PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUÍA DE APRENDIZAJE

- **Denominación del programa de formación:** Análisis y desarrollo de software.
- **Código del programa de formación:** 228118
- **Nombre del proyecto:** Construcción de software integrador de tecnologías orientadas a servicios.
- **Fase del proyecto:** Ejecución.
- **Actividad de proyecto:** Codificar los módulos del software.
- **Competencias:**
  Técnicas:
  220501096 - Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo.
- **Resultados de aprendizaje a alcanzar:**
  Técnicos:
  220501096-01 - Planear actividades de construcción del software de acuerdo con el diseño establecido.
  220501096-04 - Codificar el software de acuerdo con el diseño establecido.

- **Duración de la guía:** 48 horas Técnico: 48 horas.

2. PRESENTACIÓN

Estimado aprendiz, el SENA extiende una cordial bienvenida a la guía de aprendizaje que comprende la competencia técnica de: desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo. De acuerdo con Maida y Pacienzia (2015): la metodología en el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto que comprende los procesos a seguir para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Para el desarrollo de las actividades planteadas en esta guía, contará con el acompañamiento de los instructores asignados al programa, los cuales de forma continua y permanente lo orientarán con las pautas necesarias para el logro de las actividades de aprendizaje, brindando herramientas básicas de tipo conceptual y metodológico. Los instructores programarán encuentros de asesoría virtual, para brindar orientaciones específicas relacionadas con las temáticas a desarrollar en las actividades. Es importante que organice su tiempo, dada la exigencia que demanda la realización de esta guía de aprendizaje.

Por consiguiente, se presentan cada una de las acciones de aprendizaje que le permitirán desarrollar lo anteriormente mencionado.

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

En este apartado se describirán las actividades de aprendizaje para cada una de las competencias que plantea la fase de ejecución del proyecto formativo: construcción de software integrador de tecnologías orientadas a servicios.

3.1. Actividades de aprendizaje de la competencia 220501096 - Desarrollar la solución de software de acuerdo con el diseño y metodologías de desarrollo

Con base en las metodologías de desarrollo utilizadas en esta competencia se inicia con el desarrollo de las aplicaciones web y móviles utilizando Framework para el desarrollo ágil entre los que se encuentran SprintBoot, React, Android, Swift del lado del servidor Node, go, que en conjunto con las herramientas de versionamiento permiten realizar grandes proyectos en tiempos cortos, después de estudiadas cada una de estas tecnologías podrá escoger la que más se adapte a las características del proyecto a realizar.

3.1.1 Actividad de aprendizaje GA7-220501096-AA1 configurar herramientas de versionamiento para control de código

Duración: 12 horas.

Materiales de formación: para el desarrollo de esta actividad es importante la lectura y análisis de los componentes formativos: “Aplicación del paradigma orientado a objetos” e “Integración continua”.

Evidencias:

A continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje:

- **Evidencia de conocimiento: GA7-220501096-AA1-EV01 identifica herramientas de versionamiento**

Tomando como referencia el componente formativo “Integración continua”, realice una tabla con las diferencias entre el sistema de control de versionamiento git local y git remoto.

Elementos para tener en cuenta en el documento:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, tablas con diferencias y características y comandos de git local y git remoto.
- Realice una tabla con la descripción de los comandos básicos de git remoto y git local

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** documento informe técnico
- **Extensión:** Libre.
- **Formato:** PDF.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio asignado.

Diferencias entre el sistema de control de versionamiento git local y git remoto:

Características	Git Local	Git Remoto
Ubicación	Se ejecuta en el sistema local del usuario.	Se ejecuta en un servidor remoto.

Almacenamiento	Almacena el historial y versiones en el disco local.	Almacena el historial y versiones en un servidor (p.ej. GitHub, GitLab).
Acceso	Acceso directo y rápido a los archivos y commits.	Requiere conexión a Internet para acceder.
Colaboración	Individual, solo el usuario tiene acceso.	Permite colaboración entre múltiples usuarios.
Sincronización	No necesita sincronización, trabaja con datos locales.	Necesita sincronización con el repositorio local.
Comandos	Utiliza comandos para gestionar cambios locales.	Utiliza comandos para gestionar el estado del repositorio remoto.
Backup	No es un respaldo a menos que se guarde en otro medio.	Sirve como respaldo centralizado de versiones.

Comandos de git local y git remoto:

Comando	Descripción	Tipo
`git init`	Crea una copia local de un repositorio remoto.	Local
`git clone <url>`	Añade un archivo al área de preparación.	Remoto
`git add <archivo>`	Guarda los cambios en el repositorio local.	Local
`git commit -m "<mensaje>"`	Muestra el estado del repositorio local.	Local
`git status`	Envía los cambios locales al repositorio remoto.	Local
`git push`	Actualiza la copia local con los cambios del remoto.	Remoto
`git pull`	Descarga los cambios del remoto sin aplicarlos.	Remoto
`git fetch`	Muestra las ramas locales.	Remoto
`git branch`	Cambia a una rama específica.	Local
`git checkout <rama>`	Crea una copia local de un repositorio remoto.	Local

- **Evidencia de desempeño: GA7-220501096-AA1-EV02 instalación y configuración de herramienta de versionamiento (Local / Web)**

Tomando como referencia el componente formativo “Integración continua”, realice la instalación y configuración de las herramientas de control de versionamiento tanto local como remoto.

Elementos para tener en cuenta en el documento:

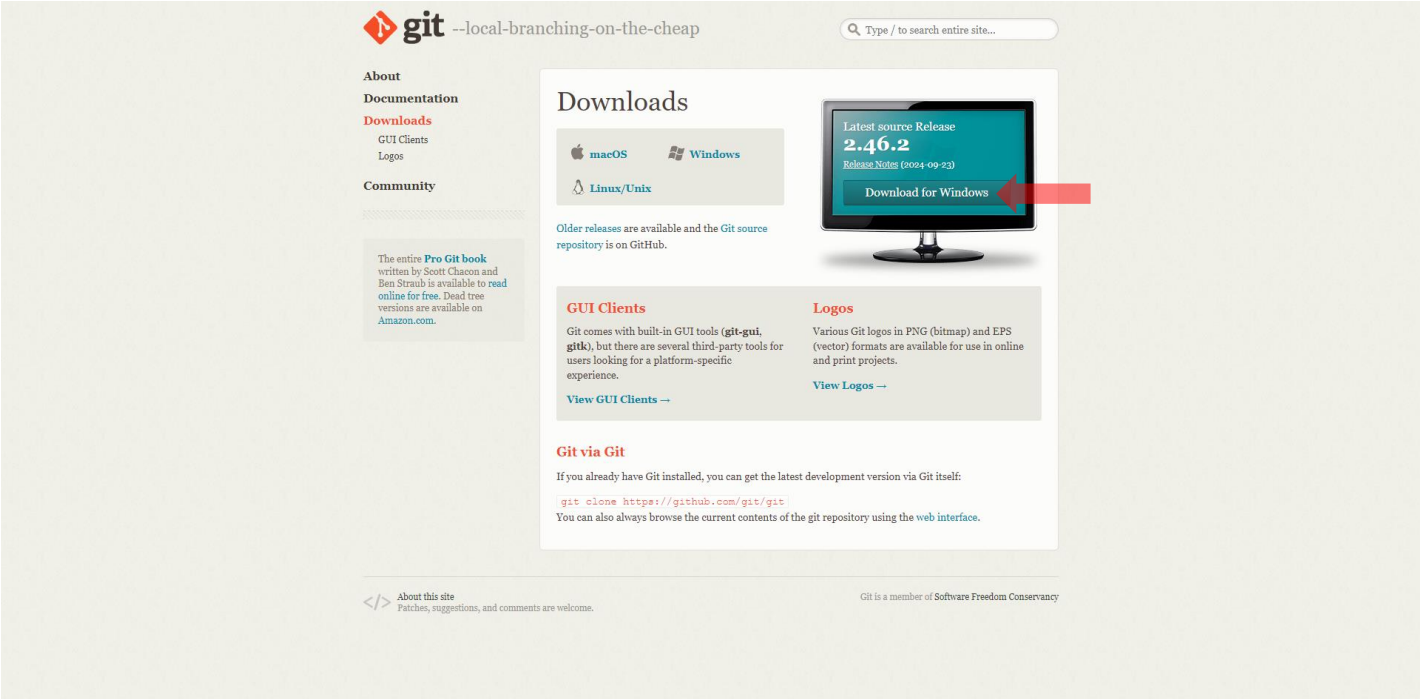
- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, paso a paso con pantallazos de la instalación de las herramientas de control de versionamiento tanto local como remota.

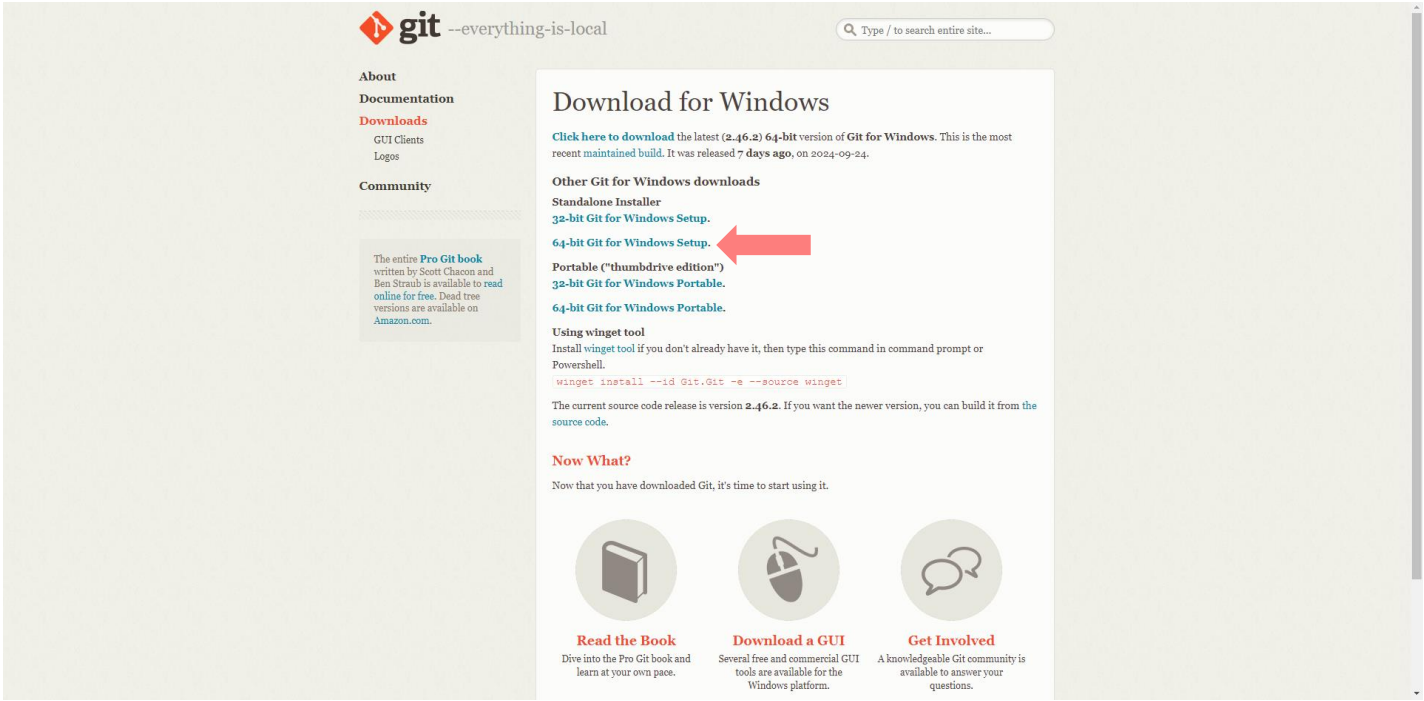
Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** documento instalación.
- **Extensión:** Libre.
- **Formato:** PDF.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio correspondiente.

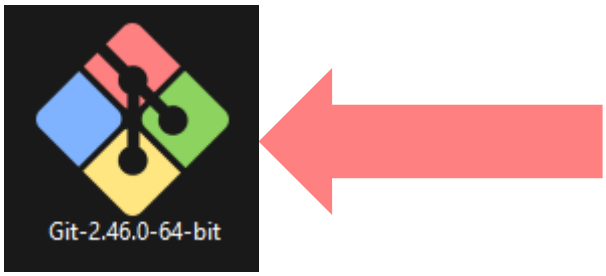
Instalación de las herramientas de control de versionamiento tanto local como remota:

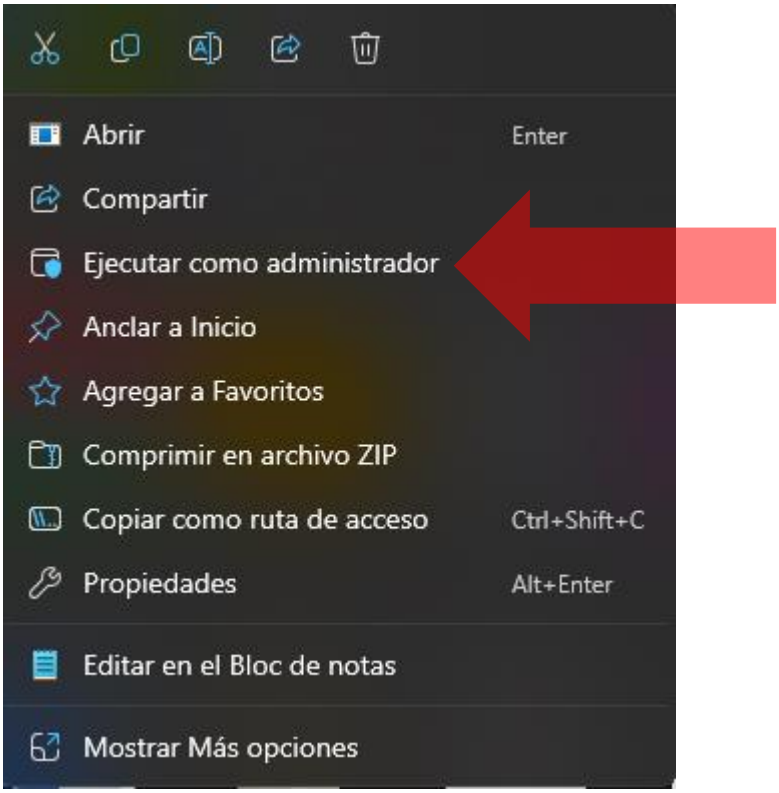
Git (Local)
https://git-scm.com/downloads



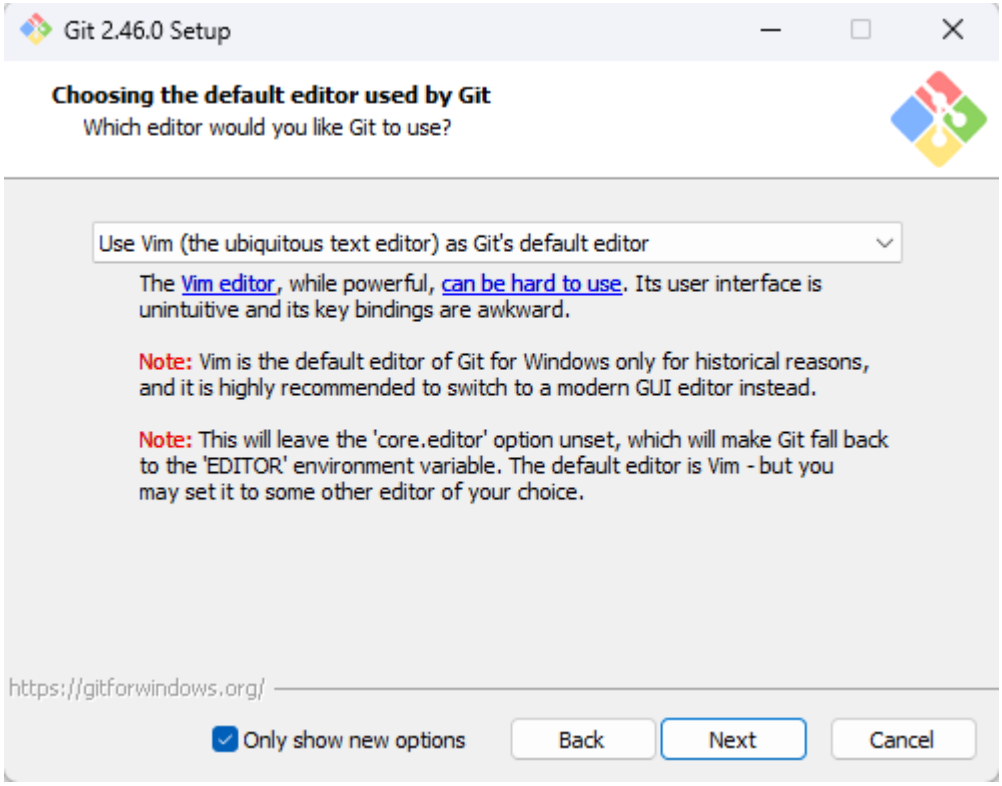

git --everything-is-local
Type / to search entire site...
About
Documentation
Downloads
GUI Clients
Logos
Community
The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.
Download for Windows
Click here to download the latest (2.46.2) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 7 days ago, on 2024-09-24.
Other Git for Windows downloads
Standalone Installer
32-bit Git for Windows Setup.
64-bit Git for Windows Setup.
Portable ("thumbdrive edition")
32-bit Git for Windows Portable.
64-bit Git for Windows Portable.
Using winget tool
Install winget tool if you don't already have it, then type this command in command prompt or Powershell.
winget install --id Git.Git -e --source winget
The current source code release is version 2.46.2. If you want the newer version, you can build it from the source code.
Now What?
Now that you have downloaded Git, it's time to start using it.
Read the Book
Dive into the Pro Git book and learn at your own pace.
Download a GUI
Several free and commercial GUI tools are available for the Windows platform.
Get Involved
A knowledgeable Git community is available to answer your questions.

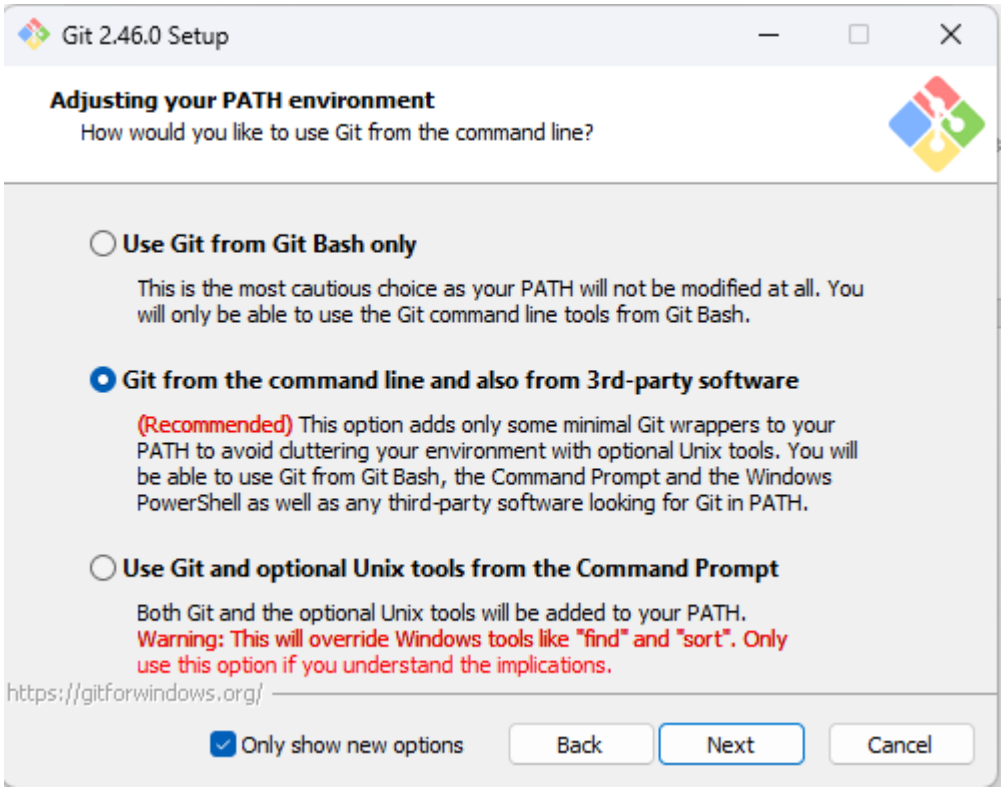

Git-2.46.0-64-bit.exe

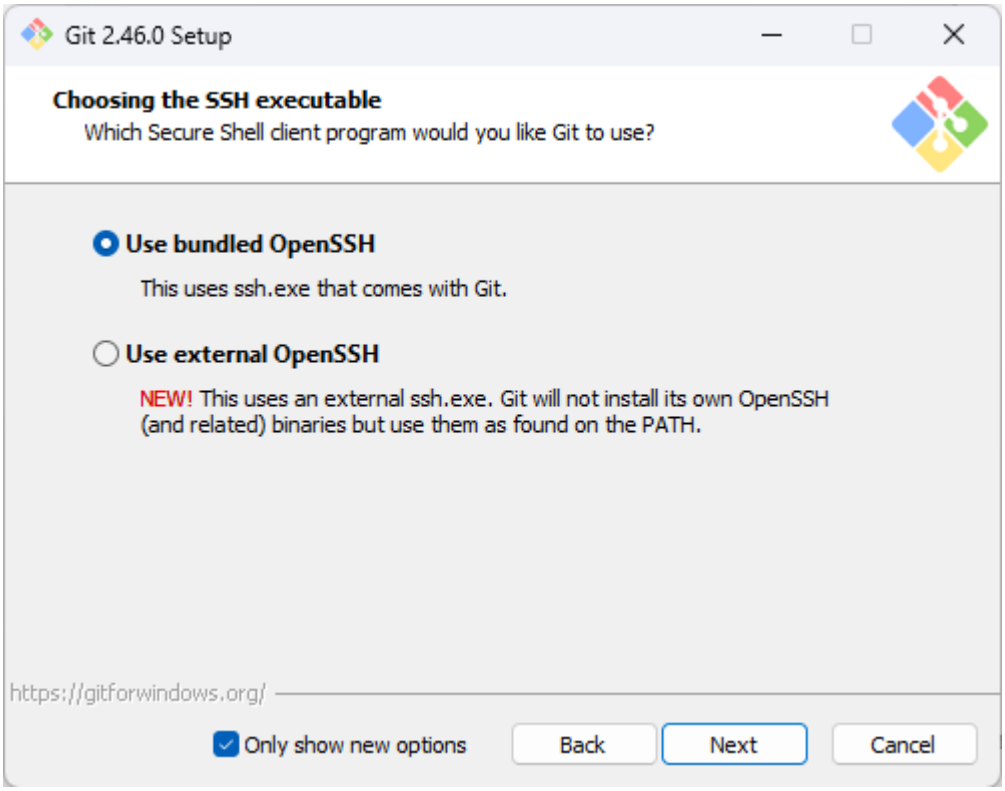

Git-2.46.0-64-bit

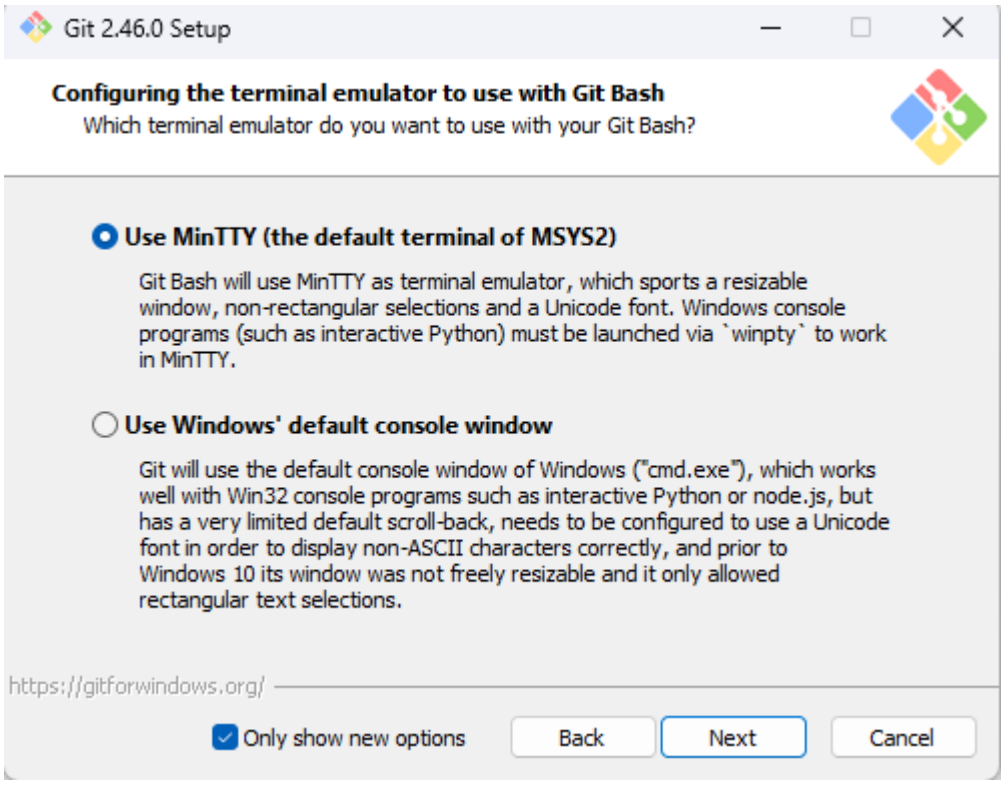

Abrir
Enter
Compartir
Ejecutar como administrador
Anclar a Inicio
Agregar a Favoritos
Comprimir en archivo ZIP
Copiar como ruta de acceso
Ctrl+Shift+C
Propiedades
Alt+Enter
Editar en el Bloc de notas
Mostrar Más opciones

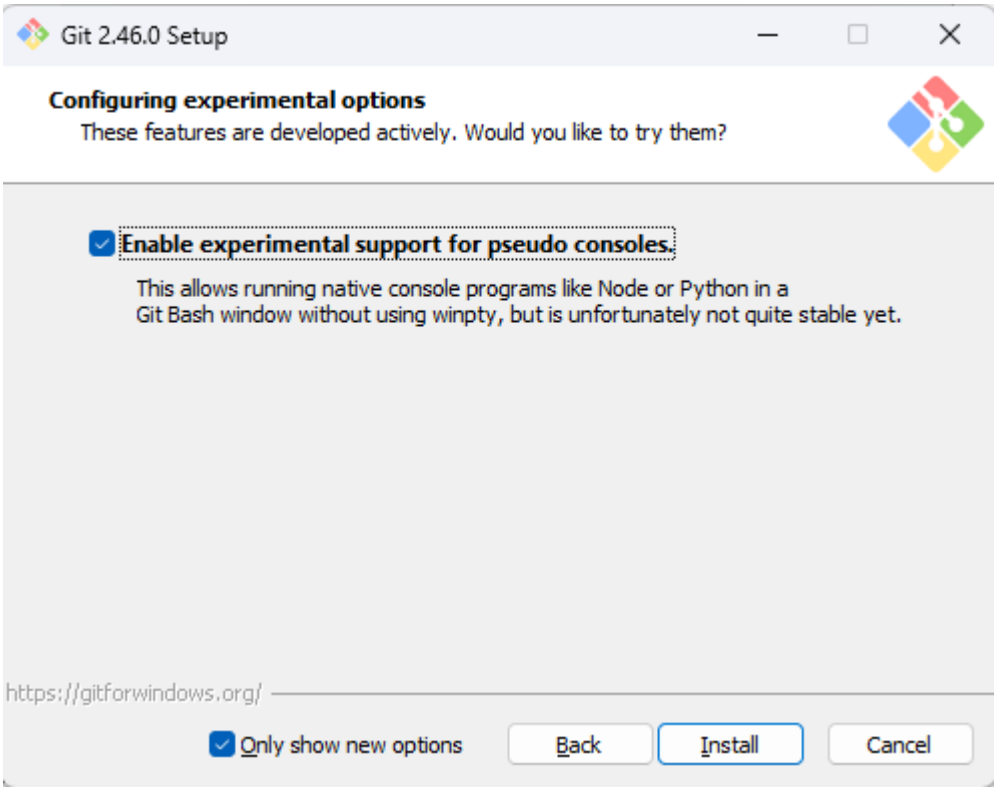

Git 2.46.0 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
GNU General Public License
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change
https://gitforwindows.org/
Only show new options
Next
Cancel

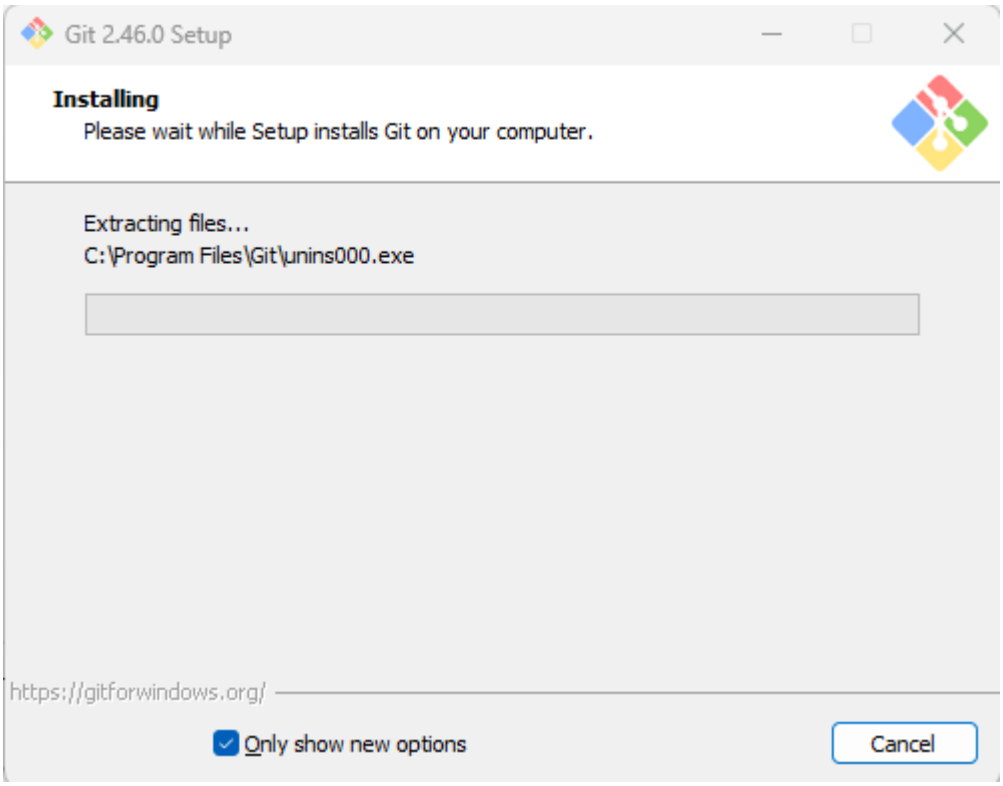

Git 2.46.0 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

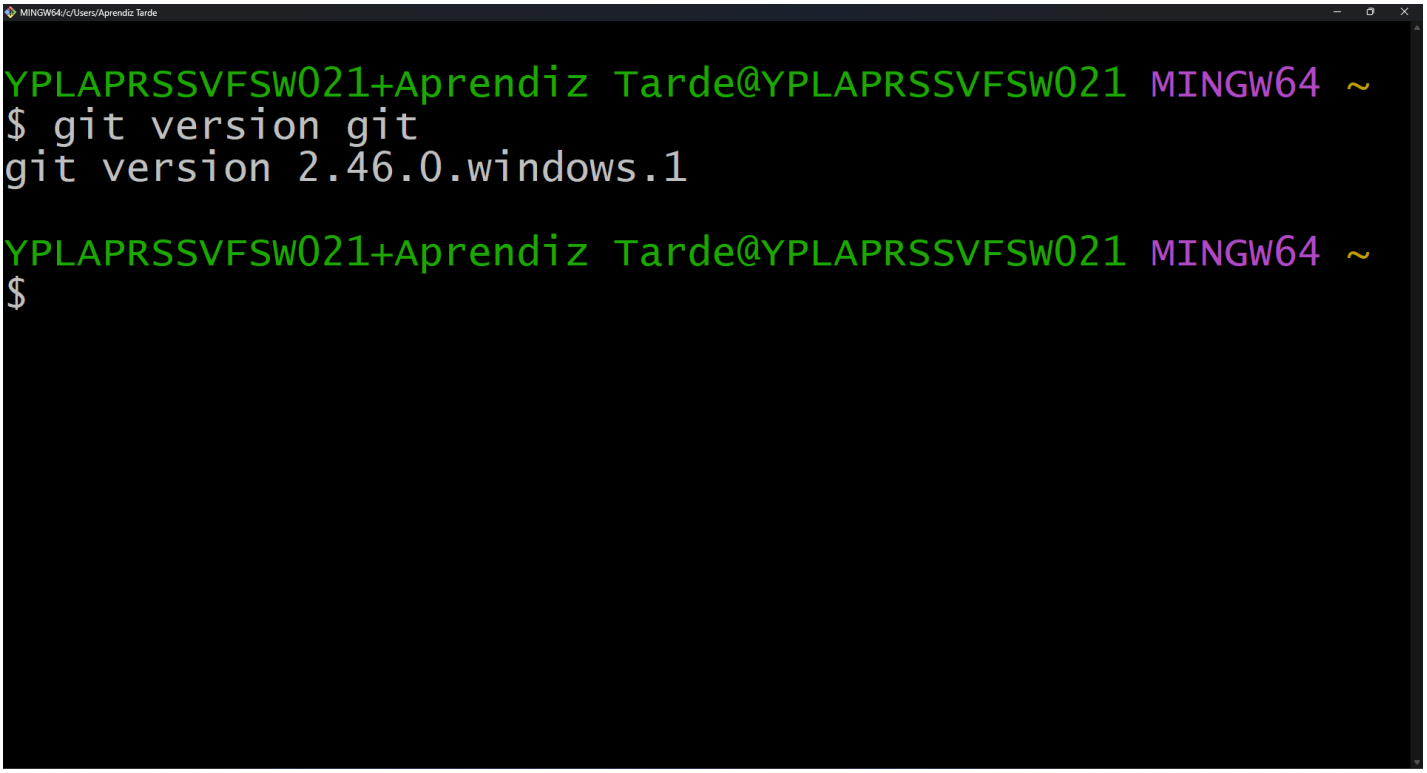

Git 2.46.0 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel


Git 2.46.0 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

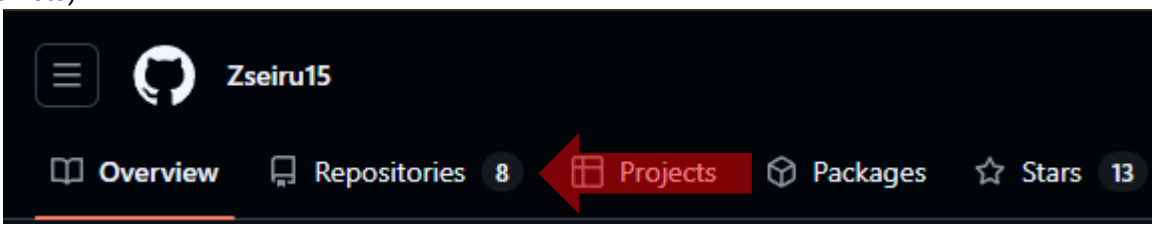

Git 2.46.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

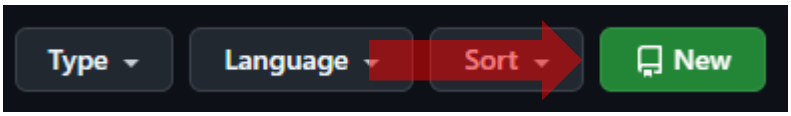

Git 2.46.0 Setup
Configuring experimental options
These features are developed actively. Would you like to try them?
Enable experimental support for pseudo consoles.
This allows running native console programs like Node or Python in a
Git Bash window without using winpty, but is unfortunately not quite stable yet.
https://gitforwindows.org/
Only show new options
Back
Install
Cancel

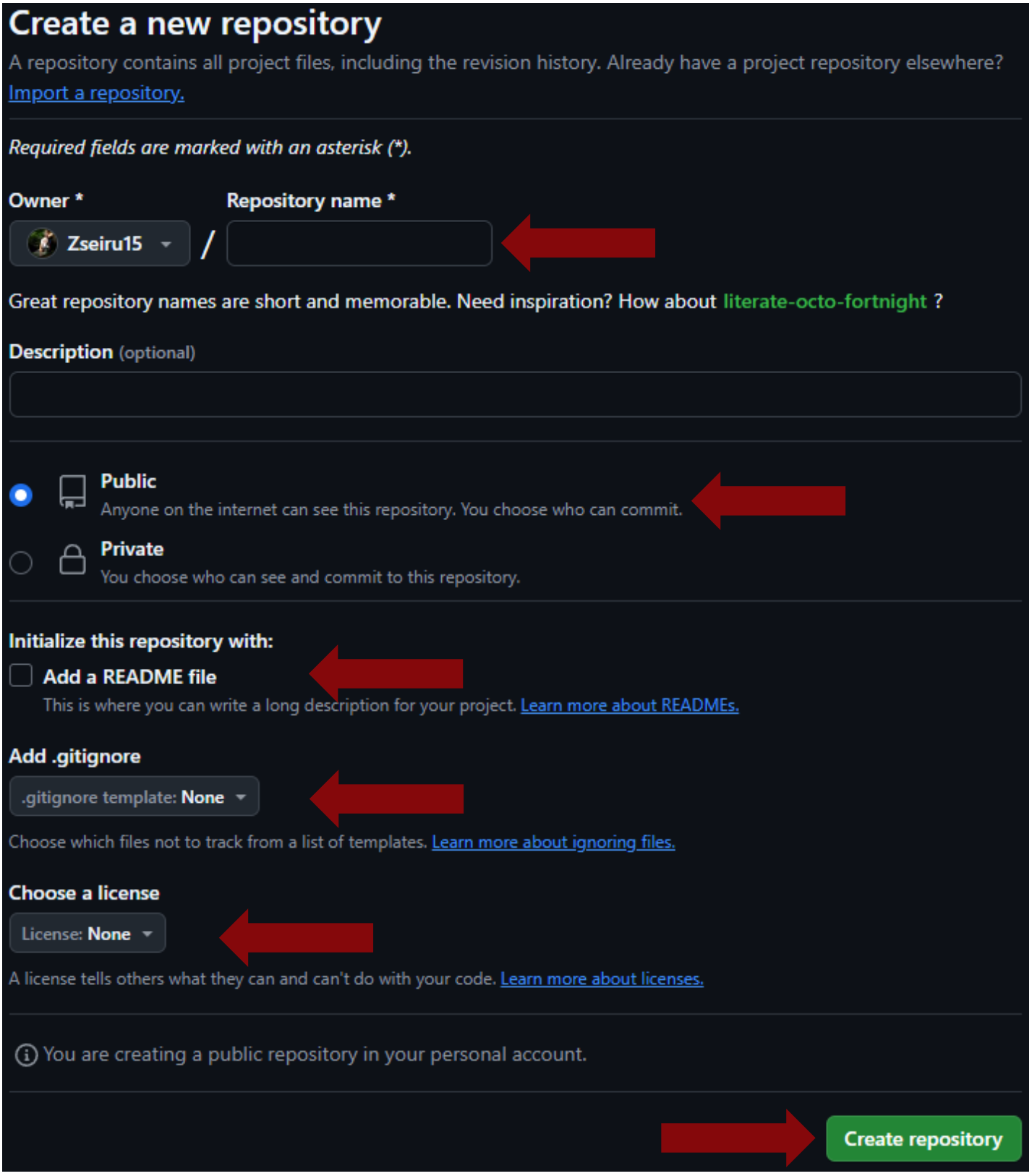

Git 2.46.0 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files\Git\unins000.exe
https://gitforwindows.org/
Only show new options
Cancel


MINGW64:/c/Users/Aprendiz Tarde
YPLAPRSSVFSW021+Aprendiz Tarde@YPLAPRSSVFSW021 MINGW64 ~
$ git version git
git version 2.46.0.windows.1
YPLAPRSSVFSW021+Aprendiz Tarde@YPLAPRSSVFSW021 MINGW64 ~
$

Git (Remoto)


Zseiru15
Overview
Repositories 8
Projects
Packages
Stars 13


Type
Language
Sort
New


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Required fields are marked with an asterisk (*).
Owner *
Repository name *
Zseiru15
/
Great repository names are short and memorable. Need inspiration? How about literate-octo-fortnight ?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.
Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.
You are creating a public repository in your personal account.
Create repository

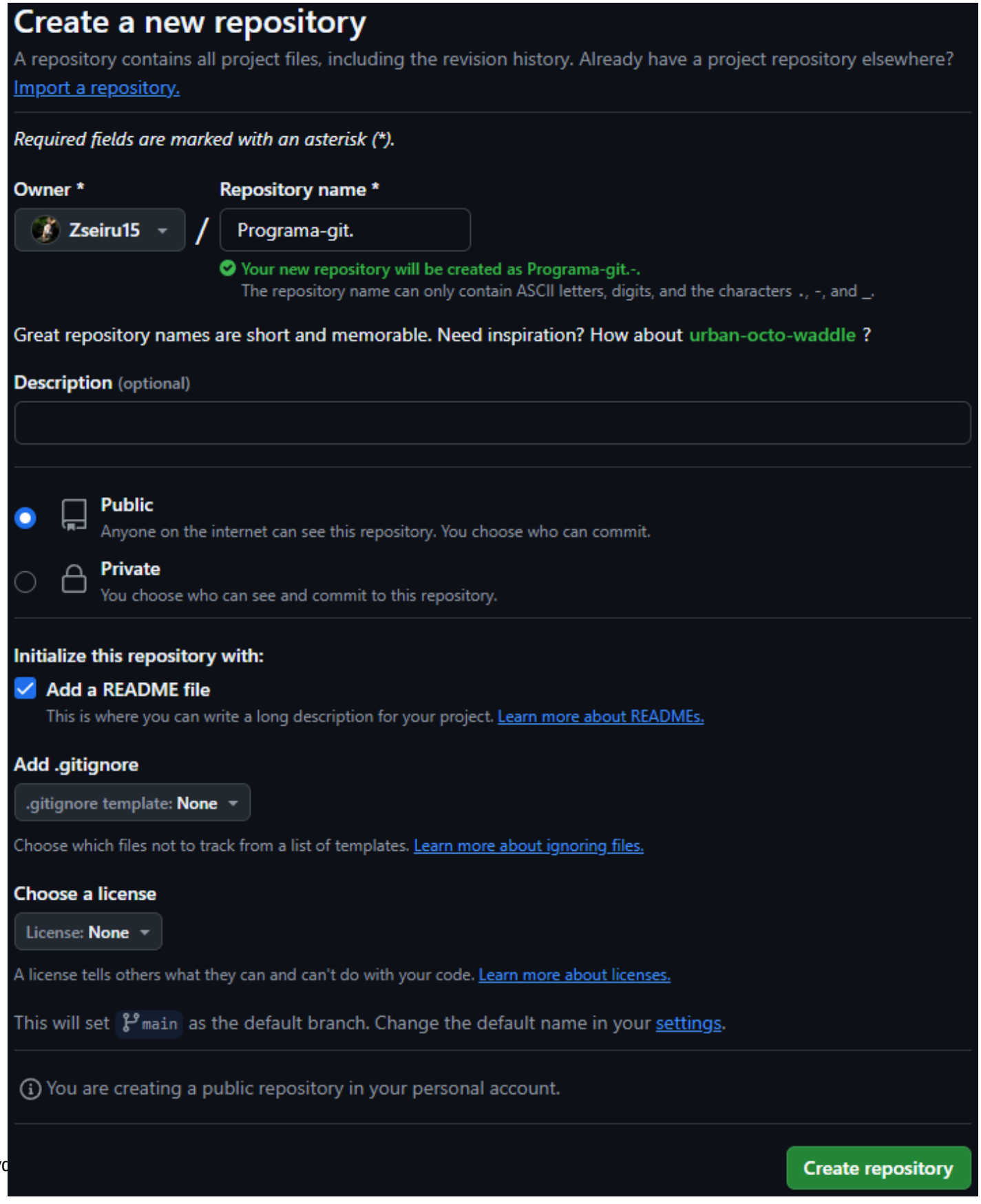
- **Evidencia de producto: GA7-220501096-AA1-EV03 herramientas de versionamiento (GIT) instalada y configurada.**

Tomando como referencia el componente formativo “Integración continua”, conectar el equipo local con el repositorio remoto por medio de Git usando los comandos vistos anteriormente.

Realice la ejecución de cada uno de los comandos básicos de git remoto y local teniendo en cuenta el ejercicio propuesto a continuación:

Paso 1

1. Crear un nuevo repositorio público en GitHub, gitLab o herramienta de su selección con el nombre Programa-git.
2. Añadirlo al repositorio local del Programa.
3. Mostrar todos los repositorios remotos configurados.


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Required fields are marked with an asterisk (*).
Owner *
Repository name *
Zseiru15
/
Programa-git.
Your new repository will be created as Programa-git.-.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.
Great repository names are short and memorable. Need inspiration? How about urban-octo-waddle ?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.
Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.
This will set main as the default branch. Change the default name in your settings.
You are creating a public repository in your personal account.
Create repository

Paso 1

```
# Crear el repositorio en GitHub y copiar su url con el protocolo https.
```

```
> git remote add github url
```

```
> git remote -v
```

```
YPLAPRSSVFSW021+Aprendiz Tarde@YPLAPRSSVFSW021 MINGW64 ~/Documents/repo_sena
$ git clone https://github.com/Zseiru15/Programa-git.git
Cloning into 'Programa-git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

Paso 2

1. Agregar los cambios del repositorio local al repositorio remoto de GitHub o GitLab.
2. Acceder a GitHub o GitLab y comprobar que se han subido los cambios mostrando el historial de versiones.

```
>git push github master
```

Paso 3

- Agregar el archivo kotlin.txt que contenga el nombre del usuario y su correo electrónico.

- Agregar los cambios.
- Realizar un commit con el mensaje “Añadido datos.”
- Cargar cambios al repositorio remoto.

```
> cat > kotlin.txt
```

```
git add .
```

```
> git commit -m
                "Añadido datos."
```

```
> git push origin master.
```

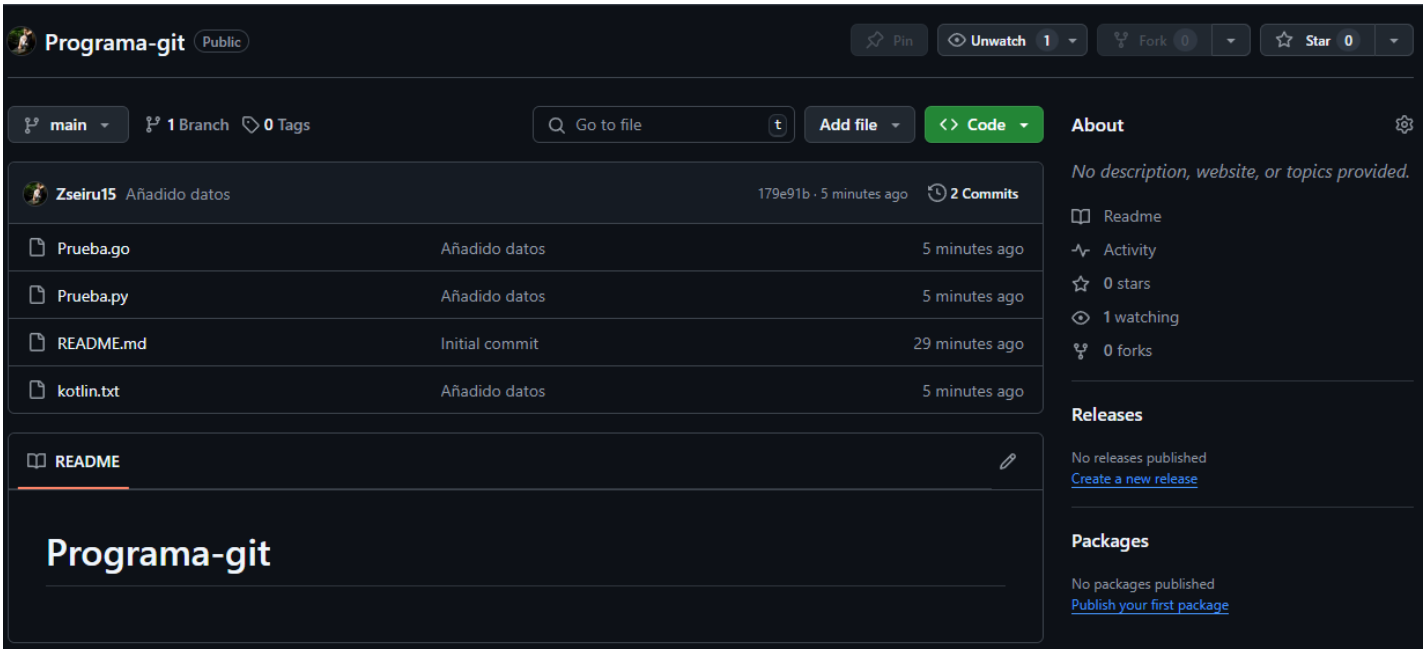
Elementos para tener en cuenta:

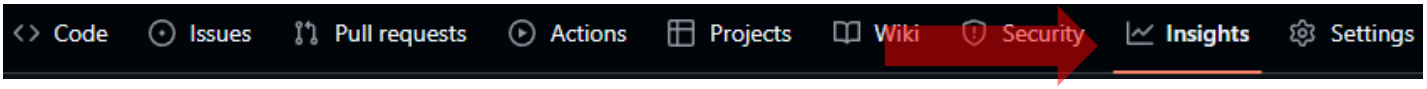
- Se debe realizar el ejercicio en las herramientas de versionamiento instaladas y tomar pantallazos ejecutando cada uno de los puntos solicitados en el ejercicio propuesto.
- Debe ir explicando cada uno de los requerimientos.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** Documento
- **Extensión:** pdf.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio asignado


```
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git> git add .
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git> git commit -m "Añadido datos"
[main 179e91b] Añadido datos
 3 files changed, 2 insertions(+)
 create mode 100644 Prueba.go
 create mode 100644 Prueba.py
 create mode 100644 kotlin.txt
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 361 bytes | 180.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Zseiru15/Programa-git.git
   87b6ccb..179e91b  main -> main
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git>
```


Programa-git
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Add file
Code
About
No description, website, or topics provided.
Zseiru15 Añadido datos
179e91b · 5 minutes ago
2 Commits
Prueba.go Añadido datos 5 minutes ago
Prueba.py Añadido datos 5 minutes ago
README.md Initial commit 29 minutes ago
kotlin.txt Añadido datos 5 minutes ago
Readme
Activity
0 stars
1 watching
0 forks
README
Releases
No releases published
Create a new release
Programa-git
Packages
No packages published
Publish your first package


Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings


Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Actions Usage Metrics
Network graph
Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.
Owners
Oct
1
Zseiru15
main

3.1.2 Actividad de aprendizaje GA7-220501096-AA2 - Aplicar estándares de codificación

Esta actividad se centra en la descripción del plan de trabajo que se debe definir según las características del software a desarrollar, detallando estándares de codificación y lenguaje a utilizar.

Duración: 24 horas

Materiales de formación a consultar: para el desarrollo de esta actividad es importante la lectura y análisis del componente formativo: “Desarrollo con lenguaje goland o python”.

Evidencias: a continuación, se describen las acciones y las correspondientes evidencias que conforma la actividad de aprendizaje**:**

- **Evidencia de desempeño: GA7-220501096-AA2-EV01 codificación de módulos del software según requerimientos del proyecto**

Ejercicio: Generar una Factura Simple en PDF

Descripción: Crea un programa en Go o python que solicite al usuario los datos necesarios para generar una factura simple en formato PDF. El programa deberá pedir al usuario que ingrese información como el nombre del cliente, la fecha, una lista de productos con sus precios y cantidades, y finalmente calcular el total.
El PDF debe incluir la siguiente información:

1. Nombre del cliente.
2. Fecha de la factura.
3. Lista de productos, precios unitarios, cantidades y el costo total por producto.
4. Total, general de la factura.

Requisitos:

1. **Sentencias de control**: Usa bucles y condiciones para gestionar la entrada de varios productos.
   If, For

```go
func compras() ([]string, []float64, []float64, float64) {    compras redeclared in this block
    var valor_total float64       // Valor total de la compra.

    for {
        var producto string
        var precio, cantidad float64

        // Captura el nombre del producto
        fmt.Println("Ingrese el nombre del producto: ")
        fmt.Scan(&producto)
        productos = append(productos, producto) // Agrega el producto a la lista de productos.

        // Captura el precio del producto
        fmt.Println("Ingrese el precio del producto: ")
        fmt.Scan(&precio)
        precios = append(precios, precio) // Agrega el precio del producto a la lista de precios.

        // Captura la cantidad de productos
        fmt.Println("Ingrese la cantidad del producto: ")
        fmt.Scan(&cantidad)
        cantidades = append(cantidades, cantidad) // Agrega la cantidad del producto a la lista de cantidades.

        // Calcula el valor total por producto (precio * cantidad)
        valor := precio * cantidad
        valor_total += valor // Suma el valor calculado al valor total.

        // Pregunta si se desea agregar otro producto
        fmt.Println("Desea agregar otro producto? (s/n): ")
        fmt.Scan(&opcion)

        // Si no se desea agregar más productos, se rompe el ciclo.
        if opcion != "s" {
            break
        }
    }
```

2. **Conversión de variables**: Convierte las cantidades y precios de string a valores numéricos.

```
// Resumen final de la factura
pdf.Ln(10)                          // Espacio antes del resumen.
pdf.SetFont("Arial", "B", 12)        // Cambia la fuente a Arial negrita, tamaño 12.
pdf.Cell(110, 7, "")                // Espacio vacío para alinear el texto a la derecha.
pdf.CellFormat(40, 7, "BASE", "0", 0, "", false, 0, "") // Texto "BASE".
pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", valor_total), "1", 0, "C", false, 0, "") // Muestra el valor total (subtotal).
pdf.Ln(10)                          // Salto de línea.
pdf.Cell(110, 7, "")                // Espacio vacío.
pdf.CellFormat(40, 7, "IVA", "0", 0, "", false, 0, "") // Texto "IVA".
iva := valor_total * 0.16           // Calcula el IVA (16% del total).
pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", iva), "1", 0, "C", false, 0, "") // Muestra el valor del IVA.
pdf.Ln(10)                          // Salto de línea.
pdf.Cell(110, 7, "")                // Espacio vacío.
pdf.CellFormat(40, 7, "TOTAL", "0", 0, "", false, 0, "") // Texto "TOTAL".
pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", valor_total+iva), "1", 0, "C", false, 0, "") // Muestra el total final (subtotal + IVA).

// Guardar el archivo PDF
err := pdf.OutputFileAndClose("PruebaGO.pdf") // Guarda el archivo PDF en el sistema con el nombre "Prueba.pdf".
if err != nil {
    fmt.Println("Error al generar el PDF:", err) // Muestra un mensaje de error si la generación falla.
}
```

3. **Impresión de datos**: Muestra por consola el resumen de la factura antes de generarla.
4. **Entrada por consola**: Permite al usuario introducir los datos necesarios.

```
Ingrese la cantidad del producto:
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git> go run Prueba.go
Ingrese el nombre del cliente:
Camilo
Ingrese el nombre del producto:
Arroz
Ingrese el precio del producto:
1000
Ingrese la cantidad del producto:
3
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Sal
Ingrese el precio del producto:
600
Ingrese la cantidad del producto:
5
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Huevos
Ingrese el precio del producto:
250
Ingrese la cantidad del producto:
4
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Salchichas
Ingrese el precio del producto:
4550
Ingrese la cantidad del producto:
2
Desea agregar otro producto? (s/n):
n
Generando PDF...
PDF generado con éxito: Factura.pdf
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git>
```

5. **Generar un PDF**: Utiliza una librería como gofpdf para generar el documento.

```go
import (
    "fmt"
    "github.com/jung-kurt/gofpdf"
    "time"
)
```

```go
// Función para generar el PDF con tablas y estructura más detallada
// Esta función genera el archivo PDF que representa la factura con los productos y el total.
func generarPDF(cliente string, productos []string, precios, cantidades []float64, valor_total float64) {    generarPDF redeclared in this block
    pdf := gofpdf.New("P", "mm", "A4", "") // Inicializa un nuevo documento PDF en tamaño A4.
    pdf.AddPage()                          // Agrega una nueva página al PDF.
```

```go
// Función para generar el PDF con tablas y estructura más detallada
// Esta función genera el archivo PDF que representa la factura con los productos y el total.
func generarPDF(cliente string, productos []string, precios, cantidades []float64, valor_total float64) {    generarPDF redeclared in t
    pdf := gofpdf.New("P", "mm", "A4", "") // Inicializa un nuevo documento PDF en tamaño A4.
    pdf.AddPage()                          // Agrega una nueva página al PDF.

    // Titulo
    pdf.SetFont("Arial", "B", 16)        // Establece la fuente Arial en negrita, tamaño 16.
    pdf.Cell(190, 10, "Factura")         // Escribe el título "Factura" en la parte superior del PDF.
    pdf.Ln(15)                           // Inserta un salto de línea.

    // Información de la factura
    pdf.SetFont("Arial", "", 12)         // Cambia la fuente a Arial normal, tamaño 12.
    pdf.Cell(100, 10, fmt.Sprintf("Cliente: %s", cliente)) // Escribe el nombre del cliente.
    pdf.Ln(5)                            // Inserta un salto de línea.
    pdf.Cell(100, 10, fmt.Sprintf("Fecha: %s", time.Now().Format("02/01/2006 - 15:04:05"))) // Escribe la fecha actual.
    pdf.Ln(15)                           // Inserta otro salto de línea.

    // Cabecera de la tabla
    pdf.SetFont("Arial", "B", 12)        // Cambia la fuente a Arial negrita, tamaño 12.
    // Añade las columnas de la tabla: Ref, Descripción, Cantidad, Precio, Importe.
    pdf.CellFormat(10, 7, "Ref", "1", 0, "C", false, 0, "")
    pdf.CellFormat(80, 7, "Descripcion", "1", 0, "C", false, 0, "")
    pdf.CellFormat(20, 7, "Cantidad", "1", 0, "C", false, 0, "")
    pdf.CellFormat(30, 7, "Precio", "1", 0, "C", false, 0, "")
    pdf.CellFormat(30, 7, "Importe", "1", 0, "C", false, 0, "")
    pdf.Ln(-1)                           // Salto de línea para iniciar el contenido de la tabla.

    // Datos de la tabla
    pdf.SetFont("Arial", "", 12)         // Cambia la fuente a Arial normal, tamaño 12.
    for i, producto := range productos { // Itera sobre los productos para llenar la tabla.
        // Cada producto ocupa una fila en la tabla.
        pdf.CellFormat(10, 7, fmt.Sprintf("%d", i+1), "1", 0, "C", false, 0, "") // Número de referencia.
        pdf.CellFormat(80, 7, producto, "1", 0, "", false, 0, "") // Nombre del producto.
        pdf.CellFormat(20, 7, fmt.Sprintf("%.2f", cantidades[i]), "1", 0, "C", false, 0, "") // Cantidad del producto.
        pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", precios[i]), "1", 0, "C", false, 0, "") // Precio del producto.
        pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", precios[i]*cantidades[i]), "1", 0, "C", false, 0, "") // Importe total.
        pdf.Ln(-1)                       // Salto de línea después de cada producto.
    }

    // Resumen final de la factura
    pdf.Ln(10)                           // Espacio antes del resumen.
    pdf.SetFont("Arial", "B", 12)        // Cambia la fuente a Arial negrita, tamaño 12.
    pdf.Cell(110, 7, "")                 // Espacio vacío para alinear el texto a la derecha.
    pdf.CellFormat(40, 7, "BASE", "0", 0, "", false, 0, "") // Texto "BASE".
    pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", valor_total), "1", 0, "C", false, 0, "") // Muestra el valor total (subtotal).
    pdf.Ln(10)                           // Salto de línea.
    pdf.Cell(110, 7, "")                 // Espacio vacío.
    pdf.CellFormat(40, 7, "IVA", "0", 0, "", false, 0, "") // Texto "IVA".
    iva := valor_total * 0.16            // Calcula el IVA (16% del total).
    pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", iva), "1", 0, "C", false, 0, "") // Muestra el valor del IVA.
    pdf.Ln(10)                           // Salto de línea.
    pdf.Cell(110, 7, "")                 // Espacio vacío.
    pdf.CellFormat(40, 7, "TOTAL", "0", 0, "", false, 0, "") // Texto "TOTAL".
    pdf.CellFormat(30, 7, fmt.Sprintf("%.2f", valor_total+iva), "1", 0, "C", false, 0, "") // Muestra el total final (subtotal + IVA).

    // Guardar el archivo PDF
    err := pdf.OutputFileAndClose("PruebaGO.pdf") // Guarda el archivo PDF en el sistema con el nombre "Prueba.pdf".
    if err != nil {
        fmt.Println("Error al generar el PDF:", err) // Muestra un mensaje de error si la generación falla.
    }
}

// Función principal
// Esta función coordina la generación de la factura y la creación del PDF.
func main() {    main redeclared in this block
    // Llama a la función Factura() para capturar los datos y generar la factura.
    cliente, productos, precios, cantidades, valor_total := Factura()    assignment mismatch: 5 variables but Factura returns 4 values

    // Genera el PDF con los datos obtenidos.
    fmt.Println("Generando PDF...")
    generarPDF(cliente, productos, precios, cantidades, valor_total) // Llama a la función para generar el PDF.    too many arguments i
    fmt.Println("PDF generado con éxito: Factura.pdf") // Mensaje de confirmación de éxito.
}
```

```go
// Función principal
// Esta función coordina la generación de la factura y la creación del PDF.
func main() {    main redeclared in this block
    // Llama a la función Factura() para capturar los datos y generar la factura.
    cliente, productos, precios, cantidades, valor_total := Factura()    assignment mismatch: 5 variables but Fa

    // Genera el PDF con los datos obtenidos.
    fmt.Println("Generando PDF...")
    generarPDF(cliente, productos, precios, cantidades, valor_total) // Llama a la función para generar el PDF.
    fmt.Println("PDF generado con éxito: Factura.pdf") // Mensaje de confirmación de éxito.
}
```

Instrucciones:

1. **Pedir datos al usuario**:
    - Nombre del cliente.

```go
// Función para capturar el nombre del cliente
// Esta función solicita al usuario que ingrese el nombre del cliente y lo retorna.
func datos_cliente() string {    datos_cliente redeclared in this block
    var cliente string
    fmt.Println("Ingrese el nombre del cliente: ") // Muestra un mensaje solicitando el nombre del cliente.
    fmt.Scan(&cliente)                             // Captura el nombre ingresado por el usuario.
    return cliente                                 // Retorna el nombre del cliente.
}
```

    - Fecha de la factura.

```go
// Información de la factura
pdf.SetFont("Arial", "", 12)         // Cambia la fuente a Arial normal, tamaño 12.
pdf.Cell(100, 10, fmt.Sprintf("Cliente: %s", cliente)) // Escribe el nombre del cliente.
pdf.Ln(5)                            // Inserta un salto de línea.
pdf.Cell(100, 10, fmt.Sprintf("Fecha: %s", time.Now().Format("02/01/2006 - 15:04:05"))) // Escribe la fecha actual.
pdf.Ln(15)                           // Inserta otro salto de línea.
```

    - Solicitar productos (nombre, cantidad, precio unitario).

```go
// Función que captura productos, precios y calcula el valor total de la compra
// Esta función permite al usuario ingresar varios productos, precios y cantidades. Calcula el total de la compra.
func compras() ([]string, []float64, []float64, float64) {    compras redeclared in this block
    var opcion string
    var productos []string        // Lista de nombres de productos.
    var precios []float64         // Lista de precios de productos.
    var cantidades []float64      // Lista de cantidades de productos.
    var valor_total float64       // Valor total de la compra.

    for {
        var producto string
        var precio, cantidad float64

        // Captura el nombre del producto
        fmt.Println("Ingrese el nombre del producto: ")
        fmt.Scan(&producto)
        productos = append(productos, producto) // Agrega el producto a la lista de productos.

        // Captura el precio del producto
        fmt.Println("Ingrese el precio del producto: ")
        fmt.Scan(&precio)
        precios = append(precios, precio) // Agrega el precio del producto a la lista de precios.

        // Captura la cantidad de productos
        fmt.Println("Ingrese la cantidad del producto: ")
        fmt.Scan(&cantidad)
        cantidades = append(cantidades, cantidad) // Agrega la cantidad del producto a la lista de cantidades.

        // Calcula el valor total por producto (precio * cantidad)
        valor := precio * cantidad
        valor_total += valor // Suma el valor calculado al valor total.

        // Pregunta si se desea agregar otro producto
        fmt.Println("Desea agregar otro producto? (s/n): ")
        fmt.Scan(&opcion)

        // Si no se desea agregar más productos, se rompe el ciclo.
        if opcion != "s" {
            break
        }
    }
    // Retorna los productos, precios, cantidades y el valor total de la compra.
    return productos, precios, cantidades, valor_total
}
```

- Permitir al usuario agregar múltiples productos hasta que indique que ha terminado.

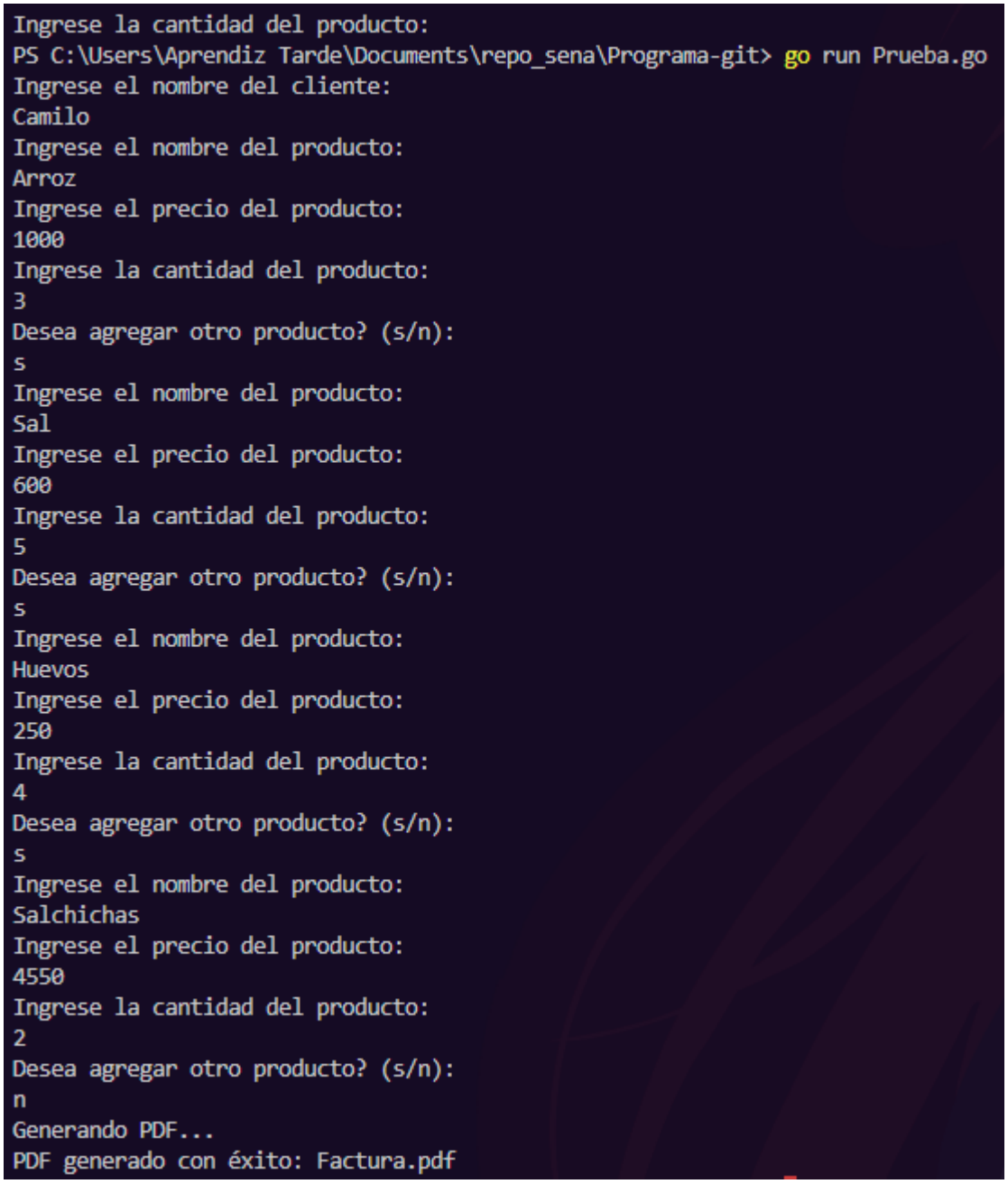
```go
// Pregunta si se desea agregar otro producto
fmt.Println("Desea agregar otro producto? (s/n): ")
fmt.Scan(&opcion)

// Si no se desea agregar más productos, se rompe el ciclo.
if opcion != "s" {
    break
}
```

2. **Calcular totales**:
    - Por cada producto, calcular el total (cantidad * precio unitario).
    - Calcular el total general de todos los productos.

```go
// Calcula el valor total por producto (precio * cantidad)
valor := precio * cantidad
valor_total += valor // Suma el valor calculado al valor total.
```

3. **Imprimir por consola** el resumen de la factura antes de generarla.
4. **Generar el PDF** con los datos ingresados.



Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** Repositorio que debe tener los siguientes archivos:
- archivos del proyecto,
- archivo con enlace del repositorio, la carpeta comprimida debe tener el nombre del aprendiz y número de la evidencia así: NOMBREAPELLIDO_AA2_EV01
- **Extensión:** PDF
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio destinado.

- **Evidencia de producto: GA7-220501096-AA2-EV02 módulos de software codificados y probados**

Teniendo en cuenta la codificación del módulo del proyecto realizada en la evidencia AA2-EV01 realizar las pruebas según requerimientos de las historias de usuario o casos de uso**.**

```
Ingrese la cantidad del producto:
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git> go run Prueba.go
Ingrese el nombre del cliente:
Camilo
Ingrese el nombre del producto:
Arroz
Ingrese el precio del producto:
1000
Ingrese la cantidad del producto:
3
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Sal
Ingrese el precio del producto:
600
Ingrese la cantidad del producto:
5
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Huevos
Ingrese el precio del producto:
250
Ingrese la cantidad del producto:
4
Desea agregar otro producto? (s/n):
s
Ingrese el nombre del producto:
Salchichas
Ingrese el precio del producto:
4550
Ingrese la cantidad del producto:
2
Desea agregar otro producto? (s/n):
n
Generando PDF...
PDF generado con éxito: Factura.pdf
PS C:\Users\Aprendiz Tarde\Documents\repo_sena\Programa-git>
```

Factura

Cliente: Camilo
Fecha: 15/10/2024 - 13:17:07

Ref	Descripcion	Cantidad	Precio	Importe
1	Arroz	3.00	1000.00	3000.00
2	Sal	5.00	600.00	3000.00
3	Huevos	4.00	250.00	1000.00
4	Salchichas	2.00	4550.00	9100.00

BASE	**16100.00**
IVA	**2576.00**
TOTAL	**18676.00**

Elementos para tener en cuenta:

- Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, se requiere tomar un pantallazo de la aplicación por cada uno de los requisitos levantados en las historias de usuario o casos de uso, ósea que debe describirse la historia de usuario y/o casos de uso y debajo debe ir el pantallazo de la interfaz de la aplicación, se deben documentar también las pruebas de validaciones de la aplicación (fechas, números, textos, caracteres especiales, longitudes etc)
- Debe presentar un documento mostrando toda la funcionalidad del módulo codificado con sus respectivas validaciones.
- Debe trabajar con herramientas de versionamiento.

Lineamientos generales para la entrega de la evidencia:

- **Productos para entregar:** carpeta comprimida que debe tener los siguientes archivos: archivos del proyecto, documento en Word
- Archivo con enlace del repositorio, la carpeta comprimida debe tener el nombre del aprendiz y número de la evidencia así: NOMBRE_APELLIDO_AA2_EV02
- **Extensión:** ZIP, RAR.
- Para hacer el envío de la evidencia remítase al área de la actividad correspondiente y acceda al espacio destinado.

4. ACTIVIDADES DE EVALUACIÓN

Evidencias de aprendizaje	Criterios de evaluación	Técnicas e instrumentos de evaluación
Evidencias de conocimiento: Informe técnico de plan de trabajo para construcción de software. **GA7-220501096-AA1-EV01** Evidencias de desempeño: Definir estándares de codificación de acuerdo a plataforma de desarrollo elegida. **GA7-220501096-AA1-EV02**	Interpreta el informe de diseño para definir el plan de trabajo en la construcción del software. Selecciona y configura herramientas de desarrollo de acuerdo con las condiciones del software a construir. Define estándares de codificación de acuerdo con las reglas de la plataforma de desarrollo seleccionada.	**IE-GA7-220501096-AA1-EV01** Lista de chequeo **IE-GA7-220501096-AA1-EV02** Lista de chequeo
Evidencia de conocimiento: Identifica herramientas de versionamiento. **GA7-220501096-AA1-EV03**		**IE-GA7-220501096-AA1-EV03** Lista de chequeo

Evidencias de desempeño: Instalación y configuración de herramienta de versionamiento (Local / Web). **GA7-220501096-AA1-EV04** Evidencia de producto: Herramientas de versionamiento (GIT) instalada y configurada. **GA7-220501096-AA1-EV05**	Selecciona y configura herramientas de versionamiento para el control de cambios en el código.	
Evidencias de desempeño: Codificación de módulos del software según requerimientos del proyecto. **GA7-220501096-AA2-EV01** Evidencia de producto: Módulos de software codificados y probados. **GA7-220501096-AA2-EV02**	Codifica los módulos del software Stand alone, web y móvil, de acuerdo con las especificaciones del diseño y el estándar de codificación.	**IE-GA7-220501096-AA2-EV01** Lista de chequeo
Evidencias de desempeño: Codificación de módulos del software Stand alone. **GA7220501096-AA3-EV01** Evidencia de producto: Módulos de software codificados y probados. **GA7-220501096-AA3-EV02**		**IE-GA7-220501096-AA2-EV02** Lista de chequeo

5. GLOSARIO

Software: soporte lógico, programas, parte no mecánica de un sistema. Serie de instrucciones necesarias para ejecutar diversas aplicaciones y tareas.

WAR: archivo de aplicación web utilizado para empaquetar las aplicaciones web hechas con tecnología JAVA. La Sun define la estructura de un WAR.

Servlet: es una clase hecha en Java utilizada para extender las capacidades de los servidores que alojan aplicaciones que siguen el modelo petición-respuesta.

RDBMS: administrador de Bases de Datos Relacionales, se conocen así a los programas que permiten la gestión completa de bases de datos y su mantenimiento.

6. REFERENTES BIBLIOGRÁFICOS

Java en Castellano. (2021, Dic 01). Servlets y JSP. https://programacion.net/articulo/servlets_y_jsp_82

Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Quintas, A. F. (2000). *Java 2: manual de usuario y tutorial*. Grupo Editorial RA-MA.

Sun. (2021, Dic 01). Java. http://java.sun.com/

Vall Mainou, A. (2010). Desarrollo para internet con tecnología JAVA.

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor(es)	Jonathan Guerrero Astaiza	Experto temático	Centro de Teleinformática y Producción Industrial	Diciembre 2021
	Zulema Leon Escobar	Experta temático	Centro de Teleinformática y Producción Industrial	Diciembre 2021
	Deivis Eduard Ramírez Martínez	Diseñador Instruccional	Regional Distrito Capital - Centro para la Industria de la Comunicación Gráfica	Diciembre 2021
	Silvia Milena Sequeda Cárdenas	Metodológa	Regional Distrito Capital - Centro de diseño y metrología	Febrero de 2022
	Rafael Neftalí Lizcano Reyes	Asesor Pedagógico	Regional Santander - Centro Industrial del Diseño y la Manufactura	Febrero de 2022
	Sandra Patricia Hoyos Sepúlveda	Corrección estilo de	Centro de Diseño y Metrología - Distrito capital	Febrero 2021
	Fabian David Barreto Sanchez	Ajuste de actividades	Regional Casanare	Septiembre del 2024

7. CONTROL DE CAMBIOS

	Nombre	Cargo	Dependencia	Fecha	Razón del cambio
Autor (es)					