



# Documento de requerimientos de software

*Fecha:* **[TAJICOR]**  
**[17/06/2020]**

## Tabla de contenido

Historial de Versiones	3
Información del Proyecto	3
Aprobaciones	3
1. Propósito	4
2. Alcance del producto / Software	4
3. Referencias	5
3.1 Referencias técnicas	5
4. Funcionalidades del producto	5
5. Clases y características de usuarios	7
6. Entorno operativo	7
7. Requerimientos funcionales	8
9.1. (Nombre de la funcionalidad 1)	8
9.2. (Nombre de la funcionalidad 2)	8
9.3. (Nombre de la funcionalidad N)	8
8. Reglas de negocio	9
9. Requerimientos de interfaces externas	10
9.1. Interfaces de usuario	10
9.2. Interfaces de hardware	10
9.3. Interfaces de software	11
9.4. Interfaces de comunicación	11
10. Requerimientos no funcionales	11
11. Otros requerimientos	12
12. Glosario	13
13. Manual de Usuario	14

## Historial de Versiones

Fecha	Versión	Organización	Descripción
24/03/2020	1.0	TAJICOR	Compilador para c, primera semana del tutorial de Nora Sandler
16/04/2020	1.3	TAJICOR	Compilador para c, segunda y tercera semana del tutorial de Nora Sandler
28/05/2020	1.3	TAJICOR	Compilador para c, segunda y tercera semana del tutorial de Nora Sandler
17/06/2020	1.4	TAJICOR	Compilador para c, cuarta semana del tutorial de Nora Sandler

## Información del Proyecto

Empresa / Organización	TAJICOR
Proyecto	COMPILADOR
Fecha de preparación	4/02/2020
Cliente	Ing. Norberto Jesus Ortigoza Marquez
Patrocinador principal	Ing. Norberto Jesus Ortigoza Marquez
Líder de Proyecto	Octavio Contró Fernández de Jáuregui
Líder de Análisis de negocio y requerimientos	Octavio Contró Fernández de Jáuregui

## Aprobaciones

Nombre y Apellido	Cargo	Fecha	Firma
Octavio Contró Fernández de Jáuregui	Líder de proyecto	27/02/2020	
Marco Antonio Jiménez Cornejo	Integrador del sistema	27/02/2020	
Mario Tomihuatzin Tabura Sánchez	Diseñador de pruebas del sistema (Tester)	27/02/2020	
Rubén Mendoza Ortega	Arquitecto del Sistema	27/02/2020	
Ing. Norberto Jesus Ortigoza Marquez	Especialista en el área	27/02/2020	



## 1. Propósito

Compilador para c

De acuerdo con lo hablado con el cliente se creará en el lenguaje elixir un compilador para C, el cual en esta primera versión puede compilar un archivo que regresa un único número "integer".

La versión 1 llevará el nombre de TAJICOR , el de la empresa, pues será el objetivo único el lograr este proyecto y lograr la correcta satisfacción del cliente.

Esta versión tendrá el alcance ya mencionado, de regresar un único número integer.

La versión 2 podrá leer e identificar operadores unarios y binarios.

.....

La versión 3 tendrá correcciones detectadas en la versión 2.

.....

La versión 4 añadirá operadores binarios, banderas, también corregirá errores detectados, teniendo en cuenta la precedencia para un mejor funcionamiento.

## 2. Alcance del producto / Software

De acuerdo con lo hablado anteriormente con el cliente, el alcance del proyecto será equivalente a un compilador el cual sea capaz de compilar elementos "integer", operadores unarios y una gran cantidad de operadores binarios.

Para llegar a este alcance propuesto nos basaremos en el tutorial de "Nora Sandler", hasta su cuarta entrega, el cual se puede revisar en su página web.

Con esto esperamos expandir los conocimientos del equipo, respecto al funcionamiento de los compiladores, ya sea para programarlos o comprender su funcionamiento general o específico.

### 3. Referencias

Aquí anotaremos las fuentes de información consultadas, como el propuesto por nuestro especialista.

Writing a C Compiler, Nora Sandler, version 1, 29 Nov 2017,  
nora@norasandler.com <https://norasandler.com/2017/11/29/Write-a-Compiler.html>  
Writing a C Compiler, Nora Sandler, version 2, 05 Dic 2017,  
nora@norasandler.com  
<https://norasandler.com/2017/12/05/Write-a-Compiler-2.html>  
Writing a C Compiler, Nora Sandler, version 3, 15 Dic 2017,  
nora@norasandler.com  
<https://norasandler.com/2017/12/15/Write-a-Compiler-3.html>  
Writing a C Compiler, Nora Sandler, version 4, 27 Dic 2017,  
nora@norasandler.com  
<https://norasandler.com/2017/12/28/Write-a-Compiler-4.html>

#### 3.1 Referencias técnicas

Lecciones básicas del funcionamiento del lenguaje utilizado.  
<https://elixirschool.com/es/>  
Manejador de versiones para un mejor manejo de la información.  
<https://github.com/hiphoox/c202-tajicoor>  
Uso de herramientas de comunicación.  
<https://app.slack.com>  
<https://web.whatsapp.com/>  
<https://zoom.us/>

## 4. Funcionalidades del producto

Se construirá un compilador para lenguaje c en la arquitectura de 64 bits, en el sistema operativo de UNIX.

En nuestras referencias anotamos las fuentes de información en la cual se basa nuestro producto, principalmente en las páginas web de Nora Sandler, este tendrá las funcionalidades allí especificadas, lo que quiere decir que será capaz de compilar los siguientes elementos:

- Elementos "int" o "Integers"
- Operadores unarios
- Operadores binarios
- Más operadores binarios

En la versión 1 el funcionamiento del compilador se limita a, compilar un archivo cuyo contenido es la función main y regresa un valor "integer". Este compilador sólo puede utilizarse en UNIX, dado que, no se han agregado las funcionalidades para otros sistemas operativos.

En la versión 2 el compilador podrá, compilar un archivo que contenga cualquier funcionamiento de la versión 1, e identificar operadores binarios y unarios, además generará un archivo ejecutable y ast.


Las operaciones unarias son:

- Negation (-) Negative Numbers
- Bitwise (~) This flips every bit in a number.
- Logical negation (!) The boolean "not" operator.

En la versión 3 incluimos algunos operadores binarios básicos como lo pide el Blog de Nora, de estos se presentan algunos problemas como es el orden o precedencia de las operaciones por jerarquías, ya que en la construcción de los árboles (AST), la forma de lectura no puede ser la misma que en las anteriores construcciones.

Las operaciones binarias son:

- Addition (+)
- Subtraction (-)
- Multiplication (\*)
- División (/)



En la versión 4, fueron agregados más operadores binarios, a pesar de que se manejaron aproximadamente las mismas normas que en la entrega anterior, se toman en cuenta algunas normas extra para los siguientes operadores:

- Logical And &&
- Logical OR ||
- Equal to ==
- Not equal to !=
- Less than <
- Less than or equal to <=
- Greater then >
- Greater than or equal to >=

## 5. Clases y características de usuarios

Nuestro producto requiere de usuarios con experiencia en al menos el lenguaje de programación C y uso de el sistema operativo UNIX, si se tiene experiencia en el lenguaje elixir ayudará al uso del mismo. Por lo que clasificamos a nuestros usuarios en:

- Clase 1: Este usuario podrá usar el compilador de una forma básica con sus funciones mínimas, para esto necesitará experiencia en uso básico de ambos casos UNIX y Lenguaje de programación C.
- Clase 2: Este usuario podrá hacer cambios estructurales para mejorar el código del compilador y utilizarlo a su máxima capacidad, pero para ello necesitará un conocimiento mediano o avanzado en los lenguajes de programación C y elixir, y un mediano o avanzado conocimiento en el sistema operativo UNIX. también requiere de conocimientos básicos de cómo funcionan los compiladores, así como las partes que lo componen.

Esta clasificación se refiere a los niveles de seguridad en el producto en el primer nivel el producto deberá compilar con las funcionalidades especificadas (sección 9 y 10). En el segundo nivel estas pueden fallar pues el usuario será responsable de cualquier cambio realizado al código que compone el compilador, lo cual libera a la empresa de cualquier responsabilidad.

## 6. Entorno operativo

El compilador será desarrollado con la arquitectura de 64 bits, pues así lo solicito el cliente.

Nuestro entorno operativo principal y donde el compilador funciona correctamente será UNIX.

Para el desarrollo del compilador usaremos la plataforma de github.com para poder controlar las versiones del software a desarrollar. Esta herramienta nos es muy útil, pues nos permite recuperar versiones anteriores a una modificación y trabajar remotamente desde nuestro equipo, y en caso de resolver alguna problemática podemos subir nuestra solución a GitHub y los demás podrán revisarla, igualmente de forma remota o localmente en github y editarla si así lo desean, de esta forma cada uno podrá utilizar el software o entorno de su conveniencia para el lenguaje que usaremos para el compilador.

También esta plataforma nos permite regresar a versiones anteriores para poder deshacer ediciones muy grandes o que provocan el mal funcionamiento del compilador.

Usaremos slack para controlar las tareas y metas a realizar del proyecto, de esta forma aseguramos estar en tiempo y forma para las entregas.



## 7. Requerimientos funcionales

En principio necesitamos saber el tipo de hardware que estamos utilizando para el uso del compilador (En este caso es basado en los requerimientos del cliente):

Hablamos de una PC con un procesador de 64 bits, suficiente memoria Ram para el sistema que queramos utilizar y los periféricos necesarios para la comodidad del usuario final (mouse, teclado) y una pantalla de tamaño decente, ya que el uso prolongado de una pc puede agotar la vista y al manejar el compilador mediante línea de comandos es cansado en una pantalla pequeña (aunque no imposible).

En el caso del software:

Un sistema operativo basado en unix (también de 64 bits) y conocimiento básico del manejo del mismo, ya que no todos los software son tan intuitivos en las diferentes distribuciones.

## 8. Reglas de negocio

El producto será desarrollado conforme a las especificaciones del cliente las cuales son:

- Arquitectura de 64 bits
- Compilador para lenguaje de programación c, desarrollado en lenguaje de programación elixir
- Sistema operativo (no especificado).
- Podrá compilar como está especificado en el tutorial de nora hasta su cuarta entrega:
  - integers
  - operadores unarios
  - operadores binarios

El cliente retribuirá a la empresa con:

- Calificación por miembro para ser sentada en actas.
- En cada etapa una aprobación para continuar con el proyecto o producto

Estas fueron las especificaciones del cliente para la empresa, entre más cercanas estén a sus expectativas, el cliente retribuirá a la empresa con una calificación la cual tiene por propósito calificar la satisfacción del cliente con el producto como el cumplimiento de los requisitos acordados.

Esta calificación tendrá que ser validada por los miembros de la empresa, en caso de ser validada será sentada en actas (sistemas siae de la UNAM).

## 9. Requerimientos de interfaces externas

### 9.1. Interfaces de usuario

Para acceder correctamente al compilador seguir los siguientes pasos desde el escritorio de el sistema operativo UNIX:

1. Utilizando la terminal de comandos accedemos a la carpeta "Compilador".
2. Ejecutamos el comando "mix escript.build".
3. Ejecutamos nuestro compilador sobre un archivo ".c".
4. "./compitajicoor --verbose semana4.c"
5. Se compila el archivo ".c" y genera un ejecutable.

\*opcional 1:

6. Ejecutamos el archivo generado en el punto anterior "./a.out".

\*opcional 2:

7. Ejecutamos "echo \$?" para ver lo que obtuvimos de compilar nuestro archivo ".c".

### 9.2. Interfaces de hardware

De momento solo se puede ocupar en la implementación del compilador, una computadora con procesador basado en una arquitectura de 64 bits y un sistema operativo basado en alguna versión de Ubuntu (en sus diferentes distribuciones), ya que no ha sido probado completamente fuera de esas especificaciones y podría llegar a generar un error en otras interfaces.

El proyecto se basa en realizar un compilador de lenguaje C, del cual ya existen diferentes versiones, por tanto lo que se busca en este proyecto es crear una forma más efectiva de compilar los programas necesarios sin abrumar los recursos como pueden llegar a manifestar otros programas preexistentes.

En este caso, es necesario tener un software basado en X64 como lo es Elixir y que ocupa menos memoria para su ejecución, así como también, se reconoce que

---

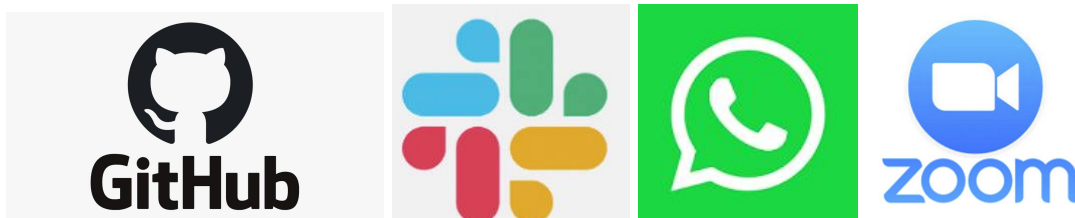
es más sencillo en la creación de compiladores, ya que contiene funciones preestablecidas para la facilitación de este tipo de proyectos.

### 9.3. Interfaces de software

La interfaz principal a utilizar será el sistema operativo UNIX con su ventana de comandos. también se puede utilizar una máquina virtual para simular el entorno de UNIX y de esta forma utilizar el compilador con los pasos ya señalados. Además tener las especificaciones del lenguaje C y Elixir instaladas para el mejor funcionamiento del compilador, sin olvidar el GCC para la creación de archivos ejecutables.

### 9.4. Interfaces de comunicación

Como medio principal de comunicación usaremos la aplicación “whatsapp”, también nos apoyaremos con “slack” para dar seguimiento del proyecto y “zoom” para situaciones más complejas o dudas más específicas. usaremos github como medio principal de versionamiento para el proyecto. También por la situación actual mundial (coronavirus) las reuniones físicas serán canceladas.



## 10. Requerimientos no funcionales

En este caso, hacemos referencia al sistema nativo en el que se manejara nuestro compilador, en cuyo caso es ajeno a otros sistemas operativos diferentes a Unix basados en las diferentes distribuciones de Ubuntu, en el cual está basado el desarrollo de nuestro compilador.

También ocupamos un requerimiento para omitir diferentes tipos de procesadores, ya que dentro de las especificaciones, se pidió específicamente que funcionara en un procesador X64, el cual tiene algunas restricciones de compatibilidad con los diferentes grupos de familias de procesadores basados en X86 .

El entorno de desarrollo se limita a Unix (distribuciones de Ubuntu), ya que las Pc ocupadas al momento de la implementación del proyecto con otros sistemas operativos se limitaban a windows en 32 bits, por tanto, es una limitación dentro de nuestro compilador.

Otra de las limitaciones que tiene el software desarrollado, es que, se necesita un usuario con conocimientos previos en el sistema operativo nativo del compilador desarrollado, así como también en el lenguaje C.

Basado en la premisa de que se puede llegar a modificar el código del proyecto en un futuro, también se necesita la instalación del software “Elixir”, ya que este lenguaje se ocupó de base para su desarrollo y por los requisitos del mismo software, es necesaria una computadora con arquitectura de 64 bits, ya que no existe versión de 32bits conocida.

Por último, en el caso de seguir con las necesidades básicas de control de versiones, se recomienda el uso de GitHub, ya que desde esa herramienta se creó el repositorio del proyecto en curso y así se tendría la accesibilidad necesaria para acceder a todas las partes del software en proceso de desarrollo y no se necesitaría el mudar los datos a otro servidor.

## 11. Otros requerimientos

Como sabemos, existen varias versiones de compiladores de lenguaje C, dentro del marco legal no podemos excluir el uso de un manual en el cual se basan algunos otros programadores (tanto compañeros como ajenos a nosotros), por lo cual es necesario tener un control de versiones, documentar las acciones dentro de lo necesario para la creación de nuestro proyecto y con esto, poder ampararse dentro de dicho marco, en contra de llegar a una estructura parecida de otro equipo de programación, ya que como se indica anteriormente, nuestro proyecto está enfocado en un compilador de lenguaje C, del cual ya hay varios software.

## 12. Glosario

**X86:** La familia x86 reagrupa los microprocesadores compatibles con el juego de instrucciones Intel 8086. Por tanto, x86 representa a ese conjunto de instrucciones, siendo también una denominación genérica dada a los correspondientes microprocesadores.

**X64:** x86-64 (también conocido como x64, x86\_64 y AMD64) es la versión de 64 bits del conjunto de instrucciones x86. Soporta una cantidad mucho mayor de memoria virtual y memoria física de lo que le es posible a sus predecesores, permitiendo a los programas almacenar grandes cantidades de datos en la memoria.

**UNIX:** es un sistema operativo portable, multitarea y multiusuario

**GCC:** GCC es parte del proyecto GNU, y tiene como objetivo mejorar el compilador usado en todos los sistemas GNU, incluyendo la variante GNU/Linux.

**Ubuntu:** es un sistema operativo de software libre y código abierto

**integer:** Es una representación de entero de un número, una representación de serie de un número, un valor de fecha o un valor de hora

**operadores unarios:** son aquellos que solo requieren un operando para funcionar.

**operadores binarios:** Son los que usan dos operandos, por ejemplo todos los operadores aritméticos: suma, resta, multiplicación, división, etc.

## 13. Manual de usuario

Para poder utilizar correctamente el compilador es necesario tener y saber los principios básicos de:

- Sistema operativo UNIX

<https://ubunlog.com/category/distribuciones-basadas-en-ubuntu/> (algunas distribuciones disponibles compatibles con el proyecto)

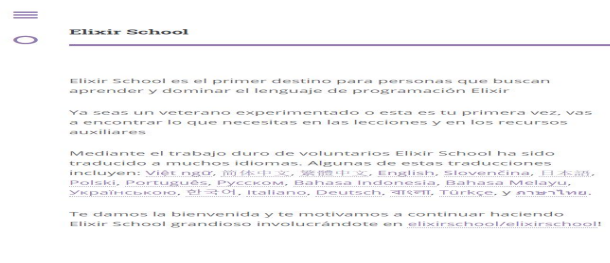
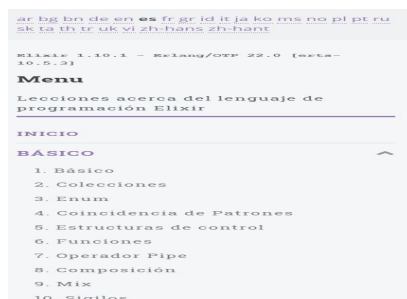


- Lenguaje de programación Elixir

<https://elixir-lang.org/install.html> (Instalacion de lenguaje Elixir)



<https://elixirschool.com/es/> (curso basico de lenguaje Elixir)



- Lenguaje de programación c

<https://sourceforge.net/projects/orwelldvcpp/> (ide utilizada en la facultad)



<https://paginas.matem.unam.mx/pderbf/images/mprogintc++.pdf> (manual de usuario de la unam para lenguaje C++ en linux)

## Manual de Programación en Lenguaje C++

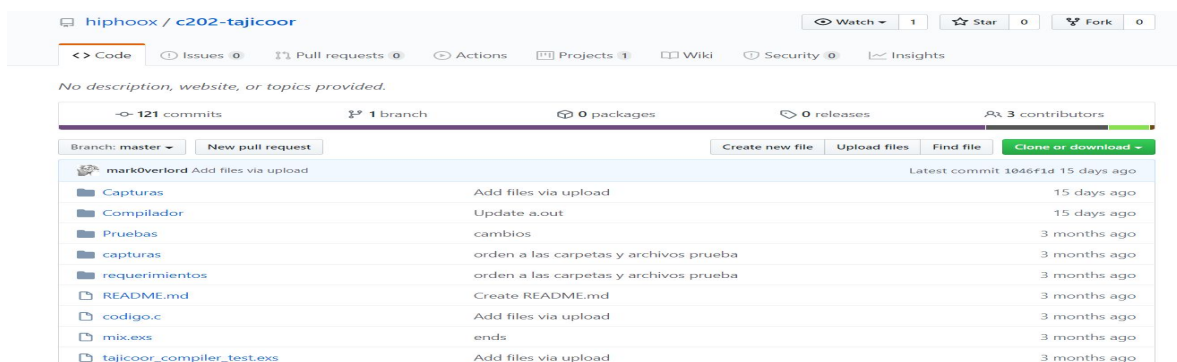
Proyecto de Investigación: Métodos de Funciones de Base Radial para la Solución de EDP.

Servicio Social - DGSCA-UNAM, 2008.

Autor: Linda I. Olivares Flores

Ya instalados los lenguajes, descargamos desde el repositorio el compilador:

<https://github.com/hiphoox/c202-tajicoor>



Ahora para compilar correctamente un archivo .c siga los siguientes pasos:

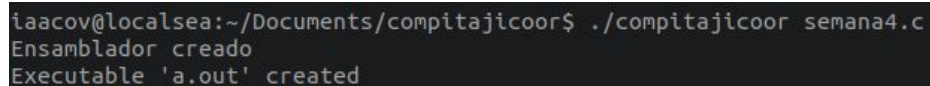
1. Utilizando la terminal de comandos accedemos a la carpeta "Compilador".
2. Ejecutamos el comando "mix escript.build".
3. Ejecutamos nuestro compilador sobre un archivo ".c".
4. "./compitajicoor --verbose semana4.c"
5. Se compila el archivo ".c" y genera un ejecutable.

\*opcional 1:

6. Ejecutamos el archivo generado en el punto anterior "./a.out".

\*opcional 2:

7. Ejecutamos "echo \$?" para ver lo que obtuvimos de compilar nuestro archivo ".c".

A terminal window with a dark background. The prompt is 'iaacov@localsea:~/Documents/compitajicoor\$'. The command entered is './compitajicoor semana4.c'. The output shows 'Ensamblador creado' and 'Executable 'a.out' created' on two separate lines.

```
iaacov@localsea:~/Documents/compitajicoor$ ./compitajicoor semana4.c
Ensamblador creado
Executable 'a.out' created
```

(Imagen muestra de línea de comandos)

## Glosario:

**Aqui se mostraran las diferentes banderas utilizadas en la línea de comandos las cuales nos ayudaran a saber qué y cómo se están realizando los diferentes procesos.**

--verbose: Muestra los pasos que realiza para compilar el archivo, Muestra el código del programa, el AST generado, así como el ensamblador creado.

--o nombreEjecutable: Llama al programa ejecutable como nombreEjecutable