Taller de sistemas de Información JAVA

Trabajo Obligatorio Edición 2020

Bruno Rodriguez

brunorodriguez540@gmail.com

Pablo Gaione

pablo.gaione@estudiantes.utec.edu.uy

German Arena

germanarena12@gmail.com

Cristhofer Travieso

cristhofert97@gmail.com

Índice:

Índice:	**2**
Resumen:	**3**
Palabras clave:	**3**
Introducción:	**3**
Marco conceptual:	**4**
Descripción del problema:	**5**
Solución planteada:	**7**
Arquitectura:	**8**
Implementación:	**9**
Evaluación de la solución:	**11**
Desarrollo del proyecto:	**12**
Conclusiones y trabajo a futuro:	**13**
Referencias:	**13**

Resumen:

En el último año la industria de los videojuegos tuvo gran crecimiento, en parte por la pandemia que obliga a las personas a permanecer en sus casas y por otro lado la gran variedad de juegos disponibles. En este artículo se presenta el diseño e implementación de una página web de compra y venta de videojuegos construida utilizando la plataforma de programación Java Enterprise Edition. Se logró construir una aplicación funcional con las herramientas planteadas y cumpliendo con los requisitos solicitados.

Palabras clave:

Videojuegos, JavaEE, REST, web services, JSF, XHTML,MySQL

Introducción:

En los últimos años, en las plataformas de ventas de videojuegos han surgido muchos videojuegos desarrollados por estudios pequeños y desconocidos o incluso desarrollados por una sola persona.
En función de esto la empresa “Steam Indie” decidió desarrollar una plataforma web que permita a los desarrolladores ofrecer sus juegos, a la vez que permite a los jugadores comprar juegos y formar una comunidad con sus pares.
La plataforma ofrecerá a los usuarios la posibilidad de crear su perfil y poder interactuar con otros usuarios de distintas partes del mundo.
El trabajo se plantea en el marco de la edición 2021 de la asignatura Taller de sistemas de información JavaEE.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el marco conceptual. En la sección 3 se presenta la descripción del problema. En la sección 4 se presenta la solución planteada, en la sección 5 se presenta la arquitectura del sistema, en la sección 6 se presenta la implementación, en la sección 7 se presenta la evaluación de la solución, en la sección 8 se presenta el desarrollo del proyecto. Por último, en la sección 9 se presentan las conclusiones del trabajo, y las tareas a futuro.

Marco conceptual:

Hibernate: es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

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:

Se busca centralizar el trabajo de desarrolladores independientes de videojuegos proporcionándoles una plataforma donde podrán publicar y comercializar sus obras.

Tambien ofrecerá a los usuarios la posibilidad de crear su perfil y poder interactuar con otros usuarios. A su vez, estos podrán comprar juegos, de los cuales recibirán un enlace de descarga, y mantener un registro (biblioteca) de los juegos comprados. Como parte del sentido de comunidad, los jugadores podrán calificar los videojuegos que hayan comprado y dejar un comentario sobre el juego que puede ser visualizado por otros jugadores.

A la hora de ofrecer los videojuegos, la plataforma ofrece un servicio de búsqueda que permite buscar juegos por su nombre, por categoría o por tags personalizados de los usuarios. Los usuarios contarán con su propia billetera en el sitio que utilizaran para comprar los videojuegos. Esta podrán cargarla a través de tarjetas de crédito, Paypal o Google Wallet.

Para los creadores de videojuegos la empresa ofrece la posibilidad de publicar sus juegos. Las publicaciones pueden incluir la descripción del juego, imágenes y videos promocionales. También ofrecen la posibilidad de suscribirse a programas de ofertas o giveaways ofrecidos por la plataforma.

La plataforma cobrará a los creadores de videojuegos un porcentaje de sus ventas, en principio este porcentaje está estipulado en 10%.

El sistema deberá contar con dos módulos. Por un lado el backoffice para administración general de la plataforma por parte de personal de Steam Indie. Y por otro lado un frontoffice que permita a los usuarios (jugadores y creadores) acceder al contenido. A su vez interesa que todas las funcionalidades del frontoffice se puedan acceder desde cualquier dispositivo. A futuro está planeado el desarrollo de una aplicación móvil.

Backoffice

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

- Crear eventos de ofertas. Estos eventos están programados para un período y aplicarán un descuento a todos los juegos participantes del evento
- Definir categorías.
- Bloquear publicaciones de videojuegos.
- Bloquear comentarios reportados.
- Consultar sus ganancias
- Obtener estadísticas como juegos más vendidos, cantidad de ventas por día

Frontoffice

El frontoffice será utilizado por los usuarios registrados, pudiendo estos ser jugadores o creadores de videojuegos, y por usuarios invitados (usuarios no

registrados).

Los invitados podrán:

- Registrarse en la plataforma.
- Buscar juegos en la tienda y consultar su contenido.

Los jugadores podrán:

Loguearse en la plataforma.

- Buscar juegos en la tienda, consultar su contenido y comprar juegos.
- Gestionar su perfil.
- Hacer publicaciones en su muro.
- Calificar y hacer comentarios en videojuegos que hayan comprado.
- Chatear con otros usuarios
- Consultar y cargar su billetera
- Reportar juegos o comentarios

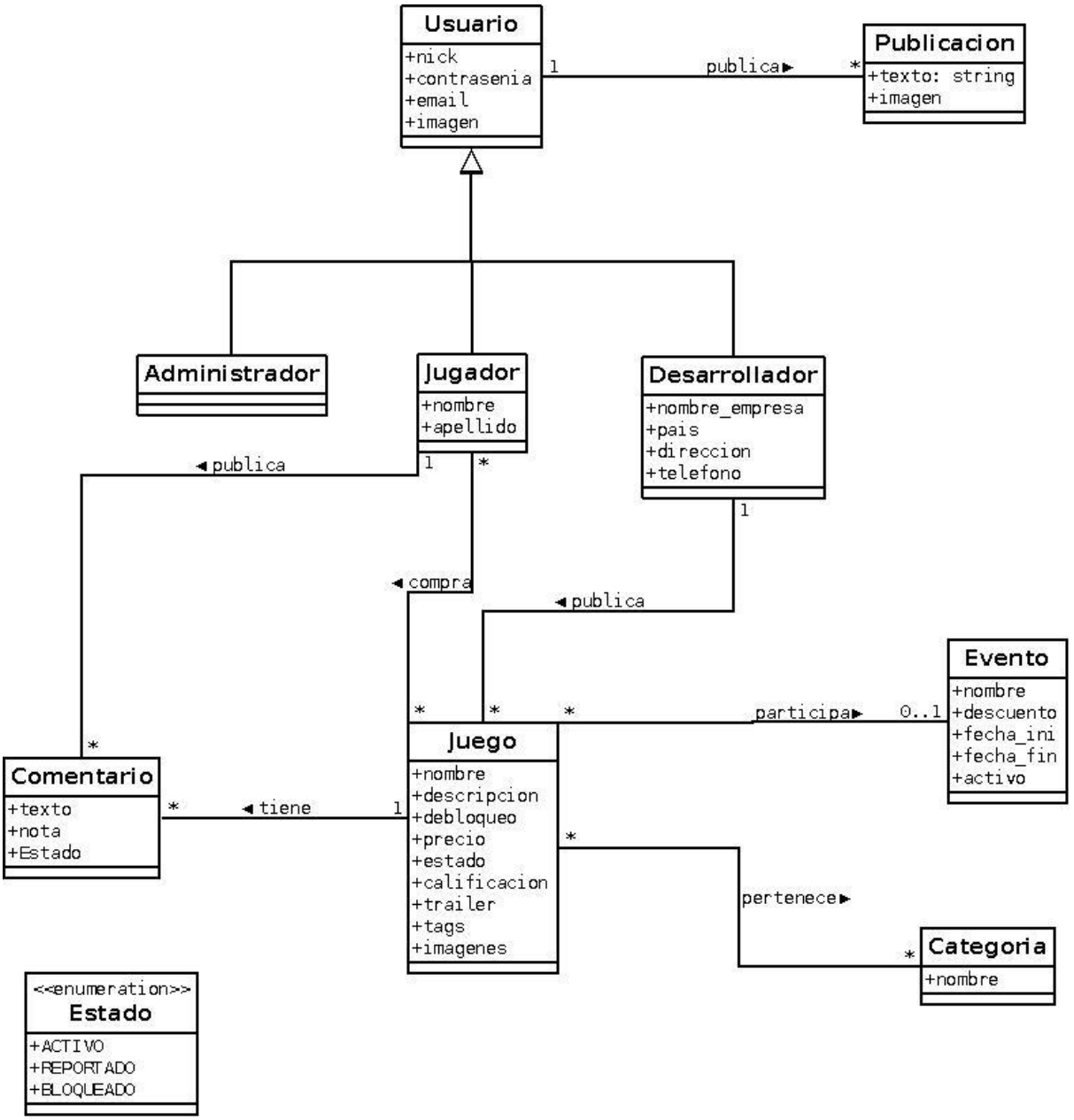
Los creadores de videojuegos podrán:

- Loguearse en la plataforma.
- Publicar videojuegos
- Consultar sus ventas (cantidad de copias vendidas, monto recaudado)
- Participar de eventos de ofertas (elegir que juegos participarán).
- Buscar juegos en la tienda, consultar su contenido
- Consultar juegos bloqueados y solicitar desbloqueo

Solución planteada:

Se logró la implementación de las funcionalidades descritas anteriormente.

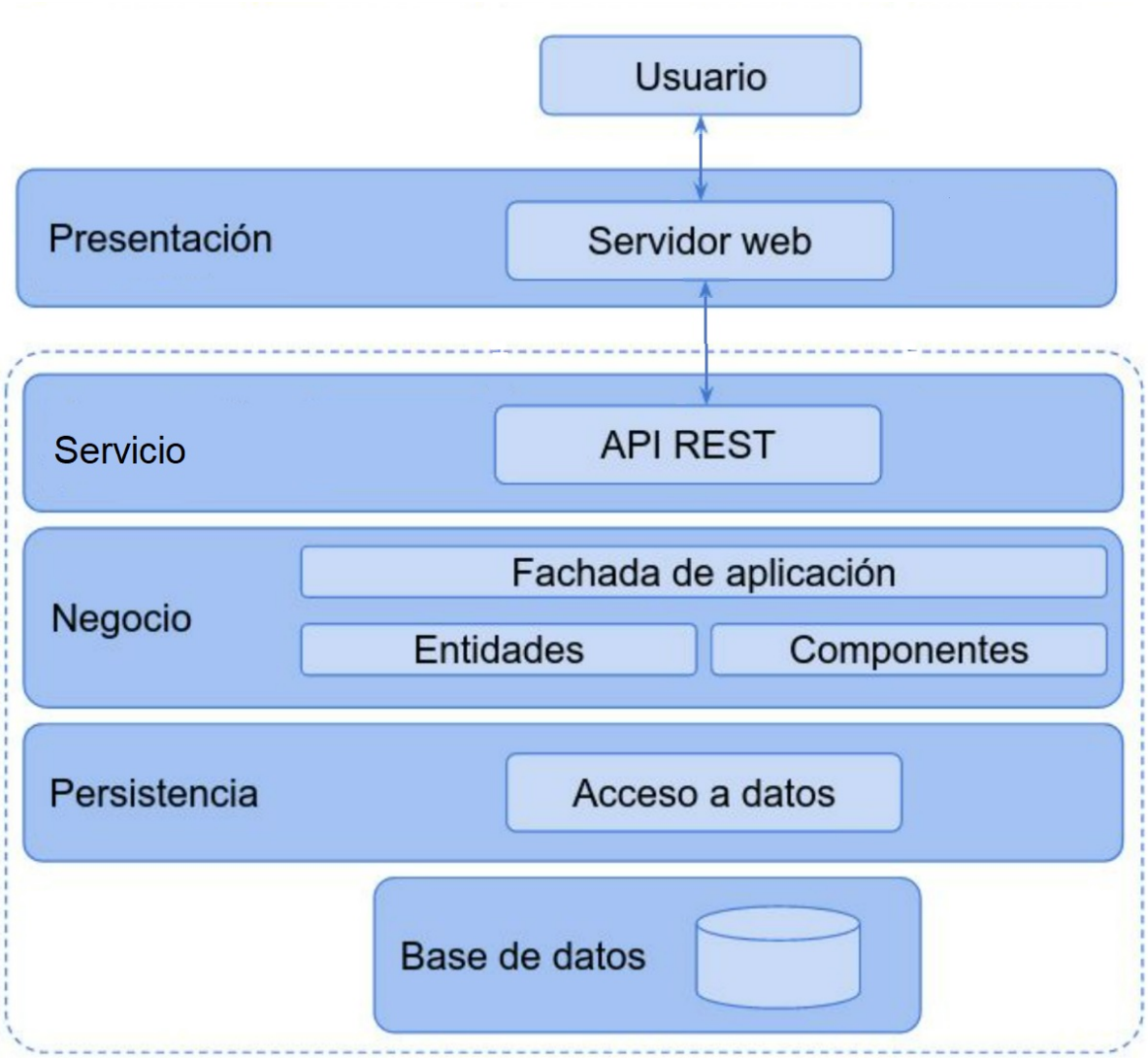
Mientras que las funcionalidades opcionales, si bien se tuvieron en consideración, no se llegaron a implementar debido a contratiempos surgidos durante la implementación de las funcionalidades principales.



Arquitectura:

En nuestro sistema usamos una arquitectura por capas, donde la capa de base de datos se comunica con la capa de persistencia enviando y recibiendo información y la lógica del sistema se maneja en la capa de negocio, donde la información se gestiona y luego es administrada en forma de servicios web REST por la capa de Servicios, los cuales son consumidos por la capa de presentación .

Las capas de Servicios, negocio, persistencia y base de datos se corresponden a la API REST del sistema, y la capa de presentación corresponde a el frontend del sistema.



Implementación:

Algunas de las herramientas utilizadas fueron:

Producto	Puntos fuertes	Puntos débiles	Evaluación general
JSF	Elementos útiles en las páginas xhtml.	Curva de aprendizaje pronunciada.	Es útil en ciertos aspectos pero creemos que hay mejores opciones.
Wildfly	Compatible de forma nativa con servicios REST(RESTEasy).	Lento y en ciertas ocasiones es necesario reiniciarlo/limpiarlo para que se actualicen los cambios.	Fácil y versátil en su uso.
Primefaces	Elementos de frontend muy útiles y simples de usar.	Documentación inconsistente en ciertos casos.	Muy útil y recomendable.
Hibernate ORM	Fácil configuración.	Consultas nativas más complicadas.	Buena herramienta.
RESTEasy	Facilita la implementación de REST services.	No encontramos puntos débiles.	Buena herramienta
Bootstrap	Muy útil para la creación y organización de los componentes en las vistas.	La personalización de algunos estilos de bootstrap es complicada.	Buena herramienta

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.

JavaEE: El uso de estas nuevas tecnologías nos trajo bastantes problemas, sobre todo al comienzo del proyecto. Si bien teníamos cierta práctica con algunas funcionalidades de JavaEE, otras no las conocíamos y nos costaron bastante.

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.

Envío de archivos a Rest API: Sobre todo con las imágenes nos costó encontrar el formato de consumo y producción de la API y el Cliente. Además adaptar esto al formato proveniente desde la vista.

Integración con PayPal: En un principio utilizamos un formulario html que servía para comunicar con la API de PayPal, pero este no proporcionaba las garantías necesarias en una transacción económica, por lo que tuvimos que utilizar otra API para desarrolladores proporcionada por PayPal.

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 utilizar la librería Jackson para mapear las relaciones bidireccionales, más específicamente la etiqueta @jasonbackreference.

Adaptación al nuevo IDE: El uso de Eclipse nos complicó un poco al comienzo ya que era la primera vez que lo utilizabamos y no estábamos muy familiarizados con sus menús y funcionalidades.

Uso de GitHub: En este aspecto tuvimos algunos problemas por malas prácticas en el uso de esta herramienta.

Evaluación de la solución:

La solución implementada cumple satisfactoriamente con los requisitos funcionales especificados.

Podemos destacar en nuestro sistema el desarrollo de la lógica de negocio como servicios. Estos se encuentran en una API independiente, la cual provee todas las funcionalidades del sistema, lo que nos da la posibilidad de en un futuro desarrollar un front-end con otras tecnologías, así como también la posibilidad de desarrollar una aplicación para dispositivos móviles.

Al mismo tiempo el front-end desarrollado íntegramente con JSF, respeta los estándares modernos y se adapta a cualquier tipo de pantallas. El diseño fue pensado con esto en mente, por lo que los menús y las opciones se colocaron de tal manera que al adaptarse a pantallas pequeñas sigan siendo accesibles y fáciles de usar.

Encadenado con lo que se menciona anteriormente, consideramos que el sistema es muy intuitivo para un usuario habitual de este tipo de servicios. Nos hemos inspirado en otras plataformas ya existentes y de uso masivo como Steam o Epicgames.

Finalmente nos gustaría destacar el completo panel estadístico pensado para el administrador del sistema. Cuenta con toda la información que este pueda requerir de una manera clara y sencilla.

Por otro lado, y a modo de autocrítica, podemos señalar algunos aspectos mejorables del sistema.

Si bien la interfaz de usuario es intuitiva y fácil de usar como mencionamos anteriormente, debemos reconocer que tiene un aspecto algo anticuado, sobre todo en los formularios de registro y los perfiles de usuario.

Sabemos que hoy en día se valora mucho la inmediatez de la información, por lo que consideramos que los tiempos de carga en estos sistemas deben ser inmediatos. Nuestro sistema tiene un margen de mejora considerable en este aspecto.

La seguridad en los sistemas informáticos es una obligación en la actualidad, cumplir con los más altos estándares en estos aspectos es algo que todo desarrollador debe garantizar. En este aspecto si bien realizamos ciertos controles de seguridad, sobre todo en la protección de los datos del usuario, como por

ejemplo encriptando sus contraseñas, o en las compras, protegiendo sus datos y garantizando la seguridad en las transacciones, existe también un amplio margen de mejora, que sin duda hay que tener en cuenta en un ámbito profesional.

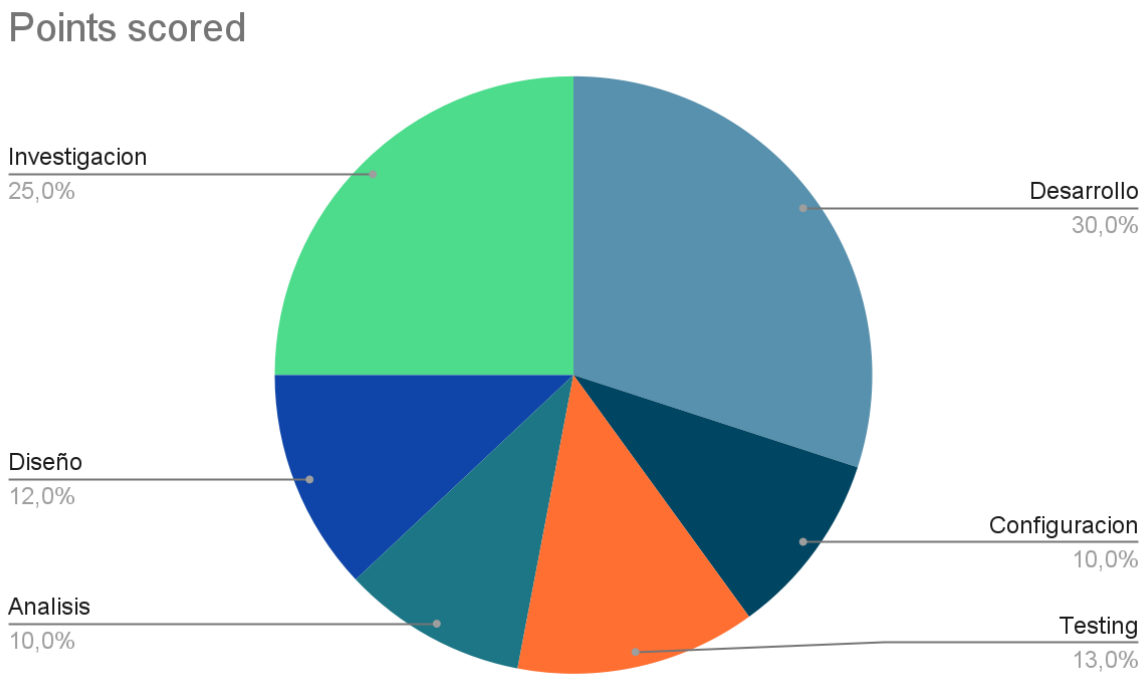
Finalmente, y en contrapartida al completo panel de estadísticas del administrador, el del desarrollador podría estar mejor presentado. En cualquier caso cuenta con la información necesaria.

Desarrollo del proyecto:

El desarrollo del proyecto comenzó de forma lenta debido a la pronunciada curva de aprendizaje de los componentes a utilizar en general. En varias ocasiones se tuvo que rediseñar algunas funcionalidades, y en varias ocasiones también se tuvo que cambiar las tecnologías utilizadas, llevando a demoras en la implementación.

Las tareas que más tiempo llevaron fueron las de investigación e implementación, en parte por lo anteriormente dicho.

Un estimado de las horas distribuidas en las distintas etapas del proyecto es el siguiente:



El total de horas para el desarrollo del proyecto fue aproximadamente de 450 horas distribuidas en 12 semanas.

A pesar de las complicaciones anteriormente mencionadas se logró cumplir con las funcionalidades solicitadas en tiempo y forma.

Conclusiones y trabajo a futuro:

El trabajo planteado nos sirvió para adquirir nuevos conocimientos en el uso de estas tecnologías. Afianzar más los que ya teníamos en programación en general. Reforzar nuestras habilidades de comunicación y trabajo en equipo, que son de gran importancia en este ámbito. Otro aspecto de gran importancia fue la distribución y delegación de tareas.

Por último podemos mencionar algunas funcionalidades que no formaban parte de los requerimientos del sistema, pero que consideramos que pueden enriquecerlo ampliamente.

- Suscripción a la plataforma con cartera de juegos incluida, regalos mensuales, etc
- Otros métodos de pago
- Agregar la posibilidad de hacer y ver streaming de otros jugadores.
- Implementar una app para móviles
- Lista de deseados
- Envío de DM entre los usuarios
- Inicio de sesión utilizando redes sociales
- Notificaciones de eventos a los jugadores.

Referencias:

1. Documentación de Java EE - https://jakarta.ee/specifications/
2. Documentación de RESTEasy - https://resteasy.github.io/docs/
3. Guia y ejemplos de JSF - https://www.tutorialspoint.com/jsf/jsf_link_tag.htm
4. Primefaces Guia - https://primefaces.github.io/primefaces/10_0_0/#/?id=main
5. Stackoverflow errores e investigación - https://stackoverflow.com/