



universidad  
de león



# **Escuela de Ingenierías**

## **Industrial, Informática y Aeroespacial**

**GRADO EN INGENIERÍA INFORMÁTICA**

**Sistemas de Información de Gestión y Business  
Intelligence**

**FIBONART**

Álvaro López-Brea Campo

16 de diciembre de 2020



# INTRODUCCIÓN

**FIBONART** nace como una aplicación web con el objetivo de ayudar al público a conocer arte clásico mediante la utilización de un sistema de recomendación propio.

A lo largo del documento se detallará qué problema trata de solucionar, qué herramientas se han utilizado para su desarrollo, cuál es su funcionamiento y características. Además, se incluirá la explicación a los algoritmos de recomendación implementados y que dan sentido al proyecto; así como diversos ejemplos de diferente naturaleza para poder comprender su funcionamiento.

Ya desde un punto de vista más general, podremos observar un análisis DAFO que trata de reflejar objetivamente la realidad de la aplicación, y concluiremos indicando las líneas de futuro y las lecciones aprendidas durante el desarrollo de FIBONART.



# ÍNDICE

|  |    |
|--|----|
| Introducción .....                               | 2  |
| 1. Descripción del problema .....                | 4  |
| 2. Herramientas.....                             | 6  |
| 2.1. Base de datos .....                         | 6  |
| 2.2. Frontend.....                               | 9  |
| 2.3. Backend.....                                | 10 |
| 2.4. Herramientas generales.....                 | 11 |
| 3. Aplicación .....                              | 12 |
| 4. Algoritmo de recomendación .....              | 30 |
| 4.1 Recomendación basada en contenido.....       | 31 |
| 4.2 Recomendación de filtrado colaborativo ..... | 34 |
| 5. Análisis de resultados .....                  | 37 |
| 6. Análisis DAFO.....                            | 42 |
| Fortalezas.....                                  | 43 |
| Debilidades .....                                | 45 |
| Amenazas .....                                   | 46 |
| Oportunidades.....                               | 46 |
| 7. Líneas de futuro .....                        | 47 |
| 8. Lecciones aprendidas.....                     | 49 |
| 9. Fuentes bibliográficas .....                  | 51 |



## 1. DESCRIPCIÓN DEL PROBLEMA

En tiempos de tecnología, donde se premia la viralidad y todo está inventado, resulta complicado encontrar un verdadero interés en el arte clásico. Sí, todos conocemos la *Mona Lisa* de Da Vinci o el *David* de Miguel Ángel, pero ¿cuántos sabrían responder a qué movimiento perteneció Caravaggio o qué técnica utilizaba Van Gogh?

Durante las escasas veces que se estudia sobre arte en los colegios, es poco habitual encontrar alumnos que realmente consideren interesante la materia, que estén dispuestos a aprender realmente más allá de saber reconocer *La Última Cena* o *La noche Estrellada*.

Los museos, a excepción quizá de aquellos que son una atracción turística en sí mismos, como el MOMA de Nueva York o el Louvre de París, o los que cuentan con alguna de las obras más importantes de la historia, como la Galería de la Academia de Florencia o el Museo Vaticano de Roma, suelen contar con una afluencia de visitas muy reducido, generando muy poco interés para la gran mayoría de la población.

Por ello, a pesar de que hoy en día el arte es accesible para toda la sociedad, resulta complicado encontrar una fuente única y suficiente de información. Como consumidor habitual de arte, y desde mi experiencia personal, mi fuente de conocimiento ha sido principalmente la información extraída de los libros, donde se recopilan una serie de obras, generalmente por escuela o por autor, y se hace una breve descripción de las técnicas utilizadas, pero al tratarse de información estática, que ofrece muy poca flexibilidad a la hora de buscar algo concreto, dificulta el aprendizaje y la posibilidad de explorar un aspecto concreto.

Habitualmente oímos eso de "en Internet se encuentra todo", "no hay mejor lugar para aprender que Internet", "en Internet está todo inventado". No seré yo quien contradiga estas afirmaciones, visto que



en la inmensa mayoría de las situaciones son completamente ciertas, pero sí que me gustaría resaltar la ausencia de un portal específico dedicado a aquellos que nos gusta el arte, donde podamos compartir valoraciones y gustos con otros usuarios y encontrar aquellas obras que cumplan con nuestro criterio y que de otra manera nunca hubiéramos encontrado. Dicho de otro modo, aunque existen cientos de sistemas de recomendación muy refinados para todo aquello imaginable para el ser humano (películas, series, música...), no existe ninguno sobre obras de arte, por lo que creí que era mi deber tratar de suplir esta ausencia. Así nace **FIBONART**, el primer portal interactivo de recomendación de obras de arte de todo Internet.

Para el proyecto, era necesario contar con un dataset de obras que proporcionara todo el conocimiento de arte que requería el sistema. Si es complicado encontrar información de calidad sobre arte, parecía imposible que hubiera publicada una base de datos con todo lo que buscaba y necesitaba. Tras buscar sin éxito durante horas, lo que me reafirmó la necesidad de crear FIBONART, llegué a la [Web Gallery of Art](#), el único portal que tenía algo de similitud con mi proyecto, ya que ofrece una base de datos de obras de arte, pero de forma completamente plana, sin ninguna interacción o sistema basado en sugerencias.

Así, para poder aportar algo que realmente tuviera una utilidad clara y determinada, vislumbré la necesidad de implementar uno o varios sistemas de recomendación de diversos tipos, de forma que el usuario no encuentre mera información estática, si no que sienta que nuestro proyecto es capaz de amoldarse a lo que realmente demanda, a sus gustos e intereses.




## 2. HERRAMIENTAS

La aplicación web está compuesta de 3 partes claramente diferenciadas, cada una habiendo sido desarrollada utilizando unos frameworks y lenguajes concretos.

### 2.1. BASE DE DATOS

Para poder crear una base de datos con la información suficiente para llevar a cabo una solución de calidad al problema, era necesario contar con un dataset lo más completo posible, que permitiera una manejabilidad en función de las necesidades del sistema. El dataset está extraído de la [Web Gallery of Art](#), un portal con datos sobre obras de arte. Sin embargo, el archivo .csv descargado no tenía el formato que deseaba para poder trabajar con la base de datos posteriormente con comodidad por lo que fue necesario aplicarle un tratamiento para acomodarlo a las necesidades de la aplicación, para lo cual utilicé:

- ◇  python™ : Uno de los lenguajes de programación más versátiles y utilizados del mundo, especialmente cómodo para la tarea de dar formato a un documento gracias a las librerías externas “csv” y “re”, que facilitaron el trabajo de limpieza del dataset. Entre las operaciones a realizar se encontraban cambiar el delimitador, eliminar obras repetidas, las entradas que no pertenecieran a los artistas más famosos de la historia, eliminar el tamaño de la obra que venía incluido en el campo “Técnica”, etc. Además, el segundo dataset del proyecto, compuesto por valoraciones de usuarios a determinadas obras, también está creado de forma íntegra en Python para su automatización.

```
import csv
with open('limpio.csv', newline='') as infile, open('limpio3.csv', 'w', newline='') as outfile:
    reader = csv.reader(infile, delimiter=';')
    writer = csv.writer(outfile, delimiter=',')
    row_prev = ""
    for row in reader:
        if (row[2].find("(detail)") == -1):
            if (row[2] != row_prev):
                writer.writerow(row)
            else:
                print(row[2])
                print(row_prev)
            row_prev = row[2]
```

Figura 1: Ejemplo de script de eliminación de obras repetidas




Tras ser aplicados este tipo de scripts en Python, el dataset principal estaba formado por 626 entradas (una cifra manejable) a partir de las 49568 con las que contaba el archivo inicial. Así, quedaba formado de la siguiente manera:

| 1                  | 2                        | 3                                  | 4       | 5          | 6                | 7                    | 8            | 9           | 10      | 11        |
|--------------------|--------------------------|------------------------------------|---------|------------|------------------|----------------------|--------------|-------------|---------|-----------|
| AUTHOR             | BORNIED                  | TITLE                              | DATE    | TECHNIQUE  | LOCATION         | URL                  | FORM         | TYPE        | SCHOOL  | TIMEFRAME |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Adoration of the Magi              | 1465-67 | Tempera    | National Galler  | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Madonna and Child with Six Saints  | c. 1470 | Tempera    | Galleria degli U | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Madonna and Child with an Angel    | c. 1470 | Tempera    | Isabella Stewar  | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | St Sebastian                       | 1474    | Tempera    | Staatliche Muse  | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | The Birth of Christ                | 1476-77 | Fresco     | Santa Maria No   | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Madonna of the Sea                 | c. 1477 | Tempera    | Galleria dell'Ac | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1444, Firenze, d. 11 | St Augustine                       | 1480    | Fresco     | Ognissanti, Fior | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Adoration of the Magi              | 1481-82 | Tempera    | National Galler  | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1444, Firenze, d. 11 | The Annunciation                   | c. 1485 | Tempera    | Metropolitan M   | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | St Sixtus II                       | 1481    | Fresco     | Cappella Sistini | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Three Temptations of Christ        | 1481-82 | Fresco     | Cappella Sistini | https://www.wga.hu/t | painting     | religious   | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Primavera                          | c. 1482 | Tempera    | Galleria degli U | https://www.wga.hu/t | painting     | mythologica | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Pallas and the Centaur             | c. 1482 | Tempera    | Galleria degli U | https://www.wga.hu/t | painting     | mythologica | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | The Birth of Venus                 | c. 1485 | Tempera    | Galleria degli U | https://www.wga.hu/t | painting     | mythologica | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Venus and Mars                     | c. 1483 | Tempera    | National Galler  | https://www.wga.hu/t | painting     | mythologica | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Venus and the Three Graces Present | c. 1484 | Fresco     | Musée du Louvi   | https://www.wga.hu/t | painting     | mythologica | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Three Angels                       | 1475-80 | Pen        | Galleria degli U | https://www.wga.hu/t | graphics     | study       | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Adoration of the Child             | c. 1495 | Pen        | Galleria degli U | https://www.wga.hu/t | graphics     | study       | Italian | 1451-1500 |
| BOTTICELLI, Sandro | (b. 1445, Firenze, d. 11 | Dante: Divina Commedia             | 1480s   | Manuscript | Biblioteca Apos  | https://www.wga.hu/t | illumination | other       | Italian | 1451-1500 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Sick Bacchus                       | c. 1593 | Oil        | Galleria Borghe  | https://www.wga.hu/t | painting     | mythologica | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Boy Peeling a Fruit                | c. 1593 | Oil        | Fondazione di S  | https://www.wga.hu/t | painting     | portrait    | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Boy with a Basket of Fruit         | c. 1593 | Oil        | Galleria Borghe  | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Boy Bitten by a Lizard             | c. 1594 | Oil        | National Galler  | https://www.wga.hu/t | painting     | portrait    | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | St Francis in Ecstasy              | c. 1595 | Oil        | Wadsworth Ath    | https://www.wga.hu/t | painting     | religious   | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Musicians                      | 1595-96 | Oil        | Metropolitan M   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Fortune Teller                 | c. 1596 | Oil        | Pinacoteca Cap   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Cardsharps                     | c. 1596 | Oil        | Kimbell Art Mu   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Lute Player                    | c. 1600 | Oil        | Metropolitan M   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Lute Player                        | c. 1596 | Oil        | The Hermitage,   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Bacchus                            | c. 1596 | Oil        | Galleria degli U | https://www.wga.hu/t | painting     | mythologica | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Fortune Teller                 | 1596-97 | Oil        | Musée du Louvi   | https://www.wga.hu/t | painting     | genre       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Mary Magdalene                     | 1596-97 | Oil        | Galleria Doria P | https://www.wga.hu/t | painting     | religious   | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Rest on Flight to Egypt            | 1596-97 | Oil        | Galleria Doria P | https://www.wga.hu/t | painting     | religious   | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Basket of Fruit                    | c. 1597 | Oil        | Pinacoteca Aml   | https://www.wga.hu/t | painting     | still-life  | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | St Catherine of Alexandria         | c. 1598 | Oil        | Museo Thyssen    | https://www.wga.hu/t | painting     | religious   | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | Interior view of the Chapel        | 1600-01 | Oil        | Cerasi Chapel,   | https://www.wga.hu/t | painting     | other       | Italian | 1551-1600 |
| CARAVAGGIO         | (b. 1571, Caravaggio, c  | The Crucifixion of Saint Peter     | 1600-01 | Oil        | Cerasi Chapel,   | https://www.wga.hu/t | painting     | religious   | Italian | 1551-1600 |

Figura 2: Parte del dataset .csv de obras definitivo

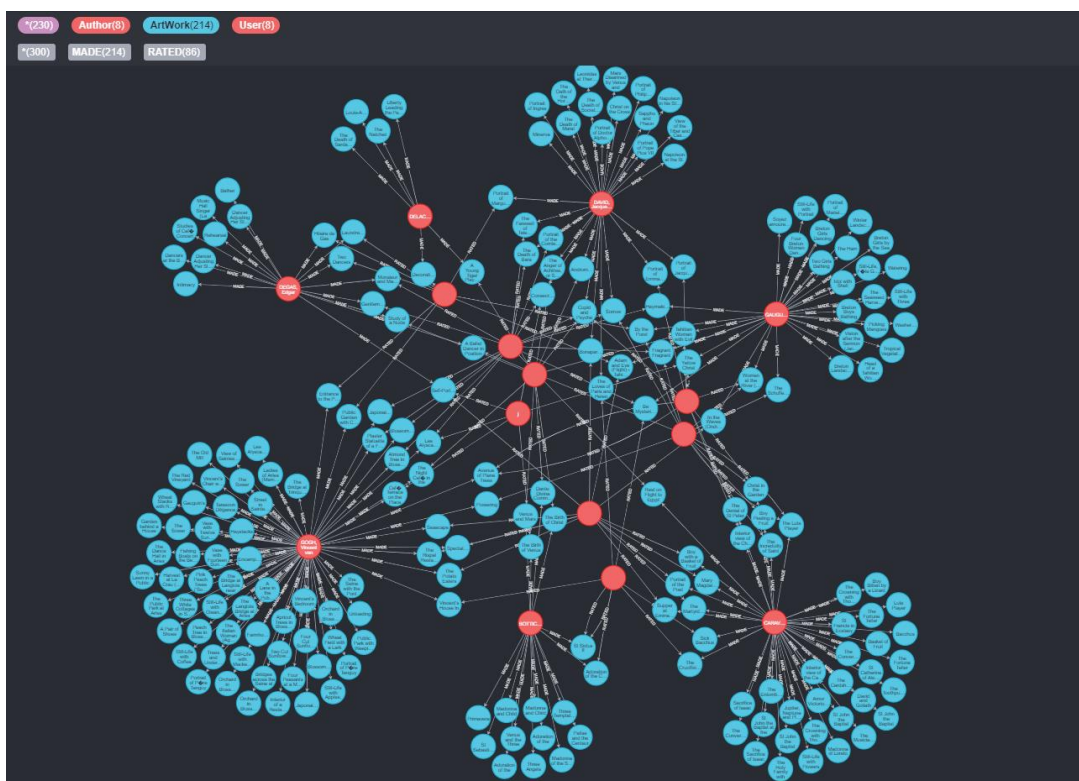
El siguiente paso era importar cada uno de los datasets (el de obras, el de usuarios y el de valoraciones) en el sistema gestor de base de datos:

- ◇  neo4j : Mi primera experiencia con una base de datos orientada a grafos, en las cuales se da la misma importancia a las relaciones entre los datos que a las entidades en sí mismas, por lo que resulta especialmente útil para trabajar con sistemas de recomendación, donde las relaciones entre los nodos juegan un papel fundamental a la hora de establecer patrones y similitudes, sin necesidad de claves foráneas u otras características de las bases de datos tradicionales, haciendo uso únicamente de nodos, las relaciones entre ellos y etiquetas de propiedades.



Además, se basa en el uso de Cypher, un lenguaje de consultas realmente sencillo de aprender y que permite una gran versatilidad con una curva de dificultad asequible.

Por último, me gustaría comentar el gran trabajo realizado por el equipo de Neo4j, que ponen a disposición del usuario una gran cantidad de documentación y cursos gratuitos que yo mismo aproveché para aprender la tecnología y contar con certificados que así lo acrediten.






*Figura 3: Parte de la base de datos importada en Neo4j*





## 2.2. FRONTEND

La parte gráfica de la aplicación, lo que vemos, está compuesta, principalmente, de HTML, CSS y JavaScript, como toda aplicación web que se puede encontrar en internet. Sin embargo, fue necesaria la utilización de otras herramientas para llevar a cabo un desarrollo más sencillo y personalizado:

- ◇  **Vue.js** : Nunca antes había utilizado un framework de JavaScript, habiendo trabajado hasta ahora con el propio lenguaje en crudo para los escasos y sencillos ejercicios de Aplicación Web que había desarrollado hasta la fecha, por lo que me presentaba ante una disyuntiva: ¿Debía utilizar Vue u otro framework más tradicional (Angular, React...)? Al final, tras investigar sobre la materia, decidí que la facilidad de aprendizaje, la flexibilidad y la capacidad de reutilización de Vue eran ideales para poder aprender la tecnología (para lo cual usé cursos completos de YouTube y de Udemy) y aplicarla en el proyecto.
- ◇  **Vuetify** : El framework de Material Design de Vue por excelencia. La inmensa cantidad de componentes que aporta la librería, junto a la facilidad de uso, capacidad de personalización y estética cuidada y moderna de los mismos no me dejó lugar a dudas sobre su implantación en la aplicación, que fue muy sencilla gracias a la documentación oficial.
- ◇  **axios** : Esta librería de JavaScript, que ejerce como cliente HTTP, simplifica las peticiones y la comunicación entre el frontend y el backend, ya que tanto la instalación como el formato de las consultas GET, POST... es extremadamente sencillo de aprender.



## 2.3. BACKEND





La capa oculta de la aplicación, donde reside toda la lógica y se resuelven todas las peticiones relacionadas con la base de datos, conlleva un conjunto de herramientas de diversos tipos para poder llevar a cabo las acciones requeridas por la aplicación:

- ◇ **JS JavaScript** : Tras mucho dudar sobre el lenguaje a utilizar para el desarrollo del backend, y después de probar una implementación en Python y no ser capaz de conseguir un resultado que me convenciera, decidí usar JavaScript, un lenguaje muy común y con una integración muy sencilla con la base de datos Neo4j, tan solo necesitando de un driver a incluir en el código principal del backend.
- ◇ **node JS** : Entorno de ejecución basado en JavaScript que ejerce como servidor y que permite ejecutar este tipo de aplicaciones fuera de un navegador, es decir, establece una estructura para poder ejecutar una aplicación con conexión a base de datos sin necesidad de que estar alojada en un servidor. Además, proporciona una gran cantidad de paquetes reutilizables que facilitan el tratamiento del código y los datos.
- ◇ **Express JS** : Framework de NodeJS que permite el tratamiento de peticiones HTTP de forma que puedan atenderse las solicitudes del backend y administra aspectos como el puerto a utilizar para la ejecución del programa. Además, su instalación y utilización es muy simple, basta con apenas un par de líneas de código para que todo funcione.



## 2.4. HERRAMIENTAS GENERALES

Además de las herramientas específicas para algunas de las partes de la aplicación, ha sido necesario el uso de otras tecnologías de carácter más general o que pueden ser comunes a varias de estas partes.

- ◇  **VS Code** : Uno de los editores de código por excelencia, que permite la utilización de varios lenguajes diferentes simultáneos y que ofrece una gran capacidad de personalización mediante los plugins y las extensiones para facilitar la tarea del programador.
- ◇  **GitHub** : La plataforma de Git más utilizada a nivel mundial, que con la creación de un repositorio facilita el control de versiones y commits, además de permitir compartir el código tanto con mis compañeros de asignatura como a todo aquel que tenga un acceso a la aplicación, en forma de carta de presentación personal.
- ◇  **OSF** : Con esta herramienta he comprendido la utilidad de un cuaderno de trabajo, que de una forma u otra facilite la documentación del sistema, tanto de la parte de código como todo aquello que no se ve a la hora de realizar un proyecto completo
- ◇  **Adobe Photoshop** : Tanto para la realización del logo como de otros gráficos, como el del análisis DAFO o el funcionamiento de la aplicación, he utilizado como editor el Adobe Photoshop, que con un pequeño conocimiento técnico permite crear componentes artísticos de calidad, a pesar de no ser yo ningún experto en la materia.



### 3. APLICACIÓN



*Figura 4: Componentes de FIBONART*

El funcionamiento de la aplicación no tiene ningún misterio, no hay componentes ocultos ni mayor interacción que frontend - backend y backend - base de datos. Así, cuando el usuario interactúa con el frontend, que compone toda la parte de interfaz gráfica de la aplicación y que se encarga de mostrar todos los elementos visuales, se ejecuta una petición al backend, la parte del sistema encargada de la lógica, que transforma lo que pide el frontend en una nueva petición, esta vez referida a la base de datos, que recoge lo que el backend le ha solicitado y, o bien devuelve unos resultados, o simplemente se encarga de almacenar nueva información. Tras esta transacción, el backend recoge el resultado de su propia petición, que puede ser en forma de información o un simple mensaje de error o éxito y, tras gestionar ese resultado de la forma adecuada (a veces, simplemente dando formato a la información recibida, y otras tratando los datos con fórmulas matemáticas y otras acciones), genera una respuesta al frontend, que recibe la información que había requerido en un principio para poder actualizarse y, en según qué ocasiones, modificar su apariencia para mostrar esa nueva información al usuario.

Para poder comprender como funciona FIBONART, vamos a explicar las diferentes posibilidades que ofrece la aplicación y como interactúan entre sí.



## BASE DE DATOS

La base de datos en Neo4j se compone de distintos tipos de nodos y relaciones, cada una con unos valores y propiedades, y que permiten el correcto funcionamiento de la aplicación. Los tipos de nodo que se pueden encontrar en la base de datos son:

- ◇ User: Representa un usuario. Cuenta con un identificador, un nombre de usuario único, un email único, un nombre, apellidos y una contraseña, que estará encriptada en todos los casos.
- ◇ ArtWork: Representa una obra. Cuenta con un identificador y los atributos título, url (para la obtención de la imagen) y fecha de creación.
- ◇ Author: Representa un autor. Cuenta con un nombre e información sobre su nacimiento y muerte.
- ◇ Technique: Representa una técnica, como podrían ser el óleo o la acuarela, o un material en el caso de la escultura, como el mármol o el bronce.
- ◇ Location: Representa la localización de una determinada obra.
- ◇ Art\_form: Representa una disciplina artística: pintura, escultura, arquitectura...
- ◇ Art\_Type: Representa la temática de las obras: religiosa, mitológica, paisajes, retratos...
- ◇ School: Representa la escuela a la que pertenece una obra: francesa, italiana, española...
- ◇ TimeFrame: Establece la época en la que una obra fue realizada en lapsos de 50 años desde 1450 hasta 1900



A parte de nodos, la base de datos también almacena relaciones, que son las responsables de que exista una lógica en la aplicación y que posibilitan la utilización de los algoritmos de recomendación de nuestro sistema, que no podrían ser efectuados si, en lugar de estar almacenados en forma de relación, estos valores se guardaran en forma de propiedad del nodo Obra:

- ◇ MADE: Relación entre una obra y un autor.
- ◇ USES\_TECHNIQUE: Relación entre obra y técnica.
- ◇ LOCATED\_AT: Relación entre obra y localización.
- ◇ ITS\_FORM\_IS: relación entre obra y disciplina.
- ◇ ITS\_TYPE\_IS: Relación entre obra y temática.
- ◇ ITS\_SCHOOL\_IS: Relación entre obra y escuela.
- ◇ AT\_TIMEFRAME: Relación entre obra y época.
- ◇ RATED: Relación entre obra y usuario, y que corresponde a una valoración aportada por el mismo. Además, cuenta con el atributo score, que almacena la puntuación otorgada. Constituye la relación más importante de la aplicación, puesto que FIBONART tiene muy en cuenta las valoraciones para su algoritmo.

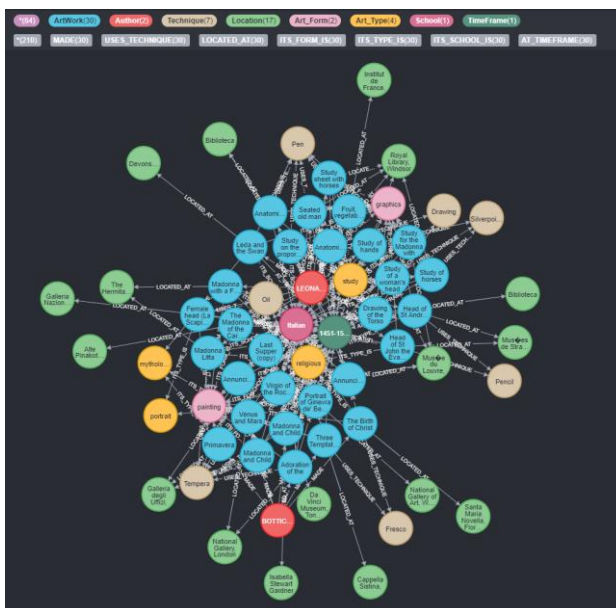


Figura 5: Grafo de los detalles de 100 obras

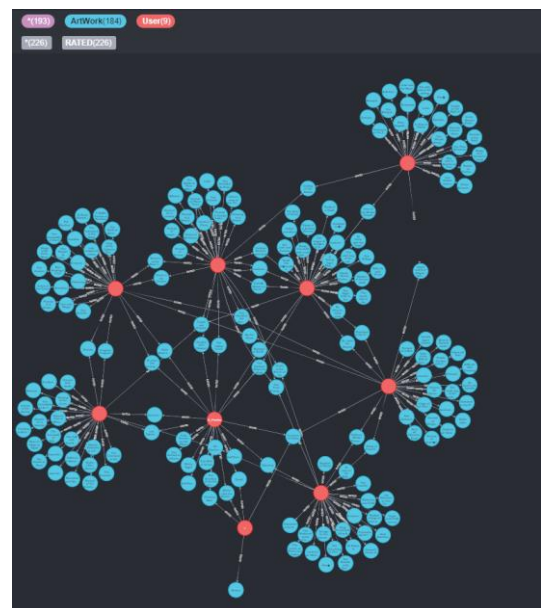
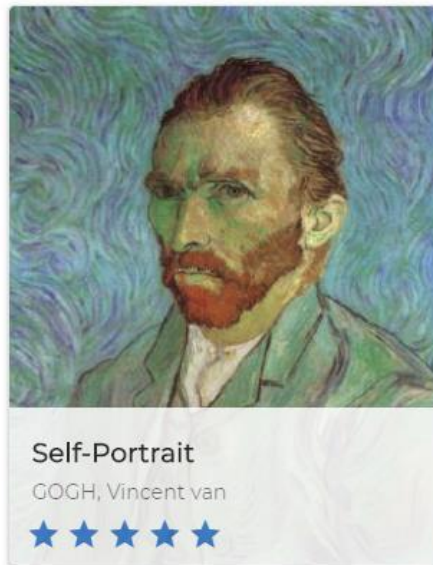


Figura 6: Grafo de las valoraciones de los usuarios.



## TARJETA DE OBRA PEQUEÑA

Este componente será muy utilizado para representar obras de forma resumida, de manera que se pueda crear una cuadrícula con un conjunto de ellas, como veremos posteriormente.



*Figura 7: Tarjeta pequeña  
con valoración previa*



*Figura 8: Tarjeta pequeña  
sin valoración previa*

En la tarjeta se representa la imagen de la obra en cuestión, su título, autor y la puntuación que el usuario le haya dado, en caso de que exista una valoración. Sin embargo, también almacenan información que no se muestra, como la técnica utilizada, localización o escuela, que sirve para mostrar en la tarjeta de obra grande.

La única interacción que permite esta tarjeta se produce cuando el usuario hace click en la tarjeta, que genera que se muestre la tarjeta de obra grande con todos los detalles sobre la obra.



## TARJETA DE OBRA GRANDE

El otro componente principal de la aplicación es la tarjeta de obra grande, que proporciona información detallada sobre la obra en cuestión.



*Figura 9: Ejemplo de Tarjeta de obra grande*

En la parte izquierda se puede apreciar una imagen de la obra, mientras que los detalles se sitúan en la parte derecha, indicando el nombre de la obra, su autor, la fecha de creación, su localización actual, la disciplina artística a la que pertenece, su temática y la escuela. Además, mediante un sistema de estrellas se muestra la valoración aportada por dicho perfil (en caso de que exista), así como la puntuación media de entre todos los usuarios del sistema.

Aquí, mediante un simple click en el número de estrellas deseado, el frontend se encarga de transmitir al backend la puntuación otorgada para que, mediante una consulta a la base de datos, cree o actualice la relación entre el usuario y la obra nada más se efectúe un click en las estrellas del frontend.

```
var query =  
  "MATCH (u:User) WHERE ID(u) = " + idUser +  
  " WITH u MATCH (w:ArtWork) WHERE ID(w) = " + idArtwork +  
  " WITH u,w MERGE (u)-[r:RATED]->(w) ON CREATE SET r.score = " + rating +  
  " ON MATCH SET r.score = " + rating +  
  " RETURN null";
```

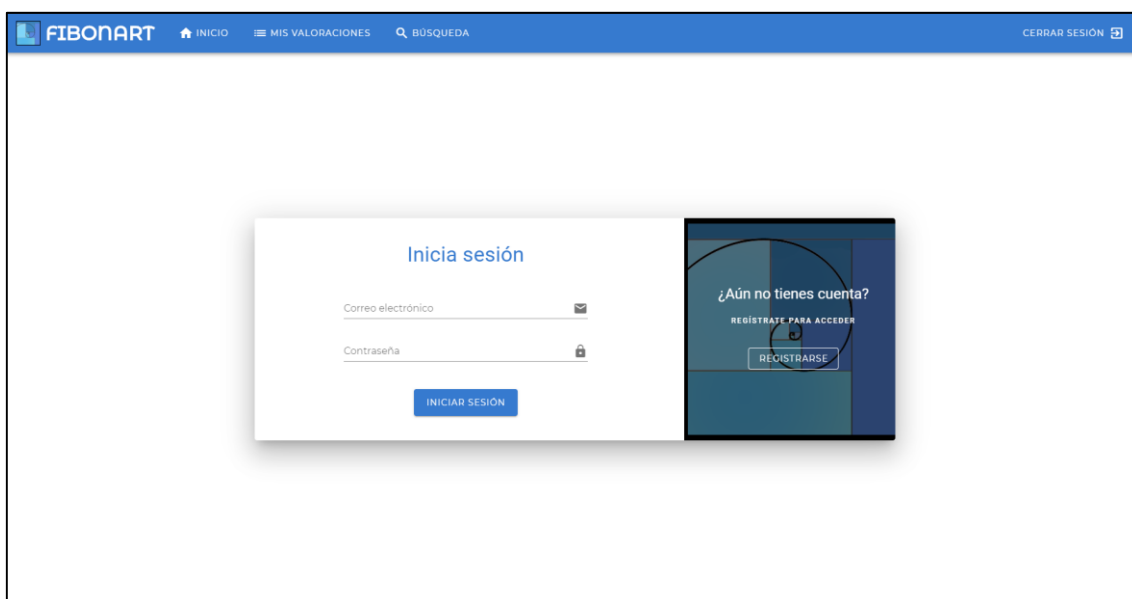
*Figura 10: Consulta para creación o modificación de la valoración de una obra*





## INICIO DE SESIÓN Y REGISTRO

La aplicación comienza mostrando la ventana en la que se comparten tanto el inicio de sesión como el registro de nuevas cuentas, ya que para poder utilizar el sistema es necesario contar con un perfil propio que pueda almacenar las valoraciones e interacciones del usuario.



*Figura 11: Ventana de inicio de sesión*

Para poder iniciar sesión, solo es necesario introducir el correo electrónico utilizado al crear la cuenta, identificador único de cada usuario, y la contraseña introducida, que por supuesto estará encriptada. En caso de que no coincida alguna de las dos variables introducidas, se alertará al usuario del error, indicando que debe cambiar los valores introducidos para poder iniciar sesión.

Cuando el usuario ha introducido sus credenciales y el sistema ha verificado que son correctas, se le mostrará automáticamente la ventana Home, que será explicada posteriormente.

Sin embargo, si el usuario no dispone aún de una cuenta, solo tendrá que clickar en la parte derecha, en el botón sobre el logo de FIBONART y, tras la animación, se le mostrará la ventana de registro:



*Figura 12: Ventana de registro*

Para registrarnos, debemos introducir un nombre y unos apellidos, un nombre de usuario, un correo y una contraseña. Antes de confirmar el registro, el sistema se asegura de que tanto el nombre de usuario como el correo electrónico son únicos, es decir, no existe otro usuario con el mismo valor en su cuenta. Además, la contraseña se encripta desde un primer momento mediante una función hash, por lo que es almacenada como una secuencia de números y caracteres incomprensibles para todo aquel que no tenga la clave de descifrado.

Tras pulsar el botón de registrarse, si todos los campos son correctos (en caso contrario, se alertaría al usuario y se impediría el registro), la cuenta habrá sido creada con éxito y se redirigirá directamente al usuario a la ventana de Introducción.

Además, mediante un simple click en el botón sobre el logo de la parte izquierda, es posible volver a la ventana de inicio de sesión.

Por último, comentar que, aunque la barra de navegación sea visible en todo momento, es imposible acceder a otra vista que no sea esta hasta que no se haya iniciado sesión, por lo que tanto a través de los enlaces directos como cambiando la propia dirección URL, será imposible acceder a otra vista, ya que el sistema redirigirá directamente a la vista de inicio de sesión.



En cuanto a la relación con el backend, podemos apreciar dos peticiones diferentes:

- ◇ Inicio de sesión: Cuando el usuario introduce el correo electrónico y la contraseña, el frontend le envía esa información al backend, que consulta en la base de datos si los datos son correctos a través de la siguiente consulta:

```
var query =  
"MATCH (u:User) WHERE (u.email) =~ ('" +  
email +  
"'') AND (u.password) = '" +  
passEncriptada +  
"' RETURN ID(u)";
```

*Figura 13: Consulta para el Inicio de sesión*

Neo4j nos devolverá el ID del usuario en caso de que los datos sean correctos, o un mensaje de error si alguno de los campos no coincide con las entradas almacenadas. En este caso, no hace falta mucho tratamiento, por lo que se devuelve al frontend, que redirige a la ventana de inicio tras almacenar el ID devuelto, o informa de que alguna de las credenciales no es correcta.

- ◇ Registro: La petición se compone de 3 consultas, una para comprobar si el nombre de usuario está en uso, la siguiente cumple la misma labor para el correo electrónico y, en caso de que nada haya fallado, la consulta que crea el nodo usuario, con sus campos rellenos tras la encriptación hash de la contraseña.

```
var query1 =  
"MATCH (u:User) WHERE TOLOWER(u.username) = '" + username + "' return ID(u)";  
var query2 =  
"MATCH (u:User) WHERE TOLOWER(u.email) = '" + email + "' return ID(u)";  
var query3 =  
"CREATE (u:User {username: '" + username +  
"', email: '" + email +  
"', name: '" + name +  
"', surname: '" + surname +  
"', password: '" + passEncriptada +  
"'}) SET u.id = ID(u) RETURN ID(u)";
```

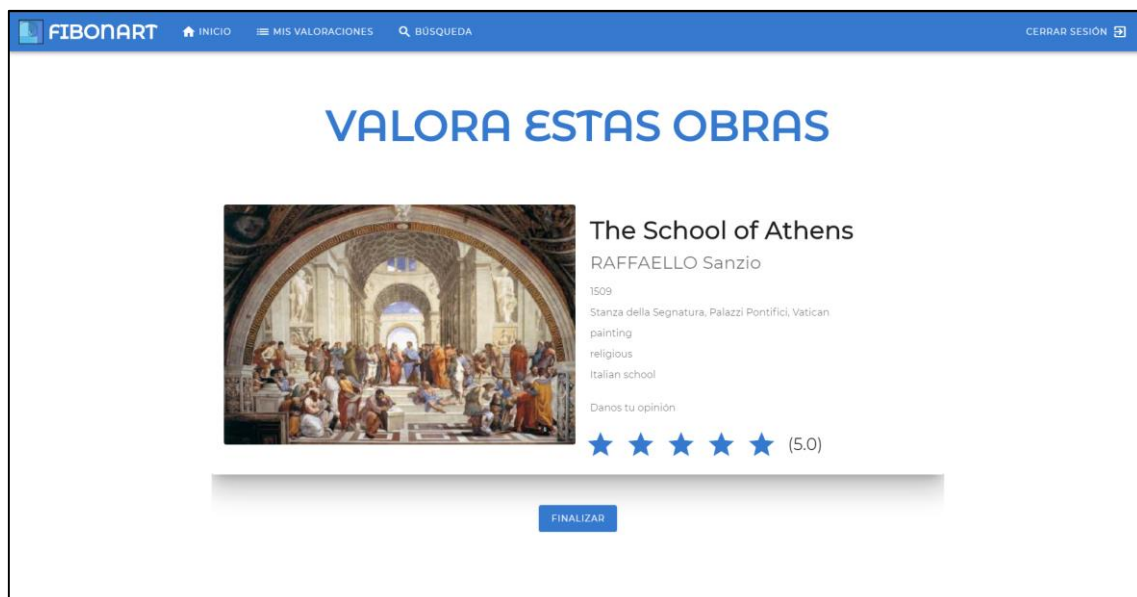
*Figura 14: Consultas para el Registro*

Si todo ha salido bien, el frontend recibirá el ID del usuario que acaba de crear, que almacena para futuras consultas, y redirige al usuario a la ventana de Introducción.



## INTRO

Cuando un usuario crea una cuenta nueva y accede por primera vez, se le mostrará la vista de Introducción, la cual se compone de 7 obras que el usuario debe valorar. De esta forma, aquel que esté utilizando el sistema por primera vez aprende cuál va a ser el funcionamiento general de las valoraciones de la aplicación, que no tienen dificultad más allá de hacer click en el número de estrellas que quieres que ejerzan la función de puntuación; y el sistema comienza a establecer un perfil de usuario basado en los gustos de la persona que esté utilizando FIBONART.



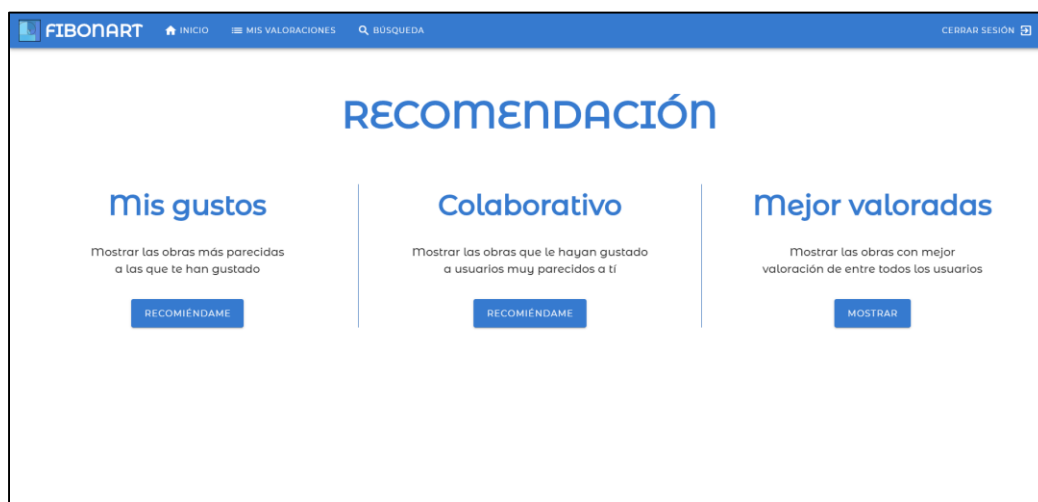
*Figura 15: Ejemplo de obra en la Ventana de introducción*

Una vez el usuario haya valorado las 7 obras que se le proponen, el frontend habrá enviado cada una de las puntuaciones otorgadas al backend, que se encarga de crear la relación con el valor correspondiente en la base de datos y, tras pulsar el botón de finalizar, el usuario será redirigido a la ventana de Inicio.



## INICIO

La ventana principal de la aplicación, con las funciones más importantes de la misma. Aquí tiene lugar la recomendación, es decir el objetivo explícito del proyecto, por lo que supone una vista imprescindible para el funcionamiento del sistema.



*Figura 16: Ventana de inicio*

Cuando accedemos a la ventana de Inicio, se nos muestran tres opciones, cada una con un título, un abreviado explicación de lo que hace, y el botón que lo ejecuta. Así, cuando el usuario decida dejarse recomendar por la aplicación, solo debe pulsar uno de estos botones a su elección.

- ◇ **MIS GUSTOS:** Se ejecuta el algoritmo de recomendación basado en contenido implementado en el sistema y que será desarrollado en el apartado 4.1 de este documento. En función de nuestras valoraciones previas, el sistema generará una serie de filtrados y cálculos para encontrar las obras que más se parezcan a aquellas que han recibido una buena puntuación por el usuario, y se presentan en forma de cuadrícula las 20 con una mayor similitud coseno al vector perfil de usuario. Aunque en el apartado 5 podremos ver con detalle algún ejemplo de esta funcionalidad, podemos ver el resultado que recibe uno de los usuarios del sistema:

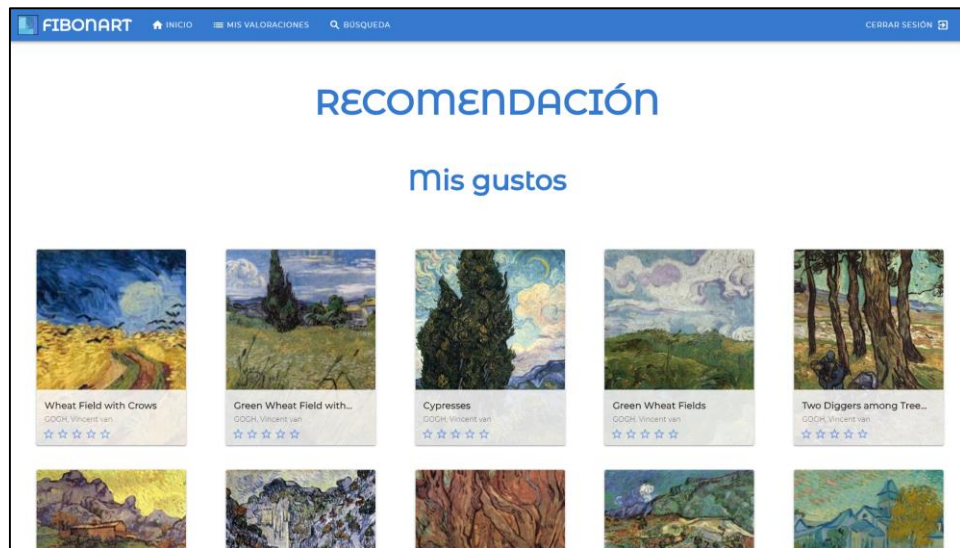


Figura 17: Recomendación basada en contenido

- ◇ **COLABORATIVO:** Se ejecuta el algoritmo de filtrado colaborativo implementado en el sistema y detallado en el apartado 4.2 del documento. Para ello, se tienen en cuenta las obras valoradas por el usuario y todas las demás cuentas del sistema, ya que trata de buscar aquellos usuarios que más se parezcan al que ha solicitado la recomendación, lo cual es calculado mediante la similitud Pearson, y genera una cuadrícula con hasta 20 obras basadas en las valoraciones de esos usuarios que comparten tus propios gustos.

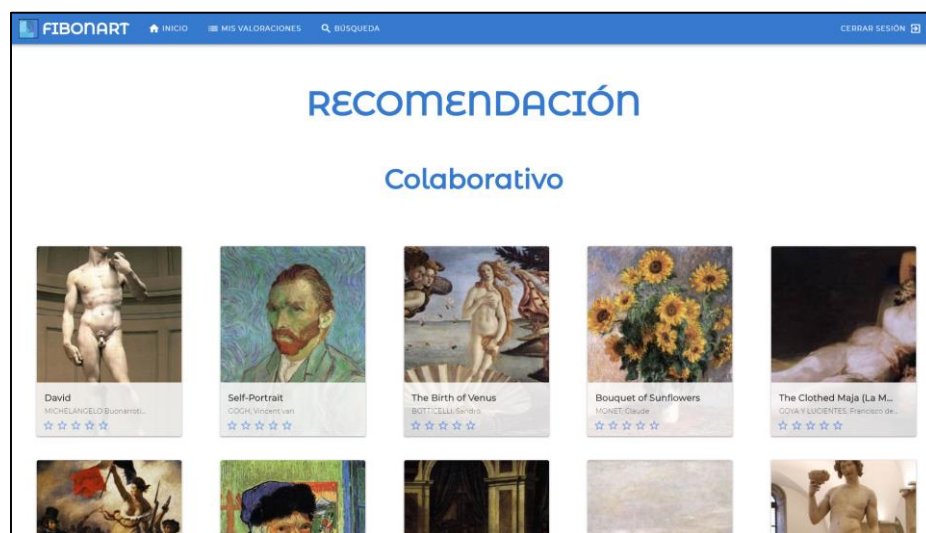
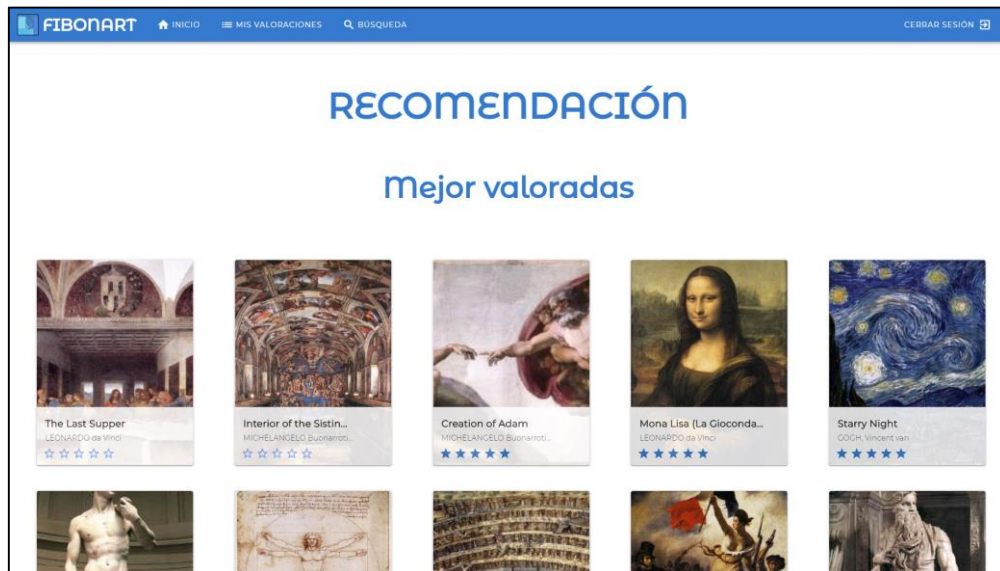


Figura 18: Recomendación por filtrado colaborativo



- ◇ **MEJOR VALORADAS:** El sistema calcula la nota media de entre todas las valoraciones otorgadas por los usuarios de las obras almacenadas en la base de datos y las muestra en orden descendente.



*Figura 19: Obras mejor valoradas*

Aunque no sea una recomendación como tal, pues el sistema no tiene en cuenta si ha sido solicitada por un usuario u otro si no que muestra lo mismo para todos, puede servir para conocer las obras más populares, que deberían ser las mejores, al menos bajo el criterio de los usuarios de la aplicación. Para ello, se lleva a cabo una consulta a la base de datos, la cual busca todas las relaciones de tipo RATED (es decir, las valoraciones) y calcula la media de puntuaciones para cada una de las obras, devolviendo todos los datos necesarios de las 20 que mayor nota media tengan. Además, en caso de que exista, también obtendremos la valoración del usuario de esa obra en concreto, para poder mostrarla en el sistema de estrellas.

```
var query =
"MATCH (:User)-[r:RATED]->(w:ArtWork), (a)-[:MADE]->(w), (w)-[:USES_TECHNIQUE]->(t), (w)-[:LOCATED_AT]->(l)," +
" (w)-[:ITS_FORM_IS]->(f), (w)-[:ITS_TYPE_IS]->(y), (w)-[:ITS_SCHOOL_IS]->(s)" +
" OPTIONAL MATCH ((u:User {id: " + idUser + "})-[ru:RATED]->(w:ArtWork)) " +
" RETURN ID(w), w.title, w.url, a.author_name, w.date, t.technique, l.location, f.art_form," +
" y.arttype, s.school, ru.score, avg(r.score) as punt order by punt desc limit 20";
```

*Figura 20: Consulta para obtener las obras mejor valoradas*





## MIS VALORACIONES

En esta vista se nos ofrecerá una lista con todas las obras que hemos valorado desde la creación de nuestra cuenta, de una manera clara y ordenada.

Así, el usuario tendrá la oportunidad de volver a ver los detalles de las obras que le han gustado, editar una puntuación otorgada que no se corresponde con el criterio actual, encontrar de una manera sencilla una obra que ha visto, aunque no recuerde su nombre o simplemente echar un vistazo al conjunto de obras de una manera más general.

Cuando se accede a la vista, se muestra una cuadrícula con todas las obras valoradas hasta el momento, indicando el título, el autor, la imagen y la puntuación otorgada por el usuario de cada una.

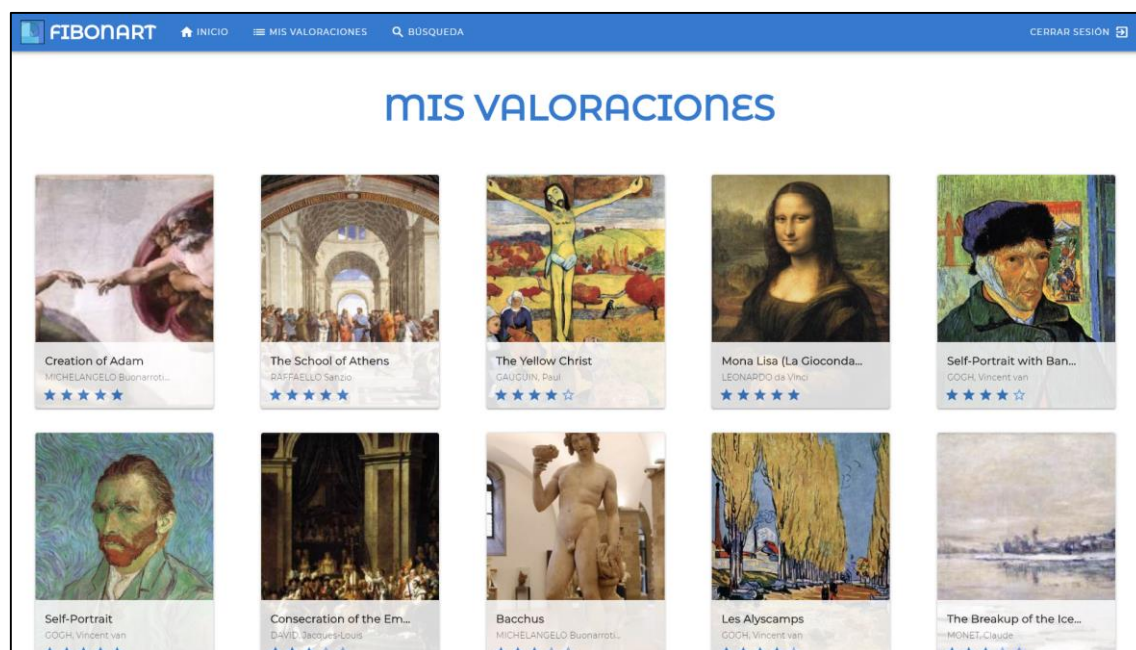


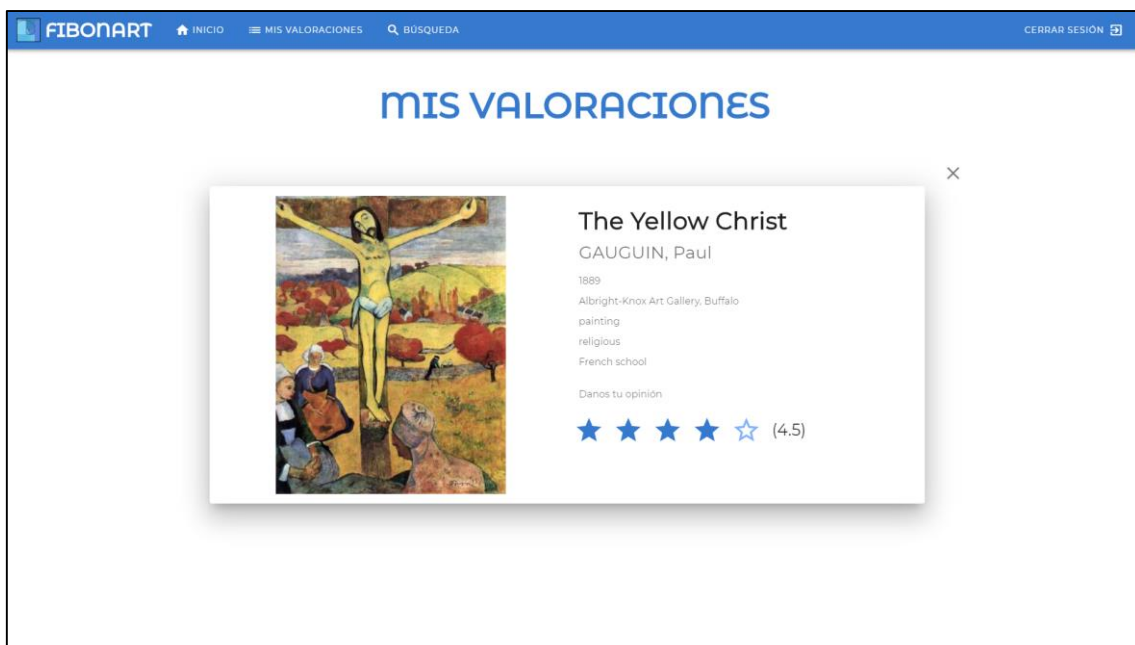
Figura 21: Cuadrícula con las obras valoradas por el usuario





Si la cuenta con la que se accede aún no ha realizado ninguna valoración (es decir, ha decidido no puntuar ninguna de las obras de la ventana de Introducción), recibirá una alerta que indica que aún no hay registro de ninguna valoración con esa cuenta.

Cuando el usuario pulsa en alguna de las obras que se le muestran en la lista, la cuadrícula se oculta y se le presenta la información más detallada de la obra en cuestión, donde se pueden ver propiedades como la fecha de creación, la localización, la temática utilizada y otras como la puntuación otorgada por el usuario y la media de entre todas las cuentas que han decidido valorar esa obra en concreto, para que el que lo vea se pueda hacer a la idea de la popularidad de lo que esté viendo. Además, podrá modificar su valoración, que queda actualizada en la base de datos de manera automática, o cerrar esa obra mediante un simple click en la cruz superior derecha, que mostrará de nuevo la cuadrícula con todas las obras valoradas.



*Figura 22: Ejemplo de obra valorada por el usuario en tarjeta grande*



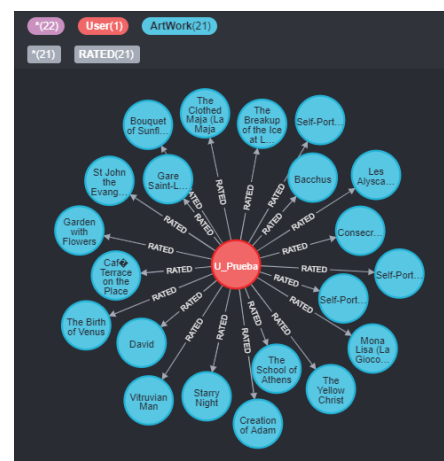
Para el correcto funcionamiento de esta vista, se ha implementado una petición concreta a la base de datos (además de la que permite un cambio en la valoración). Así, cuando el usuario accede a la vista, el frontend pide al backend que le envíe la información detallada de todas las obras para las que el usuario ha hecho una valoración con anterioridad, que el propio backend gestiona haciendo la siguiente petición a la base de datos:

```
var query =
  "MATCH (u:User)-[r:RATED]->(w:ArtWork), (a)-[:MADE]->(w), (w)-[:USES_TECHNIQUE]->(t), (w)-[:LOCATED_AT]->(l), " +
  "(w)-[:ITS_FORM_IS]->(f), (w)-[:ITS_TYPE_IS]->(y), (w)-[:ITS_SCHOOL_IS]->(s)" +
  " WHERE ID(u) = " +
  idUser +
  " OPTIONAL MATCH (:User)-[ra:RATED]->(w:ArtWork)" +
  " RETURN ID(w), w.title, w.url, a.author_name, w.date, t.technique, l.location, "+
  "f.art_form, y.arttype, s.school, r.score, avg(ra.score) as punt";
```

*Figura 23: Consulta de solicitud de todas las obras valoradas por el usuario*

El backend solicita a la base de datos Neo4j que le envíe toda la información almacenada sobre las obras que el usuario en cuestión, que es identificado mediante su ID, ha valorado previamente. Además, en caso de que existan, también calcula la media aritmética de todas las puntuaciones otorgadas a cada una de las obras entre todos los usuarios de la aplicación. Después de darle formato a estos datos mediante el JavaScript del backend, se envía toda la información resultante al frontend, que gestiona cada entrada del vector recibido para dotar de información real a los componentes visuales para mostrar al usuario las obras de la forma que hemos visto.

Para hacernos a la idea de cómo está estructurada esta información, he hecho una consulta a la base de datos pidiendo que muestre un usuario concreto (rojo) y la relación con todas las obras que ha valorado (azul) a partir de lo cual se pueden obtener todas las relaciones y propiedades requeridas.



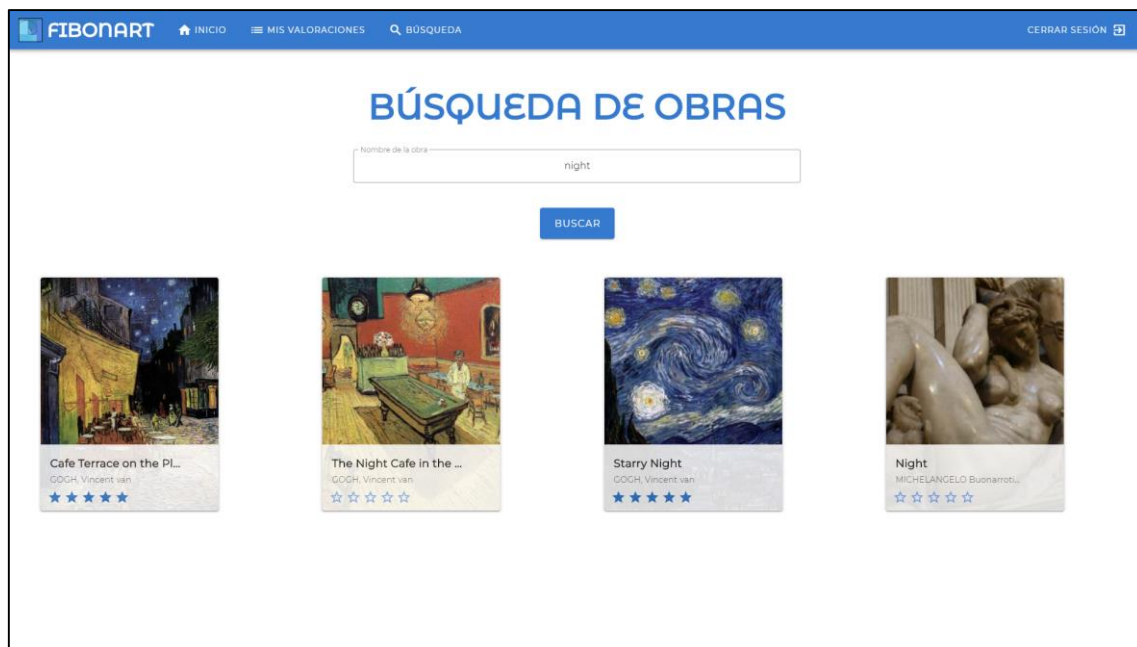
*Figura 24: Ejemplo de usuario y sus valoraciones*



## BÚSQUEDA

En esta vista, se ofrece la posibilidad de realizar una búsqueda por título de obra.

El usuario introducirá la cadena que desea buscar y pulsará el botón en el centro de la pantalla. Si la base de datos tiene almacenada alguna obra cuyo título contenga la cadena de caracteres introducida, con independencia de minúsculas o mayúsculas, mostrará dichas obras al usuario en forma de cuadrícula, indicando el título, el autor, la imagen y la puntuación otorgada por el usuario (en caso de exista) de cada una.



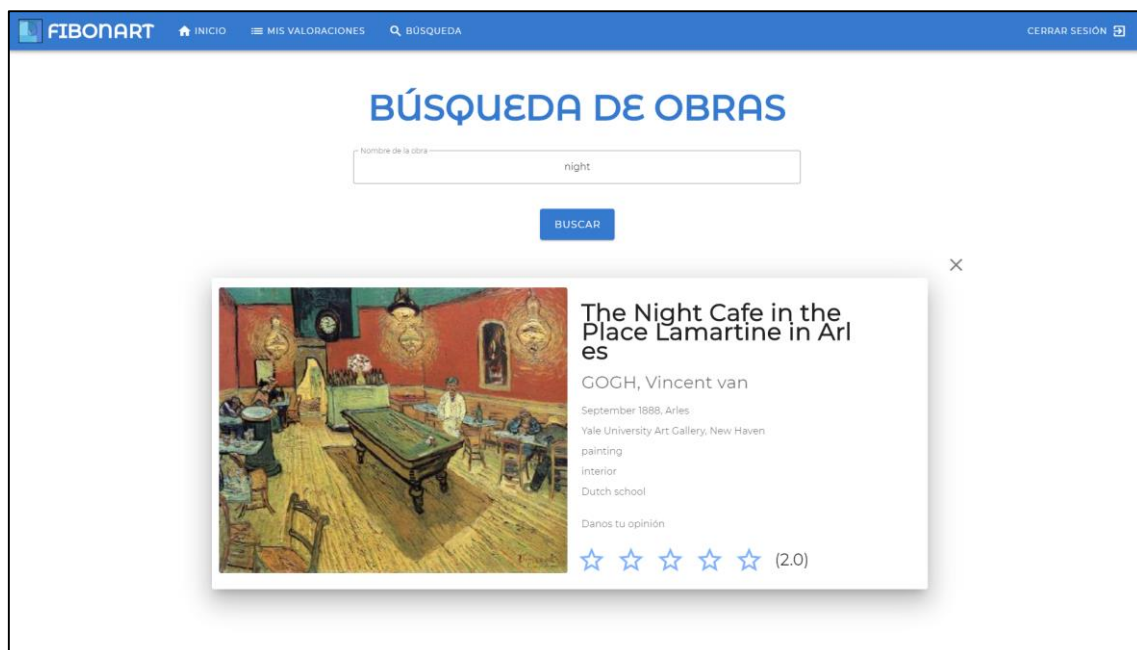
*Figura 25: Ejemplo de búsqueda por título*

Si no existe ninguna obra cuyo título contenga la cadena introducida, se avisará al usuario de que la búsqueda no ha tenido éxito.

Además, el usuario puede generar tantas búsquedas como quiera sin necesidad de cambiar de vista, únicamente introduciendo una nueva cadena en la barra de búsqueda, ya que el sistema se encarga de ocultar las obras de la búsqueda anterior para dejar sitio para las que haya recibido nuevas.



En caso de que pulse sobre alguna de las obras de la cuadrícula, esta se ocultará y se le presentará la información más detallada de la obra en cuestión, donde se pueden ver propiedades como la fecha de creación, la localización, la temática utilizada, y otras como la puntuación otorgada por el usuario (si existe) y la media de entre todas las cuentas que han decidido valorar esa obra en concreto, para que aquel que lo vea se pueda hacer a la idea de su popularidad. Además, podrá introducir o modificar su valoración, que queda actualizada en la base de datos de manera automática, o cerrar esa obra mediante un simple click en la cruz superior derecha, que mostrará de nuevo la cuadrícula con todas las obras buscadas.



*Figura 26: Una de las obras buscadas en formato Tarjeta Grande*



Para poder llevar a cabo todo el proceso, es necesaria la utilización de consultas desde el frontend hacia el backend, que, tras recibir el ID del usuario (para poder encontrar posibles valoraciones) y la cadena que ha introducido en la barra de búsqueda, efectúa la siguiente consulta a la base de datos Neo4j:

```
var query =
  "MATCH (w:ArtWork), (a)-[:MADE]->(w), (w)-[:USES_TECHNIQUE]->(t), (w)-[:LOCATED_AT]->(l)," +
  "(w)-[:ITS_FORM_IS]->(f), (w)-[:ITS_TYPE_IS]->(y), (w)-[:ITS_SCHOOL_IS]->(s)" +
  " WHERE TOLOWER(w.title) CONTAINS '" +
  title + "'" +
  " OPTIONAL MATCH ((u:User {id: " +
  idUser +
  " })-[:RATED]->(w))" +
  " WITH w,a,t,l,f,y,s,r" +
  " OPTIONAL MATCH (:User)-[:RATED]->(w)" +
  " RETURN ID(w), w.title, w.url, a.author_name, w.date, t.technique," +
  " l.location, f.art_form, y.arttype, s.school, r.score, avg(ra.score)";
```

*Figura 27: Consulta de solicitud de todas las obras que contengan una cadena en su título*

En ella, se solicitan todas las obras cuyo título contenga la cadena introducida, independientemente de las mayúsculas o minúsculas de cada uno de ellos, y se devuelve toda la información necesaria de cada obra (título completo, autor, técnica, fecha...). Además, la query cuenta con dos match opcionales: el primero sirve para, en caso de que exista, devolver la puntuación que esa cuenta le haya dado a la obra en cuestión; y el segundo, para calcular la media de entre todas las valoraciones realizadas por todos los usuarios del sistema.

Tras darle formato a estos datos devueltos, el backend devuelve al frontend toda la información de las obras que el usuario ha buscado, siendo la vista Búsqueda la encargada de crear la cuadrícula y la tarjeta con las obras; o un mensaje de error en caso de que no exista ninguna obra con el título introducido, que el frontend interpretará generando una alerta informando al usuario de la situación.



## 4. ALGORITMO DE RECOMENDACIÓN

Para el algoritmo de recomendación de la aplicación tenía claro que no quería implementar un simple sistema de búsqueda o de filtrado mediante queries simples a la base de datos, ya que no aportaría nada nuevo ni de valor al contexto de la aplicación. Para que tuviera sentido y que realmente el usuario obtuviera recomendaciones de calidad, que estuvieran secundadas por un sistema informático “inteligente”, era necesario implementar uno o varios algoritmos que, en función de la información que tuviera del usuario (en este caso, sus valoraciones y gustos), pudiera encontrar recomendaciones de interés para ese usuario. Además, esa era la razón para utilizar una base de datos basada en grafos, que son especialmente útiles para esta tarea. Para ello, el sistema cuenta con dos tipos distintos de algoritmo.



#### 4.1 RECOMENDACIÓN BASADA EN CONTENIDO

Para la recomendación basada en contenido, el sistema tiene en cuenta las características de las obras en relación con el perfil de preferencias del usuario. Este perfil se representa en forma de vector y se construye a través de una consulta a la base de datos.

Para calcular el grado de semejanza entre los vectores de preferencias y los de características de obras, he decidido utilizar la Similitud Coseno, especialmente indicada para la comparación de vectores y, por ende, para medir la semejanza entre las características de dos ítems.

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

Figura 28. Fórmula de la similitud coseno

A modo de ejemplo explicativo, he creado dos vectores de características parecidos a los que podríamos encontrar en nuestra aplicación. Para ello, supongamos que Vector 1 es el vector que representa las características de las obras que nos gustan, mientras que Vector 2 y 3 representan otras obras, y queremos elegir cuál de las dos se parece más a nuestros gustos.

|          | Caract. 1 | Caract. 2 | Caract. 3 | Caract. 4 | Caract. 5 | Caract. 6 | Caract. 7 |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Vector 1 | 1         | 0         | 0         | 1         | 1         | 1         | 0         |
| Vector 2 | 1         | 1         | 0         | 0         | 1         | 1         | 0         |
| Vector 3 | 0         | 1         | 1         | 0         | 1         | 1         | 1         |

En este caso, la similitud coseno de Vector1 y Vector2 es 0.75, mientras que entre Vector 1 y Vector 3 es de 0.44, por lo que podemos deducir que los Vectores 1 y 2 son mucho más parecidos que los vectores 1 y 3.



Ahora que hemos comprendido como se calcula la Similitud Coseno, veamos cómo se aplica en FIBONART. Comenzaremos analizando la consulta que genera el vector de características del usuario en cuestión:

```
var queryProfile =
'MATCH (feature) WHERE "Author" in labels(feature) OR "Technique" in labels(feature) '+
'OR "Art_form" in labels(feature) OR "Art_Type" in labels(feature)'+
'OR "School" in labels(feature) OR "TimeFrame" in labels(feature) '+
'WITH feature ORDER BY id(feature)'+
'MATCH (w:ArtWork)-[r1:RATED]-(u:User)'+
'WHERE u.id = ' + idUser + ' AND r1.score >= 3'+
'OPTIONAL MATCH (w)-[r:MADE|USES_TECHNIQUE|ITS_FORM_IS|ITS_TYPE_IS|ITS_SCHOOL_IS|AT_TIMEFRAME]-(feature)'+
'WITH w, collect (CASE WHEN r IS null THEN 0 ELSE 1 END) as value'+
'RETURN value';
```

*Figura 29: Consulta para la obtención del perfil del usuario*

En esta consulta se crea un vector de 1's y 0's como los comentados anteriormente para cada una de las obras que el usuario ha valorado positivamente, de forma que cada 1 representa que la obra en cuestión pertenece a un determinado artista, utiliza una técnica concreta, etc.

Una vez hemos obtenido el vector de características de cada una de esas obras, calcularemos la media de entre todos ellos, de forma que obtengamos un vector de características que represente los gustos del usuario, puesto que si, por ejemplo, valora positivamente muchas obras de Van Gogh, el vector tendrá un valor muy alto en la entrada que represente dicho artista.

A continuación, realizamos otra consulta a la base de datos para obtener el vector de características de todas las obras que el usuario no ha valorado aún. Aunque es muy parecida a la anterior, buscamos precisamente las obras que no ha valorado en lugar de las que sí.

```
var queryNonRated =
'MATCH (feature) WHERE "Author" in labels(feature) OR "Technique" in labels(feature)'+
'OR "Art_form" in labels(feature) OR "Art_Type" in labels(feature) '+
'OR "School" in labels(feature) OR "TimeFrame" in labels(feature) '+
'WITH feature ORDER BY id(feature) MATCH (w:ArtWork)'+
'WHERE NOT EXISTS ((:User{id:' + idUser +'})-[:RATED]->(w))'+
'OPTIONAL MATCH (w)-[r:MADE|USES_TECHNIQUE|ITS_FORM_IS|ITS_TYPE_IS|ITS_SCHOOL_IS|AT_TIMEFRAME]-(feature)'+
'WITH w, collect (CASE WHEN r IS null THEN 0 ELSE 1 END) as value'+
'RETURN w.id, value';
```

*Figura 30: Consulta para el vector de las obras no valoradas*





Una vez hemos obtenido todos los vectores, el procedimiento pasa por realizar el cálculo de la similitud coseno entre el vector perfil de usuario y el de cada una de las obras que no ha valorado aún. Así, tras ordenar el resultado de mayor a menor, obtenemos las obras que más se parecen a aquellas que concuerdan con los gustos del usuario, por lo que simplemente debemos pedir los detalles (autor, técnica...) de cada una de ellas para devolver al frontend.



## 4.2 RECOMENDACIÓN POR FILTRADO COLABORATIVO

El otro tipo principal de recomendación utiliza un filtrado colaborativo para decidir qué obras mostrar, es decir, el sistema utiliza las valoraciones del resto de usuarios del sistema para encontrar las obras que más se puedan adecuar a tus gustos. Para ello, se basa en la asunción de que, si dos usuarios han concordado en el pasado, probablemente lo hagan en el futuro, para lo cual utiliza el historial de valoraciones de cada uno.

A grandes rasgos, la idea del algoritmo es encontrar la similitud Pearson entre el usuario en cuestión y todos aquellos con los que comparta al menos 3 valoraciones para, de esta forma, encontrar los usuarios que más se parezcan y recomendar las obras que estos usuarios han valorado positivamente y nosotros aún no conocemos.

La similitud Pearson es una de los recursos más utilizados para comparar distintos perfiles de usuario, ya que, al estar basado en la media, es utilizable por usuarios que tengan una tendencia de valoración diferente, es decir, tiendan a dar notas más altas o más bajas. Además, en nuestro sistema solo se comparan cuentas que tengan un cierto número de valoraciones en común, puesto que, si los usuarios cuentan con pocas puntuaciones otorgadas, el sistema va a ser poco eficiente, ya que muchos usuarios se parecerán más de lo que deberían.

$$\frac{\sum_{i=1}^n (A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^n (A_i - \bar{A})^2 \sum_{i=1}^n (B_i - \bar{B})^2}}$$

*Figura 31: Similitud Pearson*



A modo de ejemplo explicativo del funcionamiento de la similitud Pearson, he preparado una tabla con las valoraciones de 4 obras para 5 usuarios diferentes, de forma que calculemos la similitud Pearson entre el Usuario 1 y todos los demás con el objetivo de encontrar aquellos que tengan un criterio parecido.

|           | Obra 1 | Obra 2 | Obra 3 | Obra 4 | Obra 5 |
|-----------|--------|--------|--------|--------|--------|
| Usuario 1 | 5      | 3      | 4      | 4      | ¿?     |
| Usuario 2 | 3      | 1      | 2      | 3      | 3      |
| Usuario 3 | 4      | 3      | 4      | 3      | 5      |
| Usuario 4 | 3      | 3      | 1      | 5      | 4      |
| Usuario 5 | 1      | 5      | 5      | 2      | 1      |

Figura 32: Tabla de ejemplo de valoraciones

Para poder calcular la similitud entre los usuarios 1 y 2, comenzamos calculando la media de entre todas sus valoraciones para, a continuación, utilizarla en la resta del sumatorio correspondiente de la fórmula. Una vez hemos calculado la similitud para todos, podemos extraer unas pequeñas conclusiones del grado de parecido entre los distintos usuarios en función de sus valoraciones.

|           | Usuario 1 |
|-----------|-----------|
| Usuario 2 | 0,85      |
| Usuario 3 | 0         |
| Usuario 4 | 0,7       |
| Usuario 5 | -0,79     |

Figura 33: Similitudes Pearson

Así, vemos que los usuarios a los que más se parece el Usuario 1 son, principalmente, el Usuario 2 y el Usuario 4, habiendo obtenido un grado de similitud muy bajo con los Usuarios 3 y 5, que no deberían ser utilizados para la recomendación, puesto que es probable que tengan pocos criterios en común. De esta forma, esos valores 0.85 y 0.7 son los que, en el sistema, multiplicaríamos por la valoración de dichos usuarios de las obras que Usuario 1 no haya puntuado y, ordenando de mayor a menor, obtendríamos la lista de las obras que más se pueden adecuar a los gustos de Usuario 1.



En FIBONART, toda la lógica en cuestión tiene lugar mediante la utilización de Cypher, sin prácticamente ningún otro tratamiento de los datos que la devolución de los detalles de las obras obtenidas, igual que hemos hecho en otras consultas de la aplicación.

```
var query =
  "MATCH (u1:User {id:" +
  idUser +
  "})-[r:RATED]->(w:ArtWork)" +
  " WITH u1, avg(r.score) AS u1_mean" +
  " MATCH (u1)-[r1:RATED]->(w:ArtWork)-[r2:RATED]-(u2)" +
  " WITH u1, u1_mean, u2, COLLECT({r1: r1, r2: r2}) AS ratings WHERE size(ratings) > 3" +
  " MATCH (u2)-[r:RATED]->(w:ArtWork)" +
  " WITH u1, u1_mean, u2, avg(r.score) AS u2_mean, ratings" +
  " UNWIND ratings AS r" +
  " WITH sum( (r.r1.score-u1_mean) * (r.r2.score-u2_mean) ) AS nom," +
  " sqrt( sum( (r.r1.score - u1_mean)^2 ) * sum( (r.r2.score - u2_mean)^2 ) ) AS denom," +
  " u1, u2 WHERE denom <> 0" +
  " WITH u1, u2, nom/denom AS pearson" +
  " ORDER BY pearson DESC LIMIT 10" +
  " MATCH (u2)-[r:RATED]->(w:ArtWork) WHERE NOT EXISTS( (u1)-[:RATED]->(w) )" +
  " WITH w, r, pearson, u1" +
  " MATCH (a)-[:MADE]->(w), (w)-[:USES_TECHNIQUE]->(t), (w)-[:LOCATED_AT]->(l)," +
  " (w)-[:ITS_FORM_IS]->(f), (w)-[:ITS_TYPE_IS]->(y), (w)-[:ITS_SCHOOL_IS]->(s)" +
  " RETURN ID(w), w.title, w.url, a.author_name, w.date, t.technique, l.location, " +
  " f.art_form, y.arttype, s.school, SUM( pearson * r.score) AS p" +
  " ORDER BY p DESC LIMIT 20";
```

Figura 34: Consulta de Filtrado Colaborativo

A modo de resumen, las primeras líneas se corresponden con la obtención del vector de valoraciones de los usuarios, así como en el cálculo de la media para cada uno, de forma que, después del unwind, tenga lugar el cálculo de la similitud Pearson de cada usuario mediante la utilización de fórmulas matemáticas simples que nos proporciona Neo4j (medias, raíces cuadradas, sumatorios...), que almacenaremos en la variable pearson.

Una vez hemos obtenido el coeficiente entre usuarios que estábamos buscando, el siguiente paso es utilizarlo para predecir las obras que más le van a gustar al usuario en cuestión, para lo cual multiplicaremos el valor de pearson del usuario a comparar por la valoración otorgada de una obra que ha valorado (pero que el primero no), de forma que, las 20 obras que tengan mayor valor para este resultado, serán devueltas junto a los detalles necesarios, puesto que el sistema habrá determinado que son las más adecuadas para el usuario que ha pedido la recomendación.



## 5. ANÁLISIS DE RESULTADOS

### EJEMPLO 1 – REC. BASADA EN CONTENIDO

Para llevar a cabo este ejemplo, hemos utilizado un usuario con un número no muy alto de valoraciones, pero suficiente para el propósito del ejemplo. Entre sus valoraciones, aunque hay de diversos tipos, destaca la presencia de 3 pinturas de paisajes de Van Gogh y otras 2 pinturas de Monet.

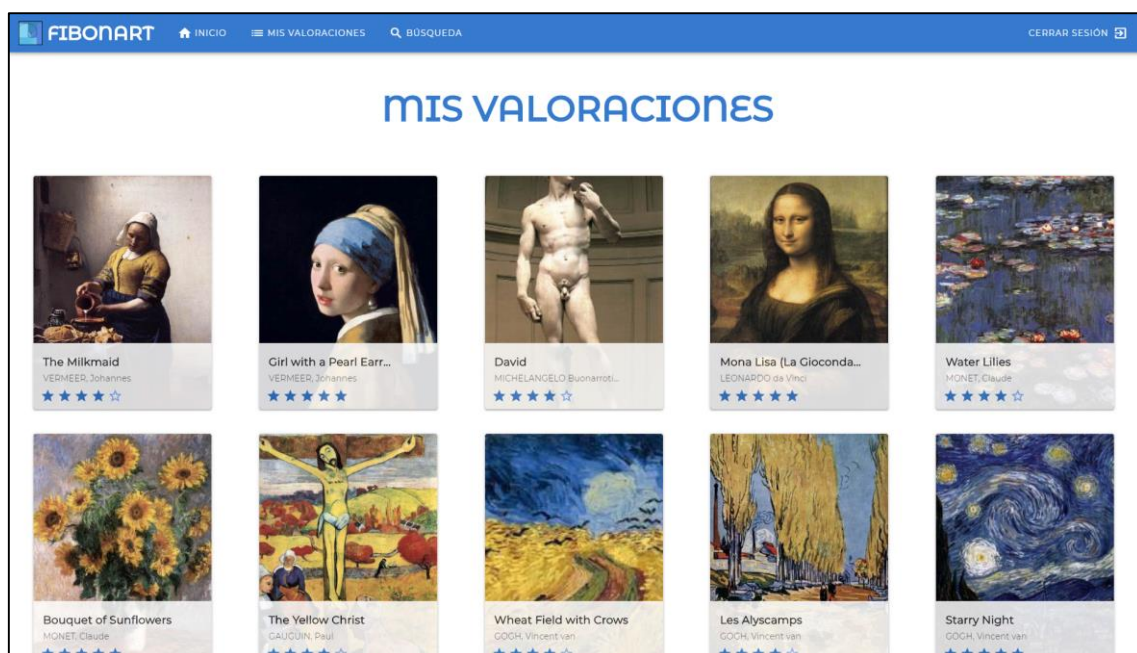


Figura 35: Valoraciones del usuario

El vector de características obtenido es el siguiente, donde destaca el 0.9, correspondiente a la disciplina pintura, el primer 0.3, correspondiente al autor Van Gogh, el último 0.5, correspondiente a la temática paisaje, entre otros valores.

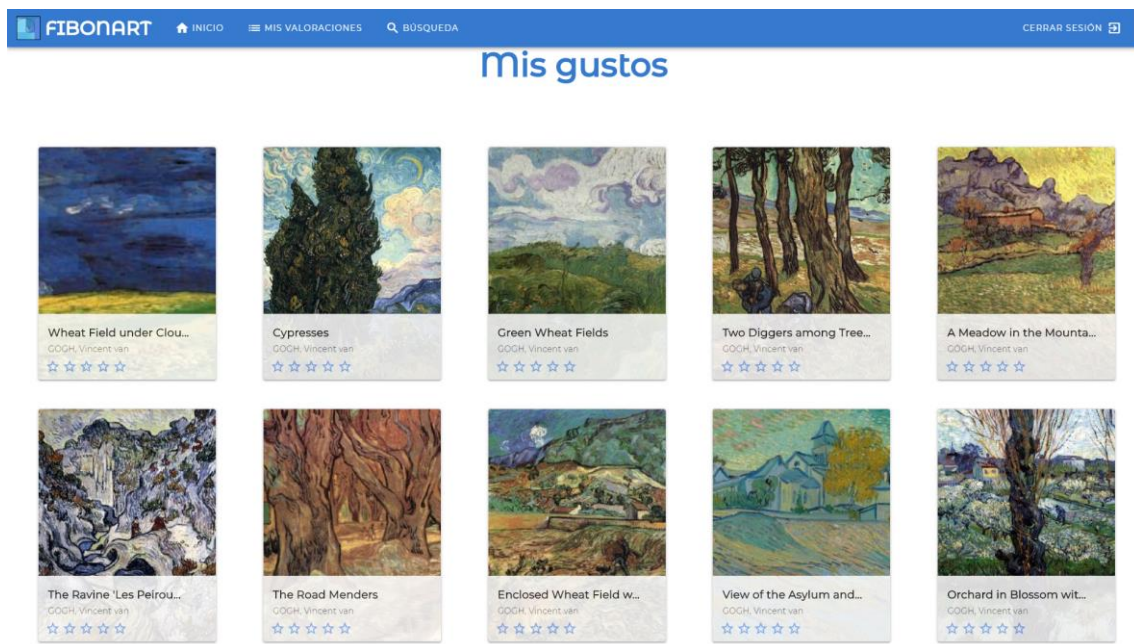
```
Vector Usuario: 0,0,0,0,0,0,0,0.1,0.3,0,0,0.1,0.1,0.2,  
0,0,0,0,0.2,0,0,0,0,0.9,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,0.1,0,0,0,0,0,0,0.2,0,0,0,0.2,0.  
1,0.1,0,0.4,0,0.2,0.3,0.5,0,0,0.1,0,0,0.6,0.1,0,0.2
```

Figura 36: Vector del usuario



Por ello, parece lógico pensar que, si pulsamos el botón de recomendar obras basadas en contenido, es probable que recibamos pinturas de paisajes de Van Gogh, ya que es lo que predomina en nuestro vector de características.

Si pulsamos dicho botón, podemos comprobar que, efectivamente, el sistema nos devuelve obras con características similares a las que hemos comentado.



*Figura 37: Resultado*

Si siguiéramos bajando, es muy probable que encontráramos otro tipo de obras, como paisajes de Monet o algún otro retrato de autores que hemos valorado positivamente.



## EJEMPLO 2 – REC. POR FILTRADO COLABORATIVO

Para el ejemplo utilizaremos un usuario que ha valorado, entre otras, las siguientes obras:

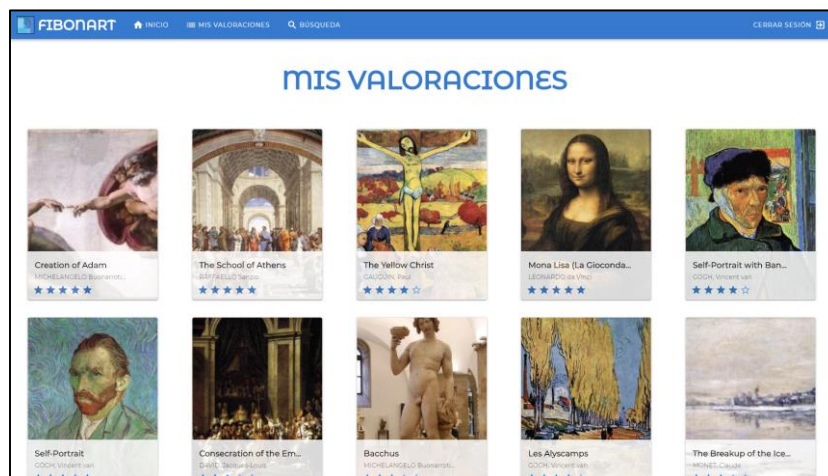


Figura 38: Algunas de las obras valoradas

Tras pulsar el botón de recomendación por filtrado colaborativo, se ejecuta la consulta establecida en el apartado 4.2, en el cual buscamos calcular la similitud Pearson de todos los usuarios que tengan valoraciones en común con el que ejecuta la llamada. Así, tras multiplicar ese coeficiente por la valoración otorgada por ese otro usuario a una obra que el primero no ha valorado, obtenemos un número indicador de la probabilidad de que esa obra encaje con sus gustos. Para el ejemplo, anterior, el resultado sería el siguiente:

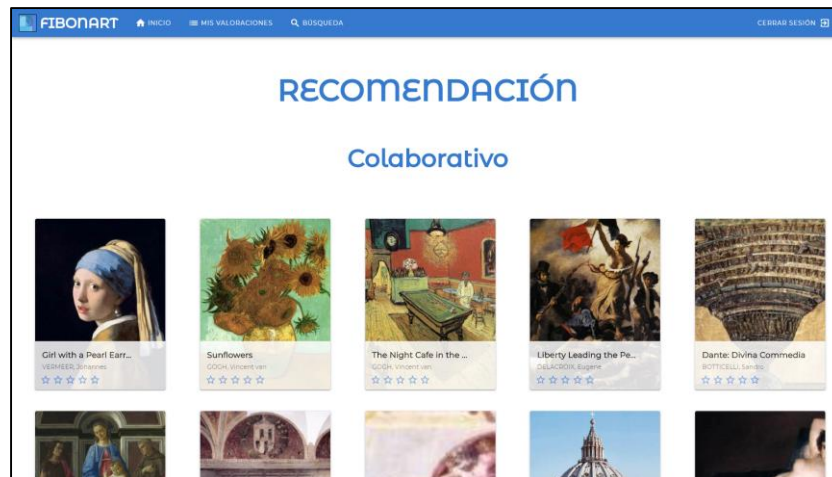
| "w.title"  | "p"                |
|--|--------------------|
| "Girl with a Pearl Earring"                                    | 11.286421464087866 |
| "Sunflowers"   | 7.414650127036882  |
| "The Night Cafe in the Place Lamartine in Arles"               | 7.414650127036882  |
| "Liberty Leading the People (28th July 1830)"                  | 5.122455700678778  |
| "Dante: Divina Commedia"                                       | 4.589229949838792  |
| "Madonna and Child with Six Saints (Sant'Ambrogio Altarpiece)" | 4.13219373247814   |
| "The Last Supper"  | 3.7432661671658485 |
| "The sixth bay of the ceiling"                                 | 3.671383959871034  |
| "Dome of St Peter's"   | 3.671383959871034  |
| "The Nude Maja (La Maja Desnuda)"                              | 3.671383959871034  |
| "Las Meninas or The Family of Philip IV"                       | 3.5308701204841704 |

Figura 39: resultado tras la similitud Pearson





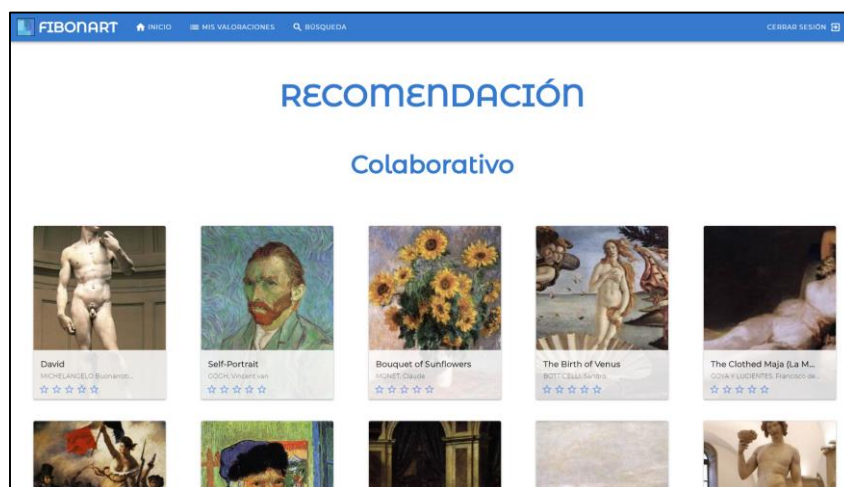
Y la vista luciría de la siguiente manera, donde podemos ver que, por supuesto, el usuario no ha establecido ninguna valoración de las obras que se le muestran.



*Figura 40: Resultado en la vista en cuestión*

Así, se trata de obras que le han gustado a usuarios que, tras un cálculo matemático, se ha llegado a la conclusión de que tienen un alto grado de similitud contigo, por lo que es muy probable que a ti también te gusten.

Como es lógico, si ejecutamos la misma petición desde otra cuenta, el resultado obtenido será bastante diferente, ya que pueden no coincidir los mismos usuarios similares o las obras valoradas.



*Figura 41: Resultado para otro usuario*







## 6. ANÁLISIS DAFO

Para poder comprender un poco más a fondo el proyecto, y de la manera más objetiva posible, sin dejarse llevar por sentimientos u opiniones personales, he elaborado este análisis DAFO que trata de reflejar la realidad de FIBONART. Para ello, no ha sido sencillo tratar de analizar el sistema desde un punto de vista externo, identificar cuáles son los puntos fuertes y débiles de la aplicación.



Figura 45: Gráfico DAFO de **FIBONART**



## FORTALEZAS

- ◇ Diversa oferta de tipos de recomendación: En mi opinión, era necesario que el sistema permitiera distintos tipos de recomendación en función de lo que requiriera el usuario en cada momento, contando con uno basado en contenido (obras parecidas a las que te han gustado), un filtrado colaborativo (obras que le han gustado a usuarios parecidos) y, aunque no sea una recomendación como tal, una lista de las obras mejor valoradas globalmente por el conjunto de los usuarios.
- ◇ Algoritmo eficaz: Cuando dispone de la suficiente información sobre el usuario, el sistema es capaz de predecir con bastante fiabilidad obras que realmente concuerdan con sus gustos, tanto el algoritmo de contenido como el colaborativo, gracias a la lógica que llevan detrás y las fórmulas de comparación, basados en una similitud coseno y similitud Pearson respectivamente.
- ◇ Recomendaciones personalizadas: Desde mi punto de vista, no tenía ningún sentido que el sistema de recomendación se basara en una simple consulta a la base de datos, sin lógica detrás, por lo que decidí implementar diversos sistemas que se basen en lo que realmente le gusta al usuario, de forma que no obtenga la misma recomendación un usuario que otro.
- ◇ Interfaz sencilla y moderna: Aunque no fuera la prioridad número uno de la aplicación, sí que considero importante que la aplicación sea agradable a la vista, con los detalles cuidados y una interfaz accesible y fácil de utilizar para los usuarios. Con una identidad muy cuidada y determinada a través de la utilización de la misma gama de colores, fuentes y componentes para todo el sistema, resulta cómodo y gratificante navegar por la aplicación.



- ◇ Ausencia de datos sensibles: Al no tratar con información privada de los usuarios, más allá de un correo y una contraseña encriptada mediante una función hash, la seguridad no supone un problema en caso de que haya un ataque u otra situación similar, ya que en la base de datos solo se almacena un correo, una contraseña encriptada y un nombre de usuario, además de la información sobre las obras y las relaciones entre ellas y con los nodos usuario.
- ◇ Posibilidad de edición de las valoraciones: Si deseas cambiar la valoración aportada de una determinada obra, en base a un cambio en el criterio o un simple error, es tan sencillo como buscar la obra a modificar y pulsar en la valoración deseada, de forma que queda actualizado automáticamente para que el sistema de recomendación pueda actuar con eficacia instantáneamente.
- ◇ Escalabilidad: El sistema permite que se pueda tratar con una mayor cantidad de información, sean obras, autores, propiedades, tipos de arte... Siempre que los datos tengan el detalle necesario, es posible aplicarlos al sistema mediante una simple importación y modificación de algunos aspectos del código.



## DEBILIDADES

- ◇ Cantidad reducida de valoraciones reales: Para poder crear un sistema de recomendación que fuera operativo, me vi obligado a crear valoraciones artificiales mediante un script en Python, que no tiene por qué reflejar los gustos reales de los usuarios. A pesar de que sí que he creado perfiles donde las valoraciones se han creado con sentido, sería necesario que una gran cantidad de usuarios utilizara el sistema durante un tiempo para que el algoritmo de recomendación de filtrado colaborativo implementado fuera lo más fiable posible.
- ◇ Escasez de atributos de detalle sobre las obras: La recomendación basada en contenido utiliza los valores de factores bastante generales incluidos en la base de datos, como autor, técnica, época, tipo, forma, escuela... Sin embargo, es posible que el usuario tuviera unos gustos más concretos, como podrían ser las obras que tengan una predominancia de determinado color o que traten un tema en concreto, valores que no pueden tenerse en cuenta ya que ningún dataset de internet cuenta con ese tipo de datos, que sí que podrían ayudar a refinar la recomendación.
- ◇ Necesidad de mucha interacción al principio: Para que el sistema pueda hacer una recomendación precisa, es conveniente que tenga la mayor cantidad de información sobre los gustos del usuario. Así, si solo has valorado 5 o 10 obras y pides una recomendación personalizada, es posible que el resultado no se adecúe a lo que realmente encaja contigo. Sin embargo, a medida que vas valorando obras nuevas, el sistema de recomendación tiene más factores a tener en cuenta antes de recomendarte una obra u otra.
- ◇ Difícil adaptación de idioma: Todos los títulos de obras están escritos en inglés, y resultaría bastante tedioso traducir uno por uno al español u otro idioma deseado, ya que sabemos que las obras tienen un nombre concreto en cada uno de los lenguajes.



## AMENAZAS

- ◇ Interés reducido del público: Al tratarse de arte clásico, es posible que la aplicación no tuviera un público muy amplio. Al fin y al cabo, no trata un contenido viral o moderno, y aunque la intención concreta es precisamente acercar el arte a aquellos que no son expertos en la materia, es posible que no consiga conectar, o que simplemente no haya gente interesada en aprender sobre el tema.
- ◇ Usuarios que decidan descompensar el sistema: Existe la posibilidad de que uno o varios usuarios decidan boicotear la recomendación mediante la creación de una gran cantidad de cuentas y valoraciones no reales que cambien la puntuación media de las obras y que provoquen que las recomendaciones no sean precisas.

## OPORTUNIDADES

- ◇ Ausencia de competencia: Como ya se ha comentado anteriormente, no existe ningún portal en internet que ofrezca un servicio similar más allá de reviews o artículos planos que, aunque hagan un gran trabajo, nada tienen que ver con el sistema de recomendación personalizado de FIBONART, diseñado para satisfacer los gustos de cada usuario.
- ◇ Posible ampliación a otras formas de arte: La base de datos contiene información de obras pertenecientes, principalmente, al ámbito de la Pintura y la Escultura, aunque también contiene obras de Arquitectura y Dibujo, entre otras. Así, hay disciplinas no contempladas, como la música o el cine, que podrían ser implementadas en el sistema si contáramos con un dataset suficientemente completo y unas pequeñas modificaciones en los distintos algoritmos de recomendación.



## 7. LÍNEAS DE FUTURO

Si tratamos de ser lo más realistas posibles, el sistema es ampliamente mejorable. Al fin y al cabo, al tener finalidad didáctica, se trata de dar estructura a una idea con la intención de aprender cómo realizar un sistema de recomendación, pero con un alcance limitado, que podría ser ampliado con los recursos, el tiempo y la dedicación suficiente. Así, hay distintos enfoques de la aplicación que podrían ser claramente perfeccionados:

- ◇ Tipos de arte a tratar: Puesto que la temática de FIBONART es el arte como concepto general, especialmente el clásico, sería muy interesante incluir otras disciplinas no contempladas hasta ahora, como podría ser la música, la danza o incluso el cine, ya que podrían adecuarse perfectamente al objetivo del proyecto, dar a conocer al usuario obras de arte de las que no tuviera conocimiento previo a ser partícipe de nuestro sistema.
- ◇ Variedad e información de las obras: Para maximizar los resultados a la hora de ayudar al usuario a encontrar arte, sería interesante contar con una base de datos más amplia, con más información de carácter didáctico acerca de las obras, como información sobre el movimiento artístico al que pertenece o la biografía del autor. Además, el sistema solo dispone de obras hasta 1900, por lo que sería muy interesante incluir entradas más recientes, ya que algunos de los artistas más importantes, como Dalí o Picasso, no llegan a aparecer; y que podría ser extensible hasta nuestra actualidad.
- ◇ Accesibilidad e interfaz gráfica: Si el objetivo es extender el arte al mayor número de personas posibles, sería muy interesante tener más en cuenta aquellos que no tengan un acceso sencillo a FIBONART, bien porque no tienen acceso a un ordenador, por lo que sería imprescindible la creación de una aplicación móvil que ejerza la misma funcionalidad, o bien por alguna condición



física o psíquica, como puede ser un trastorno o una enfermedad. Así, la interfaz de la aplicación podría mejorarse hasta el punto de poder ser utilizada por personas con problemas de visión o trastornos psicológicos, entre otros.

- ◇ Algoritmo de recomendación: A pesar de que la intención de implementar un algoritmo de recomendación basado en contenido y otro de filtrado colaborativo se ha podido completar, distan mucho de ser algoritmos perfectos. Especialmente, en caso de que quisiéramos ampliar el número de obras almacenadas en la base de datos, sería necesario modificar ambos para que pudieran gestionar una mayor complejidad de una forma más eficiente. Para darle el último toque de profesionalidad, sería ideal la aplicación de técnicas de aprendizaje automático, que dotarían al sistema de un sistema de recomendación altamente eficaz.





## 8. LECCIONES APRENDIDAS

A pesar de ser una persona serena y que muy rara vez se abruma por una determinada carga de trabajo, sí he de reconocer que el hecho de realizar un proyecto completo de esta magnitud me abrumaba un poco, ya que, hasta ahora, todos los trabajos full-stack los había realizado en conjunto con uno o varios compañeros. Así, aunque nunca he tenido problema para programar de manera individual, he podido comprobar que puedo gestionar todo aquello que conlleva el desarrollo de una aplicación.

Además, antes de comenzar este proyecto no conocía el funcionamiento de ningún tipo de sistema de recomendación, por lo que he tenido que informarme desde cero. Para ello, he utilizado principalmente el libro *Graph Powered Machine Learning (MEAP)*. Aunque no hasta ahora no ha sido mi principal fuente de información, trataré de darle más importancia a los libros, ya que la cantidad de conocimiento que almacenan es incomparable y, por lo menos en este caso, me ha sido muy útil para aprender sobre algoritmos de recomendación. Ahora sé en qué consiste un filtrado colaborativo y una recomendación basada en contenido, o cuándo debo utilizar la similitud de Pearson y la de Cosenos para comparaciones.

Del mismo modo, he concebido la utilidad de los cuadernos de trabajo. Aunque al principio me resultaba un poco antinatural tener que documentar cada paso realizado durante el desarrollo, he comprendido que, en nuestro caso el OSF, sirve para marcarse unas pautas y no olvidar la importante tarea de la documentación, que muchas veces los ingenieros descuidamos. Creo que con un poco más de práctica en algún otro proyecto, podría llegar a integrarlo en mi forma de trabajar como algo natural.



Por último, he comprendido la importancia de establecer una planificación a la hora de trabajar en un proyecto de este calibre. Llegó un punto en el que me di cuenta de que, si no establecía unos horarios y unas fechas límite autoimpuestas, me iba a ser muy complicado finalizar a tiempo la aplicación con todas las funcionalidades que quería que tuviera. Para la próxima ocasión, comenzaré estableciendo este tipo de estrategias desde un primer momento, de forma que, en caso de surgir imprevistos, no suponga un problema en los plazos.

Para concluir, y aunque no sea el objetivo principal de este punto, sí me gustaría hacer un comentario sobre las distintas tecnologías que no conocía y he aprendido gracias a la realización de FIBONART, como Neo4j, Cypher, Vue, NodeJs... Aunque la primera reacción sea un pequeño vértigo, con algo de esfuerzo y de tiempo es posible aprender cualquier lenguaje o herramienta nueva, como me he podido demostrar con este proyecto, por lo que en futuras ocasiones trataré de convertir ese supuesto “miedo” en ganas de aprender.



## 9. FUENTES BIBLIOGRÁFICAS

- ◇ *Graph Powered Machine Learning (MEAP)*, Alessandro Negro
- ◇ [Web Gallery of Art](#)
- ◇ <https://neo4j.com>
- ◇ <https://vuejs.org>
- ◇ <https://vuetifyjs.com/en/>
- ◇ <http://guides.neo4j.com/sandbox/recommendations>
- ◇ <http://zetcode.com/javascript/axios/>
- ◇ <https://www.youtube.com/playlist?list=PLPl81lqbj-4J-gfAERGDCdOQtVgRhSvIT>
- ◇ <https://www.udemy.com/course/primeros-pasos-con-el-framework-vuejs/>