Proyecto curso

“FORMACIÓN INICIAL JAVA/DIGITAL”
LucaSteam
[Desarrollo de la gestión de un listado de juegos online]
[Lucatic]

14/06/2021 (Ed.24)(Online)

ÍNDICE DE CONTENIDOS

Planteamiento

Una empresa quiere disponer de una **plataforma tipo Steam** para juegos. Es un proyecto con RA para juegos inmersivos. De momento en fase beta y trabajando la **parte administrativa**.

Y como no hay fondos, pues tiran de gente de esa que sabe algo, que hacen directos, que se desmotivan porque el E3 no trae muchas novedades, que se van cayendo cada dos por tres, gente que busca el fresquito en las paredes o en los suelos o que no tienen otra cosa que hacer y se pasan ahí el día y las noches de jajas y salseo con el Slack. Prometen ser mano de obra barata… mu responsable y que quieren aprender... buena gente, sis… tías y tíos (espera, que a va ser de no) preparas’... orgullosas de ella (esto del escribir sólo pa tías se me va de las manos), que rentan mazo, con tolerancia a la frustración, que trabajan en equipo… gente tecnológica, que se come los problemas... que los resuelven... que analizan... que buscan soluciones... que son como perros de presa, que pim-pam-pim-pam que tum pam pam que tumpampam quetepetepete… que se adaptan... que son máquinas... son bestias pardas... son mu güenas… son mu chetas… y lo sabes. GL & HF

Grupos de Trabajo

Para trabajar se realizarán **cuatro grupos**.

Grupo 01	Grupo 02	Grupo 03	Grupo 04
Daniela	Jennifer	Anabel	Ana Díaz
Desiree	María C.	Delia	Ana María
Irene	Patricia	Marta	Rebeca
María H.	Sara Silvo	Natalia	Rocío
	Usoa	Noemi	Sara Sevillano

Roles de trabajo

Los propios de la metodología SCRUM

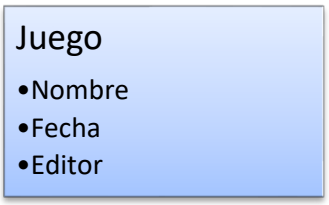
- **SCRUM Master**
- **Equipo de desarrollo**
- **Product Owner**:
  - Figura realizada por el profesor en última instancia para decisiones finales
  - Desarrollada por el equipo para crear el Product Backlog

Idea de referencia

En el futuro, LucaSteam será un servicio de juegos online que recopila los titulos y los adapta de todas las plataformas existentes.

De momento necesitan ayuda para la gestión de la lista, ya que cuando se comercialice tendrá un catálogo especializado (se trabajarán por género y plataforma).

Tanto el género como la plataforma serán enumerados



Objetivos marcados

El **objetivo principal** de este proyecto se centra en tres aspectos:

- Conseguir que el grupo de alumnos sepa **utilizar** y mezclar la **teoría** y las **técnicas** vistas durante el curso relacionadas con Java y Pruebas Unitarias.
- Ayudar a que el alumno **desarrolle** y **potencie** las **competencias** marcadas o previstas para este curso:

Análisis y resolución de problemas	Cuenta con mi hacha	Autonomía
Trabajo en equipo	Comunicación	~~Thug Life~~
Flexibilidad	Proactividad	Motivación
Tolerancia a la frustración	Iniciativa	Interés a tope, lo voy a petar
	Soy tu BAE	Capacidad de aprendizaje
	Responsabilidad	

- Aprender y entender cómo funciona una **metodología de trabajo** como **SCRUM**

Descripción del proyecto (Tareas de Programación)

El grueso del proyecto **se centrará de forma exclusiva en la zona de <u>ADMINISTRACIÓN</u>** y dispondrá de las siguientes tareas:

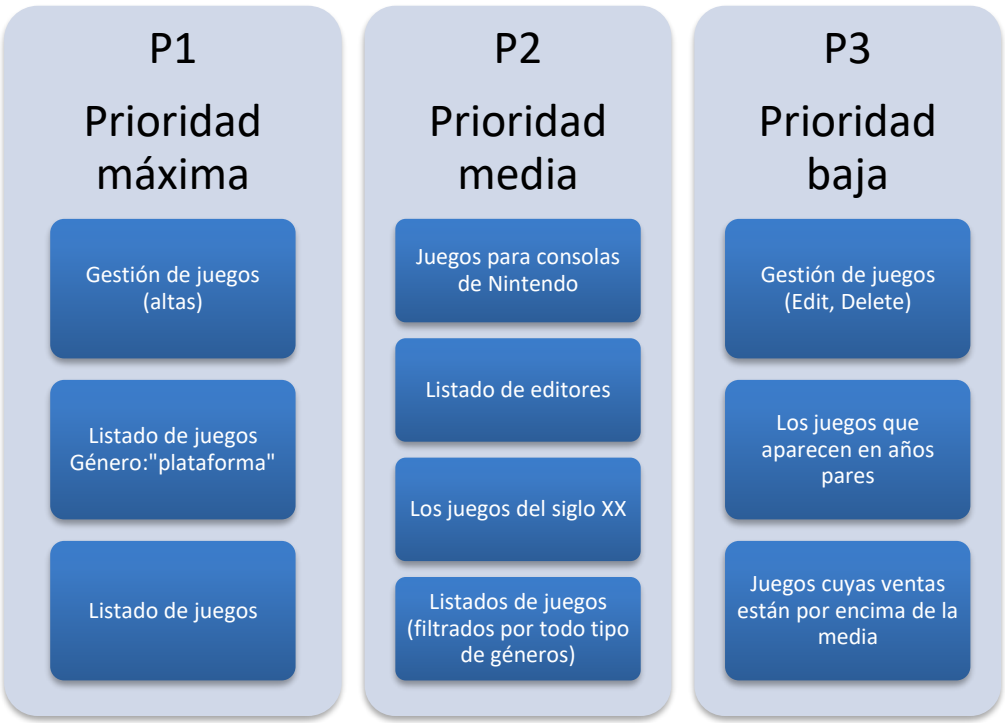
a. Gestión

1. Gestión de juegos

b. Informes

1. Listado de juegos
2. Listado de editores (Publisher)
3. Listado de juegos (filtrado por géneros)
4. Los juegos del Siglo XX
5. Juegos aparecidos en años pares
6. Juegos con media de ventas en Europa por encima de la media en Europa
7. Juegos para las consolas de Nintendo

Las tareas a realizar pueden dividirse en tres grupos dependiendo de su prioridad.



NOTA 01:	Dispondremos de un **listado de juegos** a partir de un dataset extraído de https://www.kaggle.com/gregorut/videogamesales[1]. Se usará **para realizar la carga inicial** y las primeras pruebas. Ese fichero estará en **formato CSV**
NOTA 02:	El listado de plataformas, editores y géneros se obtendrá a partir del CSV. Para vuestra información hay más de 10 valores en cada caso.
NOTA 03:	Trabajar con los datos debe ser también una tarea. Los datos están en CSV y se puede trabajar con todos o con una batería de unos 1000 registros
NOTA 04:	De las ventas sólo interesan las de Europa. Si son muchos datos de ventas, se pueden eliminar las otras columnas

Entregas a realizar a lo largo del proyecto

a) Cada uno de los sprints marcados en la metodología de Sprint
b) El sprint final viene marcado por la entrega del proyecto.
c) A partir de ese día no se seguirá trabajando en el proyecto.
d) La presentación se realizará el último día de curso.

Trabajo a realizar a lo largo del proyecto

Cada grupo de alumnos/as debe **desarrollar** y **programar** las distintas partes marcadas en la "Descripción del Proyecto".

Esas tareas pueden ser de dos tipos:

- **Tareas de Programación**: definen las acciones que debe realizar el grupo a lo largo del proyecto para crear la aplicación.
- **Tareas Transversales**: definen las acciones no relacionadas de forma directa con la programación pero que permiten la mejora competencial o la calidad del trabajo realizado.

El equipo debe **usar la metodología SCRUM** y crear un backlog de productos (revisable y ampliable durante el proyecto), un backlog de tareas por cada sprint y los cuatro tipos de reuniones (reuniones diarias, planificación de sprint, revisión y presentación de producto).

[1] Tanto https://www.kaggle.com/datasets como https://corgis-edu.github.io/corgis/ incluyen datasets

Descripción del proyecto Web (Tareas transversales)

Hay una serie de tareas a pedir a lo largo del proyecto (las 6 son de Prioridad Máxima)

En general estas taras deben ser asumidas por TODO el equipo

- **Documentación Colaborativa**: Generar una documentación "formal/informal"
  - Posibles elementos: tareas realizadas, explicación breve de procesos y/o arquitectura, trabajo Sprint realizado, pantallazos de product backlog, etc.
- **Documentación JavaDoc**: Generada a partir de los scripts. Puede incluirse en la documentación colaborativa.
- **Uso de repositorios**: para compartir las versiones de código.

  Ideas: GitHub, GitLab, Bitbucket.
- **Herramientas de Seguimiento de Proyectos y Gestión**: que permiten gestionar proyectos Scrum, tiempos, etc. (o cualquier acción que se considere necesaria).
  - Para este proyecto se usarán, al menos, Jira y Slack.
- **Pruebas de calidad**: El código de un método/clase **NUNCA ESTARÁ COMPLETO** si no pasa estas pruebas.

Para Clases

- Todos los paquetes van en minúsculas
- No se puede usar en el import .*
- En el import no pueden existir clases no utilizadas
- El código debe estar correctamente formateado
- Todo el código tiene que tener una cabecera en Javadoc que incluya: nombre de la clase, descripción, fecha, versión y autor.
- Toda clase (model) debe incluir setter/getter, constructor (default) y toString.
- Intenta simplificar con Lombok si es posible

Para métodos

- Las llaves de inicio del método están en la misma línea que la cabecera
- Todos los nombres de variables son descriptivos
- Usar sistema de logging para la información del programador

- **Pruebas unitarias**: define, de forma previa, varias pruebas unitarias (con Junit)
  - Tienes que usarlas en todos los sprint. Pueden ser sencillas.
  - Hay que realizar al menos 15 pruebas unitarias que cubran aspectos de todo tipo. Algunas ideas son: comprobar que un elemento concreto exista, comprobar que no exista, asegurar que una lista vacía funciona, comprobar que no se puede eliminar un juego de una lista vacía, observar que ocurre cuando no hay juegos de una plataforma (o de un género o de un editor), comprobar que los datos se leen bien, comprobar que pasa si el csv está mal, comprobar que la suma de los juegos de Nintendo es igual a los de cada plataforma individual, comprobar que pasa cuando se pasa un año posterior o inferior a 1958, incluir un dato de una plataforma no existente (o género o editor), comprobar que se puede editar, comprobar que algo se ha dado de alta, comprobar datos de tipos incorrectos, etc.

Normas a cumplir

Hay una serie de normas e ideas que debes implementar en tu proyecto

- Uso de **interfaces**
- Uso de capas de **Servicios**, **Controladores** y **Datos** (o parecidas)
- Uso de **Excepciones**
- Dividir en distintos **paquetes**
- Cada "**pantalla gráfica**" debe ser un método que muestre esa información
- Utilizar una clase externa (y static) para leer datos

La pauta básica para todo el proyecto es la misma:

Ante la duda… pregunta al profe

Ante la "reduda"… simplifica

Si un proyecto está suspenso… todos están suspensos

¿Cómo se valorará el proyecto?

Para valorar el proyecto se tendrán en cuenta tres aspectos

a. **Valoración técnica**: se revisará la calidad de la solución, elementos empleados, técnicas usadas, empleo de tecnologías, aplicación de los conocimientos vistos.

b. **Materiales aportados**: tanto el código como la documentación si se ha elaborado, valorando la calidad de la misma, la aportación al proyecto, la implicación del equipo en la misma, etc.

c. **Valoración competencial**: tomando como referencia algunas ideas basadas en competencias profesionales como el trabajo en equipo, análisis y resolución de problemas, flexibilidad, etc.

d. **Directrices SCRUM**: haber seguido los parámetros indicados por la tecnología, seguir las necesidades del cliente, etc.

ANEXO 01 - BACKLOG DEL SPRINT (Ejemplo)

A continuación, se muestra como ejemplo el posible desglose de una tarea para el sprint. En este caso se elige como primera tarea "LISTADO DE PERSONAS".
Supondremos que el listado sólo muestra el nombre y los apellidos

03) Listado de Personas (sin filtrar)
- 03.1. Crear Estructura Principal del proyecto
- 03.2. Crear sistema de repositorio de código si se va a emplear (y dar de alta usuarios)
- ~~03.3. Crear estructura principal de base de datos~~
- ~~03.4. Crear tabla PERSONAS (básica)~~
- 03.5. Crear entidad PERSONAS
- 03.6. Crear método getPersonas():List<Personas> en capa DAO (Datos)
  - 03.6a. Crear interface DAOPersonas
  - 03.6b. Crear clase DAOPersonasImpl
  - 03.6c. Dar de alta el método getPersonas():List<Personas> en la interface
  - 03.6d. Crear el algoritmo getPersonas():List<Personas> en la clase
  - 03.6e. Documentar el método
  - 03.6f. Comprobar las pruebas de calidad
  - 03.6g. Completar las pruebas unitarias (si es necesario)
- ~~03.7. Configurar conexión a BBDD.~~
- 03.8. Crear método getPersonas():List<Personas> en capa Service
- 03.9. Crear método getPersonas():List<Personas> en capa Controller
- ~~03.10. Realizar prototipo página HTML Listado~~
- 03.11. Montar ~~Página~~ y parte gráfica

ANEXO 02 - Definición de tarea acabada

Una tarea se considera acabada cuando se han completado los siguientes pasos:

- Se ha actualizado la interface (si es necesario)
- Se ha completado el algoritmo
- Se ha comprobado que realiza lo que se pedía
- Se ha completado la documentación necesaria
- Se han realizado las pruebas de calidad
- Se han completado las pruebas unitarias en el caso de necesitarse
- Se ha subido al repositorio de código compartido que se haya utilizado
- Se ha notificado en el sistema de control (Jira)
- El cliente la acepta si fuera necesario (siempre que sea a nivel de Historia de usuario)