UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA
SUPERIOR
Gºen Ingeniería en Informática

TFG Ingeniería Informática:

Retrogaming Recommender

Presentado por Raúl Ollés Clavelo
en Burgos el 3 junio de 2019
Tutores D. José Manuel Galán Ordax
y D. José Ignacio Santos Martín

D. José Manuel Galán Ordax y D. José Ignacio Santos Martín, profesores del departamento Ingeniería Civl, área de Organización de Empresas

Expone:

Que el alumno D. Raúl Ollés Clavelo., con DNI 73257232R, ha realizado el TFG Ingeniería Informática titulado: Retrogaming Recommender.

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a 3 de junio de 2019

Vº. Bº del Tutor

D. José Manuel Galán Ordax

Vº. Bº. del Tutor

José Ignacio Santos Martín

Resumen

Retrogaming Recommender es una aplicación web que permite a los usuarios registrados el disponer de recomendaciones personalizadas de juegos de la mítica Commodore 64.

Las recomendaciones están basadas en tres tipos de filtros:

- Filtro colaborativo basado en modelos.
- Filtro colaborativo basado en memoria de usuarios.
- Filtro colaborativo basado en memoria de juegos.

Tanto los datos identificativos de cada juego como la posibilidad de jugar directamente a él por medio de un simulador web, se realiza gracias a la web archivae.org.

En Retrogaming Recommender podrá valorar cada juego, ver recomendaciones, jugar en la propia web, listar juegos según datos estadísticos y buscar un juego o juegos según una amplia lista de criterios.

La aplicación web está disponible en https://retro-gaming.herokuapp.com/

Descripciones

Recomendación, predicción, filtro colaborativo basado en modelos, filtro colaborativo basado en memoria de usuarios, filtro colaborativo basado en memoria de productos, aplicación web, python.

Abstract

The web application is available in https://retro-gaming.herokuapp.com/

Keywords

Recommendation, Prediction, ……, web application, python.

Índice General

Índice de figuras

A. Introducción

Casi sin darnos cuenta nos hemos acostumbrado a utilizar aplicaciones donde se dispone de una infinidad de productos. Pero, no somos los únicos que las utilizan, hay muchas personas con costumbres o hábitos similares a los nuestros que estarán accediendo a ellas simultáneamente.

La forma más básica de llegar hasta un producto sería localizarlo a través de una lista general o dividida por secciones o tipos de productos. Pero si nuestra aplicación dispone de muchos productos, el buscar uno de ellos podría llegar a ser tedioso y aburrido, y al final, perderíamos el gusto en utilizarla.

¿Cómo podemos ayudar a los usuarios de nuestras aplicaciones a acceder a productos de su gusto más rápidamente, especialmente en catálogos amplios? ¿Cómo podemos ofrecer contenido personalizado según las características de un usuario?

Para eso han nacido los **sistemas de recomendación**, los que hacen posible predecir los gustos y tendencias de un usuario teniendo en cuenta los productos que disponemos y el historial del resto de los usuarios [1].

El poder ofrecer productos que sabemos (o predecimos) que le interesan y gustan a nuestro cliente tiene un máximo interés si nuestro negocio se centra en el comercio electrónico.

Pero, también si estamos dirigidos al ocio y ofrecemos algún tipo de contenido. Pensemos en la gran diferencia de acceder a una aplicación web y ver directamente el contenido que a nosotros más nos interesa, con el que tengamos que buscarlo entre una infinidad de secciones y productos.

Ejemplos cotidianos de predicciones para usuarios son Amazon [2], Netflix [3] o Youtube [4], aunque en muchas otras pequeñas y grandes aplicaciones también se utilizan.

Además de los sistemas de recomendación utilizados, otra parte importante del proyecto la forma el sistema de recogida de datos de juegos utilizando **scraping**.

La aplicación que se ha desarrollado en este proyecto (**Retrogaming Recommender**) ofrece para cada usuario predicciones de valoraciones o gustos para cada usuario de entre unos diez mil juegos. Estos juegos, junto con sus datos estadísticos, no han sido introducidos previamente en la aplicación.

Para disponer de la información de los juegos a ofrecer se ha realizado una labor de scraping sobre la página archive.org, seleccionando de los juegos que corresponden a la categoría commodore 64, los datos de interés.

1.1. Estructura de la memoria

La memoria está estructurada de la siguiente forma:

- **Introducción**. Breve descripción del contenido del trabajo. Estructura de la memoria. Recursos disponibles en internet.
- **Objetivos del proyecto**. Exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos.** Explicación de los conceptos teóricos clave para el desarrollo del proyecto. Se da mayor importancia a los conceptos teóricos que no sean de común utilización.
- **Técnicas y herramientas**. Explicación de las técnicas y herramientas utilizadas para llevar a cabo el proyecto.
- **Aspectos relevantes del desarrollo**. Explicación de aquellos aspectos que han resultado de interés durante el desarrollo del proyecto.
- **Trabajos relacionados**. Pequeño resumen de trabajos y proyectos ya realizados en el campo del proyecto.
- **Conclusiones y líneas de trabajo futuras**. Conclusiones que se derivan del desarrollo del proyecto. Informe de mejora y continuación.

Anexos a la memoria.

- **Planificación**. Planificación temporal. Estudio económico.
- **Especificación de Requisitos**. Objetivos generales. Catálogo de requisitos. Especificación de requisitos.
- **Especificación de Diseño**. Diseño de datos. Diseño de paquetes. Diseño gráfico.
- **Documentación técnica de programación**. Estructura de directorios. Manual del programador. Pruebas unitarias.
- **Documentación de Usuario**. Requisitos de usuario. Instalación. Manual de usuario.

1.2. Recursos disponibles en internet

- Proyecto en web.
  - https://retro-gaming.herokuapp.com/
  - Generados 1.000 usuarios con nombres de usuario del **user1** al **user999** y contraseña igual para todos "**parola**"

- Repositorio del proyecto.
  - https://github.com/raulolles/TFG_GII_O_MA_18.08
  - Aplicación web desarrollada en Flask
  - Python scripts para scraping

- Pruebas unitarias de aplicación web
- Pruebas unitarias de tablas scraping

B. Objetivos del proyecto

Retrogaming Recommender pretende ayudar a los nostálgicos de Commodore 64 el acceder rápidamente a los juegos más cercanos a sus preferencias.

Para ello nos centramos en los siguientes objetivos:

2.1. Objetivos generales

- Desarrollar una aplicación web que ofrezca a sus usuarios la mejor recomendación predictiva de juegos para Commodore 64.

- Desarrollar las herramientas que permitan ofrecer el contenido y las predicciones a la web:
  - Recogida de datos desde la web suministradora de contenidos de juegos.
  - Diferentes predicciones de juegos según los sistemas de recomendación implementados.
  - Permitir al usuario jugar al juego seleccionado desde la propia web.

- Dotar a la web de otras utilidades como:
  - Listados de juegos según datos estadísticos.
  - Búsqueda avanzada de juegos

- Desarrollar una aplicación web que de cara al usuario resulte:
  - Estéticamente atractiva.
  - Fácil de interactuar.

2.2. Objetivos técnicos

- Utilizar técnicas de scraping con Selenium y ChromeDriver para recabar los datos necesarios de juegos.

- Utilizar algoritmos de predicción basados en filtros colaborativos.

- Utilizar Python y Flask para el desarrollo de una web que ofrezca a los usuarios las predicciones de sus juegos preferidos.

- Utilizar el poder de las extensiones de Flask.

- Utilizar la arquitectura MVP (*Model-View-Presenter*)
- Aplicar los principios de la metodología ágil y concretamente a través de Scrum.
- Utilizar las diferentes herramientas que proporciona GitHub para el control de versiones.
- Utilizar herramientas de control de la calidad del código como Codacy.
- Realizar test unitarios que garanticen la calidad del producto.
- Utilizar herramientas de software libre.
- Hacer disponible la aplicación a través de la web.
- Desarrollar una aplicación web que resulte agradable y fácil de interactuar con el usuario.

2.3. Objetivos personales

- Ampliar conocimientos de:
  - HTML
  - JavaScript
  - Framworks Python HTML
- Ver de forma práctica y personal conocimientos adquiridos durante la carrera como:
  - Principios de metodología ágil.
  - Herramientas de control de calidad de código.
  - Pruebas de control.
  - Sistemas de predicción.

C. Conceptos teóricos

En la presente sección se explicarán algunos conceptos que, siendo parte importante del proyecto, requieren profundizar en su significado y en la implicación dentro del conjunto.

El proyecto, que al final se ve materializado en una aplicación web, tiene tres grandes pilares:

- Scraping. Recogida de datos de la web.
- Sistema de recomendación o predicción. Sugerencias predictivas a los usuarios.
- Diseño y desarrollo web.

De los tres mencionados, se considera que son los dos primeros los que dan una singularidad al proyecto, siendo el tercero la mera forma de ofrecer lo conseguido. Además, de los tres es el diseño y desarrollo web el más conocido. Por eso, a continuación, nos centraremos en la explicación teórica de scraping y de sistemas de recomendación o predicción.

3.1. Scraping

Web scraping (rastreo) es una técnica que permite extraer información de internet automáticamente utilizando software que simula la navegación web [5] [6].

Por lo tanto, podremos conseguir de la web cantidades industriales de información sin teclear una sola palabra. A través de algoritmos de búsqueda podremos rastrear centenares de webs para extraer sólo aquella información que necesitamos [7].

Como el web scraping simula el comportamiento humano en la navegación web, debemos conocer la estructura de la web a rastrear para entender en que parte y con qué atributos html se encuentran los datos que nos interesan.

Ejemplos de utilización de web scraping:

- **Marketing de contenidos**: Diseñar un “robot” que haga un scraping (rastreo) de datos concretos de una web, utilizándolos para generar nuestro propio contenido.
    - Ejemplo 1: scraping los datos estadísticos la web oficial de una liga de fútbol para generar nuestra propia base de datos.
    - Ejemplo 2: **Retrogaming Recommender**. Scraping de datos de otras webs (archive.org) para nutrir de contenido la nuestra propia (datos de juegos commodore 64)

- **Ganar visibilidad en redes sociales**: Utilizar los datos de un scraping para interactuar a través de un robot con usuarios en redes sociales.
    - Ejemplo: crear un bot en instagram que seleccione los links de cada foto y luego programar un comentario en cada entrada.

- **Controlar la imagen y la visibilidad de nuestra marca en internet**: A través de un scraping podemos automatizar la posición por la que varios artículos de nuestra web se posicionan en Google o, por ejemplo, controlar la presencia del nombre de nuestra marca en determinados foros.
    - Ejemplo: rastrear la posición en Google de todas las entradas de nuestro blog.

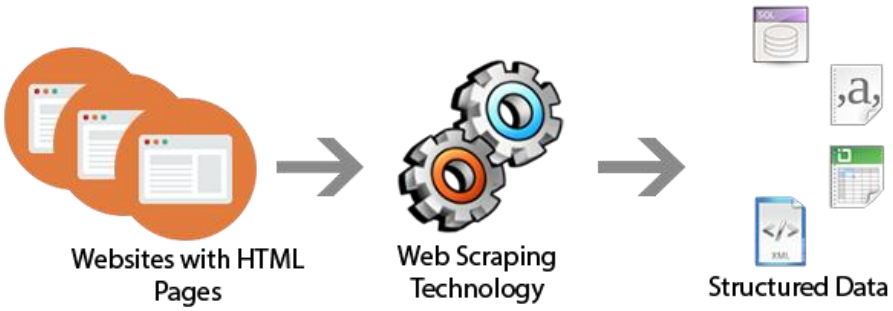

Figura C. 1. Scraping [8]

Scraping como alternativa a API

Una API (interface de programación de aplicaciones) es una herramienta que nos permite intercambiar datos entre webs. Normalmente existe un suministrador de los datos, a través de la API, que cobrará por ello.

Si para los datos que buscamos no se dispone de API o buscamos eliminar el coste de compra de los datos, podemos utilizar scraping para buscarlos de otros sitios web.

3.2. Sistemas de Recomendación o Predicción

Los sistemas de recomendación [9] [10] son una parte importante del ecosistema de la información y comercio electrónico. Representan un método poderoso que permite a los usuarios filtrar grandes espacios de información y productos.

Años de investigación sobre filtros colaborativos han llevado a un conjunto variado de algoritmos y una rica colección de herramientas para evaluar su desempeño.

Una implementación correcta debe comenzar con un análisis cuidadoso de los posibles usuarios y sus objetivos.

Un sistema de recomendación o predicción ofrecerá valoraciones para productos o servicios que el usuario aún no ha consumido o valorado.

Por lo tanto, cualquier sistema de filtrado nos lo podemos imaginar como una matriz (productos, usuarios) donde se guarda la valoración de los usuarios a los productos. Algunas celdas tienen un valor, que ha sido asignado por los propios usuarios. Y, será el filtro el encargado de predecir los valores [producto, usuario] vacíos.

Esta visión del problema es especialmente concreta en los sistemas filtrado colaborativo.

Y es ahí, donde esa predicción se puede convertir en una recomendación. Si predecimos que un producto es muy interesante para un usuario (y que él aún no lo sabe) estaremos interesados en ofrecérselo.

En el ejemplo de la figura C.2. disponemos de una serie de valoraciones del 1 al 5 y los puntos representan los productos sin valorar.

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	2	•	•	2	•	3	4	•	4	1	•
U2	•	•	1	5	•	5	3	1	•	5	2	1
U3	1	•	•	2	•	1	•	3	4	•	•	•
U4	•	1	4	4	•	•	3	•	5	4	•	1
U5	2	•	5	•	1	•	1	•	•	•	2	1
U6	•	•	5	2	1	•	•	4	•	1	•	2

Figura C. 2. Ejemplo de tabla (usuarios, productos) [11]

Entre los distintos sistemas y algoritmos de filtrado se han elegido tres:

- Filtro colaborativo basado en modelos.
- Filtro colaborativo basado en memoria de usuarios.
- Filtro colaborativo basado en memoria de productos.

A continuación, se ofrece algunos detalles de los algoritmos utilizados en el proyecto para construir los filtros. Aún utilizando el mimo nombre el algoritmo puede variar sustancialmente.

Filtro colaborativo basado en modelos.

Se basa para calcular las predicciones utilizando regresión lineal.

Sea:

- u número de usuarios
- m número de productos
- y matriz de valoraciones de usuarios a productos
- $y^{(i,j)}$ valoración del usuario j para el producto i
- n número de características de un producto,

  donde cada producto es representado por el vector (n+1) características.
- $\theta^{(j)}$ vector de preferencias del usuario j

Para saber los vectores de preferencias de todos los usuarios minimizaremos la siguiente función objetivo:

$$\theta^{(1)}, ..., \theta^{(u)} min \frac{1}{2} \sum_{j=1}^{u} \sum_{i:r(i,j)=1} \left(x^{(i)}(\theta^{(j)})^T - y^{(i,j)}\right)^2 + \quad \frac{\lambda}{2} \sum_{j=1}^{u} \sum_{k=0}^{n} (\theta_k^{(j)})^2$$

Figura C. 3. Función objetivo a minimizar en filtro basado en modelos

Filtro colaborativo basado en memoria de usuarios.

Basado en los gustos o decisiones de los usuarios.

Si nuestros gustos sobre un conjunto de datos han sido similares a los de otro usuario será muy probable que para un producto que desconozco tenga también gustos similares a los suyos.

Si pensamos en buscar dos usuarios parecidos podemos utilizar algoritmos de vecindad y para definir la similitud entre dos usuarios utilizaremos algoritmos de correlación como Correlación de Pearson.

Sea:

- r matriz de valoraciones de usuarios a productos
- $r^{(u,j)}$ valoración del usuario u para el producto i

Para nuestro caso concreto se ha utilizado:

- Coeficiente de Pearson:

$$sim_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)\left(r_{u,j} - \bar{r}_j\right)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

Figura C. 4. Coeficiente de Pearson

- Predicción para el usurario u sobre el producto i:

$$\hat{r}_{u,i} = \overline{r}_u + \frac{\sum_{v \in U} (r_{v,i} - \overline{r}_v) sim_{u,v}}{\sum_{v \in U} |sim_{u,v}|}$$

Figura C. 5. Formula predicción filtro basado en usuarios

Filtro colaborativo basado en memoria de productos.

Basado en la similitud de valoraciones de los productos. Si los usuarios valoran similarmente un producto, sería lógico pensar que un nuevo usuario lo valorase de forma parecida.

Sea:

- r matriz de valoraciones de usuarios a productos
- $r^{(u,j)}$ valoración del usuario u para el producto i

Para nuestro caso concreto se ha utilizado:

- Coeficiente de Pearson. Igual que para filtro colaborativo basado en memoria de usuarios.

- Predicción para el usurario u sobre el producto i:

$$\hat{r}_{u,i} = \frac{\sum_{j \in I} r_{u,j} sim_{i,j}}{\sum_{j \in I} |sim_{i,j}|}$$

Figura C. 6. Formula predicción filtro basado en usuarios

D. Técnicas y herramientas

En la presente sección se detallan las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.

4.1. Metodología ágil - Scrum

- **Motivación**: Técnica aprendida y utilizada durante la carrera.

Scrum[12] es el nombre con el que se denomina a los marcos de desarrollo ágiles.

Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de proyectos, caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapar las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.

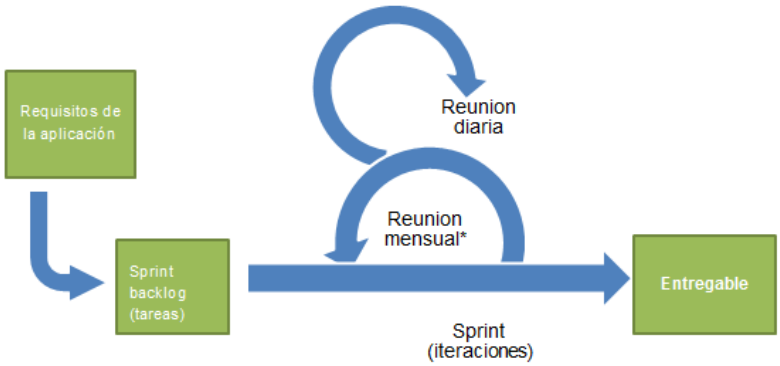

Figura D. 1. Scrum

4.2. Patrón de diseño – Model-View-Presenter (MVP)

- **Motivación:** Especialmente interesante para desarrollo de aplicaciones web en Python utilizando Flask.

MVC [13] es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

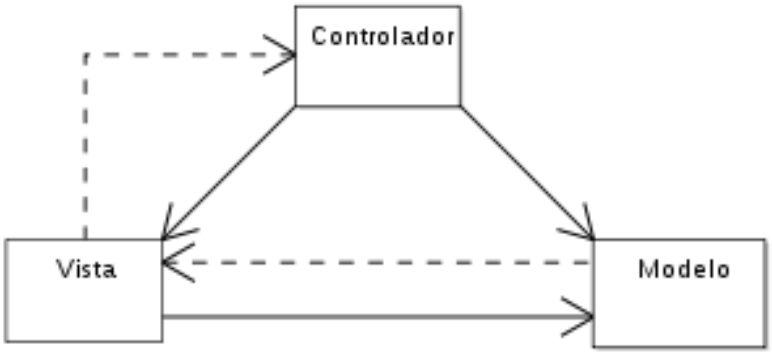

Figura D. 2. Modelo MVC [14]

Modelo MVC en la aplicación

La forma práctica de su implementación con Flask ha sido:

- **Modelo**. Paquetes encargados de la lógica de la aplicación.
- **Vista**. Templates en formato HTML.
- **Controlador**. Para el caso de Flask es el propio framework el que gestiona el controlador.

4.3. Control de versiones

- **Opción elegida**: Git - GitHub
- **Motivación:** Aplicación utilizada durante la carrera.

Git [15] es un sistema de control de versiones distribuido, diseñado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

GitHub [16] es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.

Además, la utilización de GitHub para desktop ha ayudado a que la actualización del repositorio sea mucho más sencilla.

Una peculiaridad de la utilización de Git es que, aunque se ha utilizado desde el principio, luego fue una imposición para poder trabajar con Heroku (se trata más adelante la utilización de Heroku)

4.4. Gestión de proyecto

- **Opción elegida**: ZenHub
- **Motivación**: Aplicación utilizada durante la carrera.

ZenHub [17] es la única herramienta de gestión de proyectos que se integra de forma nativa dentro de la interfaz de usuario de GitHub.

Nos permite ver las issues en forma de tarjeta sobre un tablero canvas y administrar su momento de desarrollo. Además, nos proporciona interesantes informes y gráficos.

4.5. Calidad del código

- **Opciones consideradas**: SonarQube, Codacy
- **Opción elegida**: Codacy
- **Motivación**: Incorporación de informes de cobertura.

Tanto SonarQube como Codacy no tienen integrada la cobertura de código para Python en sus análisis. Para poder incluirlo la forma es similar, se debe utilizar alguna aplicación que analice la cobertura y genere un informe en formato .xml, que será enviado a SonarQube o Codacy.

Para la naturaleza y extensión del proyecto, tanto SonaQube como Codacy resultan excelentes herramientas, aunque es cierto que SonrQube ofrece informes y gráficas más detalladas. Ambos nos ofrecen de una forma clara el estado de la calidad del proyecto como código duplicado, complejidad ciclomática,

La elección de Codacy se ha debido a la posibilidad de incorporar los informes de cobertura.

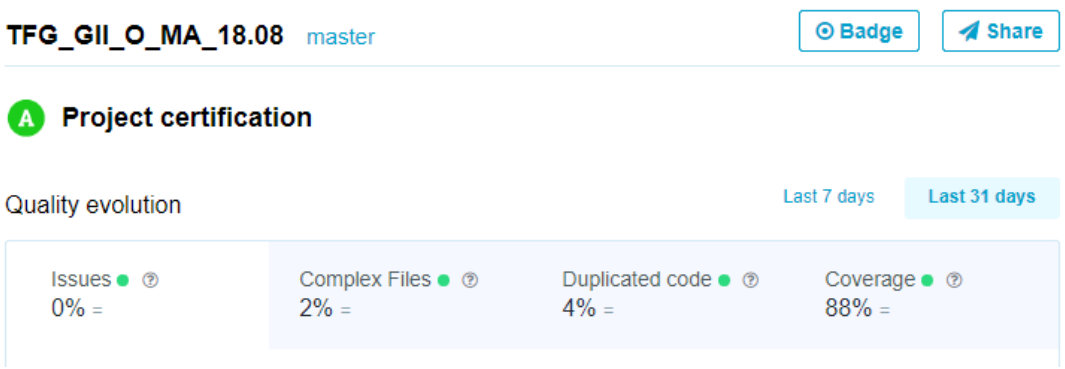

Figura D. 3. Informe de Codacy

Figura D. 4. Links a calidad Codacy desde GitHub

4.6. Lenguaje de programación

- **Opción elegida**: Python
- **Motivación**: herramienta impuesta.

4.7. Módulos y Paquetes

Se mencionan algunos módulos de Python utilizados y que son de especial interés.

Jellyfish

Jellyfish es una librería de funciones para calcular la distancia fonética entre cadenas de caracteres. Utilizada en la función de búsqueda de juegos.

Gunicorn

Gunicorn [18] es un servidor HTTP WSGI (Web Server Getaway Interface) de Python compatible con Flask.

Coverage

Coverage [19] es un módulo de Python que mide la cobertura de código durante su ejecución, con la posibilidad de emitir informes en formato .xml.

4.8. Cobertura del código

- **Opción elegida**: Unittest
- **Motivación**: Sencilla utilización. Permite el crear una clase de pruebas en nuestro código pasándole unittest.TestCase.

4.9. Framework Python web

- **Opciones consideradas**: Flask, Django, Pyramid
- **Opción elegida**: Flask

- **Motivación**: Aconsejable para pequeños proyectos web. Excelentes tutoriales.

A la hora de elegir un framework para diseño web es importante en pensar en el tamaño, complejidad y estructura del proyecto. En nuestro caso, diseñamos el sitio web de Retrogaming Recommender, y aunque puede tener alguna complejidad los algoritmos internos (filtros de recomendaciones y selecciones de ofertas) la estructura de la web no es extremadamente complicada.

Un framework[20] web es una colección de paquetes o módulos que permiten a los desarrolladores escribir aplicaciones web o servicios sin tener que manejar detalles de bajo nivel como los protocolos. El desarrollador se conectará al framework delegándole las labores de bajo nivel, centrándose en la lógica de la aplicación.

Generalmente un framework integrará las siguientes actividades:

- Interpretación de solicitudes. Obtención de parámetros de formularios, manejo de cookies y sesiones.
- Producción de respuestas. Presentación de datos como HTML o en otros formatos.
- Almacenamiento de datos en forma persistente.

Actualmente muchos frameworks ofrecen componentes que permiten modular su uso e instalación.

A continuación, se explican a grandes rasgos las características principales de los frameworks propuestos y analizados y el motivo de elegir a Flask.

Todo parte del planteamiento inicial:

- La aplicación puede tener una cierta complejidad en su lógica (filtros de recomendación y selección de ofertas) que no dejan de ser algoritmos en Python que podrían funcionar en cualquier interface.
- La aplicación no requiere una gran complejidad de diseño web.
- No buscamos el framework más completo sino el que mejor se adapte a nuestras necesidades. Y por lo tanto buscamos que:
    - Poder empezar a desarrollar el sitio web de la aplicación rápidamente.
    - Disponer de manuales y tutoriales que resulten amigables y comprensibles.

Django

Probablemente sea el mayor framework web basado en Python. Tiene una gran comunidad y activa. Dispone de su propio lenguaje de plantillas.

Potente pero considerado más complicado de iniciarse en él que Flask.

Flask

MicroFramework basado en Werkzeua, Jinja 2.

- Jinja: Uno de los motores de plantillas más utilizados para Python.
- Werkzeua: Biblioteca de aplicaciones web WSGI.

Flask es simple hasta en su propio sitio web oficial, donde da la impresión que estás en un pequeño proyecto, con varias reseñas y enlaces a su repositorio en GitHub.

Otro ejemplo de sus "intenciones" como herramienta lo tenemos en la frase de bienvenida en su sitio web: *"Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!"*

No dispone de una gran cantidad de módulos y de complicaciones iniciales para empezar con el diseño de nuestro sitio web. Esto que podría parecer un problema (posiblemente lo sea para grandes aplicaciones) en nuestro caso es una virtud. Dispone tan sólo de unas pocas extensiones, pero han sido suficientes para nuestra aplicación.

Todo empieza con una página en blanco. En unos pocos minutos se puede tener la ilusión de ver que la página web está empezando.

Al igual que con otras herramientas utilizadas un motivo importante para su elección ha sido los excelentes tutoriales encontrados. Podemos encontrar un buen tutorial tanto en el sitio oficial de Flask [21] como en otros sitios web [22].

Pyramid

Al igual que Flask se ofrece como un pequeño framework capaz de ofrecer el inicio rápido de aplicaciones web.

Se ha descartado Pyramid al no entender como práctico sus sistema de ayuda y tutoriales de iniciación, así como no encontrar tutoriales profesionales por parte de terceros.

4.10. Despliegue de la aplicación

- **Opciones consideradas**: Heroku, Google Could Platform
- **Opción elegida**: Heroku
- **Motivación**: Facilidad de uso y excelente tutorial.

Heroku es una buena opción para poder hacer las primeras pruebas de nuestra aplicación en la web. Es fácil de utilizar, de entender y de instalar, gracias a su excelente tutorial. El servicio básico es gratuito, aunque para aplicaciones profesionales está seriamente limitado.

Para un futuro en el que se pretenda la utilización de la aplicación a nivel profesional se recomienda estudiar opciones de hospedaje, aunque sean de pago.

Por otro lado, aunque Google Could Platform se autodefine como una forma rápida y sencilla de trabajar, cualquier acción es complicada de entender. Incluso es difícil de saber el último coste económico de sus servicios.

4.11. Scraping

- **Opciones consideradas**: Selenium, Webscaper, Import.
- **Opción elegida**: Selenium
- **Motivación**: herramienta impuesta.

Al ser una parte importante en el proyecto se ha decidido investigar en algunas otras aplicaciones de scraping, aunque Selenium era una herramienta elegida desde el principio.

Selenium

La gran ventaja de Selenium [23] [24] para Python es que trabajaremos directamente a nivel de código.

Webscaper

Webscaper es un puling para Google Chrome, lo que hace más fácil su utilización.

Import

En el caso de Import aún facilita más el scrapring ya que nos permite hacer scraping sin conocimientos de programación. Pero todas estas ventajas si carecemos de conocimientos técnicos pueden llegar a ser una desventaja, al no poder operar con tanta precisión como nos ofrece Selenium. Además, debemos considerar que estas herramientas, según su funcionalidad, pueden llegar a ser de pago.

4.12. Gestor de referencias bibliográficas

- **Opción elegida**: Zotero
- **Motivación**: De sencilla utilización e integrado en Google Chrome.

Zotero [25] es un gestor de referencias bibliográficas, libre y abierto. Su funcionamiento se basa en los siguientes principios:

- Recopilar.
- Organizar.
- Citar.
- Sincronizar.
- Colaborar.

E. Aspectos relevantes del desarrollo del proyecto

En la presente sección se explicarán aquellos aspectos se han considerado más relevantes para llevar a buen fin el proyecto. No se pretende volver a explicar la sección anterior de herramientas y metodología.

Se explicarán los problemas afrontados, las decisiones adoptadas y las conclusiones a las que llega.

5.1. Formación

Situación

A lo largo de la carrera se ha dado mucha importancia a formar una capacidad lógica que nos permita el desarrollar algoritmos con una cierta facilidad. Además, se nos ha entrenado en buscar la calidad en el diseño y desarrollo del software.

No obstante, los conocimientos adquiridos en diseño web han sido casi nulos.

Por eso, la parte de lógica (que podría representar un 80% de la definición del proyecto) ha resultado mucho más fácil que la parte de la vista (el otro 20% restante)

Como ejemplo mencionaremos las técnicas de scraping, que lo incluiremos en lógica y programación. Aunque no habíamos visto nada durante la carrera, estábamos preparados para su rápida comprensión y no ha exigido un excesivo esfuerzo el poder comprender las partes implicadas y como desarrollarlo correctamente para nuestro objetivo.

Por otro lado, ha estado toda la parte que se podría relacionar con el diseño web, no afrontada durante la carrera y siendo un mundo enorme. Para esta pequeña parte del total del proyecto ha sido necesario el dedicar un gran esfuerzo.

Evidentemente no ha sido necesaria sólo formación suplementaria en diseño web. Ya se ha hablado del scraping, a lo que habría que añadir módulos Python para problemas concretos, pruebas de cobertura, aplicaciones de control de calidad, etc.

Decisión adoptada

Buscar la forma más eficiente de formarse en las materias o aspectos necesarios.

Desde un principio se ha visto que la forma de recibir la formación suplementaria necesaria es tan importante o más que las herramientas a elegir.

Por eso, a la hora de elegir una herramienta o buscar una solución a un problema me he centrado en que los manuales y tutoriales disponibles fuesen comprensibles y se centrasen en la situación concreta del proyecto.

Conclusión

El mundo tecnológico cambia a una velocidad tan alta que es difícil el poder estar al día de todos los detalles.

La formación continua es una parte esencial de un buen profesional tecnológico.

Se debe ser capaz de sintetizar el problema o la necesidad, analizar las ofertas formativas que se disponen y elegir la que ofrezca mejor solución para la necesidad actual.

No sorprende que en este caso se cumpla la regla del 80/20 o ley de Pareto. Se ha necesitado un 20% del esfuerzo para conseguir el 80% del proyecto y para realizar el 20% restante del proyecto se ha necesitado un 80% del esfuerzo.

5.2. Establecimiento de requisitos

Situación

Se parte de la definición del proyecto: *"El objetivo del TFG es implementar uno u varios sistemas de recomendación (basado en modelos, filtros colaborativos, etc) con el fin de recomendar juegos antiguos en función de las valoraciones de los usuarios sobre juegos previos"*

En primeras reuniones se define:

- Los filtros a utilizar.
- El sitio web sobre el que se realizará el scraping para nutrirnos de los juegos.
- Generación de 1000 clientes ficticios con 100 valoraciones cada uno. Necesario para poder hacer funcionar los filtros de recomendación.
- Se deja a libre decisión las herramientas para desarrollar el sitio web del proyecto.

El problema personal fue el dar por sentado que con esto se terminaban la descripción de los requisitos.

Aún es más, en un ejercicio de autocrítica, ni siquiera describí de manera profesional los requisitos y casos de uso. Pensé que lo tenía claro en mi mente y que con eso era suficiente. Gravísimo error.

Por eso, cuando aparecen nuevas funcionalidades se debe reestructurar la lógica de la aplicación para darles cabida.

Las nuevas funcionalidades fueron:

- Mayor definición de forma de valorar un juego por parte de un cliente.
- Mayor definición de datos estadísticos propios de Retrogaming Recommender a ofrecer en la visión de un juego.
- Nuevas selecciones de ofertas de juegos, no basadas en predicciones sino en estadísticas del origen de los datos.
- Búsqueda simple.
- Búsqueda avanzada.
- Cambio de idioma.

Como una pequeña muestra nos centraremos en el cambio de idioma. No es una gran implementación; puede tener varios planteamientos y todos relativamente fáciles de desarrollar.

La forma elegida fue un diccionario de diccionarios donde cada uno representaba a un idioma. El idioma de cada sesión está guardado en una variable de sesión. Después utilizando el formato de las plantillas de Jinja, tan sólo hay que cambiar la frase por el diccionario[frase].

Pero había alguna frase que no estaba en la plantilla Jinja (no dependía de la capa de Vista) sino que estaba en la lógica (capa Modelo). Por lo que hubo que buscar una solución a este inconveniente.

Esta perdida de tiempo, posibilidad de errores y de generación de código no muy elegante se hubiese evitado si hubiese dedicado mucho más tiempo en detallar los requisitos.

Decisión adoptada

Desarrollar las nuevas funcionalidades y modificar la aplicación en la parte que corresponda.

Conclusión

Importancia de la planificación en el desarrollo de software.

Es mejor dedicar tiempo a planificar como se hará algo desde el principio que como hay que modificarlo.

Hay fases en el desarrollo del software que, aunque parezcan inservibles (en este caso la planificación – establecimiento de requisitos) son imprescindibles y asegurarán la futura tranquilidad.

Es imprescindible disponer requisitos bien definidos, consensuados con el cliente y preferiblemente cerrados. Si no vemos claro (tanto nosotros como nuestro cliente) que es así sería mejor no continuar.

Hay un antiguo proverbio que define esta realidad: “*No es del veloz la carrera”*.

5.3. Hosting de la aplicación

Situación

El que el proyecto está pensado para recoger datos de la web (scraping) y servirlos en web podría estar planteado para su desarrollo actual en local accediendo a través de localhost. No obstante, se planteó el que se hiciese disponible en la web alojándolo en un servidor.

Como se ha mencionado en la sección anterior se decidió el alojarlo utilizando los servicios gratuitos de Heroku.

El hospedaje del proyecto web desarrollado, al estar desarrollado en Python, debe de tener en cuenta que se deberá instalar en el servidor tanto Python como lo módulos que vayan a utilizar.

Para ver detalles de hospedaje en Heroku puede consultar el anexo D. Documentación Técnica de Programación, D.3 Manual del programador.

Situación sobrevenida

Llegado a este punto se ve que la aplicación es accesible correctamente en el servidor de Heroku y que tiene un funcionamiento correcto (aunque más lento que en local)

Se realizan pruebas de sistema al acceder con diferentes sesiones. El funcionamiento es correcto en lo que concierne a las sesiones; diferentes usuarios acceden independientemente al mismo tiempo.

Pero se detecta un problema en las selecciones ofrecidas. Aunque se mantenía correctamente la variable de sesión se mezclaban selecciones de listas de juegos entre ellos. Después de analizar el problema se entiende que se debe a que se guarda como variable global una lista con la selección de juegos y que por la estructura de gunicorn en Flask puede llegar a dar problemas.

Decisión adoptada

Se analizaron dos diferentes opciones:

- **Ajustar los parámetros de gunicorn**

Recordamos que gunicorn es uno de los módulos que le pasamos a Heroku en el fichero requiriments.txt con los módulos que debe instalar.

Gunicorn es el servidor WSGI HTTP para Python que se ha utilizado en el proyecto. Nos permite administrar las peticiones simultaneas que nuestra aplicación reciba, con la posibilidad de limitar el número de ellas.

Este dinamismo en el número de conexiones se suele utilizar si queremos optimizar el número de conexiones al número de CPUs disponibles. En nuestro caso lo utilizaríamos para que no se mezclen las conexiones limitándola a una única.

Para nuestro caso en concreto tendríamos que haber modificado el fichero Procfile que es donde se ejecuta gunicorn de la siguiente forma, donde se limita el número de "workers" a 1:

```
gunicorn -w 1 retrogaming:app
```

Funciona correctamente, desaparece el problema, pero no es la opción adoptada por las limitaciones que ofrece.

- **Eliminar la variable global de selecciones**

Finalmente, esta fue la opción elegida.

Aunque requiere un mayor esfuerzo de tiempo y programación la solución es más profesional. No limitamos el número de accesos, que iría en contra del espíritu de hacer la aplicación disponible en la web.

Conclusión

Una vez más vemos lo importante de la planificación y la especificación correcta, detallada y definitiva de requisitos.

Importancia de utilizar un sistema de control de versiones (en este caso Git). Puede ser requisito para otras aplicaciones.

La aplicación es mejorable en el sistema elegido de almacenamiento de datos y su acceso. Se plantea posteriormente como desarrollo futuro.

5.4. Control de calidad del código

Situación

El proyecto se ha desarrollado en Python sin utilizar ningún IDE.

Inicialmente se utilizó SonarQube para realizar el seguimiento de la calidad de código, modificando aquellos detalles que aparecían como mejorables para conseguir la máxima calificación.

SonarQube es un muy buen programa de control de calidad de código, que permite el disponer de los indicadores de calidad de una forma clara y de diferentes gráficos de interés.

Desde un principio se desarrollan las pruebas unitarias de cobertura a la vez que se desarrolla el código.

Pero se detecta que, a diferencia de proyectos desarrollados en otros lenguajes, para Python no está automatiza los informes de cobertura en SonarQube.

El usuario debe utilizar una herramienta de informe de cobertura que emita un informe en formato .xml y reportarla SonarQube.

Según la documentación de SonarQube sobraría con generar los informes de cobertura de la siguiente forma [26]:

```
coverage erase

coverage run --branch --source=<python packages> <program>

coverage xml -i
```

Y modificar el fichero de propiedades de sonar de la siguiente forma:

```
sonar.python.coverage.reportPaths=coverage.xml
```

No obstante, no se consigue ningún resultado.

Decisión adoptada

Se decide cambiar de herramienta de control de calidad del código a Codacy.

En realidad, Codacy plantea el mismo problema, con Python no automatiza la cobertura y debe ser generada y enviada.

Aunque en el caso de Codacy [27] debemos instalar un módulo propio de Python que será el encargado de enviar la cobertura.

```
pip install codacy-covarage
```

Configuramos la variable del sistema CODACY_PROJECT_TOKEN, con el token del proyecto.

```
set CODACY_PROJECT_TOKEN=%Project_Token%
```

Por último, enviamos el informe de cobertura a Codacy

```
python-codacy-coverage -r coverage.xm.
```

Se revisa el informe de Codacy y está correcto. Sí que incorpora los informes de cobertura.

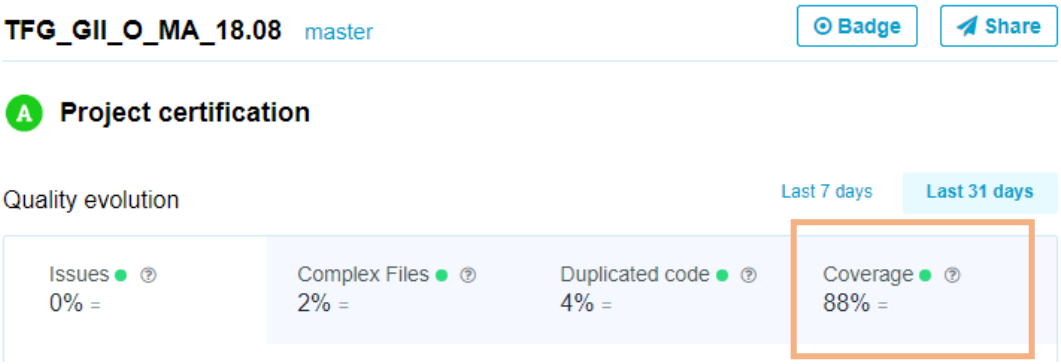

Figura E. 1. Informe de cobertura de Codacy

Figura E. 2. Links a cobertura Codacy desde GitHub

No obstante, se observa una situación peculiar. Pasado un tiempo desaparece del Dashboard de Codacy el informe de cobertura aunque mantiene el "Codacy Badge" para cobertura incluido en GitHub.

Conclusión

Sobre los sistemas de control de calidad

Aunque con el mismo espíritu cada sistema de control de calidad puede marcar issues diferentes poniendo acento en diferentes detalles.

Es bueno el disponer de algún sistema de calidad de código y utilizarlo desde el principio. Esto nos permitirá el eliminar defectos que de otra forma se podrían arrastrar al resto del código

Sobre el lenguaje de programación e IDEs

Llegados a este punto se aprecia que este problema (integración de cobertura de código) no hubiese surgido si se hubiese utilizado otro planteamiento como Java con Eclipse.

Ya no sólo hubiese estado integrado de una forma natural, sino que Eclipse dispone de extensiones que nos permiten ver directamente el grado de cobertura como EclEmma.

5.5. Informes de Cobertura en Python

Situación

Dada la falta de integración de los informes de cobertura en los sistemas de control de calidad del código, y siendo para ambos necesario el generar un informe de cobertura en formato .xml, se busca el que resulte más adecuado.

Decisión adoptada

Tanto SonarQube como Codacy sugieren la generación de informes de cobertura con coverge [28]. Por lo tanto, se instala y se prueba su funcionamiento.

Resulta sencillo de utilizar y práctico. Dando la opción de ofrecer informes en .xml o html.

Conclusión

Existen alternativas menos automáticas que los IDE pero que también pueden ser prácticas y sencillas.

5.6. Aplicación multi idioma.

Situación

Se plantea el que la aplicación pueda ser ofrecida en diversos idiomas.

Decisión adoptada

Se estudian dos opciones:

- Utilizar algún servicio de traducción instantánea.
- Utilizar una estructura de datos que nos permita ofrecer la traducción.

Servicio de traducción instantánea

Los principales servicios de traducción son Google Could Traslation API y Microsoft TRanslator Text API.

Aunque ambos son de pago Microsoft ofrece una opción de bajo volumen de entradas gratis.

Esta opción nos podría ayudar a automatizar la traducción del sitio web pero plantea un problema, dejaríamos la traducción total en manos del traductor, por lo que en un momento dado podría aparecer alguna traducción errónea que podría dar una imagen ridícula a nuestro sitio.

Aunque no sea la mejor opción para nuestra situación, se reconoce que puede llegar a ser una excelente para sitios con mucho texto o texto incluido por terceros (siempre que se hagan las advertencias pertinentes que es un texto traducido automáticamente)

Estructura de datos que nos permita ofrecer la traducción

Esta es la opción elegida.

El encargado de su gestión es el paquete internalización situado en (src/app/internalizacion). El paquete incluye una función que recibido un idioma devolverá un diccionario con todos los textos necesarios en nuestro sitio para ese idioma.

A nivel de estructura de datos lo que tenemos es un diccionario de diccionarios, o un diccionario con los diccionarios para cada idioma.

Se guarda en variable de sesión el idioma en el que está configurado el sitio web.

En el caso de que no exista la variable de sesión se configura en español.

Cuando se solicita un cambio de idioma se cambiará el diccionario de texto al idioma elegido y se modificará la variable de sesión con el identificador del idioma.

Figura E. 3. Selección de idioma

Conclusión

No siempre la forma más “adelantada” o complicada será la mejor. Necesitamos una solución práctica para nuestra situación en particular.

5.7. Búsqueda avanzada

Situación

Se ha mencionado en el punto 5.1 que, desde el punto de vista personal, un reto de este proyecto ha sido el equilibrar la diferencia de formación entre los conocimientos de programación con los de diseño. Se menciona la implementación de búsqueda avanzada como uno de los muchos casos que se ha dado esta situación.

Se propone el realizar una búsqueda avanzada donde un usuario pueda modificar el rango de búsqueda de varios parámetros.

Desde la programación una buena práctica sería el poder limitar esa búsqueda según el mínimo y el máximo de un parámetro.

Por ejemplo, pensemos que queremos centrarnos en el número de visitas que han tenido los juegos. En este caso ficticio el juego que menos visitas ha tenido es 5 y el que más ha tenido es 32. Lo normal sería que el usuario pudiese ver desde un principio que el rango de selección está entre 5 y 32.

Pero ahora viene el reto de ofrecer al usuario esta información y la posibilidad de configurar su búsqueda (o elegir los rangos de búsqueda) de una forma agradable. Es aquí donde entra la parte de diseño.

Decisión adoptada

Html nos ofrece varias opciones para poder introducir datos en un formulario, con el gran inconveniente que para todas necesitaremos una para el mínimo y otras para el máximo:

- Etiqueta <input type=”text”>. Deberíamos controlar con JavaScript que los valores están dentro del rango aceptado e informar al usuario en caso de error en su cumplimentación.
- Etiqueta <input type=”number”>. Permite establecer mínimo y máximo. Pero pensemos que debemos rellenar el límite inferior y superior de una búsqueda, por lo que necesitaríamos dos campos con mínimo y máximo. Un campo del formulario con el límite inferior (con su mínimo y su máximo) y un campo límite superior (con su mínimo y con su máximo). Tan pronto introducimos un valor en uno de los campos cambiaría el mínimo o el máximo del otro. Por ejemplo, si deseo centrar la búsqueda en número de

vistas y los límites están en [12, 14781], los campos a introducir serán min_vistas [12, 14781] y max_vistas con [12, 14781]. Si ahora introduzco en min_vistas el valor 200, debería cambia max_vistas a [200, 14781]. Esto se controló con JavaScript, aunque se ve que estéticamente no es la mejor opción.

- Etiqueta <input type=”range”>. Un deslizador sería la mejor opción estética, pero html ofrece sólo la opción de deslizadores unidireccionales. Esto plantea el mismo problema que con la opción anterior, podemos marcar mínimos y máximos, pero se debe controlar su sincronización con JavaScript. Esta solución, inicialmente más estética, añade un efecto óptico de ver como se meuve el deslizador para el que se actualiza su mínimo y máximo que puede llegar a despistar al usuario por no entender su funcionamiento.

Figura E. 4. Deslizador Html “range”

Se piensa en cómo conseguir un deslizador bidireccional. Ya hemos comentado que html no lo ofrece como opción input, pero tenemos dos opciones:

- Crearlo con JavaScript.
- Importarlo de algún fichero externo creado por terceros. Esta ha sido la solución adoptada, importándolo desde

  https://cdnjs.cloudflare.com/ajax/libs/noUiSlider/13.0.0/nouislider.js

Figura E. 5. Búsqueda avanzada con deslizadores bidireccionales

Conclusión

Nuestra aplicación de cara al cliente no deja de ser una de ocio. Por lo tanto, debemos de tener en cuenta los detalles estéticos. Es tan importante la presentación como el funcionamiento interno.

Volvemos a pensar en el punto inicial de esta sección, lo importante que es la formación continua y estar abiertos a buscar nuevas y mejores formas de realizar el trabajo.

H. Trabajos relacionados

Cada vez son más los sitios web que utilizan sistemas de recomendación predictivos para ofrecer contenido personalizado a los clientes.

A continuación, resaltaremos como es utilizado por gigantes de internet, a la vez que podremos ver un ejemplo de como la recomendación de contenidos puede ser una ventaja estratégica desde un principio.

6.1. Amazon.

Amazon autodefine su sistema de recomendación [29] [30] como filtrado colaborativo ítem a ítem. Se trata de un desarrollo de la casa, patentado en los Estados Unidos, y que nació porque ninguna de las propuestas existentes por aquel entonces (principios de década) servían para grandes conjuntos de datos.

Asocia a cada producto comprado por un usuario con una lista de productos similares, que se obtiene en función de los elementos que hayan sido adquiridos en un mismo pedido, añadidos a un carrito de la compra, o almacenados en una wish list.

Este proceso puede llegar a ser extremadamente costoso en términos de computación. Pero Amazon se las ha arreglado para implementarlo y ejecutarlo de una forma muy eficiente, de modo que funciona aceptablemente bien incluso con conjuntos de datos enormes.

De interés:

- Siendo Amazon pionera en el comercio electrónico de grandes volúmenes de productos se ve obligada a desarrollar un sistema de recomendación ad hoc.

6.2. Netflix.

Según describe Netflix [31] el cálculo de la predicción se realiza teniendo en cuenta:

- Las interacciones del usuario con el servicio (como el historial de visualización y clasificación otros títulos)
- Otros usuarios con gustos y preferencias similares en el servicio.
- Información sobre los títulos, como su género, categorías, actores, año de estreno, etc.

Además de saber lo que se ha visto en Netflix para personalizar mejor las recomendaciones, también se analizan aspectos como:

- La hora del día en que se ve Netflix.
- Dispositivos utilizados para verlo.
- Durante cuánto tiempo se utiliza.

Por lo que vemos que realmente Netflix no utiliza un único sistema de recomendación, sino que combina varios algoritmos de recomendación con varios fines [32]:

- Personalized Video Ranker (PVR), que ordena el catálogo de vídeos (o subcatálogo por microgénero)
- Top-N Video Ranker, que produce las recomendaciones en la fila “Top Picks”
- Un algoritmo para la fila “Trending Now”: el algoritmo identifica los dos tipos de tendencias: la tendencia de tipo anual como los vídeos románticos en San Valentín, por ejemplo, y las tendencias “one-off”
- Un algoritmo de “Continúa viendo”
- Otro de similitud entre vídeos: “Because you watched” o BYW
- El algoritmo que prioriza las filas (Page Generation)
- Otros (de evidencia, de búsqueda, etc.)

De interés:

- Netflix a pasado de ser un pequeño negocio a un gigante gracias a su sistema de recomendaciones. Más de un 80% de sus visualizaciones son de contenido descubierto en la propia plataforma, lo que indica el éxito de su sistema de recomendaciones.

6.3. FilmAffinity

Resultará de interés FilmAffinity porque siendo una pequeña iniciativa se ha basado desde el principio en la recomendación de contenidos.

FilmAffinity [33] fue creado en Madrid en el año 2002. Desde un principio la página constaba de un sistema de recomendación de películas llamado "Almas gemelas", el cual mostraba las personas más afines en función de las puntuaciones que se dan a las películas. Tres años después se lanzó la sección de críticas, en donde los usuarios expresaban su opinión sobre una película.

Ya la descripción comercial del tipo de recomendación (“almas gemelas”) nos da una idea clara de que tipo de filtro se utiliza, uno de memoria de usuarios.

De interés:

- Pequeña iniciativa basada en las recomendaciones como factor diferenciador.
- Actualmente cuenta con datos de 9.257 clientes, 19.970 películas y 5.333.988 valoraciones, y se vanaglorian que su base de datos les permite calcular predicciones mejores que Netflix [34]

I. Conclusiones y líneas de trabajo futuras

En la presente sección se exponen las conclusiones del desarrollo del proyecto, así como de las posibles líneas de trabajo futuras que permitirán dar continuidad al mismo.

7.1. Conclusiones

Generales

El proyecto ha permitido sentir en la práctica conceptos aprendidos durante la carrera:

- Importancia de seguir un método de planificación y no saltarse los pasos importantes. Se ha comentado lo importante de fijar y consensuar claramente los requisitos con el cliente.
- Utilizar el control de versiones en un proyecto personal.
- Control de calidad de código y de pruebas de cobertura, que van a ayudar a disponer de un código testado, de confianza y fácil de continuar con él futuros desarrolladores.

Scraping

Además, ha permitido adentrarse de una forma práctica en posibilidades, personalmente desconocidas, como el scraping y sus múltiples utilizaciones.

Aunque es cierto que en el total del tiempo dedicado al proyecto ha representado una pequeña parte, ha sido una de las más interesantes por lo práctico que ha resultado ser y lo inspirador de multitud de futuras posibilidades.

En el caso del proyecto nos ha permitido el dar contenido a nuestro sitio web, con más de 10.000 juegos, desde el primer día.

No sólo se ha alimentado nuestro sitio web con contenido, sino que además lo hemos aligerado de la capacidad de almacenamiento que sería necesaria. Las imágenes que ofrecemos en nuestro sitio web no las tenemos almacenadas en nuestro servidor. Por medio de scraping hemos seleccionado la ruta en su servidor y las accedemos desde ahí.

Esto nos hace pensar en cuantas posibilidades tiene esta técnica de recopilación de datos. No olvidemos que la información es poder. Así como el poder unir toda la información obtenida por scraping con otros conceptos aprendidos durante la carrera como los relacionados a minería de datos.

Sistemas de recomendación

Los sistemas de recomendación están ya en nuestra vida. Se han implantado en la mayoría de las aplicaciones de comercio electrónico o de ocio y casi sin darnos cuenta. El mundo comercial está deseoso de conocer nuestros gustos para podérnoslo ofrecer.

Se ha podido apreciar como en la práctica un buen sistema de recomendación no es un algoritmo cerrado.

En nuestro caso concreto disponíamos de tres (modelos, usuarios y productos), algo que para un sitio web como el nuestro en un principio podría parecer algo excesivo. Hemos visto como sitios con muchos más productos y usuarios, como FilmAffinity, disponen de uno sólo. No obstante, para nuestro proyecto, está justificado el número de filtros de recomendación por su naturaleza didáctica.

En todo caso, es muy importante que las recomendaciones estén lo más cercanas posibles a los gustos y deseos de nuestros usuarios. Mejor dicho, de nuestro usuario, porque sí, el objetivo último del sistema de recomendación es personalizar lo máximo una recomendación o una predicción de gusto.

De ahí, entendemos lo importante de personalizar nuestro sistema de recomendación. Debemos conocer que características tienen nuestros productos y nuestros clientes y con ello adaptar el algoritmo de recomendación para llegar a la “formula mágica” de poder adelantarnos a un deseo.

7.2. Líneas de trabajo futuras

Interactuación con la aplicación y otros usuarios

La aplicación tendrá mayor valor cuanto más tiempo pasen los usuarios conectados a ella. Por eso, buscamos formas de que los usuarios puedan interactuar con ella y con otros usuarios.

En la versión actual, la única interactuación que se realiza es por medio de la valoración de los juegos, recomendaciones, listados y búsquedas. Pero sería sugerible el conseguir algo mucho más personal o humano.

Posibles opciones son:

- Permitir el comentar los juegos por los usuarios. Esto requeriría que cada juego tuviese relacionada una zona de comentarios donde los clientes participaran. Además, estaría permitiendo el responder cada comentario por el resto de la comunidad.
- Generar un servicio chat en la aplicación.
- Creación de campeonatos de juegos ofrecidos.
- Creación de retos personales.

- Establecer grupos de juego para crear retos por grupo.

Scraping

Generar una rutina que realice en segundo plano scraping cada cierto tiempo prefijado.

Usuarios

Permitir a los usuarios realizar labores básicas (bajas y modificaciones) de gestión de sus datos además del servicio de alta que actualmente se dispone.

Administración

Crear una interfaz de administración para los administradores del sitio web, en el que se les permita realizar labores propias de su cargo.

Hosting

Estudiar alternativas de pago al hospedaje actual.

Integración de jugar en la aplicación

Actualmente se permite el jugar dentro la aplicación incluyéndolo en un iframe. Esto puede suponer algunos inconvenientes de alertas de seguridad y de rendimiento. Por lo tanto, se propone el seguir estudiando la estructura del sitio web archive.org y de su almacenamiento de datos para poder incluir su sistema de juego en nuestra aplicación.

Bibliografía

[1] Ricci, F., Rokach, L., y Shapira, B., *Introduction to recommender systems handbook. In Recommender systems handbook.*, Springer, Boston, MA., 2011.

[2] Smith, B. y Linden, G, *Two decades of recommender systems at Amazon. com. Ieee internet computing*, 21(3), 2017.

[3] McSherry, F. y Mironov, I., *Differentially private recommender systems: Building privacy into the netflix prize contenders. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, s. f.).

[4] Covington, P., Adams, J., y Sargin, E., *Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems* (ACM, s. f.).

[5] Atindra Bandi, «Web Scraping Using Selenium — Python», Towards Data Science, 3 de octubre de 2018, https://towardsdatascience.com/web-scraping-using-selenium-python-8a60f4cf40ab.

[6] «Web scraping», en *Wikipedia, la enciclopedia libre*, 30 de octubre de 2018, https://es.wikipedia.org/w/index.php?title=Web_scraping&oldid=111655695.

[7] «Qué es el Web scraping? Introducción y herramientas», Sitelabs, 8 de abril de 2016, https://sitelabs.es/web-scraping-introduccion-y-herramientas/.

[8] Emmita, «WEB SCRAPING», *Emmita* (blog), 14 de octubre de 2017, https://medium.com/@Emmitta/web-scraping-7f87930face4.

[9] Michael D. Ekstrand, John T. Riedl, y Joseph A. Konstan, «Collaborative Filtering Recommender Systems», *Foundations and Trends® in Human–Computer Interaction* 4, n.º 2 (5 de mayo de 2011): 81-173, https://doi.org/10.1561/1100000009.

[10] Sergio Manuel Galán Nieto, «Filtrado Colaborativo y Sistemas de Recomendación», s. f., 8.

[11] «Sistemas de Recomendación basados en Filtrado Colaborativo (K-Vecinos)», *Jarroba* (blog), 24 de septiembre de 2013, https://jarroba.com/sistemas-de-recomendacion-basados-en-filtrado-colaborativo-k-vecinos/.

[12] «Scrum (desarrollo de software)», en *Wikipedia, la enciclopedia libre*, 11 de mayo de 2019, https://es.wikipedia.org/w/index.php?title=Scrum_(desarrollo_de_software)&oldid=115872313.

[13] «Modelo vista controlador (MVC)», accedido 14 de mayo de 2019, https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

[14] «Modelo–vista–controlador», en *Wikipedia, la enciclopedia libre*, 13 de abril de 2019, https://es.wikipedia.org/w/index.php?title=Modelo%E2%80%93vista%E2%80%93controlador&oldid=115244051.

[15] «Git», en *Wikipedia, la enciclopedia libre*, 11 de abril de 2019, https://es.wikipedia.org/w/index.php?title=Git&oldid=115198516.

[16] «GitHub», en *Wikipedia, la enciclopedia libre*, 6 de abril de 2019, https://es.wikipedia.org/w/index.php?title=GitHub&oldid=115084565.

[17] «ZenHub», GetApp, accedido 16 de mayo de 2019, https://www.getapp.es/software/110953/zenhub.

[18] «Standalone WSGI Containers — Flask 1.0.2 documentation», accedido 16 de mayo de 2019, http://flask.pocoo.org/docs/1.0/deploying/wsgi-standalone/.

[19] Ned Batchelder and 100 others, *coverage: Code coverage measurement for Python*, versión 4.5.3, OS Independent, Python, accedido 17 de mayo de 2019, https://github.com/nedbat/coveragepy.

[20] «WebFrameworks - Python Wiki», accedido 17 de mayo de 2019, https://wiki.python.org/moin/WebFrameworks.

[21] «Tutorial — Flask 1.0.2 documentation», accedido 17 de mayo de 2019, http://flask.pocoo.org/docs/1.0/tutorial/.

[22] «miguelgrinberg.com», accedido 17 de mayo de 2019, https://blog.miguelgrinberg.com/index.

[23] «Selenium with Python - Documentación de Selenium Python Bindings 2», accedido 16 de mayo de 2019, https://selenium-python.readthedocs.io/index.html.

[24] «ChromeDriver - WebDriver for Chrome», accedido 22 de noviembre de 2018, https://sites.google.com/a/chromium.org/chromedriver/.

[25] «Zotero», en *Wikipedia, la enciclopedia libre*, 11 de mayo de 2019, https://es.wikipedia.org/w/index.php?title=Zotero&oldid=115883403.

[26] «Python Coverage Results Import - Plugins - Doc SonarQube», accedido 17 de mayo de 2019, https://docs.sonarqube.org/display/PLUG/Python+Coverage+Results+Import.

[27] *Python coverage reporter for Codacy. Contribute to codacy/python-codacy-coverage development by creating an account on GitHub*, Python (2015; repr., Codacy, 2019), https://github.com/codacy/python-codacy-coverage.

[28] «Coverage.py — Coverage.py 4.5.3 documentation», accedido 17 de mayo de 2019, https://coverage.readthedocs.io/en/v4.5.x/index.html.

[29] G. Linden, B. Smith, y J. York, «Amazon.Com Recommendations: Item-to-Item Collaborative Filtering», *IEEE Internet Computing* 7, n.º 1 (enero de 2003): 76-80, https://doi.org/10.1109/MIC.2003.1167344.

[30] Manu Mateos, «Así funcionan las recomendaciones de Amazon», Genbeta, 31 de enero de 2014, https://www.genbeta.com/web/asi-funcionan-las-recomendaciones-de-amazon.

[31] «Cómo funciona el sistema de recomendaciones de Netflix», Centro de ayuda, accedido 17 de mayo de 2019, https://help.netflix.com/es-ES/node/100639.

[32] «El sistema de recomendación de Netflix», accedido 17 de mayo de 2019, https://www.linkedin.com/pulse/el-sistema-de-recomendaci%C3%B3n-netflix-in%C3%A9s-jim%C3%A9nez.

[33] «FilmAffinity», en *Wikipedia, la enciclopedia libre*, 19 de abril de 2019, https://es.wikipedia.org/w/index.php?title=FilmAffinity&oldid=115365324.

[34] «FilmAffinity», FilmAffinity, accedido 17 de mayo de 2019, https://www.filmaffinity.com/en/aboutsm.php.