Daniel Riesgo González

Descripción breve

Esta es la memoria de mi proyecto de curso.  
Aquí explico los motivos de que la he hecho, sus requisitos,  
tecnologías utilizadas y un manual de uso

Spotistats

Proyecto fin de curso

1. INTRODUCCIÓN
   1. Contexto.

Decidí crear esta web porque consideraba útil tener una página web que te diese más control sobre la información de tus hábitos de escucha en Spotify además de aunarlo un poco con más información de dichos artistas y canciones.

Además, esta web te permite consultar acerca de los eventos cercanos de dichos artistas y noticias dentro de la propia web, cosa que consideraba útil ya que es una función de la que Spotify como tal no tiene y creo que esto hace una web completa ya que permite ver tus hábitos de escucha, crear playlists en base a artistas y canciones, ver eventos más cercanos, poder escuchar tus canciones y discos dentro de la propia web y además consultar noticias acerca de ellos.

* 1. Presentación y Objetivos

A continuación, voy a describir el trabajo realizado para el proyecto final del Ciclo Superior de FP de DAW en el IES Monte Naranco. Dicho proyecto consiste en una web que te permita hacer login con tu cuenta de Spotify y consultar información acerca de ella, aparte de crear nueva información con ello.

Utilizaremos la **API de Spotify** para gestionar el login y en base a ese login, obtener el token de acceso a la API para hacer las peticiones. El proyecto usa un **frontend** diseñado en **React** con un **backend** hecho en **.NET** que gestiona las peticiones a la API de Spotify y luego se lo manda al frontend.

Dicha web debe ser capaz de hacer los objetivos a continuación:

* Poder hacer login con tu cuenta de Spotify
* Con ese login anteriormente mencionado, poder consultar tus artistas y canciones más escuchadas en un rango de tiempo entre 4 semanas, 6 meses y toda la duración de tu cuenta.
* Poder consultar tu historial de reproducción reciente
* Obtener más información acerca de las canciones y artistas que más escuches
* Crear nuevas playlist con sugerencias de canciones en base a un artista o canción que seleccionases

1. ANÁLISIS Y DISEÑO DE LA APLICACIÓN
2. ESPECIFICACIÓN REQUISITOS

El proyecto está pensado para usarse en navegador web, lo que permite acceder a muchos tipos de dispositivos acceder a ella mientras anden conectados a la red. Al depender principalmente de una API como es Spotify, evidentemente aparte del requisito del internet, se encuentra el que los servicios de Spotify funcionen correctamente.

Además evidentemente necesita al servidor capaz de atender las peticiones y ser entendible dando igual el cliente desde el que se quiera usar.

La web dispone de dos tipos de usuario:

|  |  |
| --- | --- |
| Tipo de usuario | Usuario no registrado |
| Registro | No necesita tener tipo de registro |
| Actividad | Ver información de canciones/artistas |

|  |  |
| --- | --- |
| Tipo de usuario | Usuario registrado |
| Registro | Registro |
| Actividad | Ver información de canciones/artistas más poder comentar |

|  |  |
| --- | --- |
| Tipo de usuario | Usuario registrado con login de Spotify realizado |
| Registro | Registro con Spotify |
| Actividad | Lo anteriormente mencionado + poder consultar información de su perfil de Spotify y crear playlist en base a artista/canción |

|  |  |
| --- | --- |
| Tipo de usuario | Administrador |
| Registro | No necesita registro |
| Actividad | Se encarga de consultar las playlist creadas + los comentarios en las canciones y puede consultar artistas/canciones + estadísticas de usuarios |

1. Requerimientos funcionales

A continuación, van los requerimientos funcionales de la web:

|  |  |
| --- | --- |
| Nombre del requerimiento | Registrar usuarios |
| Características | Los usuarios deben poder registrarse para poder acceder a luego poder juntarla con la de Spotify |
| Descripción | Se le debe permitir al usuario registrarse. El usuario debe suministrar algunos datos como su nombre, apellido, nombre de usuario dentro de la propia web y su contraseña |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Nombre del requerimiento | Permitir al usuario registrado hacer el login con Spotify |
| Características | Una vez ya registrado el usuario, conectar la cuenta con la de Spotify |
| Descripción | Se le debe permitir al usuario una vez ya registrado poder juntar su cuenta con la de Spotify |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Nombre del requerimiento | Permitir al usuario registrado hacer el log out |
| Características | El usuario debe poder cerrar su sesión |
| Descripción | El usuario debe hacer click sobre Cerrar Sesión |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Nombre del requerimiento | Permitir al usuario registrado con la cuenta de Spotify desvincular la cuenta |
| Características | Se le debe permitir al usuario una vez ya registrado que haya utilizado su cuenta de Spotify, desvincularla por si quisiese cambiarse de cuenta o algún problema |
| Descripción | El usuario debe hacer click sobre Desvincular Spotify |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Nombre del requerimiento | Consultar tu perfil de Spotify, historial y tops |
| Características | Con la cuenta ya registrada, poder acceder a la visualización de la parte del perfil |
| Descripción | Una vez ya registrado con la cuenta de Spotify en su cuenta, el usuario podría visualizar su perfil (Foto, nombre, top artistas/canciones, historial de reproducción) |
| Prioridad | Alta |

|  |  |
| --- | --- |
| Nombre del requerimiento | Poder filtrar en base al tiempo dichos tops |
| Características | Dentro del perfil, poder seleccionar si escoger el rango entre 4 semanas, 6 meses y todo el tiempo de la cuenta (las opciones que permite Spotify) |
| Descripción | Se encarga de consultar las playlist creadas + los comentarios en las canciones y puede consultar artistas/canciones + estadísticas de usuarios |
| Prioridad | Media |

|  |  |
| --- | --- |
| Nombre del requerimiento | Poder consultar en mayor detalle el top tanto de artistas como canciones y filtrarlo por tiempo |
| Características | Dentro de cada apartado de ver más, poder ver un top 50 de artistas o canciones pudiendo filtrar por las opciones que permite de 4 semanas, 6 meses o todo el tiempo de la cuenta |
| Descripción | Consultar en mayor detalle tus hábitos de escucha en artistas y canciones |
| Prioridad | Media |

|  |  |
| --- | --- |
| Nombre del requerimiento | Ver tu historial de reproducción |
| Características |  |
| Descripción |  |
| Prioridad | Media |

|  |  |
| --- | --- |
| Nombre del requerimiento | Poder filtrar en base al tiempo dichos tops |
| Características | Dentro del perfil, poder seleccionar si escoger el rango entre 4 semanas, 6 meses y todo el tiempo de la cuenta (las opciones que permite Spotify) |
| Descripción | Se encarga de consultar las playlist creadas + los comentarios en las canciones y puede consultar artistas/canciones + estadísticas de usuarios |
| Prioridad | Media |

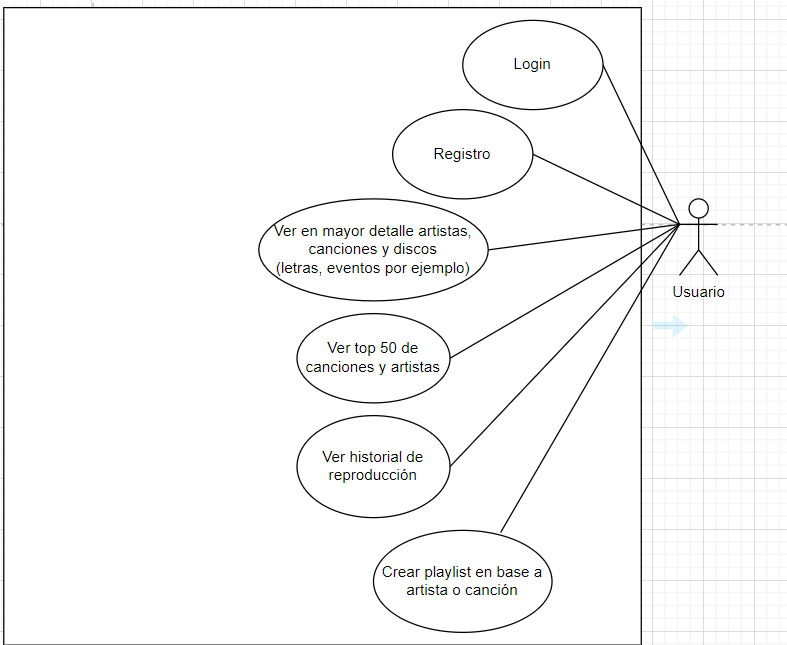
DIAGRAMA DE CASOS DE USO

Los diagramas de caso de uso exponen las acciones que pueda realizar un actor (usuario, administrador, etc) para interactuar con el sistema mediante los casos de uso, los óvalos son los propios casos de uso que son apuntadas por el actor para indicar que son las acciones que puede realizar.

Estos son los diagramas de caso de uso que utiliza mi web, en primer lugar el del usuario:

El usuario puede obviamente hacer el login y logout.

Dentro de la zona del usuario, puede ver en mayor detalle tanto las canciones como los artistas y discos en las respectivas vistas, es decir por ejemplo ver las letras en los canciones, los eventos en los artistas…), consultar un top 50 tanto de artistas como de canciones en un rango de tiempo determinado, ver el historial de reproducción del usuario (50 últimos resultados, lo máximo que nos permite la API) y crear una playlist en base a un artista (EJ: Crear una playlist en base a Judeline como artista o Tánger como canción)



En segundo lugar, estos son los casos de uso del Administrador:

Como se puede ver en la foto, el Admin puede consultar un artista, disco o canción en la zona del Admin, consultar estadísticas de la web en el panel del Admin junto al CRUD de usuarios y también ver el historial de las acciones de los usuarios (crear Playlist, consultar x canción, así esto le puede permitir tener información de poder consultar que contenidos se consumen mayormente en la web), además de obviamente al igual que el usuario hacer login y logout aunque no figura.

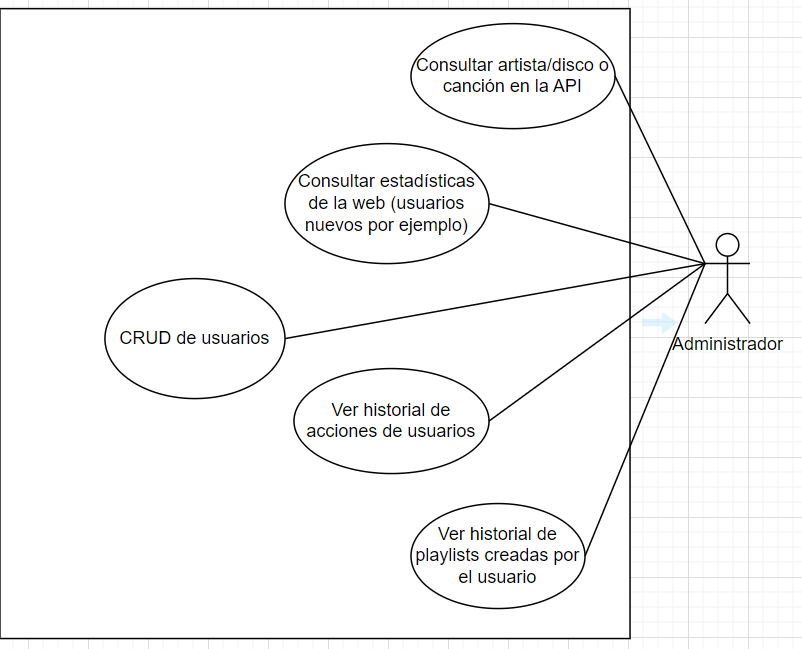


DIAGRAMA DE CLASES

ESQUEMA RELACIONAL

DIAGRAMA DE CASOS DE USO

1. TECNOLOGÍAS UTILIZADAS

En un inicio, mi plan era utilizar React para desarrollar mi frontend y otra tecnología como Spring.js o una similar para poder aprender otro tipo de tecnología que no fuese con PHP como Laravel.

Finalmente por motivos personales, las tecnologías utilizadas han sido React para el frontend mientras el Backend hace de intermediario entre las peticiones del front a él mismo y del back a la API de Spotify.

A continuación, expongo las tecnologías utilizadas:

HTML

HTML (HyperText Markup Language) es el lenguaje estándar utilizado para crear y estructurar el contenido de las páginas web. Es un lenguaje de marcado que utiliza etiquetas para definir la estructura y el significado semántico de los elementos en una página web. Estas etiquetas permiten a los desarrolladores especificar cómo se deben mostrar los elementos en un navegador web. HTML se compone de una serie de elementos que pueden contener texto, imágenes, enlaces, formularios, videos y otros elementos multimedia.

Es decir, se encarga de la estructura y el contenido de una página web.

CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir el aspecto y la presentación de un documento HTML. Con CSS, los desarrolladores pueden definir reglas que controlan el diseño, los colores, las fuentes, el espaciado y otros aspectos visuales de los elementos HTML. CSS se utiliza para separar el contenido y la estructura de una página web de su estilo y presentación. Esto permite mantener un código HTML más limpio y modular, y facilita la modificación y actualización del diseño sin tener que editar cada página individualmente.

Se encarga del diseño y presentación web de la página.

AWS

AWS (Amazon Web Services) es una plataforma de servicios en la nube que ofrece una amplia gama de servicios de computación, almacenamiento, bases de datos, análisis, redes, inteligencia artificial y más. Permite a individuos y organizaciones alojar aplicaciones y servicios en la nube, lo que proporciona escalabilidad, flexibilidad y confiabilidad. AWS ofrece una gama amplia de servicios, además es muy completa y se usa en una variedad amplia de situaciones para múltiples usos.

En términos sencillos, AWS es una colección de servicios en la nube que permiten a los desarrolladores construir y administrar aplicaciones y servicios sin necesidad de adquirir infraestructura física. Estos servicios se ofrecen a través de centros de datos distribuidos globalmente y están diseñados para ser altamente seguros, escalables y confiables.

BOOTSTRAP

Bootstrap es un popular framework de desarrollo web frontend que proporciona un conjunto de herramientas y componentes preestilizados para crear interfaces de usuario modernas y receptivas. Está basado en HTML, CSS y JavaScript, y su objetivo principal es simplificar el proceso de diseño y desarrollo de sitios web y aplicaciones web.

En resumen, Bootstrap es una biblioteca de código abierto que proporciona estilos predefinidos, plantillas y componentes reutilizables que pueden ser fácilmente integrados en proyectos web. Está diseñado para ayudar a los desarrolladores a construir sitios web rápidamente, garantizando una apariencia y un funcionamiento coherentes en diferentes dispositivos y tamaños de pantalla.

Entre sus características claves anda el sistema de rejilla adaptable para distribuir el contenido en filas y columnas, su compatibilidad con múltiples navegadores y su personalización.

En general, es ampliamente utilizado por desarrolladores web para acelerar el proceso de desarrollo, mejorar la consistencia visual y garantizar la capacidad de respuesta de los sitios web.

REACT

React es una biblioteca de JavaScript de código abierto que se utiliza para construir interfaces de usuario interactivas y reactivas. Fue desarrollada por Facebook y se ha convertido en una de las opciones más populares para el desarrollo de aplicaciones web de una sola página (SPA) y aplicaciones móviles.

La principal característica de React es su enfoque en la construcción de componentes reutilizables. Un componente en React es una pieza de interfaz de usuario que encapsula su propio estado y comportamiento. Estos componentes pueden ser combinados para formar interfaces de usuario más complejas. React utiliza una sintaxis especial llamada JSX, que permite mezclar HTML y JavaScript, para definir la estructura y la apariencia de los componentes.

React se integra bien con otras bibliotecas y frameworks de JavaScript y se puede utilizar tanto en el desarrollo de aplicaciones web como en el desarrollo de aplicaciones móviles nativas a través de herramientas como React Native.

En resumen, React es una biblioteca de JavaScript que permite construir interfaces de usuario interactivas y reactivas mediante el uso de componentes reutilizables. Su enfoque en la eficiencia y el rendimiento, junto con su comunidad activa y su ecosistema robusto, lo convierten en una elección popular para el desarrollo de aplicaciones web y móviles modernas.

ASP .NET

ASP.NET es un framework de desarrollo web creado por Microsoft que permite construir aplicaciones web escalables y de alto rendimiento utilizando el lenguaje de programación C# (C Sharp) o Visual Basic.NET. ASP.NET es parte del framework más amplio de .NET y proporciona un entorno de desarrollo robusto y poderoso para crear aplicaciones web.

ASP.NET se integra bien con otros servicios y tecnologías de Microsoft, como el uso de bases de datos SQL Server, servicios de autenticación y autorización de Azure, servicios de mensajería y mucho más.

En general, ASP.NET proporciona un conjunto completo de herramientas y funcionalidades para desarrollar aplicaciones web sólidas y de alto rendimiento. Su enfoque en la productividad del desarrollador, la seguridad y la escalabilidad lo convierten en una opción popular para construir aplicaciones web empresariales y sitios web de alto tráfico.

SQL SERVER

SQL Server es un sistema de gestión de bases de datos relacional desarrollado por Microsoft. Es una plataforma robusta y ampliamente utilizada en entornos empresariales para el almacenamiento, consulta y administración de grandes cantidades de datos.

SQL Server utiliza el lenguaje SQL (Structured Query Language) para interactuar con la base de datos y permite a los usuarios crear, modificar y consultar datos de manera eficiente. Ofrece una amplia gama de características y funcionalidades, algunas de las cuales incluyen:

* Escalabilidad: SQL Server puede manejar grandes volúmenes de datos y admite sistemas distribuidos, lo que permite un crecimiento flexible a medida que aumentan los requisitos de almacenamiento y rendimiento.
* Seguridad: Proporciona mecanismos robustos para proteger los datos, incluyendo autenticación, autorización y encriptación. También permite el control de acceso a nivel de fila y columna para proteger la confidencialidad de la información.
* Integración: Se integra estrechamente con otras herramientas y tecnologías de Microsoft, como el sistema operativo Windows, .NET Framework, Active Directory y Microsoft Azure. Esto facilita la creación de soluciones completas y permite el intercambio de datos con otras aplicaciones.
* Herramientas de administración: SQL Server proporciona una variedad de herramientas de administración, como el SQL Server Management Studio, que permiten a los administradores de bases de datos gestionar eficientemente la configuración, el rendimiento y la seguridad de la base de datos.

POSTMAN

Postman es una herramienta de colaboración y desarrollo de API que permite probar, documentar y compartir APIs de manera eficiente. Es una aplicación de escritorio y una plataforma en línea que brinda la capacidad de enviar solicitudes HTTP, simular respuestas del servidor, realizar pruebas automatizadas, crear y compartir documentación de API.

En resumen, Postman es una interfaz gráfica de usuario (GUI) que facilita la interacción con API al proporcionar una forma intuitiva de enviar y recibir solicitudes HTTP, establecer parámetros y encabezados, y examinar las respuestas del servidor. Postman permite realizar pruebas funcionales, probar diferentes escenarios, depurar problemas, automatizar tareas y monitorear el rendimiento de las API.

Postman se ha convertido en una herramienta popular y ampliamente utilizada en el desarrollo de software, especialmente en el desarrollo de aplicaciones web y móviles que interactúan con servicios web y APIs. Ayuda a agilizar el proceso de desarrollo, depuración y prueba de APIs, mejorando la eficiencia y la calidad del trabajo.

API SPOTIFY

Una API (Application Programming Interface) es un conjunto de reglas y protocolos que permite a las aplicaciones comunicarse entre sí. En el caso de la API de Spotify, proporciona una interfaz para que los desarrolladores accedan a los datos y funcionalidades de Spotify, como la búsqueda de canciones, la reproducción de música, la gestión de listas de reproducción y más.

Al utilizar la API de Spotify en el desarrollo de aplicaciones, podemos integrar la funcionalidad de Spotify en sus propias aplicaciones, ya sea para crear reproductores personalizados, acceder a información sobre artistas y canciones, crear y gestionar listas de reproducción o interactuar con las características sociales de Spotify.

La API de Spotify utiliza estándares y tecnologías web como HTTP, JSON y OAuth para la comunicación entre la aplicación y los servidores de Spotify. Esto nos permite enviar solicitudes a la API de Spotify, recibir respuestas en formato JSON y autenticar a los usuarios para acceder a sus datos y realizar acciones en su nombre.

En resumen, la API de Spotify es una tecnología utilizada en el desarrollo de aplicaciones que desean aprovechar las funcionalidades y los datos proporcionados por Spotify. Permite a los desarrolladores crear aplicaciones personalizadas que se integran con el ecosistema de Spotify y brindar una experiencia más enriquecedora y personalizada.

1. CONCLUSIONES

Valoración personal del trabajo realizado

En general, considero que el resultado final ha quedado mejor de lo que esperaba en algunas partes y quizás algo peor de lo que me gustaría en algunas, pero considero que no teniendo todo el tiempo que me hubiese gustado y habiendo aprendido por el camino nuevas tecnologías como puede ser .NET y SQL Server en vez de MySQL, aunque en el caso de SQL Server son muy similares.

Creo que ha quedado un proyecto medianamente complejo y completo en cuanto a cosas que te deja hacer y que además me ha permitido aprender otro tipo de Backend, que es algo que quería hacer para no solo conocer PHP y Laravel.

Posibles ampliaciones

Respecto a posibles ampliaciones que me hubiese gustado hacer en el proyecto, hice una lista de ellas y revisé cómo implementarlas en caso de tener tiempo suficiente. Sin embargo, debido a restricciones de tiempo o problemas imprevistos, no pude llevar a cabo todas las mejoras que tenía planificadas.

\*Revisar\*

* Sistema de compartir canciones/discos/artistas

Esta la mire y no era muy compleja, así que mi plan era hacerlo para poder compartir tus canciones, discos y artistas por Twitter mediante algún tweet, pero recientes cambios en la API de Twitter y que no quedaba claro si finalmente se haría de pago o no, me hizo descartar la idea por si acaso un tiempo de un día o dos pudiese servir al final para nada.

* + - * Sistema de amigos

Lo que tenía pensado y se habrá visto en alguno de los commits en la zona del perfil, era añadir un buscador que te permitiese buscar perfiles de otra gente y poder mandarles una solicitud de amistad, no era complejo por la parte del buscador pero si por el hecho de que suponía añadir una nueva tabla, crear todo el sistema y era mucho tiempo aparte de cierto temor de que pudiese dar errores y al final decidí descartarlo.

* Chat con tus amigos

Dependía del sistema de amigos y de hecho, en su momento al mirar lo de arriba, había comprobado si era viable y parecía viable de hacer pero como descarte el sistema de amigos, lo terminé descartando.

* Notificaciones
* Un aún mayor control de tus estadísticas

Esta si me causo rabia no poder hacerla porque me hubiese gustado bastante, pero fueron sucediendo problemas bajo la marcha. La web actualmente lo que permite es ver tu top 20 (o 50 en caso del ver más) de artistas y canciones en tu perfil en un rango de 4 semanas, 6 meses o todo el tiempo de tu cuenta que es los rangos de tiempo que te permite la API de Spotify.

Al inicio, pensaba que eran los que venían por defecto pero yo también podía seleccionar un rango de tiempo o incluso horas, por tanto mi idea general radicaba en varias funciones como poder ver cuantas veces habías escuchado x disco, artista o canción y mostrar un gráfico de cuanto en cada mes y dentro de él más específicamente en que días y horas y con ello poder añadir estadísticas como has escuchado x tiempo esta canción, esta nueva canción es de tus favoritas, como un panel con datos más secundarios tanto en discos y artistas como canciones, pero finalmente la API como tal no lo permite.

No obstante, hay ciertas formas de saltarse este “bloqueo”, de hecho habría dos que se me ocurren pero ambas son complejas y una la veo muy poco viable.

La primera es que el usuario descargue sus datos personales de Spotify que eso es un proceso que se puede solicitar en la web y Spotify da todos los datos que tiene de ti, entonces con un código podría procesar esos datos y realizar lo de arriba, pero esta la veo muy compleja ya que no tengo mucha idea de si ese código sería complejo o no y además podría haber volúmenes grandes de datos.

La segunda es solicitar al usuario permisos para vincular su cuenta a una cuenta de Last.fm y en base a la API de Last.fm, si poder tener dicho control para lo de arriba, pero esta la veo poco viable porque requiere una cuenta en otra web aparte de esta y la de Spotify, así que no lo veo viable para el usuario.

BIBLIOGRAFIA

IAS

<https://chat.openai.com/>

<https://you.com/>

.NET

<https://www.youtube.com/watch?v=MV1rEWlcW7U&ab_channel=CodAffection>

<https://www.youtube.com/watch?v=Y_qD79PLdeM&ab_channel=gavilanch2>

<https://learn.microsoft.com/es-es/dotnet/>

<https://learn.microsoft.com/en-us/visualstudio/javascript/tutorial-asp-net-core-with-react>

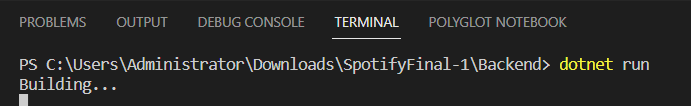
* 1. MANUAL DE USUARIO

Despliegue de la aplicación

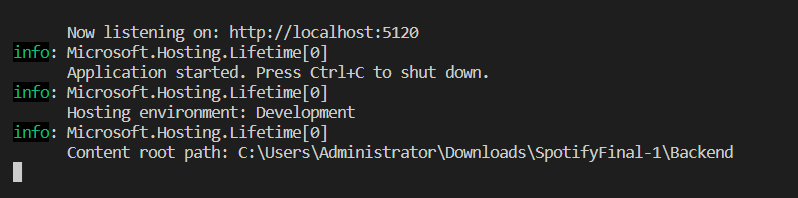
En primer lugar, el despliegue es bastante sencillo.

Sería descargar el proyecto desde el repositorio de Github (<https://github.com/Danielrg37/SpotifyFinal.git>) y abrir dos terminales, uno en la carpeta de Frontend y otro en la carpeta de Backend.

En la carpeta de Backend, se realiza este comando:

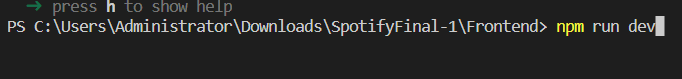


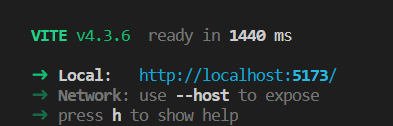
Y una vez veamos esto, es que el Backend con todas sus rutas ya está activo.



Y en la carpeta de frontend, sin más realizamos un npm install vite –force (generará muchos warns por incompatibilidades de versiones, pero no es un problema para el uso de la web) y luego un npm install por si acaso.

Por último, realizamos un npm run dev para ejecutar la web:





Usuario administrador

Usuario normal