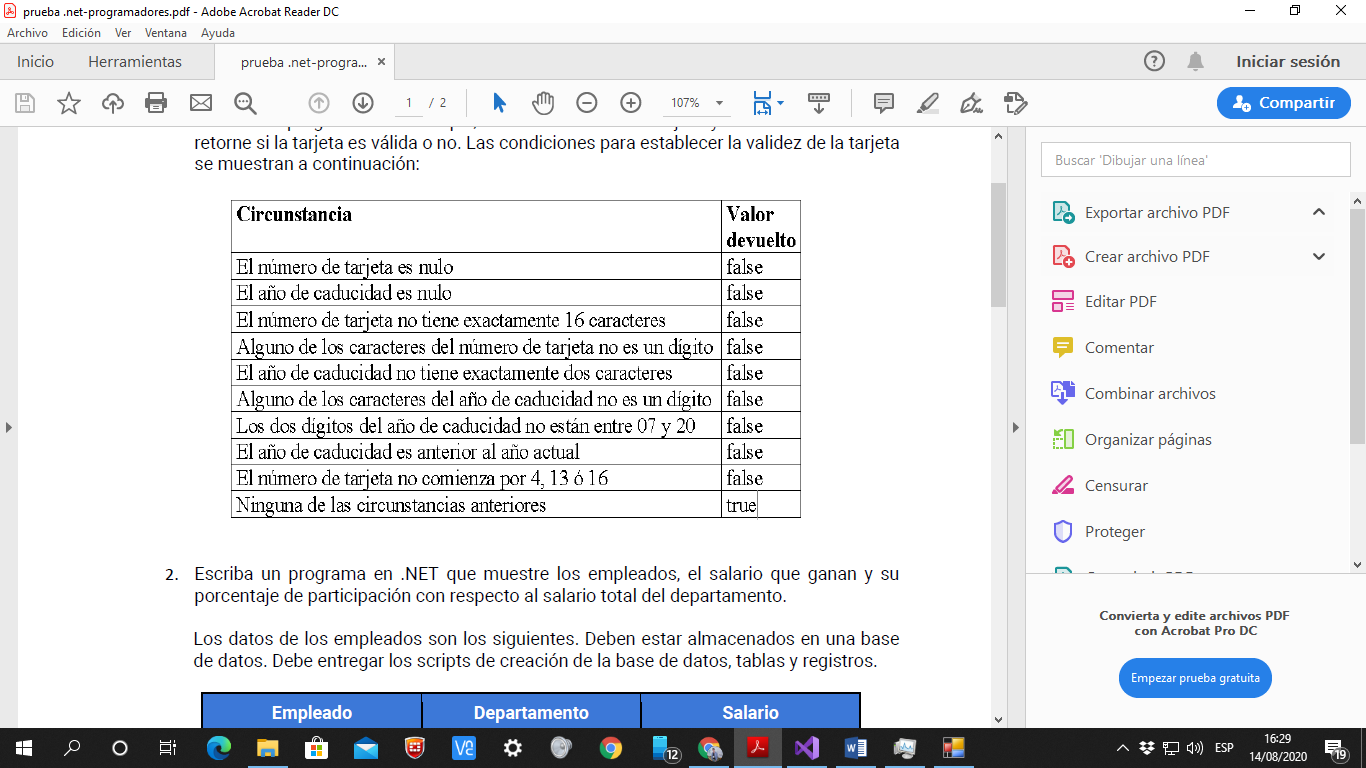
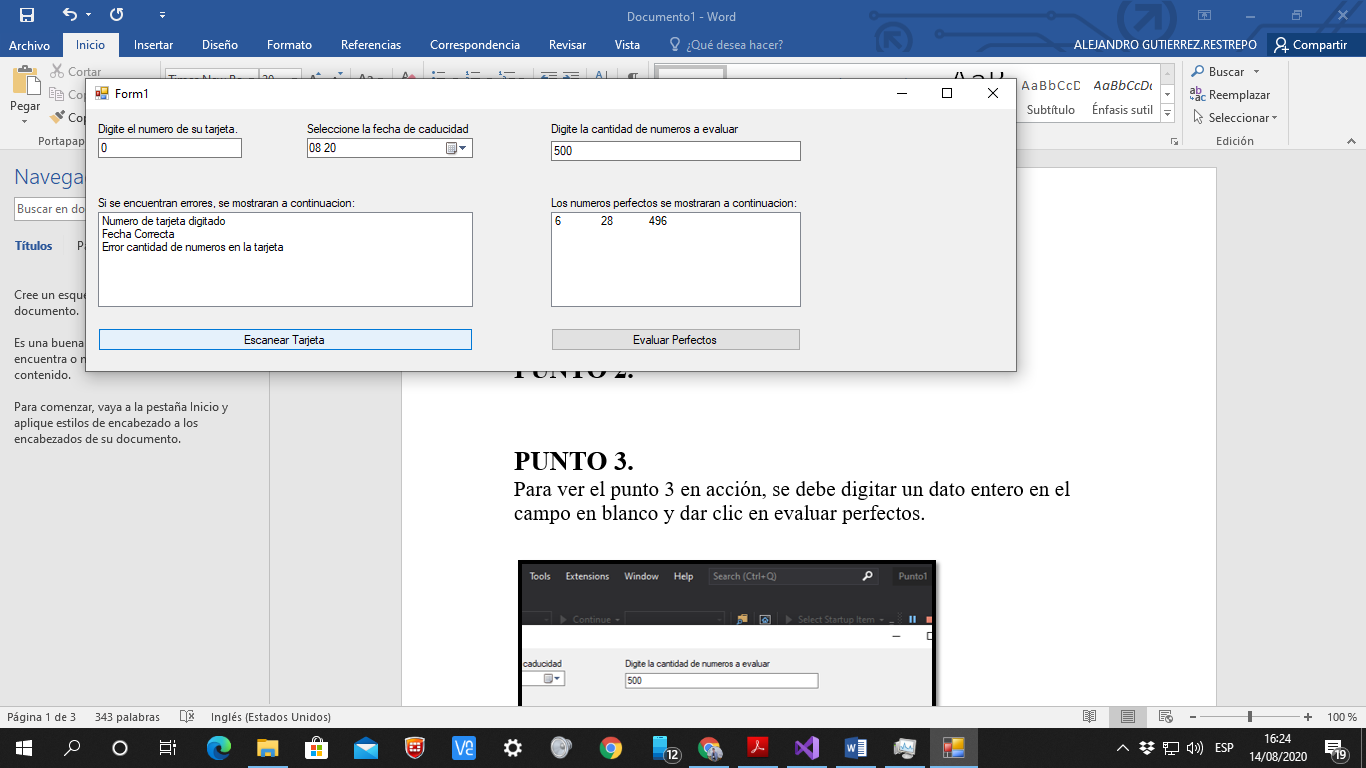
**PUNTO 1.**

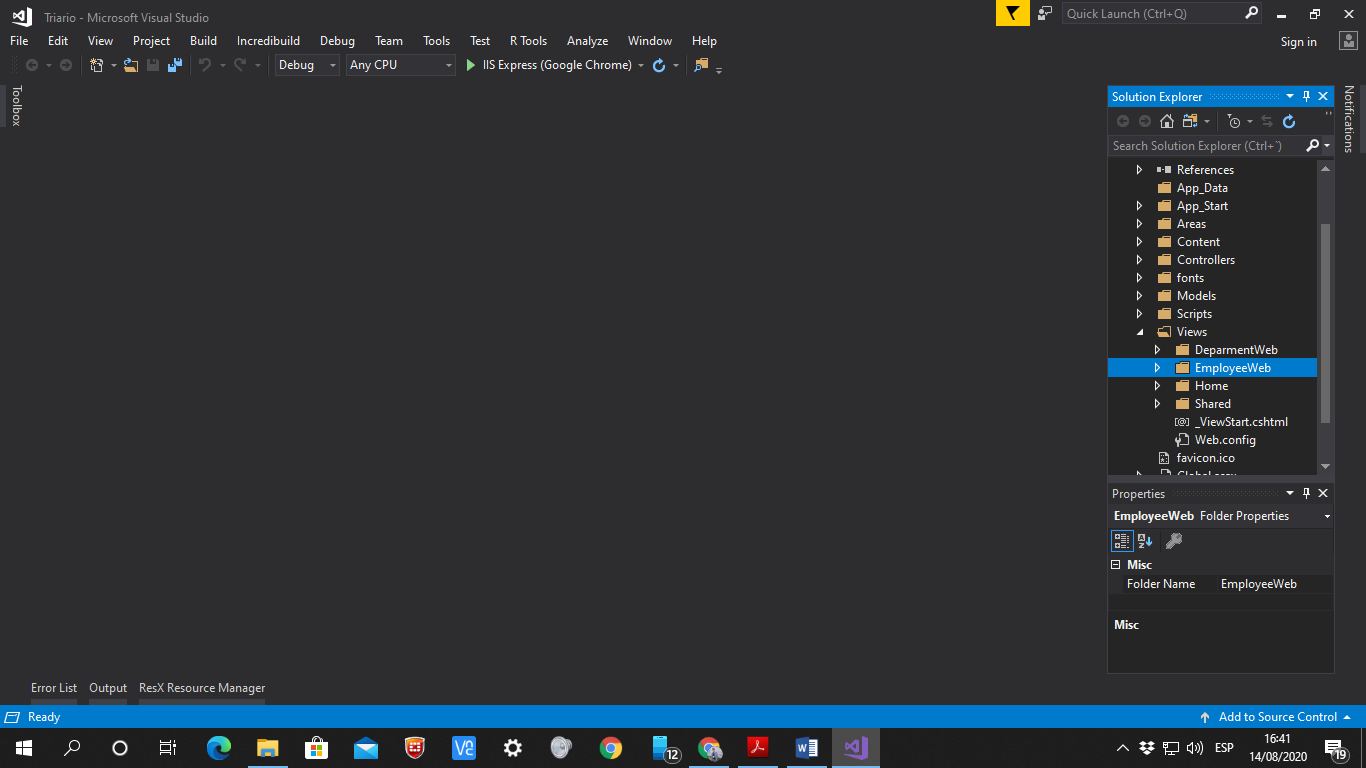
Para ver el punto 1 en acción, se debe digitar un dato entero en el campo en blanco, seleccionar la fecha de vencimiento (aunque se selecciona día, mes, año, solo se toman en cuenta mes año “MM yy”) y dar clic en Escanear Tarjeta.

En caso de haber errores de digitación como los que se pidieron a continuación(Hay que tener en cuenta que algunos errores se evitaban directamente condicionando el formulario entonces no se tuvieron en cuenta dentro de la lista):

Entonces el error sobre el cual se paró la ejecución se mostrará al final del listado en blanco en la parte inferior, la ejecución no continua hasta que el error haya sido cambiado, al terminar la ejecución correctamente al final sale “Se terminó todo correctamente estado : true”.

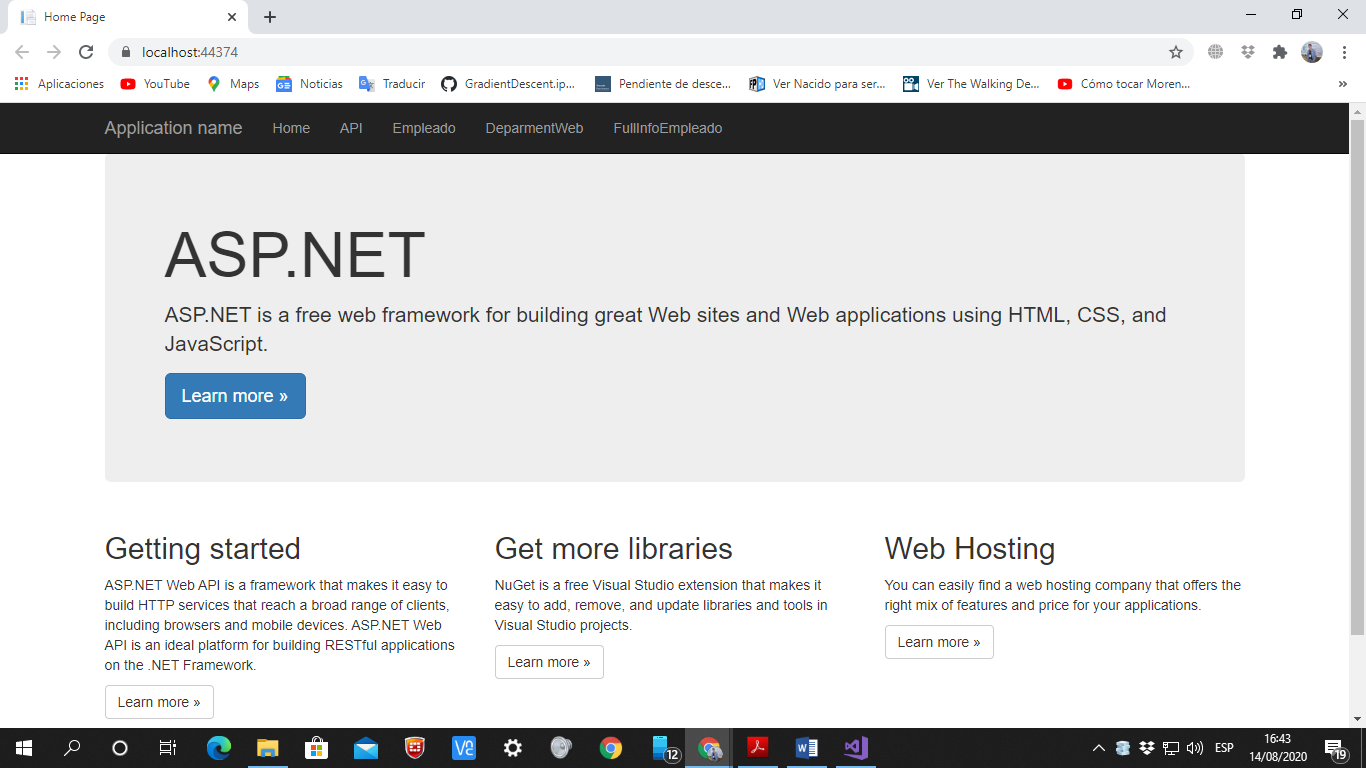


**PUNTO 2.**

El punto 2 fue el motivo principal sobre el cual se realizó este documento ya que se realizó por medio de servicios REST consumiendo un api, de todas formas no se publicó, por ende el proyecto aunque lo consume como api, la ruta está marcada en localhost, debe ser cambiada la ruta si se desea ser probado en otra máquina ya que puede variar el puerto, y no se toma automáticamente.

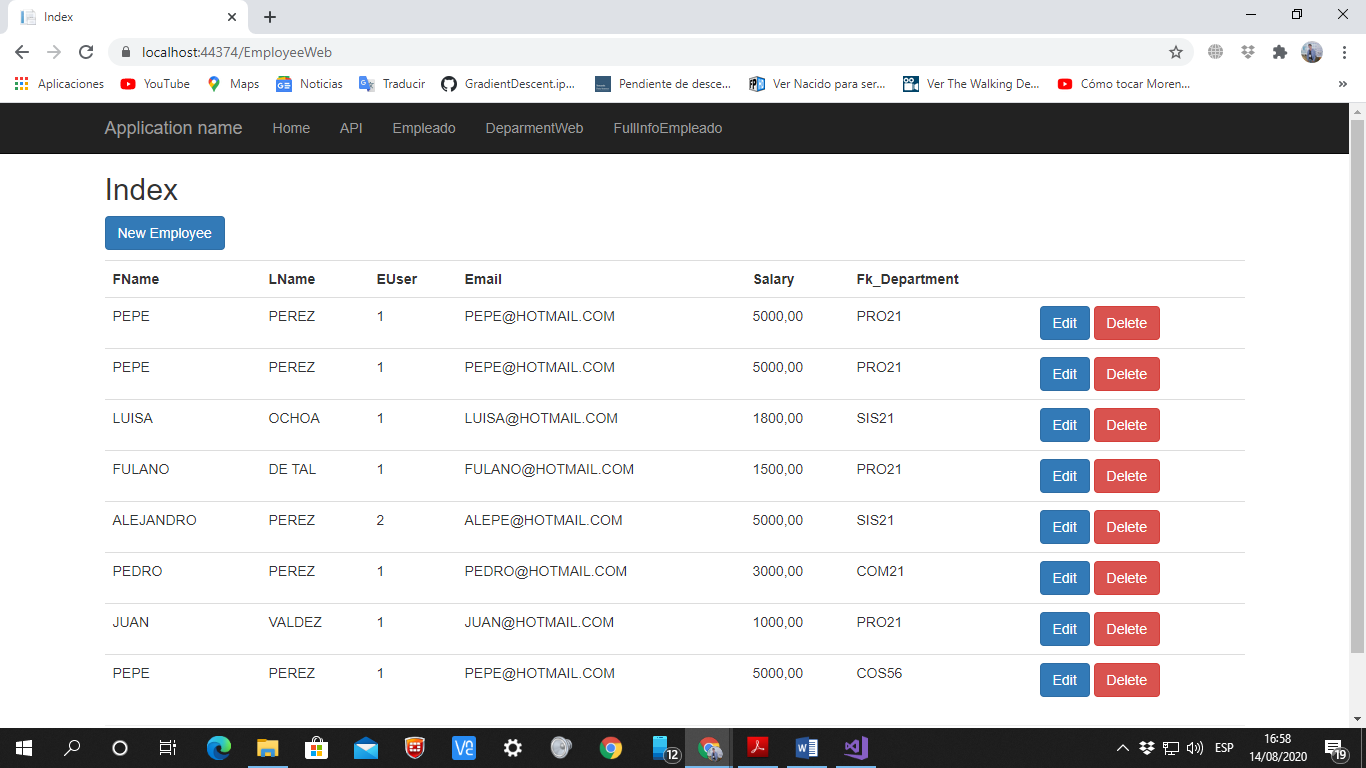
En el proyecto pueden encontrar las vistas en DepartmentWeb y EmployeeWeb.

Aunque al correrlo directamente les muestra la siguiente vista general.



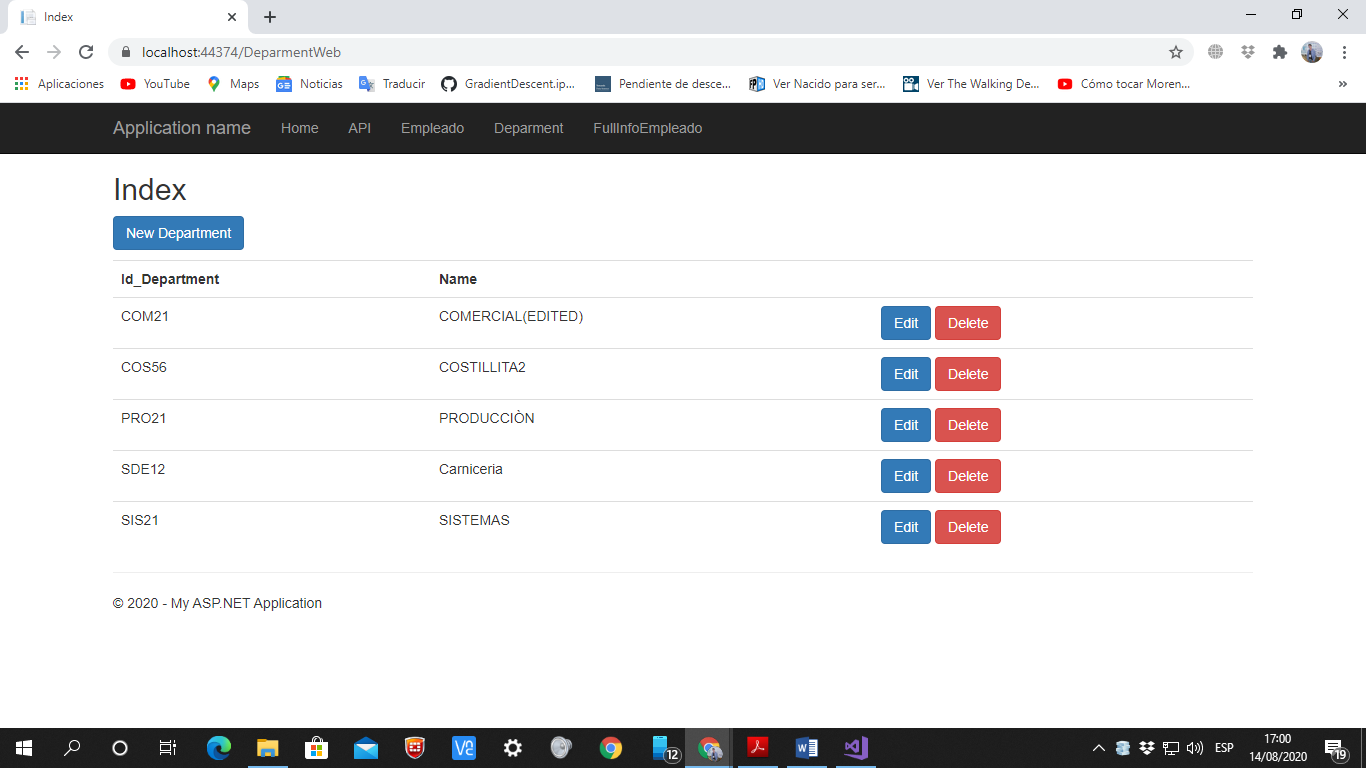
**Aquí podemos encontrar 3 botones importantes dentro del menú, Empleado, Departmentweb y FullInfoEmpleado.**

**Empleado:**



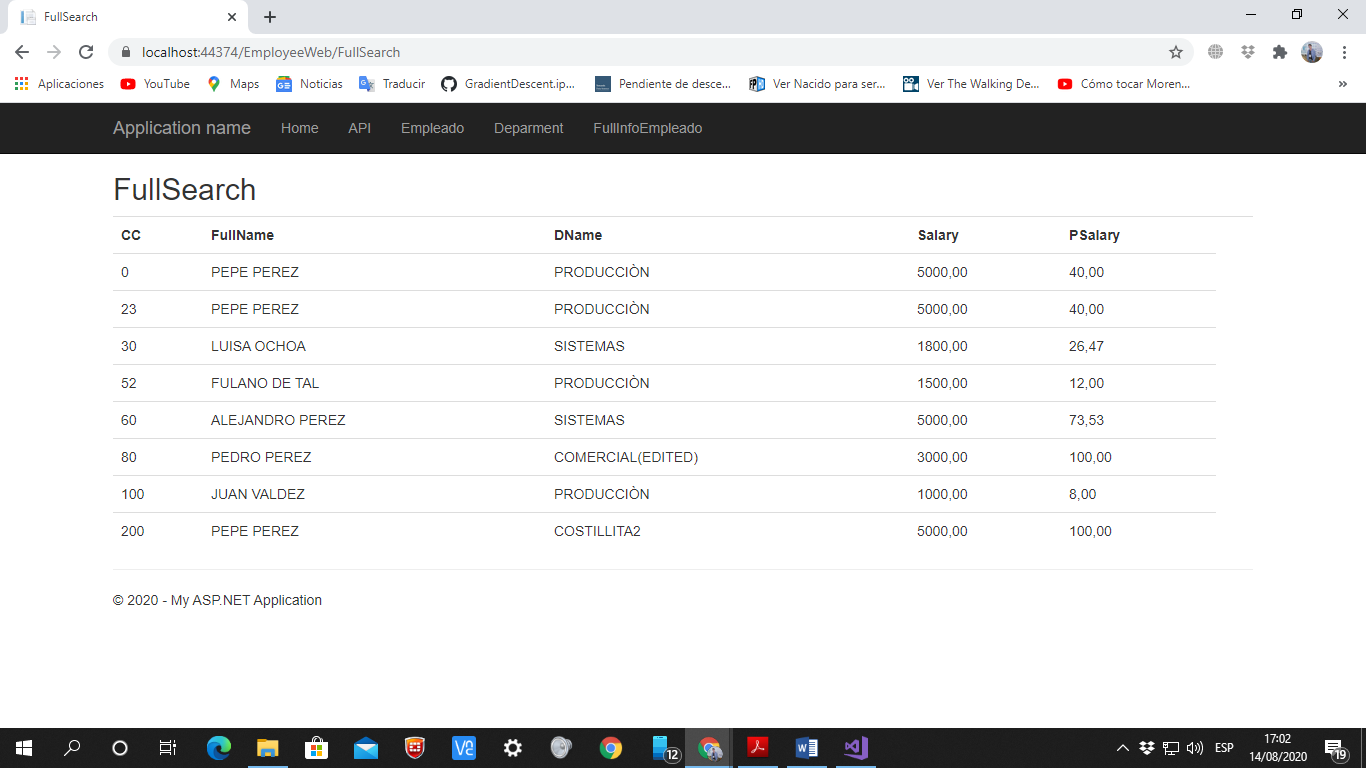
Donde puedes encontrar los diferentes botones de navegación, edición y creación de los mismos empleados.

**Department:**



Donde puedes encontrar los diferentes botones de navegación, edición y creación de los mismos Departamentos, hay que tener en cuenta que no puedes eliminar un departamento si ya fue asignado a un empleado.

**FULL EMPLEADO:**

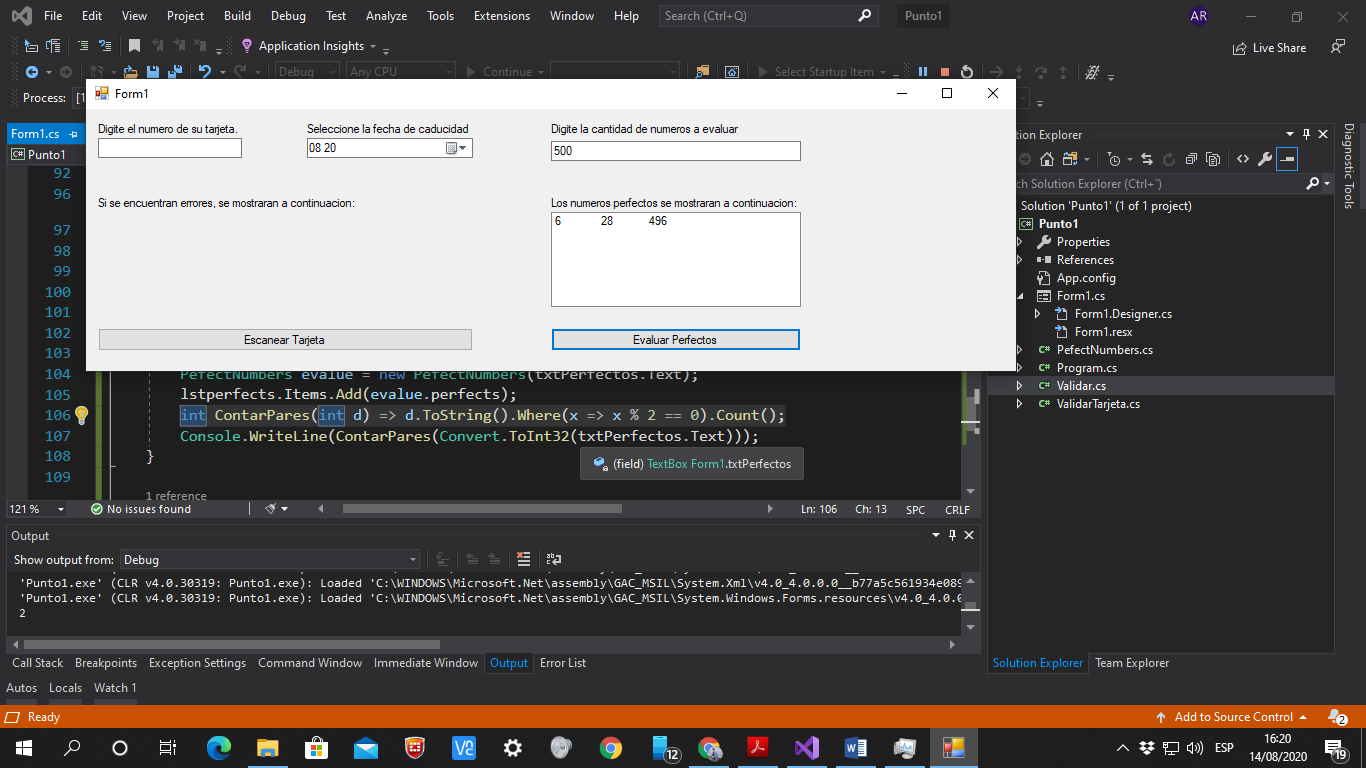


Aquí es donde se ve la información pedida dentro del requerimiento, la cual es el % de consumo del salario por usuario dentro del neto total del por departamento. Esta tabla no deja ser editada.

IMPORTANTE: los métodos que se generaron para este programa fueron generados en su mayoría a mano, la vista se usó el auto generador y solo fueron arregladas algunos detallitos a mano, los métodos dentro de los controladores devuelven más datos que usualmente se usan para evaluar como el estado y mensajes, cada procedure esta con tratamiento de errores al igual que los mismos métodos. Hay más métodos que los que se usan como tal dentro del controlador, todo fue realizado con procedures, lo que tenía que ver con consulta en BD.

**PUNTO 3.**

Para ver el punto 3 en acción, se debe digitar un dato entero en el campo en blanco y dar clic en evaluar perfectos.

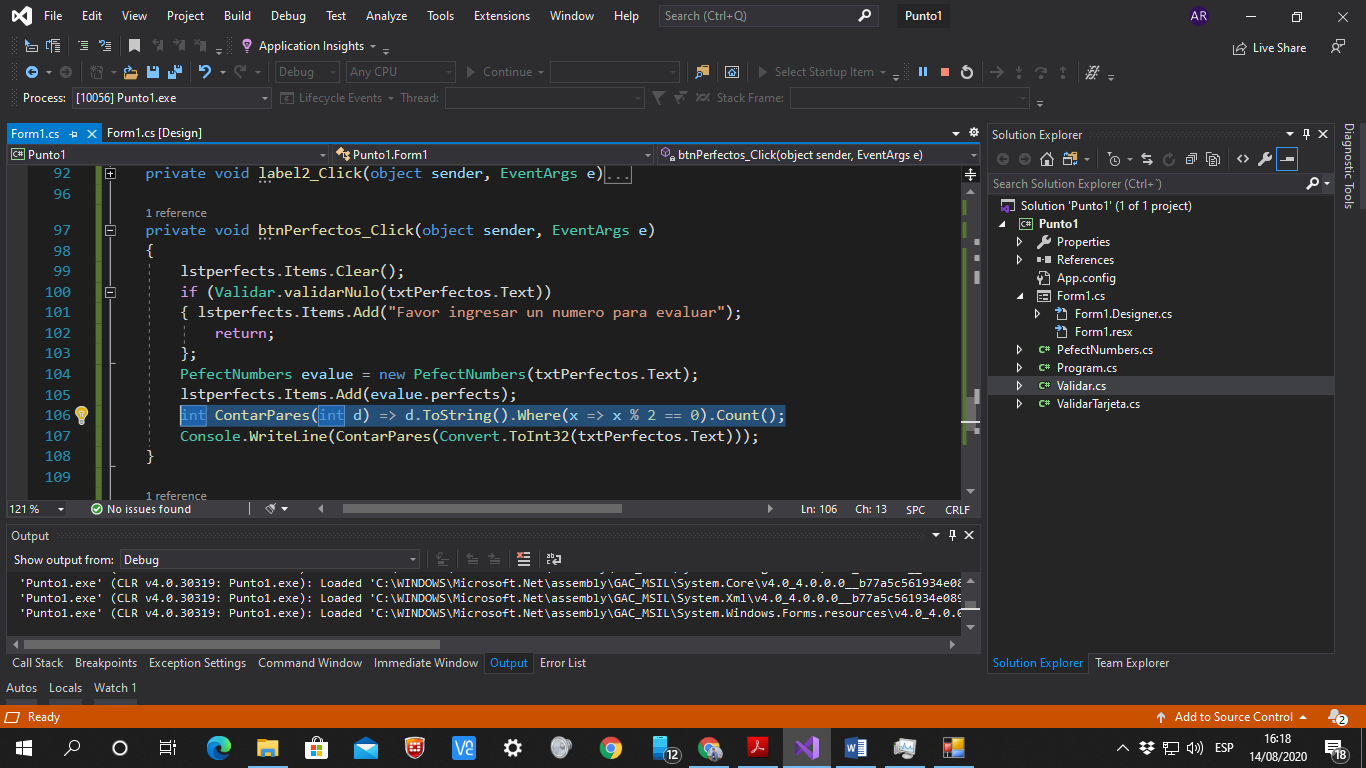


**PUNTO 4.**

La siguiente expression realiza el conteo de pares dado un numero entero.

int ContarPares(int d) => d.ToString().Where(x => x % 2 == 0).Count();

si desean ver las líneas en acción, dirigirse al Proyecto del punto 1 y 3, lo escribe en consola ya que no pertenece al programa.



**PUNTO 5.**

**ANTES**

1. public void metodoA(int[],int pos)
2. {
3. int sum=0;
4. while (i =pos){
5. sum=sum+vec[1];
6. }
7. return sum;
8. }

**DESPUES**

public int metodoA(int[] vec, int pos)

{

int i = 0;

int sum = 0;

while (i <= pos)

{

sum = sum + vec[i];

i++;

}

return sum;

}

**Explicación de porque la respuesta:**

**Línea 1:** el método tiene 2 errores, indica que es un vector que no devuelve nada pero tiene un retorno en la línea 7, también, dentro de los parámetros que se reciben solo pone el tipo del 1º parámetro “int[]”, falto ponerle el alias.

**Línea 3:** se agrega la inicialización de i=0, para el funcionamiento del “while”;

**Línea 4**: suponiendo que la intención del “while” es que se rompa cuando “i” sea igual a “pos”, entonces le falta otro igual. Adicional a esto, cuando uno conoce el límite de iteraciones se recomienda usar un “for” no un “While” evitándose así un acumulador externo.

**Línea 5:** suponiendo ya que tiene una sintaxis de acumulador, que lo que busca es sumar los datos que contiene el vector, dentro de vec[], debe contener a “i” y no un 1, también puede cambiar la sintaxis de la línea por sum+=vec[i];

**Línea 6:** se necesita que i aumente su valor para poder que el while se rompa en algún momento por ende se agrega en la línea i++}

**Línea 7:** normalmente si un método no retorna, no adiciona el “return”, al suponer que se requiere el retorno del acumulador sum entonces se cambia la línea 1