GENERADOR DE MEMES

Manual técnico

Desarrolladores
Enrique Querini
JAVIER ACOSTA

Tabla de contenido

Introducción

En este manual se detallan los aspectos técnicos fundamentales para el óptimo funcionamiento y comprensión de la aplicación Generador de Memes. Se describen las herramientas y software utilizados, así como la arquitectura del proyecto.

Generador de Memes es una aplicación web que puede ejecutarse en cualquier navegador sin necesidad de herramientas externas, solo requiere conexión a internet.

El proyecto está estructurado en una arquitectura que incluye dos contextos para gestionar los cambios en tiempo real en todos los componentes y entre los contextos.

La aplicación está desarrollada en el framework Next.js, que simplifica los métodos de renderizado. Las imágenes son proporcionadas por Imgflip y la interfaz de usuario está completamente diseñada con Tailwind CSS.

Generador de memes

Generador de memes es una aplicación web desarrollada con el propósito de fomentar un sitio donde las personas puedan dar rienda suelta a su creatividad en la creación de sus propios memes.

Objetivos

a) General

Desarrollar una aplicación que permita a los usuarios cargar imágenes, agregar texto personalizado encima de ellas y guardarlas como memes. La aplicación proporcionará una interfaz intuitiva y fácil de usar para que los usuarios puedan crear memes de manera rápida y sencilla.

b) Específicos

1. Implementar un sistema de carga de imágenes que permita a los usuarios seleccionar y cargar imágenes desde sus dispositivos o a través de URL externas de forma rápida y sencilla.
2. Desarrollar un editor de texto intuitivo que permita a los usuarios agregar y personalizar texto encima de las imágenes cargadas, incluyendo opciones para cambiar el tamaño, la fuente y el color del texto.
3. Crear un sistema de guardado de memes que permita a los usuarios guardar las imágenes editadas como memes en sus dispositivos, asegurando que el proceso sea fácil y rápido.
4. Diseñar una interfaz de usuario intuitiva y fácil de usar que permita a los usuarios realizar todas las acciones de manera rápida y sencilla, garantizando una experiencia fluida y satisfactoria.
5. Ofrecer opciones de personalización adicionales, como la capacidad de añadir stickers o filtros a las imágenes, para que los usuarios puedan crear memes únicos y creativos.

1. Integrar una variedad de plantillas de memes que los usuarios puedan utilizar como base para crear sus propios memes, proporcionando una selección diversa y atractiva.
2. Desarrollar la aplicación para que sea compatible con múltiples plataformas, incluyendo web, para que los usuarios puedan acceder y utilizar la aplicación desde diferentes dispositivos.
3. Implementar funcionalidades de optimización de imágenes, como la carga perezosa, que permitan mejorar el rendimiento de la aplicación y la experiencia del usuario al cargar y visualizar las imágenes de manera eficiente.

Alcance

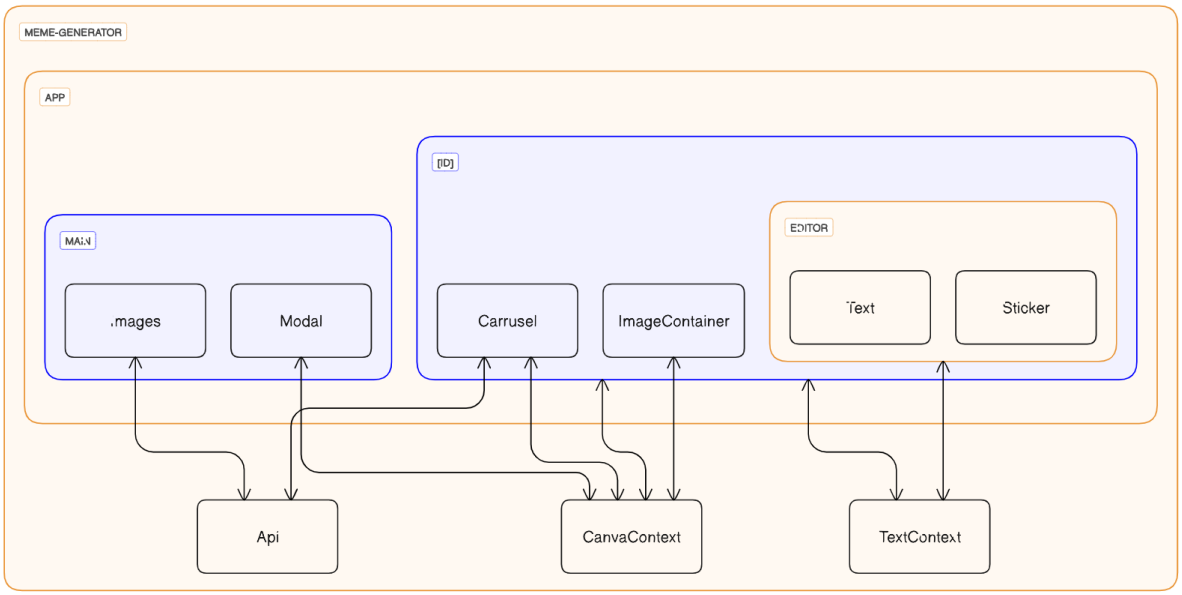
El alcance de la aplicación incluiría las siguientes funcionalidades:

1. **Carga de imágenes**: Los usuarios podrán cargar imágenes desde sus dispositivos o desde fuentes externas como URL.
2. **Edición de texto**: Los usuarios podrán agregar texto personalizado encima de las imágenes cargadas. Esto podría incluir la capacidad de cambiar el tamaño, la fuente y el color de texto.
3. **Guardado de memes**: Los usuarios podrán guardar las imágenes editadas como memes en sus dispositivos.
4. **Interfaz intuitiva:** La aplicación contará con una interfaz intuitiva y fácil de usar, que permita a los usuarios realizar todas las acciones de manera rápida y sencilla.
5. **Personalización:** Se podrían incluir opciones de personalización adicionales, como la capacidad de añadir sticker o filtros a las imágenes.
6. **Plantillas de memes:** La aplicación proporcionará una selección de imágenes que los usuarios podrán utilizar para crear sus propios memes.
7. **Compatibilidad multiplataforma:** La aplicación estará disponible para plataformas web solamente.
8. **Optimización de imágenes:** La aplicación podría incluir funcionalidades para optimizar las imágenes de carga, como la carga perezosa, que permitiría cargar las imágenes de manera diferida para mejorar el rendimiento de la aplicación y la experiencia del usuario.

Arquitectura del software

El Generador de Memes se basa en una arquitectura de software escalable y segura que se ejecuta en la nube con Vercel. El frontend, construido con Next.js y React, ofrece una interfaz moderna y receptiva para la creación de memes. Los usuarios pueden seleccionar una plantilla prediseñada de Imgflip o cargar una imagen personalizada, escribir y personalizar el texto del meme.

La arquitectura del software demuestra cómo los contextos (Text y Canva) mantienen el flujo de los estados de la aplicación, asegurando una experiencia de usuario fluida y coherente.



CanvaContext

CanvaContext es el proveedor encargado de gestionar toda la información relacionada con la imagen en proceso de edición en la aplicación. Este contexto se encarga de modificar los estados de la imagen y establece un flujo de datos con imágenes y carruseles.

- Estados
    - Modal

      Recibe como parámetros un booleano, que si recibes **true**, muestra un modal que te deja ingresar las imágenes o la url para carga la imagen.

    - Boxes

      Es un arreglo que tiene como valor inicial uno vacio, recibe las palabras y sticker, como salida nos muestra los sticker y las palabras agregadas.

    - imageRef

      Es una referencia de la imagen, que sirve para alterar la imagen en sí.

    - CanvasRef

      Es una referencia al cuadro que contiene la imagen, sirve para que no se salga los sticker y las palabras.

- Funcionales
    - handleStickerOnMouseDown, handleOnMouseDown, handleOnTouchStart

      Recibe el evento del click o del touch del móvil y retorna el movimiendo de las imágenes o sticker, lo que da como resultado el efecto de drag and drop de los sticker o imágenes.

    - resetEdit, resetFilter

      Vuelve sus estados originales todos los filtros que se aplicaron en la aplicación.

TextContext

TextContext es el proveedor encargado de gestionar la información de los inputs de la aplicación y de los stickers. Este contexto se encarga de todos los cambios relacionados con el texto o los componentes que lo manejan.

- Estados
    - StyleText

      Maneja los estados del texto.

    - prevTextRef, Sticker Selected

      Recibe la referencia del objeto y de ahí extrae todos los valores de dicho contenedor

- Funcionalidades
    - updateStyleText

      Se pasa la propiedad del texto y el valor de este, devuelve la actualización del valor de la propiedad

    - resetTextState, resetTextInput

      Recibe el evento del click que hace que se limpien todos los campos y que vuelvan a su estado original

API

La API es responsable de proporcionar las plantillas de memes a la aplicación. Además, se ha creado un servicio con la data del API que permite buscar por ID y proporciona funcionalidades de paginación para las peticiones.

- getMemeImages

  Se hace una petición a la api y nos trae las imágenes.

- getMemeImageById

  Se le pasa como parámetro el id del producto, y nos devuelve la imagen con el id correspondiente

Los demás componentes de la aplicación tienen la función de manejar la data suministrada por los contextos.

EditPage

- Carrusel
    - **Recibe** el id de la imagen
    - **Retorna** el carrousel con los imágenes de memes
- Editor
    - Sticker
        - **Recibe** las propiedades de editar y eliminar sticker
        - **Retorna** los cambios hecho en el sticker o lo elimina
    - Texto
        - **Recibe** las propiedades de agregar texto, agregar texto al input, eliminar y clonar
        - **Retorna** los valores del texto al provider
- Header
    - **Recibe** el id y los memes
    - **Retorna** retorna dependiendo de lo que se haga, ya que puede devolver un sticker que se seleccione, o cambia el meme con el carrousel
- Image
    - ImageContainer({loading})
        - **Recibe** el loading que es lo que viene del index, que es un estado para saber sobre la carga de la imagen
        - **Retorna** la imagen que se observa
- Index
  EditContainr({id})
    - **Recibe** como paramatro el id de la imagen
    - **Retorna** la imagen, junto con los widges para editar la imagen
    - handleSelectElement
        - **Recibe** el elemento selecciona cuando le das click a un texto o un sticker
        - **Retorna** que su zona aparezca de otro color y la que no estan seleccionada, se quede en blanco y parezca al resto
    - memeImageById
        - **Recibe** el id del meme que se envio anteriormente
        - **Retorna** si el id es una url, lo deja como este, si es un id, lo busca en la api para retornarla

Roles del usuario del sistema

En el sistema del Generador de Memes, el único rol de usuario que interactúa con la aplicación es el "Usuario Creador de Memes". Este usuario tiene la capacidad de cargar imágenes, agregar texto personalizado encima de ellas para crear memes, personalizar los memes con stickers o filtros, seleccionar plantillas predefinidas, y guardar los memes creados en su dispositivo. La interfaz está diseñada para que este proceso sea intuitivo y fácil de usar, fomentando la creatividad de los usuarios.

Tecnologías utilizadas

En la aplicación se usaron las siguientes tecnologías:

1. **Next.js 14.1.4**: Utilizado como framework de desarrollo web.
2. **React 18**: Biblioteca de JavaScript para construir interfaces de usuario.
3. **React DOM 18**: Paquete para manipular el árbol DOM en aplicaciones React.
4. **@headlessui/react 1.7.18**: Componentes accesibles para React, utilizados probablemente para la interfaz de usuario.
5. **@heroicons/react 2.1.3**: Iconos SVG personalizables para React, posiblemente utilizados en la interfaz de usuario.
6. **clsx 2.1.0**: Utilizado para generar listas de clases de manera condicional en componentes de React.
7. **downloadjs 1.4.7**: Utilizado para generar descargas de archivos desde el navegador.
8. **html-to-image 1.11.11**: Utilizado para convertir contenido HTML en una imagen.
9. **@hello-pangea/dnd 16.6.0**: Biblioteca de arrastrar y soltar, posiblemente utilizada para la funcionalidad de arrastrar y soltar en la interfaz de usuario.
10. **uuid 9.0.1**: Utilizado para generar identificadores únicos.

Requisitos del sistema

Hardware

Los requisitos mínimos para ejecutar una aplicación de Next.js en una computadora son:

1. **Sistema operativo:** Windows, macOS o Linux.
2. **Procesador:** Procesador x86-64 o compatible.
3. **Memoria RAM:** Se recomienda al menos 4 GB de RAM.
4. **Espacio en disco:** Al menos 100 MB de espacio en disco para la instalación.

Software

Los requisitos mínimos de software para ejecutar una aplicación de Next.js son los siguientes:

1. **Node.js:** Debes tener Node.js instalado en tu computadora.
2. **NPM:** Se instala automáticamente junto con Node.js. Es necesario para instalar las dependencias del proyecto.
3. **Navegador web:** Para ver y probar la aplicación, necesitarás un navegador web como Google Chrome, Mozilla Firefox, Safari, etc.

Configuración del entorno de desarrollo

Para configurar el entorno de desarrollo, sigue estos pasos:

1. Clona el repositorio de GitHub: **git clone https://github.com/captainsparrow10/desarrollo5-proyecto.git**
2. Instala las dependencias: **npm install**
3. Inicia el servidor de desarrollo: **npm run dev**
4. Accede a la aplicación en **http://localhost:3000**

Despliegue

Para desplegar la aplicación en producción, sigue estos pasos:

1. Asegurarse de haber hecho la configuración del entorno de desarrollo
2. Instala Vercel CLI si aún no lo has hecho. Puedes instalarlo usando npm:
   npm install -g vercel
3. Inicia sesión en Vercel CLI utilizando el comando:
   vercel login
4. Preparar tu proyecto:

   Asegúrate de que tu proyecto de "Generador de Memes" está listo para ser desplegado. Debe tener un archivo package.json con todas las dependencias necesarias y un script de inicio (start) configurado en el scripts sección para ejecutar tu aplicación en modo de producción.

5. Despliegue en Vercel:
   a. Desde la terminal, navega hasta el directorio raíz de tu proyecto.
   b. Ejecuta el siguiente comando para desplegar tu aplicación en Vercel:
   vercel
   c. Vercel te guiará a través de un proceso interactivo de despliegue. Asegúrate de configurar la configuración de tu proyecto según sea necesario.
   d. Una vez que el despliegue haya finalizado con éxito, Vercel te proporcionará una URL pública donde puedes acceder a tu aplicación desplegada.

Si el despliegue ya esta y necesitas hacer un cambio en producción, el proceso de despliegue en Vercel para hacer cambios en producción, siguiendo un flujo seguro de trabajo:

1. Crear una rama para el cambio: Crea una nueva rama en tu repositorio de git para los cambios que quieres hacer en producción. Por ejemplo:

   git checkout -b nombre-de-branch

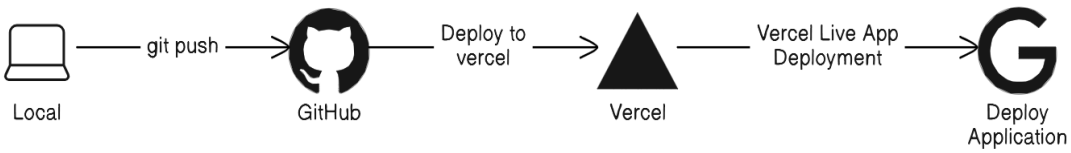
2. Realizar los cambios: Haz los cambios necesarios en tu código y realiza las pruebas locales que consideres necesarias.
3. Hacer commit y push: Haz commit de tus cambios y luego haz push a la rama que creaste en tu repositorio remoto (por ejemplo, GitHub):

   git add .

   git commit -m "cambio a produccion"

git push origin nombre-de-branch

4. Crear un pull request (PR): En tu repositorio en GitHub, crea un pull request desde la rama que acabas de empujar hacia la rama principal (o la rama de producción).
5. Revisar y aprobar el PR: Revisa el pull request y asegúrate de que todo esté correcto. Una vez revisado, aprueba y fusiona el pull request en la rama principal.
6. Despliegue automático en Vercel: Si tienes configurada la integración continua (CI) entre GitHub y Vercel, Vercel detectará automáticamente el cambio en la rama principal y desplegará la nueva versión de tu aplicación en producción. Si no tienes configurada la CI, puedes hacer un deploy manual desde la interfaz de Vercel.
7. Verificar el despliegue: Una vez que Vercel haya desplegado los cambios, verifica que todo esté funcionando correctamente en tu aplicación en producción.



Guía de desarrollo

Estructura del proyecto

- **public:** Contiene archivos públicos de la aplicación.
    - **icons:** Archivos tsx de los iconos utilizados en la aplicación.
    - **sticker:** Imágenes utilizadas como stickers en la aplicación.
- **src:** Directorio que contiene el código público de la aplicación organizado de la siguiente manera:
    - **app:** Lo que se ve en la aplicación.
    - **components:** Archivos reutilizables con funciones específicas usadas en la aplicación.
    - **context:** Contiene el contexto de la aplicación para el manejo global de la data.
    - **services:** Contiene los servicios del API de la aplicación.
    - **types:** Contiene los tipos utilizados en la aplicación para mantener una estructura de información estándar.
    - **util:** Contiene funciones utilizadas en toda la aplicación.
- **.eslintrc.json:** Archivo de configuración para ESLint, una herramienta de análisis estático de código para identificar problemas en el código JavaScript.
- **.gitignore:** Archivo que especifica qué archivos y directorios debe ignorar Git, por ejemplo, los archivos generados automáticamente o los archivos de dependencias.
- **.prettierrc.js:** Archivo de configuración para Prettier, una herramienta de formateo de código para mantener un estilo de código consistente en todo el proyecto.
- **next-env.d.ts:** Archivo de definiciones de TypeScript para Next.js, que ayuda a TypeScript a reconocer los tipos específicos de Next.js.
- **next.config.mjs:** Archivo de configuración de Next.js, utilizado para configurar opciones personalizadas para el proyecto Next.js.
- **package-lock.json:** Archivo generado automáticamente por npm para mantener un registro de las versiones exactas de las dependencias instaladas en el proyecto.
- **package.json:** Archivo de configuración de npm que contiene metadatos sobre el proyecto y las dependencias utilizadas, así como scripts para ejecutar tareas comunes.
- **postcss.config.js:** Archivo de configuración para PostCSS, una herramienta utilizada para transformar CSS con JavaScript.

- **tailwind.config.ts:** Archivo de configuración para Tailwind CSS, un framework de CSS utilitario, utilizado para personalizar la configuración de Tailwind CSS en el proyecto.
- **tsconfig.json:** Archivo de configuración de TypeScript, utilizado para configurar las opciones de compilación de TypeScript en el proyecto.

Convenciones de Codificación

- Utilizar nombres descriptivos y significativos para las variables, funciones y componentes.
- Seguir las convenciones de estilo de código recomendadas por ESLint y Prettier para mantener un código limpio y legible.
- Comentar el código de forma clara y concisa para explicar su propósito y funcionamiento, especialmente en partes complejas o difíciles de entender.

Integración de Herramientas de Desarrollo

- **Control de Versiones:** Utilizar Git para el control de versiones del proyecto, con repositorios alojados en plataformas como GitHub o GitLab.
- **Sistemas de Construcción:** Utilizar npm como gestor de paquetes para instalar dependencias y ejecutar scripts de construcción, pruebas y despliegue.