

Validación y Pruebas

(Práctica 2 - EncuestasUBU)



UNIVERSIDAD DE BURGOS

Autores:

Mario Ubierna San Mamés

Jorge Navarro González



Índice de Contenido

Definición de los requisitos	4
ProductBackLog	5
SprintBackLog	7
Historia de Usuario	8
Casos de Prueba	10
Historia Gráfica	13
Sprint1.....	14
estadoEncuestas	14
numeroRespuestas	20
numeroEncuestas	24
rankingEncestasPorRespuestas	26
rankingEncuestasPorValoracion	29
Media.....	33
Mediana.....	36
Desvest	39
numRespRangosPorEncuesta	42
numRespRangos	46
Sprint2.....	50
respuestasPorAnios	50
respuestasPorMeses.....	53
respuestasPorSemanas.....	56
Escuela Politécnica Superior	-2-



respuestasPorHoras.....	59
mediaPorEncuesta.....	62
medianaPorEncuesta.....	64
desvEstPorEncuesta.....	67
Conclusiones.....	70
Bibliografía.....	71



Definición de los requisitos

Los requisitos para la realización de esta práctica son los siguiente (nos hemos basado en el PDF que se nos ha proporcionado en ubuvirtual):

Requisitos	Descripción
R1	Se desea obtener el estado y el número de encuestas en dicho estado.
R2	Se desea obtener el número de encuestas que han recibido alguna respuesta.
R3	Se desea obtener el número de respuestas que ha recibido cada una de las encuestas.
R4	Se desea obtener el número de respuestas recibidas agrupadas por años.
R5	Se desea obtener el número de respuestas recibidas agrupadas por meses.
R6	Se desea obtener el número de respuestas recibidas agrupadas por semanas.
R7	Se desea obtener el número de respuestas recibidas agrupadas por horas.
R8	Se desea obtener el número de respuestas recibidas por cada encuesta.
R9	Se desea obtener la nota más alta por cada una de las encuestas.
R10	Se desea obtener la nota media por cada una de las encuestas.
R11	Se desea obtener la valoración media de todas las respuestas.
R12	Se desea obtener la mediana por cada una de las encuestas.
R13	Se desea obtener la mediana de las valoraciones de todas las respuestas.
R14	Se desea obtener la desviación de las valoraciones por cada encuesta.
R15	Se desea obtener la desviación estadística de las valoraciones de todas las respuestas.
R16	Se desea obtener dada una encuesta la menor y mayor valoración
R17	Se desea obtener todas las encuestas con su menor y su mayor valoración.



ProductBackLog

El ProductBackLog lo hemos hecho siguiendo el PDF proporcionado en ubuvirtual. Para ello hemos asignado a cada función un requisito, les hemos asignado un esfuerzo (éste fue elegido entre nosotros dos, siendo 1 una tarea “sencilla” de resolver y siendo 5 una tarea “difícil” de resolver), también les hemos asignado una prioridad (esta ha sido establecida dependiendo del esfuerzo, a menor esfuerzo mayor prioridad ya que así le podemos entregar el cliente una versión del producto más rápido).

Requisito	Esfuerzo	Prioridad
R1 - Estado encuestas	1	1
R2 - Número encuestas	2	3
R3 - Número respuestas	1	2
R4 - Respuestas por años	5	11
R5 - Respuestas por meses	5	12
R6 - Respuestas por semanas	5	13
R7 - Respuestas por horas	5	14
R8 - Ranking encuestas por respuestas	2	4
R9 - Ranking encuestas por valoración	3	5
R10 - Media por encuesta	5	15
R11 - Media	3	6
R12 - Mediana por encuesta	5	16
R13 - Mediana	3	7
R14 - Desviación por encuesta	5	17
R15 - Desviación	3	8
R16 - Encuesta con mayor y menos valoración	3	9
R17 - Encuestas con mayores y menores valoraciones	3	10



Esta tabla es igual que la anterior, pero la ordenamos dependiendo de la prioridad, esta tabla ha sido la que hemos seguido para realizar los sprints y para saber qué método hacer antes.

Requisito	Esfuerzo	Prioridad
R1 - Estado encuestas	1	1
R3 - Número respuestas	1	2
R2 - Número encuestas	2	3
R8 - Ranking encuestas por respuestas	2	4
R9 - Ranking encuestas por valoración	3	5
R11 - Media	3	6
R13 - Mediana	3	7
R15 - Desviación	3	8
R16 - Encuesta con mayor y menos valoración	3	9
R17 - Encuestas con mayores y menores valoraciones	3	10
R4 - Respuestas por años	5	11
R5 - Respuestas por meses	5	12
R6 - Respuestas por semanas	5	13
R7 - Respuestas por horas	5	14
R10 - Media por encuesta	5	15
R12 - Mediana por encuesta	5	16
R14 - Desviación por encuesta	5	17



SprintBackLog

En el primer sprint vamos a realizar los siguientes requisitos, consideramos que deberían ser estos ya que son los que menos esfuerzo nos va a llevar:

Requisito	Esfuerzo	Prioridad
R1 - Estado encuestas	1	1
R3 - Número respuestas	1	2
R2 - Número encuestas	2	3
R8 - Ranking encuestas por respuestas	2	4
R9 - Ranking encuestas por valoración	3	5
R11 - Media	3	6
R13 - Mediana	3	7
R15 - Desviación	3	8
R16 - Encuesta con mayor y menos valoración	3	9
R17 - Encuestas con mayores y menores valoraciones	3	10

En el siguiente sprint vamos a realizar los siguientes requisitos, esto se debe a que son los que más esfuerzo nos va a llevar (porque antes no guardábamos la fecha para las respuestas, por lo que tuvimos que modificar la práctica anterior):

Requisito	Esfuerzo	Prioridad
R4 - Respuestas por años	5	11
R5 - Respuestas por meses	5	12
R6 - Respuestas por semanas	5	13
R7 - Respuestas por horas	5	14
R10 - Media por encuesta	5	15
R12 - Mediana por encuesta	5	16
R14 - Desviación por encuesta	5	17



Historia de Usuario

Descripción corta: se desea obtener las estadísticas de la aplicación.

Descripción larga: como administradores queremos poder tener acceso a las estadísticas de la aplicación.

Estas estadísticas nos van a proporcionar información sobre:

- Se desea obtener el estado y el número de encuestas en dicho estado. Esta información la podemos obtener gracias al método `estadoEncuesta()`.
- Se desea obtener el número de encuestas que han recibido alguna respuesta. Esta información la podemos obtener gracias al método `numeroEncuestas()`.
- Se desea obtener el número de respuestas que ha recibido cada una de las encuestas. Esta información la podemos obtener gracias al método `numeroRespuestas()`.
- Se desea obtener el número de respuestas recibidas agrupadas por años. Esta información la podemos obtener gracias el método `respuestasPorAnios()`.
- Se desea obtener el número de respuestas recibidas agrupadas por meses. Esta información la podemos obtener gracias al método `respuestasPorMeses()`.
- Se desea obtener el número de respuestas recibidas agrupadas por semanas. Esta información la podemos obtener gracias al método `respuestasPorSemanas()`.
- Se desea obtener el número de respuestas recibidas agrupadas por horas. Esta información la podemos obtener gracias al método `respuestasPorHoras()`.
- Se desea obtener el número de respuestas recibidas por cada encuesta. Esta información la podemos obtener gracias al método `rankingEncuestasPorRespuestas()`.
- Se desea obtener la nota más alta por cada una de las encuestas. Esta información la podemos obtener gracias al método `rankingEncuestasPorValoracion()`.
- Se desea obtener la nota media por cada una de las encuestas. Esta información la podemos obtener gracias el método `mediaPorEncuesta()`.
- Se desea obtener la valoración media de todas las respuestas. Esta información la podemos obtener gracias al método `media()`.
- Se desea obtener la mediana por cada una de las encuestas. Esta información la podemos obtener gracias al método `medianaPorEncuesta()`.
- Se desea obtener la mediana de las valoraciones de todas las respuestas. Esta información la podemos obtener gracias al método `mediana()`.
- Se desea obtener la desviación de las valoraciones por cada encuesta. Esta información la podemos obtener gracias al método `desvEstPorEncuesta()`.
- Se desea obtener la desviación estadística de las valoraciones de todas las respuestas. Esta información la podemos obtener gracias al método `desvest()`.



- Se desea obtener dada una encuesta la menor y mayor valoración. Esta información la podemos obtener gracias al método `numRespRangosPorEncuesta(string encuesta)`.
- Se desea obtener todas las encuestas con su menor y su mayor valoración. Esta información la podemos obtener gracias al método `numRespRangos()`.

Pruebas unitarias: para la realización de este apartado, hemos creado una clase `EstadisticasTest`, la cual evalúa la clase `Estadísticas`. Cada método tiene sus propias pruebas, las cuales van a ser explicadas en el apartado casos de prueba.

Casos de Prueba

Por cada uno de los requisitos hemos hecho un caso de prueba, las pruebas que hemos realizado para cada requisito del sprint 1 se recogen en la siguiente tabla.

Requisito	Pruebas
R1 - Estado encuestas	Comprobamos que cargamos de forma correcta los valores por defecto.
	Comprobamos que las filas están en orden, primero desactivas y luego activas.
	Comprobamos que si añadimos una nueva encuesta, tenemos una encuesta más en las de tipo desactivas.
	Comprobamos que, si borramos una encuesta activa tenemos una encuesta activa menos.
	Comprobamos que si modificamos una encuesta de desactiva a activa, tenemos una encuesta activa más.
R3 - Número respuestas	Comprobamos que se cargan los valores por defecto.
	Comprobamos que, al añadir una nueva opinión la encuesta determinada tiene una opinión más.
	Comprobamos que al borrar una encuesta, ya no existe ni ella ni sus opiniones.
	Comprobamos que al añadir una encuesta sus opiniones son 0.
R2 - Número encuestas	Comprobamos que cargamos los valores por defecto.
	Comprobamos que al borrar una encuesta determinada se reduce en uno el número de encuestas que hay en el sistema.
	Comprobamos que al añadir una encuesta sin opiniones, el número sigue siendo el mismo que en el punto anterior.
R8 - Ranking encuestas por respuestas	Comprobamos que cargamos los valores por defecto.
	Comprobamos que al añadir una encuesta sin opiniones, ésta se encuentra en la última posición.
	Comprobamos que al añadir 5 opiniones a una encuesta, ésta pasa a la primera posición.
R9 - Ranking encuestas por valoración	Comprobamos que cargamos los valores por defecto.
	Comprobamos que al añadir una nueva encuesta con valor 1 está en la última posición.
	Comprobamos que añadiendo a la encuesta anterior una opinión con nuevo valor, el ranking de dicha encuesta cambia.
R11 - Media	Comprobamos que cargamos los valores por defecto.
	Comprobamos que si añadimos una nueva opinión aumenta la media.
	Comprobamos que si borramos una encuesta con respuestas se reduce la media.
R13 - Mediana	Comprobamos que cargamos los valores por defecto.



	Comprobamos que al añadir una nueva respuesta, ésta no afecta a la mediana.
	Comprobamos que al borrar una encuesta, ésta no afecta al valor de la mediana.
R15 - Desviación	Comprobamos que cargamos los valores por defecto.
	Comprobamos que al añadir una nueva respuesta cambia el valor de la desviación.
	Comprobamos que al borrar una encuesta con respuestas cambia el valor de la desviación.
R16 - Encuesta con mayor y menos valoración	Comprobamos que cargamos los valores por defecto.
	Comprobamos que si añadimos una encuesta con dos opiniones, nos devuelve el resultado que esperamos su valor mínimo y su valor máximo.
	Comprobamos que si una encuesta no tiene respuestas, entonces su valor mínimo y máximo son 0.
R17 - Encuestas con mayores y menores valoraciones	Comprobamos que cargamos los valores por defecto.
	Comprobamos que si añadimos una encuesta con dos opiniones, sus valores mínimo y máximo son correctos.
	Comprobamos que si una encuesta no tiene respuestas, sus valores mínimo y máximo son 0.



Por cada uno de los requisitos hemos hecho un caso de prueba, las pruebas que hemos realizado para cada requisito del sprint 2 se recogen en la siguiente tabla.

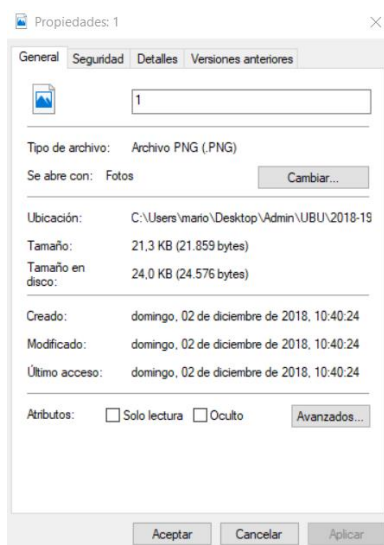
Requisito	Pruebas
R4 - Respuestas por años	Comprobamos que se cargan los valores por defecto.
	Comprobamos que si añadimos una respuesta a un año nos lo contabiliza.
R5 - Respuestas por meses	Comprobamos que cargamos los valores por defecto.
	Comprobamos que si añadimos una respuesta a un mes nos lo contabiliza.
R6 - Respuestas por semanas	Comprobamos que carga lo valores por defecto.
	Comprobamos que si añadimos una respuesta a un día nos lo contabiliza.
R7 - Respuestas por horas	Comprobamos que carga los valores por defecto.
	Comprobamos que si añadimos una nueva respuesta nos contabiliza el nuevo valor.
R10 - Media por encuesta	Comprobamos que carga los valores por defecto.
	Comprobamos que si añadimos una nueva respuesta a una determinada encuesta nos aumenta el valor de la media.
	Comprobamos que al borrar una encuesta determinada con respuestas se reduce la media.
R12 - Mediana por encuesta	Comprobamos que cargamos los valores por defecto.
	Comprobamos que si añadimos una nueva respuesta a una determinada encuesta, en nuestro caso la Encuesta1, ésta pasa a ser la primera del ranking.
	Comprobamos que si borramos la encuesta que está en el top del ranking, la segunda encuesta pasa a ser la primera del ranking.
R14 - Desviación por encuesta	Comprobamos que cargamos los valores por defecto.

Historia Gráfica

En este apartado vamos a ver cómo hemos hecho esta práctica, es decir, los pasos en TDD que hemos seguido tal y como se nos indicaba en ubuvirtual.

Para que sea más fácil el seguir cómo lo hemos hecho, vamos a ir explicando para cada método las tres fases (rojo-verde-refactorizar).

Si algún apartado no ha quedado claro o hay algún problema, envíanos un correo para así poder enseñarte que lo hemos hecho mediante TDD. La fecha y la hora de las imágenes están sacadas de las propiedades de las capturas, es decir:



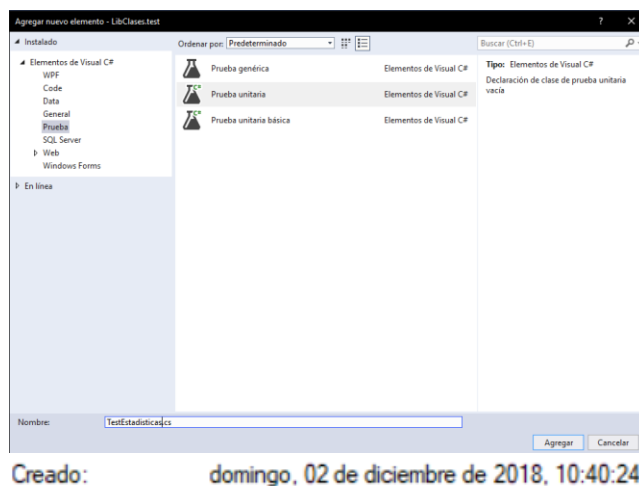
Dando botón derecho a la imagen, luego damos a propiedades y nos fijamos en la fecha de Creado.

Aun así, enviaremos las imágenes en el proyecto, para que puedas comprobar que es verdad.

Sprint1

estadoEncuestas

Esta fase para este método ha sido la más “compleja”, ya que es aquí donde hemos creado la clase EstadisticasTest para así poder empezar a realizar las pruebas.



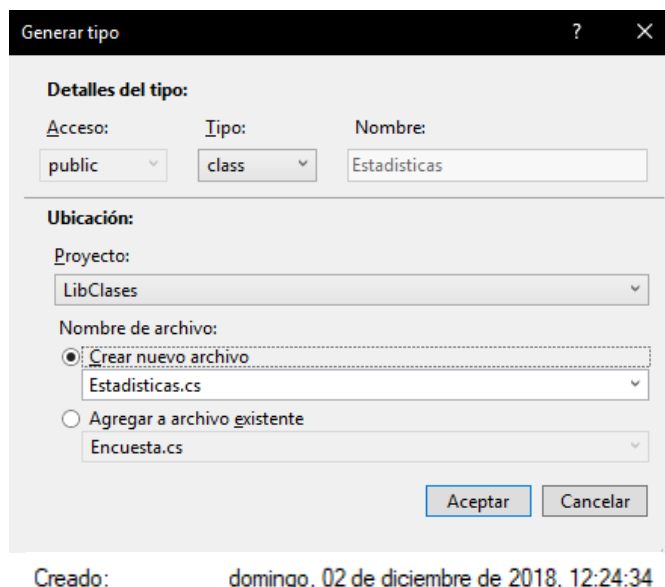
Una vez creada la clase, ya pudimos empezar a realizar el test. Cabe mencionar que para los dos primeros métodos hicimos primero una captura de la carga del test y cuando ya hicimos estos dos primeros métodos, sacamos una captura de todo el test, es por ello que tienen fechas distintas, aun así como podemos apreciar en la siguiente imagen se hizo mediante TDD, ya nos sale en rojo la clase Estadisticas (esto se debe a que no está creada) y también sale en rojo el método getEstadisticas() (ya que la base de datos todavía no contenía dicho método).

```
//Sprint1 - estadoEncuestas()  
Estadisticas estadisticas = db.getEstadisticas();  
DataTable dataTable = estadisticas.estadoEncuestas();  
Assert.AreEqual(dataTable.Rows[0][0], "activas");  
Assert.AreEqual(dataTable.Rows[0][1], 5);  
Assert.AreEqual(dataTable.Rows[1][0], "desactivas");  
Assert.AreEqual(dataTable.Rows[1][1], 5);
```

Creado: domingo, 02 de diciembre de 2018, 12:21:42

Una vez realizado el test, tuvimos que dar botón derecho a lo rojo para poder crear la clase y el método correspondiente, aunque no sale como rojo el método estadoEncuestas(), esto se debe a que todavía no estaba creada la clase Estadisticas, por lo que nos daba error en la primera línea sin tener en cuenta el error en la segunda línea.

Una vez que damos botón derecho a Estadísticas podemos generar la clase tal y como aparece en la siguiente captura.



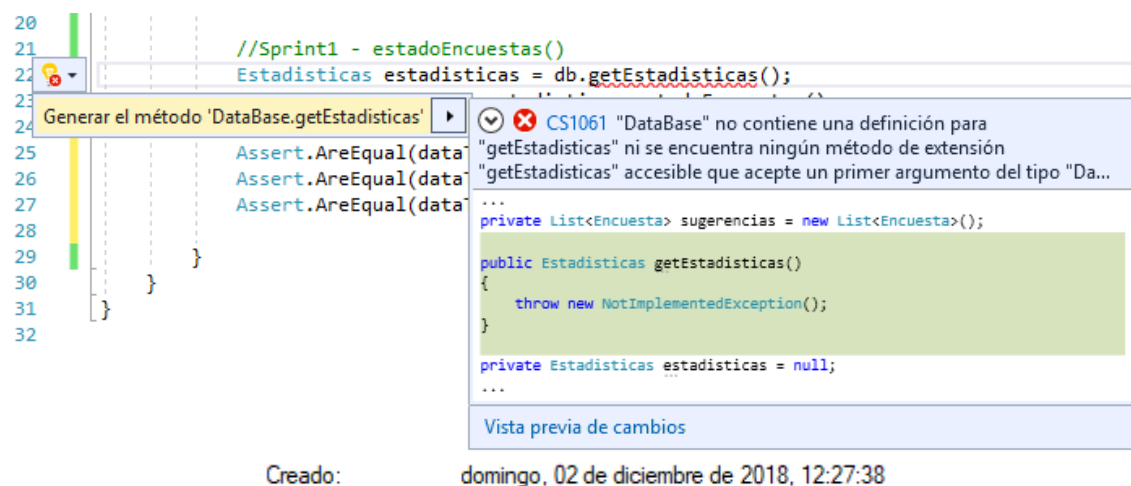
Creado: domingo, 02 de diciembre de 2018, 12:24:34

Al crear la clase nos genera el siguiente código:

```
namespace LibClases
{
    public class Estadísticas
    {
    }
}
```

Creado: domingo, 02 de diciembre de 2018, 12:25:40

De esta forma conseguimos que se nos quite el error de que no exista la clase. Luego damos botón derecho al `db.getEstadísticas()` y le indicamos que genere el método en la clase `DataBase`.



Creado: domingo, 02 de diciembre de 2018, 12:27:38

Al generar dicho código, nos genera el método correspondiente en la DataBase con el siguiente código:

```
public Estadisticas getEstadisticas()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 12:28:50

Una vez generado el método, incluimos nosotros el código que necesitamos para que la base de datos nos devuelva el objeto de tipo Estadísticas.

```
public Estadisticas getEstadisticas()
{
    estadisticas = new Estadisticas(this.getEncuestas(), this.getValoraciones());
    return estadisticas;
}
```

Creado: domingo, 02 de diciembre de 2018, 12:32:06

Como podemos apreciar en la imagen anterior nos da un error en la clase Estadísticas, esto se debe a que no tiene un constructor como el que hemos indicado en la captura anterior, es por ello que generamos el constructor, para que así deje de estar rojo.

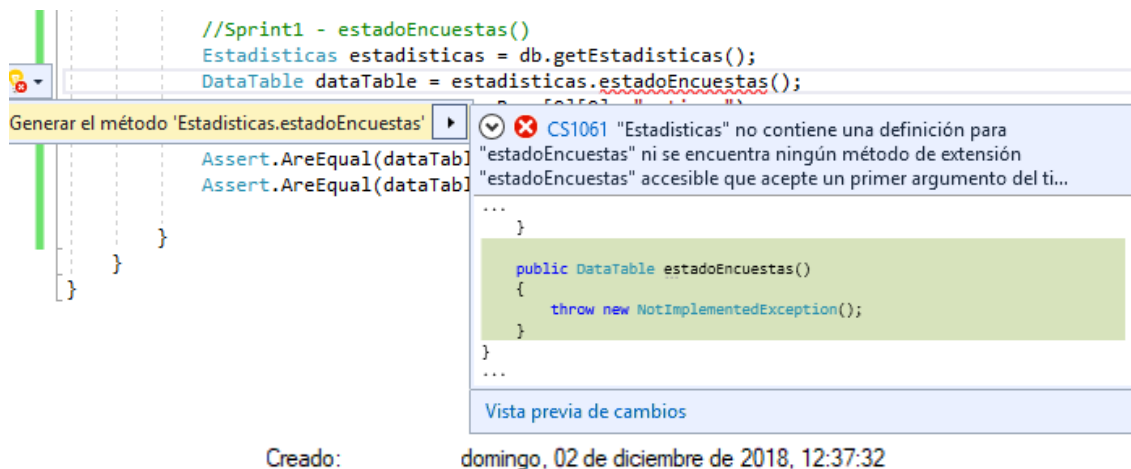


```
public class Estadisticas
{
    private List<Encuesta> encuestas;
    private List<Valoracion> valoraciones;

    public Estadisticas(List<Encuesta> encuestas, List<Valoracion> valoraciones)
    {
        this.encuestas = encuestas;
        this.valoraciones = valoraciones;
    }
}
```

Creado: domingo, 02 de diciembre de 2018, 12:35:11

Una vez solucionado estos errores, podemos volver al test, y como podemos apreciar en la siguiente imagen ahora sí que sale en rojo el método estadoEncuestas().



Para solucionarlo, damos botón derecho y generamos el método.

```
public DataTable estadoEncuestas()  
{  
    throw new NotImplementedException();  
}
```

Creado: domingo, 02 de diciembre de 2018, 12:38:07

Y rellenamos el método para que nos devuelva por cada tipo de encuesta, el número que están activas y desactivas.

```
public DataTable estadoEncuestas()
{
    DataTable dataTable = new DataTable();
    int activas = 0;
    int desactivas = 0;

    foreach(Encuesta e in encuestas)
    {
        if(e.Activa)
        {
            ++activas;
        }
        else
        {
            ++desactivas;
        }
    }
    dataTable.Columns.Add("Estado", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));
    dataTable.Rows.Add("activas", activas);
    dataTable.Rows.Add("desactivas", desactivas);

    return dataTable;
}
```

Creado:

domingo, 02 de diciembre de 2018, 12:43:19

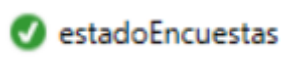
Como bien hemos mencionado antes, para los dos primeros métodos al principio solo hicimos una pequeña captura del test (porque no sabíamos si ponerte toda la imagen del test o no), por lo que después cambiamos de idea y sacamos una nueva captura del test entero, es por eso que tienen distinta fecha.

```
[TestMethod]
public void estadoEncuestas()
{
    DataBase db = new DataBase();

    //Sprint1 - estadoEncuestas()
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.estadoEncuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "activas");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
    Assert.AreEqual(dataTable.Rows[1][0], "desactivas");
    Assert.AreEqual(dataTable.Rows[1][1], 5);
    //Comprobamos que las filas están en orden
    Assert.AreNotEqual(dataTable.Rows[0][0], "desactivas");
    Assert.AreNotEqual(dataTable.Rows[1][0], "activas");
    //Comprobamos que si añadimos una nueva encuesta, tenemos una nueva encuesta desactivada
    db.addEncuesta("Encuesta1", "Encuesta1descripción");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.estadoEncuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "activas");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
    Assert.AreEqual(dataTable.Rows[1][0], "desactivas");
    Assert.AreEqual(dataTable.Rows[1][1], 6);
    //Comprobamos que si borramos una encuesta activa, tenemos una encuesta activa menos
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.estadoEncuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "activas");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[1][0], "desactivas");
    Assert.AreEqual(dataTable.Rows[1][1], 6);
    //Comprobamos que si modificar una encuesta de desactiva a activa, tenemos 5 activas y 5 desactivas
    db.getEncuesta("Encuesta2").Activa = true;
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.estadoEncuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "activas");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
    Assert.AreEqual(dataTable.Rows[1][0], "desactivas");
    Assert.AreEqual(dataTable.Rows[1][1], 5);
}
```

Creado: domingo, 02 de diciembre de 2018, 13:00:38

Una vez que ya hemos hecho todo el código, podemos ejecutar el test, y como podemos observar en la siguiente imagen, lo pasa correctamente:



En cuanto a la refactorización, en este caso no hemos considerado que hiciera falta, ya que el código que hemos generado ya estaba refactorizado.

numeroRespuestas

Para realizar este método, hemos seguido los mismos pasos que en el caso anterior, es decir, primero generamos el test.

```
//Sprint1 - numeroRespuestas()
[TestMethod]
public void numeroRespuestas()
{
    DataBase db = new DataBase();

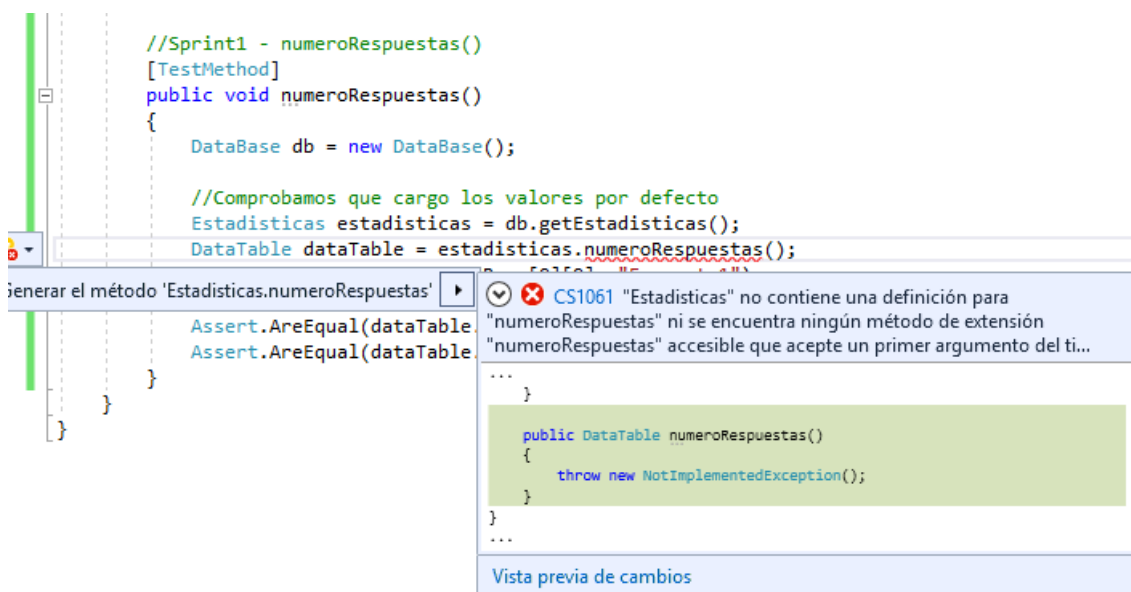
    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.numeroRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 4);
}
```

Creado: domingo, 02 de diciembre de 2018, 13:35:49

Como podemos apreciar en la captura anterior, ya no nos sale en rojo ni la clase Estadísticas, ni el método de la base de datos que nos devuelve las estadísticas, esto se debe a que ya lo hemos creado en el paso anterior.

Sin embargo, vemos que sí que nos aparece en rojo el `numeroRespuestas()`, ya que este método no está creado en la clase Estadísticas.

Para ello hacemos lo mismo, damos botón derecho y generamos el código correspondiente.



```
public DataTable numeroRespuestas()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 13:36:55

Una vez que se nos genera el código por defecto, introducimos nosotros el código necesario para que pase el test.

```
public DataTable numeroRespuestas()
{
    DataTable dataTable = new DataTable();

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach(Encuesta e in encuestas)
    {
        dataTable.Rows.Add(e.Nombre, e.getOpiniones().Count);
    }

    return dataTable;
}
```

Creado: domingo, 02 de diciembre de 2018, 13:39:52


Como ya comenté en el caso anterior, para los dos primeros métodos al principio solo hicimos una pequeña captura del test (porque no sabíamos si ponerte toda la imagen del test o no), por lo que después cambiamos de idea y sacamos una nueva captura del test entero, es por eso que tienen distinta fecha.

```
//Sprint1 - numeroRespuestas()
[TestMethod]
public void numeroRespuestas()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.numeroRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 4);
    //Comprobamos que al añadir una nueva opinión se cambia
    db.getEncuesta("Encuesta1").setOpinion(3);
    estadisticas = db.getEstadisticas(); ;
    dataTable = estadisticas.numeroRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 4);
    //Comprobamos que al borrar una encuesta ya no existe ni ella ni sus opciones
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas(); ;
    dataTable = estadisticas.numeroRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta2");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[8][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[8][1], 4);
    //Comprobamos que al añadir una encuesta sus opciones son 0
    db.addEncuesta("Encuesta1", "Encuesta1descripción");
    estadisticas = db.getEstadisticas(); ;
    dataTable = estadisticas.numeroRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta2");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[9][1], 0);
}

Creado: domingo, 02 de diciembre de 2018, 13:48:47
```

Como podemos observar en la siguiente captura, al ejecutar el test nos lo indica que está verde, es decir, que pasa el test.

 **numeroRespuestas**

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python



en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

numeroEncuestas

Al igual que sucedía en los casos anterior, primero tenemos que hacer el test, para seguir con la metodología TDD.

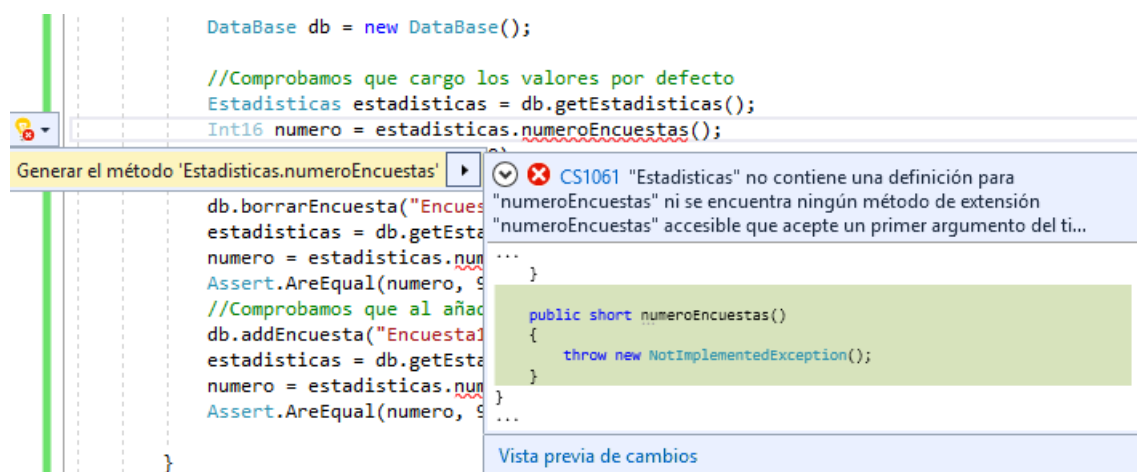
```
//Sprint1 - numeroEncuestas()
[TestMethod]
public void numeroEncuestas()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    Int16 numero = estadisticas.numeroEncuestas();
    Assert.AreEqual(numero, 10);
    //Comprobamos que si borro una encuesta el número se reduce
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    numero = estadisticas.numeroEncuestas();
    Assert.AreEqual(numero, 9);
    //Comprobamos que al añadir una encuesta sin opciones el número sigue siendo 9
    db.addEncuesta("Encuesta1", "Encuestaldescripción");
    estadisticas = db.getEstadisticas();
    numero = estadisticas.numeroEncuestas();
    Assert.AreEqual(numero, 9);
}
```

Creado: domingo, 02 de diciembre de 2018, 13:56:31

Como podemos apreciar en la captura anterior nos muestra en rojo el método, ya que todavía no está creado en la clase Estadísticas.

Para ello pulsamos botón derecho y generamos el código por defecto.




```
public short numeroEncuestas()
{
    throw new NotImplementedException();
}
Creado: domingo, 02 de diciembre de 2018, 13:57:56
```

Una vez que ya tenemos el código, podemos introducir el nuestro para así poder ejecutar los test de forma correcta.

```
public short numeroEncuestas()
{
    short numero = 0;

    foreach(Encuesta e in encuestas)
    {
        if(e.getOpiniones().Count != 0)
        {
            ++numero;
        }
    }
    return numero;
}
Creado: domingo, 02 de diciembre de 2018, 14:00:11
```

Una vez que ya hemos hecho el código, podemos ejecutar el test para comprobar que es correcto, tal y como podemos observar en la imagen pasa los test.

✓ numeroEncuestas

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

rankingEncuestasPorRespuestas

Al igual que sucedía en los casos anteriores, primero hacemos el test para así seguir con la metodología TDD.

```
//Sprint1 - rankingEncuestasPorRespuestas()
[TestMethod]
public void TestRankingEncuestasPorRespuestas()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.rankingEncuestasPorRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 4);
    //Comprobamos que al añadir una encuesta sin opciones está en la última posición
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 0);
    //Comprobamos que al añadir 5 opciones pasa a la primera posición
    db.getEncuesta("Encuesta11").setOpinion(3);
    db.getEncuesta("Encuesta11").setOpinion(2);
    db.getEncuesta("Encuesta11").setOpinion(1);
    db.getEncuesta("Encuesta11").setOpinion(4);
    db.getEncuesta("Encuesta11").setOpinion(3);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
}
```

Creado: domingo, 02 de diciembre de 2018, 14:12:56

Como podemos observar en la captura anterior, el método está en rojo, por lo que pulsamos botón derecho y generamos el código por defecto.

```
[TestMethod]
public void rankingEncuestasPorRespuestas()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.rankingEncuestasPorRespuestas();

    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[9][1], 5);
    //Comprobamos que al añadir una encuesta
    db.addEncuesta("Encuesta11", "Encuesta11");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorRespuestas();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 5);
    //Comprobamos que al añadir 5 opiniones
    db.getEncuesta("Encuesta11").setOpinion(3);
}
```

Generar el método 'Estadisticas.rankingEncuestasPorRespuestas'

CS1061 "Estadisticas" no contiene una definición para "rankingEncuestasPorRespuestas" ni se encuentra ningún método de extensión "rankingEncuestasPorRespuestas" accesible que acepte un prim...

```
public DataTable rankingEncuestasPorRespuestas()
{
    throw new NotImplementedException();
}
```

Vista previa de cambios

```
public DataTable rankingEncuestasPorRespuestas()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 14:13:57

Una vez que está generado el código, podemos introducir el nuestro para que así pasen los test:

```
public DataTable rankingEncuestasPorRespuestas()
{
    DataTable dataTable = new DataTable();
    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach (Encuesta e in encuestas)
    {
        dataTable.Rows.Add(e.Nombre, e.getOpiniones().Count);
    }

    DataView dv = dataTable.DefaultView;
    dv.Sort = "Numero desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 14:20:00

Una vez que hemos hecho el código, tenemos que probar si de verdad pasa el test, para ello ejecutamos la prueba correspondiente y como podemos observar en la siguiente captura lo pasa:

✓ rankingEncuestasPorRespuestas



En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

rankingEncuestasPorValoracion

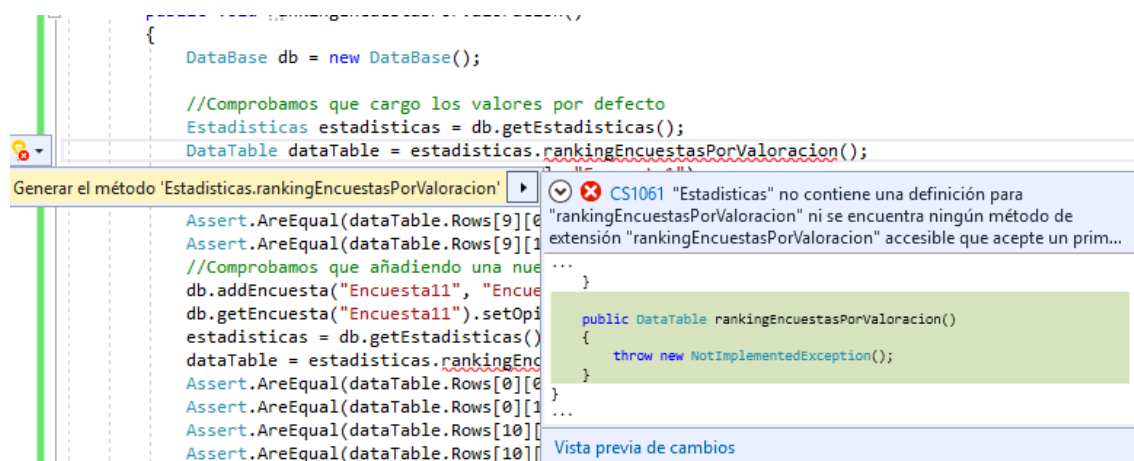
Lo primero que tenemos que hacer es generar el test:

```
//Sprint1 - rankingEncuestasPorValoración()
[TestMethod]
public void rankingEncuestasPorValoracion()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 3);
    //Comprobamos que añadiendo una nueva encuesta con valor 1, está la última
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    db.getEncuesta("Encuesta11").setOpinion(1);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 1);
    //Comprobamos que añadiendo la encuesta anterior con valor 4
    db.getEncuesta("Encuesta11").setOpinion(4);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[10][1], 3);
}
```

Creado: domingo, 02 de diciembre de 2018, 14:34:30

Como podemos observar nos sale en rojo el método, ya que éste no está creado, para crearlo damos botón derecho y generamos el código automáticamente.



```
public DataTable rankingEncuestasPorValoracion()
{
    throw new NotImplementedException();
}
Creado: domingo, 02 de diciembre de 2018, 14:35:15
```

Una vez generado el código, introducimos nosotros el nuestro para que pase el test:

```
public DataTable rankingEncuestasPorValoracion()
{
    DataTable dataTable = new DataTable();
    int nota_max = 1;
    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Nota", typeof(int));

    foreach (Encuesta e in encuestas)
    {
        nota_max = 1;
        foreach (Valoracion v in e.getOpiniones())
        {
            if(nota_max < v.Valor)
            {
                nota_max = v.Valor;
            }
        }
        dataTable.Rows.Add(e.Nombre, nota_max);
    }

    DataView dv = dataTable.DefaultView;
    dv.Sort = "Nota desc";
    return dv.ToTable();
}
Creado: domingo, 02 de diciembre de 2018, 14:42:55
```

Una vez generado el código, podemos ejecutar el test, como bien te mencionamos el último día en prácticas, para algún método tuvimos que corregir el test inicial, ya que al hacer a mano las cuentas te puedes equivocar, el test corregido sería el siguiente:

```
//Sprint1 - rankingEncuestasPorValoración()
[TestMethod]
public void rankingEncuestasPorValoracion()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta2");
    Assert.AreEqual(dataTable.Rows[9][1], 2);
    //Comprobamos que añadiendo una nueva encuesta con valor 1, está la última
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    db.getEncuesta("Encuesta11").setOpinion(1);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 1);
    //Comprobamos que añadiendo la encuesta anterior con valor 4
    db.getEncuesta("Encuesta11").setOpinion(4);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.rankingEncuestasPorValoracion();
    Assert.AreEqual(dataTable.Rows[4][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[4][1], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta2");
    Assert.AreEqual(dataTable.Rows[10][1], 2);
}
```

Creado: domingo, 02 de diciembre de 2018, 14:56:06

Una vez corregido el test ya podemos ejecutar el test de forma satisfactoria.

✓ rankingEncuestasPorValoracion

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no



podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

Media

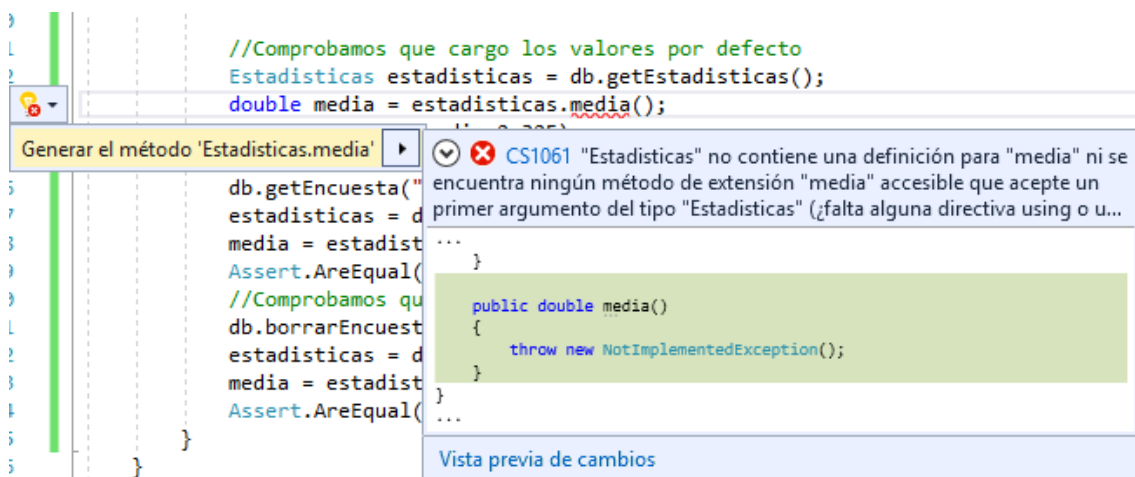
Continuando con la metodología TDD, lo primero que hacemos es hacer el test.

```
//Sprint1 - media()
[TestMethod]
public void media()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    double media = estadisticas.media();
    Assert.AreEqual(media, 2.325);
    //Comprobamos que si añadimos una nueva respuesta con un valor de 4 aumenta la media
    db.getEncuesta("Encuesta1").setOpinion(4);
    estadisticas = db.getEstadisticas();
    media = estadisticas.media();
    Assert.AreEqual(media, 2.36585365853659);
    //Comprobamos que al borrar una encuesta con respuestas se reduce la media
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    media = estadisticas.media();
    Assert.AreEqual(media, 2.27777777777778);
}
```

Creado: domingo, 02 de diciembre de 2018, 15:21:25

Al igual que pasa en los casos anteriores, nos da un error ya que todavía no está creado el método. Para ello lo generemos gracias a Visual.



```
public double media()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 15:22:03

Una vez generado el método, introducimos el código correspondiente:

```
public double media()
{
    double media = 0;

    foreach(Valoracion v in valoraciones)
    {
        media += v.Valor;
    }

    return (double)media / (double)valoraciones.Count;
}
```

Creado: domingo, 02 de diciembre de 2018, 15:33:28

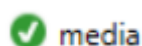
Finalmente, podemos ejecutar el test. Sin embargo, al igual que nos sucedía en el caso anterior, nos fallaba el test, en este caso se debe a que Visual maneja más decimales que Excel, es por ello por lo que tuvimos que modificar el test.

```
//Sprint1 - media()
[TestMethod]
public void media()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    double media = estadisticas.media();
    Assert.AreEqual(media, 2.325);
    //Comprobamos que si añadimos una nueva respuesta con un valor de 4 aumenta la media
    db.getEncuesta("Encuesta1").setOpinion(4);
    estadisticas = db.getEstadisticas();
    media = estadisticas.media();
    Assert.AreEqual(media, (double)97/(double)41);
    //Comprobamos que al borrar una encuesta con respuestas se reduce la media
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    media = estadisticas.media();
    Assert.AreEqual(media, (double)82/(double)36);
}
```

Creado: domingo, 02 de diciembre de 2018, 15:33:54

Una vez modificar el test, ya podemos ejecutar, y como podemos observar en la siguiente captura, no nos da ningún problema.



En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:



```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

Mediana

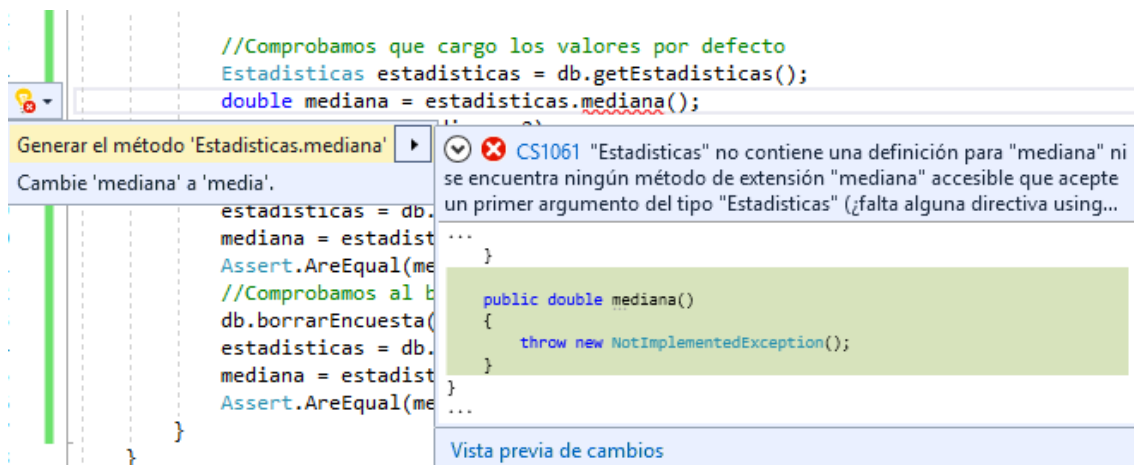
Siguiendo con la metodología TDD, primero realizamos el test.

```
//Sprint1 - mediana()
[TestMethod]
public void mediana()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    double mediana = estadisticas.mediana();
    Assert.AreEqual(mediana, 2);
    //Comprobamos al añadir una nueva respuesta con un valor de 4
    db.getEncuesta("Encuesta1").setOpinion(4);
    estadisticas = db.getEstadisticas();
    mediana = estadisticas.mediana();
    Assert.AreEqual(mediana, 2);
    //Comprobamos al borrar una encuesta con respuestas
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    mediana = estadisticas.mediana();
    Assert.AreEqual(mediana, 2);
}
```

Creado: domingo, 02 de diciembre de 2018, 15:43:23

Como podemos observar en la imagen anterior, nos da un error ya que el método no está creado, para solucionarlo generamos el código.



```
//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
double mediana = estadisticas.mediana();

estadisticas = db.getEstadisticas();
mediana = estadisticas.mediana();
Assert.AreEqual(mediana, 2);

//Comprobamos al borrar una encuesta con respuestas
db.borrarEncuesta("Encuesta1");
estadisticas = db.getEstadisticas();
mediana = estadisticas.mediana();
Assert.AreEqual(mediana, 2);
```

```
public double mediana()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 15:44:38

Una vez que ya tenemos el código generado, escribimos nosotros el nuestro.

```
public double mediana()
{
    List<double> aux = new List<double>();

    foreach(Valoracion v in valoraciones)
    {
        aux.Add(v.Valor);
    }
    aux.Sort();
    return aux[aux.Count / 2];
}
```

Creado: domingo, 02 de diciembre de 2018, 15:49:30

La captura anterior fue el primer código que realizamos, aunque pasaba los test, el código estaba mal porque faltaba poner un -1 al índice, es decir, que devolviera una posición anterior ya que los arrays empiezan en 0. Por lo tanto, corregimos ese error y el código final es el siguiente:

```
public double mediana()
{
    List<double> aux = new List<double>();

    foreach(Valoracion v in valoraciones)
    {
        aux.Add((double)v.Valor);
    }
    aux.Sort();
    int indice = (int)Math.Round((double)aux.Count / (double)2) - 1;
    return aux[indice];
}
```

Creado: lunes, 03 de diciembre de 2018, 11:07:51

Finalmente, ejecutamos los test y los pasa como se muestra a continuación.

✓ mediana

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```



Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

Desvest

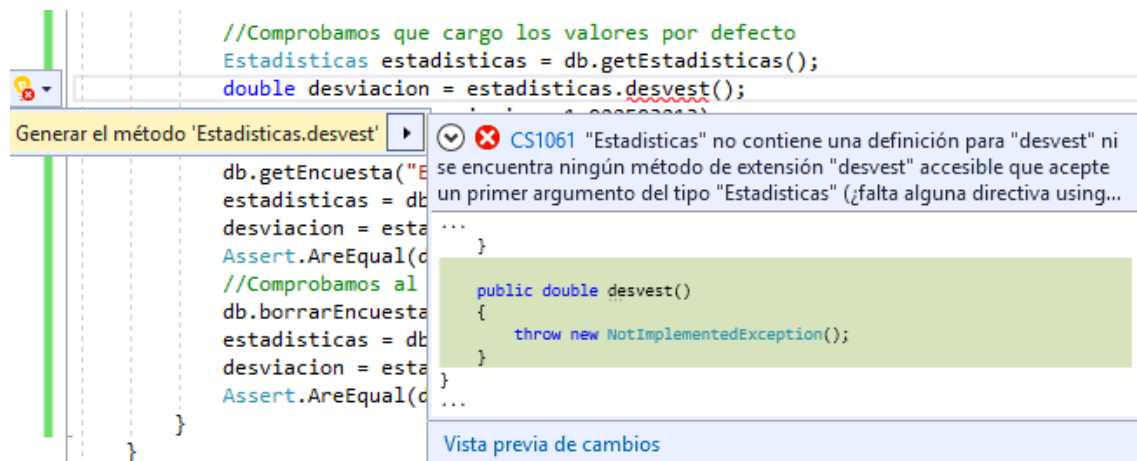
Lo primero que tenemos que hacer es realizar el test.

```
//Sprint1 - desviaciones()
[TestMethod]
public void desvest()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    double desviacion = estadisticas.desvest();
    Assert.AreEqual(desviacion, 1.022503213);
    //Comprobamos al añadir una nueva respuesta con un valor de 4
    db.getEncuesta("Encuesta1").setOpinion(4);
    estadisticas = db.getEstadisticas();
    desviacion = estadisticas.desvest();
    Assert.AreEqual(desviacion, 1.042978848);
    //Comprobamos al borrar una encuesta con respuestas
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    desviacion = estadisticas.desvest();
    Assert.AreEqual(desviacion, 1.00316958);
}
```

Creado: domingo, 02 de diciembre de 2018, 16:06:52

Como podemos apreciar en la imagen, nos da un error ya que el método no está creado.



```
public double desvest()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 16:07:39

Una vez generado el código, ya podemos introducir el nuestro:

```
public double desvest()  
{  
    double datos = 0;  
    foreach(Valoracion v in valoraciones)  
    {  
        datos += Math.Pow((double)v.Valor-(double)media(), 2);  
    }  
    return (double)Math.Sqrt(datos / (valoraciones.Count - 1));  
}
```

Creado: domingo, 02 de diciembre de 2018, 16:13:28

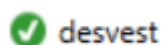
Finalmente, al ejecutar el test vimos que nos daba error, y nos dimos cuenta que se debe a que C# maneja más decimales que los que nos muestra, por lo tanto, para solucionarlo lo que hacemos es redondear el número.

Para ello tuvimos que modificar el test original.

```
//Sprint1 - desviaciones()  
[TestMethod]  
public void desvest()  
{  
    DataBase db = new DataBase();  
  
    //Comprobamos que cargo los valores por defecto  
    Estadisticas estadisticas = db.getEstadisticas();  
    double desviacion = estadisticas.desvest();  
    Assert.AreEqual(Math.Round(desviacion), Math.Round(1.02250321295966));  
    //Comprobamos al añadir una nueva respuesta con un valor de 4  
    db.getEncuesta("Encuesta1").setOpinion(4);  
    estadisticas = db.getEstadisticas();  
    desviacion = estadisticas.desvest();  
    Assert.AreEqual(Math.Round(desviacion), Math.Round(1.04297884832281));  
    //Comprobamos al borrar una encuesta con respuestas  
    db.borrarEncuesta("Encuesta1");  
    estadisticas = db.getEstadisticas();  
    desviacion = estadisticas.desvest();  
    Assert.AreEqual(Math.Round(desviacion), Math.Round(1.00316958005574));  
}
```

Creado: domingo, 02 de diciembre de 2018, 16:25:39

Una vez realizada la modificación, ejecutamos los test y los pasa perfectamente.



En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:



```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

numRespRangosPorEncuesta

Para continuar con la metodología TDD, primero tenemos que hacer las pruebas.

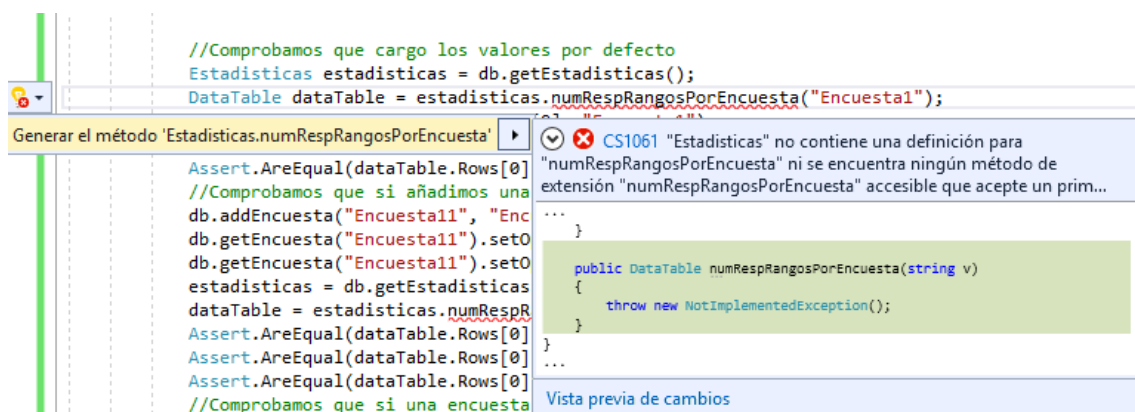
```
//Sprint1 - numRespRangosPorEncuesta()
[TestMethod]
public void numRespRangosPorEncuesta()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.numRespRangosPorEncuesta("Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    //Comprobamos que si añadimos una encuesta con dos opciones, son correctos
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    db.getEncuesta("Encuesta11").setOpinion(2);
    db.getEncuesta("Encuesta11").setOpinion(3);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangosPorEncuesta("Encuesta11");
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[0][1], 2);
    Assert.AreEqual(dataTable.Rows[0][2], 3);
    //Comprobamos que si una encuesta no tiene opciones tiene valores 0
    db.addEncuesta("Encuesta12", "Encuesta12descripción");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangosPorEncuesta("Encuesta12");
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta12");
    Assert.AreEqual(dataTable.Rows[0][1], 0);
    Assert.AreEqual(dataTable.Rows[0][2], 0);
}
```

Creado: domingo, 02 de diciembre de 2018, 16:42:14

Como podemos observar en la captura anterior, nos muestra un error ya que el método no está generado.

Para ello hacemos click con el botón derecho y generamos el código.



```
public DataTable numRespRangosPorEncuesta(string v)
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 16:43:25

Una vez generado el código, ya podemos introducir el nuestro.

```
public DataTable numRespRangosPorEncuesta(string nombre)
{
    DataTable dataTable = new DataTable();
    int minimo = 0;
    int maximo = 0;

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Minimo", typeof(int));
    dataTable.Columns.Add("Maximo", typeof(int));

    foreach(Encuesta e in encuestas)
    {
        if(e.Nombre == nombre && e.getOpiniones().Count != 0)
        {
            minimo = 5;
            maximo = 0;
            foreach(Valoracion v in e.getOpiniones())
            {
                if(maximo < v.Valor)
                {
                    maximo = v.Valor;
                }
                if(minimo > v.Valor)
                {
                    minimo = v.Valor;
                }
            }
            dataTable.Rows.Add(e.Nombre, minimo, maximo);
            break;
        }
        if(e.Nombre == nombre && e.getOpiniones().Count == 0)
        {
            dataTable.Rows.Add(e.Nombre, 0, 0);
            break;
        }
    }
    return dataTable;
}
```

Creado: domingo, 02 de diciembre de 2018, 17:00:00

Finalmente, cuando ya hemos hecho el código podemos ejecutar el test, y como podemos observar en la siguiente imagen no hay ningún problema.

✓ numRespRangosPorEncuesta

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:



```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.



numRespRangos

Continuando con la metodología TDD, primero hacemos el test.

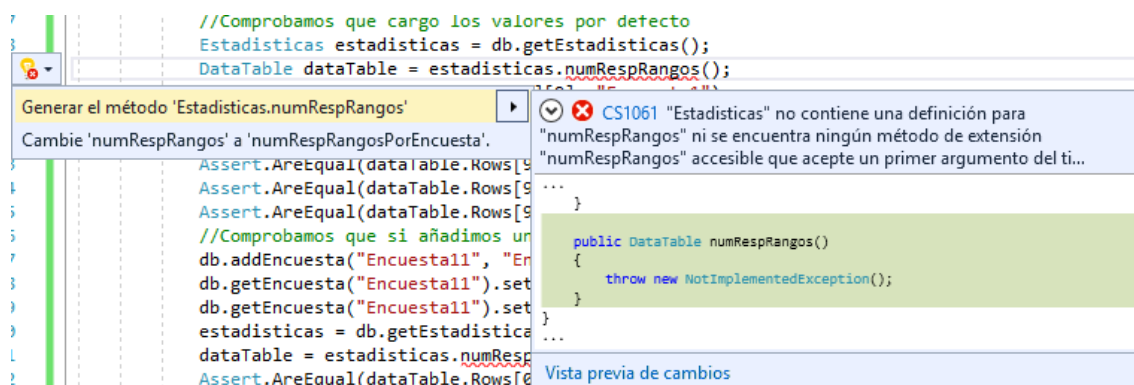
```
//Sprint1 - numRespRangos()
[TestMethod]
public void numRespRangos()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 1);
    Assert.AreEqual(dataTable.Rows[9][2], 3);
    //Comprobamos que si añadimos una encuesta con dos opiniones, son correctos
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    db.getEncuesta("Encuesta11").setOpinion(2);
    db.getEncuesta("Encuesta11").setOpinion(3);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 2);
    Assert.AreEqual(dataTable.Rows[10][2], 3);
    //Comprobamos que si una encuesta no tiene opiniones tiene valores 0
    db.addEncuesta("Encuesta12", "Encuesta12descripción");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[12][0], "Encuesta12");
    Assert.AreEqual(dataTable.Rows[12][1], 0);
    Assert.AreEqual(dataTable.Rows[12][2], 0);
}
```

Creado:

domingo, 02 de diciembre de 2018, 17:25:36

Al igual que sucedía anteriormente, nos da un error porque el método todavía no está creado.



```
public DataTable numRespRangos()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 17:28:19

Una vez generado el código, introducimos el nuestro:

```
public DataTable numRespRangos()
{
    DataTable dataTable = new DataTable();

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Minimo", typeof(int));
    dataTable.Columns.Add("Maximo", typeof(int));

    foreach (Encuesta e in encuestas)
    {
        DataTable aux = numRespRangosPorEncuesta(e.Nombre);
        dataTable.Rows.Add(aux.Rows[0][0], aux.Rows[0][1], aux.Rows[0][2]);
    }

    return dataTable;
}
```

Creado: domingo, 02 de diciembre de 2018, 17:33:10

Finalmente, cuando ya tenemos el código podemos ejecutar el test, al principio nos dio un error porque en vez de usar el índice 12 teníamos que usar el índice 11 ya que los arrays empiezan en 0.

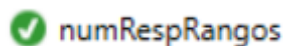
```
//Sprint1 - numRespRangos()
[TestMethod]
public void numRespRangos()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[9][0], "Encuesta10");
    Assert.AreEqual(dataTable.Rows[9][1], 1);
    Assert.AreEqual(dataTable.Rows[9][2], 3);
    //Comprobamos que si añadimos una encuesta con dos opiniones, son correctos
    db.addEncuesta("Encuesta11", "Encuesta11descripción");
    db.getEncuesta("Encuesta11").setOpinion(2);
    db.getEncuesta("Encuesta11").setOpinion(3);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[10][0], "Encuesta11");
    Assert.AreEqual(dataTable.Rows[10][1], 2);
    Assert.AreEqual(dataTable.Rows[10][2], 3);

    //Comprobamos que si una encuesta no tiene opciones tiene valores 0
    db.addEncuesta("Encuesta12", "Encuesta12descripción");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.numRespRangos();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[0][2], 4);
    Assert.AreEqual(dataTable.Rows[11][0], "Encuesta12");
    Assert.AreEqual(dataTable.Rows[11][1], 0);
    Assert.AreEqual(dataTable.Rows[11][2], 0);
}
```

Creado: domingo, 02 de diciembre de 2018, 17:36:17

Como podemos observar en la imagen no tenemos ningún problema al ejecutar el test una vez que hemos solucionado ese error.



En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:


```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

En este caso tenemos que tener en cuenta que para usar este método llamamos al anterior, es decir, no generamos de primeras código duplicado, por lo que no tuvimos que refactorizar.

Sprint2

respuestasPorAños

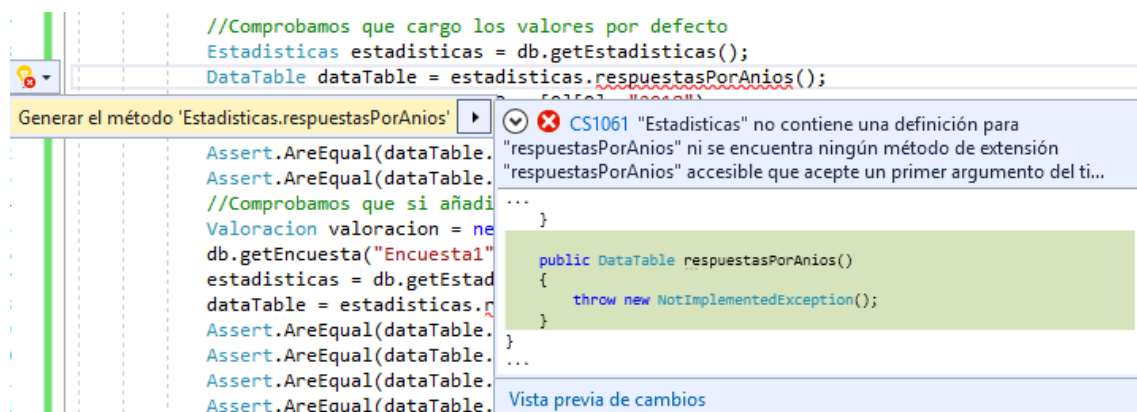
Lo primero que debemos hacer es realizar el código del test.

```
//Sprint2 - respuestasPorAños()
[TestMethod]
public void respuestasPorAños()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.respuestasPorAños();
    Assert.AreEqual(dataTable.Rows[0][0], "2018");
    Assert.AreEqual(dataTable.Rows[0][1], 30);
    Assert.AreEqual(dataTable.Rows[1][0], "2017");
    Assert.AreEqual(dataTable.Rows[1][1], 10);
    //Comprobamos que si añadimos una respuesta del 2016 nos lo contabiliza
    Valoracion valoracion = new Valoracion(4, "muy buena", new DateTime(2016, 12, 12, 14, 30, 15));
    db.getEncuesta("Encuesta1").setOpinionCSV(valoracion);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.respuestasPorAños();
    Assert.AreEqual(dataTable.Rows[0][0], "2018");
    Assert.AreEqual(dataTable.Rows[0][1], 30);
    Assert.AreEqual(dataTable.Rows[1][0], "2017");
    Assert.AreEqual(dataTable.Rows[1][1], 10);
    Assert.AreEqual(dataTable.Rows[2][0], "2016");
    Assert.AreEqual(dataTable.Rows[2][1], 1);
}
```

Creado: domingo, 02 de diciembre de 2018, 18:25:41

Como podemos observar en la imagen, nos da un error ya que no está el método correspondiente. Por lo que generamos el código como en los casos anteriores.



```
public DataTable respuestasPorAños()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 18:26:27

Una vez que hemos generado el código, podemos introducir el nuestro para así poder ejecutar el test:


```
public DataTable respuestasPorAnios()
{
    DataTable dataTable = new DataTable();
    Dictionary<string, int> diccionario = new Dictionary<string, int>();

    dataTable.Columns.Add("Anio", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach(Valoracion v in valoraciones)
    {
        string clave = v.Fecha.Year.ToString();
        if (diccionario.ContainsKey(clave))
        {
            int valor = diccionario[clave];
            diccionario[clave] = ++valor;
        }
        else
        {
            diccionario.Add(clave, 1);
        }
    }
    foreach(string clave in diccionario.Keys)
    {
        dataTable.Rows.Add(clave, diccionario[clave]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Numero desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 18:33:46

Una vez que tenemos el código, ya podemos ejecutar el test, y como podemos observar en la siguiente captura no tenemos ningún problema:

 respuestasPorAnios

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no



podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

respuestasPorMeses

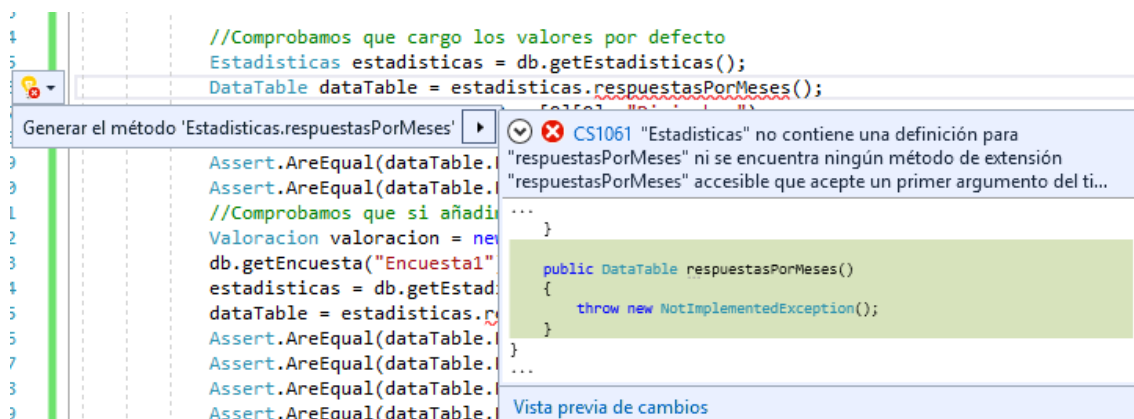
Para continuar con la metodología TDD, lo primero que tenemos que hacer es generar el test.

```
//Sprint2 - respuestasPorAnios()
[TestMethod]
public void respuestasPorMeses()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.respuestasPorMeses();
    Assert.AreEqual(dataTable.Rows[0][0], "Diciembre");
    Assert.AreEqual(dataTable.Rows[0][1], 1);
    Assert.AreEqual(dataTable.Rows[8][0], "Enero");
    Assert.AreEqual(dataTable.Rows[8][1], 6);
    //Comprobamos que si añadimos una respuesta de diciembre nos los contabiliza
    Valoracion valoracion = new Valoracion(4, "muy buena", new DateTime(2016, 12, 12, 14, 30, 15));
    db.getEncuesta("Encuesta1").setOpinionCSV(valoracion);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.respuestasPorMeses();
    Assert.AreEqual(dataTable.Rows[0][0], "Diciembre");
    Assert.AreEqual(dataTable.Rows[0][1], 2);
    Assert.AreEqual(dataTable.Rows[8][0], "Enero");
    Assert.AreEqual(dataTable.Rows[8][1], 6);
}
```

Creado: domingo, 02 de diciembre de 2018, 18:43:42

Como podemos apreciar en la imagen, nos da un error ya que el método no está creado, por lo que para solucionarlo generamos el código con la ayuda de Visual.



```
public DataTable respuestasPorMeses()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 18:44:33

Después, introducimos nuestro código en el método para que así funcione los test:

```
public DataTable respuestasPorMeses()
{
    DataTable dataTable = new DataTable();
    Dictionary<string, int> diccionario = new Dictionary<string, int>();

    dataTable.Columns.Add("Mes", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach(Valoracion v in valoraciones)
    {
        string clave = mes(v.Fecha.Month.ToString());
        if (diccionario.ContainsKey(clave))
        {
            int valor = diccionario[clave];
            diccionario[clave] = ++valor;
        }
        else
        {
            diccionario.Add(clave, 1);
        }
    }
    foreach(string clave in diccionario.Keys)
    {
        dataTable.Rows.Add(clave, diccionario[clave]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Numero desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 18:46:13

Una vez que ya tenemos el código, podemos ejecutar el test, pero esta vez no dio error porque en vez de Enero era Febrero, esto pasa cuando calculas las cosas a mano, el test solucionado es el siguiente:

```
//Sprint2 - respuestasPorAnios()
[TestMethod]
public void respuestasPorMeses()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.respuestasPorMeses();
    Assert.AreEqual(dataTable.Rows[0][0], "Febrero");
    Assert.AreEqual(dataTable.Rows[0][1], 7);
    Assert.AreEqual(dataTable.Rows[8][0], "Diciembre");
    Assert.AreEqual(dataTable.Rows[8][1], 1);
    //Comprobamos que si añadimos una respuesta de diciembre nos los contabiliza
    Valoracion valoracion = new Valoracion(4, "muy buena", new DateTime(2016, 12, 12, 14, 30, 15));
    db.getEncuesta("Encuesta1").setOpinionCSV(valoracion);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.respuestasPorMeses();
    Assert.AreEqual(dataTable.Rows[0][0], "Febrero");
    Assert.AreEqual(dataTable.Rows[0][1], 7);
    Assert.AreEqual(dataTable.Rows[8][0], "Diciembre");
    Assert.AreEqual(dataTable.Rows[8][1], 2);
}
```



Creado: domingo, 02 de diciembre de 2018, 19:07:49

Una vez corregido el test podemos ejecutar, y vemos que no nos da ningún problema:

✓ respuestasPorMeses

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();  
  
//Comprobamos que cargo los valores por defecto  
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

En este caso y en respuestasPorAnios, respuestasPorHoras, respuestasPorMeses, respuestasPorSemanas, se podría refactorizar estas líneas, ya que en todos estos métodos hacemos lo mismo:

```
foreach(string clave in diccionario.Keys)  
{  
    dataTable.Rows.Add(clave, diccionario[clave]);  
}  
DataView dv = dataTable.DefaultView;  
dv.Sort = "Numero desc";
```

Pero no sale rentable, ya que tardas más en llamar a una función a la cual la tenemos que pasar el diccionario y hacer lo que hay en la captura anterior, que hacer esa parte en todos los métodos.

respuestasPorSemanas

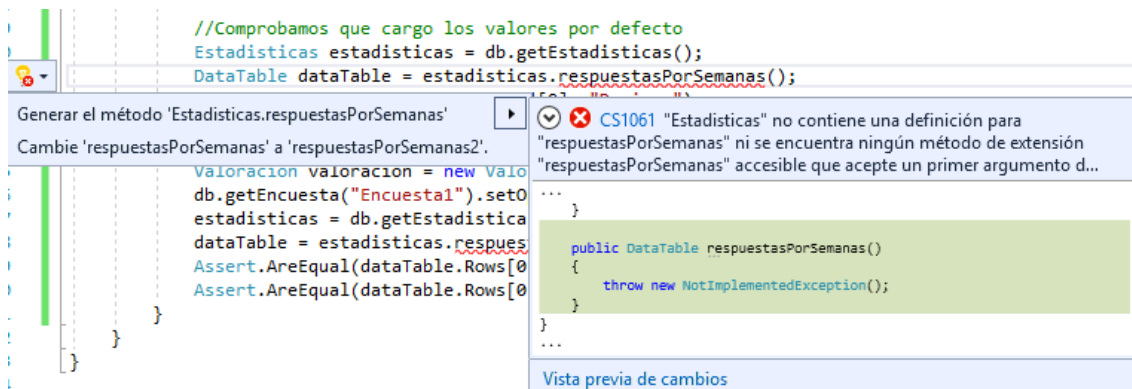
Para seguir con la metodología TDD, lo primero que tenemos que hacer es hacer el test.

```
//Sprint2 - respuestasPorSemanas()
[TestMethod]
public void respuestasPorSemanas()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.respuestasPorSemanas();
    Assert.AreEqual(dataTable.Rows[0][0], "Domingo");
    Assert.AreEqual(dataTable.Rows[0][1], 11);
    //Comprobamos que si añadimos una respuesta de lunes nos los contabiliza
    Valoracion valoracion = new Valoracion(4, "muy buena", new DateTime(2018, 12, 2, 14, 30, 15));
    db.getEncuesta("Encuesta1").setOpinionCSV(valoracion);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.respuestasPorSemanas();
    Assert.AreEqual(dataTable.Rows[0][0], "Domingo");
    Assert.AreEqual(dataTable.Rows[0][1], 12);
}
```

Creado: domingo, 02 de diciembre de 2018, 19:31:18

Como podemos observar en la captura anterior, nos da error porque el método todavía no está creado.



```
public DataTable respuestasPorSemanas()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 19:33:07

Una vez que hemos generado el código, podemos introducir nosotros el nuestro para que pase el test:



```
public DataTable respuestasPorSemanas()
{
    DataTable dataTable = new DataTable();
    Dictionary<string, int> diccionario = new Dictionary<string, int>();

    dataTable.Columns.Add("Semana", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach (Valoracion v in valoraciones)
    {
        string clave = dias(v.Fecha.DayOfWeek.ToString());
        if (diccionario.ContainsKey(clave))
        {
            int valor = diccionario[clave];
            diccionario[clave] = ++valor;
        }
        else
        {
            diccionario.Add(clave, 1);
        }
    }
    foreach (string clave in diccionario.Keys)
    {
        dataTable.Rows.Add(clave, diccionario[clave]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Numero desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 19:38:40

Finalmente, podemos ejecutar el test y como podemos ver no tenemos ningún problema:

 respuestasPorSemanas

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no

podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

En este caso y en `respuestasPorAnios`, `respuestasPorHoras`, `respuestasPorMeses`, `respuestasPorSemanas`, se podría refactorizar estas líneas, ya que en todos estos métodos hacemos lo mismo:

```
foreach(string clave in diccionario.Keys)
{
    dataTable.Rows.Add(clave, diccionario[clave]);
}
DataView dv = dataTable.DefaultView;
dv.Sort = "Numero desc";
```

Pero no sale rentable, ya que tardas más en llamar a una función a la cual la tenemos que pasar el diccionario y hacer lo que hay en la captura anterior, que hacer esa parte en todos los métodos.

respuestasPorHoras

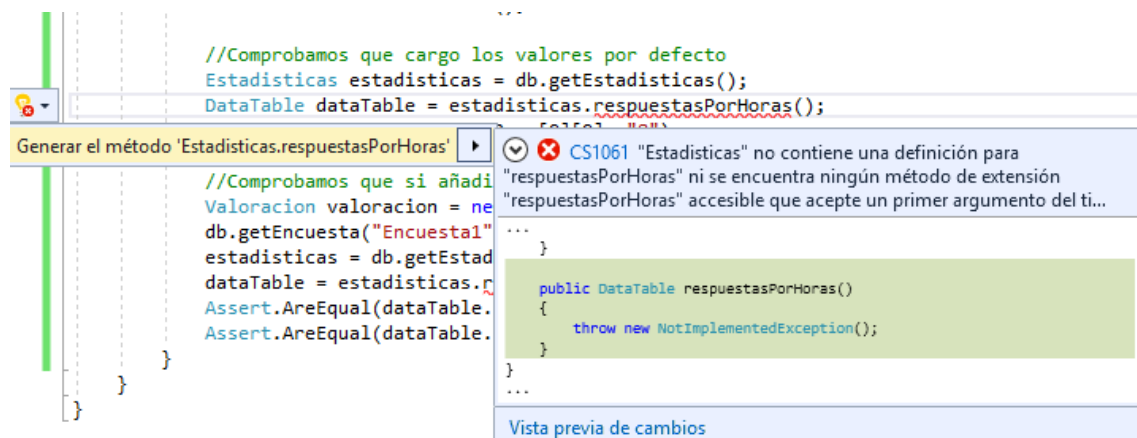
Lo primero que tenemos que hacer es el test, para así seguir con la metodología TDD.

```
//Sprint2 - respuestasPorHoras()
[TestMethod]
public void respuestasPorHoras()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.respuestasPorHoras();
    Assert.AreEqual(dataTable.Rows[0][0], "8");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    //Comprobamos que si añadimos una respuesta de lunes nos los contabiliza
    Valoracion valoracion = new Valoracion(4, "muy buena", new DateTime(2018, 12, 2, 8, 30, 15));
    db.getEncuesta("Encuesta1").setOpinionCSV(valoracion);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.respuestasPorHoras();
    Assert.AreEqual(dataTable.Rows[0][0], "8");
    Assert.AreEqual(dataTable.Rows[0][1], 5);
}
```

Creado: domingo, 02 de diciembre de 2018, 19:42:45

Luego lo que tenemos que hacer es generar el código del método que nos da error.



```
public DataTable respuestasPorHoras()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 19:43:27

Una vez que ya tenemos el código generado introducimos el nuestro para que así pase el test:

```
public DataTable respuestasPorHoras()
{
    DataTable dataTable = new DataTable();
    Dictionary<string, int> diccionario = new Dictionary<string, int>();

    dataTable.Columns.Add("Hora", typeof(string));
    dataTable.Columns.Add("Numero", typeof(int));

    foreach (Valoracion v in valoraciones)
    {
        string clave = v.Fecha.Hour.ToString();
        if (diccionario.ContainsKey(clave))
        {
            int valor = diccionario[clave];
            diccionario[clave] = ++valor;
        }
        else
        {
            diccionario.Add(clave, 1);
        }
    }
    foreach (string clave in diccionario.Keys)
    {
        dataTable.Rows.Add(clave, diccionario[clave]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Numero desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 19:44:50

Finalmente, cuando ya tenemos el código podemos ejecutar las pruebas y como podemos ver no nos da ningún problema:

✓ respuestasPorHoras

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada



método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

En este caso y en respuestasPorAnios, respuestasPorHoras, respuestasPorMeses, respuestasPorSemanas, se podría refactorizar estas líneas, ya que en todos estos métodos hacemos lo mismo:

```
foreach(string clave in diccionario.Keys)
{
    dataTable.Rows.Add(clave, diccionario[clave]);
}
DataView dv = dataTable.DefaultView;
dv.Sort = "Numero desc";
```

Pero no sale rentable, ya que tardas más en llamar a una función a la cual la tenemos que pasar el diccionario y hacer lo que hay en la captura anterior, que hacer esa parte en todos los métodos.

mediaPorEncuesta

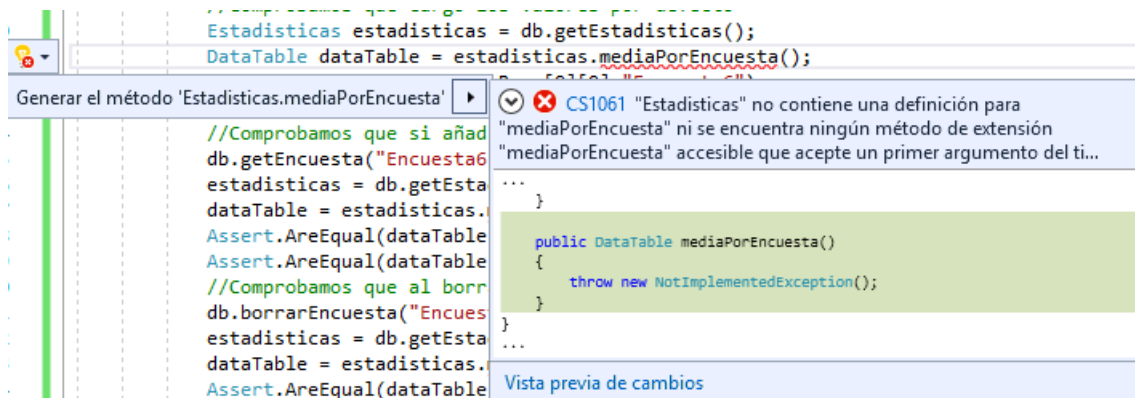
Lo primero que tenemos que hacer es el test, para así seguir con la metodología TDD.

```
//Sprint2 - mediaPorEncuesta()
[TestMethod]
public void mediaPorEncuesta()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.mediaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta6");
    Assert.AreEqual(dataTable.Rows[0][1], (double)15/(double)4);
    //Comprobamos que si añadimos una nueva respuesta con un valor de 4 aumenta la media
    db.getEncuesta("Encuesta6").setOpinion(4);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.mediaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta6");
    Assert.AreEqual(dataTable.Rows[0][1], (double)19 / (double)5);
    //Comprobamos que al borrar una encuesta con respuestas se reduce la media
    db.borrarEncuesta("Encuesta6");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.mediaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], (double)11 / (double)4);
}
```

Creado: domingo, 02 de diciembre de 2018, 20:07:49

Una vez que tenemos el test, ya podemos generar el código del método que nos da el error.



```
public DataTable mediaPorEncuesta()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 20:08:38

Después generamos el código para que así pasa el test:

```
public DataTable mediaPorEncuesta()
{
    DataTable dataTable = new DataTable();

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Media", typeof(double));

    foreach(Encuesta e in encuestas)
    {
        double media = 0;
        foreach(Valoracion v in e.getOpiniones())
        {
            media += v.Valor;
        }
        dataTable.Rows.Add(e.Nombre, (double)media / (double)e.getOpiniones().Count);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Media desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 20:13:35

Finalmente, ya podemos ejecutar el test, y como podemos observar en la siguiente imagen no tenemos ningún problema:

✓ mediaPorEncuesta

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

medianaPorEncuesta

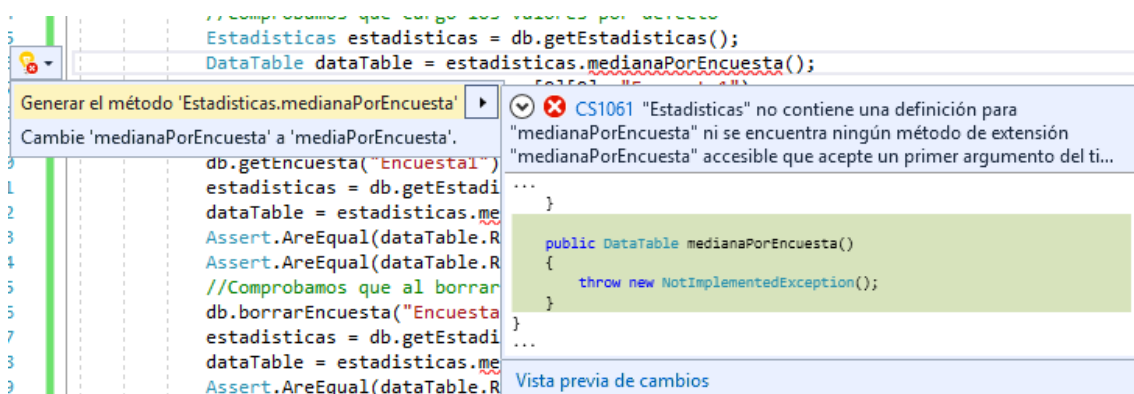
Para continuar con la metodología TDD, lo primero que tenemos que hacer es realizar el test.

```
//Sprint2 - medianaPorEncuesta()
[TestMethod]
public void medianaPorEncuesta()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    //Comprobamos que si añadimos una nueva respuesta con un valor de 4 aumenta la media
    db.getEncuesta("Encuesta1").setOpinion(1);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta6");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    //Comprobamos que al borrar una encuesta con respuestas se reduce la media
    db.borrarEncuesta("Encuesta6");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta3");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
}
```

Creado: domingo, 02 de diciembre de 2018, 20:20:20

El siguiente paso que tenemos que hacer es generar el código del método en rojo que hay en la captura anterior.



```
public DataTable medianaPorEncuesta()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 20:21:04

El siguiente paso es introducir el código en el método, para que nos pase el test:


```
public DataTable medianaPorEncuesta()
{
    DataTable dataTable = new DataTable();

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Mediana", typeof(int));

    foreach(Encuesta e in encuestas)
    {
        List<double> aux = new List<double>();
        foreach (Valoracion v in e.getOpiniones())
        {
            aux.Add(v.Valor);
        }
        aux.Sort();
        dataTable.Rows.Add(e.Nombre, aux[aux.Count / 2]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Mediana desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 20:26:29

Al día siguiente nos dimos cuenta de que este código está mal, ya que hay que hacer un -1 al índice, tal y como es el siguiente código:

```
public DataTable medianaPorEncuesta()
{
    DataTable dataTable = new DataTable();

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Mediana", typeof(int));

    foreach(Encuesta e in encuestas)
    {
        List<double> aux = new List<double>();
        foreach (Valoracion v in e.getOpiniones())
        {
            aux.Add(v.Valor);
        }
        aux.Sort();
        int indice = (int)Math.Round((double)aux.Count / (double)2) - 1;
        dataTable.Rows.Add(e.Nombre, aux[indice]);
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Mediana desc";
    return dv.ToTable();
}
```

Creado: lunes, 03 de diciembre de 2018, 11:10:00

Por lo tanto, tuvimos que corregir el test, y el nuevo test es el siguiente:

```
//Sprint2 - medianaPorEncuesta()
[TestMethod]
public void medianaPorEncuesta()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta6");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    //Comprobamos que si añadimos una nueva respuesta con un valor de 4 aumenta la media
    db.getEncuesta("Encuesta1").setOpinion(4);
    db.getEncuesta("Encuesta1").setOpinion(4);
    db.getEncuesta("Encuesta1").setOpinion(4);
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
    //Comprobamos que al borrar una encuesta con respuestas se reduce la media
    db.borrarEncuesta("Encuesta1");
    estadisticas = db.getEstadisticas();
    dataTable = estadisticas.medianaPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta6");
    Assert.AreEqual(dataTable.Rows[0][1], 4);
}
```

Creado: lunes, 03 de diciembre de 2018, 11:16:07

Una vez solucionado el problema, ya podemos ejecutar el test y como podemos observar no tenemos ningún problema:

✓ medianaPorEncuesta

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

desvEstPorEncuesta

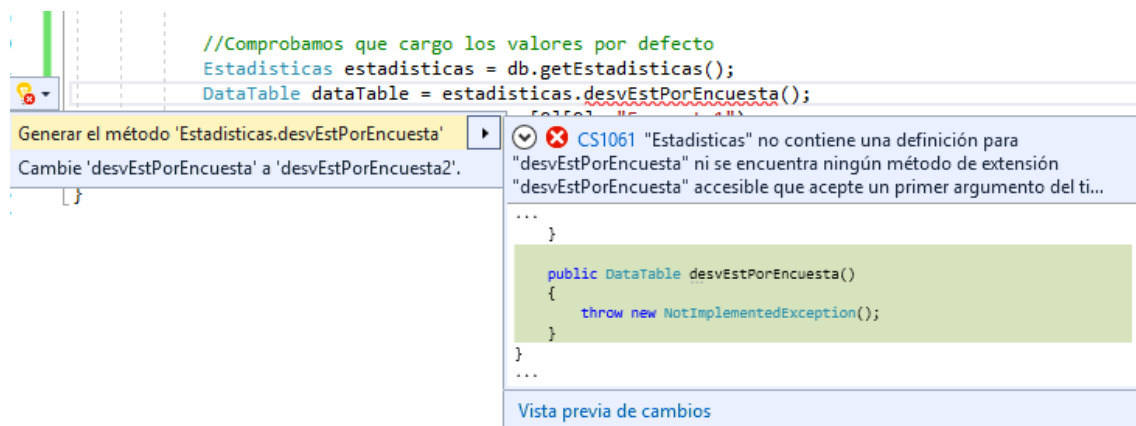
Lo primero que tenemos que hacer es hacer el test.

```
//Sprint2 - desvEstPorEncuesta()
[TestMethod]
public void desvEstPorEncuesta()
{
    DataBase db = new DataBase();

    //Comprobamos que cargo los valores por defecto
    Estadisticas estadisticas = db.getEstadisticas();
    DataTable dataTable = estadisticas.desvEstPorEncuesta();
    Assert.AreEqual(dataTable.Rows[0][0], "Encuesta1");
}
```

Creado: domingo, 02 de diciembre de 2018, 21:15:24

Como podemos apreciar en la imagen anterior, nos da error el método ya que no está creado, así que lo primero que debemos hacer es generar el método con la ayuda de Visual.



```
public DataTable desvEstPorEncuesta()
{
    throw new NotImplementedException();
}
```

Creado: domingo, 02 de diciembre de 2018, 21:16:24

Una vez generado el código, podemos introducir el nuestro para que así pase los test:

```
public DataTable desvEstPorEncuesta()
{
    DataTable dataTable = new DataTable();
    List<double> medias = new List<double>();
    double media = 0;
    int indice = -1;

    dataTable.Columns.Add("Encuesta", typeof(string));
    dataTable.Columns.Add("Desviacion", typeof(int));

    foreach(Encuesta e in encuestas)
    {
        media = 0;
        foreach(Valoracion v in e.getOpiniones())
        {
            media += v.Valor;
        }
        medias.Add((double)media / (double)e.getOpiniones().Count);
    }

    foreach (Encuesta e in encuestas)
    {
        double datos = 0;
        ++indice;
        foreach (Valoracion v in e.getOpiniones())
        {
            datos += Math.Pow((double)v.Valor - (double)medias[indice], 2);
        }
        dataTable.Rows.Add(e.Nombre, (double)Math.Sqrt(datos / e.getOpiniones().Count));
    }
    DataView dv = dataTable.DefaultView;
    dv.Sort = "Desviacion desc";
    return dv.ToTable();
}
```

Creado: domingo, 02 de diciembre de 2018, 21:22:23

Finalmente, podemos ejecutar el test y como podemos ver no hay ningún problema:

✓ desvEstPorEncuesta

En cuanto a la refactorización, en un principio consideramos refactorizar las dos primeras líneas del método, ya que como podrás observar luego se repiten en todos los métodos, a estas dos líneas nos referimos:

```
DataBase db = new DataBase();

//Comprobamos que cargo los valores por defecto
Estadisticas estadisticas = db.getEstadisticas();
```

Si sacamos esas dos líneas a un método, nos surge el problema de que en cada método necesitamos una instancia a la base de datos y otra a las estadísticas, por lo que no podemos agrupar esas dos líneas en un único método (ya que esto no es como Python en el que podemos devolver dos cosas a la vez). Por lo tanto, da igual dejar en cada



método estas dos líneas, que si hacemos un par de llamadas una para obtener la base de datos y otra para obtener las estadísticas, fin de cuentas sería lo mismo.

Por otro lado, el calcular las medias no sirve de mucho refactorizar, ya que generar un método solo para eso perdemos más tiempo en llamar a la función y pasarle la encuesta y que el método se encargue de generar para cada encuesta la media, que si hacemos dos bucles anidados, al fin y al cabo la complejidad va a seguir siendo la misma.



Conclusiones

Para la realización de esta práctica nos hemos basado principalmente en los links que hay en la bibliografía.

En cuanto a nuestras impresiones sobre esta práctica, ambos coincidimos. Aunque la práctica ha sido algo más fácil que la anterior, acostumbrarse al TDD no ha sido tan fácil, ya que por norma general cuando programamos generamos primero el código de la función y luego los test.

Un problema al que nos hemos tenido que enfrentar es que al hacer primero el test y luego el código, si cometíamos un pequeño error al calcular los valores para el test, teníamos que cambiar de nuevo el test, esto nos hizo perder algo más de tiempo que lo normal.

Otro problema que hemos tenido, es que programamos el doble, es decir, en vez de que uno se dedique a hacer los métodos y el otro las pruebas, nos hemos dividido el trabajo de tal forma que uno hizo el primer sprint (métodos y pruebas incluidas) y otro hizo el otro sprint, y cuando estás haciendo el código te das cuenta que estás tardando más de lo que esperábamos, a cambio es verdad que la aplicación es más robusta.

Una ventaja que hemos visto al usar la metodología TDD, es que al hacer primero el test tienes que saber muy bien cómo quieres que funcione el método, así cuando hacíamos el método tardábamos poco, porque ya teníamos pensado como hacerlo gracias al test.

Otra ventaja que hemos visto es que apenas se crea código redundante o innecesario, ya que cómo vas programando poco a poco y eso que programas lo pruebas, es bastante fácil poner solo lo que necesitas. Por lo que refactorizar se convierte en una tarea bastante sencilla.

También al realizar la práctica mediante TDD, cuando ya nos hemos acostumbrado a esta metodología, íbamos bastante rápido ya que vas probando todo poco a poco, lo cual se hace más fácil y reduce mucho tiempo.

Al igual que hemos comentado antes, al testear poco a poco es bastante sencillo la depuración, ya que antes de avanzar sabes por qué te da un problema y lo puedes solucionar antes de pasar a otra cosa.



Bibliografía

Enunciado de la práctica:

https://ubuvirtual.ubu.es/pluginfile.php/3772706/mod_resource/content/2/GI-NF-6376-2018-PR-PR2%20Dise%C3%B1o%20de%20una%20clase%20mediante%20TDD%20y%20test%20dirigidos%20por%20datos.pdf

Cómo funciona el DateTime:

<https://docs.microsoft.com/es-es/dotnet/api/system.datetime.today?view=netframework-4.7.2>

<https://docs.microsoft.com/es-es/dotnet/api/system.datetime.-ctor?view=netframework-4.7.2>

Cómo funciona el DataTable:

<https://docs.microsoft.com/es-es/dotnet/api/system.data.datatable?view=netframework-4.7.2>

<https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/dataset-datatable-dataview/creating-a-datatable>

<https://www.dotnetperls.com/datatable>

Cómo funciona el Dictionary:

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=netframework-4.7.2>

<https://docs.microsoft.com/es-es/dotnet/api/system.collections.generic.dictionary-2.containskey?view=netframework-4.7.2>