A close-up of a logo

Description automatically generated

**Subastas DASimios**

**Sprint 3**

**Ulises Díez Santaolalla**

**Teresa Franco Corzo**

**Ignacio Felices Vera**

**Grupo A**

**Desarrollo de Aplicaciones y Servicios**

**3º Grado en Ingeniería Matemática e Inteligencia Artificial**

# Contenido

[Contenido 2](#_Toc186558594)

[1. Objetivos previstos 3](#_Toc1030930656)

[2. Tareas que realizar 4](#_Toc1045392437)

[3. Distribución del trabajo 10](#_Toc2059072859)

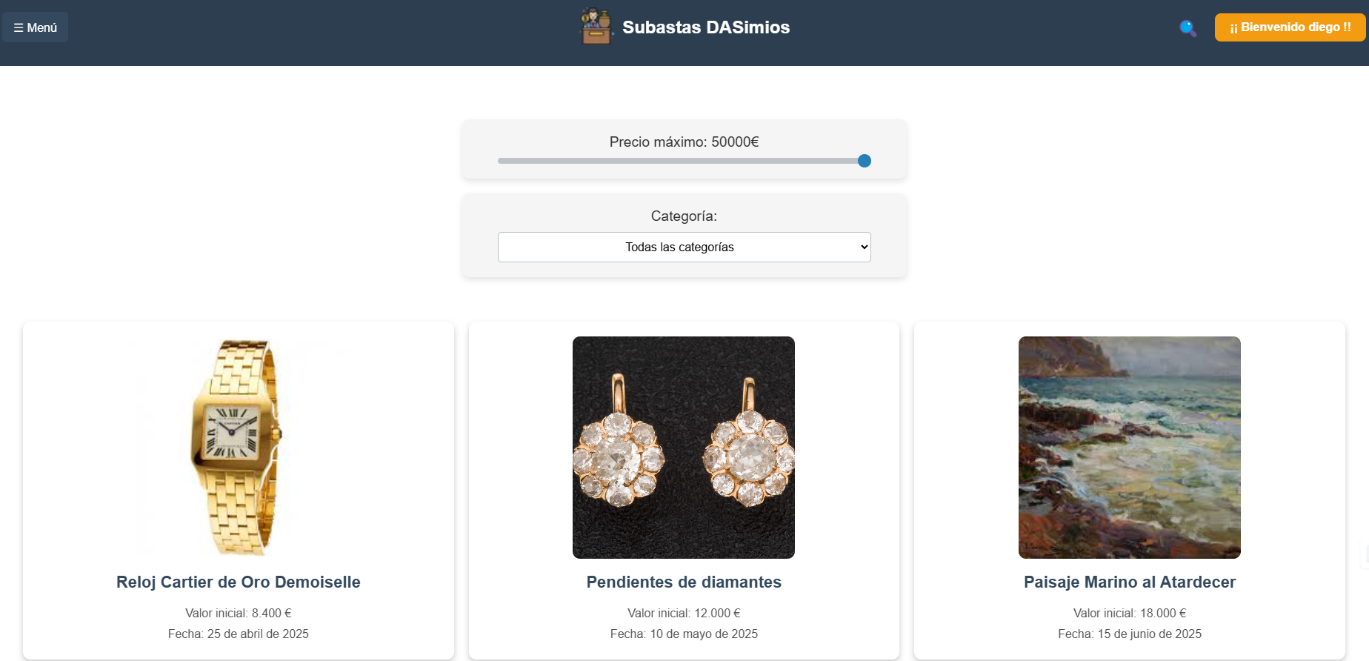
# Objetivos previstos

En esta práctica, nuestro objetivo es migrar el Sprint anterior, desarrollado en HTML, CSS y JavaScript, a React con el framework Next.js. Esta transición nos permitirá mejorar la estructura del proyecto, optimizando la reutilización de código y facilitando el mantenimiento.

Para ello, transformaremos elementos esenciales como el Header y el Footer en componentes reutilizables para que aparezcan en todas las páginas. También trasladaremos toda la funcionalidad y dinamismo a JSX, permitiendo una mejor integración y eficiencia en el desarrollo.

A lo largo del proceso, organizaremos las tareas en fases, desde la planificación hasta la implementación de funcionalidades clave, como filtros en la pantalla de subastas, autenticación de usuarios y personalización de la interfaz. Finalmente, desplegaremos el proyecto en Vercel, asegurando su disponibilidad online.

Esta migración representa un avance importante en la evolución del proyecto, haciéndolo más escalable, eficiente y fácil de gestionar.



# Tareas que realizar

Fase 1: Creación de un nuevo Milestone y definir historias de usuario

Creamos un Milestone que se llama Sprint2-Versión REACT que contiene 4 historias de usuario que tienen la intención de abarcar un poco todo el contenido que tenemos que modificar y crear. En un principio asignamos estas tareas arbitrariamente, pero esto fue cambiando a medida que nos involucrábamos en el proyecto.

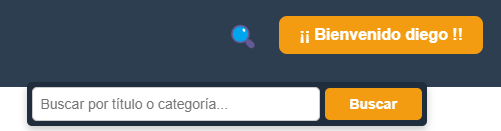
* La primera historia de usuario describe la necesidad de tener el Header y Footer definidos en el layout para toda la navegación en la web.
* La segunda describe los filtros sobre la pantalla de subasta para categorizar de forma más rápida los intereses del cliente y facilitarle la búsqueda en la web.
* La tercera y cuartan indica la personalización a través del uso de API backend para lograr identificar el usuario después de que se registre e inicie sesión, con el respectivo cambio en los botones de la navegación. También indican la necesitada de un apantalla del detalle de usuario una vez inicie sesión.

Fase 2: Crear la app de NextJS

Para crear la app nos descargamos Node.js y realizamos las configuraciones necesarias para comenzar una web con este framework. Guardamos nuestro Sprint 1 en una carpeta “estático” para no perder todos los desarrollos realizados. Preparamos la página principal y el layout global, realizamos los componentes de *Header* y *Footer* con sus propios estilos CSS. También coincidimos que queríamos realizar un componente principal *ListaSubastas* para que si se cargasen en distintos subdominios estuviesen con el mismo estilo reciclado.

Fase 3: Layout del pie de página y header

Para crear el Header, hemos reciclado el estilo, pero hemos cambiado la distribución de los elementos con grid y flex para que sean homogéneos en toda la web. Para el despliegue de la barra de navegación hemos modificado que siempre que esta se despliegue y se busque, independientemente de la página en la que se esté, se dirija al subdominio de “subastas” donde aparecerán filtradas por la barra de navegación. El botón de menú de momento sigue teniendo dos posibles rutas, la de Inicio y la de Contacto, extensión funcional para los grupos de 3. Por último, el botón de Inicio de Sesión que, al pulsarlo, da la opción de registrarse si no se posee una cuenta. Para el cambio de este botón al autenticar al usuario se habla más adelante.



Fase 4: Filtros sobre la pantalla de subastas y Fase 5: Apartado para grupos de 3 e implementación del id de las subastas

A screenshot of a computer

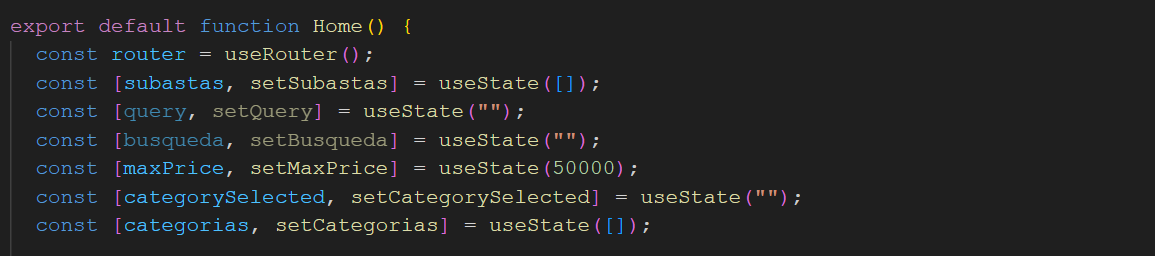
AI-generated content may be incorrect.Primero de todo se tuvo que dar la migración del codigo de HTML y JavaScript a jsonx. Esta tarea no fue sencilla dado que para la página del Registro se implementaba mucho codigo de javascript para verificar muchos requisitos como puede ser la edad del usuario junto con los requisitos de cada campo.

Un aspecto en el que tuve que implicar más tiempo fue en el CSS, ya que al migrarlo cambió mucho, y tuve que darme cuenta sobretodo del arquitectura display: flex con la que se estaba tratando, y poder crear dentro de la columna un mismo container aparte.

A screenshot of a computer

AI-generated content may be incorrect.

En la pantalla de listado de subastas, se han implementado filtros de búsqueda que permiten a los usuarios seleccionar un precio máximo y una categoría específica para visualizar únicamente las subastas que cumplen con estos criterios. Para ello, se ha utilizado React con Next.js, gestionando el estado con el hook useState y actualizando la URL mediante useRouter.

En esta implementación, se han utilizado los hooks useState y useEffect de React para gestionar el estado de los filtros y la lista de subastas, así como para actualizar dinámicamente la interfaz de usuario en función de los valores seleccionados. Para ello hemos creado las siguientes variables donde cada una de estas se inicializa con un valor por defecto y se actualiza dinámicamente en respuesta a las interacciones del usuario.

Uso de useEffect para manejar efectos secundarios.

Se han utilizado varios useEffect para diferentes propósitos:

a) Obtener los parámetros de la URL al cargar la página

Cuando el usuario accede a la página, los filtros pueden estar predefinidos en la URL. Para reflejar estos valores en los controles de la interfaz, se usa un useEffect que se ejecuta solo una vez al cargar la página (debido a la dependencia []):

Se obtiene el valor de maxPrice y categorySelected de los parámetros de la URL.

Si existen valores en la URL, se actualizan los estados correspondientes con setMaxPrice y setCategorySelected.

Este mecanismo permite que los filtros se mantengan consistentes incluso si el usuario recarga la página.

A computer code on a black background

AI-generated content may be incorrect.

b) Cargar la lista de subastas y extraer las categorías disponibles

Otro useEffect se encarga de obtener los datos de las subastas desde un archivo JSON externo. Una vez que se cargan los datos:

Se almacena la lista de subastas en el estado subastas.

Se extraen las categorías únicas de las subastas disponibles y se almacenan en categorias para poblar el menú desplegable de selección de categorías.

Este efecto se ejecuta una sola vez al cargar la página.

c) Actualizar la URL con los filtros seleccionados

A close-up of a number

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.Cada vez que el usuario cambia el precio máximo (maxPrice) o la categoría (categorySelected), se actualiza la URL con los valores seleccionados. Esto se logra mediante otro useEffect que observa los cambios en estas dos variables y modifica la URL con router.push. Es predeterminado es maxPrice=5000, y al no haber seleccionado

ningúna categoría no aparece.

El uso de router.push con la opción { shallow: true } permite actualizar la URL sin recargar la página, mejorando la experiencia del usuario y permitiendo compartir la URL con filtros aplicados.

Filtrado de subastas según los filtros seleccionados

La lista de subastas mostrada en pantalla se filtra dinámicamente en función de los valores de maxPrice y categorySelected. Esto se realiza mediante el método filter, que recorre la lista de subastas y devuelve solo aquellas que cumplen con los siguientes criterios:

Su precio (valor) es menor o igual al maxPrice seleccionado.

Su categoría coincide con categorySelected, a menos que se haya seleccionado "Todas las categorías".

Este filtrado ocurre en tiempo real y se aplica antes de renderizar la lista de subastas.

4. Interacción del Usuario con los Controles de Filtro

Se han implementado dos controles interactivos:

Un control deslizante (input type="range") para seleccionar el precio máximo, que actualiza maxPrice con cada cambio.

Un menú desplegable (select) que permite elegir una categoría de la lista, actualizando categorySelected.

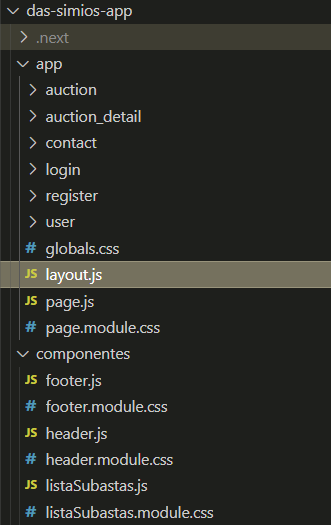
Ambos controles están vinculados a su respectivo estado con onChange, lo que permite que la interfaz reaccione de manera inmediata a las selecciones del usuario.

A screenshot of a computer

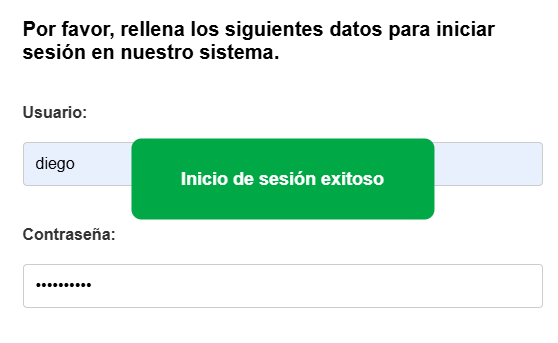
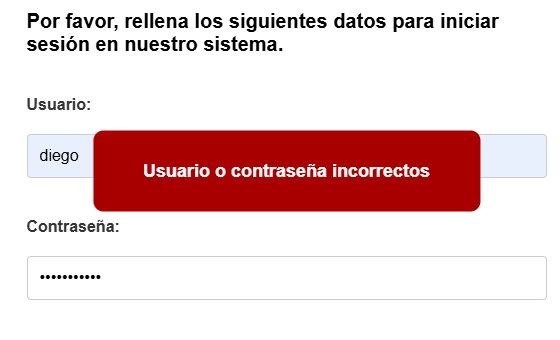
AI-generated content may be incorrect.

Para terminar, el uso de useState y useEffect ha permitido implementar filtros dinámicos y eficientes para la pantalla de listado de subastas. La actualización automática de la URL, junto con el filtrado en tiempo real, mejora la experiencia del usuario y facilita la navegación manteniendo los filtros aplicados.

Fase 6: Autenticación del usurario, registro y login

En este apartado se llevó, en primer lugar y al igual que en el resto de apartado, la página del modo estático al modo dinámico, donde se crearon los respectivos componentes de cada apartado de la página y se cargaron en la página principal. Para el trabajo desde Next.js, la página principal donde se cargan el header y footer de manera predefinida, para después ir cargando de manera dinámica el contenido del medio, se sitúan en la carpeta “componentes”, mientras que la estructura de cada subpágina web se carga desde la carpeta “app.

Una vez realizado toda la migración a dinámico, se trabajó con el backend para que los datos de registro y login se enviasen y cargasen a través del mismo. Para esto, en el caso del registro, se modificaron los requisitos del formulario para que cuadrasen con el del backend, como solicitar una contraseña alfanumérica y que fuese de longitud mayor a 8. Todos los datos del formulario se envían a “[https://das-p2-backend.onrender.com/api/users/register/”](https://das-p2-backend.onrender.com/api/users/register/) a través de un método Post, mientras que en la página de login se realiza una petición Get con los datos de usuario y contraseña. Además de esto, se introdujo la lógica para mostrar un mensaje de éxito o no dependiendo de si ha sido exitosa la petición al backend, con un mensaje de cargar mientras que se recibe la respuesta del servidor, para informar al usuario de que sea paciente. En caso de éxito, se redirige al usuario a la página principal, mientras que, en caso de no funcionar correctamente, se personaliza el mensaje para indicar al usuario en que ha fallado.



Fase 7: Personalización y control de botones

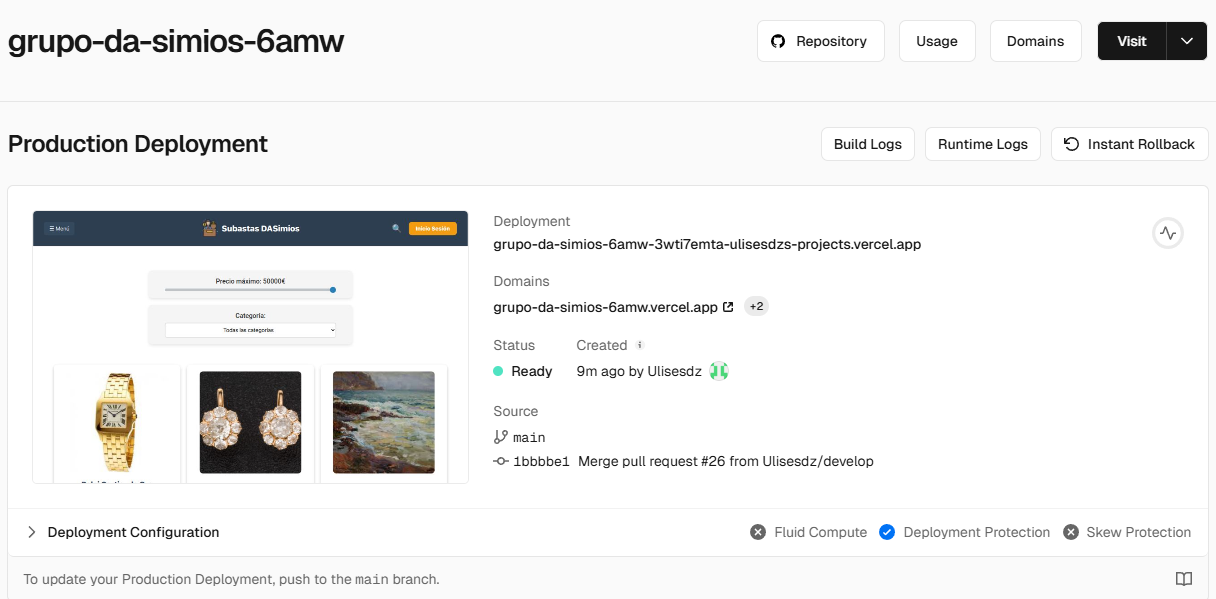
Para esta adaptación se realiza como en la anterior Fase, la llamada GET a la api para obtener los datos de registro del login y se disponen en un subdominio “user” que contiene el código preparado para en el futuro, implementar la posible modificación de los campos. En el header, hemos creado un evento global que avise a este de modificarse de manera que en vez de disponer “Inicio Sesión” implemente “Bienvenido {usuario}” con el nombre del usuario que ha iniciado sesión. Para ello usamos un control ternario sobre el evento para disponer un mensaje u otro en el botón de la web.

Fase 8: Despliegue en Vercel

Para poder ver el despliegue de nuestra página web en acción, sin trabajar de manera local como se ha hecho durante el desarrollo con el comando “npm run dev”, se trabajó con Vercel al igual que en el Sprint 1.

En primer lugar, se hizo el merge de la rama feature/P2-react a la rama develop, para posteriormente y tras la correción de la estructura, se realizó el merge final de la rama develop a la rama main, con el nombre del commit con la fecha de entrega del producto final. De esta manera, y con todo el proyecto en la rama princiapl, desde vercel se pudo desplegar un proyecto de Next.js con la carpeta de la página web dinámica, pudiendo ver el resultado final con el enlace de a continuación:

<https://grupo-da-simios-zlt9.vercel.app/?maxPrice=50000>



# Distribución del trabajo

Para todo ello, hemos ido adaptándonos poco a poco, creando los subdominios en la carpeta app. En este caso hemos creado una única rama *feature/P2-react* en la que hemos sabido trabajar sin pisar los trabajos de los otros y sin tener conflictos de merge.

|  |  |  |  |
| --- | --- | --- | --- |
| Fases | Ulises Díez | Ignacio Felices | Teresa Franco |
| Fase 1 | Reparto de historias. | Reparto de historias. | Despliegue del milestone y creación de las historia de usuario. |
| Fase 2 | Organización de la app en Github. | Organización de la app en Github. | Organización de la app en Github. |
| Fase 3 | Realiza Footer. | Realizar layout. | Realiza header y ListaSubastas. |
| Fase 4 | Implementa login y registro. | Implementa los filtros de subastas. | Implementa página principal y barra de búsqueda. |
| Fase 5 |  | Implementa detalle de subastas con id, apartado de 3 usando los parámetros de filtro. |  |
| Fase 6 | Llamada GET al backend desde login y usuario, llamada POST al backend desde registro. |  |  |
| Fase 7 |  |  | Llamada GET al backend, campos modificables, y botón de bienvenida. |
| Fase 8 | Despliegue en vercel y unificación en Github. | Ultimas actualizaciones y despliegue en vercel y unificación en Github | Ultimas actualizaciones. |