Universidad Carlos III

Sistemas Interactivos y Ubicuos

Curso 2024-25

P2 - Implementación del prototipo

Grupo 14

24/04/2025

Paula Morales García 100472240@alumnos.uc3m.es | g82

Inés Fuai Guillén Peña 100495752@alumnos.uc3m.es | g82

Salvador Ayala Iglesias 100495832@alumnos.uc3m.es | g82

Índice

Introducción

La aplicación que hemos diseñado surge de la necesidad real que tenemos muchos estudiantes de organizarnos mejor en nuestro día a día. La idea principal es crear un tablón digital que nos permita tener en un solo sitio todas nuestras tareas, fechas de entrega, exámenes y demás cosas importantes que solemos ir apuntando en sitios distintos.

El sistema está pensado para que se pueda consultar desde cualquier lugar y dispositivo, lo que es clave teniendo en cuenta que pasamos el día moviéndonos entre casa, la universidad, la biblioteca o incluso el transporte público. Además, hemos querido que no sea solo un simple calendario o lista de tareas, sino que tenga funciones inteligentes como recordatorios según el sitio en el que estemos, escaneo de apuntes escritos a mano o incluso reconocimiento de voz para crear tareas sin tener que escribir.

El objetivo principal es facilitar la organización personal y académica, adaptándose a nuestras rutinas y ayudándonos a no olvidar nada importante sin necesidad de estar revisando constantemente la app. Al final, lo que queremos es reducir ese caos que muchas veces tenemos con tantas cosas en la cabeza y poder centrarnos mejor en lo que toca en cada momento.

Especificaciones técnicas

Arquitectura del sistema (diagramas de componentes y comunicación)

La arquitectura del sistema sigue un modelo cliente-servidor sencillo pero funcional, centrado en la gestión de usuarios, tareas y eventos mediante una interfaz web. Todos los datos se almacenan en ficheros *.json*, organizados en carpetas específicas dentro del backend. La comunicación entre cliente y servidor se realiza mediante llamadas HTTP a los distintos endpoints que exponen los servidores *Node.js.*

Los componentes principales son:

- **Frontend:** desarrollado con HTML, CSS y JavaScript. La interacción del usuario (inicio de sesión, registro, visualización de tareas/eventos) se gestiona desde aquí.

- **Backend**: dividido en módulos independientes para usuarios, eventos y tareas.
    - gestionUsuarios.js
    - gestionEventos.js
    - gestionTareas.js

    Cada uno de estos módulos se encarga de la lectura, modificación y almacenamiento de información en sus respectivos archivos *.json*, que actúan como tablas, simulando una base de datos.

        - *usuarios.json* para almacenar los objetos de los usuarios. Cada usuario estará compuesto por los campos email, nombre y contraseña. Para facilitar la consulta de los datos de los usuarios, hemos decidido no encriptar las

contraseñas, manteniendo así la simplicidad del prototipo. De implementarse una aplicación real, se debería encriptar esta información.

- *eventos.json* para guardar los objetos de los eventos de cada usuario. Cada evento está compuesto por el email del usuario al que pertenecen y una lista de eventos, donde cada elemento tiene su nombre y fecha obligatorios, y la hora (por defecto 9:00) y lugar.

- *tareas.json* para almacenar los objetos de las tareas de cada usuario. Las tareas tienen una estructura similar a la de los eventos, a diferencia de la fecha, que es opcional y un campo adicional opcional de descripción.

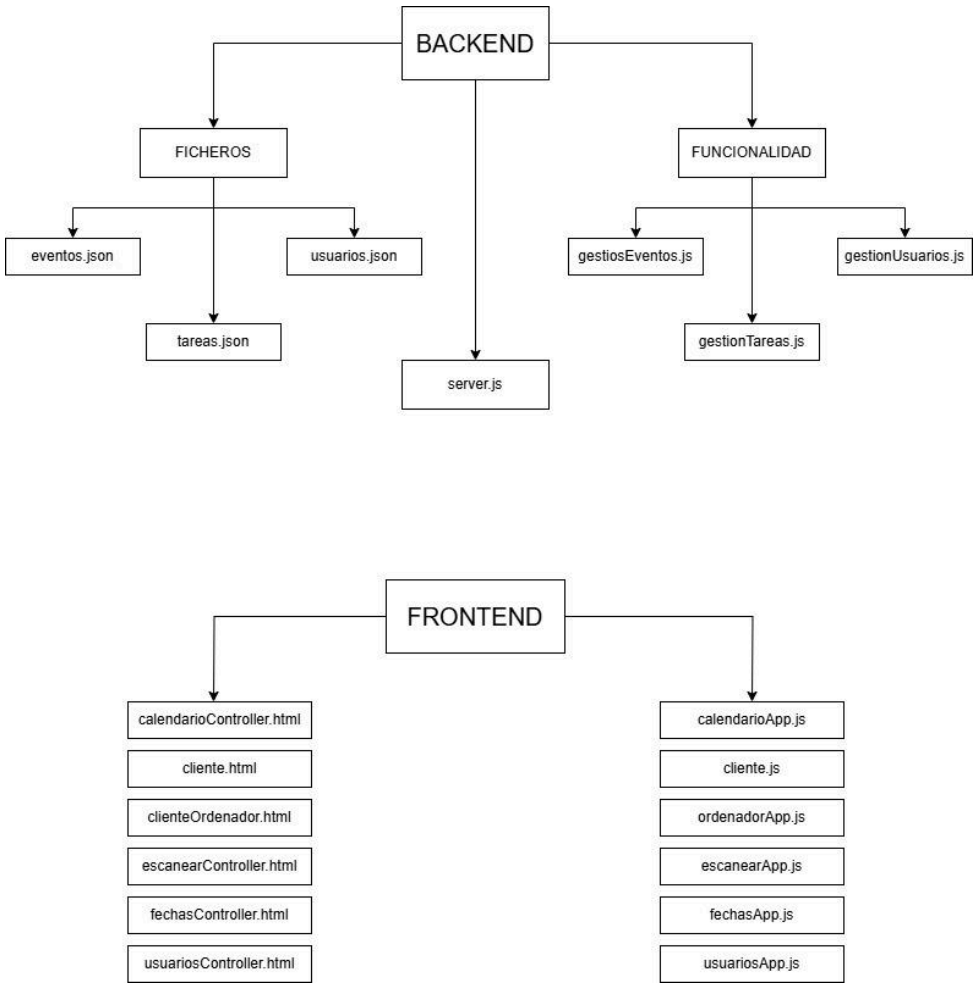

Figura 1: Diagrama de componentes del frontend y backend.

Comunicaciones: el frontend realiza peticiones HTTP al backend, el cual devuelve códigos de estado (200, 400, 401) según el resultado de la operación. Esto permite al cliente gestionar las respuestas y mostrar los resultados apropiadamente al usuario.

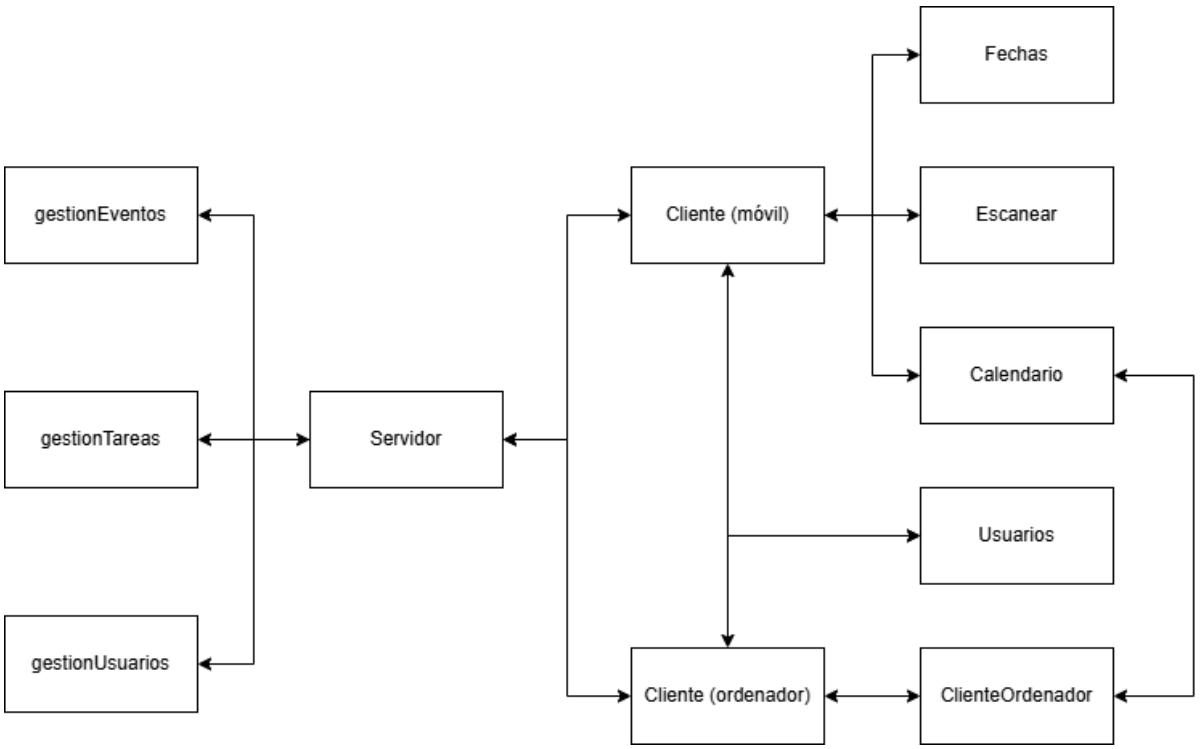

Figura 2: Diagrama de comunicaciones cliente-servidor.

Tecnologías utilizadas

Frontend:

- **HTML y CSS**: para la estructura y el estilo de las vistas.
- **JavaScript**: para gestionar la lógica del cliente, cookies, navegación entre páginas y llamadas HTTP.

Backend:

- **Node.js**: como entorno de ejecución del backend.
- **Ficheros JSON**: se utilizan en lugar de una base de datos tradicional. Esto hace que el desarrollo del prototipo sea más sencillo.

Cookies: se utilizan para guardar datos de cada sesión sin que se pierda con el cambio de archivos .html.

- **email:** guarda el email de la sesión del usuario para mandarla a todas las llamadas al servidor para que realice las operaciones sobre ese usuario.
- **nombre:** guarda el nombre del usuario para que en el menú principal, se imprima un mensaje de bienvenida personalizado con el nombre del usuario.
- **dispositivo:** indica si se está accediendo desde un ordenador o desde un móvil, para poder acceder a sus funcionalidades correspondientes.
- **fechaActual:** en el caso de consulta del mes, guarda el mes actual y en caso de consulta del día, guarda la fecha exacta. Esto servirá para poder moverse a la siguiente o anterior fecha e imprimirlo por pantalla.
- **consulta:** almacena si en el calendario se está consultando mes por mes o día por día.

- **fecha:** indica si lo que se va a guardar es un evento o una tarea. Se registra cuando se indica el comando de voz para en el fichero de guardar eventos o tareas, se imprima uno u otro en función de esta cookie.

Implementación de los prototipos

Descripción y justificación de las interacciones implementadas

En el desarrollo del prototipo hemos tenido en cuenta tanto la funcionalidad como la experiencia de usuario, tratando de simplificar al máximo el flujo de uso para que sea lo más intuitivo posible. La aplicación se basa en una interacción sencilla y directa, donde el usuario puede realizar las acciones principales (registrarse, iniciar sesión, consultar su calendario, añadir, completar o posponer tareas, etc.) sin necesidad de conocimientos técnicos.

Interacción principal: gestión de usuarios

La primera interacción que encuentra el usuario es la de iniciar sesión o registrarse. Esto se gestiona desde el archivo *UsuariosController.html*, apoyado en la lógica del backend definida en *gestionUsuarios.js*. Este paso es crucial, ya que permite personalizar la experiencia para cada usuario, garantizando que las tareas y eventos estén correctamente asociados a su cuenta.

Aquí hemos implementado tres funciones clave:
- **Registro de nuevos usuarios**: comprobando si ya existen (para evitar duplicados).
- **Inicio de sesión**: el usuario introduce su correo electrónico y contraseña y, si los datos son válidos, se le redirige a la pantalla principal. Se utilizan las cookies para mantener la sesión iniciada, lo que evita que el usuario tenga que iniciar sesión repetidamente durante su uso.
- **Eliminación de cuenta**: además implica borrar toda la información asociada (usuario, tareas y eventos), gracias a llamadas combinadas al backend.

Interacción con tareas y eventos

Una vez logueado, el usuario accede a su calendario personal. Desde aquí puede consultar, crear tanto **eventos** como **tareas**. En el backend, esto se gestiona mediante los archivos *gestionEventos.js* y *gestionTareas.js*, donde definimos métodos específicos para cada operación. Las interacciones implementadas son:

Eventos (*gestionEventos.js*):
- `nuevoEvento(email, evento)`: añade un evento nuevo si no está repetido.
- `modificarEvento(email, evento)`: actualiza un evento existente (nombre, fecha u hora).
- `borrarEvento(email, evento)`: elimina un evento concreto.
- `recuperarEventosMes(email, fechaInicio, fechaFinal)` y `recuperarEventosDia(email, fecha)`: devuelven los eventos de un mes o día determinado.

- `borrarEventosUsuario(email)`: se ejecuta cuando un usuario se da de baja para borrar todos sus eventos.

Tareas (*gestionTareas.js*):

- `nuevaTarea(email, tarea)`: añade una nueva tarea solo si no está repetida y contiene un nombre.
- `borrarTarea(email, tarea)`: borra una tarea concreta.
- `modificarTarea(email, tarea)`: permite cambiar cualquier campo de la tarea manteniendo el nombre como identificador.
- `recuperarTareasMes(email, fechaInicio, fechaFin)`: filtra las tareas que aún no completadas en un mes.
- `recuperarTareasPendientes(email)`: muestra todas las tareas que aún no han sido completadas.
- `posponerTareas(email, tarea, nuevaFecha)`: permite cambiar la fecha límite de una tarea no completada. En este caso, el cliente siempre va a posponerlo una semana.
- `completarTarea(email, tarea)`: marca la tarea como realizada.
- `borrarTareasUsuario(email)`: se ejecuta cuando un usuario se da de baja para borrar todas sus tareas.

Interacción con el calendario

A través del calendario (*calendarioApp.js*), que es la interfaz principal, el usuario puede consultar los eventos y tareas asociadas a un día o mes específico. Aquí el usuario puede:

- Consultar los días del mes actual, con acceso a las tareas y eventos por cada fecha.
- Avanzar o retroceder de mes

Cada vez que se cambia de mes, se llama al backend para recuperar sólo la información necesaria, optimizando así el rendimiento.

Interacción mediante reconocimiento de voz y gestos

Uno de los aspectos más innovadores de este prototipo es la integración de gestos y reconocimiento de voz para interactuar con el sistema de manera natural. El uso de estas tecnologías hace que la experiencia sea más accesible y adaptativa, permitiendo a los usuarios realizar acciones sin necesidad de tocar el dispositivo.

- **Reconocimiento de voz**: a través del reconocimiento de voz, el usuario puede dictar comandos para consultar eventos de un mes específico. Esta funcionalidad se activa automáticamente mediante la inclinación del dispositivo, lo que hace que la experiencia sea completamente manos libres. El sistema reconoce los comandos de voz, como por ejemplo "enero" o "febrero", para mostrar los eventos del mes indicado. También se activa por botones en el caso de guardar eventos y tareas por voz.

- **Gestos con el acelerómetro**: además del reconocimiento de voz, el prototipo también utiliza gestos basados en la inclinación del dispositivo. Dependiendo de la dirección en la que se incline el dispositivo (hacia la derecha, izquierda o atrás), se realizan distintas acciones como avanzar al siguiente día o mes, consultar eventos o

regresar al menú principal. Esta interacción con gestos hace la aplicación más dinámica y divertida de usar, a la vez que ofrece una forma alternativa de interactuar con la interfaz.

- **Gestos usando la cámara**: por último, el prototipo detecta gestos que el usuario realiza con la mano. De esta forma, al consultar el calendario, si se detecta que el índice está levantado, avanza de fecha, y en caso de que detecte que el pulgar está levantado, retrocede.

Las interacciones están diseñadas con una lógica centrada en el usuario. La separación entre frontend y backend nos permite que, en fases posteriores, podamos migrar fácilmente a un entorno móvil o usar otras tecnologías sin modificar la lógica principal. El uso de archivos *.json* como base de datos permite simular la persistencia de forma sencilla, ideal para un prototipo funcional sin necesidad de servidores complejos.

Videos que muestren el prototipo en funcionamiento

Para mostrar el funcionamiento del prototipo, hemos grabado un vídeo.

Prototipado experiencial y bodystorming

Durante la fase de desarrollo, hemos ido probando las funcionalidades y analizando su utilidad y su facilidad de uso para el usuario.

Descripción de las sesiones realizadas

Se realizó una sesión principal de bodystorming, en la que simulamos el escenario de un estudiante haciendo uso de la aplicación. Sin embargo, a lo largo de todo el desarrollo se fueron probando las funcionalidades y las interacciones entre ellas. De esta manera, se fue iterando para encontrar problemas y nuevas ideas.

Se probó el reconocimiento de voz, el uso del giroscopio como control principal y el uso de gestos para ciertas funcionalidades. En todas ellas, determinamos que eran realmente útiles para el usuario y que eran sencillas de usar.

Reflexión sobre cómo influyeron en la implementación del prototipo

La sesión de bodystorming fue realmente útil para pensar nuevas funcionalidades que facilitarán el uso de la aplicación, como el escaneo de notas o el control del calendario por gestos. Por otro lado, iterar sobre las funcionalidades, comprobando su utilidad y facilidad de uso nos ha permitido corregir problemas e idear alternativas más cómodas para el usuario.

Descubrimos, además, que el uso de la transcripción de voz a texto es realmente sencillo de usar y funciona perfectamente, por lo que nos permite anotar rápidamente todos los datos que necesitamos.

Videos de bodystorming y análisis de los hallazgos obtenidos.

Además, hemos grabado un vídeo que contiene una serie de casos en los que se hace uso del prototipo.

Iteraciones y mejoras

A medida que se iban desarrollando funcionalidades, se iba comprobando su utilidad, de manera iterativa. Gracias a ello, se han encontrado los siguientes problemas e ideas.

Problemas encontrados y cómo se abordaron

Se encontraron errores en el código que se corrigieron rápidamente.

Por otro lado, vimos que la manera de controlar el calendario con el giroscopio del móvil en ocasiones era incómoda, ya que requería tener una mano ocupada sosteniendo el móvil. A raíz de esto, pensamos en control por gestos, lo cual nos permitiría tener las manos libres y, en caso de necesitarlo, hacer un gesto rápido para controlar la aplicación. Esto es especialmente útil para un estudiante que está estudiando, generalmente escribiendo, y no quiere tener que coger el móvil ni usar ningún dispositivo para controlar la aplicación.

El reconocimiento de voz nos dio problemas a la hora de reconocer fechas y horas, ya que reconocía el número en su forma escrita y no en su forma con dígitos. Para solventar este problema, hicimos una función con un diccionario que transcribiera cada número a su forma en dígitos, pudiendo reconocer más fácilmente las horas y fechas de las tareas y eventos. En el caso de los años, hay veces que detecta el año en dígito y otras veces como caracteres. No hemos sabido muy bien cómo solventar esto ya que un año contiene más de una palabra, por lo que en caso de que detecte el año como caracteres, hay que repetir la fecha hasta que la detecte como dígito. En el caso de las horas, los minutos deberán de ser palabras simples, es decir, del 1 al 30, 40 y 50.

Por otro lado, en la consulta del calendario, solo hemos implementado que se diga el mes y no el año, por lo que las fechas a consultar solo podrán ser de 2025.

Cambios realizados respecto a las ideas originales

Inicialmente, se pensó en la posibilidad de modificar y borrar tareas. Sin embargo, en nuestro prototipo, esto no ha llegado a implementarse. En el backend las funciones existen, pero el cliente nunca llega a implementarlas. En caso de ser necesario, se podría implementar rápidamente, pero es una idea que ha quedado descartada de cara al prototipo.

Además, pensamos en darle al calendario del mes completo un formato de tabla (como si de un calendario real se tratase). Sin embargo, debido a complicaciones con el apartado visual, se decidió dejar para futuras implementaciones, ya que para el prototipo no era necesario.

Nuevas ideas generadas

La principal idea generada a partir de las iteraciones fue el escaneo de tareas escritas en notas de papel o pizarras. Descubrimos que, muchas veces, tendemos a planificar nuestras tareas sobre una pizarra o un cuaderno, ya que es lo que más nos pilla a mano en ese momento y, además, nos permite borrar y reescribir rápidamente, agilizando el proceso.

Para integrar este descubrimiento en nuestra aplicación, decidimos que sería buena idea integrar una API de reconocimiento de texto en imágenes que nos transcribiera dichas tareas y eventos una vez organizadas. A pesar de que la API falla en ocasiones, hemos encontrado esta funcionalidad realmente útil.

Reflexión final

Este trabajo nos ha permitido desarrollar una comprensión más profunda sobre el proceso de creación y gestión de tareas dentro de un sistema. A través de las diferentes fases del proyecto, hemos podido experimentar cómo cada decisión impacta tanto en la experiencia del usuario como en la parte técnica.

Uno de los puntos clave fue el uso de la gestión de tareas como concepto central del sistema. Trabajar con archivos JSON para almacenar y modificar datos nos permitió mejorar nuestras habilidades en la manipulación de datos en formato JSON y gestionar archivos de forma eficiente en un entorno Node.js. Además, nos dimos cuenta de la importancia de tener una estructura de datos clara y bien organizada, que es esencial para que el sistema funcione correctamente y sea escalable.

También fue muy importante tener en cuenta la experiencia del usuario en todo momento. Implementar las diferentes funciones no solo se trataba de lograr que el código funcionara, sino de hacerlo de manera que fuera fácil de usar y entendible para el usuario final.