**Practica V: Herramientas colaborativas**



En esta práctica se va a poner a prueba vuestra capacidad de organización y trabajo en equipo, usando herramientas de trabajo colaborativo.

Esta práctica consiste en realizar, en equipos de 4-5 personas, las siguientes tareas:

* Creación de un proyecto en C# que realice todas las tareas que veis en el Anexo I. **Todas las tareas deben estar en la misma solución.** Habrá un programa en cada fichero que mostrará un menú general, que permitirá elegir la operación deseada.
* Crear una web, de por lo menos 3 páginas, usando css. La web ha de ser responsive. El product owner de la web será el profesor y el jefe de proyecto de otro grupo.
* Realizar todas las consultas asociadas con una base de datos de juegos de steam, con casi 13000 juegos . Ver Anexo II.

De las tres tareas, el PO será el profesor, excepto en la web, que será el profesor y el jefe de proyecto de otro grupo.

Lo que se va a hacer en esta práctica es crear una organización en la que estaréis los 5 miembros del grupo, y donde haréis el(los) repositorio(s).

Realizaremos un **fork**, es decir, crear una copia del proyecto, en nuestro perfil de GitHub. Este fork es una copia del repositorio original, pero todos los cambios no afectarán al original, si no a nuestra copia únicamente.

Hay que hacer UN repositorio como mínimo por cada tarea, y UNA pila de producto por cada tarea.

TODOS haréis un fork, pero la idea es que hagáis esta práctica todos juntos, modificando el proyecto, pero dividiéndoos las tareas, **el equipo estará formado por un jefe y cuatro desarrolladores.**

El jefe **tendrá que añadir colaboradores a la organización**, **además de crear la wiki y las tareas de todos los desarrolladores**. Así mismo, deberá encargarse de que los desarrolladores suben los cambios a la rama adecuada, hacer los commits con mensajes descriptivos…

Cada jefe debe hacer las siguientes acciones:

1. Crear la wiki de su proyecto (en GitHub) donde describirá las acciones a llevar a cabo por cada desarrollador.
2. Crear tres etiquetas (médium, high, low) para indicar el tamaño de la tarea a realizar (las que sean high se tendrán que dividir en otras más pequeñas al pasarlas a la pila del sprint)
3. Crear un **Proyecto** con las columnas *BACKLOG, TO DO*, *DOING* y *DONE*, creando las tareas a realizar por cada desarrollador y asignárselos al mismo, simulando una pizarra ágil sencilla. Cada desarrollador tiene que informar del estado de la tarea.
4. Revisar el trabajo realizado por cada desarrollador y validarlo (mediante pull requests)
5. Realizar el Scrum daily y gestionar las tareas hechas y por hacer al final del sprint.
6. Presentar un producto mejorado al final de cada Sprint y modificar la definición de hecho.

Cada desarrollador debe hacer las siguientes acciones:

1. Desarrollar las tareas que le asigne su jefe de proyecto.
2. Comprometer y subir las tareas al repositorio asociado.
3. Cuando finalice una mejora deberá actualizar la tarea y establecerla como cerrada.

Normas a tener en cuenta:

1. NO se desarrolla nunca sobre la rama master
2. TODOS los merges que se hagan al repositorio final han de ser usando pull requests.
3. ***TODOS los repositorios han de ser trabajador por DOS desarrolladores por lo menos.***
4. Dividir el trabajo en tareas pequeñas es lo ideal, pero evitad hacerlas ***demasiado*** pequeñas, de forma que inunde la pila y estéis desbordados de trabajo. Con el paso de los sprints se sabe cuál es el ritmo de cada alumno y qué es lo que puede hacer.
5. El PO es un cliente un poco maleducado, por mucho que parezca que habéis limpiado la pila de tareas, parece ser que se inventa nuevas…
6. Al final de cada sprint semanal y al final de proyecto, hay que añadir un resumen del mismo en un documento el cual será escrito por el jefe de proyecto: nº de tareas estimadas y finalizadas, previsión del siguiente sprint, definición de hecho del siguiente sprint, errores surgidos, distribución de las tareas…

**Entrega**

La fecha de entrega será el 31/1/2022 a las 23:59 como máximo.

Hay que entregar:

* Las tres tareas finalizadas y probadas en sus respectivo(s) repositorio(s) , cumpliendo el concepto de ***hecho*** especificado al principio de la práctica.
* El Excel relleno con el número de horas dedicadas y las tareas finalizadas.
* Un documento .pdf explicando, al final de cada sprint:
  + El estado de las tareas (podéis hacer pantallazo de la pizarra del GitHub) y reorganización de las mismas (alguna que pasa a ser más/menos prioritaria…)
  + La actualización de la definición de hecho (qué es lo que va a otorga valor al final del siguiente sprint)
  + El resumen del reparto de trabajo.
  + Errores que hayáis tenido con Git, cualquier cosa a comentar de utilidad…

La nota se dividirá en:

* **4p**: Se ha logrado satisfacer la definición de hecho al final de cada sprint y al final del proyecto. No satisfacer esta condición al final de algún sprint será 1 p menos. No satisfacerla al final conlleva perder toda esta nota.
* **4p**: Se ha usado el GIT de forma adecuada siguiendo las buenas prácticas:
  + Diseño de repositorios y ramas que favorezcan el trabajo en paralelo.
  + **Creación de ramas por cada funcionalidad**
  + **Commits con mensajes** [**siguiendo las buenas prácticas**](https://gist.github.com/robertpainsi/b632364184e70900af4ab688decf6f53)
  + **Fusión de ramas tras la finalización de las tareas**
  + **Uso de pull request para traer cambios al repositorio oficial.**
  + **Contribución equitativa de todos los desarrolladores.**
* **2p: Se ha usado Scrum de forma correcta:**
  + Uso de la pizarra de GitHub, y de los issues para la gestión y asignación de tareas.
  + Contabilización de las horas y tareas realizada por cada miembro de trabajo en el excel adjunto.
  + Reordenación de la pila de producto tras un sprint.
  + Creación del documento de resumen del final de cada sprint.
  + Se ha especificado la definición de hecho para cada sprint, y se ha cumplido. Esta definición se escribirá en un Excel común a los dos grupos (A y B)
    - [DAW B](https://docs.google.com/spreadsheets/d/1SWJJASqN-a1VdsrPbzmK6c0u13y1YYMnqynemVtjVYU/edit#gid=0)
    - [DAW A](https://docs.google.com/spreadsheets/d/1yg1wfrKwbEHP7EcLGE_ItWocI5i5Q4FzCOGWL2DghIk/edit#gid=0)

Además, se realizará una heteroevaluación sobre el trabajo del resto de compañeros.

## Anexo I

## Crear en C#, una solución que incluya un proyecto por cada punto que veis debajo. Cada subpunto debe ser una funcionalidad de cada proyecto.

## Cada proyecto debe ofrecer un menú interactivo que permita al usuario introducir datos por teclado, y seleccionar la función a realizar.

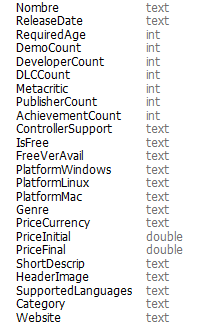
* Array
  + Calcular máximo
  + Calcular mínimo
  + Calcular la mediana
  + Calcular la media
  + Ordenar un Array de menor a mayor
  + Calcular la desviación típica de un array
  + Binarizar un array en base a un número n. Esta función devolverá un nuevo array, cuyos elementos valdrán 0 o 1. Si el componente i del array original es menor que n, el componente i del array resultado será 0, y 1 en el caso contrario.
* Numero
  + Calcular si el número es primo
  + Calcular el factorial
  + Dado un número en segundos, devolver horas y minutos (3680 segundos son 1 hora, 1 min, y 20 segundos).
  + Devolver los primeros 15 números de la [serie de Pell](https://es.wikipedia.org/wiki/N%C3%BAmero_de_Pell)
  + Devolver si un número es un [número de Armstrong](https://es.wikipedia.org/wiki/N%C3%BAmero_narcisista) o no
  + Devolver si un número tiene todos sus dígitos diferentes
* String
  + Devolver si el string es un palíndromo
  + Escribir el nombre todo en mayúsculas y todo en minúsculas
  + Devolver el número de ocurrencias de una letra
  + Dados dos Strings distintos, hacer que se repitan las veces que informe el usuario.(Alejandro, crack, 2 -> Alejandro crack Alejandro crack)
  + Dado un String, repetir cada ocurrencia de su letra. Por ejemplo: NBA se convierte en NNBBAA, Madrid en MMaaddrriidd.
  + Dado dos Strings, borrar del primero las apariencias del segundo. Por ejemplo, dado *Real Zaragoza es de la realeza* y *Real*¸ la función devolverá *Zaragoza es de la eza*.
* Decimales
  + Redondear el número al alza
  + Eliminar los decimales del número
  + Dados tres coeficientes ejecutar la ecuación de segundo grado

## Anexo II

Realiza estas queries sobre la base de datos *steam*, la cual contiene información sobre más de 13000 juegos disponibles en la plataforma de distribución Steam.

Dada tu previa experiencia como ***gamer™*** la liga de videojuegos canaria te quiere contratar para que analices los datos y obtengas información sobre qué juegos poder presentar en la nueva temporada.

En la tabla, las columnas que tenemos modeladas son:



Una breve descripción de algunas columnas:

* *Genre*: El género del juego (deportes, acción). Puede tener más de un género
* *Metacritic*: Media de puntuación del juego
* *SupportedLanguages*: El número de idiomas en los que se puede jugar
* *PublisherCount*: El número de estudios que han desarrollado el juego
* *Platform[Linux,Mac,Windows]*: True si está disponible en dichos SO, false si no
* *Category*: La categoría del juego (MMO, single, multiplayer…)

Las consultas que hay que hacer son:

1. Lista todas las columnas de la tabla.
2. Lista todos los diferentes géneros, eliminando duplicados.
3. Lista únicamente aquellos juegos que puedan ser jugados únicamente en Linux
4. Lista todos los juegos de rol, ordenador por nota.
5. Lista todos aquellos juegos que puedan ser jugados en Español pero no en Frances
6. Lista el nombre del juego, su precio inicial, su precio final y la diferencia entre ambos en una nueva columna.
7. ¿Cuántos juegos hay del siglo pasado?
8. Devuelve todos los juegos que terminen o en N o en O.
9. Devuelve todos los juegos que empiecen por A y que se puedan jugar con mando.
10. Devuelve todos aquellos juegos que estén entre el 2000 y el 2010.
11. Devuelve todos los juegos que sean de la saga Final Fantasy.
12. Devuelve todos los juegos que sean de deportes y haya trabajado más de 3 desarrolladores.
13. ¿Cuántos juegos hay asociados a cada categoría?
14. ¿Cuántos juegos se han sacado en cada año?
15. En base a la consulta anterior, devuelve aquellos años en los que la media de puntuación esté entre un 6 y un 8,
16. ¿Cuál es la máxima, mínima y puntuación mínima por género?
17. Usando LIMIT, devuelve el top 10 de juegos con mayor puntuación del 2012.
18. Usando LIMIT, devuelve el top 10 de juegos más nuevos de género single player.
19. Devuelve la media de nota de todos aquellos juegos que sean para mayores de 18 años.
20. ¿Cuántos juegos hay asociados a cada tipo (mayor de 18, de 17…)?
21. Devuelve todos aquellos años en los que haya menos de 300 juegos.
22. Devuelve todos los juegos que estén para Mac pero no para Windows.
23. Devuelve todos los juegos donde su precio final sea mayor a su precio inicial.
24. Devuelve todos los juegos que no estén valorados en dólares.
25. Devuelve todos los juegos que tengan una mayor nota que 0, pero que hayan suspendido.
26. Devuelve el top 15 de juegos con mayor número de DLC.
27. Devuelve la información de los juegos que sólo se puedan jugar en Inglés.
28. Devuelve el nombre(en minúscula) y la web (en mayúscula) de los juegos de acción o casuales.
29. ¿Cuál es el juego indie con mayor nota?
30. ¿Y con menor nota?

Parte complicada

Si llegas aquí es que hay que usar subconsultas. Si no sabéis lo que es contactad con vuestro product owner 😊

1. Devuelve toda la información del juego con menor nota, siempre que sea mayor a cero.
2. Devuelve aquellos juegos que tengan mayor nota que la media.
3. Devuelve el juego con mayor nota del año 2008.
4. Devuelve toda la información de los juegos que valgan más que la media.
5. Devuelve toda la información de los juegos de Linux que tengan el mayor número de logros (*achivements*)