Mondragon Unibertsitatea

Escuela Politécnica Superior

Ingeniería informática

POPBL 6

Daniel Luengo, Egoitz San Martin, Unai Aguinaco,
Unai Mendieta, Unai Orive

MUDLEY

Mondragon Unibertsitatea Escuela Politécnica Superior

TABLA DE ILUSTRACIONES

Resumen

Este documento describe el proceso seguido y los recursos utilizados para la creación de la aplicación web Mudley. Esta servirá para que artistas y organizadores de eventos puedan contactar de manera eficaz y sencilla, además de recomendar, gracias a una inteligencia artificial, grupos basándose en lo buscado por el usuario previamente y usuarios similares. La aplicación estará conectada a varios microservicios. Se han utilizado las buenas prácticas de ITIL; se ha garantizado la confidencialidad, integridad y disponibilidad del sistema, además de monitorizarse.

Abstract

This document describes the process followed and the resources used to create the Mudley web application. This will serve so that artists and event organizers can contact efficiently and easily, in addition to recommending, thanks to artificial intelligence, groups based on what the user previously searched and similar users. The application will be connected to various micro services. Good ITIL practices have been used; the confidentiality, integrity and availability of the system has been guaranteed, as well as being monitored.

Laburpena

Dokumentu honetan Mudley web aplikazioa sortzeko jarraitutako prozesua eta baliabideak deskribatzen dira. Honek artistek eta ekitaldien antolatzaileek modu eraginkor eta errazean harremanetan jartzeko balioko du, gainera, adimen artifizialari esker, erabiltzaileak aurretik bilatutakoaren eta antzeko erabiltzaileen arabera taldeak gomendatuko ditu. Aplikazioa hainbat mikro zerbitzuekin konektatuko da. ITIL praktika onak erabili dira; sistemaren konfidentzialtasuna, osotasuna eta erabilgarritasuna bermatu dira, baita monitorizatu ere.

1.Introducción

En este documento se van a tratar todos los procedimientos empleados para crear una página web para impulsar a los artistas locales con un sistema de recomendación Cold Start. También se han especificado los diferentes procesos y métodos que se han seguido para crear el proyecto.

1.1. Descripción del problema

La contratación de grupos de música en eventos como fiestas, verbenas o conciertos locales sigue un sistema arcaico y consume mucho tiempo. Grupos, asociaciones, fundaciones… utilizan las redes sociales como Instagram, Twitter o Facebook para enviar mensajes a grupos locales con el objetivo de contratarlos para los distintos eventos.

Esto conlleva distintos problemas a la hora de establecer buena comunicación. Hay malentendidos, no se acuerdan correctamente las condiciones de la contratación, la posibilidad de borrar mensajes… Además, una gran cantidad de veces, se contratan grupos que no coinciden con las necesidades del organizador.

Esto demuestra que hay una necesidad de un sistema específico para la contratación en estos casos.

Actualmente se puede encontrar alrededor de 5 millones de bandas y solistas en España. Por otro lado, 438000 conciertos en 2019 en España según la INE. Por lo que se puede observar que hay grupos con problemas para poder organizar algún concierto.

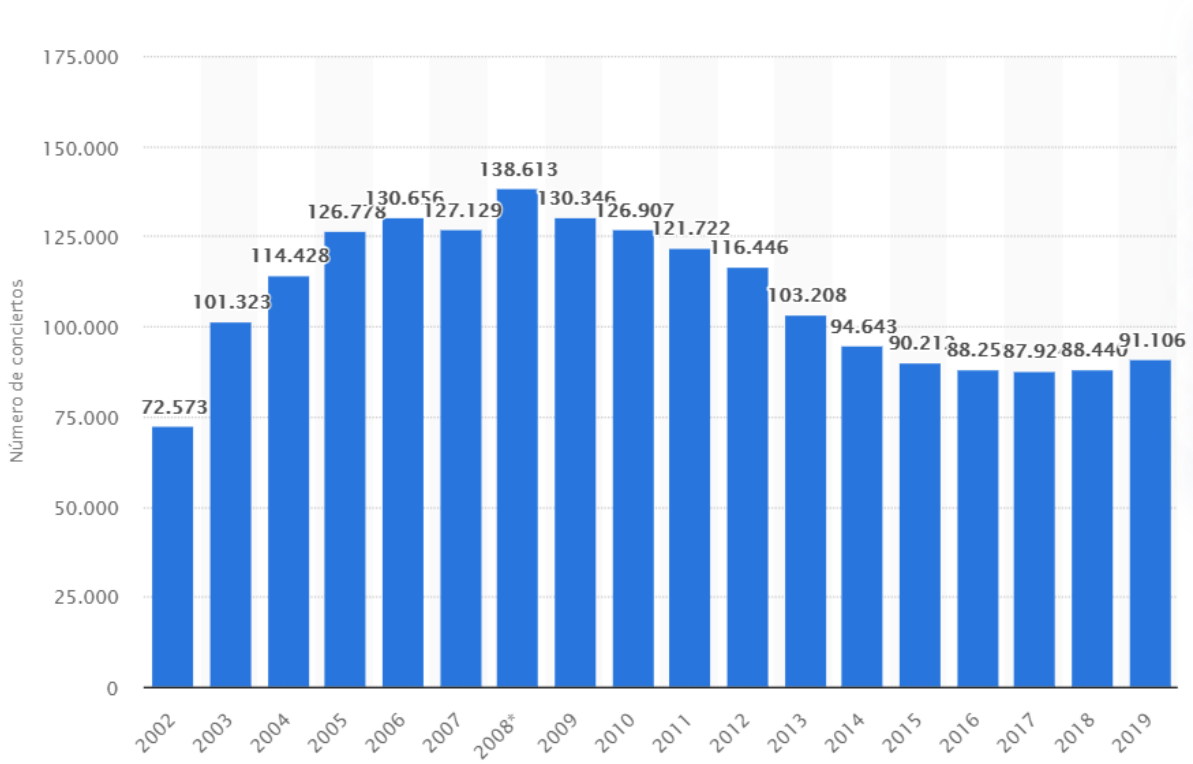


Ilustración 1 Número de conciertos en España en los últimos años

1.2. Clientes

La aplicación dispondrá de tres tipos de clientes diferenciales; por un lado, están aquellas asociaciones, grupos o fundaciones que requieran de la contratación de un grupo o solista para la realización del evento que organizan. Por otro lado, se encuentran estas mismas bandas de música que quieren promover su desarrollo, publicitarse y ofrecer sus servicios para que sean tenidas en cuenta. Por último, están los intermediarios que tienen como objetivo juntar a las dos partes anteriormente mencionadas

Esta aplicación estará disponible a través de un servicio web para el que ambos clientes requerirán un conocimiento básico de la tecnología. Dado que los métodos ahora vigentes requieren el uso de redes sociales los usuarios no encontrarán ningún inconveniente al usar la página web.

Esta, tiene como objetivo ayudar a organizadores o artistas que no disponen de los medios para establecer un canal de comunicación con la otra parte. Por lo tanto, a pesar de que artistas de renombre mundial pueden registrarse; la aplicación tiene como cliente objetivo los artistas novatos o de poca fama y los organizadores de eventos pequeños.

1.2.1. Asociaciones

Este grupo de clientes engloba a cualquier persona, grupo de personas o entidad que necesite un artista para realizar una actividad de ocio relacionada con la música. Aunque la aplicación ayudará a organizar eventos de gran renombre su objetivo es facilitar esta contratación a gente que no dispone de los recursos o conocimientos necesarios para elegir al grupo que mejor se ajuste a sus necesidades (el personal encargado de gestionar las fiestas de un pueblo o un colegio que quiera un concierto para celebrar la graduación de sus alumnos)

1.2.2. Artistas

Aquí se incluirá a todo músico o grupo de música que ofrezcan la posibilidad de realizar conciertos y que quieran dar a conocer tanto su disponibilidad para hacerlos como su propio grupo en sí para que estos sean tomados en cuenta por el otro tipo de clientes mencionado anteriormente cuando quieran organizar un evento relacionado con la música.

1.2.3. Intermediario

Aquí se incluirá a cualquier persona o grupo de personas que es contratado por una de las dos partes y que tiene como objetivo gestionar las actividades de su contratador, ya sea para gestionar la agenda de sus artistas o para gestionar el evento encomendado.

1.3. Competencia

Para contactar con los artistas el método utilizado es ponerse en contacto con su representante o con el grupo directamente. Para ello se requiere un número de contacto del encargado de la gestión de los eventos o establecer un método de comunicación. Hoy en día destacan dos metodologías para lograr este objetivo: el uso de las redes sociales y los catálogos de internet.

Redes sociales: las redes sociales han supuesto un innegable cambio en nuestra sociedad, es por ello que la gran mayoría de artistas han optado por el uso de algunas de estas para publicitarse y ofrecer una forma de comunicación pública accesible para cualquier persona.

La contraparte de este sistema es que el encargado de gestionar las redes sociales no tiene forma de distinguir cuando recibe un mensaje si es con intención de contratar a artistas o es un mensaje de cualquier otro tipo y este problema se agrava con grupos con cierto público ya que el volumen de mensajes que reciben dificulta enormemente la comunicación.

Catálogos: por otro lado, podemos acceder a catálogos de la red como “La Factoría del Show” o “Impronta Music” que ofrecen un sistema de búsqueda para ponerse en contacto con el artista.

A pesar de lo fácil que este método hace la comunicación estas páginas web no muestran ni el presupuesto ni la disponibilidad de los cantantes por lo que puede llegar a ser una pérdida de tiempo si no disponemos del dinero suficiente o el artista no tiene la disponibilidad que necesitamos. Por último, estas páginas no ofrecen ninguna forma de registrar a usuarios para que puedan ser contratados imposibilitando el desarrollo de grupos pequeños.

1.4. Puesta en valor

Como se puede apreciar la forma de contratar solistas o bandas no está bien definida (especialmente para artistas pequeños). La aplicación ayudará a publicitar o dar a conocer a todas las bandas o solistas que lo soliciten y facilitará la tarea a los colectivos o individuales que requieran de los servicios de uno de estas. Para ello se implementarán las siguientes funcionalidades:

Se impulsará el crecimiento de artistas con poco renombre o que estén empezando a través del sistema de recomendaciones para fomentar su desarrollo.

Proporcionará a los organizadores una herramienta para encontrar al grupo o solista que más se ajuste a sus necesidades basándose en ciertos parámetros como el género de música al que se dedican o el rango de presupuestos en el que se encuentran.

Por último, ofrece la posibilidad de establecer una comunicación rápida y eficaz entre ambas partes con el objetivo de mejorar los métodos actuales.

1.5. Acciones y recursos clave

Se utilizarán algoritmos de inteligencia artificial para crear un sistema de recomendación de grupos en base a las necesidades de los organizadores, recomendando a su vez a grupos más pequeños.

Por otra parte, se utilizará un broker de mensajería para la comunicación entre cliente y servidor y la posible escalabilidad del sistema.

Para garantizar la confidencialidad, integridad y disponibilidad de los mensajes, se aplicarán medidas de seguridad como firewalls, IDSs o sistemas de control de acceso.

También se utilizará formatos de datos estándar con el objetivo de que plataformas y herramientas de terceros se puedan integrar fácilmente con la aplicación.

2. Análisis del sistema

En este apartado se detallarán todos los aspectos relacionados con el sistema, es decir los servicios que se van a ofrecer, la arquitectura empresarial, las funcionalidades de estos servicios...

2.1. Objetivos del proyecto

- Reducir en un 50% el tiempo necesario para la contratación de una banda musical.
- Que la contratación de un grupo cumpla con las expectativas del contratante un 90% de las veces (Cumple o no cumple).
- Aumentar la cantidad de conciertos de grupos primerizos en sus primeros dos años un 80%.

2.2. Definición de la empresa y los servicios

Para una correcta definición del producto, es importante establecer cómo va a ser la empresa y los servicios que se van a ofrecer.

2.2.1. Empresa/Negocio (situación actual)

La empresa Mudley es un equipo de 5 personas que trabaja en el desarrollo de páginas web, actualmente, en la página Mudley, una página web para la contratación de grupos locales de música. La empresa se encarga de todo el desarrollo de las páginas, es decir, planificación, diseño, desarrollo e implementación.

2.2.1.1. Organigrama

En la imagen de abajo se puede ver el organigrama de la empresa, en el cual aparecen los diferentes miembros y sus ocupaciones.

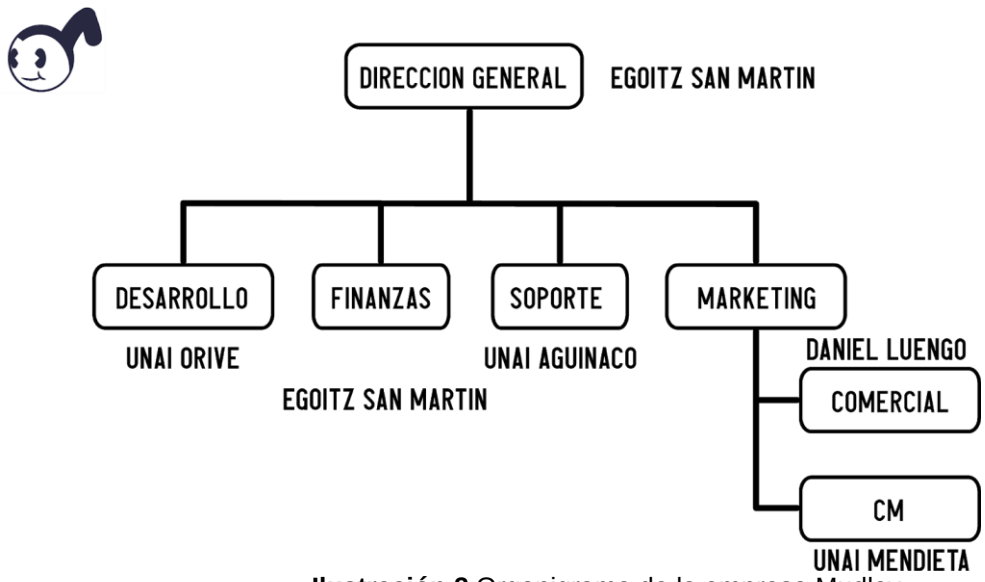


Ilustración 2 Organigrama de la empresa Mudley

- **Dirección general:** El director general será el encargado de buscar alternativas u opciones, evaluándolas según los criterios de la empresa. También será el encargado de tomar las decisiones.

- **Desarrollo:** Este se encargará de crear y mantener todos los elementos relacionados con el producto y sus componentes.

- **Finanzas:** Estará a cargo de llevar todos los pagos al día y de validar las diferentes facturas de la empresa. También verificará si las inversiones que se van a realizar son rentables o no.

- **Soporte:** Su cometido será proporcionar ayuda a los clientes, solucionando sus problemas y dudas.

- **Marketing, Comercial:** Tendrá el objetivo de vender el producto o servicio a empresas o individuos, desarrollando una estrategia de marketing entre el cliente y el producto.

- **Marketing, Community Manager:** Será el profesional del marketing digital, siendo responsable de la gestión y desarrollo de la comunidad online de la empresa.

2.2.1.2. Arquitectura empresarial

La arquitectura empresarial de Mudley está compuesta por varias secciones con el objetivo de generar ganancias con la aplicación web. Aquí abajo se muestra la arquitectura de una manera más detallada. (Ver Anexo A -Arquitectura empresarial)

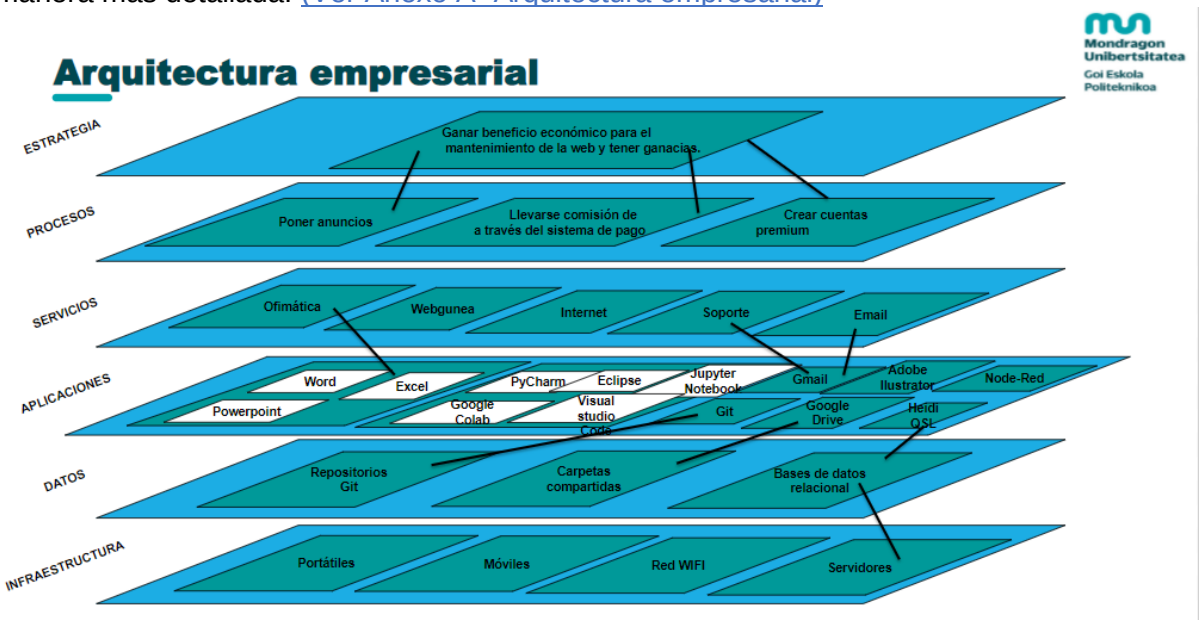


Ilustración 3 Arquitectura empresarial de la empresa MUdley

2.2.2. Definición del producto

El producto será una página web en la que se podrán registrar dos tipos de usuarios. Por un lado, se encontrarán a los artistas, estos introducirán los datos de la banda y las fechas que tienen disponibles para hacer conciertos. Por otro lado, estarán los organizadores, que serán los encargados de ponerse en contacto con los artistas para llegar a un acuerdo y organizar el concierto.

En resumen, será una aplicación que facilite el proceso de crear eventos o conciertos para los dos usuarios. Creando una comunicación más eficaz.

- **Ventajas:**

El producto ofrecerá a sus clientes una plataforma para centralizar la contratación de grupos locales ayudando así tanto a artistas pequeños como a las organizaciones que quieran gestionar sus eventos. La web filtrará a los artistas en base a sus preferencias: presupuesto, localización, género musical… facilitando así encontrar al grupo idóneo para cada usuario.

- **Desventajas:**

Al utilizar la aplicación, solo se podrá contactar con los artistas que se hayan registrado previamente. Por lo que la dimensión que podrá alcanzar la web dependerá del número de usuarios participen.

- **Elementos distintivos:**

El servicio dispondrá de un sistema de recomendación creado con inteligencia artificial. Se les recomendará a los organizadores los artistas que más se ajusten a sus necesidades.

2.3. Arquitectura del sistema

Teniendo en cuenta las necesidades del sistema, el grupo ha diseñado una arquitectura con una web a la que accederá directamente el usuario, un servicio de IA para el sistema de recomendación de la página web, un servidor de SonarQube para hacer un análisis estático del código, un servidor de Jenkins para automatizar los procesos del sistema, un servidor de Zabbix para monitorizar los distintos servidores, una base de datos para guardar y gestionar toda la información necesaria y, por último, el servicio de Proactivanet para la gestión de servicios TI.

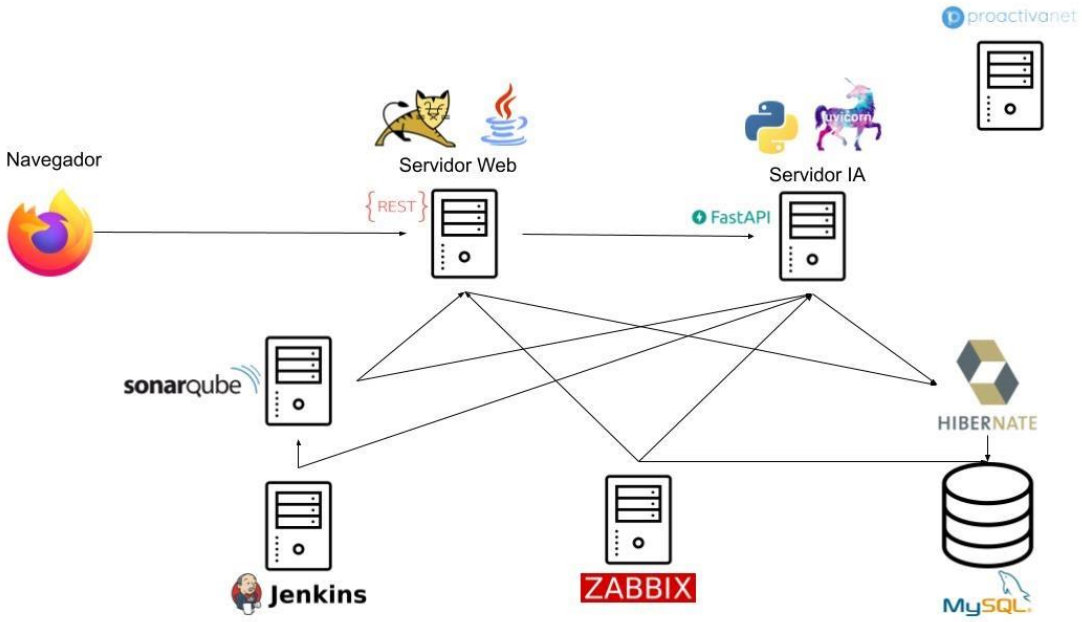


Ilustración 4 Arquitectura del sistema de la empresa Mudley

2.4. Requisitos funcionales y no funcionales

Para tener un proyecto bien definido es necesario tener definido los dos tipos de requisitos. Un requisito funcional define una función del sistema de software o sus componentes. Y por otro lado los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

2.4.1. Requisitos funcionales

- **Permitir la autenticación de los usuarios:**
  La aplicación debe permitir autenticarse a los diferentes usuarios mediante su nombre de usuario y contraseña.

- **Registrarse**:
  Los nuevos usuarios podrán crearse una nueva cuenta antes de iniciar sesión en la aplicación web. No se podrá registrar con un nombre de usuario que haya sido previamente registrado.

- **Ver chats:**
  Los usuarios, tanto los organizadores como los artistas, tendrán la posibilidad de ver todos los chats disponibles que tengan.

- **Elegir chats:**
  Todos los usuarios, tanto los organizadores como los artistas, tendrán la posibilidad de elegir entre todos los chats disponibles que tengan el que ellos deseen. Solo los organizadores tendrán la posibilidad de seleccionar un chat con artistas con los que no hayan hablado previamente.

- **Chatear:**
  En el chat creado entre los artistas y los organizadores, ambos usuarios tendrán la posibilidad de intercambiar mensajes estableciendo un método de comunicación a través de la aplicación.

- **Ver calendario:**
  Todos los usuarios tendrán la posibilidad de ver calendarios, pero solo los artistas poseerán dichos calendarios.

- **Poner fecha:**
  Los artistas tendrán la posibilidad de elegir qué días están disponibles para que los organizadores puedan contratarlos añadiéndoles al calendario.

- **Quitar fecha:**
  Los artistas tendrán la posibilidad de quitar los días que hayan seleccionado previamente como fechas de posible contratación.

- **Ver artistas:**
  La web proporcionará una lista para visualizar los artistas de una zona que se elegirá mediante un mapa.

- **Buscar artistas:**
  Los usuarios podrán escribir en un buscador el nombre de cualquier artista que se encuentre registrado en la página.

- **Elegir artista:**
  Los usuarios podrán elegir entre una lista de artistas un artista en concreto clicando sobre el artista que deseen.

- **Abrir enlace externo:**
  Los artistas tendrán un botón en su página que al clicar abrirá una página externa con la música de dicho artista.

- **Ver información:**
  Los usuarios podrán acceder a los datos que hayan introducido previamente a la hora de registrarse.

- **Modificar ajustes:**
  Los usuarios tendrán la posibilidad de cambiar los datos que hayan introducido a la hora de registrarse.

2.4.2. Requisitos no funcionales

En este apartado se detallarán los requisitos no funcionales que ofrecerá la aplicación.

2.4.2.1. Requisitos del producto

- **Usabilidad**

La aplicación no requiere de una gran capacidad de procesamiento, dado que será desplegada en un servidor web, lo que permitirá que se ejecute en prácticamente cualquier ordenador o dispositivo con acceso a navegadores de internet, independientemente de su potencia.
La interfaz del usuario debe estar orientada a todo tipo de personas, siendo una interfaz intuitiva y fácil de usar. Para ello, se optará por el uso de una estética simple y moderna. Después de que un usuario inicie sesión este avanzará a una pantalla en la que aparecerá un buscador donde tendrá que establecer los filtros, también podrá acceder a los ajustes y a la lista de chats disponibles. El usuario podrá acceder a los datos específicos de un artista clicando en su perfil una vez hecha la búsqueda, pero solo podrá iniciar un chat con él si está registrado como organizador. Dentro de la pantalla que visualiza los datos específicos se podrá acceder al calendario del artista. Podrá volver en todo momento a la pantalla de la lista mediante un botón. Los artistas podrán cambiar sus fechas disponibles, cambiando el calendario en ajustes. La manera de interactuar con el programa será mediante teclado y ratón.

- **Eficiencia**

El inicio de sesión en la web debe ser rápido para mejorar la eficiencia del sistema, por lo que no se debe tardar más de 3 segundos en verificar los datos introducidos, por otro lado, el registro en la aplicación también tendrá que ser rápido.
Además, el sistema permitirá estar conectados un gran número de usuarios sin influir en el rendimiento. Ya que las peticiones serán atendidas mediante una creación de nuevos hilos sin afectar al hilo principal del programa.
Para los organizadores, tendrán a su disposición un sistema de recomendación en el que podrán encontrar los artistas que más se ajuste a sus necesidades de una manera más eficaz que buscándolo manualmente.

- **Seguridad**

Para gestionar la accesibilidad de los datos, habrá dos tipos de usuarios: artistas y organizadores. Los primeros tendrán la posibilidad de modificar sus calendarios de disponibilidad, y los organizadores tendrán la posibilidad de crear chats con artistas si no han tenido una conversación anteriormente.

Para la confidencialidad la web seguirá una política de privacidad donde se definen todas las medidas que aplica Mudley para garantizar la seguridad y el uso lícito de los datos de los usuarios o clientes que recoge en el contexto de la relación comercial.

Los servidores adoptarán medidas de seguridad como firewalls, sistema de detección de intrusos o sistemas de control de acceso para garantizar la protección de los datos confidenciales, como los datos personales de los usuarios o los datos confidenciales que posea la empresa.

Todos los usuarios deberán cumplir los requisitos mínimos de seguridad que se establecen en la aplicación. La contraseña deberá tener un mínimo de 10 caracteres, incluyendo mayúsculas, minúsculas, números y carácter especial. Cada vez se inicie la aplicación deberá iniciar sesión, independientemente de haberlo hecho anteriormente.

- **Disponibilidad:**

Tomando en cuenta las necesidades y requerimientos de los diferentes organizadores y artistas, el sistema deberá estar disponible las 24 horas. Se crearán copias de seguridad de los servidores cada cierto tiempo para poder restablecer el funcionamiento de la web lo antes posible.

- **Portabilidad:**

Al ser una aplicación web no será necesario la instalación de ningún programa externo. Simplemente con tener conexión a internet y tener a disposición un navegador web ya se podrá acceder a la aplicación. Por lo que no será dependiente del dispositivo con el que se acceda a la web, ya que se podrá acceder con cualquiera que tenga un navegador.

2.4.2.2. Requisitos externos

- **Interoperabilidad:**

Como ya se ha comentado anteriormente, los datos de la aplicación serán almacenados en varios servidores. Para enviar datos mediante diferentes intermediarios, se enviarán en un formato estándar, usando los formatos de texto XML y JSON, mejorando así la interoperabilidad del sistema. De esta manera, aplicaciones y herramientas de terceros podrán comunicarse con nuestra web de una manera más sencilla y eficaz.

- **Éticos:**

La aplicación estará construida, en primera instancia, en un idioma: *español*, ya que es un producto fabricado en el estado español. De todos modos, en un futuro se podrá añadir nuevos lenguajes, con especial importancia el inglés, por ser el idioma con un uso más extendido, y el euskera, dado que la aplicación ha sido desarrollada en el país vasco, suponiendo este un gran nicho de mercado.

- **Legislativos:**

Los permisos legislativos recopilan los requisitos que debe cumplir la aplicación de cara a las leyes vigentes en el país, en este caso el estado español.

En el caso de esta aplicación web, la ley que se deberá tener en cuenta será la Ley Orgánica 3/2018, la ley de Protección de Datos Personales y garantía de los derechos digitales. La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.

2.5. Casos de uso

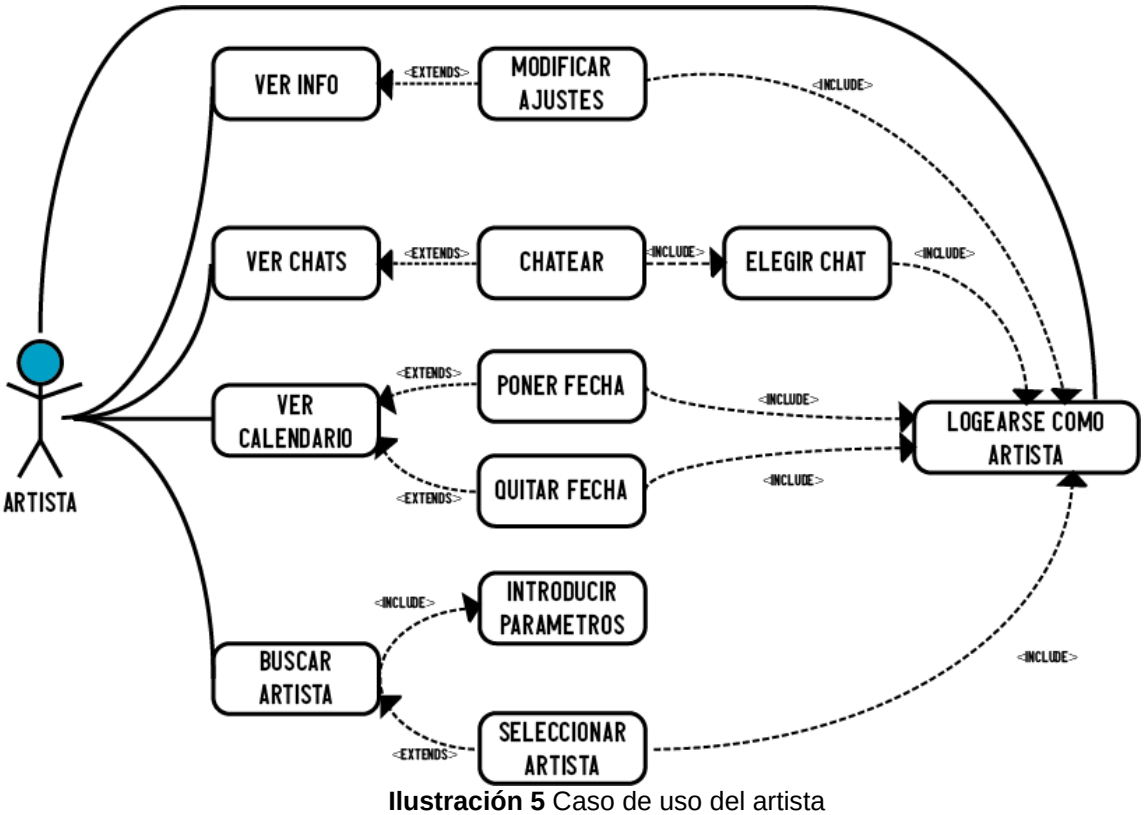


Ilustración 5 Caso de uso del artista

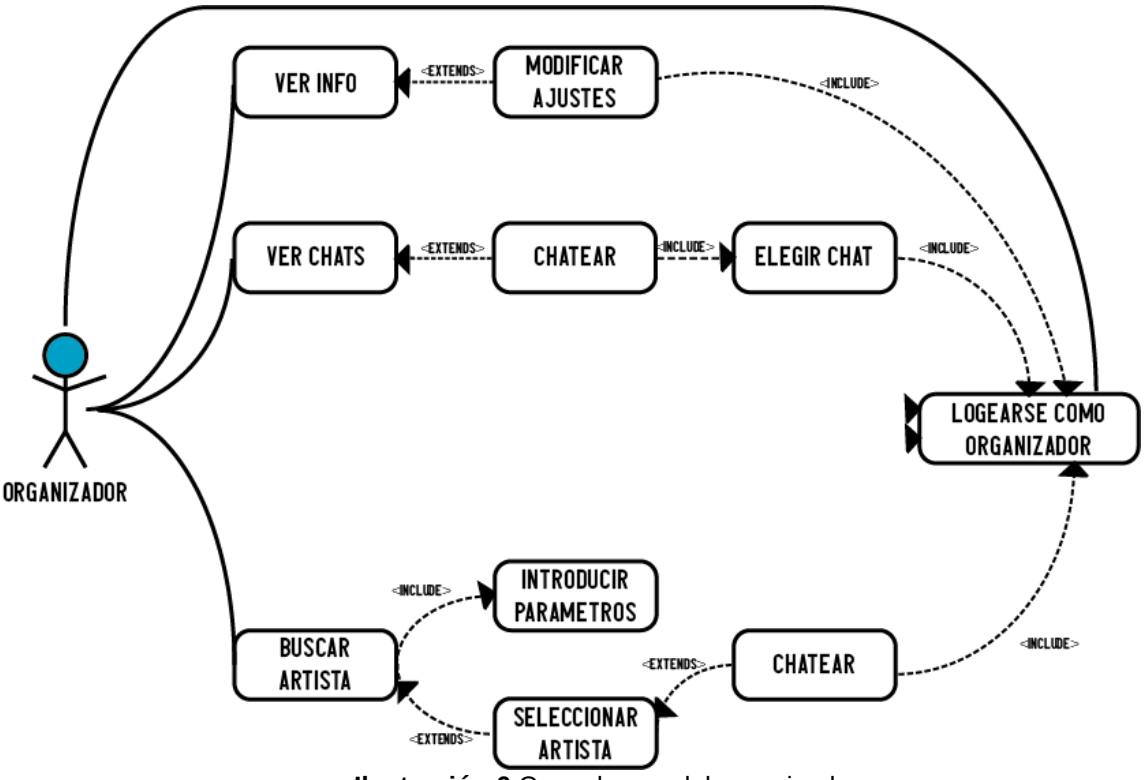


Ilustración 6 Caso de uso del organizador

2.6. Diagrama entidad-relación

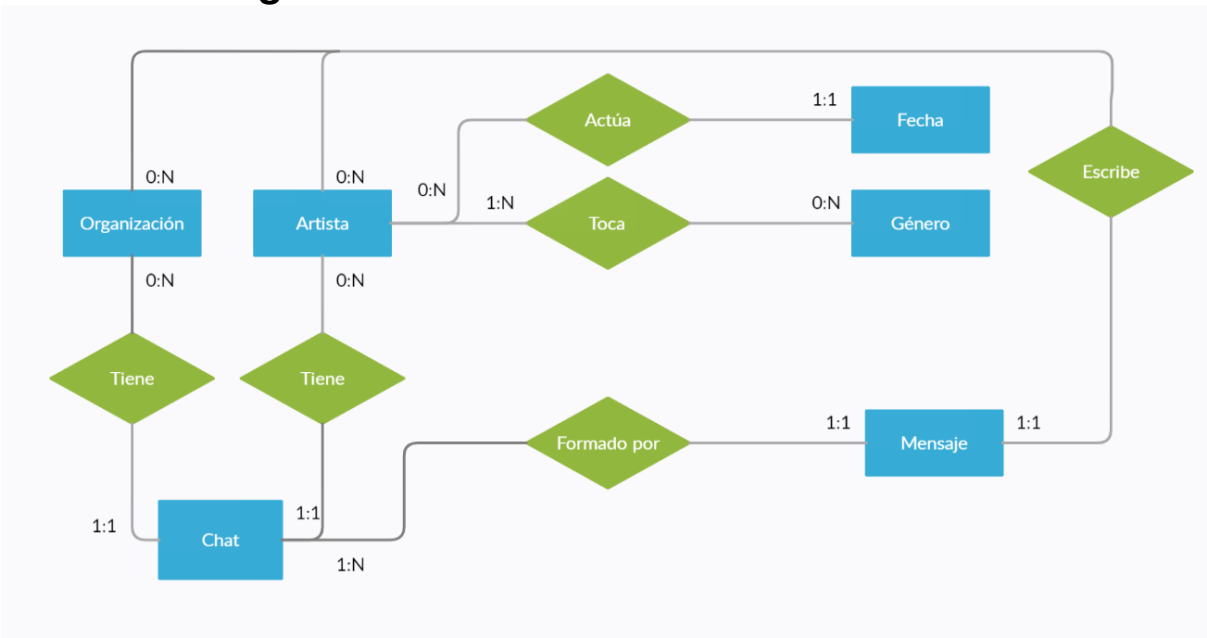


Ilustración 7 Diagrama entidad-relación de la base de datos de MUdley

2.7. Diagrama de clases

A continuación, se muestran los diagramas de clases en los que se basará el programa desarrollado en Java.

Diagrama de clases del artista:

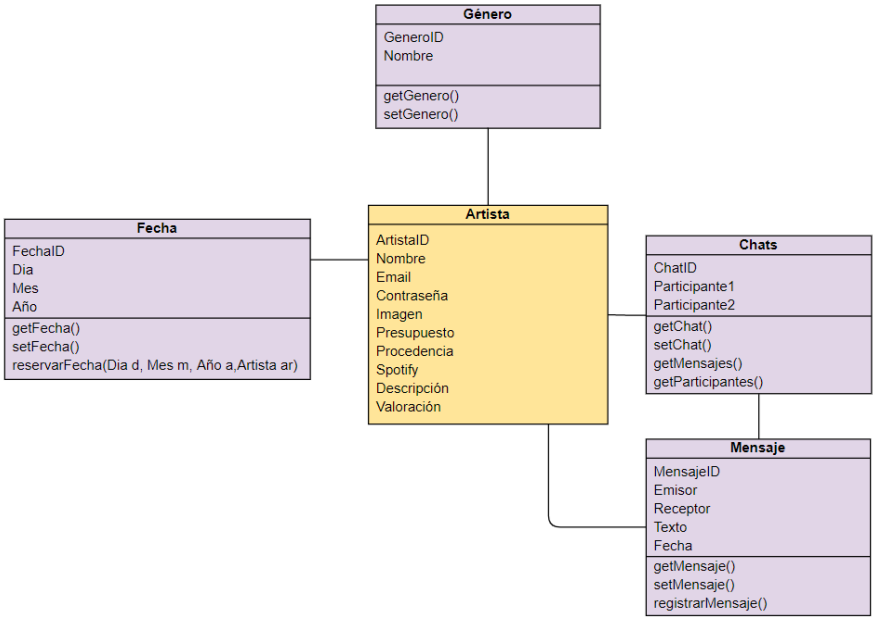


Ilustración 8 Diagrama de clases del artista

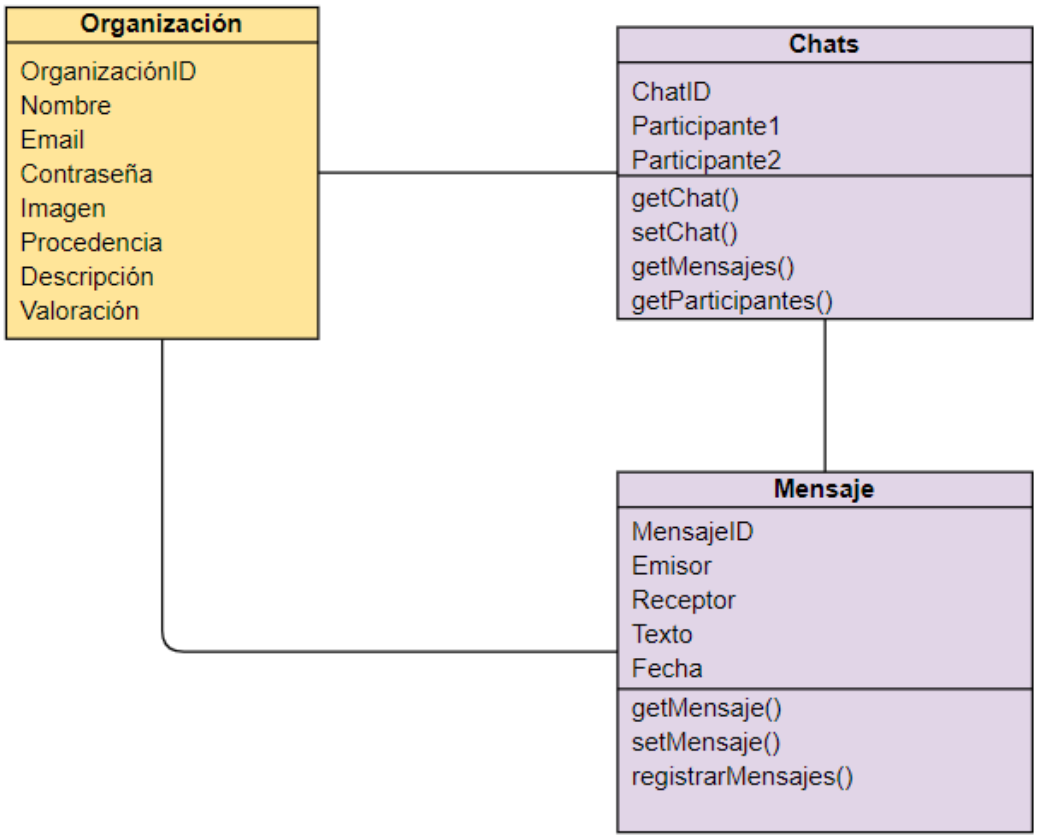


Ilustración 9 Diagrama de clases de la organización

3. Organización del proyecto

A continuación, se detallarán los procedimientos que se van a seguir para una correcta gestión del proyecto planteado.

3.1. Planificación inicial del proyecto

Para una correcta gestión del proyecto se ha decidido emplear Kanban, un sistema de información que controla de modo armónico la fabricación de los productos necesarios en la cantidad y tiempo necesarios en cada uno de los procesos que tienen lugar.

3.1.1. Funciones y responsabilidades

El equipo está compuesto por 5 integrantes y siguiendo la metodología Agile asignamos el “product owner” y el “flow master” a Egoitz San Martin y Unai Aguinaco respectivamente.
El propietario del producto será responsable del éxito del producto y es quien se preocupa por las necesidades del usuario y tiene una visión de dónde llevar el producto.

El maestro de flujo tiene dos funciones principales: asegurarse de que los elementos de trabajo fluyan y facilitar el cambio y las actividades de mejora continua. El maestro de flujo debe verificar el tablero Kanban y asegurarse de que ningún elemento de trabajo haya sido bloqueado durante demasiado tiempo, verificar si hay un problema o riesgo de algún tipo y ofrecer ayuda y asegurarse de que los miembros del equipo sigan las políticas.

3.1.2. Diagrama de flujo acumulativo

En Kanban se utiliza un diagrama llamado Diagrama de flujo acumulativo para representar el progreso realizado en las diferentes fases del proyecto contando la cantidad de tareas completadas. Esto es útil para ayudar a las partes interesadas y a los miembros del equipo a conocer el ritmo de trabajo y si alguna de las fases se atasca o se detiene en un cuello de botella.

El eje y representa las tareas o Kanbans realizados y el eje x la línea de tiempo que se ha dividido en semanas. Las líneas de diferentes colores representan las fases del proyecto y el número de tareas en cada fase. De modo que el trabajo en progreso se puede medir con este gráfico como la suma de las tareas de cada fase y también el tiempo del ciclo, también conocido como tiempo de entrega, que es el tiempo promedio que tarda una sola tarea en completarse.

El Diagrama de Flujo Acumulado representa las 4 fases que tiene el proyecto: análisis, implementación, pruebas y el acumulado hecho. Cada una de estas fases tiene diferentes límites WIP que representan el número máximo de tareas que pueden estar en progreso en esa fase. Si alguna de las fases alcanza el límite, significa que tiene un problema de cuello de botella y significa que esa fase está limitando la capacidad del progreso.

3.1.3. Implementación de la gestión

Las principales funcionalidades del proyecto se han establecido como PBIs o historias de usuario en el backlog del producto detallando los criterios de aceptación, su prioridad y la fecha estimada de entrega, así como la épica a la que pertenece cada una de ellas. La prioridad está representada por un número de 0 (prioridad más baja) a 100 (más alta).

Las tarjetas Kanban se han creado a partir de las PBI de la cartera de productos, dividiéndolas en diferentes tareas más pequeñas llamadas elementos de trabajo, que son necesarias para lograr las funcionalidades del backlog del producto.

Las reuniones o los bucles de retroalimentación que se han realizado son las reuniones diarias, reuniones semanales con el tutor del grupo que pueden ser consideradas como reunión de revisión del scrum y reuniones retrospectivas.

La reunión diaria es una reunión breve para poner en común con los miembros del grupo el progreso de cada día. El objetivo de la reunión de revisión con el tutor es presentar el incremento o proceso realizado durante la semana y recopilar la retroalimentación necesaria para refinar el backlog y definir los próximos pasos. Por otro lado, la reunión retrospectiva se utiliza para mejorar la comunicación del equipo y la forma de trabajo que se lleva a cabo según las necesidades del equipo.

En cuanto a los roles del proyecto, se ha asignado un maestro de flujo para controlar el flujo de entrega del proyecto y no perder de vista los avances realizados.

3.1.4. Políticas explícitas

Las políticas explícitas son importantes para tener reglas claras y sencillas entre los compañeros de equipo y también para mejorar continuamente como es imprescindible en Kanban. Estas políticas siguen el principio de "siempre aplicadas y fácilmente modificables". Las políticas definidas para estos proyectos son las siguientes:

• La definición del límite de WIP determina el número máximo de tareas o elementos de trabajo que pueden estar en progreso en una fase específica.

- Fase de diseño: 5 elementos de trabajo como máximo
- Fase de desarrollo: 8 elementos de trabajo como máximo
- Fase de test: 3 elementos de trabajo como máximo

• La definición de hecho es un conjunto acordado de elementos que deben completarse antes de que una historia de usuario o un elemento de trabajo se pueda considerar completo. Es una puerta oficial que separa las tareas de "en progreso" a "terminadas".

- Análisis: una tarea en la fase de análisis se considera realizada cuando se logran los conocimientos necesarios para implementar la tarea.
- Implementación: una tarea en la fase de implementación se considera realizada cuando funciona como se esperaba, pero necesita ser probada o testeada.
- Testing: una tarea en testing se considera realizada cuando funciona y funciona correctamente y se cumplen los criterios de aceptación de la cartera de productos.

• Codificación de prioridades: como Kanban se basa en visualizar gráficamente el progreso, los elementos de trabajo se diferencian por colores que significan el nivel de priorización de cada tarea.

- Urgente: verde
- Fecha fija: magenta
- Estándar: amarillo
- Documentación: rosa
- Seguridad del sistema: Azul

3.2. Descripción del entorno de desarrollo

En este apartado se explicará todas las medidas adoptadas para el entorno de desarrollo, tanto la estrategia de control de ramas como la documentación que se utilizará.

3.2.1. Documentación

Es importante tener la documentación correcta del código que permitirá a otros desarrolladores entender nuestro código. Una documentación incorrecta podría resultar en comentarios innecesarios y confusión en la comprensión del código.

3.2.2. Documentación de Javadoc

Para documentar el código, la herramienta principal utilizada será el complemento Javadoc Tools. Esto facilita la documentación al generar los parámetros y excepciones utilizados y el valor de retorno en cada función de forma automática. Luego, se completarán los comentarios con una breve descripción de cada función en la primera línea y definiremos el tipo de cada parámetro para aclarar su uso en la función. Si la función tiene algún tipo de excepción también se comentará. Y si hay un valor de retorno, se especificará el tipo y una breve explicación.

3.2.3. Diagramas UML

Para generar automáticamente los diagramas UML, se recomienda utilizar el doclet UML de YWorks junto con Javadoc. La configuración que usaremos es la siguiente:

- Como el "comando Javadoc" es necesario para insertar la ruta javadoc.exe
- Seleccione el proyecto del que se van a generar los diagramas Javadoc y UML
- Seleccione "Usar doclet personalizado"
- Nombre del doclet: ydoc.doclets.YStandard
- Ruta de clase de Doclet: C: \ yworks-uml-doclet-3.1-jdk1.8 \ lib \ ydoc.jar (ruta a la biblioteca ydoc)

3.2.4. Control de versiones

Para realizar un correcto control de versiones se implementará Git usando la siguiente estrategia de ramas:

- Master: En esta rama estará únicamente la versión de la aplicación que esté disponible para los clientes.
- Hotfix: En esta rama se traerán las versiones del producto que presenten errores que no requieran de una resolución compleja. Aquí se tratarán esos errores para después juntarlos en la rama principal.
- Release: Aquí estarán las versiones del código que estén preparadas para entrar en master pero que todavía no se hayan testeado adecuadamente.
- Development: Esta será la principal rama en la que se desarrollará la aplicación
- Feature: Cada vez se quiera añadir una nueva funcionalidad se recoge el código de la rama de development y se desarrolla en esta rama.

3.2.5. “Merging” de las ramas

En este proyecto se utilizará la estrategia recursiva de fusiones de ramas. Esta es la estrategia de “merging” predeterminada al extraer o fusionar una rama, y esta es la forma en que vamos a fusionar cada rama:

- Feature: Esta rama se enviará a development cuando la funcionalidad nueva esté implementada
- Release: recibe las versiones de development y en esta se arreglan fallos o bugs menores antes de ser enviadas a master
- Hotfix: se fusiona con la rama de master para recibir las versiones y se mandan a development si el fallo es significante o se solucionan en la misma rama y se envían de nuevo a el master.

- Development: cuando los cambios deben fusionarse en la rama de desarrollo, es una buena práctica crear una nueva rama de características para trabajar de forma independiente.

3.2.6. Estrategia de versiones

Como el proyecto está en constante cambio por parte de todos los compañeros del grupo, hacer un seguimiento de la versión del código en la que estamos trabajando es fundamental para el correcto desarrollo del producto.

La estrategia de versión que seguimos se basa en 3 números que representan el número de versión maestra, el número de versión de la función y el número de versión de revisión. Se seguirá un control de versiones semántico, que es un esquema construido alrededor de X.Y.Z:

- Incrementa Z cuando algo está arreglado.
- Incrementa Y cuando se agrega una característica.
- Incrementar X cuando se rompe la compatibilidad con versiones anteriores o se agregan características principales.

3.2.7. Perfiles de calidad

Para este proyecto se han creado varios perfiles de calidad, uno para cada uno de los lenguajes utilizados (Java, CSS, HTML y JavaScript). Estos perfiles heredarán las reglas por defecto del perfil "Sonar way" pero añadiendo y eliminando diferentes reglas de calidad, con el fin de adaptarlo a nuestro proyecto. Algunas de las reglas que se han agregado son las siguientes:

Para Java se agregaron:

- Los bloques inútiles "if (true) {…}" y "if (false) {…}" deben eliminarse
- Las pruebas unitarias fallidas deben solucionarse
- Se deben combinar varios bucles sobre el mismo conjunto
- Las construcciones "if ... else if" deben terminar con cláusulas "else"

Para JavaScript se agregaron:

- Los bloques inútiles "si (verdadero) {...}" y "si (falso) {...}" deben eliminarse
- Los objetos no deben crearse para soltarse inmediatamente sin ser utilizados
- No se debe incluir contenido que no sea de confianza
- Los nombres de funciones y métodos deben cumplir con una convención de nomenclatura
- Las declaraciones "switch" deben tener cláusulas "predeterminadas"

Para CSS se agregaron:

- Las pruebas unitarias fallidas deben solucionarse
- Los archivos de origen no deben tener bloques duplicados

Para HTML se agregaron:

- Las etiquetas deben definirse en el paquete de recursos
- Todas las etiquetas HTML deben estar cerradas
- Los archivos de origen no deben tener bloques duplicados
- Rastrear la falta de un elemento requerido con el "id" requerido

3.2.8. Quality gate

Estas son las condiciones definidas para todos los perfiles de calidad:

- Los métodos no deben tener una complejidad cognitiva superior a 10
- La cobertura de la afección debe ser superior al 80%
- Las líneas duplicadas deben ser inferiores al 3%.
- La calificación de mantenibilidad debe ser A
- Los puntos de acceso de seguridad revisados deben ser del 100%
- La clasificación de seguridad debe ser A
- Las líneas descubiertas no deben exceder la suma de 100

3.2.9. Construcciones automáticas

Para hacer las compilaciones para la bifurcación, crearemos un archivo de compilación de automatización para crear las 6 bifurcaciones en GitHub.

Siempre que se comete un cambio en alguna de las ramas, la compilación automática notificará al desarrollador y realizará los cambios correspondientes en las ramas y fusionará las diferentes versiones para evitar errores.

3.2.10. Integración continua

Usaremos Jenkins para resolver el problema de la integración continua. Esta técnica es útil para este tipo de proyectos porque nosotros, como desarrolladores, integramos el código en un repositorio compartido muchas veces y es necesario tener una versión común que no difiera demasiado ya que hacemos cambios individuales con mucha frecuencia. Esto también es útil para verificar las compilaciones y las pruebas automatizadas. Además, esto ayudará a detectar errores de código rápidamente y localizarlos más fácilmente.

4. Desarrollo

La aplicación Mudley es una aplicación web que permite a los artistas publicitarse y ofrecerse disponibles para conciertos y a los organizadores seleccionar a los grupos que más se ajustan a sus necesidades.

4.1. Funcionamiento

Cuando los usuarios ingresan a la aplicación solo tienen permiso para acceder a la visualización principal y a los menús de registro e inicio de sesión. Si es la primera vez que acceden, tienen que crearse una cuenta y especificar qué tipo de usuario van a ser; una vez creada la cuenta podrán acceder a la aplicación.

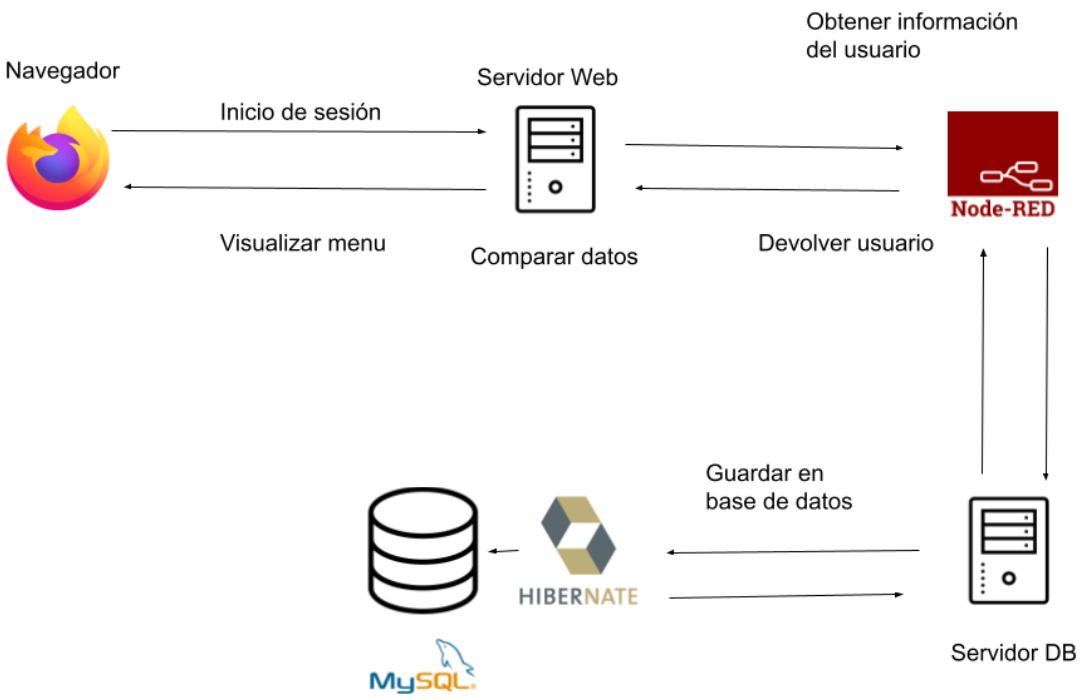


Ilustración 10 Comunicación entre la página web y la BD

Una vez dentro, gracias a la implementación de Cold Start en la inteligencia artificial recibirá recomendaciones aun sin haber hecho consultas anteriores. Para visualizar estas recomendaciones el servidor web enviará peticiones a través de Node-RED que serán validadas mediante esquemas de JSON y se realizará una petición a la API de inteligencia artificial que devolverá unas listas con artistas. Esta lista se devolverá a Node-RED para ser enviada al servidor y que pueda ser vista por el usuario.

Si el usuario (organizador) realiza una búsqueda en la barra superior se tomará una lista de recomendaciones (siguiendo el procedimiento mencionado anteriormente) y se organizarán los resultados en base a los parámetros introducidos en la búsqueda y así visualizar primeros los que cumplan todos los requisitos y estén recomendados para este artista. En caso de que haya más de un grupo que cumpla los criterios se visualizarán primero los que la IA considere mejor recomendación para ese usuario.

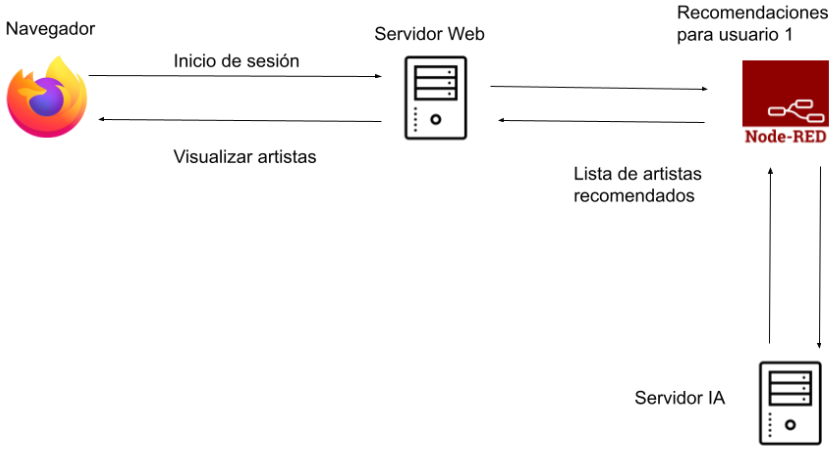


Ilustración 11 Comunicación entre la página web y la IA

Cuando el usuario hace clic en uno de estos artistas podrá visualizar su perfil, ver su disponibilidad y establecer un chat con ese grupo. Cuando decide iniciar un chat el servidor envía varias peticiones a Node-RED que le pedirá a la base de datos todos los chats y mensajes de ese usuario para poder visualizarlos y así, si tiene un chat ya creado cargar todos los mensajes. Mientras chatean, los mensajes serán guardados en la base de datos periódicamente para que no se pierda la información y no pueda haber reclamaciones como hay con el sistema de contratación actual. Los organizadores tienen la opción de proponer un concierto para un día en específico. Esta proposición aparecerá a los dos integrantes en el chat y el artista puede aceptar o rechazar esta proposición en cualquier momento o seguir chateando para clarificar los términos.

Los artistas por otro lado tienen la posibilidad de establecer qué días tienen ocupados, ya sea porque han sido contratados por otro método que no sea la aplicación o porque alguno de los integrantes no está disponible ese día. Esta información se almacenará en la base de datos siguiendo el mismo procedimiento que con los chats.

Esta opción se sitúa dentro de sus propios ajustes en los que pueden cambiar su foto de perfil o su contraseña además de visualizar todos sus datos personales.

4.2. Servidor de Inteligencia Artificial

En el siguiente apartado se definirá todo lo implementado en el servidor de inteligencia artificial.

4.2.1. Diseño del agente

Para realizar un sistema de recomendación se ha decidido hacer un sistema “cold start” o arranque en frío que se utiliza para solventar el problema principal de los sistemas de recomendación que son incapaces de recomendar a usuarios que acaban de entrar en la aplicación (Para más información acerca de todo lo relacionado con la IA ver Anexo B – Documentación IA).

Este sistema de recomendación utiliza dos metodologías mezcladas entre sí para lograr una solución: la recomendación basada en contenido y la recomendación basada en usuarios.

4.2.2. Filtrado basado en contenido

El filtrado basado en contenido utiliza la técnica para analizar un conjunto de documentos y descripciones de elementos previamente calificados por un usuario, y luego construir un perfil o modelo de los intereses del usuario basado en las características de esos elementos calificados. Usando el perfil, el sistema de recomendación puede filtrar las sugerencias que encajaría con el usuario.
El problema con el sistema de recomendación basado en contenido es que, si el contenido no contiene suficiente información para discriminar los elementos con precisión, la recomendación no será precisamente al final.

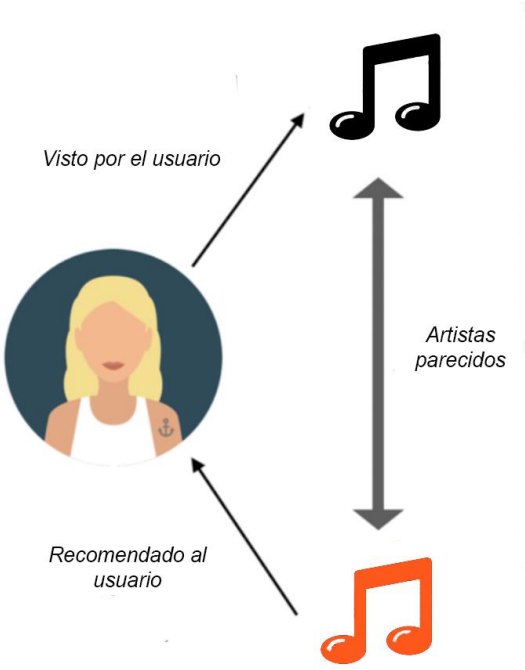


Ilustración 12 Recomendación basada en contenido

4.2.3. Sistema de recomendaciones basado en la colaboración

La idea clave detrás del sistema de recomendación basado en colaboración es que usuarios similares comparten el mismo interés y que a un usuario le gustan los elementos similares. Pero estos no pueden abordar el problema del usuario en frío.

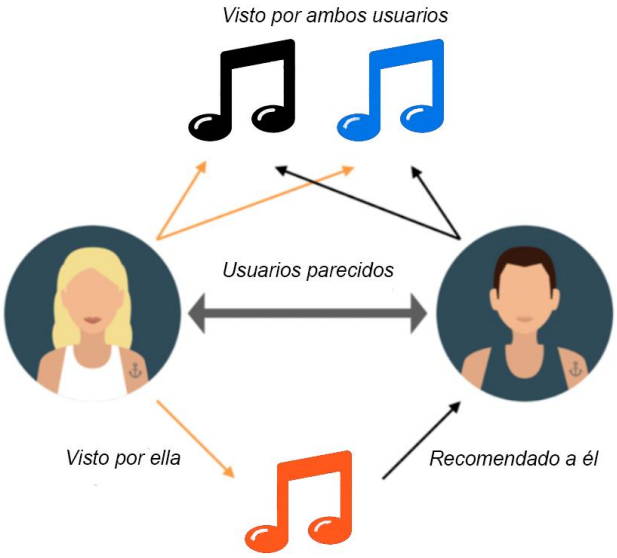


Ilustración 13 Recomendación basada en la colaboración

4.2.4. Obtención de la información

Para que el sistema de recomendación sea posible se ha utilizado una base de datos que recoge 17.000 distintos artistas con su respectiva información y las veces que 2100 usuarios han escuchado a esos artistas (Alcanzando hasta 90.000 diferentes entradas en el fichero que relaciona ambos). Esta información será la que el sistema de recomendación tendrá en cuenta para generar el modelo.

Este sistema mixto necesita dos tipos de matrices para generar el modelo mixto de recomendación; una de ellas es la matriz COO y otra la CSR que se detallaran a continuación.

4.2.4.1. Matriz dispersa

Una matriz dispersa es una matriz que tiene un valor de 0 para la mayoría de los elementos. Si la relación de Number Nsitu Zero (NNZ) elementos en el tamaño es menor que 0,5, la matriz es escasa. Si bien esta es la definición matemática, usaré el término disperso para matrices con solo elementos NNZ y denso para matrices con todos los elementos.

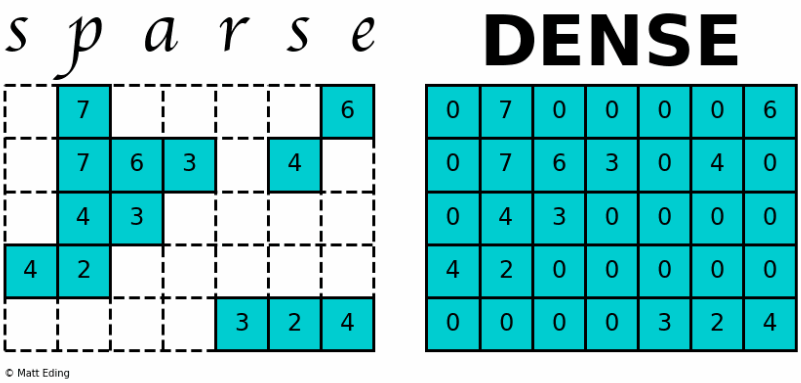


Ilustración 14 Matriz dispersa

Almacenar información sobre todos los elementos 0 es ineficiente, por lo que asumiremos que los elementos no especificados son 0. Con este esquema, las matrices dispersas pueden realizar operaciones más rápidas y usar menos memoria que su representación de matriz densa correspondiente, lo cual es especialmente importante cuando se trabaja con datos grandes. conjuntos en ciencia de datos.

4.2.4.2. Matriz de coordenadas

La matriz se usa para establecer relaciones entre los usuarios y los artistas y entre los artistas y sus características. Esta variante utiliza tres submatrices para almacenar los valores de los elementos y sus posiciones de coordenadas.

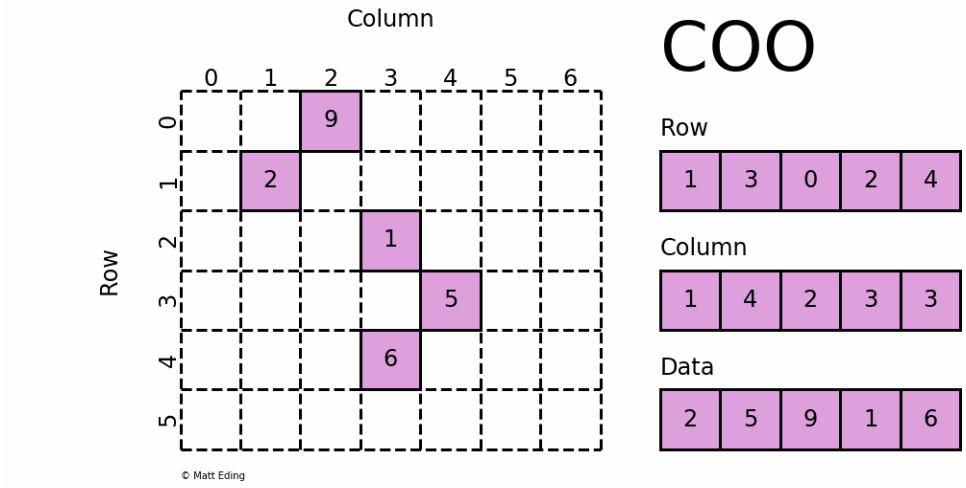


Ilustración 15 Matriz de coordenadas

El ahorro en el consumo de memoria es bastante sustancial a medida que aumenta el tamaño de la matriz. La gestión de datos en una estructura dispersa es un costo fijo a diferencia del caso de matrices densas. La sobrecarga incurrida por la necesidad de administrar los subarreglos se vuelve insignificante a medida que aumentan los datos, lo que la convierte en una excelente opción para algunos conjuntos de datos.

4.2.4.3. Matriz dispersa comprimida de filas

Los formatos descritos anteriormente son excelentes para construir matrices dispersas, pero no son tan eficaces computacionalmente como las formas más especializadas. Lo contrario es cierto para la familia de matrices dispersas comprimidas, que deben tratarse como de solo lectura en lugar de solo de escritura. La Compressed Sparse Row/Column (CSR) determina dos cosas.

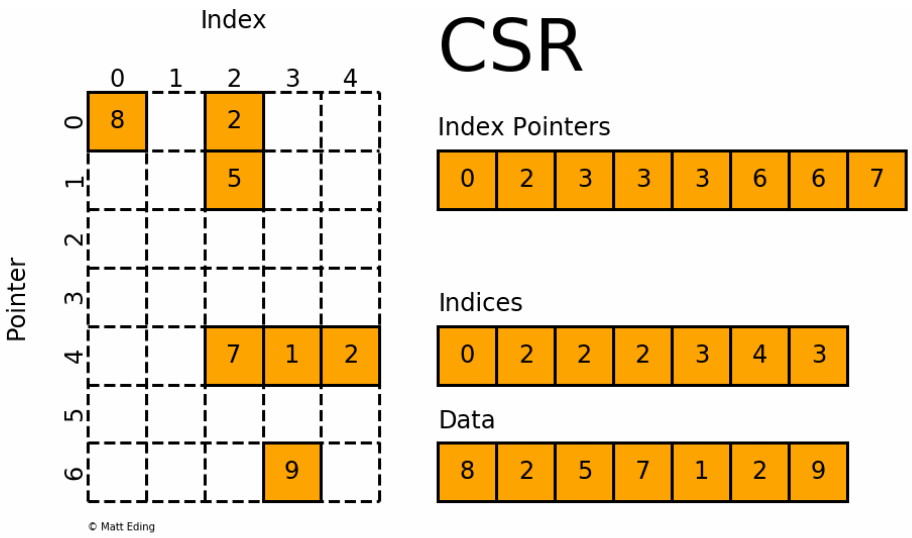


Ilustración 16 Matriz dispersa comprimida de filas

Primero, su posición en la matriz de punteros es el número de fila. En segundo lugar, estos valores representan la porción [inicio: parada] de la matriz de índices, y su diferencia son los elementos NNZ en cada fila.

4.2.5. Creación de las matrices

Para la generación de las matrices necesarias para el modelo se va a usar las funciones que proporciona LightFM para las matrices de user-features, item-features y las interacciones entre estas.

La matriz de las interacciones se utiliza para hacer la división entre los datos que se van a utilizar para entrenar el modelo y los que se van a dejar para probar la tasa de acierto. Esta división de los datos no puede ser la normal porque no tiene en cuenta a todos los elementos para entrenar el modelo. Por este motivo se ha utilizado el random test split que asegura que el tamaño de train y el de test sea el mismo (este split es el proporcionado por la librería).

4.2.6. Creación del modelo

Para la selección de un modelo se han desarrollado dos IAs: una utilizando la librería de LightFM y otra creando una red neuronal. Tras analizar la precisión de los dos modelos se concluye que el de LightFM es más preciso al obtener un 88% de precisión frente al 69% que ofrece la red neuronal. Otro aspecto importante a tener en cuenta es la velocidad de entrenado de los dos modelos; en este aspecto el modelo de LightFM es claramente superior al necesitar solo 25s en entrenar el modelo frente a los 17 minutos que necesita la red neuronal para entrenarse.

Debido a la velocidad de entendimiento del modelo elegido se utilizará Node-RED para entrenar el modelo una vez al día y así conseguir que los resultados sean lo más ajustados posibles a cada usuario.

Uno de los parámetros a tener en cuenta para la creación del modelo de LightFM es la pérdida o loss entre las que destacan dos funciones BRP y WARP. Aunque el objetivo es lograr el resultado que más se ajuste a los intereses del usuario, se ha hecho un análisis para ver cuál de los dos modelos es el más conveniente.

Para realizar este análisis se va a realizar un entrenamiento durante 70 “epochs” o épocas para ver tanto la velocidad como la precisión de entrenamiento de cada uno de ellos.

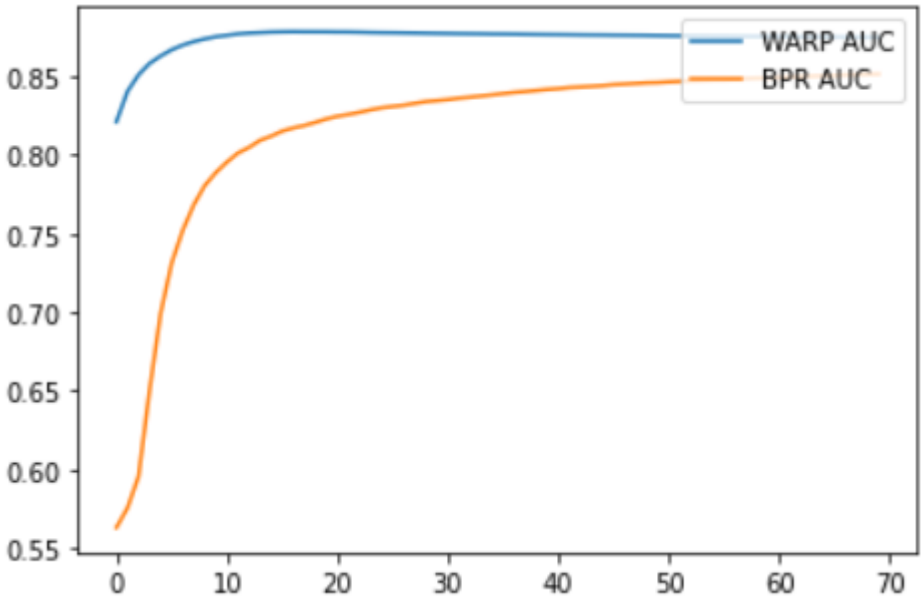


Ilustración 17 Diferencia de precisión entre WARP y BRP

Como se puede observar, el modelo WARP empieza y acaba con una mejor tasa de acierto llegando hasta el 87% mientras que el BPR a pesar de que empieza con un 56% alcanza un 84% al final de las épocas.

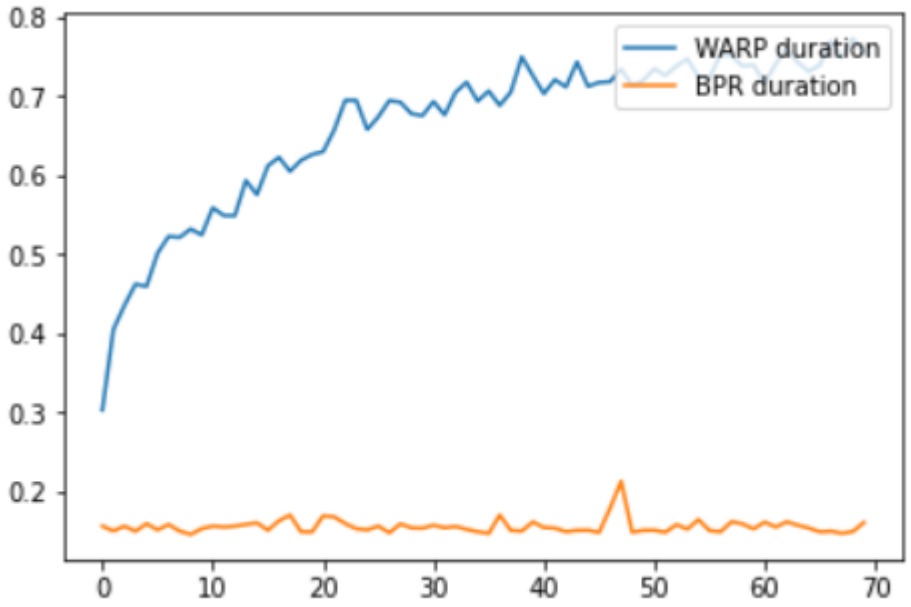


Ilustración 18 Diferencia de tiempo entre WARP y BRP

Teniendo en cuenta la velocidad que se necesita para entrenar el modelo es evidente que el modelo BPR es mucho más rápido que el WARP, pero como la diferencia es ínfima y el objetivo es la precisión el modelo WARP elegido.

Una vez seleccionado el tipo de pérdida que se va a utilizar es imprescindible seleccionar qué programa de ritmo de aprendizaje se va a utilizar. para esta librería existen dos tipos: adagrad y adadelta.

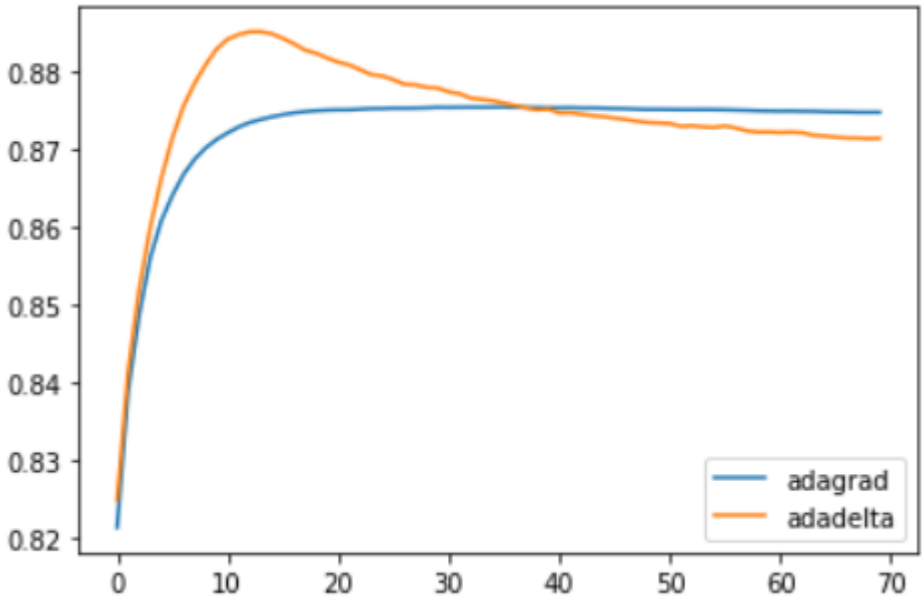


Ilustración 19 Diferencia de precisión entre adagrad y adadelta

Cómo se puede ver, aunque el modelo adadelta produzca mejores resultados al principio del entrenamiento, decae a través del tiempo y obtiene peores resultados al final de las épocas

respecto al modelo adadelta. Es por esto que se ha decidido utilizar el modelo adagrad ya que proporciona una mayor firmeza y una precisión que se mantiene a través del tiempo.

4.2.7. Entrenado del modelo

Una vez creado el modelo tenemos que entrenarlo con el fit de la librería con los datos del train, en el que se le especifican el número de threads y el de épocas y las características de tanto los usuarios como los artistas.

4.2.8. Predicción

Por último, se hace una predicción del modelo introduciendo como parámetro de entrada el identificador de un usuario. Este dará como resultado una lista con todos los artistas recomendados y se obtendrán los mejores resultados después de ordenarlos de mayor a menor valoración.

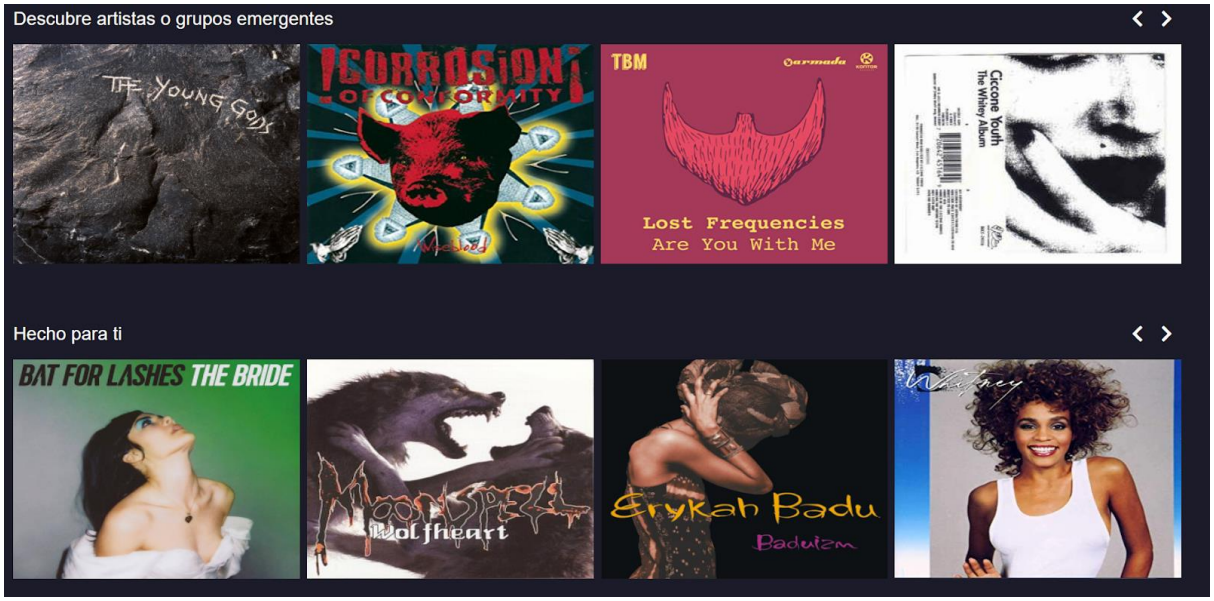


Ilustración 20 Visualización de recomendaciones para un usuario

4.2.9. Conexión con el servidor web

Para que los resultados de cada usuario puedan ser directamente visualizados por el usuario es imprescindible que ambos servicios intercambien información. Para crear el servicio de la base de datos se ha hecho uso del marco FastAPI que permite crear una API de una forma rápida y sencilla.

El funcionamiento es similar al servidor web en el que recibirá un GET o un POST con un JSON y ejecutará la función a la que se le haya llamado. Por último, hay que definir en qué dirección se va a crear y en qué puerto.

4.3. Servidor de Base de Datos

Se ha decidido ubicar la base de datos en un servidor diferente al de la página web para así si en un futuro se quiere añadir una nueva aplicación, la comunicación entre esta y la base de

datos será más sencilla mejorando así la escalabilidad del producto. Esto se pudo ver reflejado al añadir la aplicación de valoraciones.

4.3.1. Funcionamiento

Cuando el usuario hace una solicitud con el método GET, desde Node-RED se le llamará a una función que mediante Hibernate se obtendrán los datos de la base de datos solicitados por el usuario. Estos se convertirán a JSON mediante GSON y se validaran con JSON Schema Validator en Node-RED, si la estructura del mensaje es correcta se le enviará a dicho usuario la respuesta a la solicitud que ha hecho. En cambio, si la solicitud del usuario se hace con un método post, desde el backend se enviará el JSON correspondiente y éste será validado de la forma anteriormente descrita. Si la validación es correcta, el JSON será parseado al tipo de objeto que se solicita y este será guardado en la base de datos mediante Hibernate.

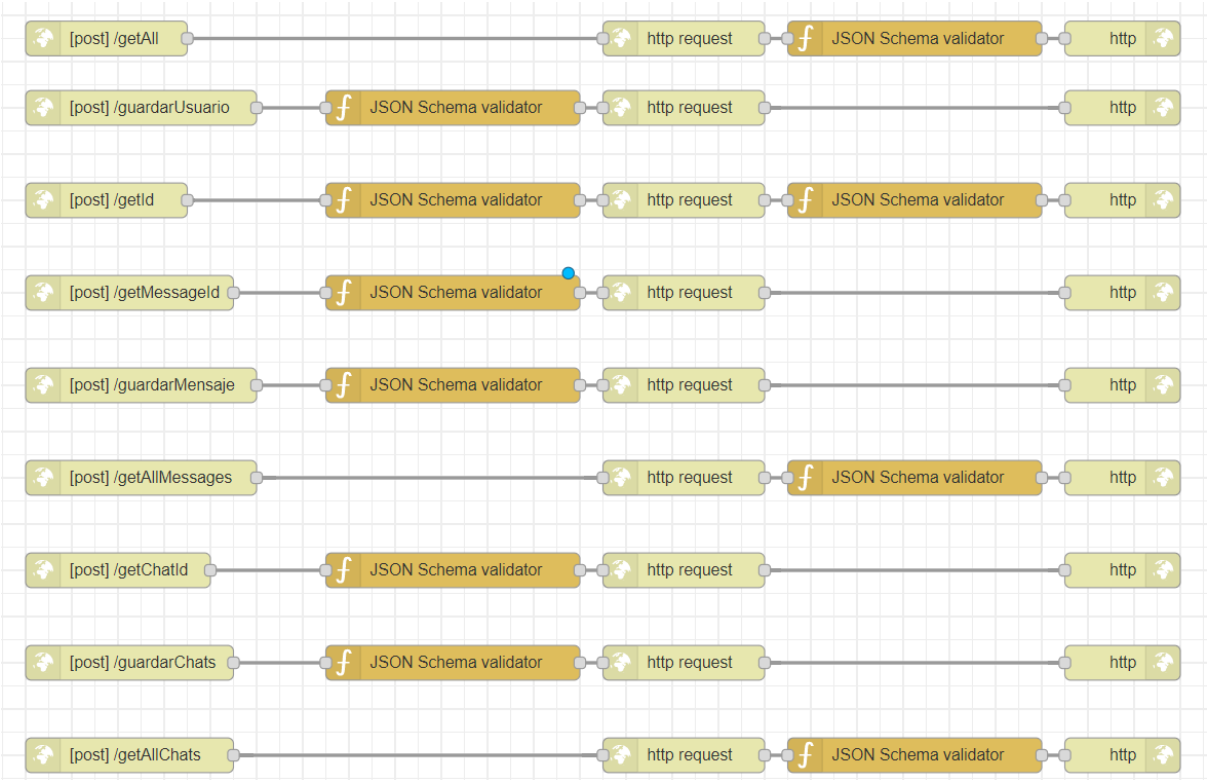


Ilustración 21 Comunicación a través de Node-RED para la base de datos

4.4. Servidor Web

En este servidor se encuentran el servicio de la página web, el de valoraciones, el de RabbitMQ y el servicio de Node-RED.

4.4.1. Página web

Para poder hacer el frontend de la página web, se ha utilizado Thymeleaf el motor de plantillas que proporciona Java, este motor permite trabajar con facilidad en aplicaciones basadas en

MVC y sustituir los ficheros de tipo JavaServer Pages (JSP) por HTML5. Para crear el backend y hacer que la página web sea dinámica, es decir, que la información presentada en el navegador sea generada a partir de una petición del usuario, se ha utilizado el marco Spring que sigue el patrón de diseño Modelo-Vista-Controlador. Mediante este marco se ha creado el mapeo completo del flujo de navegación haciendo que la página web sea totalmente funcional.

4.4.2. Valoraciones

Este servicio se ejecutará una vez al día en el servidor, mediante este programa se revisarán las contrataciones hechas para el día anterior y se les enviará a los organizadores de los conciertos un correo electrónico para que puedan evaluar cómo ha sido el servicio otorgado por el artista o grupo contratado. Cuando el organizador realice la valoración, se volverá a calcular la media de valoraciones del contratado y este valor se enseñará junto a la cantidad de valoraciones que tiene en el perfil del usuario.

Ilustración 22 Correo electrónico de valoración

4.4.3. RabbitMQ

Este servicio ha sido usado para notificar a los organizadores cuando un nuevo artista se registre en la página web y para enviarle un correo electrónico de bienvenida al nuevo usuario. Para notificar a los organizadores, cuando el backend de la página web recibe el mensaje

desde RabbitMQ, este enviará el contenido de dicho mensaje mediante correo electrónico a todos los organizadores registrados en la página. Para que esto no consuma mucho tiempo ni ralentice el programa, se ha creado un Thread Pool para poder dividir este trabajo entre los procesadores que estén disponibles en ese momento.

Ilustración 23 Correo electrónico de nuevo usuario

4.4.4. Node-RED

Mediante este servicio se ha conseguido hacer la conexión entre los diferentes servicios que proporciona Mudley que están tanto en el mismo servidor que Node-RED y los que no. Para poder enviar información de un punto a otro, se usarán objetos JSON que serán validados con su respectivo JSON Schema Validator tanto cuando lleguen y cuando salgan del flujo de Node-RED. Con estos validadores se logra que solo se puedan enviar y recibir objetos JSON con una estructura específica mejorando así la seguridad de la conexión.

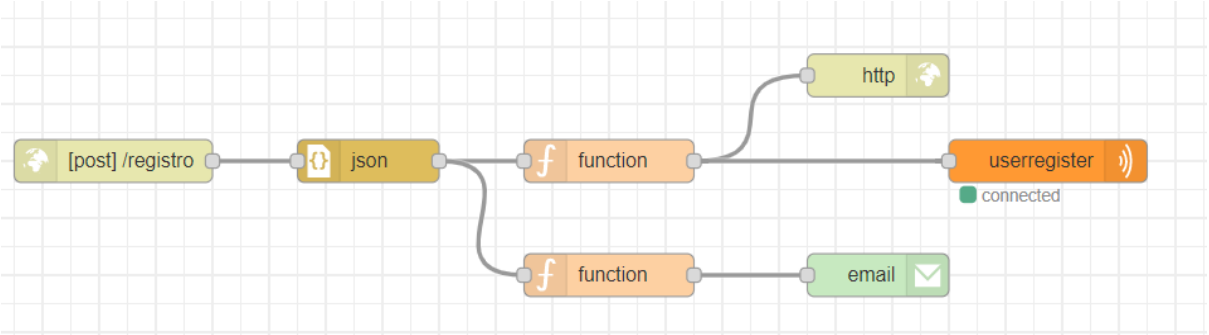


Ilustración 24 Comunicación entre la aplicación y RabbitMQ

4.5. Servidor de Monitorización

Es importante garantizar que todos los servidores funcionen de manera correcta. El objetivo del servidor de monitorización es garantizar que el resto de los servicios funcionen bien, para ello, se ha utilizado Zabbix.

4.5.1. Monitorización con Zabbix

Zabbix es un Sistema de Monitorización de Redes diseñado para monitorizar y registrar el estado de varios servicios de red, Servidores, y hardware de red. En Mudley se utiliza este servicio para monitorizar los diferentes servidores.

Zabbix despliega diferentes gráficos con información útil para tomar decisiones respecto al servidor. El sistema también está configurado para que, en caso de que exista algún problema, notifique a los encargados vía Gmail.

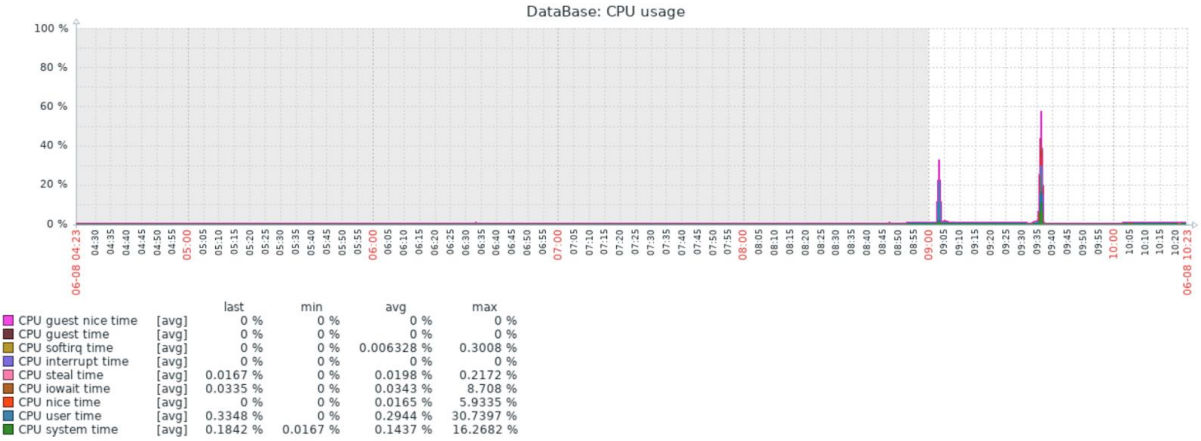


Ilustración 25 Gráfico de monitorización de Zabbix

Los errores que existen en Zabbix pueden variar, al igual que los estados del servidor. En la imagen que se muestra abajo se pueden ver cómo están categorizados. En caso de que salte alguno de estos errores, o el servidor deje de estar disponible, empezará el protocolo de correo electrónico, enviando un correo a los responsables.

Ilustración 26 Tipos de errores de Zabbix

4.6. Servidor de análisis e integración continua

Se ha creado un servidor utilizando Amazon Web Services para implementar SonarQube para el análisis del código de la aplicación, y también para implementar la integración continúa utilizando Jenkins.

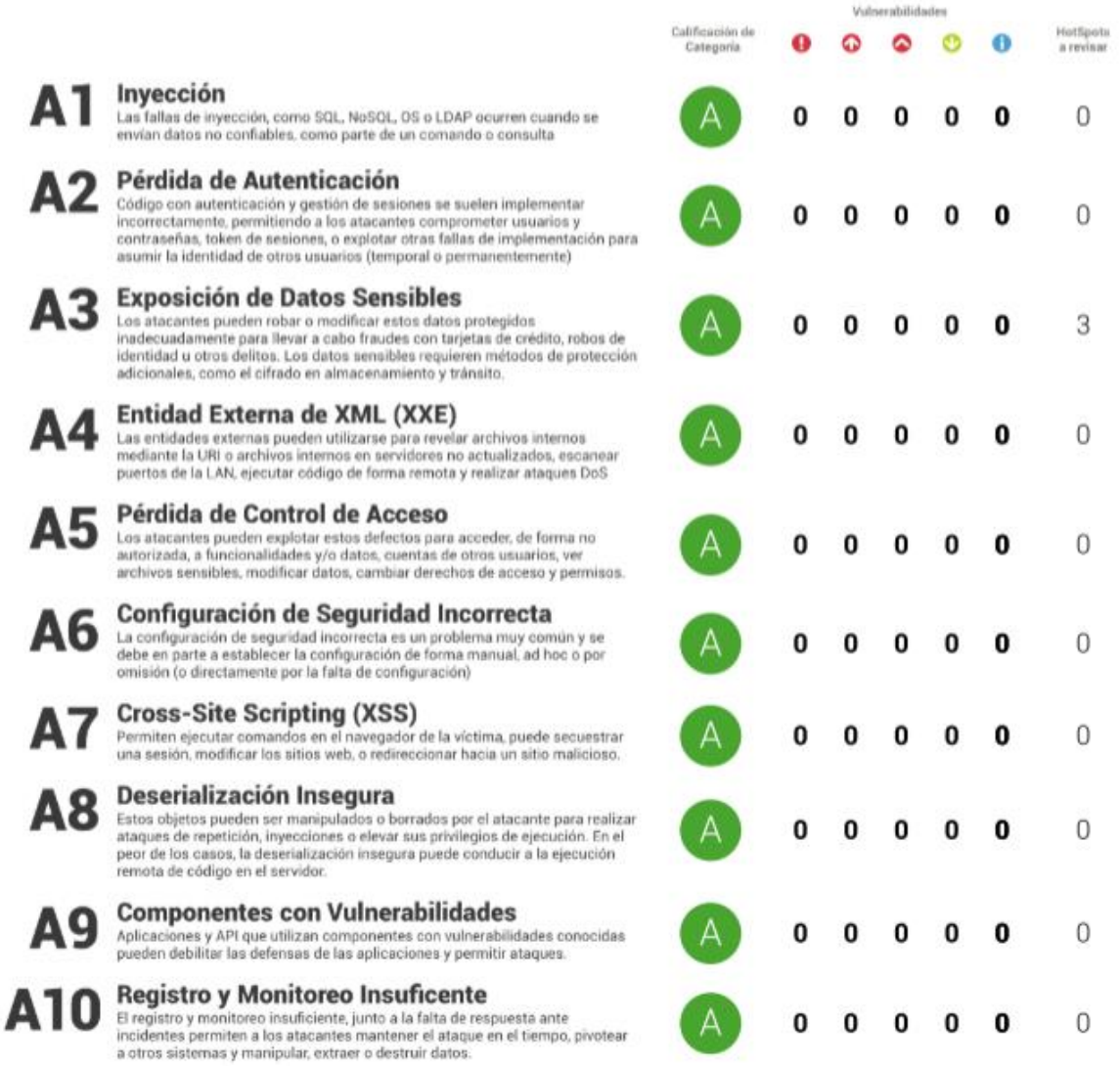
4.6.1. Análisis del código

El análisis de código es una metodología que establece una serie de reglas, así como si el código generado está bien formado y estructurado. Para ello hay herramientas de análisis que evalúan el código y en base a las reglas aplicadas, nos reportará una serie de errores a corregir. Todos estos errores dependen del lenguaje que estemos utilizando, aunque algunos serán estándar entre todos ellos.

4.6.1.1. SonarQube

Para desarrollar el código de la aplicación web correctamente, se ha hecho uso de SonarQube durante este proyecto. Este es una plataforma de software libre que se utiliza para evaluar el código fuente mediante herramientas de análisis estático. Lo que hace es identificar los puntos susceptibles de mejora, que facilitarán la obtención de métricas necesarias para la optimización del código.

Por otro lado, se ha hecho uso de Bitegarden, que es un plugin de SonarQube. Gracias a esta extensión se podrá conseguir una mayor robustez del código, ya que nos añadirá la posibilidad de analizar nuevas funcionalidades. Así como el Cross-Site Scripting, la exposición de datos sensibles o la pérdida de autenticación entre otras muchas.



		Calificación de Categoría	Vulnerabilidades					HotSpots a revisar
A1	**Inyección** Las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables, como parte de un comando o consulta	A	0	0	0	0	0	0
A2	**Pérdida de Autenticación** Código con autenticación y gestión de sesiones se suelen implementar incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente)	A	0	0	0	0	0	0
A3	**Exposición de Datos Sensibles** Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.	A	0	0	0	0	0	3
A4	**Entidad Externa de XML (XXE)** Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques DoS	A	0	0	0	0	0	0
A5	**Pérdida de Control de Acceso** Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos.	A	0	0	0	0	0	0
A6	**Configuración de Seguridad Incorrecta** La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, ad hoc o por omisión (o directamente por la falta de configuración)	A	0	0	0	0	0	0
A7	**Cross-Site Scripting (XSS)** Permiten ejecutar comandos en el navegador de la víctima, puede secuestrar una sesión, modificar los sitios web, o redireccionar hacia un sitio malicioso.	A	0	0	0	0	0	0
A8	**Deserialización Insegura** Estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.	A	0	0	0	0	0	0
A9	**Componentes con Vulnerabilidades** Aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir ataques.	A	0	0	0	0	0	0
A10	**Registro y Monitoreo Insuficente** El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos.	A	0	0	0	0	0	0

Ilustración 27 Análisis de OWASP en SonarQube

4.6.2. Integración continua

La integración continua es una práctica de ingeniería de software que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes. Entendemos por integración la compilación y ejecución de pruebas de todo un proyecto.

4.6.2.1. Jenkins

Para realizar la integración continua se ha utilizado Jenkins, que es un servidor de automatización open source escrito en Java. Este ha sido utilizado durante el desarrollo completo del proyecto. Cada vez que se hacía un cambio en GitLab se realizaban los siguientes pasos: “Building”, “Testing” y el análisis estático mediante SonarQube.

Gracias al uso de Jenkins todos los integrantes del grupo podían saber el estado del proyecto, facilitando la mejora y la detección de errores del código.

4.7. Gestión de activos y servicios

La gestión de activos físicos es un elemento fundamental para asegurar la competitividad de las organizaciones que generan valor mediante la utilización de estos activos, pero además resulta una herramienta importantísima para la toma de decisiones relacionadas con las finanzas corporativas.

Para llevar esto a cabo, se ha hecho uso de la herramienta de Proactivanet. Esta es una solución para gestionar activos y servicios TI (ITAM & ITSM), que con sus módulos de Discovery & Gestión de Activos, Service Desk y CMDB, ayuda a alcanzar un nuevo nivel de madurez de una forma ágil y ordenada, estableciendo nuevos procesos de gestión TI para aumentar la productividad, automatizar procesos, mejorar el control, disminuir riesgos y reducir costes de operación aportando valor al negocio.

4.7.1. CMDB

La CMDB (Configuration Management Database) es el sistema que permite registrar la información de la infraestructura y gestión del servicio mediante entidades denominadas CIs (Configuration Items). El uso adecuado de la CMDB ayuda a tener un mejor conocimiento, en una organización, de la infraestructura TI. Ayudando al negocio a tomar decisiones más adecuadamente teniendo en cuenta los servicios que se prestan.

Los CI son los puntos de enfoque en una CMDB, es decir, son los elementos de configuración que son componentes de una infraestructura estando bajo el manejo de configuración.

En una CMDB debe aparecer las relaciones que tienen los diferentes componentes entre sí. Para poder comprender mejor las funcionalidades de estas en el sistema y para ayudar a mantener el seguimiento de sus configuraciones.

En el caso de la empresa Mudley, se creará tres tipos de CI. Todos ellos serán los servicios que esta empresa ofrece. Se podrán diferenciar en 3 diferentes grupos: servicios externos, servicios técnicos y servicios de negocio. (Ver Anexo C – Proactivanet)

Ilustración 28 Servicios Técnicos

4.7.2. Roles

Para llevar una correcta gestión de los problemas y de los incidentes que la empresa puede tener, es imprescindible que los trabajadores tengan sus roles.

Un rol define un conjunto de habilidades, competencias y responsabilidades que están relacionadas. Los miembros individuales de la empresa tendrán distintas funciones o realizarán roles diferentes. Cada trabajador puede tener múltiples roles simultáneamente.

En el caso de la empresa Mudley podremos diferenciar dos tipos de roles: Soporte nivel 1 y nivel 2. (Ver Anexo C – Proactivanet)

Ilustración 29 Roles de la empresa Mudley

4.7.3. Gestión de incidencias

Una incidencia es algo que se produce en el transcurso de algún asunto, relato, etc… y que repercute el sistema interrumpiendo o alterando.

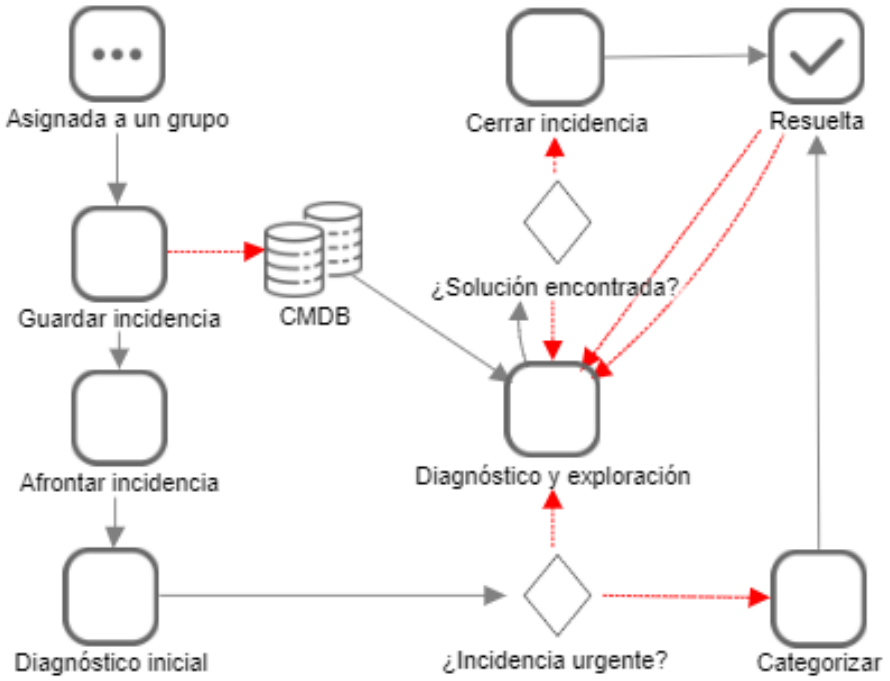


Ilustración 30 Flujo de trabajo del gestor de incidencias

Para ello es importante implementar un gestor de incidencia con su respectivo flujo de trabajo. El objetivo de este gestor es planificar y facilitar todas las actividades involucradas en el proceso de gestión de incidentes. Asegurarse de que se siga el proceso correcto para todos los tiques y corregir cualquier desviación. Y, por último, también coordinarse y comunicarse con el propietario del proceso. (Ver Anexo C – Proactivanet)

4.7.4. Gestión de problemas

Un problema en ITIL es algo que causa una o más incidencias. Por lo que es importante implementar una gestión de problemas.

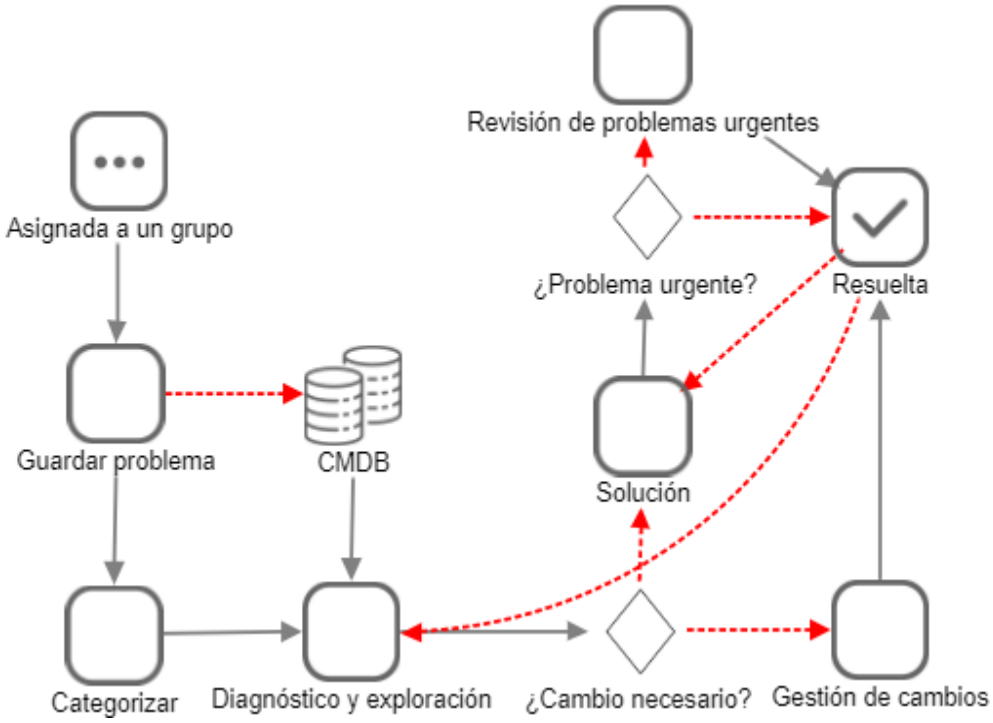


Ilustración 31 Flujo de trabajo del gestor de problemas

Para ello es importante implementar un gestor de problemas con su respectivo flujo de trabajo. La gestión de problemas es un procedimiento para minimizar los incidentes causados por las operaciones de infraestructura de TI al profundizar en los incidentes para determinar la causa raíz y encontrar soluciones, y también para reducir la gravedad de los incidentes al documentar los problemas existentes y proporcionar soluciones alternativas. (Ver Anexo C – Proactivanet)

4.7.5. Catálogo y portfolio de servicios

Durante el desarrollo del proyecto de Mudley se ha diseñado un catálogo de servicios y un portfolio de servicios. Es importante contar con ellos para dar a conocer a los clientes o potenciales clientes cuál es el trabajo que realiza la empresa. Si son creados de manera completa y con calidad, no quedan dudas de qué es lo que se está vendiendo.

El Catálogo de Servicios es la única parte del portafolio de servicios publicada a los clientes, y se utiliza para apoyar la venta y la entrega de servicios de TI. El Catálogo de Servicios incluye información sobre entregas, precios, puntos de contacto, pedidos y procesos de solicitud. Por otro lado, el portfolio aparecerá los servicios que aún no están implementados y que están en desarrollo y en el catálogo solamente los servicios que estén disponibles actualmente. (Ver Anexo D – Catálogo de Servicios) (Ver Anexo E – Portfolio de servicios)

4.7.6. Plan de viabilidad

Para poder calcular si la inversión que se realiza en la empresa tiene un beneficio económico rentable, es necesario calcular el ROI (retorno de la inversión). Por un lado, se calculará cuánto es la inversión y por el otro las ganancias.

Para el correcto funcionamiento del sistema, la empresa tiene implementado 5 servidores utilizando los servicios que nos ofrecen Amazon Web Service. Se encontrará los servidores

de: web, base de datos, Sonar y Jenkins (Estos dos se encontrarán en el mismo), Zabbix e inteligencia artificial. Estos servidores habrá que pagarlos por el tiempo que estén operativos. Teniendo en cuenta que deberán estar en funcionamiento siempre, para poder ofrecer los servicios a los usuarios en cualquier momento. Habrá que pagar a Amazon lo mismo por servidor anualmente, ya que todos los servidores tienen las mismas características actualmente. Se ha elegido “t2.small” como tamaño de los cinco servidores, con sus correspondientes características. Por otro lado, la única ganancia de la aplicación actualmente consiste en la visualización de los anuncios.

Con esto se puede determinar si la inversión es rentable o no. El ROI empieza siendo positivo, pero, además, cuanto más pase el tiempo, el ROI se vuelve cada vez más positivo. Así que a la larga la inversión llegará a ser más rentable. Así que cada año aumentarán las ganancias de la empresa. (Ver Anexo F – Factura)

4.7.7. Análisis de riesgos empresariales

Se ha realizado un análisis de los riesgos empresariales para que la empresa pueda hacerlos frente, estando previamente preparados.

Los análisis de riesgos conforman una herramienta muy importante en cualquier empresa. Estos consisten en una serie de técnicas y evaluaciones de carácter cualitativo y cuantitativo que ayudan a encontrar riesgos de distinta naturaleza para una compañía.

A diferencia del análisis de riesgos en el campo de la ciberseguridad, este se centrará en los riesgos que amenacen en el campo empresarial. (Ver Anexo G - Clientes, procesos de interacción, herramientas de desarrollo, análisis de riesgos empresariales)

4.7.8. Acuerdo de nivel de servicio

El SLA (Service Level Agreement o en español acuerdo de nivel de servicio) es un documento donde se describe el contrato que se tiene con los clientes como empresa Mudley, en el que se ofrecerá un servicio web para la gestión de grupos de música.

En él se encontrará la disponibilidad, la capacidad y la descripción del servicio que la empresa ofrece. Por otro lado, también se especificará como se podrá recompensar a los clientes en el caso del incorrecto funcionamiento de los servidos, a su vez de los servicios, que pueden afectar negativamente a los clientes. (Ver Anexo H – SLA)

4.8. Seguridad

Tener un buen sistema de seguridad ayudará a proteger el sistema contra diversas amenazas como el ransomware, malware, entre otros. Así, tus datos y redes estarán seguras evitando el ingreso de usuarios no autorizados que puedan tener malas intenciones.

4.8.1. Análisis de Riesgos

Antes de definir las medidas de seguridad necesarias para nuestro sistema, es necesario analizar e identificar qué medidas se deberían tomar en base a la arquitectura y requisitos del sistema. Para identificar las soluciones se ha hecho un análisis de riesgos siguiendo la metodología de Magerit v3.

En el análisis de riesgos se han tenido en cuenta los activos de la empresa, las posibles amenazas para cada activo y los salvaguardas previamente instaladas en el sistema. Una vez identificados, se han analizado los posibles riesgos, la probabilidad de estos y su impacto, y con esta información, las salvaguardas necesarias para mitigar los riesgos analizados.

Además, una vez instaladas todas las medidas definidas, se ha hecho un segundo análisis de riesgos para identificar los riesgos residuales y las medidas para mitigar estos riesgos. (Ver Anexo I – Análisis de riesgos)

4.8.2. Salvaguardas

Estos son las salvaguardas aplicadas para garantizar la confidencialidad, integridad y disponibilidad de nuestro sistema:

4.8.2.1. TLS/SSL

Utilizamos TLS/SSL para cifrar el tráfico de datos entre los servicios REST además del servidor de RabbitMQ. Utilizamos los certificados de Let's Encrypt en el servidor de RabbitMQ y en Node-RED puesto que todas las conexiones entre servidores se hacen mediante Node-RED. Los certificados son almacenados en el servidor, pero Let's Encrypt tiene la capacidad de actualizar y renovar los certificados

El objetivo de estos certificados es conseguir un sistema de criptografía asimétrica. El emisor utiliza la clave pública del servidor para cifrar el mensaje. Este mensaje se envía cifrado por la red, lo que impide que los atacantes puedan interceptar la información. Una vez los datos llegan al servidor, este utiliza su clave privada para desencriptar el mensaje. De esta manera, se consigue una conexión segura sin el riesgo de enviar las claves criptográficas por la red.

Ilustración 32 Criptografía asimétrica

4.8.2.2. HTTPS

El protocolo de red HTTPS utiliza TLS/SSL, por lo que la metodología es la misma que en la medida previamente explicada. El servidor web utiliza los certificados de Let's Encrypt para cifrar el tráfico mediante TLS y garantiza así a los usuarios de nuestra plataforma la seguridad y veracidad de nuestra página web.

Ilustración 33 Creación de certificado HTTPS

4.8.2.3. BCrypt

Debido a nuestra política de privacidad aseguramos que las contraseñas de todos los usuarios estarán protegidas y que ningún atacante pueda hacerse con ellas. Para ello, utilizamos BCrypt, una función de hash de contraseñas que además añade salt a estas para hacer imposible la obtención de contraseñas.

Esto se pretende solventar aplicando las siguientes medidas: Como primera instancia, se establecerá el tipo “password” en los inputs de HTML para evitar que otro individuo vea al usuario de la aplicación introducir la contraseña y anotarla. También se tendrá en cuenta el robo de contraseñas desde la base de datos, por lo que se establecerá un usuario y contraseña específicos que serán los únicos que tendrán permitido el acceso a las consultas y modificaciones de la base de datos.

Una vez la contraseña sea introducida por el usuario será encriptada usando Bcrypt. Esta, es una función de hash-eado de contraseñas basada en el cifrado Blowfish (ver apéndice 6.5). Además, utiliza una sal criptográfica (ver apéndice 6.6) para proteger la aplicación de posibles ataques usando “rainbow tables” o tablas de arcoíris (ver apéndice 6.7). Bcrypt es una función adaptativa: con el tiempo, el recuento de iteraciones se puede aumentar para hacerlo más lento, por lo que sigue siendo resistente a los ataques de búsqueda de fuerza bruta incluso con un poder de cálculo creciente.

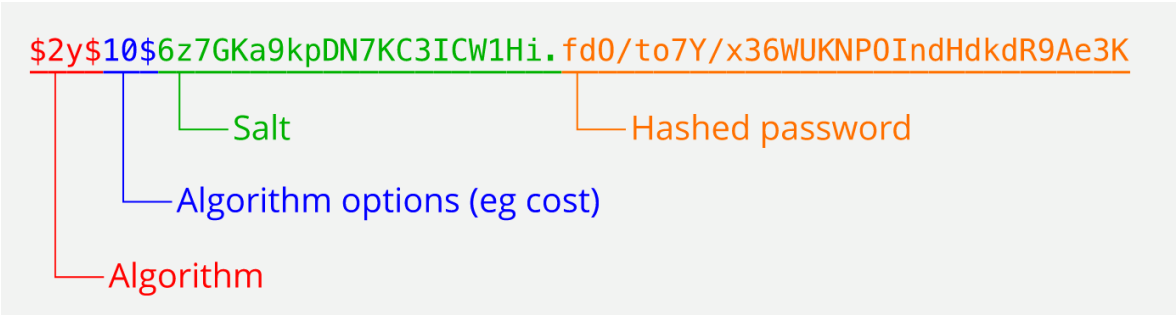


Ilustración 34 Construcción de hash de contraseña

4.8.2.4. Spring Security

Aprovechando que se ha utilizado Spring MVC se ha implementado Spring Security que es un marco Java / Java EE que proporciona autenticación, autorización y otras características

de seguridad para aplicaciones empresariales. Estos son los pasos que se siguen para autenticar a un usuario:

1. Se carga la configuración de SecurityConfiguration.
2. Cuando el usuario introduce sus credenciales y éstas se envían, el filtro de autenticación de Spring Security intercepta la petición y se crea un objeto. UsernamePasswordAuthenticationToken con las credenciales.
3. loadUserByUsername() recibe el nombre de usuario.
4. Se crea un objeto MyUserDetails con el nombre de usuario enviado y todo lo demás con valores hardcoded (simulando ser un usuario que tengamos en base de datos) y se compara MyUserDetails con el objeto UsernamePasswordAuthenticationToken.
5. Si todo es correcto se accede al recurso, en caso contrario, permiso denegado.

Spring Security utiliza roles para identificar los permisos para cada usuario; es decir, un usuario no identificado en la aplicación carece de sesión y por tanto de rol y solo podrá visualizar la página principal y las de “login” y “register”. De esta forma, si por algún casual conociera la dirección de otra de las páginas introducidas y decidiera introducirla no podría acceder a su contenido. Este mecanismo de roles es indispensable para la aplicación de Mudley ya que al haber dos tipos de usuarios podemos diferenciarlos dándoles un rol específico a cada uno. Así, se puede asegurar que un artista no pueda iniciar un chat con un usuario, pero sí que sea al revés.

4.8.2.5. Firewall

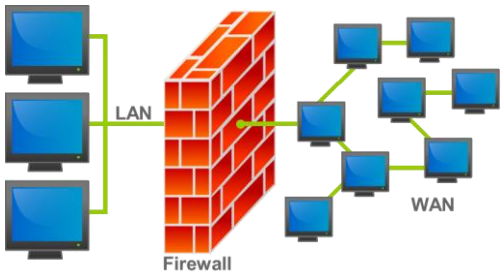


Ilustración 35 Red con firewall

Es importante asegurar que el acceso a los servidores sea el adecuado. Es necesario definir qué puertos se van a utilizar y desde donde se podrá acceder. Para ello, utilizamos el firewall que proporciona AWS, habilitando solo los puertos necesarios para evitar ataques a puertos que no se utilizan. (Ver Anexo J – Reglas Firewall)

Debido a que nuestros servicios deben ser accesibles desde cualquier lugar, se ha habilitado el acceso desde cualquier red.

4.8.2.6. JSON Schema Validator

Toda la información que circula en nuestro sistema se envía y recibe en formato JSON, lo cual facilita mucho la transmisión de datos y la compatibilidad entre distintos sistemas. Sin embargo, es importante validar que estos JSON cumplen con las necesidades de cada sistema.

Por este motivo, hemos decidido utilizar JSON Schemas para validar los JSON recibidos. Para ello, hemos creado diferentes esquemas, adecuados a cada sistema, y los hemos insertado en Node-RED. Utilizamos una función de Node-RED para verificar que los JSON recibidos son correctos antes de enviarlos a otro sistema. De esta manera nos aseguramos de que la información enviada sea correcta y nuestros sistemas queden fuera de peligro.

4.8.2.7. Contraseñas seguras

También es importante para la seguridad limitar las opciones de los usuarios a la hora de enviar datos al servidor web: limitando el número de caracteres, utilizando checkboxes para elegir opciones, utilizando listas para seleccionar el país de origen…

Entre estas medidas, hemos implementado una política de contraseñas para indicar al usuario la seguridad de la contraseña para evitar que utilicen contraseñas débiles. Para lograr una contraseña fuerte se define un patrón de contraseña con más de 8 caracteres, alternancia entre mayúsculas y minúsculas, utilización de caracteres especiales y números.

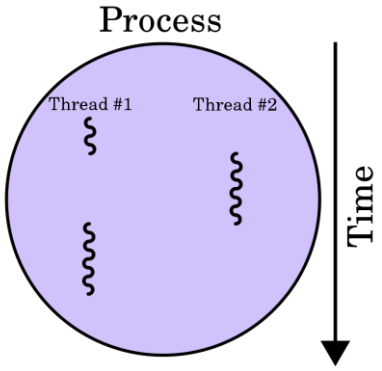


Ilustración 36 Ejecución de hilos en un proceso

4.8.2.8. Multi-threading

La disponibilidad de nuestros servicios también es una característica que debemos conservar y asegurar. Debido a que AWS es el responsable del mantenimiento de nuestros servidores, nuestra única opción es asegurar que mientras el servidor esté en marcha sea accesible y funcione debidamente. Ya que nuestro servicio está pensado para que muchos usuarios lo utilicen simultáneamente, es esencial utilizar medidas para evitar que el sistema se quede sin recursos.

Hemos decidido utilizar multithreading, para evitar el agotamiento de recursos de nuestro sistema. Utilizando más de un hilo, conseguimos que el procesador de nuestro servidor no quede a la espera de otros servicios, y, de esta manera, sea lo más eficiente posible a la hora de responder peticiones.

4.8.2.9. Copias de seguridad

No solo es importante asegurar la disponibilidad cuando el sistema funciona adecuadamente, también es imprescindible tener la posibilidad de recuperar el sistema lo más rápido posible si este cae.

Para ello hacemos copias de seguridad cada cierto tiempo, asegurando así la posibilidad de recuperar el sistema en caso de que este deje de funcionar

4.8.2.10. Persistencia

Siguiendo con la disponibilidad del sistema, se han tomado estas medidas para evitar que, en caso de caída del servicio los servidores de RabbitMQ y Node-RED mantengan la información en disco.

Utilizamos la opción de “Connection Keep Alive” en Node-RED y colas y exchanges “durables” en RabbitMQ. En ambos casos, los datos que se envían al servidor quedan guardados en disco, para que, en el caso de que el servidor caiga, si vuelve a estar disponibles, los datos no se pierdan.

4.8.2.11. CSRF tokens

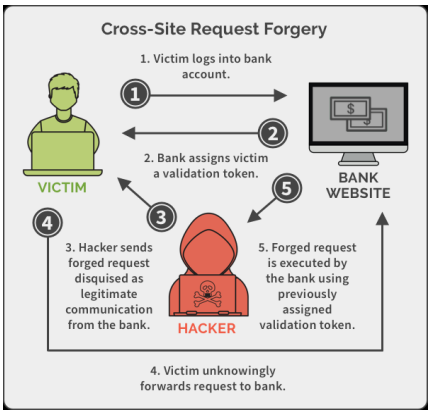


Ilustración 37 Ataque de CSRF

El CSRF (del inglés Cross-site request forgery o falsificación de petición en sitios cruzados) es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. Para evitar este tipo de ataque, utilizamos CSRF tokens.

El funcionamiento de estos tokens consiste en que el servidor envía un código generado aleatoriamente cuando el usuario hace una petición HTTP/HTTPS. Cuando el usuario hace una nueva petición, el servidor valida que la petición incluye el token previamente enviado. Esto impide al hacker perpetuar un ataque de CSRF, puesto que le es imposible predecir o generar el token.

4.8.2.12. Monitorización de los servidores

Como se ha mencionado anteriormente, se han puesto en marcha agentes de Zabbix en todos los servidores para monitorizarlos. Esta medida permite gestionar el registro de logs y establecer alertas si las medidas establecidas son sobrepasadas; es decir, establecer que se envíe un correo electrónico a los integrantes de Mudley en caso de que el uso del procesador del usuario sobrepase el 80%.

Gracias a Zabbix se pueden gestionar todo este tipo de alertas para priorizar las que más impacto tengan y prevenirlas antes de que ocurran. Un ejemplo de esto siguiendo con lo anterior sería prevenir la caída del servidor frente a un ataque DDos al ver como la capacidad del servidor sobrepasa cierta medida.

4.8.3. Audiorias de código

Para realizar un análisis de robustez del código se han utilizado dos herramientas principales, OWASP ZAP para realizar un fuzzing de los formularios y comprobar que son seguros y utilizar su herramienta llamada “spider” o araña.

La “araña” es una herramienta que se utiliza para descubrir automáticamente nuevos recursos (URL) en un sitio en particular. Comienza con una lista de URL para visitar, llamadas semillas, que depende de cómo se inicie Spider. Luego, Spider visita estas URL, identifica todos los hipervínculos en la página y los agrega a la lista de URL para visitar y el proceso continúa de forma recursiva siempre que se encuentren nuevos recursos.

Por otro lado, se ha aprovechado SonarQube para realizar un análisis estático del código y se ha implementado el plugin de Bitegarden de seguridad, basado en los estándares de seguridad OWASP, CWE, WASC, SANS y CERT, Security Plugin para SonarQube recopila una lista de vulnerabilidades detectadas en forma de problemas en SonarQube.

Tras la ejecución de estos análisis se han tomado ciertas medidas para corregir estos errores y evitarlos en el futuro. A continuación, se detallarán los más relevantes.

4.8.4. Falta de tokens Anti-CSRF

Tras un análisis con OWASP ZAP el principal problema que ocurrió fue que todos los formularios carecían de estos tokens y permiten explotar la vulnerabilidad detallada anteriormente. Dado que la aplicación está montada en Spring se ha utilizado las herramientas que proporciona para proteger estos formularios.

Estos tokens se deben introducir en los formularios como un input oculto, pero para aprovechar la versatilidad que ofrece Spring Security se ha habilitado el CSRF en la configuración para que introduzca automáticamente estos y así evitar posibles futuros errores.

4.8.5. Importaciones con HTTP

Las librerías de Ajax proporcionan una gran versatilidad y dado que el código fue inicialmente desarrollado para http todas las veces que se importaba la librería se utilizaba la de http y eso generaba fallas de seguridad al migrar la aplicación a https. Para solucionar este inconveniente se ha utilizado el “import” de https para Ajax.

Como solución a posibles futuros problemas se ha establecido como política de la empresa que todas las librerías externas que se empleen funciones con https.

4.8.6. Validacion de inputs

Al realizar un fuzzing en los diferentes formularios de la página web usando OWASP ZAP (ver Anexo K - Fuzzing), se observó que en los inputs de dichos formularios estaba permitido introducir todo tipo de valores o caracteres creando la posibilidad de poder hacer todo tipo de inyecciones haciendo que la infraestructura del sistema pueda ser atacada y dando la posibilidad de robar información o infectar a los usuarios de Mudley.

Para solucionar este problema, se han creado patterns para cada uno de los inputs obligando a los usuarios que quieran completar dichos campos usar unos tipos de caracteres o un patrón

de caracteres específico. Se ha creado una política de seguridad para cada input que se utilice en el futuro y así evitar posibles fallos humanos.

4.8.7. Visualizacion del stack trace

Aunque la visualización del stack trace es una herramienta imprescindible para el grupo para ayudar a localizar errores de una manera eficiente; sin embargo, el análisis realizado en Bitegarden lo destaca como error importante. Esto es debido a que el atacante puede ver también esta información y realizar un ataque en base a los datos obtenidos.

Por este motivo se ha adoptado como política suprimir todos los stack trace una vez el trabajador termine el desarrollo de la funcionalidad para evitar futuros fallos de seguridad.

4.8.8. Permisos en la página web

Ciertas funcionalidades de la página web, como la creación de nuevos chats, están limitadas para un tipo específico cliente, para poder hacer esto se determinaron dos tipos de roles para los usuarios: Artistas y Organizaciones. Para esto se utilizó Spring Security para confirmar que los usuarios estaban identificados y establecer las paginas a las que cada rol tendrá permiso para acceder, pero al usar las mismas páginas para hacer diferentes funcionalidades que un tipo de usuario puede utilizar y el otro no, no se podía evitar que los usuarios pudiesen usar libremente todas las funcionalidades sin ningún tipo de restricción por roles.

Para poder arreglar este problema se utilizó la función sec:authorize que tiene Thymeleaf para ver el rol que tiene el usuario que está usando la web. Esto nos permite ocultar las funcionalidades de una página que un tipo de usuario no pueda usar.

4.9. Optimización

Para poder garantizar el funcionamiento óptimo de los diferentes servicios que ofrece Mudley, se ha hecho un estudio relacionado con la optimización de los servidores. El apartado donde más se ha incidido es en la concurrencia de los tres principales servidores que se han detallado en apartados anteriores.

4.9.1. Concurrencia

El proyecto está pensado para trabajar en exclusiva en el País Vasco, con posibilidades de ampliar el mercado, llegando incluso a trabajar a nivel nacional o internacional. Teniendo en cuenta que todos los servidores están montados en los servidores EC2 de Amazon Web Service, se han tomado ciertas decisiones.

A la hora de poder optimizar la concurrencia se ha optado por ajustar al número de usuarios concurrentes que se estima que va a tener Mudley, 200 usuarios. Los tres servidores funcionan de manera diferente y requieren ajustes personalizados.

Escala	CORES	Usuarios		MaxRequest Workers	ServerLimit
		máximo	Concurrentes		
País vasco	4	10000	200	16	20
Nacional	16	80000	1000	64	80
Internacional	64	400000	4000	256	300

Ilustración 38 Análisis de la concurrencia en un servidor

El servidor de la Web va a tener un procesador de 4 núcleos, con lo que puede soportar 16 hilos con facilidad; se ha estimado que los hilos solo van a trabajar una cuarta parte del tiempo, el resto estará esperando; es decir, por cada núcleo un hilo.

En el caso de la Inteligencia artificial, se ha optado por utilizar 8 núcleos (divisible en 2 servidores de 4 núcleos para aligerar la carga de trabajo). Como la IA va a soportar cargas de trabajo bastante altas, se ha decidido que cada núcleo solo podrá atender un hilo.

Por último, el servidor de la base de datos se ha decidido utilizar 2 núcleos, cada uno de ellos soportará 2 hilos.

Para conseguir información más detallada sobre la concurrencia véase también el documento (ver Anexo L - Análisis de la concurrencia).

5. Conclusiones

Tras el test de usabilidad realizado con algunos ayuntamientos pequeños (para ver los resultados completos (ver Anexo M – Test de usabilidad Mudley (respuestas)) se ha comprobado que los grupos recomendados tras la búsqueda cumplen con las expectativas de los usuarios y logran llegar al objetivo de cumplir con las expectativas del contratante en un 90% de las veces ya que de los 23 participantes 22 han confirmado que los grupos recomendados cumplen con sus expectativas.

Para contratar un grupo o banda y formalizar esa acción el tiempo de espera promedio es de aproximadamente un día. Este periodo de tiempo se reduciría a unas horas gracias a la funcionalidad para proponer un concierto directamente a los artistas y que estos puedan aceptarlo y formalizarlo a través de un canal seguro.

Gracias a la inteligencia artificial los grupos que se acaban de registrar podrán ser vistos por todos los organizadores gracias a los artistas que se visualizan en el apartado de artistas emergentes. Este apartado les proporcionará mayor visibilidad y les permitirá ser contratados con mayor facilidad, ya que 11 de los usuarios que realizaron el test alegaron que no les hacía falta siquiera buscar que algunos de esos artistas ya cumplen con sus expectativas.

5.1. Conclusiones metodológicas

El proyecto se ha realizado por un grupo de 5 personas que han trabajado en diferentes aspectos para poder desarrollar el proyecto de la manera más adecuada. Tras haber logrado un hito marcado en la planificación, la persona o grupo que lo ha realizado le explicaba al resto de integrantes del grupo el trabajo realizado de manera que todos los miembros del grupo pudieran entender el trabajo que se había hecho.

Seguir la planificación inicialmente marcada ha sido difícil, debido a distintos contratiempos que han podido surgir durante el desarrollo del POPBL. Durante este proceso también ha variado el planteamiento de la planificación, debido a los diferentes problemas y las nuevas necesidades que han surgido.

5.2. Conclusiones técnicas

El apartado que el grupo más valora es; no solo aplicar diferentes conocimientos en el proyecto, sino el hecho de poder unirlos en uno solo. Se ha conseguido que varios servidores sean capaces de interactuar entre sí, mediante la utilización de diferentes tecnologías y lenguajes de programación.

Pero, ante todo, se ha conseguido un producto fiel a la idea original: una aplicación que permite que artistas y organizadores puedan interactuar entre ellos de una manera sencilla y eficaz, ayudándoles a llegar a un acuerdo para organizar un concierto.

6.Lineas futuras

Todos los servicios funcionan de forma independiente lo que facilita la escalabilidad; es por esto que el primer objetivo sería implementar una aplicación móvil que reciba notificaciones. Esta aplicación sería de rápida implementación ya que solo tendría que conectarse a la base de datos que está establecida como un servicio independiente a la aplicación web, con solo enviar un JSON al servicio de Node-RED podría acceder a las funcionalidades de la base de datos.

Para lograr una mejor disponibilidad para la aplicación, además de las medidas ya impuestas (establecer en Node-RED y RabbitMQ que la información no se pierda si se cae el servidor, las copias de seguridad…) Se pretende instalar un servidor esclavo para suplir al principal en caso de fallo. Para ello todos los servidores tendrían un servidor esclavo y el de bases de datos tendría una réplica de Mysql por si hubiera alguna alteración en la principal.

RabbitMQ y Node-RED utilizan TLS para cifrar las conexiones. Sin embargo, debido a que Let’s Encrypt solo proporciona certificados de servidores, se pretende utilizar también certificados de cliente en un futuro para la autenticación, y de esta manera evitar que usuarios sin permiso puedan acceder a estos servidores.

Actualmente el producto está destinado a un grupo reducido de usuarios, que solo comprende el País Vasco. Se pretende ofrecer el servicio a más usuarios, llegando a nivel nacional e internacional.

7. Apéndice

En el apéndice saldrá información que ayudará a los lectores a comprender la tesis o proporcionará información esencial sobre el proceso de la investigación.

7.1. BPR

Pérdida por pares de clasificación Bayesiana personalizada [1]. Maximiza la diferencia de predicción entre un ejemplo positivo y un ejemplo negativo elegido al azar. Útil cuando sólo están presentes interacciones positivas y se desea optimizar el AUC de ROC.

7.2. WARP

Pérdida ponderada de rango aproximado por pares [2]. Maximiza el rango de ejemplos positivos muestreando repetidamente ejemplos negativos hasta que se encuentre uno que viola el rango. Útil cuando sólo están presentes interacciones positivas y se desea optimizar la parte superior de la lista de recomendaciones (precisión @ k).

7.3. Precisión en K

Mida la precisión en la métrica k para un modelo: la fracción de positivos conocidos en las primeras k posiciones de la lista clasificada de resultados. Una puntuación perfecta es 1.0.

7.4. ROC AUC

La roc auc mide la probabilidad de que un ejemplo positivo elegido al azar tenga una puntuación más alta que un ejemplo negativo elegido al azar. Una puntuación perfecta es 1.0.

7.5. Blowfish

En criptografía, Blowfish es un codificador de bloques simétricos, diseñado por Bruce Schneier en 1993 e incluido en un gran número de conjuntos de codificadores y productos de cifrado. No se han encontrado técnicas de criptoanálisis efectivas contra el Blowfish.

Blowfish usa bloques de 64 bits y claves que van desde los 32 bits hasta 448 bits. Es un codificador de 16 rondas Feistel y usa llaves que dependen de las Cajas-S. Tiene una estructura similar a CAST-128, el cual usa Cajas-S fijas.

El diagrama muestra la acción de Blowfish. Cada línea representa 32 bits. El algoritmo guarda 2 arrays de subclaves: El array P de 18 entradas y 4 cajas-S de 256 entradas. Una entrada del array P es usada cada ronda, después de la ronda final, a cada mitad del bloque de datos se le aplica un XOR con una de las 2 entradas del array P que no han sido utilizadas.

La función divide las entradas de 32 bits en 4 bloques de 8 bits, y usa los bloques como entradas para las cajas-S. Las salidas deben estar en el módulo 232 y se les aplica un XOR para producir la salida final de 32 bits.

Debido a que Blowfish está en la red Feistel, puede ser invertido aplicando un XOR entre P17 y P18 al bloque texto codificado, y así sucesivamente se usan las P-entradas en orden reversivo.

La generación de claves comienza inicializando los P-arrays y las cajas-S con los valores derivados de los dígitos hexadecimales de pi, los cuales no contienen patrones obvios. A la clave secreta se le aplica un XOR con las P-entradas en orden (ciclando la clave si es necesario). Un bloque de 64 bits de puros ceros es cifrado con el algoritmo como se indica. El texto codificado resultante reemplaza a P1 y P2. Entonces el texto codificado es cifrado de nuevo con las nuevas subclaves, P3 y P4 son reemplazados por el nuevo texto codificado. Esto continúa, reemplazando todas las entradas del P-array y todas las entradas de las cajas-S. En total, el algoritmo de cifrado Blowfish correrá 521 veces para generar todas las subclaves, cerca de 4KB de datos son procesados.

7.6. Sal criptográfica

En criptografía, la sal (en inglés, salt) comprende bits aleatorios que se usan como una de las entradas en una función derivadora de claves. La otra entrada es habitualmente una contraseña. La salida de la función derivada de claves se almacena como la versión cifrada de la contraseña. La sal también puede usarse como parte de una clave en un cifrado u otro algoritmo criptográfico. La función de derivación de claves generalmente usa una función hash. A veces se usa como sal el vector de inicialización, un valor generado previamente.

Los datos con sal complican los ataques de diccionario que cifran cada una de las entradas de este: cada bit de sal duplica la cantidad de almacenamiento y computación requeridas.

7.7. Tablas arcoiris

Las contraseñas en un sistema informático no se almacenan directamente como texto sin formato, sino que se codifican mediante cifrado. Una función hash es una función unidireccional, lo que significa que no se puede descifrar. Siempre que un usuario ingresa una contraseña, se convierte en un valor hash y se compara con el valor hash ya almacenado. Si los valores coinciden, el usuario está autenticado.
Una tabla de arco iris es una base de datos que se utiliza para obtener la autenticación descifrando el hash de la contraseña. Es un diccionario precalculado de contraseñas de texto sin formato y sus valores hash correspondientes que se pueden utilizar para averiguar qué contraseña de texto sin formato produce un hash en particular.

Los ataques de tablas de arco iris se pueden prevenir fácilmente utilizando técnicas de sal, que son datos aleatorios que se pasan a la función hash junto con el texto sin formato. Esto asegura que cada contraseña tenga un hash generado único y, por lo tanto, se evita el ataque de tabla de arco iris, que funciona según el principio de que más de un texto puede tener el mismo valor de hash.

8.Bibliografía

Sparse Matrices · Matt Eding. (2019, 25 abril). Python & Data Science Blog. https://matteding.github.io/2019/04/25/sparse-matrices/

Wikipedia. (2019, 11 julio). Blowfish. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Blowfish

Bruce Schneier (1993). «Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)». Fast Software Encryption, Cambridge Security Workshop Proceedings (Springer-Verlag): 191-204.

Morris, Robert; Thompson, Ken (3 de abril de 1978). Password Security: A Case History (en inglés). Murray Hill, NJ, USA: Bell Laboratories.

GeeksforGeeks. (2018, 10 junio). Understanding Rainbow Table Attack. https://www.geeksforgeeks.org/understanding-rainbow-table-attack/

MAGERIT v.3 : Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información. (2012). MAGERIT v.3. https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Magerit.html#.YKYNZibtais

Rendle, Steffen, et al. “BPR: Bayesian personalized ranking from implicit feedback.” Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009.

Weston, Jason, Samy Bengio, and Nicolas Usunier. “Wsabie: Scaling up to large vocabulary image annotation.” IJCAI. Vol. 11. 2011.

Welcome to LightFM’s documentation! — LightFM 1.15 documentation. (2016). LightFM. https://making.lyst.com/lightfm/docs/index.html

Documentation : Node-RED. (2016). Node Red Documentation. https://nodered.org/docs/

O'Leary, Nick. "Releases". GitHub. Retrieved April 29, 2021.
Heath, Nick (March 13, 2014). "How IBM's Node-RED is hacking together the Internet of things". techrepublic.com. CBS Interactive. Retrieved January 16, 2017.

Community staff writer (June 14, 2016). "Version 0.14 released". nodered.org/blog. Node-RED. p. 1. Retrieved July 6, 2016. MQTT with TLS support

Diaz, Angel Luis (October 17, 2016). "IBM and partners launch JS Foundation - Cloud computing news". IBM. Retrieved October 20, 2017.

Powers, Calvin; Watson, Todd; Lewis, Ashley (October 17, 2016). "Node-RED Joins the JS Foundation". IBM developerWorks TV/video channel. YouTube. Retrieved October 20, 2017.

Lewis, Karen (October 17, 2016). "Node-RED visual programming for the Internet of Things (IoT) is now a JS Foundation Project". IBM Internet of Things blog. IBM. Retrieved February 7, 2017.

RabbitMQ Tutorials — RabbitMQ. (2007). RabbitMQ. https://www.rabbitmq.com/getstarted.html

Joern Barthel (13 de septiembre de 2009). «Getting started with AMQP and RabbitMQ» (en inglés). InfoQ.

Peter Cooper (9 de abril de 2009). «RabbitMQ - A Fast, Reliable Queuing Option for Rubyists» (en inglés). RubyInside.

«RabbitMQ: An Open Source Messaging Broker That Just Works» (en inglés). Google Tech Talks. 25 de septiembre de 2008.

B. (2021, 5 junio). Spring Tutorial. Baeldung. https://www.baeldung.com/spring-tutorial