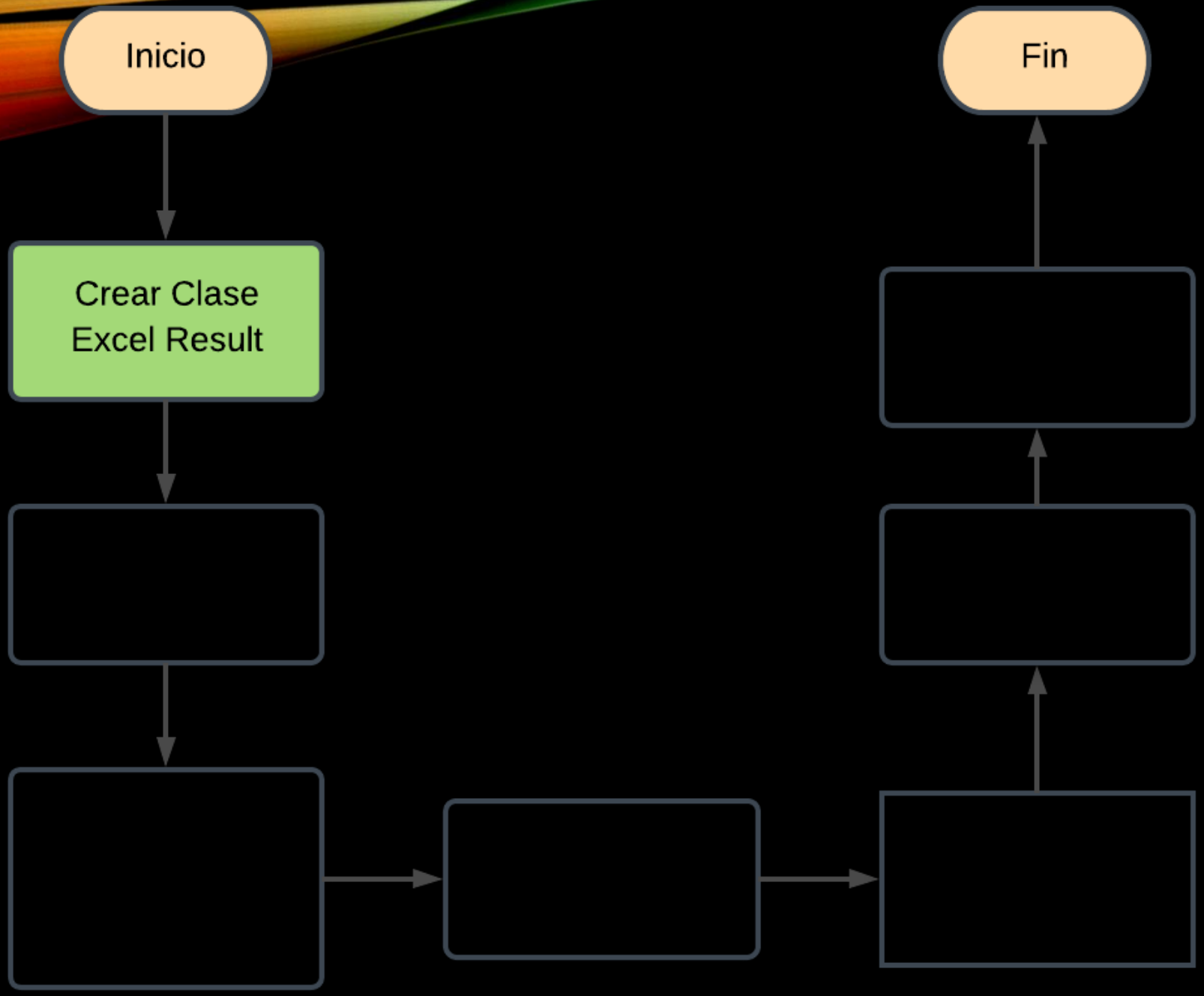
CLOSEDXML

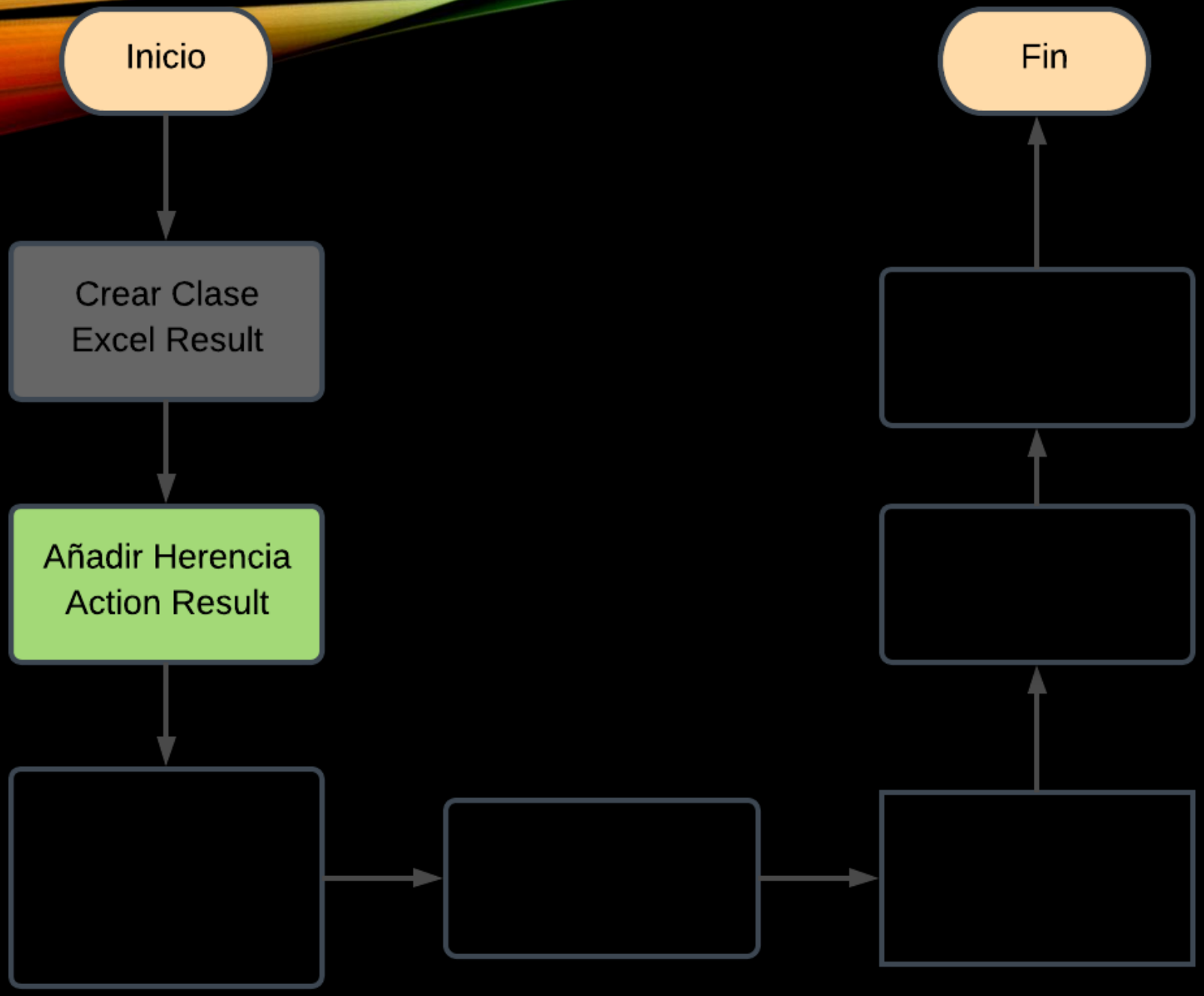
2.- Descargar archivo


Inicio
Crear Clase
Excel Result
Fin

1.- CREACIÓN CLASE RESULT

```
1   using ClosedXML.Excel;
2   using System;
3   using System.Collections.Generic;
4   using System.IO;
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult
12      {
13
14
15
16      }
17  }
```

- Creamos nuestra clase la cual nos ayudara a regresar el tipo de dato Excel para descargarlo.


Inicio
Crear Clase
Excel Result
Añadir Herencia
Action Result
Fin

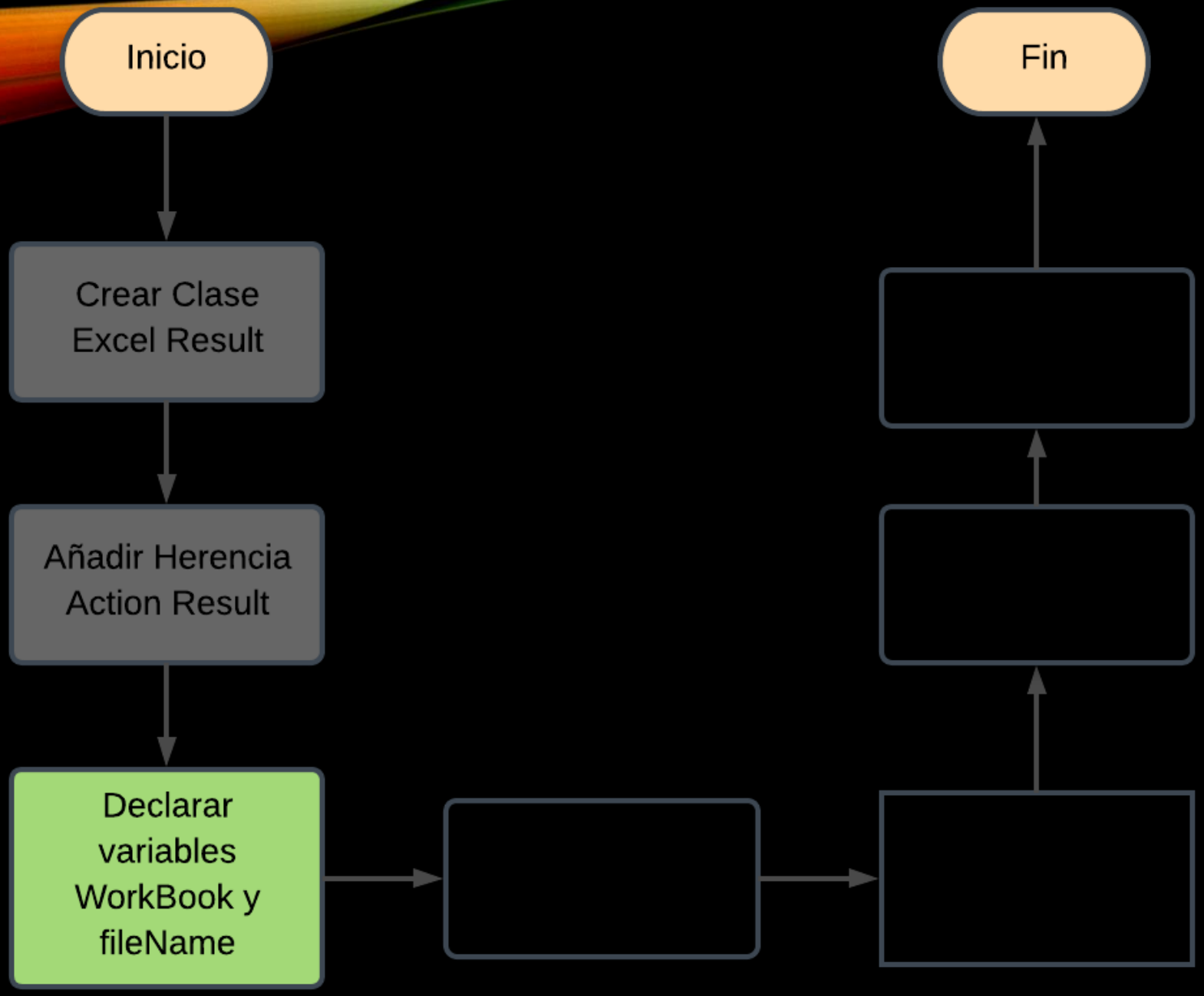
2.- AÑADIR HERENCIA

ExcelResult.cs | HomeController.cs

ExcelTest | ExcelTest.Models.ExcelResult

```
1   using ClosedXML.Excel;
2   using System;
3   using System.Collections.Generic;
4   using System.IO;
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13
14
15
16      }
17  }
```

- Añadir herencia action result.


Inicio
Crear Clase Excel Result
Añadir Herencia Action Result
Declarar variables WorkBook y fileName
Fin

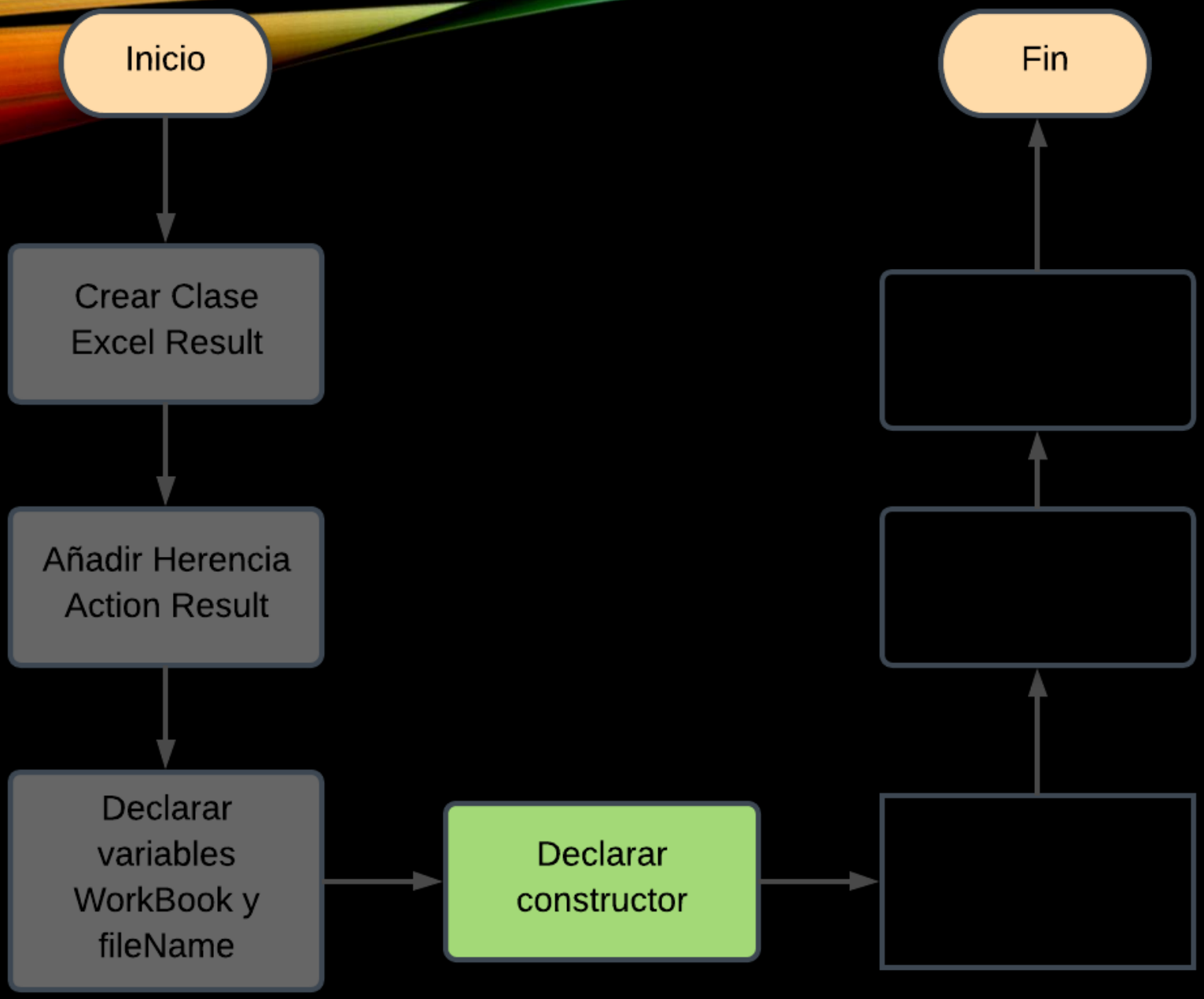
3.- DECLARAR VARIABLES

ExcelResult.cs* | HomeController.cs | ActionResult [de metadatos]

ExcelTest | ExcelTest.Models.ExcelResult | _workbook

```
1   using ClosedXML.Excel;
2   using System;
3   using System.Collections.Generic;
4   using System.IO;
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13          private readonly XLWorkbook _workbook;
14          private readonly string _fileName;
15
16
17      }
18  }
19
20
```

- Declaramos las 2 variables que necesitamos el Workbook y el nombre del archivo.


Inicio
Crear Clase Excel Result
Añadir Herencia Action Result
Declarar variables WorkBook y fileName
Declarar constructor
Fin

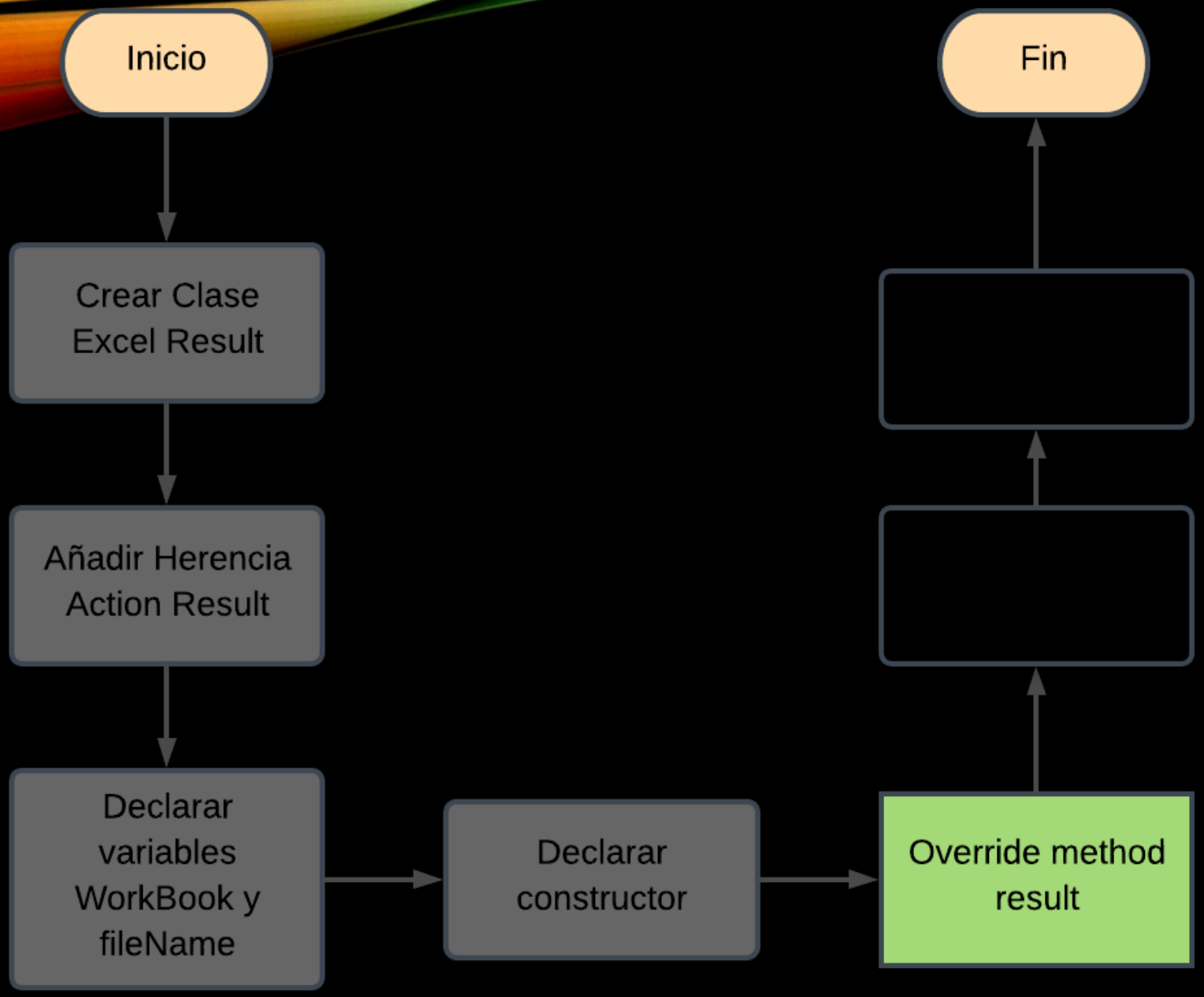
4.- DECLARAR CONSTRUCTOR

ExcelResult.cs | HomeController.cs | ActionResult [de metadatos]

ExcelTest | ExcelTest.Models.ExcelResult | ExcelResult(XLWorkbook workb

```
1   using ClosedXML.Excel;
2   using System;
3   using System.Collections.Generic;
4   using System.IO;
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13          private readonly XLWorkbook _workbook;
14          private readonly string _fileName;
15
16          public ExcelResult(XLWorkbook workbook, string fileName)
17          {
18              _workbook = workbook;
19              _fileName = fileName;
20          }
21
22
23      }
24  }
25
26
```

Declaramos el constructor para asignar nuestros valores de entrada a nuestros atributos.


Inicio
Crear Clase Excel Result
Añadir Herencia Action Result
Declarar variables WorkBook y fileName
Declarar constructor
Override method result
Fin

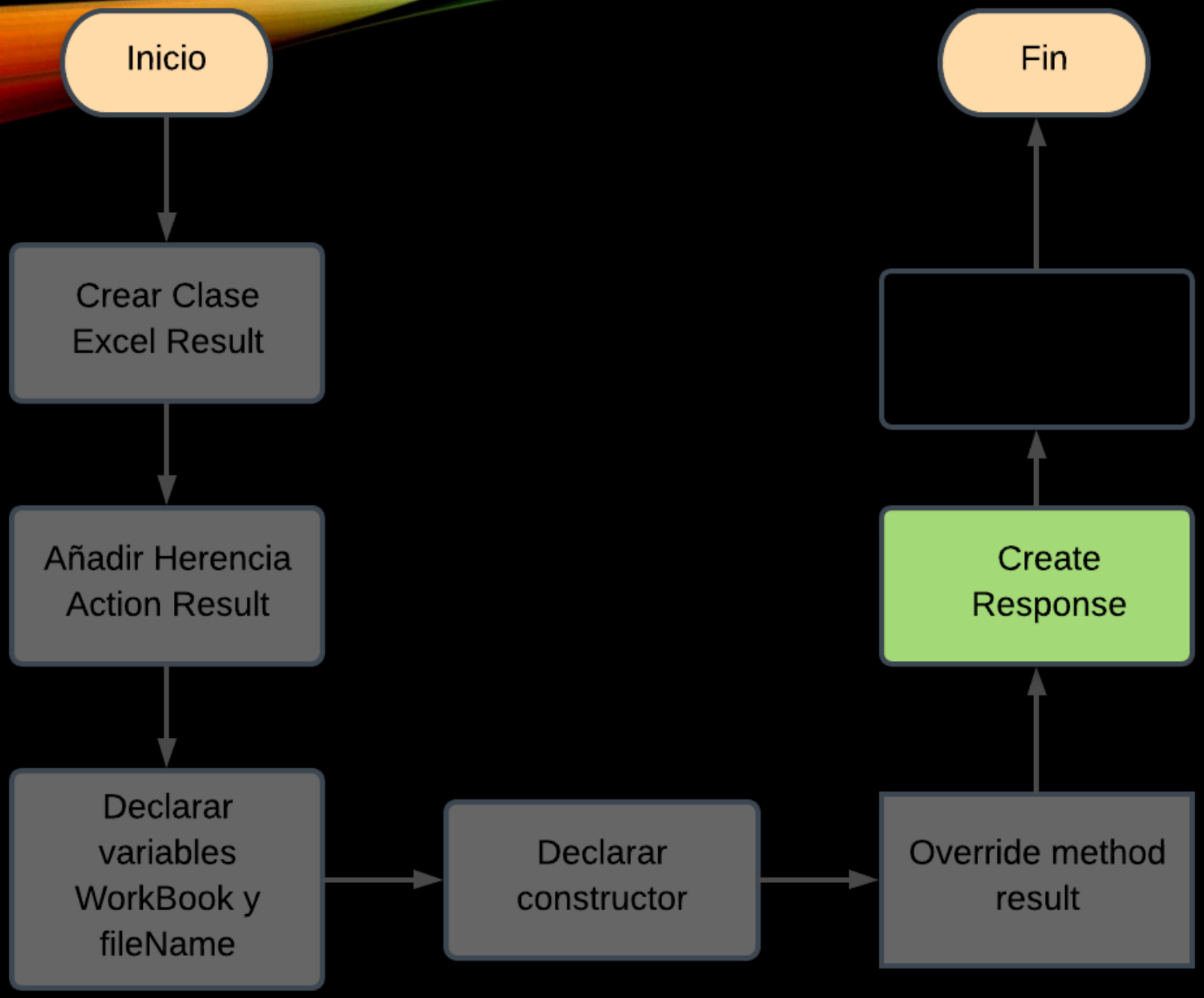
5.-OVERRIDE METHOD RESULT

ExcelResult.cs | HomeController.cs | ActionResult [de metadatos]

ExcelTest | ExcelTest.Models.ExcelResult | ExecuteResult(ControllerContext

```
1   using ClosedXML.Excel;
2   using System;
3   using System.Collections.Generic;
4   using System.IO;
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13          Atributos
17          public ExcelResult(XLWorkbook workbook, string fileName)...
22          public override void ExecuteResult(ControllerContext context)
23          {
24
25          }
26
27      }
28  }
29
30
```

- Es el método que necesita todo ActionResult para retornar sin el nos va a tirar error, como estamos haciendo un ActionResult nuevo debemos sobre escribirlo.


Inicio
Crear Clase Excel Result
Añadir Herencia Action Result
Declarar variables WorkBook y fileName
Declarar constructor
Override method result
Create Response
Fin

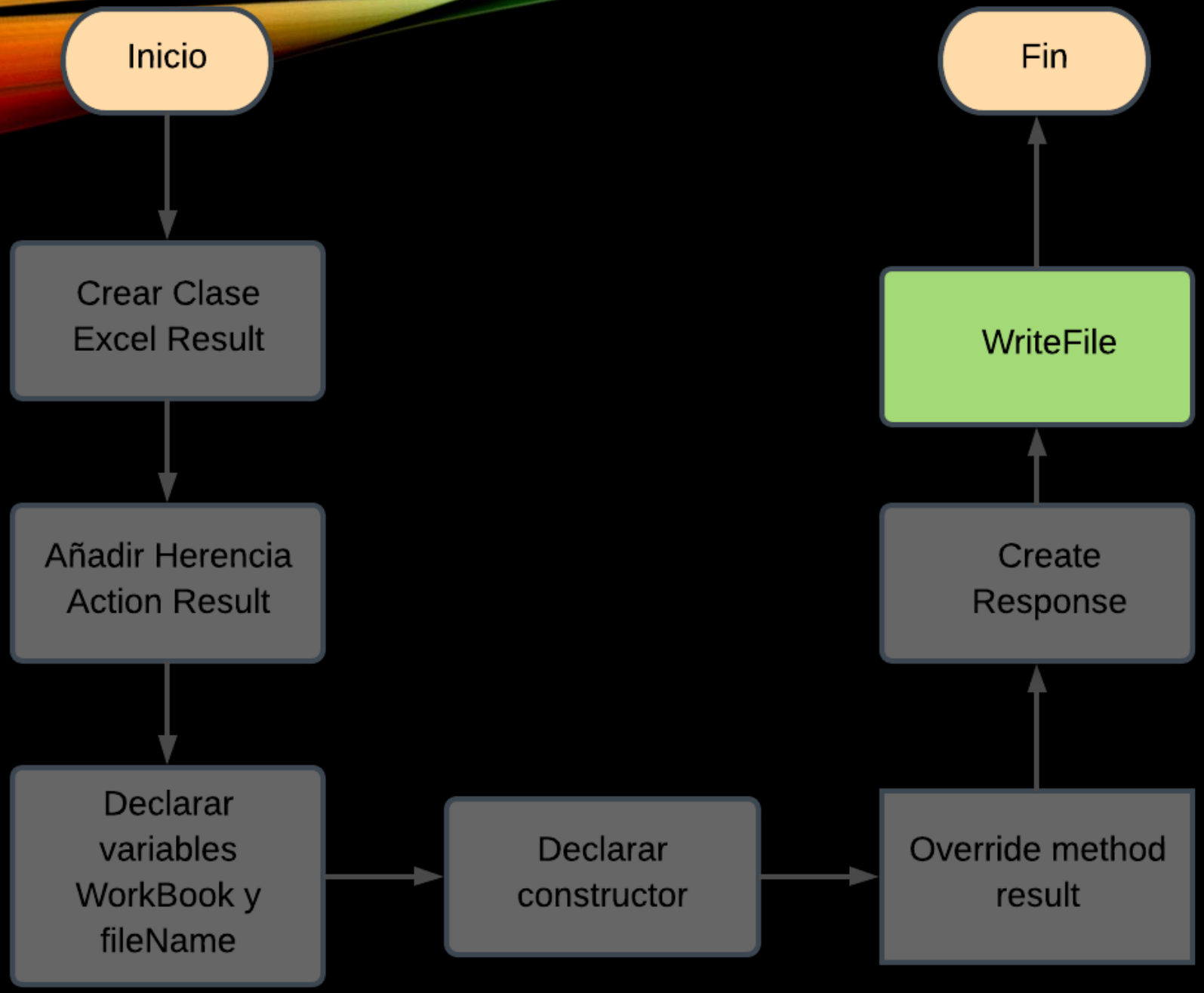
6.- CREATE RESPONSE

ExcelResult.cs | HomeController.cs

ExcelTest | ExcelTest.Models.ExcelResult

```
5   using System.Linq;
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13          Atributos
17          public ExcelResult(XLWorkbook workbook, string fileName)...
22          public override void ExecuteResult(ControllerContext context)
23          {
24              var response = context.HttpContext.Response;
25              response.Clear();
26              response.ContentType = "application/vnd.openxmlformats-officedocument."
27                                   + "spreadsheetml.sheet";
28              response.AddHeader("content-disposition",
29                                 "attachment;filename=\"" + _fileName + ".xlsx\"");
30
31
32              response.End();
33          }
34
35      }
36  }
37
```

- El método de comunicación con el browser del cliente son los HTTP responses usualmente regresan texto plano, pero como esta vez regresaremos un archivo Excel tenemos que escribir en el response que lo que estamos regresando es un archivo y no un texto.
- Así como decirle como se llama el archivo.


Inicio
Crear Clase Excel Result
Añadir Herencia Action Result
Declarar variables WorkBook y fileName
Declarar constructor
Override method result
Create Response
WriteFile
Fin

7.- ESCRIBIR ARCHIVO

ExcelResult.cs | HomeController.cs

ExcelTest | ExcelTest.Models.ExcelResult

```
6   using System.Web;
7   using System.Web.Mvc;
8
9   namespace ExcelTest.Models
10  {
11      public class ExcelResult : ActionResult
12      {
13          Atributos
17          public ExcelResult(XLWorkbook workbook, string fileName)...
22          public override void ExecuteResult(ControllerContext context)
23          {
24              var response = context.HttpContext.Response;
25              response.Clear();
26              response.ContentType = "application/vnd.openxmlformats-officedocument."
27                                     + "spreadsheetml.sheet";
28              response.AddHeader("content-disposition",
29                                 "attachment;filename=\"" + _fileName + ".xlsx\"");
30
31              using (var memoryStream = new MemoryStream())
32              {
33                  _workbook.SaveAs(memoryStream);
34                  memoryStream.WriteTo(response.OutputStream);
35              }
36
37              response.End();
38          }
39
40      }
```

- Con memoryStream creados y escribimos el archivo

```
public ActionResult About()
{
    using (var wb = new XLWorkbook())
    {
        var worksheet = wb.Worksheets.Add("Sheet 1");
        worksheet.Cell(1, 1).Value = "Hello, world!";

        return new ExcelResult(wb, "NombreDeArchivo");
    }
}
```