Rave Daddy

Desarrollo de Aplicaciones Web

|  |  |
| --- | --- |
| Trabajo fin de ciclo presentado por: | Jose Mª Rubio Robles |
| Tutor/a: | Ramon Merchán Sanzano |
| Fecha: | 13 diciembre de 2025 |

**IES Juan de la Cierva**

Departamento de Informática

Memoria del proyecto

Documento técnico

Índice

[1. Introducción 3](#_Toc216132543)

[1.1. Descripción y contexto del proyecto 3](#_Toc216132544)

[1.2. Motivación del proyecto 3](#_Toc216132545)

[1.3. Beneficios esperados 3](#_Toc216132546)

[2. Objetivos 4](#_Toc216132547)

[2.1. Objetivos generales 4](#_Toc216132548)

[2.2. Objetivos específicos 5](#_Toc216132549)

[3. Contexto actual 5](#_Toc216132550)

[3.1. Estado del arte. 5](#_Toc216132551)

[3.2. Conceptos clave. 6](#_Toc216132552)

[4. Planificación 7](#_Toc216132553)

[4.1. Metodología de gestión del proyecto 7](#_Toc216132554)

[4.2. Acciones y tareas 8](#_Toc216132555)

[4.3. Temporización y secuenciación 9](#_Toc216132556)

[5. Análisis de requisitos 10](#_Toc216132557)

[5.1. Diagrama de casos de uso. 10](#_Toc216132558)

[5.2. Requisitos funcionales/negocio 10](#_Toc216132559)

[5.3. Requisitos no funcionales. 11](#_Toc216132560)

[5.4. Descripción de los usuarios y sus necesidades. 13](#_Toc216132561)

[6. Diseño de la aplicación 14](#_Toc216132562)

[6.1. Mockups o wireframes o prototipos de la interfaz gráfica de usuario. 14](#_Toc216132563)

[6.2. Arquitectura del sistema. 15](#_Toc216132564)

[6.3. Diagramas Entidad/Relación. 15](#_Toc216132565)

[6.4. Diagrama de Clases. 15](#_Toc216132566)

[7. Desarrollo de la aplicación 17](#_Toc216132567)

[7.1. Tecnologías y herramientas utilizadas 17](#_Toc216132568)

[7.2. Descripción de las principales funcionalidades 17](#_Toc216132569)

[8. Pruebas y Validación. 17](#_Toc216132570)

[8.1. Pruebas unitarias 18](#_Toc216132571)

[8.2. Pruebas de integración. 18](#_Toc216132572)

[9. Despliegue y mantenimiento . 18](#_Toc216132573)

[9.1. Despliegue de la aplicación 18](#_Toc216132574)

[10. Conclusiones y trabajo futuro. 19](#_Toc216132575)

[10.1. Trabajo futuro 19](#_Toc216132576)

[11. Relación del proyecto con los módulos del ciclo 20](#_Toc216132577)

[12. Bibliografía/Webgrafía 20](#_Toc216132578)

[13. Anexos 20](#_Toc216132579)

# Introducción



# Descripción y contexto del proyecto

Rave Daddy es una aplicación de web desarrollada con la finalidad de proporcionar a los usuarios un espacio consolidado en donde accedan a la información sobre los eventos y fiestas de música electrónica que se organizan en su entorno. Nace para solucionar la percepción muy común de la sensación de "perderse algo" en el momento de salir en la escena de la música electrónica, ofreciendo un instrumento que reporta información actualizada. También, tiene la intención de ser un punto de partida para aquellas personas que pretenden introducirse en el universo de la música electrónica, proporcionando una experiencia asequible y estructurada.

# Motivación del proyecto

La motivación principal de este proyecto nace de mi afición a la música electrónica y de mi curiosidad por el funcionamiento y las posibilidades de la inteligencia artificial. La combinación de ambos intereses dio lugar a Rave Daddy, un proyecto que me permite explorar cómo integrar la IA dentro de un entorno web orientado a la experiencia del usuario.

# Beneficios esperados

Los principales beneficios esperados son el aprender a estructurar y planificar un proyecto completo, asegurando el cumplimiento de objetivos en tiempo y forma. Además, se busca comprender el funcionamiento práctico de un agente de IA a pequeña escala y intentar entender como interactúa con las distintas tools que se le proporcionen. En conjunto, el proyecto pretende aportar tanto una mejora técnica a nivel de planificación como una mejor comprensión del proceso de desarrollo web con IA.

# Objetivos

El proyecto tiene como objetivo final desarrollar una aplicación web capaz de centralizar información sobre eventos y ofrecer un asistente basado en inteligencia artificial que interactúe con el usuario de forma natural. Para conseguirlo, se establecen una seria de objetivos generales y específicos que guiaran el desarrollo técnico del sistema.

# Objetivos generales

* Desarrollar un sistema de obtención y actualización automática de eventos y DJs usando técnicas de webscraping y ejecución programada de tareas por cron.
* Implementar un agente inteligente capaz de acceder a la base de datos, consultar información y ejecutar funciones de webscraping garantizando una interacción correcta con el usuario.
* Integrar un modelo de lenguaje (LLM) basado en Gemini para conseguir respuestas comprensibles para el usuario.
* Desarrollar un sistema de sesiones para el agente utilizando las sesiones dentro de las cookies.
* Diseñar una aplicación web funcional, sencilla e intuitiva que permita al usuario consultar eventos con rapidez.
* Utilizar como base la plataforma de Docker para empaquetar la aplicación y hacerla lo mas portable posible.



# Objetivos específicos

* Analizar e identificar los requisitos de tipo funcional y no funcional de la aplicación para orientar su desarrollo.
* Diseñar los modelos de datos que se requieren para almacenar información de eventos, DJs y consultas.
* Implementar un webscraping de Xceed de forma que se obtengan eventos, detalles y DJs y se estructure correctamente la información obtenida.
* Configurar un cron de forma que se automatice la obtención periódica de los datos obtenidos.
* Diseñar una interfaz web que sea clara y sencilla, teniendo como objetivo una experiencia de usuario clara y directa.
* Integrar el LLM Gemini con LangChain desarrollando un prompt de sistema y un agente que pueda seleccionar la herramienta adecuada para cada ejecución.
* Gestionar la memoria conversacional mediante sesiones para permitir la continuidad de la interacción con el usuario.
* Implementar el acceso a la consulta de temperatura desde API externa para ampliar la información proporcionada sobre cada evento.
* Realizar pruebas funcionales y de integración verificando el correcto funcionamiento del webscraping, del agente IA y de las APIs desarrolladas.

# Contexto actual

# Estado del arte.

Rave Daddy se sitúa en un punto intermedio entre dos tipos de soluciones ampliamente extendidas: los chatbots conversacionales y las aplicaciones de búsqueda y reserva de eventos. En el ámbito de los chatbots, existen múltiples referentes consolidados como ChatGPT o distintos copilots disponibles en el mercado, que destacan por su capacidad de interacción natural y por el uso de modelos de lenguaje avanzados. Aunque estas herramientas poseen un alto nivel de computación, no están orientadas de forma específica a la gestión de eventos musicales ni integran funciones propias de plataformas de booking.

Por otra parte, aplicaciones que giran en torno a una búsqueda y una reserva de experiencias, como Booking y similares, constituyen el modelo más análogo en el que se mezclan la consulta de información y el filtrado de contenidos. No obstante, la mayoría de estos mismos tipos de plataformas sólo ofrecen chatbots conversacionales muy restringidos, pero no con la adecuada integración con agentes que permitan ejecutar herramientas, procesar información o personalizar la búsqueda de eventos.

Hasta donde ha llegado la revisión realizada no se han podido encontrar propuestas que hagan uso de una manera u otra de forma conjunta, un sistema conversacional basado en IA y una plataforma de descubrimiento de eventos de música electrónica. La inexistencia de soluciones híbridas permite que Rave Daddy sobresalga en el sector. No es un océano azul, pero la combinación de un chatbot funcional y una aplicación de búsqueda de eventos cubren un terreno poco explorado. Además, el mercado de eventos electrónicos es un mercado muy fragmentado, puesto que la información está dispersa entre clubes, promotores y páginas independientes, lo que hace que el proyecto tenga cabida y oportunidad dentro el mercado.

# Conceptos clave.

* Inteligencia artificial generativa: Tecnología basada en modelos de lenguaje capaces de comprender y generar texto, utilizada para permitir la interacción conversacional entre el usuario y el sistema.
* Agente IA: Componente inteligente que emplea herramientas para obtener información y responder de forma contextualizada según las instrucciones del LLM.
* LLM (Large Language Model o Modelo de lenguaje de gran escala): En este caso Gemini 2.5 flash-lite. Procesa las entradas del usuario y decide que acciones debe ejecutar el agente para generar una respuesta con sentido.
* Webscraping: Técnica automatizada para extraer datos de páginas web, utilizada para conseguir información sobre los eventos y los DJs de plataformas externas.
* API REST: Interfaz de programación de aplicaciones que sigue los principios de la arquitectura de Transferencia de Estado Representacional (REST) para conectar sistemas de manera flexible y estandarizada, principalmente a través de solicitudes y respuestas HTTP.
* Sistemas de programación de tareas (Cron): herramienta en segundo plano utilizado en sistemas tipo Unix para programar y automatizar la ejecución de tareas repetitivas en momentos o intervalos específicos mediante el uso de archivos crontab.
* Base de datos relacional: es un sistema que organiza y almacena datos estructurados que se interconectan mediante relaciones predefinidas e identificadores únicos. Guardando información persistente.
* Framework: es una estructura predefinida con herramientas, librerías y convenciones que sirve de base para desarrollar software. En este caso el framework de Python, Django
* Docker: Plataforma basada en contenedores e imágenes que permite ejecutar proyectos en entornos aislados, facilitando el despliegue y la portabilidad del mismo.

# Planificación

# Metodología de gestión del proyecto

El desarrollo del proyecto se ha gestionado siguiendo una metodología ágil basada en SCRUM, adaptada al contexto de un proyecto individual. Se han realizado revisiones quincenales con el tutor, que han servido como puntos de control para valorar el progreso, detectar incidencias y redefinir prioridades.

Aunque no ha existido un equipo de trabajo, se ha mantenido una organización estructurada del proyecto mediante la división en tareas y objetivos parciales, simulando ciclos de trabajo similares a sprints.

El control de versiones del software se ha llevado a cabo mediante GitHub, utilizando un repositorio con histórico de cambios que ha permitido mantener el control del estado del proyecto, gestionar versiones estables y volver a estados anteriores en caso de errores.

La gestión de incidencias y el seguimiento del progreso se ha realizado de forma continua, ajustando los tiempos y prioridades conforme avanzaba el desarrollo.

# Acciones y tareas

El desarrollo del proyecto se ha llevado a cabo a lo largo de aproximadamente dos meses y medio. Durante las primeras semanas, el tiempo dedicado fue de entre ocho y diez horas semanales, mientras que en el último mes el tiempo llego hasta aproximadamente veinte horas semanales y con algún fin de semana más, debido a la carga de trabajo asociada a la implementación de las funcionalidades principales, las pruebas y la documentación.

Primero, se realizó una fase de toma de requisitos en la que se definieron los objetivos generales del sistema, el alcance del proyecto y las necesidades principales que debía cubrir la aplicación. Esta fase tuvo como finalidad establecer una base clara para orientar las decisiones técnicas posteriores. Posteriormente, se desarrolló la fase de análisis de requisitos, en la que se estudiaron en detalle los aspectos funcionales y no funcionales de la aplicación. Durante esta etapa se revisaron distintas alternativas técnicas y se evaluaron las tecnologías más adecuadas para su implementación. Una vez definidos los requisitos, se llevó a cabo la configuración del entorno de desarrollo y la preparación de la infraestructura mediante contenedores Docker, lo que permitió garantizar la portabilidad y homogeneidad del entorno de ejecución. La fase de diseño incluyó la definición de la arquitectura general del sistema, el diseño de los modelos de datos y la planificación de los principales flujos de interacción entre el usuario y la aplicación.

A continuación, se iniciaron los trabajos de codificación, desarrollándose la estructura base de la aplicación web desde el diseño de la misma y desarrollando los principales módulos, las vistas, los controladores y la lógica de negocio correspondiente. Tras la construcción de la base de la aplicación, se realizó la integración del agente de inteligencia artificial, con lo que se enlazó el sistema con el modelo de lenguaje y se definieron sus capacidades de acceso de base de datos y ejecución de herramientas. De forma paralela se llevó a cabo la implementación del sistema de webscraping encargado de recoger información de plataformas externas y la programación de tareas automáticas mediante cron para la actualización periódica de datos. Las pruebas se ejecutaron de forma continua a lo largo del desarrollo, incluyendo pruebas funcionales, de integración y de validación de resultados, con la finalidad de detectar y corregir errores de forma anticipada. Finalmente, la documentación del proyecto fue construida de forma progresiva desde las fases iniciales hasta su finalización, completándose a medida que se cerraban las distintas etapas del desarrollo, lo cual me ha permitido recoger de forma fiel la evolución real del proyecto.

# Temporización y secuenciación



# Análisis de requisitos

Se dividen entre:

* Requisitos funcionales (de negocio)

Funcionales: Conjunto de necesidades que cubre el proyecto en relación a los usuarios que utilizan el sistema. Es una relación de los requisitos que ha de realizar la aplicación expresados por el cliente.

* Requisitos no funcionales (requisitos técnicos, rendimiento seguridad, etc.).

No funcionales: Características de la plataforma / sistemas utilizados durante la implementación.

# Diagrama de casos de uso.

Inicialmente se suele incluir Diagrama/s de Casos de uso para tener una visión gráfica de alto nivel. De los requisitos funcionales de la aplicación y ver la interacción de los distintos actores con el sistema.

# Requisitos funcionales/negocio

Es algo que debe comprender una persona no técnica pero que entiende del negocio sobre el que se aplicará la plataforma.

El sistema se puede subdividir en subsistemas/módulos. Los requisitos suelen aparecer como **verbos en infinitivo** y suelen estar relacionados con los casos de uso (interacción usuario/sistemas con el sistema).

Ejemplo: La aplicación dispone de 3 módulos, accesibles según el perfil del usuario de la misma:

* Módulo de Autenticación:
  + **Registrar nuevo jugador**: Registrarse utilizando un correo electrónico y una contraseña para iniciar sesión.
  + **Iniciar sesión**: Usar el correo y contraseña guardados en el registro para acceder a nuestra aldea.
  + **Cerrar sesión**: Desvincular la cuenta de usuario y todo lo relacionado con ella (notificaciones, partida guardada, etc.) del dispositivo.
  + …
* Módulo de facturación de equipajes
  + …
* …

# Requisitos no funcionales.

Descripción de los requisitos Técnicos, rendimiento, seguridad, usabilidad, etc

Ejemplo de requisitos Técnicos:

* Sistema gestor de base de datos relacional/orientado a documentos, accesible desde internet
* Servidor de páginas con soporte https
* Paginas con interfaz de usuario Android / JavaFX / Swing / HTML5
* …

Ejemplos de otros requisitos no funcionales:

* Rendimiento:
  + El sistema debe ofrecer un tiempo de respuesta medio inferior a 1 segundo para las operaciones básicas
* Seguridad:
  + Se debe implementar autenticación segura para el acceso de los usuarios, evitando accesos no autenticados.
  + El sistema debe registrar todas las acciones importantes realizadas por los usuarios para asegurar la trazabilidad.
  + Los datos sensibles deben ser encriptados tanto en tránsito como en reposo.
* Usabilidad:
  + El diseño debe ser responsive, asegurando una experiencia de usuario consistente en dispositivos móviles.
  + Se debe proporcionar una guía o tutorial inicial para ayudar a los nuevos jugadores a familiarizarse con las mecánicas del juego.
  + …
* Disponibilidad:
  + El sistema debe estar disponible al menos el 99.9% del tiempo, asegurando que los usuarios puedan acceder al sistema en cualquier momento.
  + …

# Descripción de los usuarios y sus necesidades.

Tipos de perfiles de usuarios que van a utilizar la aplicación (Administrador, Gestor, Usuario…).

Sus necesidades suelen estar vinculadas con requisitos (funcionales/no funcionales).

# Diseño de la aplicación

Fase de diseño, fase previa a la implementación. Es esta fase se deben incluir aquí distintos diagramas que ofrezcan una visión de alto nivel, sin entrar en los detalles de la codificación/implementación.

* Diseño interfaz gráfica (Si es que la tiene tiene)
* Arquitectura del sistema
* Diseño de base/s de datos : Entidad relación, diseño relacional, …
* Diseño de clases: Diagrama de clases; habitualmente las de clase de persistencia; y algún apartado especialmente relevante.

Django,etc  
El motor de base de datos, base de datos con las tablas

Diagrama de clases.

# Mockups o wireframes o prototipos de la interfaz gráfica de usuario.

Capturas de pantalla / diagrama

Y enseñar el mockup (idea inicial de la pagina)

Diseños del interfaz de usuario:

* LowDifelity : MockUp en papel, Wireframes
* HighFidelity: Figma, captura del aspecto de las páginas en el navegador

*Wireframe*

Ilustración 3. Diseños del Interfaz de usuario

Muy recomendable: Un plano de navegación entre las pantallas de la aplicación (si es que el sistema tiene interfaz gráfico).

# Arquitectura del sistema.

Como funciona django. La localización de los archivos, etc.

Ademas de este framework he utilizado x api rest

# Diagramas Entidad/Relación.

# Diagrama de Clases.

Ilustración 2. Diagrama de módulos de la aplicación

# Desarrollo de la aplicación

# Tecnologías y herramientas utilizadas

Programas con su versión y librerías.

En este apartado se indican los lenguajes de programación, frameworks, Sistema/s gestores de bases de datos, etc.

# Descripción de las principales funcionalidades

Algo técnicamente interesante

En este apartado se muestran las partes del código más significativas y relevantes.

Se ilustrarán con fragmentos de código relevantes.

# Pruebas y Validación.

He hecho pruebas unitarias propias. Pruebas de integración de usuario/ bot/ api, etc

Anexar captura de las pruebas.

Descripción de los planes de pruebas que se hayan diseñado/realizado y utilizado en el proyecto.

Las siguientes secciones son las típicas de cualquier proyecto software.

Normalmente aquí se enumeran ( dentro de cada sección ) y los registros (evidencias de que se han realizado, se dejan como anexos)

* Pruebas unitarias
* Pruebas de integración (si hay varios subsistemas)
* Pruebas funcionales
* Pruebas finales de usuario

Poner la lista de funcionalidades que se han podido probar

# Pruebas unitarias

# Pruebas de integración.

# Despliegue y mantenimiento .

Docker con x estructura.

Cuando se entrega un producto final, una vez consolidado el software, los fuentes se congelan en un repositorio (tipo GitHub) y se genera la versión final compilada/con datos para su utilización en un entorno real.

# Despliegue de la aplicación

Mirar a ver si se puede desplegar.

Se explica el proceso de puesta en funcionamiento (en producción), que puede implicar , dependiendo de la naturaleza del proyecto:

* Lista de software/hardware
* Instalar servidores y frameworks base de ejecución
* Inicializar bases de datos
* Cargar datos iniciales
* Desplegar librerías de terceros
* Copiar ficheros fuentes y/o ejecutables (según lenguaje plataforma) en los directorios finales
* Dar de alta/registrar perfiles de usuarios de aplicación
* Instalar/desplegar procesos de backup, seguridad, etc.

Si se elige la opción de alojar el proyecto en un hosting, se deberá proporcionar la url para acceder al sistema en el momento de la demostración (defensa del proyecto)

# Conclusiones y trabajo futuro.

Objetivos que se han conseguido y lo que se gustaría implementaar

Apartado en el que alumno responde de una manera estructura y sintetizada sus conclusiones a la finalización del proyecto:

* ¿Se han cumplido los objetivos previstos?
* Lecciones aprendidas, limitaciones del producto, decisiones tomadas para poder cumplir los objetivos.

# Trabajo futuro

Además se presenta una relación de mejoras al proyecto si se dispusiese de más recursos: tiempo, personal, dinero…

# Relación del proyecto con los módulos del ciclo

Realizad un análisis de las distintas partes del proyecto y cómo se relacionan con módulos que habéis cursado.

Seguro que podéis encontrar relaciones con todos o casi todos los módulos.

Haced un esfuerzo, especialmente, con los módulos de los profesores que os han marcado o/y de los que os van a calificar el proyecto.

Ejemplo:

* Bases de Datos
  + Diseño de la/s bases de datos (Diagramas entidad/relación y relacional)
  + Implementación de la base de datos (DDL).
  + Sentencias de consulta y actualización (DML)
  + …

# Bibliografía/Webgrafía

* Páginas web:
  + OpenAI, “ChatGPT”, <https://chatgpt.com>

# Anexos

Toda la documentación de detalle para consulta que se estime necesaria, sin límite en su extensión.