Proyecto curso

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

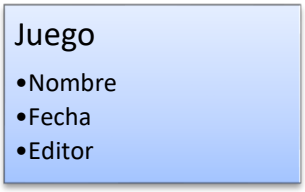
16/01/2023 (Ed. E04)(Online)

ÍNDICE DE CONTENIDOS

Idea de referencia

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

De momento necesitan ayuda SOLO para gestionar listados, ya que cuando se comercialice quieren disponer de un catálogo especializado (se trabajarán por género y plataforma).



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 Python.
- 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	Responsabilidad
Trabajo en equipo	Comunicación	Autonomía
Flexibilidad	Proactividad	Motivación
Tolerancia a la frustración	Iniciativa	Interés a tope, lo voy a petar
	Mi grupo es mi BAE	Capacidad de aprendizaje

- Aprender y entender cómo funciona una **metodología de trabajo** como **SCRUM** y otras herramientas como **Git**, **Jira**, **Confluence**, etc.

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

Grupos de Trabajo

Para trabajar se realizarán **4 grupos**.

Grupo 01	Grupo 02	Grupo 03	Grupo 04
Álvaro	Ángel B.	Alberto	Anna
Antonio	Ángel F.	Kevin	Carlos
Jesús	Eric	Rubén	Eduard
Moisés			Eloy

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:

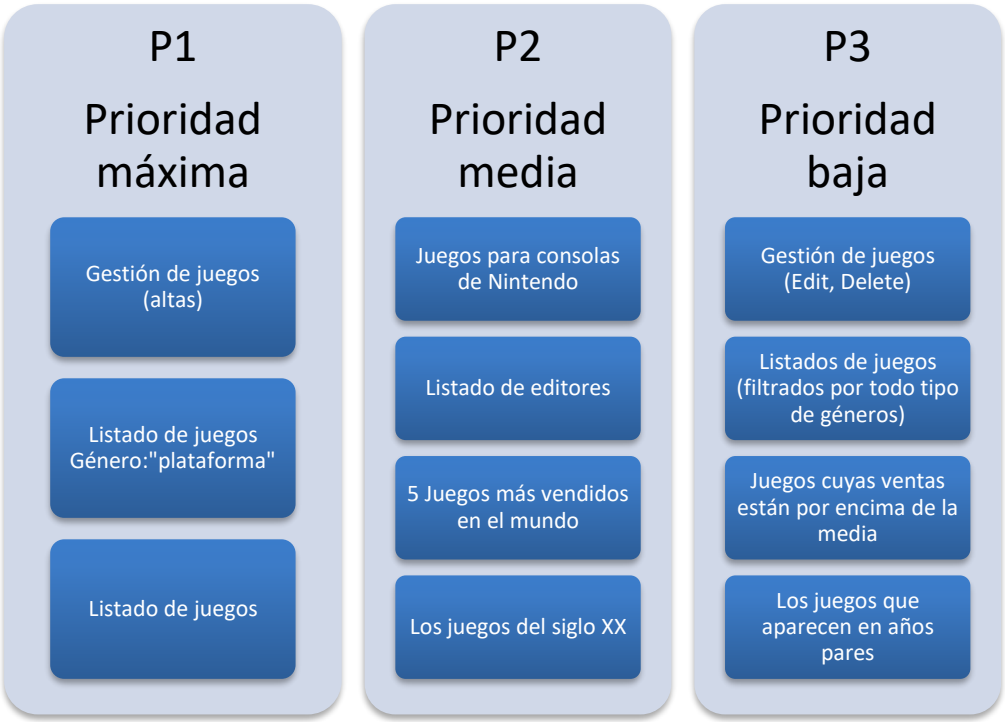
a. Gestión

1. Gestión de juegos (alta, update,…, CRUD)

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 para las consolas de Nintendo
7. Juegos con media de ventas en Europa por encima de la media en Europa
8. Los cinco juegos más vendidos en NorteAmérica, Europa, Japón, Resto del mundo

Las tareas para 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 está en **formato CSV**
NOTA 02:	El listado de plataformas (Platform), editores (Publisher) y géneros (Genre) se obtendrá a partir del CSV. ¿Es posible automatizarlo?
NOTA 03:	Trabajar con los datos debe ser también una tarea. Los datos están en CSV y se debe trabajar con todos los registros

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.

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

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 "Paralelas"**: 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).

Descripción del proyecto Web (Tareas "paralelas")

Hay una serie de tareas a pedir a lo largo del proyecto (las hay de distintas prioridades)

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.
  - Realizada en Confluence
- ~~**Documentación DocString**: 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 Discord.
- **Pruebas de calidad**: El código de funciones/clases/… **NUNCA ESTARÍA COMPLETO** si no pasa estas pruebas.
  - Todo el código debe seguir las pautas PEP 8[2]
  - También puedes usar herramientas de tipo linting como Flaske8, Pylint, Pyflaskes, etc.[3]
- **Pruebas unitarias**: define, de forma previa, varias pruebas unitarias (con unittest, por ejemplo)
  - Pueden ser sencillas. Realizar al menos 10-12 pruebas unitarias que cubran aspectos de todo tipo. Algunas ideas son:
    - Comprobar que un elemento concreto exista… o que no exista
    - Asegurar que una lista vacía funciona
    - Comprobar que no se puede eliminar un juego de una lista vacía
    - ¿Qué ocurre cuando no hay juegos de un género (o editor o…)?
    - 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 pre/post 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.

[2] https://peps.python.org/pep-0008/
[3] https://donjayamanne.github.io/pythonVSCodeDocs/docs/linting/

Normas a cumplir

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

- En la raíz de cada uno de los módulos del programa SOLO pueden existir 1 ó 2 líneas "sueltas" (El resto en una **función main**)
- Dividir en distintos **paquetes**
- Agrupar las funciones en **módulos**
- Uso de **Excepciones**
- Cada "**pantalla gráfica**" debe ser una función que muestre esa información

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) Yo como administrador quiero obtener un listado de Personas para poder acceder a información detallada de cada uno

- 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. Montar "parte gráfica" (menú opciones)
- 03.6. Crear clase PERSONAS
- 03.7. Crear la función get_personas():List<Personas>
  - 03.6a. Crear modulo gestion_personas.py
  - 03.6b. Crear paquete para gestión de personas (si fuera necesario)
  - 03.6c. Crear el algoritmo get_personas():List<Personas> en el módulo
  - 03.6d. Documentar el método (opcional)
- 03.8. Comprobar las pruebas de calidad
- 03.9. Representar la información obtenida
- 03.10. Completar la Prueba Unitaria 01
- 03.11. Completar la Prueba Unitaria 02…

ANEXO 02 – Definición de tarea acabada

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

- 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)