar es una tarea larga ya que requiere no solo de uso de herramientas que puede ofrecer el proyecto de GNU/Linux sino que también implica el aprendizaje de las mismas y por otro lado también implica el uso de herramientas que permitan realizar pruebas de estrés y testing como puede realizarlo algunas herramientas de scripting como lo son *Bash*, *sh*, *Ruby*, *Python*.

1.8. Paradigma de Desarrollo

El paradigma a utilizar para el proyecto sera el uso de prototipos, así como en la escritura de código sera en el paradigma Estructurado ya que este nos permitirá trabajar con las herramientas que nos ofrece el Sistema Operativo, y un poco de paradigma Orientado a Objetos para hacer uso de algunas herramientas que nos ofrece el mismo, como la modularidad y el uso de algunos empaquetamientos. Nos facilitara la detección de errores o *bugs* en el proyecto durante su desarrollo.

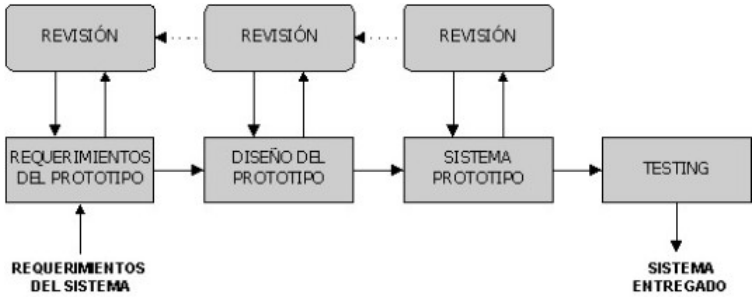


Figura 3: Esquema de "Modelo de Prototipos"

1.9. Técnicas de Desarrollo

Para la recolección de requisitos se llevara acabo una encuesta en la cual nos permitirá recabar información, en la cual podremos interpretar la dirección final y los detalles a pulir del proyecto, en los cuales podremos saber si los encuestados tienen una facilidad o dificultad a la hora de trabajar con un lenguaje de programación compilado.
Los hitos planificados del proyecto es trabajarlo con base en el modelo de prototipos para después poder refinar los hitos consiguientes, debido a que es un modelo más sencillos para la realización del producto.
También se planea el uso de paradigmas como el orientado a objetos y el estructurado para el desarrollo de los módulos/prototipos.

1.9.1. Anexo de Prototipo Encuesta 1:

1. ¿Consideras que es muy difícil el escribir un programa en algún lenguaje de programación?
2. ¿Consideras importante el costo de energía cuando ejecutas un programa?
3. Bajo tu experiencia. ¿Te interesaría que el compilador que estés manejando sea óptimo en tiempo y en uso de recursos de memoria?
4. En lo personal. ¿Prefieres que el compilador que estés utilizando sea el mas óptimo en recursos o prefieres que el tiempo de compilación sea menor?
5. ¿Te agradaría utilizar un compilador bajo la ideología del software libre así como el obtener acceso a su código fuente para mejorar, como un proyecto bajo el estilo de GNU?
6. ¿Te agradaría la integración de un compilador ligero que permita tener actualizaciones gratuitas como las que se ven disponibles en los repositorios de distribuciones Linux?

2. Métricas y estimación

2.1. Tabla KLDC y Function Point

Parámetro de medición	Simple	Complejidad Media	Compleja	Total
Número de entradas 3	3	3	3	27
Número de salidas 2	2	3	3	16
Número de archivos 12	4	4	5	156
Número de interfaces externas 3	3	4	4	33
-	-	-	Total	232

De acuerdo a las 14 preguntas que se realizan en este método obtenemos la suma de $\Sigma fi = 63$ por lo que obtendremos el valor de Point Function PF igual a $PF = 232 * [0{,}65 + 0{,}01 * \Sigma fi = 296$.
Ahora con los datos de otra tabla.

Costo	Personas	Meses	Pag. Doc.	Errores	Esfuerzo
2899	4	3	15000	9	-

De los cuales podemos obtener los siguientes datos

- Costo = 9.7622 por función
- Calidad = 0.00026
- Productividad = 74.24 funciones por persona
- Documentación = 50.56 paginas por función

Para el caso de KLDC, tomando un KLDC aproximado de 800K :

- Costo = 0.014445
- Calidad = 0.00002
- Productividad = 50000
- Esfuerzo = 12

Si bien podemos notar que este tipo de análisis nos arroja datos que son importantes en la toma de decisiones y adecuaciones del proyecto el planteamiento y la misión de la realización del producto de software se planea no solo para su trabajo a nivel académico, sino que de igual forma podamos lanzar el producto para que la amplia comunidad de software libre pueda ser participe del trabajo.

2.2. Estimación con COCOMO

Si bien existen algunos métodos para hacer estimación de un software como por ejemplo el modelo de Putnam Myers, el más utilizado es el propuesto por Barry Boehm, llamado COCOMO o Modelo Constructivo de Costos.
Este modelo se divide en tres submodelos: básico, intermedio y avanzado.

- El COCOMO básico calcula el esfuerzo y costo del software en función del tamaño del programa expresado en lineas de código.
- El COCOMO intermedio calcula el esfuerzo del desarrollo de software en función del tamaño del programa y de un conjunto de conductores de coste que incluye la evaluación subjetiva del producto hardware, personal y de los atributos del proyecto.
- El COCOMO avanzado incorpora todas las características de la versión intermedia pero además lleva acabo una evaluación de los impactos de coste en cada etapa (análisis, diseño, etc).

De igual manera el COCOMO básico e intermedio tienen en consideración tres modos de desarrollo: orgánico, semi-acoplado y empotrado.

- En el Modo Orgánico hablamos de proyectos pequeños en el que trabajas con equipos reducidos con experiencia en la aplicación sobre un conjunto de requisitos poco rígidos.
- El Modo Semi-acoplado es para proyectos de software intermedios en tamaño y complejidad, variados niveles de experiencia que deben satisfacer requisitos poco o medio rígidos.
- El Modo Empotrado se utiliza para proyectos de software avanzados desarrollados en un conjuntos de hardware, software y restricciones operativos muy fuertes.

A continuación se presentan las tablas de valores y ecuaciones para hacer la estimación con el modelo de COCOMO.
Tabla de valores para COCOMO básico:

Proyecto de SW	ab	bb	cb	db
Orgánico	2.4	1.05	2.5	0.36
Semi-acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.35

Ecuaciones para calcular estimación para COCOMO básico:

$$\text{Esfuerzo} = \text{E} = abKLDC^{bb}$$
$$\text{Duración} = \text{D} = cbE^{bb}$$

Tabla de valores para COCOMO Intermedio:

Proyecto de SW	ai	bi
Orgánico	3.2	1.05
Semi-acoplado	3.0	1.12
Empotrado	2.8	1.20

Ecuación para calcular estimación para COCOMO Intermedio:

$$\text{Esfuerzo} = \text{E} = aiKLDC^{bi}XFAE$$

Ahora que se ha planteado el Modelo de estimación COCOMO, para nuestro proyecto se ha decidido colocar dentro del submodelo básico en modo semi-acoplado con un número de KLDC aproximado a 800. Por lo que los cálculos para el proyecto son:

$$\text{Esfuerzo} = 3{,}0(800)^{1,12} = 5352$$
$$\text{Duración} = 2{,}5(5352)^{1,12} = 37.486$$

3. Riesgos

En la siguiente tabla podemos observar los riesgos a los que el proyecto esta expuesto, con la probabilidad de que sucedan y el impacto que tendrán que son desde algo sencillo hasta algo critico y de igual forma tenemos las estrategias a tomar para solucionar estos riesgos.

3.1. Tabla de Riesgos

RIESGO	PROBABILIDAD	IMPACTO	ESTRATEGIA DE CONTINGENCIA
RIESGO DEL PROYECTO			
PRESUPUESTO	0.35	MA	Sumar 35% al costo rea de software
PLANIFICACION TEMPORAL	0.05	MA	Realizar un 5% de descuento por cada mes de retraso
PERSONAL (ASIGNACION Y ORGANIZACION)	0.02	DE	Capacitar al personal
RECURSOS	0.30	MA	Optimizar los recursos
CLIENTE Y SUS REQUISITOS	0.15	MA	Buena comunicacion con el cliente
IMPACTO	0.50	MA	Buen desarrollo del prototipo 1
RIESGO TECNICO			
DISENO	0.60	MA	Crear un diseno que le agrade a los usuarios finales
IMPLEMENTACION	0.85	MA	El funcionamiento sea el minimo esperado
INTERFAZ	0.80	MA	La interfaz sea sencilla de usar
VERIFICACION	0.75	MA	Pototipo de preba sea en al menos un mes
MANTENIMIENTO	0.60	MA	Generar versiones de parches de errores
RIESGO DEL NEGOCIO			
RIESGO DEL MERCADO	0.43	Cr	Crear una version de prueba almercado y que tenga aceptacion
RIESGO ESTRATEGICO	0.30	MA	Los usuarios finales requieran cambios en la interfaz
RIESGO DE MERCADOTECNIA	0.16	DE	El software sea comercialiado
RIESGO DE PERDER EL CONTACTO CON EL PERSONAL DEL NEGOCIO	0.20	MA	Tramites legales
CIERRE DEL NEGOCIO	0.36	DE	Que los usuarios prefieran otro tipo de compilador

4. Agenda

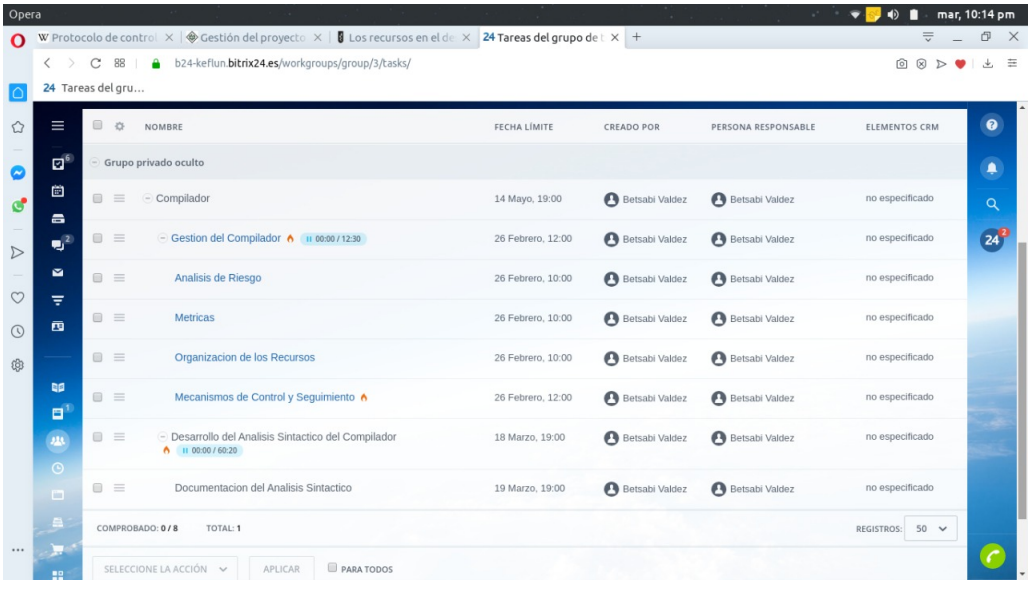
4.1. Cronograma

En el cronograma veremos las actividades a realizar durante 19 semanas aproximadamente, iniciando en la tercera semana de Enero y concluyendo en tercera semana de Mayo, específicamente el día 22 de Mayo de 2020, día en el cual sera expuesto el proyecto en *ExpoEscom*.

Planeación	Actividad/Periodo	Compihinador																		
No. de Actividad	Semanas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	*Introducción*																			
2	*Métricas*																			
3	*Riesgos*																			
4	*Agenda*																			
5	*Recursos*																			
6	*Organización del Personal*																			
7	*Mecanismos de Seguimiento y Control*																			
8	*Analisis*																			
9	*Diseño*																			
10	*Actividades de Negocio*																			
11	*Mercadotecnia*																			
12	*Aspectos de la Mercadotecnia*																			
13	*Calidad*																			
14	*Implantación Satisfactoria*																			
15	*Anexos*																			
16	*Documentación*																			
17	*Programación del Proyecto*																			
18	*Pruebas de Proyecto*																			
19	*Presentación del Proyecto*																			

4.2. Gráfica de Gantt

En la gráfica de Gantt designaremos a los responsables de cada actividad mencionada anteriormente, con una fecha de inicio y una fecha de termino, en algunos casos se presentara mas de 1 responsable, en el caso de que la actividad no pueda ser finalizada en el tiempo establecido, esta tomara la prioridad de todos los integrantes para ser terminada de la manera mas rápida.



5. Recursos

A continuación se especificaran los recursos que utilizaremos durante la elaboración, desarrollo y programación del proyecto, así como en la documentación de este.

5.1. Descripción

5.1.1. Hardware

Los equipos varían un poco respecto a sus características y capacidades, pero cumplen con lo esencial para poder desarrollar el proyecto en cada uno de sus sentidos.

Modelo	Procesador	RAM	Sistema Operativo
Thinkpad	Core i3 1GHz	4 GB	Linux
HP 14 Notebook	AMD E1-2100 1GHz	4 GB	Windows
Lenovo	Core i5 2.49 GHz	8 GB	Linux
Dell Inspiron 3000	Core i3 2.4 GHz	8 GB	Windows

5.1.2. Software

Nombre	Especificaciones	Usabilidad	Flexibilidad	Seguridad
Software de Sistema	Sistema operativo Linux Ubuntu 18.04	Fácil de Usar	Modificable por los desarrolladores	Buen nivel de seguridad
Software de Sistema	Sistema operativo Windows 10	Fácil de Usar	Sin modificaciones por el usuario	Mediano nivel de seguridad
Software de Aplicación	Word	Fácil de Usar	Modificaciones por el usuario	Mediano nivel de seguridad
Software de Aplicación	IDE's	Fácil de Usar	Modificaciones por el usuario	Mediano nivel de seguridad

6. Organización del Personal

Para definir este aspecto del proyecto se tomo en cuenta la organización del personal según Mantei, quien define los siguientes tipos de organización:

Centralizado Controlado (CC)

- El jefe de equipo se encarga de la solución de problemas a alto nivel y la coordinación interna del equipo.
- La comunicación entre el jefe y los miembros del equipo es vertical.

Centralizado Democrático (CD)

- Hay comunicación vertical a lo largo de la jerarquía de control.
- Las decisiones y los enfoques se hacen por consenso.

Descentralizado Controlado (DC)

- Tiene un jefe que coordina tareas específicas y jefes secundarios que tienen responsabilidades sobre subtareas.
- La solución de problemas es una actividad del grupo, pero la implementación de soluciones se reparte entre los subgrupos por el jefe de equipo.

- La comunicación entre subgrupos e individuos es horizontal.
- También hay comunicación vertical a lo largo de la jerarquía de control

Descentralizado Democrático (DD)

- No tiene un jefe permanente. Se nombran coordinadores de tareas a corto plazo y se sustituyen por otros para diferentes tareas.
- Las decisiones y los enfoques se hacen por consenso.
- La comunicación entre los miembros del equipo es horizontal.

6.1. Descentralizado Democrático (DD)

Ya que se conocen los tipos de organización del personal para el proyecto se ha decidido tomar el tipo de organización **descentralizado democrático** porque se planea realizar el proyecto dividiendo la carga de trabajo de forma proporcional manteniendo una comunicación al mismo nivel, es decir, de forma horizontal y que todos los integrantes sean participes de las decisiones y acciones tomadas para la realización del proyecto.

7. Mecanismos de Seguimiento y Control

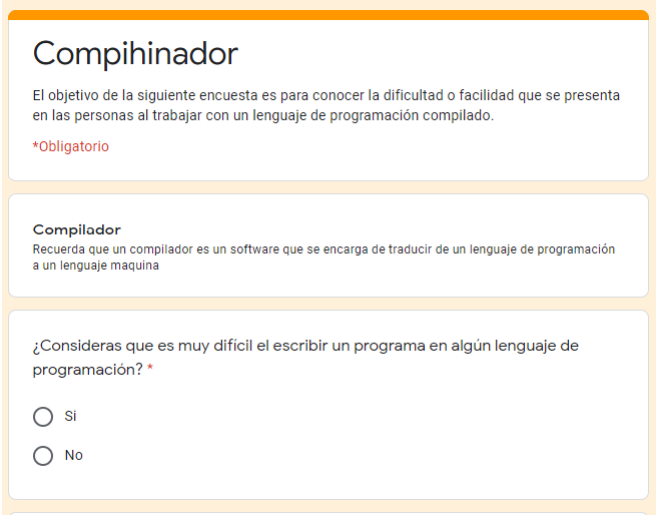
El proyecto presentado a través del diagrama de planificación es realizado con el fin de que cada miembro del equipo cumpla en tiempo y forma las tareas pendientes a realizar, así como de igual forma, en conjunto de las herramientas de desarrollo a trabajar y algunas adecuaciones que en la marcha serán pulidas e implementadas, entre las cuales esta el uso de herramientas online para la escritura de reportes y de control de versiones de los avances que se estén realizando en el avance del proyecto. Cada tarea sera asignada en especifico a algún miembro del equipo con un tiempo definido para ser realizada.

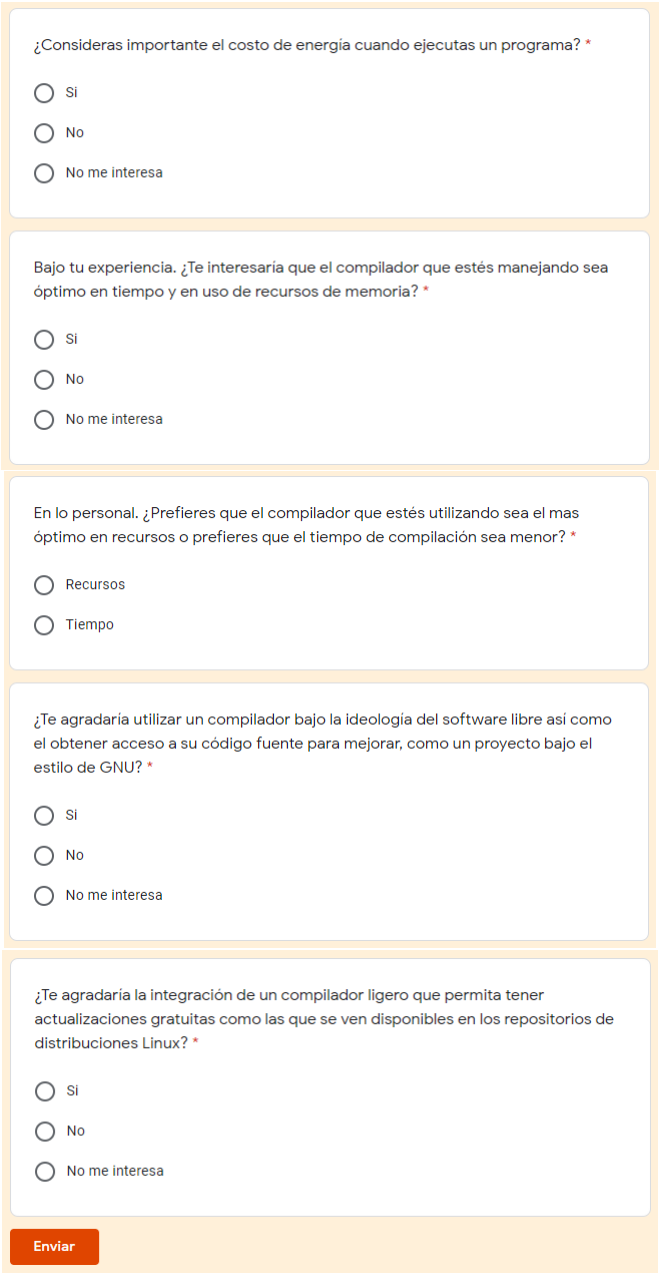
8. Anexos

En esta sección se encuentran todos los enlaces y referencias que se utilizo para la realización del proyecto.

8.1. Encuesta

Encuesta aplicada a usuarios. Enlace aquí





Figuras 5,6,7 y 8: Encuesta aplicada a usuarios acerca del proyecto

8.2. Tabla de Lenguajes

Tabla de lenguajes en uso de energía y recursos, obtenido de: Enlace aquí o escanea: