

ICC343 -Tutorial 10 - 2021-2

Basic Elements in a Web C# Project

Ph.D. Carlos Cares

The goal of this tutorial is to advance in the implementation of the [*“game of life”*](#). This is a 2D matrix of spaces which can contain an alive cell or a dead cell. We will make some changes to an existing web application to reach this project milestone. Let's advance.

Work 1. (15%). Creation of a basic blazorserver project

This has previous requirements (your assignment):

- You must previously have installed .NET Core (please install 3.1 version)
- You must previously have installed Visual Studio Code (I have 1.49 version)

W1-Step 1. In the terminal execute the command:

```
dotnet new blazorserver -o blazorserver01
```

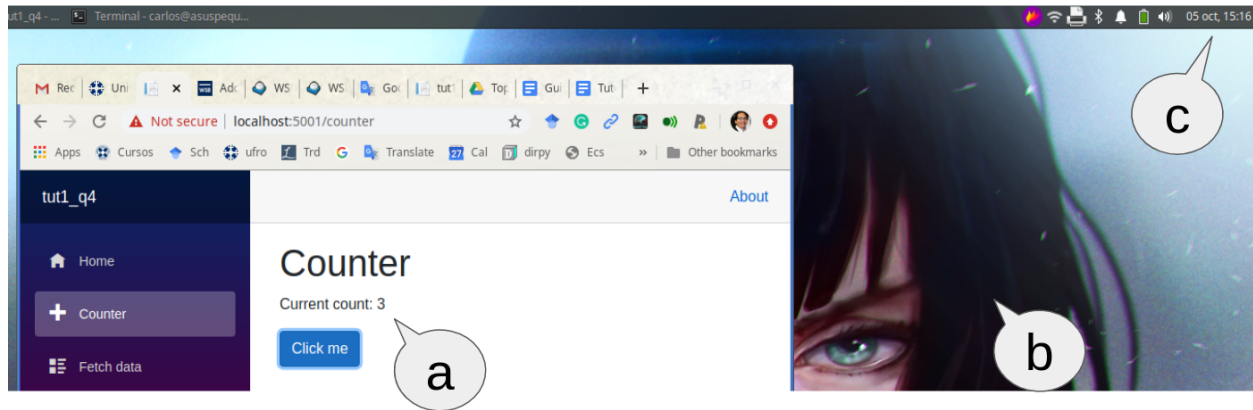
The last expression “blazorserver01” is the name you choose for your project, thus, you can change it. This expression creates a project having a menu and a web page in each option.

W1-Step 2. Execute the project

The easy way of executing the project is, in the same terminal, INSIDE the directory of the project, you must execute:

```
dotnet watch run
```

W1- Step 3. Get your first photo (screenshot) of the successful creation. In the Counter option, click the button “Click me” three times, and get a photo. Your photo MUST accomplish three requirements: (a) It should show the requested screen with the counter in 3, (b) it should show a part of your desktop, and (c) It should show the time and date of your operating system. The next screenshot is an example of which you must upload as W1-Answer.



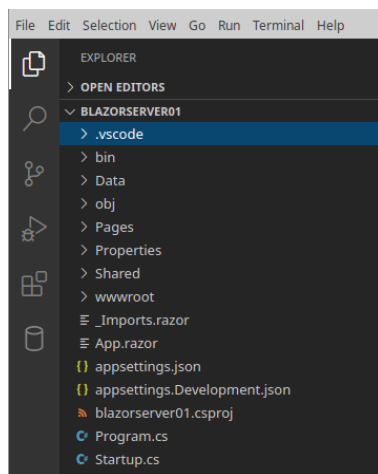
W1- Step 4. Upload your screenshots as answers of Work 1 using the following labels:

Tut10-W1-A. The output in the terminal windows after running the project

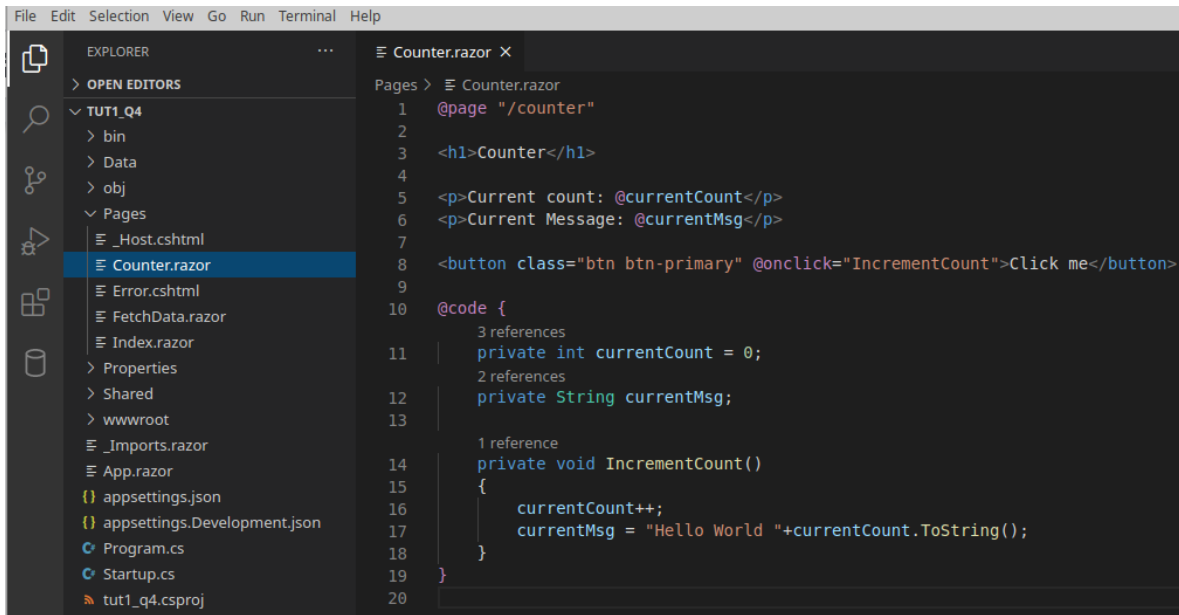
Tut10-W1-B. The Counter.razor page after three clicks on the button “Click me”.

Work 2. (15%). Adding new variables to the View

W2-Step 1. Open the created project, this is the option “File->Open Folder”. You should see something like this when you activate the first icon from the left panel:



W2-Step 2. Modifying the view Counter.razor. Open the file Counter.razor. Files having “razor” extensions are html files with embedding variables and code from C# programs. The lines to add are shown in the **next figure**. The changes are in lines 6, 12, and 17.



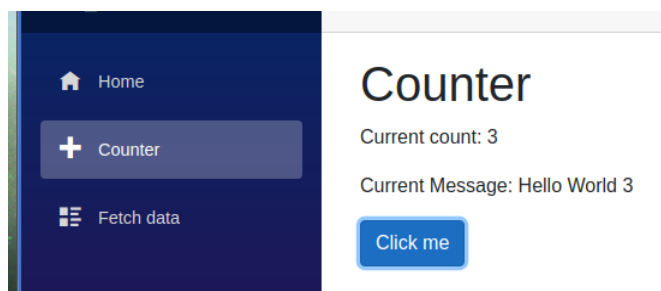
```
File Edit Selection View Go Run Terminal Help
EXPLORER
> OPEN EDITORS
TUT1_Q4
> bin
> Data
> obj
> Pages
  Counter.razor
  Error.cshtml
  FetchData.razor
  Index.razor
> Properties
> Shared
> wwwroot
  _Imports.razor
  App.razor
  appsettings.json
  appsettings.Development.json
  Program.cs
  Startup.cs
  tut1_q4.csproj
Counter.razor X
Pages > Counter.razor
1 @page "/counter"
2
3 <h1>Counter</h1>
4
5 <p>Current count: @currentCount</p>
6 <p>Current Message: @currentMsg</p>
7
8 <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
9
10 @code {
11     3 references
12     private int currentCount = 0;
13     2 references
14     private String currentMsg;
15
16     1 reference
17     private void IncrementCount()
18     {
19         currentCount++;
20         currentMsg = "Hello World "+currentCount.ToString();
21     }
22 }
```

Line6: Add the display of the `currentMsg` variable. The editor will advertise some warning or error in the meanwhile until you define the variable after that. Just add it. “@” means that C# starts, if you choose a variable, method, or expression, the final value is displayed.

Line 12: Add the definition of the variable `currentMsg`

Line 17: Add this line. The variable `currentMsg` will have the value of the String “Hello World” concatenated to the current value of the `currentCount` variable. The operator “+” does not work between a String and an int, therefore, it is needed to transform the int to String. The `toString` method does this.

W2-Step 3. Test the changes running the project again. You should have the following output after three clicks on the button “Click me”.



W2- Step 4. Change the word **World by Your Name**, and get your screenshot of the successful changes of this stage. This screenshot should follow the same requirements as the previous

screenshot, i.e., it should show the expected figure (with your name), the date and hour, and part of the desktop.

W2- Step 5. Upload your screenshots as answers to Work 2 using the following labels:

Tut10-W2-A. Screenshot of the new Counter.razor page

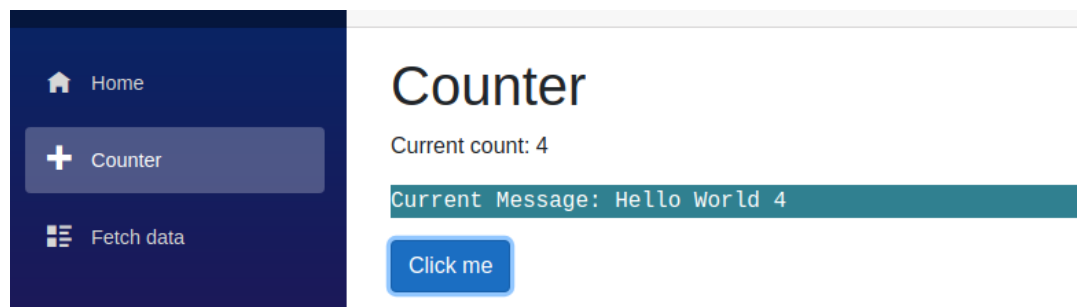
Tut10-W2-B. Output in the browser of the Counter page.

Work 3. (15%). Adding styles to the View

W3-Step 1. Due to “the View” being an HTML specification then we can use the style in the tags that accept them. For example, in the file Counter.razor we can change the output of the message as is specified in the next screenshot (Line 6) -try it-:

```
5 <p>Current count: @currentCount</p>
6 <p style="background-color:#308090;color:white;font-family:courier">
7   Current Message: @currentMsg</p>
8
9 <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

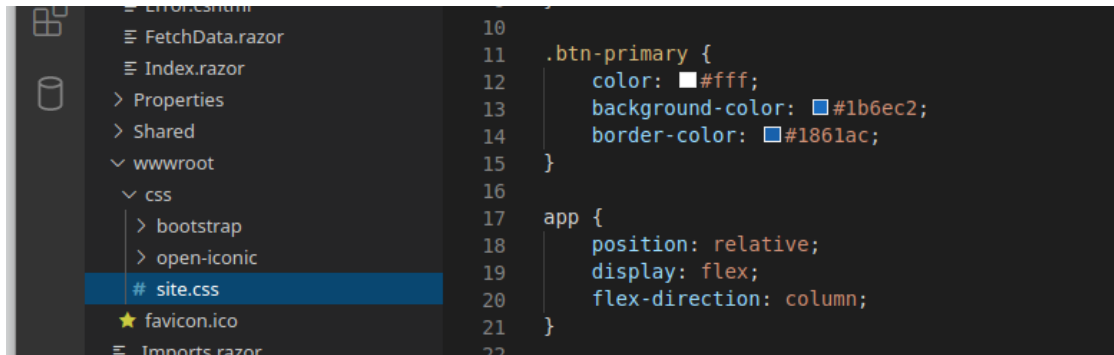
You should have the following output:



However, it is a very bad practice mixing HTML and Styles in the same file. Therefore we follow the recommendation of having styles in CSS files. This work is for reaching this objective. Firstly, note that we can use existing styles. In the css subfolder there are different already defined styles that (lamentably very probably) could be better than we would define. Let's see the site.css file. Note that the style “btn-primary” is already defined, thus we can use it.

W3-Step 2. Using an existing style.

Please, open the file site.css which is in the folder wwwroot/css. The following image shows the style *btn-primary* in the file site.css



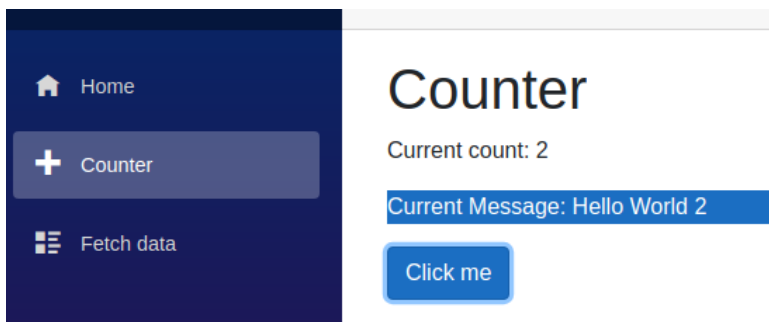
Therefore, in place of specifying a set of styles, we can use the defined style class. Thus I have changed the attribute from “style” to “class”. Try the following change (Line 6) in Counter.razor:

```

5 <p>Current count: @currentCount</p>
6 <p class="btn-primary">Current Message: @currentMsg</p>
7
8 <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
9

```

And note that the output changes to



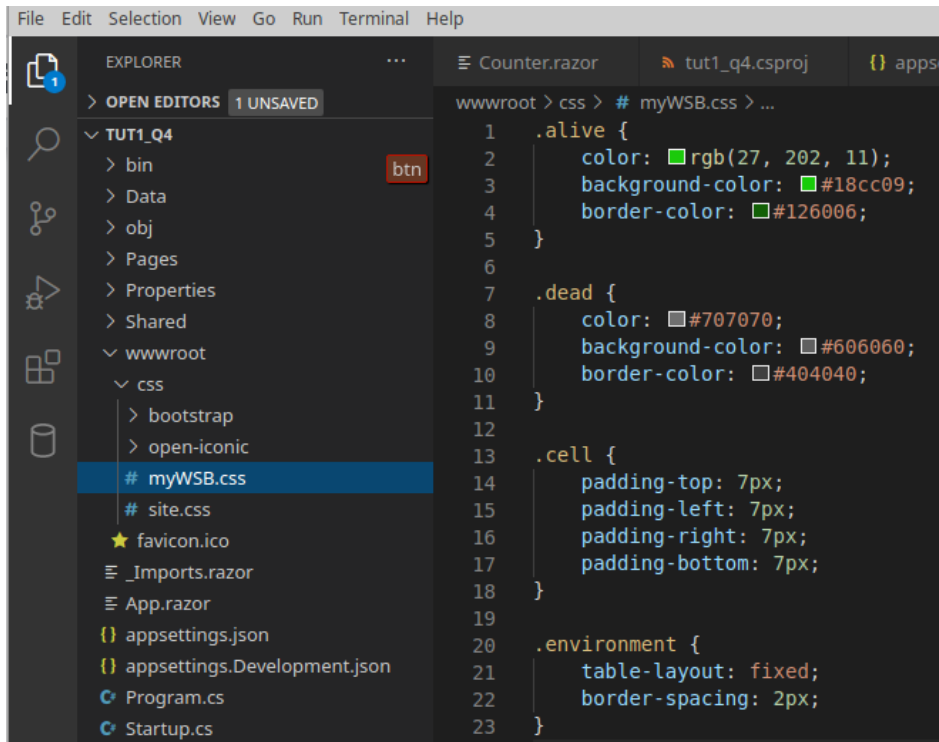
W3- Step 3. Upload the following screenshots:

Tut10-W3-A The new code fragment in the site.css file

Tut10-W3-B Output in the browser of the Counter page

Work 4. (15%). Creating and using our own styles

W4-Step 1. In order to advance to an implementation of the problem of “*Game of Life*” we will define a file of styles to display an environment (a table) that will contain “dead” cells and alive cells. The file name should be myWSB.css. To create it you can press the right button on the css folder and create a new file. Then create its content following the next screenshot (23 lines):



W4-Step 2. To make these styles visible in the project you must go to the **site.css** file, which is the css referenced file, and add the **@import** sentence for myWSB.css (Line 2) as is in the following figure:



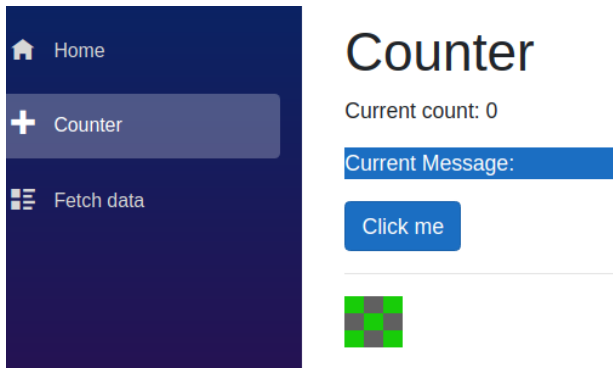
W4-Step 3. We will use the recently created styles by generating a small table of 3x3 in the Counter.razor file. Please add the lines 9 to 27 which create a table of three rows (tags <tr>) having three cells each one (tags <td>). The cells have no content but the style “cell” fixes the dimensions and the style “alive” and “dead” fixes the colors.

```

 8 <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
 9 <hr>
10 <table class="environment">
11   <tr>
12     <td class="cell alive"></td>
13     <td class="cell dead"></td>
14     <td class="cell alive"></td>
15   </tr>
16   <tr>
17     <td class="cell dead"></td>
18     <td class="cell alive"></td>
19     <td class="cell dead"></td>
20   </tr>
21   <tr>
22     <td class="cell alive"></td>
23     <td class="cell dead"></td>
24     <td class="cell alive"></td>
25   </tr>
26 </table>
27 <hr>
28
29 @code

```

Run the system with these changes. You should obtain a screenshot like this:



W4-Step 4. Upload the screenshots using the following labels:

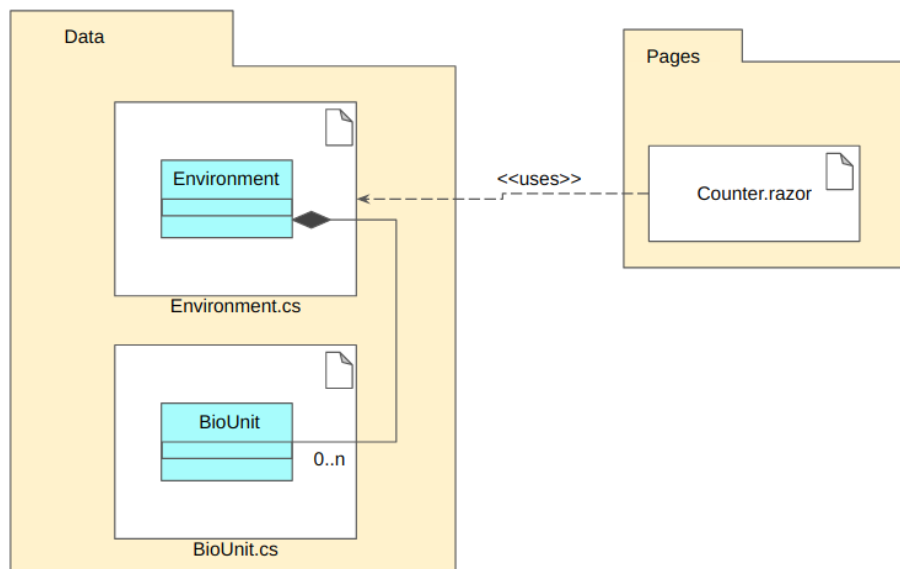
Tut10-W4-A The code of the new CSS file

Tut10-W4-B The code of the table on the Counter.razor page

Tut10-W4-C Output in the browser of Counter page

Work 5. (20%). Adding the Model classes of the Game of Life and generating the View.

W5-Step 1. As part of the material of this tutorial the file **Model_01.zip** was provided. It contains two C# classes. Environment and BioUnit. The class Environment, in the file Environment.cs, represents Conway's Environment in the Game of Life. This class contains a two-dimensional array of biological units, represented by objects of the class BioUnit. I have preferred to call them BioUnits in order, maybe in the future, to simulate Environments having different kinds of species. Put both files in the folder Data. The UML deployment diagram is presented as documentation of this project.



The BioUnit class basically is a boolean (alive) which represents the state of the biological unit (called “cell” in the Game of Life). The following is the provided code. **You must ONLY change the namespace using the name of your project in the first part.** The second part will continue to be “Data”


```

Data > BioUnit.cs > {} tut1_q4.Data
1 namespace tut1_q4.Data
2 {
3     3 references
4     public class BioUnit
5     {
6         5 references
7         private bool alive;
8         1 reference
9         public BioUnit() {
10             this.alive = false;
11         }
12         1 reference
13         public bool is_alive() {
14             return this.alive;
15         }
16         0 references
17         public bool is_dead() {
18             return !this.alive;
19         }
20         1 reference
21         public void live() {
22             this.alive = true;
23         }
24         1 reference
25         public void die() {
26             this.alive = false;
27         }
28     }
29 }

```

The following is the provided code for the Environment. The constructor is the more complex method (from lines 8 to 15) because it fills the matrix of BioUnit objects. **You must change the namespace in this file too.**

```

1 namespace tut1_q4.Data
2 {
3     1 reference
4     public class Environment
5     {
6         5 references
7         private int rows = 1;
8         5 references
9         private int cols = 1;
10        5 references
11        private BioUnit[,] cell;
12        1 reference
13        public Environment(int rows_,int columns_) {
14            this.rows = rows_;
15            this.cols = columns_;
16            this.cell = new BioUnit[this.rows,this.cols];
17            for(var i=0; i<this.rows; i++)
18            for(var j=0; j<this.cols; j++)
19                this.cell[i,j] = new BioUnit();
20        }
21        1 reference
22        public int total_of_rows(){
23            return this.rows;
24        }
25        1 reference
26        public int total_of_cols() {
27            return this.cols;
28        }
29    }

```

```

22        3 references
23        public void live(int i,int j) {
24            if(this.rightPos(i,j))
25                this.cell[i,j].live();
26        }
27        0 references
28        public void die(int i,int j) {
29            if(this.rightPos(i,j))
30                this.cell[i,j].die();
31        }
32        3 references
33        private bool rightPos(int i,int j){
34            return i>=0 && i<this.rows && j>=0 && j<this.cols;
35        }
36        1 reference
37        public bool is_alive(int i,int j) {
38            if(this.rightPos(i,j))
39                return this.cell[i,j].is_alive();
40            return false;
41        }
42    }

```

W5-Step 2. Change the View in order to put a generic table using the Environment methods for parsing the Environment. Now Counter.razor must look like this (you must change it):

```

Pages > Counter.razor
1 @page "/counter"
2
3 <h1>Game of Live</h1>
4 <p>Current count: @currentCount</p>
5 <hr>
6 @e
7     e.live(2,3);
8     e.live(4,5);
9
10 <table class="environment">
11     @for(var i=0; i<@e.total_of_rows() ; i++) {
12         <tr>
13             @for(var j=0; j<@e.total_of_cols(); j++) {
14                 @if(e.is_alive(i,j)) {
15                     <td class="cell alive"></td>
16                 }
17                 else
18                 {
19                     <td class="cell dead"></td>
20                 }
21             }
22         </tr>
23     }
24 </table>

```

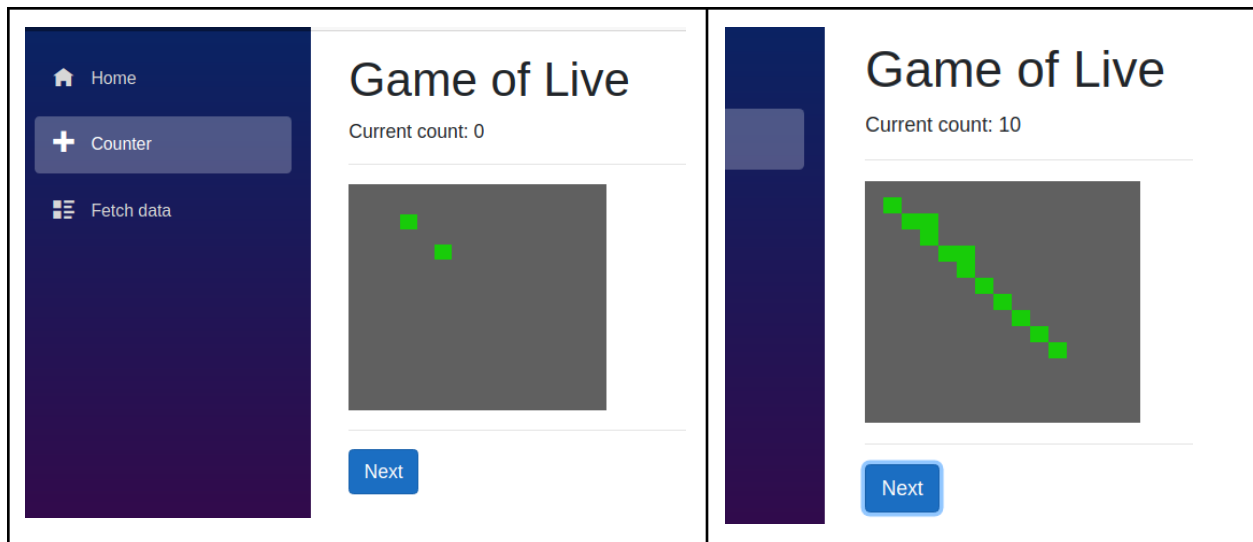
```

25 <hr>
26 <button class="btn btn-primary" @onclick="IncrementCount">Next</button>
27
28 @code {
29     3 references
    private int currentCount = 0;
30     6 references
    private Data.Environment e = new Data.Environment(15,15);
31     1 reference
    private void IncrementCount()
32     {
33         currentCount++;
34         e.live(currentCount,currentCount);
35     }
36 }

```

The changes in Counter.cs are two: (1) The inclusion of the object Environment, and the use of the method “live” for transforming a cell to an alive state and, (2) The change on displaying the table. In this case, we have used the values provided by the object e (Environment) corresponding to the number of rows and columns. To display each cell we rescue the value of the corresponding cell (green for alive). Note that syntax is the same that c# and the only difference is that control sentences begin with “@”.

W5-Step 3. Execute the resulting project. You should obtain the following output before and after 10 clicks on the button “Next”.



W5-Step 4. Upload some screenshots using the following labels:

Tut10-W5-A Screenshot of the first part of BioUnit class

Tut10-W5-B Screenshot of the first part of Environment class

Tut10-W5-C Output in the browser of the Counter page.

Work 6. (20%). Completing the Game of Life

W6-Step 1. In order to complete the Game of Life, we need to count the number of neighbors of each cell. I do this in the class Environment. For this purpose, I have implemented the following version which uses “inline if” and also the fact that *is_alive* method returns false for unexisting cells.

```
public int aliveNeighbors(int i,int j) {  
    int c=0;  
    c += this.is_alive(i-1,j-1)?1:0;  
    c += this.is_alive(i-1,j)?1:0;  
    c += this.is_alive(i-1,j+1)?1:0;  
    c += this.is_alive(i,j-1)?1:0;  
    c += this.is_alive(i,j+1)?1:0;  
    c += this.is_alive(i+1,j-1)?1:0;  
    c += this.is_alive(i+1,j)?1:0;  
    c += this.is_alive(i+1,j+1)?1:0;  
    return c;  
}
```

Note that it is not an image.

A little more complex is the method `nextConwayStep`, also in the class Environment. This method calculates the next generation of cells. Due to this can provoke changes to cells, it is necessary to keep a buffer (auxiliary memory) for the new generation. This requires copying the buffer at the end of the process. The code is the following (again it is not an image).

```
public void nextConwayStep() {  
    int n;  
    bool[,] aux = new bool[this.rows,this.cols];  
    for(var i=0; i<this.rows; i++)  
        for(var j=0; j<this.cols; j++) {  
            n = this.aliveNeighbors(i,j);  
            if(n==3) //Conway's original rule  
                aux[i,j] = true;  
            else if (n==2 && this.is_alive(i,j))  
                aux[i,j] = true;  
            else  
                aux[i,j] = false;  
        }  
}
```

```

        for(var i=0; i<this.rows; i++)
        for(var j=0; j<this.cols; j++) {
            if(aux[i,j])
                this.live(i,j);
            else
                this.die(i,j);
        }
    }
}

```

Put these two methods as part of the class Environment.

W6-Step 2. Modify the file Counter.cs to its new version which follows:

```

1  @page "/counter"
2
3  <h1>Game of Live</h1>
4  <p>Current count: @currentCount</p>
5  <hr>
6  @{
7      //only the first time (initial pattern)
8      if(currentCount==0) {
9          e.live(3,3);
10         e.live(3,4);
11         e.live(3,5);
12         e.live(2,5);
13         e.live(1,4);
14     }
15
16 }
17 <table class="environment">
18     @for(var i=0; i<@e.total_of_rows() ; i++) {
19         <tr>
20             @for(var j=0; j<@e.total_of_cols(); j++) {
21                 @if(e.is_alive(i,j)) {
22                     <td class="cell alive"></td>
23                 }
24                 else
25                 {
26                     <td class="cell dead"></td>
27                 }
28             }
29         </tr>
30     }
31 </table>
32 <hr>
33 <button class="btn btn-primary" @onclick="IncrementCount">Next</button>
34
35 @code {
36     3 references
37     private int currentCount = 0;
38     9 references
39     private Data.Environment e = new Data.Environment(15,15);
40     1 reference
41     private void IncrementCount()
42     {
43         currentCount++;
44         e.nextConwayStep();
45     }
46 }

```

Sorry, this is an image. But, most of the code is already done.

The first part (Lines 6 to 16) is the current way of specifying a pattern or initial state. The lines 17 to 33 were implemented in the previous work. In lines 35 to 43 the environment is defined and the function IncrementCount adds one Conway's step.

W6-Step 3. Run the resulting system. You should have an output like this:



W6-Step 4. Upload some screenshots using the following labels:

Tut10-W6-A The new version of Counter.razor file

Tut10-W6-B Two outputs in the browser like the previous one.

If you reach this step, you have reached 100% !

Moreover, you have your own version of the Game of Life in C#.