Taller de sistemas de Información JAVA

Trabajo Obligatorio Edición 2020

Juan Alvarez

jjap96@gmail.com

Julio Arrieta

julioarrieta23@gmail.com

Lucas Garrido

LuC31G@gmail.com

Carlos Balbiani

karloxx09@gmail.com

Índice

Resumen

Dado el auge que existe en torno a los artistas independientes, los cuales desean comercializar sus creaciones por medio de internet, se han popularizado los sitios web que permiten la compra y venta de los mismos.

En este artículo se plantea el diseño e implementación de la plataforma "TodoArte", propiedad de la productora “De Contenidos”, la cual permitirá a los artistas crear sitios en donde compartir y vender su trabajo.

La nueva plataforma ayudará a centralizar las obras creadas por los artistas, y les permitirá llevar el control de sus ingresos por ventas, así como recibir feedback de los fans.

El proyecto fue finalizado exitosamente utilizando Java Enterprise Edition, aplicando los conceptos adquiridos en clase y durante la investigación, logrando cumplir con los requerimientos solicitados en el plazo establecido.

Palabras clave

Java EE, JPA, Wildfly, JSF, API REST, WebSocket, XHTML

Introducción

Dado el auge que existe en torno a los artistas independientes la productora “De Contenidos” decidió crear una plataforma de nombre código “TodoArte”, que permita a los artistas crear sitios en donde compartir y vender su trabajo.

La plataforma a desarrollar deberá ser accesible desde diferentes dispositivos adaptándose correctamente a los diferentes escenarios de uso.

Se trata de una plataforma genérica de contenidos multimedia para servir a múltiples artistas se puede buscar mediante categorías de sitios de diferentes artistas. Los usuarios pueden registrarse en los sitios de los artistas y convertirse en fan del artista.

Para obtener ingresos y amortizar la inversión, el sistema cobrará a los artistas un monto base mensual, así como otros intereses.

La plataforma permite que el contenido sea calificado y que los usuarios (fans) pueden publicar comentarios.

Los artistas podrán programar Q&A (preguntas y respuestas) con integrantes de la fanbase.

Los artistas pueden bloquear a un fan de su sitio.

El registro de artista es totalmente automático.

Cada sitios es personalizable, sin verse afectado por los otros artistas de la plataforma.

El sistema cuenta con dos módulos. Por un lado el backoffice y por otro lado el frontoffice

El trabajo se plantea en el marco de la edición 2020 de la asignatura Taller de Sistemas de Información JAVA.

Marco conceptual

JPA: (Java Persistence API), La API de persistencia JPA es el estándar de Java para mapear objetos Java a una base de datos relacional. El mapeo o asignación de objetos Java a tablas de bases de datos y viceversa se denomina mapeo relacional de objetos en inglés también conocido por sus siglas como ORM.

Wildfly: (formalmente WildFly Application Server), es un servidor de aplicaciones Java EE de código abierto implementado en Java puro, más concretamente la especificación Java EE. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java.

JSF: Es un framework MVC (Modelo-Vista-Controlador) basado en el API de Servlets que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets.

API REST: Es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

WebSocket: Es una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Con esta API, puede enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para una respuesta.

XHTML: Es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros.

Descripción del problema

La plataforma busca centralizar el trabajo de los artistas permitiendo que sus trabajos sean accesibles en una plataforma totalmente independiente del tipo de dispositivo.

Se trata de una plataforma genérica de contenidos multimedia para servir a múltiples artistas (video, imágenes, cómics, música,etc), la misma ofrece a los usuarios la posibilidad de buscar mediante categorías de sitios de diferentes artistas.

Posteriormente los usuarios podrán registrase en los sitios de los artistas y convertirse en fan del artista, La plataforma permite a los artistas gestionar sus trabajos así como también comercializar y/o ofrecer accesos premium al contenido a quienes sean parte de su fanbase, mediante esto los artistas pueden obtener ganancias por su trabajo.

El sistema cobrará a los artistas un monto base mensual por mantener el sitio, así como un 10% de las ganancias obtenidas por usuarios premium y un 5% de las ganancias por la venta de contenido.

La plataforma permite que el contenido sea calificado y que los usuarios (fans) pueden publicar comentarios.

La plataforma brinda la posibilidad de generar Q&A (preguntas y respuestas) con integrantes de la fanbase.

Los artistas pueden bloquear a un fan de su sitio.

El proceso de registro para un artista que se propone generar un sitio deberá ser totalmente automático.

Cada uno de estos sitios contará con un alto grado de personalización en la interfaz de usuario, esto implica, una estética distinta, sin verse afectado por los otros artistas de la plataforma.

El sistema deberá contar con dos módulos. Por un lado el backoffice para administración general de la plataforma.

Por otro lado el frontoffice que permite a los usuarios (artistas y fans) acceder al contenido. Todas las funcionalidades del front office se podran acceder desde cualquier dispositivo.

A continuación se exponen las funcionalidades que debe proveer el sistema, así como las opcionales.

El backoffice

El backoffice será utilizados por los usuarios administradores de la plataforma quienes deberán estar logeados al sistema. Estos podrán:

- Bloquear/Desbloquear artistas.
- Revisar contenido reportado.
- Bloquear contenido.
- Notificar a los artistas.
- Obtener información sobre pagos.
- Definir categorías.

El frontoffice

El frontoffice será utilizados por los artistas, fans (usuarios registrados) e invitados.

Artistas

- Registrarse y loguearse en la plataforma.
- Publicar, modificar y eliminar contenido.
- Programar y participar de Q&As.
- Consultar y recargar su saldo. (mediante tarjeta de crédito o Paypal).
- Obtener estadísticas de los contenidos publicados.
- Consultar información de sus fans (ubicación, sexo, gustos).
- Bloquear/Desbloquear fans.

Fans

- Loguearse en el sitio de un artista.
- Suscribirse y acceder a los Q&A.
- Calificar los contenidos.
- Comentar en los contenidos.
- Recibir notificaciones.
- Comprar contenidos y paquetes premium.
- Buscar artistas.
- Reportar contenido.

Visitantes

- Podrán ver contenido que sea público
- Registrarse.
- Buscar artistas.

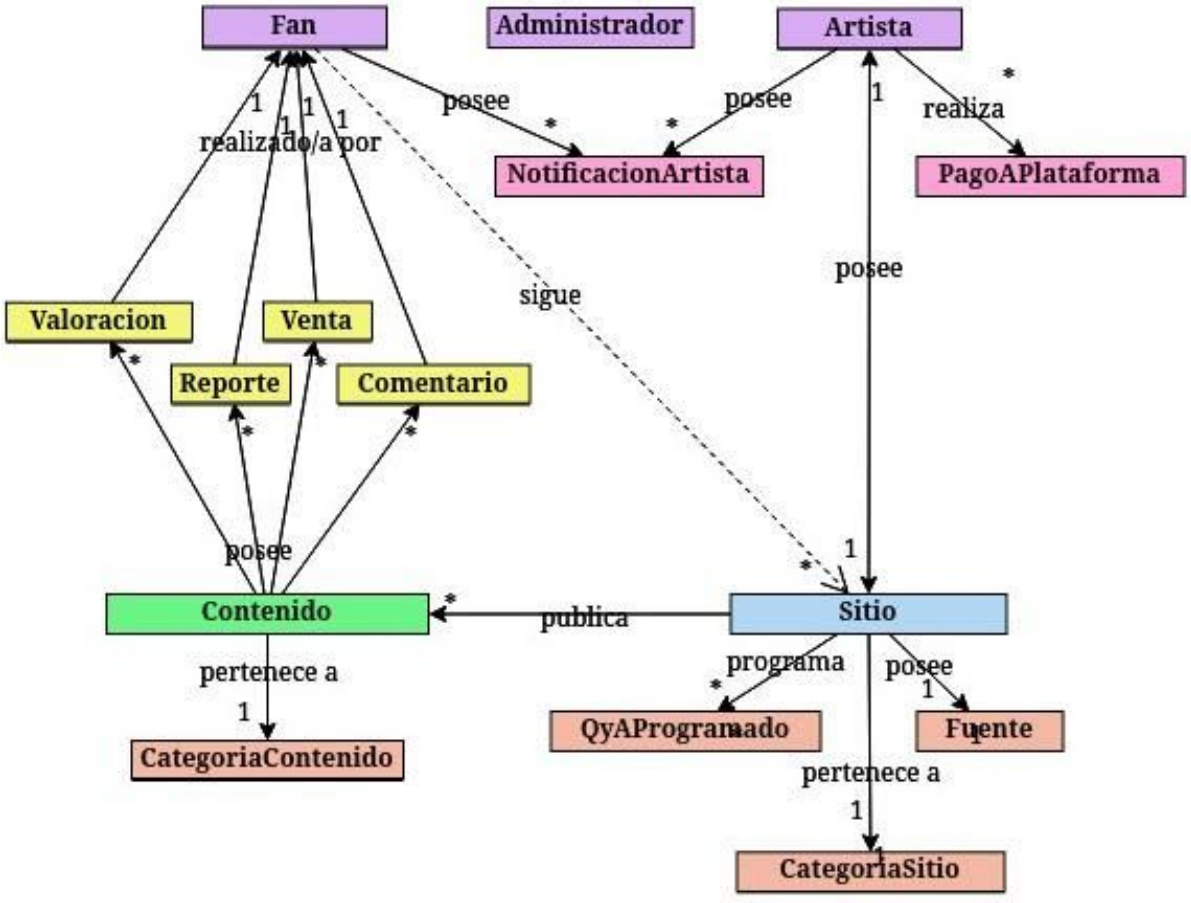
Requerimientos opcionales

- Inicio de sesión utilizando redes sociales.
- Pago de productos con medios de pago como Paypal.
- Utilización de una base de datos NoSQL (por ej: MongoDB) para la persistencia de una parte de los datos del sistema.
- Cubrimiento de al menos 80% de la aplicación con test automatizados con Junit.

Solución planteada

Se logró la implementación de las funcionalidades descritas anteriormente, aunque con alguna variación en lo respectivo al QyA, los cuales no son programados,sino que el chat está siempre disponible para los fans de un sitio y el artista propietario del mismo.

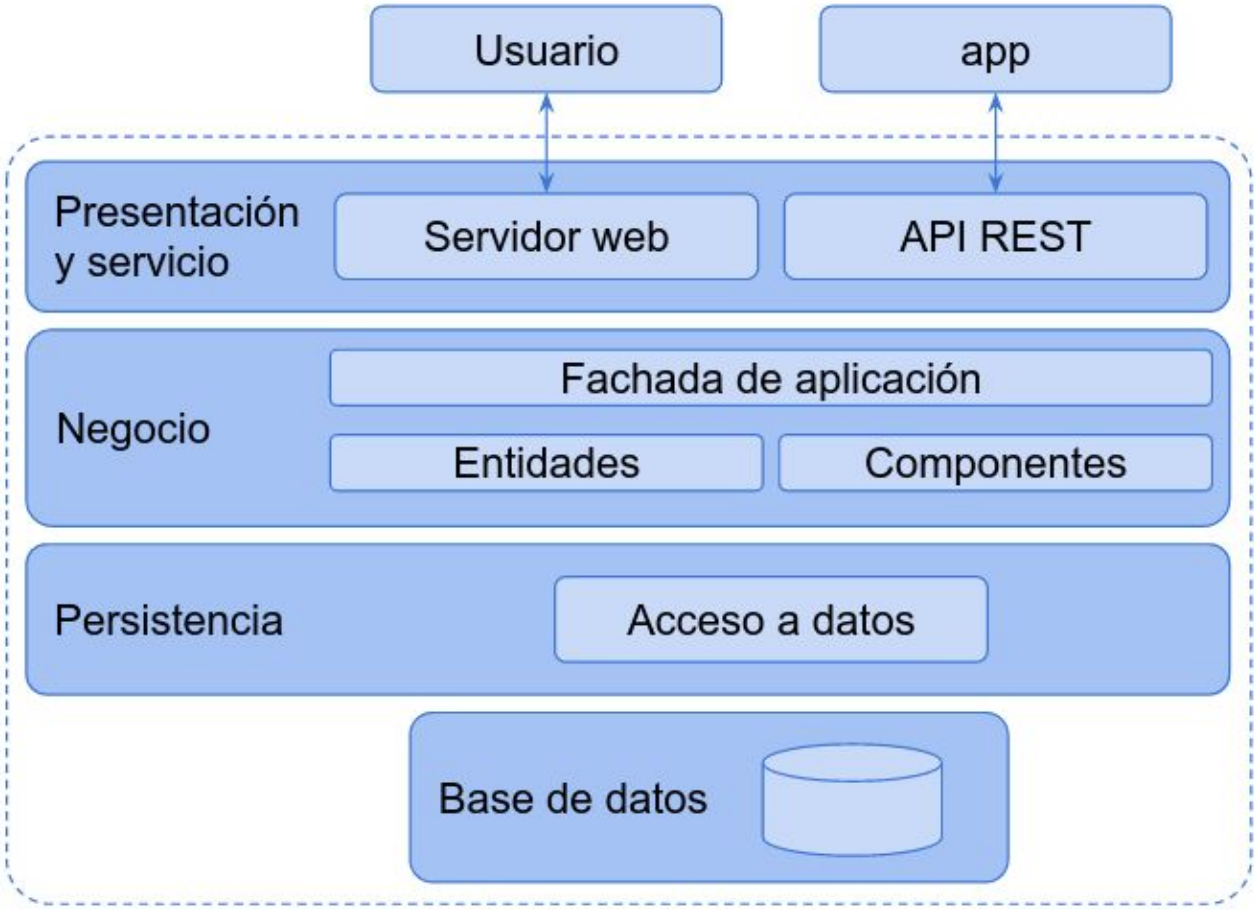
De las funcionalidades opcionales se tuvieron en consideración, pero no se llegaron a implementar debido a contratiempos surgidos durante la implementación de las funcionalidades principales.



Arquitectura del sistema

El sistema presenta una arquitectura dividida en capas, donde cada una realiza tareas específicas.

- **Capa de presentación y servicio:** Es donde se encuentra el servidor de páginas web y la implementación de la API REST.
- **Capa de negocio:** Es donde se verifican y procesan los datos necesarios para el funcionamiento del sistema.
- **Capa de persistencia:** Es La encargada del manejo de los datos que se deben guardar y cargar desde la base de datos.
- **Base de datos:** Es donde se almacenan los datos de los usuarios



Implementación

Productos y Herramientas

Evaluación de productos y herramientas utilizadas para la implementación de la solución:

Herramienta o Producto	Puntos fuertes	Puntos débiles	Evaluación general
Eclipse	Buen autocompletado, gran cantidad de complementos	Lento, alto consumo de recurso	IDE que a tiene muchos años y deja mucho que desear
GitKraken	Facilita la comprensión del estado del proyecto y su gestión	No es gratuito para repositorios privados	Altamente recomendable, muy buena herramienta
JSF	Clara separación entre el comportamiento y la presentación	Curva de aprendizaje muy empinada	En comparación a sistemas modernos, hay otras opciones más atractivas
WildFly	Rápida puesta en marcha, Bajo consumo de memoria	Requiere reinicio para compilar cambios. Errores poco descriptivos.	Aceptable pero no recomendable
JPA	Nos permite desarrollar más rápido, abstraerse de la base de datos, aumenta la seguridad	Se debe conocer su uso o puede dar problemas inesperados	Recomendable para cualquier proyecto sin importar el porte
PostgeSQL	Ilimitado y gratuito Gran escalabilidad Estabilidad y confiabilidad Potencia y robustez	Lento en inserciones y actualizaciones en bases de datos pequeñas, No cuenta con un soporte en línea	Útil en aplicaciones de gran escala, no es lo mejor para pequeñas.

Problemas Encontrados

Configuración del entorno: Durante la configuración del entorno tuvimos problemas con las versiones, no podíamos abrir los archivos XHTML. La solución fue, cambiar de versión para lograr la compatibilidad de los componentes.

Estructura del proyecto incorrecta: Tuvimos un problema con la estructura elegida, encontramos un fallo desconocido que repercutió en un consumo excesivo de memoria. La solución fue cambiar la estructura, dejando de usar el EAR y migrando a un proyecto WAR.

Utilización de JSF: Al utilizar JSF encontramos varios problemas pero el más importante fue entender cómo funcionaba la comunicación entre el xhtml con el bean. En otras tecnologías se llama al controlador y luego se renderiza la vista, pero en JSF no funciona de esa manera.

Scroll chat: Cuando un nuevo usuario ingresa un nuevo mensaje en el chat, el scroll no funciona correctamente. Para solucionarlo movemos el scroll una gran cantidad de pixeles.

Falta del uso de Maven: Debido a que no utilizamos maben, fue dificil encontrar informacion que no utilizará Maven. Debimos trabajar con menos fuentes, las cuales no utilizaban maven.

Bucles infinitos en API REST: Debido a que en la estructura de objetos existen dobles visibilidades, la API REST fallaba al intentar devolver ciertos objetos JSON. La solución fue la implementación de una función recursiva que realiza el desacople de dichos objetos.

Evaluación de la solución

La solución implementada cumple satisfactoriamente con los requisitos funcionales especificados.

La implementación realizada podría haber tenido una mejor estructura de proyecto, pero debido a los problemas mencionados anteriormente, no fue posible estructurarlo de la manera deseada inicialmente. Esto conlleva a que la aplicación no puede distribuirse entre diferentes servidores para repartir la carga de procesamiento dado que el resultado es una aplicación monolítica.

En lo que a mantenibilidad y escalabilidad concierne, el diseño realizado permite la fácil adición de nuevas y funcionalidades, pero dado que el componente de páginas web es independiente del componente de la API REST, al momento de implementar nuevas funcionalidades, se deben actualizar estos componentes por separado.

Lo anterior puede ser visto como una ventaja dado que se logra cierta independencia entre los componentes mencionados.

Al estar todo en un solo modulo es facil desplegarlo en el servidor.

Desarrollo del proyecto

Durante el transcurso del proyecto hubieron momentos en donde el avance fue rápido, pero en poco tiempo se veía obstruido por diferentes motivos, ralentizando la implementación y generando frustración en los integrantes del equipo.

Tiempos estimados y requeridos para la realización de las actividades (por integrante):

Actividad	Tiempo estimado (hs)	Tiempo requerido (hs)
Relevamiento y planificación	10	12
Investigación	18	49
Definición de requerimientos	11	12
Diseño del sistema	12	15
Implementación	92	123
Testing y corrección	24	29

Se cumplió adecuadamente con los plazos establecidos a pesar de que la implementación necesitó más tiempo de lo esperado debido a los problemas encontrados durante el proceso. El equipo logró una buena coordinación, la cual permitió la paralelización de las tareas entre los miembros del mismo.

Conclusiones y trabajo a futuro

Para concluir, se consiguió completar la totalidad del proyecto exitosamente en el plazo establecido, aplicando los conocimientos adquiridos en clase, y durante la investigación.

Como trabajo a futuro se puede mencionar: la implementación del Inicio de sesión utilizando redes sociales, pago de productos con medios de pago como Paypal, un mayor grado de personalización de los sitios de los artistas permitiendo previsualizar el resultado de la configuración elegida, permitir la carga de más tipos de contenidos, la programación de QyA, entre otras cosas.

Referencias

[1] API Java EE 8

https://javaee.github.io/javaee-spec/javadocs/

[Consulta: Mayo 2020]

[2] A Guide to the Java API for WebSocket

https://www.baeldung.com/java-websockets

[Consulta: Mayo 2020]

[3] Status Code Definitions HTTP
https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.2.2

[Consulta: Mayo 2020]

[4]Guide to RESTEasy

https://www.baeldung.com/resteasy-tutorial

[Consulta: Junio 2020]

[5] El MVC en JavaServer Faces
http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion02-apuntes.html

[Consulta: Junio 2020]