

Guía de Actividades Práctico-Experimentales Nro. 006

1. Datos Generales

Nombre del estudiante	Miguel Angel Luna Yunga
Asignatura	Desarrollo Basado en Plataformas
Ciclo	5 "A"
Unidad	2
Resultado de aprendizaje de la unidad	R1. Diseña e implementa aplicaciones Web básicas, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad R2. Describe las diferencias entre software-comoservicio y los productos tradicionales de software, bajo los principios de solidaridad, transparencia, responsabilidad y honestid
Práctica Nro.	006
Título de la Práctica	Desarrollo de un prototipo de interfaz en Stitch y HTML/CSS con ocho pantallas documentadas en el repositorio del equipo.
Nombre del Docente	Edison Leonardo Coronel Romero
Fecha	Viernes 28 de noviembre
Horario	07h30 – 10h30
Lugar	Aula 232
Tiempo planificado en el Sílabo	3 horas

2. Objetivo(s) de la Práctica:

- Diseñar y construir un prototipo sencillo de interfaz que represente al menos un flujo completo del proyecto (8 pantallas), utilizando Stitch como herramienta de apoyo al diseño y HTML/CSS para la implementación base.
- Aplicar principios de diseño responsivo, maquetación semántica y criterios básicos de accesibilidad y usabilidad.
- Documentar el prototipo mediante capturas de pantalla y una descripción técnica en el repositorio del equipo.

3. Materiales, Reactivos, Equipos y Herramientas

- Computador con acceso a Internet.
- Navegador web actualizado.
- Editor de código (VS Code o similar).
- Herramienta de prototipado Stitch (o entorno definido por la asignatura).
- Repositorio del proyecto en GitHub / GitLab.
- Lineamientos de UI/UX del proyecto (si ya existen).
- Laboratorio de Desarrollo de Software o equipos personales.
- VS Code (o equivalente) con extensión Live Server (opcional).
- Stitch para diseño y organización de pantallas.
- Sistema de control de versiones Git + cliente (GitKraken, CLI, etc.).
- Navegador con DevTools para ver el diseño responsivo.

4. **Procedimiento / Metodología Ejecutada** Inicio

Paso 1: Definición rápida del flujo.

Tema de Software escogido:

Aplicación de Floración

- **Elegir un flujo funcional del proyecto:**

El flujo adoptado se estructura de la siguiente manera: Login, Dashboard, selección de origen de datos (Mapas o Cargas), Análisis IA, módulos de soporte a la decisión y salida del sistema, el proceso inicia con la Autenticación, donde el usuario ingresa sus credenciales y, si son correctas, accede al Dashboard, que funciona como el centro de mando mostrando métricas globales, un mapa resumido y alertas recientes desde aquí el usuario elige cómo obtener los datos a analizar mediante dos vías principales: la Vía A, que corresponde al Explorador de Mapas con navegación global, activación de capas, filtros de fecha y nubosidad, selección de región y la acción Analizar Región; o la Vía B, que consiste en la Carga de Imágenes locales (GeoTIFF/RGB), donde tras el procesamiento se habilita Ver Resultados, ambas vías convergen en la pantalla de Análisis IA, donde se visualiza la comparativa Antes/Después mediante un slider, se revisan las máscaras U-Net y se descargan los reportes técnicos.

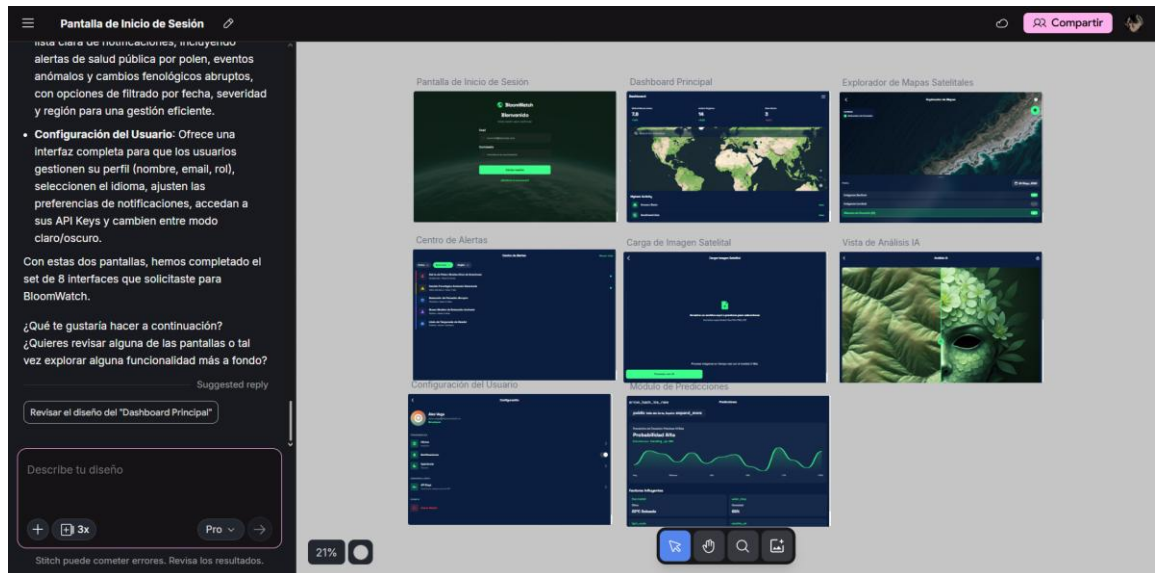
A continuación, el usuario puede acceder a los Módulos de Soporte a la Decisión, que incluyen Predicciones con gráficos y mapas de calor, así como el Centro de Alertas con notificaciones críticas. Finalmente, desde el apartado de Gestión se accede a Configuración y Perfil para personalizar idioma, notificaciones y API Keys, o se ejecuta Logout para retornar a la pantalla de Autenticación. **El flujo general puede resumirse como:** Login → Dashboard → (Mapas o Cargas) → Análisis IA → Reportes / Toma de decisiones.

- **Definir las 8 pantallas que formarán parte del flujo y sus relaciones**

Las 8 pantallas desarrolladas son: Inicio de Sesión, Dashboard, Explorador de Mapas, Carga de Imagen Satelital, Análisis de IA, Predicciones, Centro de Alertas y Configuración, cada una cumple un rol específico dentro de la navegación: el Inicio de Sesión actúa como puerta de acceso; el Dashboard funciona como el hub central desde el cual parten las rutas principales; el Explorador de Mapas y la Carga de Imagen Satelital conforman las dos vías para obtener datos de entrada; ambas desembocan en el Análisis de IA, que concentra el procesamiento principal; posteriormente, el usuario puede complementar la toma de decisiones con Predicciones y revisar información crítica en el Centro de Alertas; finalmente, la pantalla de Configuración permite ajustar preferencias y gestionar el uso del sistema. En conjunto, estas pantallas conforman un flujo continuo, coherente y orientado a la acción.

Paso 2: Prototipado en Stitch

- Crear en Stitch las 8 pantallas con:
 - ✓ Distribución básica de elementos (header, menú, contenido, footer),
 - ✓ Ubicaciones aproximadas de botones, formularios, tablas/listas, jerarquía visual (títulos, subtítulos, texto)
- Ajustar la navegación entre pantallas (enlaces o interacciones de Stitch).



Paso 3: Implementación en HTML/CSS

```

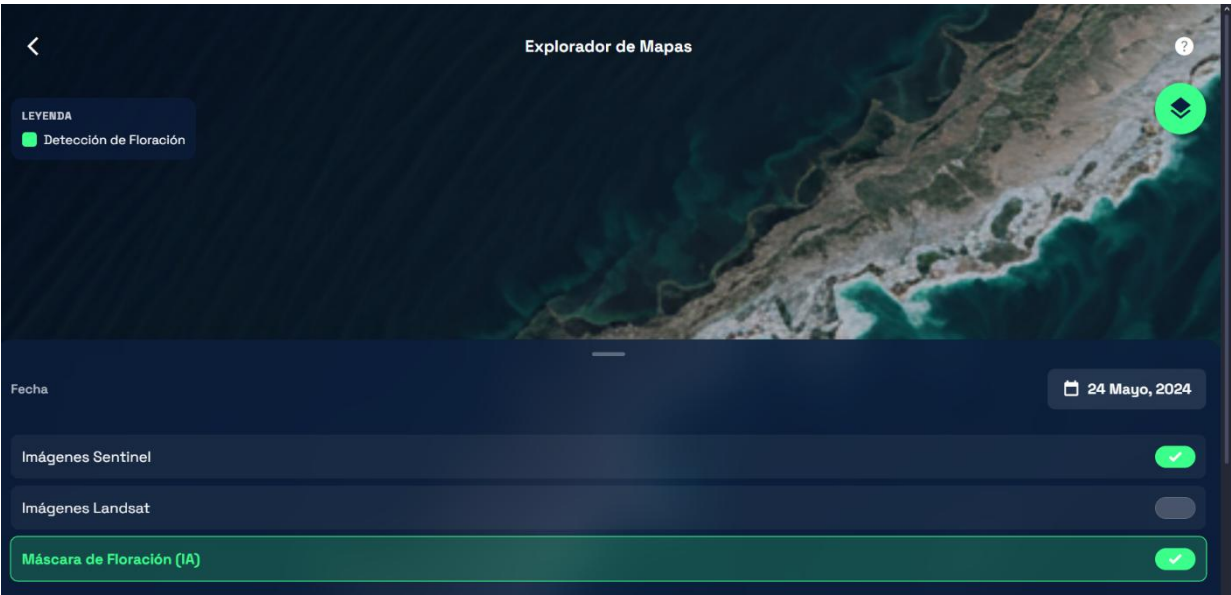
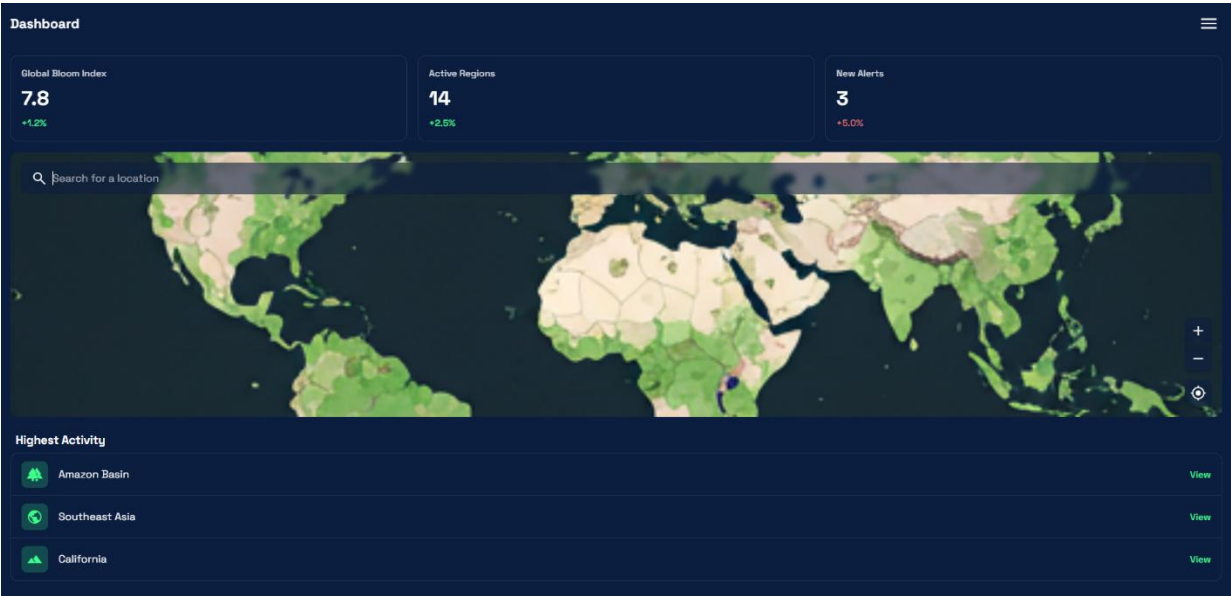
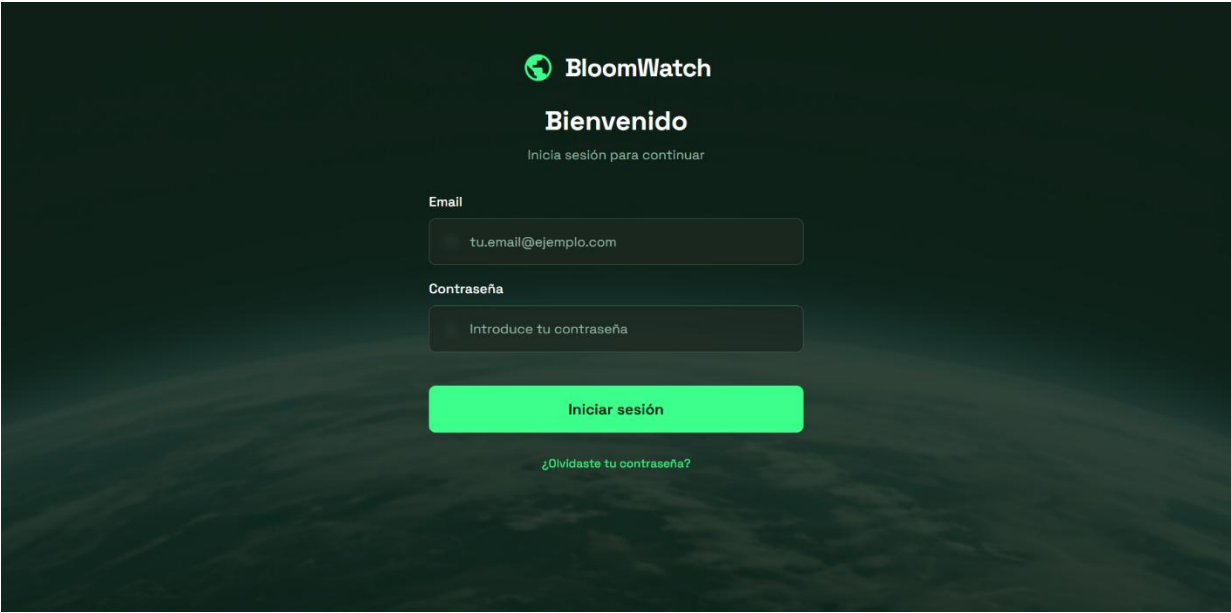
BLOOMWATCH
├── components
├── img
├── node_modules
├── screens
├── .env.local
├── .gitignore
├── App.tsx
├── index.html
├── index.tsx
├── metadata.json
├── package-lock.json
├── package.json
├── README.md
├── tsconfig.json
├── types.ts
├── vite.config.ts
├── ESQUEMA
├── LÍNEA DE TIEMPO
├── VS CODE POKÉMON
└── index.html
  1 <!DOCTYPE html>
  2 <html lang="es">
  3 <head>
  4 <meta charset="UTF-8" />
  5 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  6 <title>BloomWatch | Earth Observation Platform</title>
  7 <script src="https://cdn.tailwindcss.com"></script>
  8 <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&family=JetBrainsMono:wght@400;500&display=swap" rel="stylesheet">
  9
  10 <!-- Leaflet CSS & JS -->
  11 <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" integrity="sha256-p4NlxaoVBNyHm4RzRCF9TD/mizYoHSobTR98MY=" c
  12 <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js" integrity="sha256-20nQcchB9coBq1j2RGuK2/Z9VW+kliyxW1v1ZBo=" crossorigin=""></s
  13
  14 <script>
  15   tailwind.config = {
  16     darkMode: 'class',
  17     theme: {
  18       extend: {
  19         colors: {
  20           bw: {
  21             dark: '#0B1E3F',
  22             darker: '#050F21',
  23             neon: '#3DFF8A',
  24             neonDim: 'rgb(61, 255, 138, 0.1)',
  25             text: '#E2E8F0',
  26             muted: '#94A3B8'
  27           },
  28         },
  29         fontFamily: {
  30           sans: ['Inter', 'sans-serif'],
  31           mono: ['JetBrains Mono', 'monospace'],
  32         },
  33       },
  34     },
  35   }
  36 </script>
  37 </style>

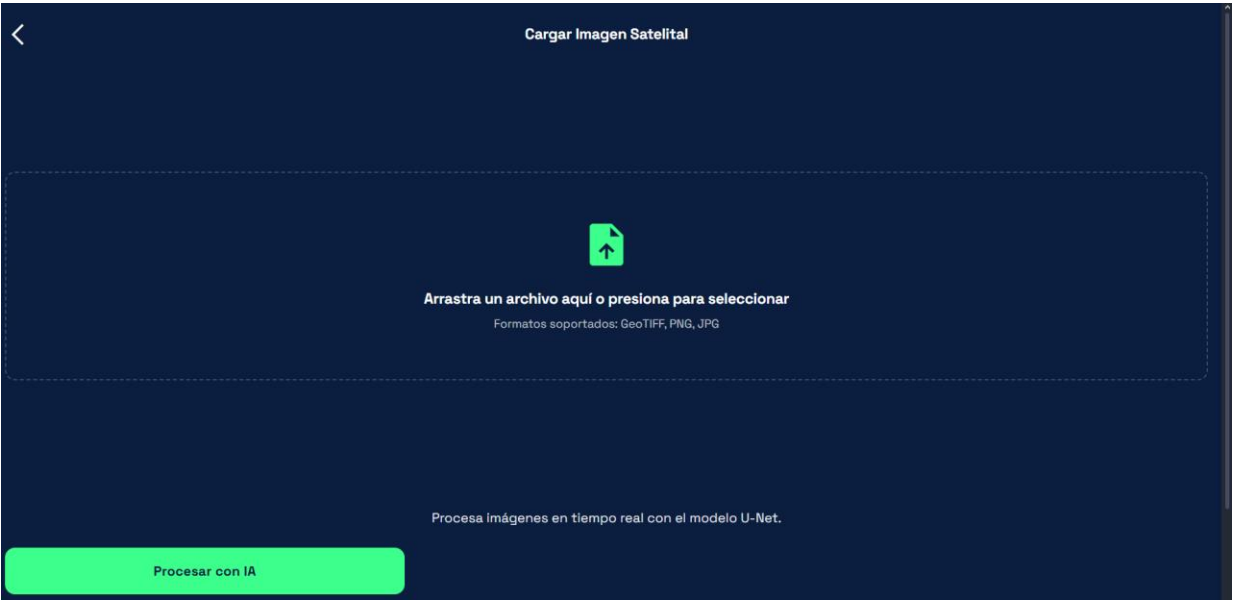
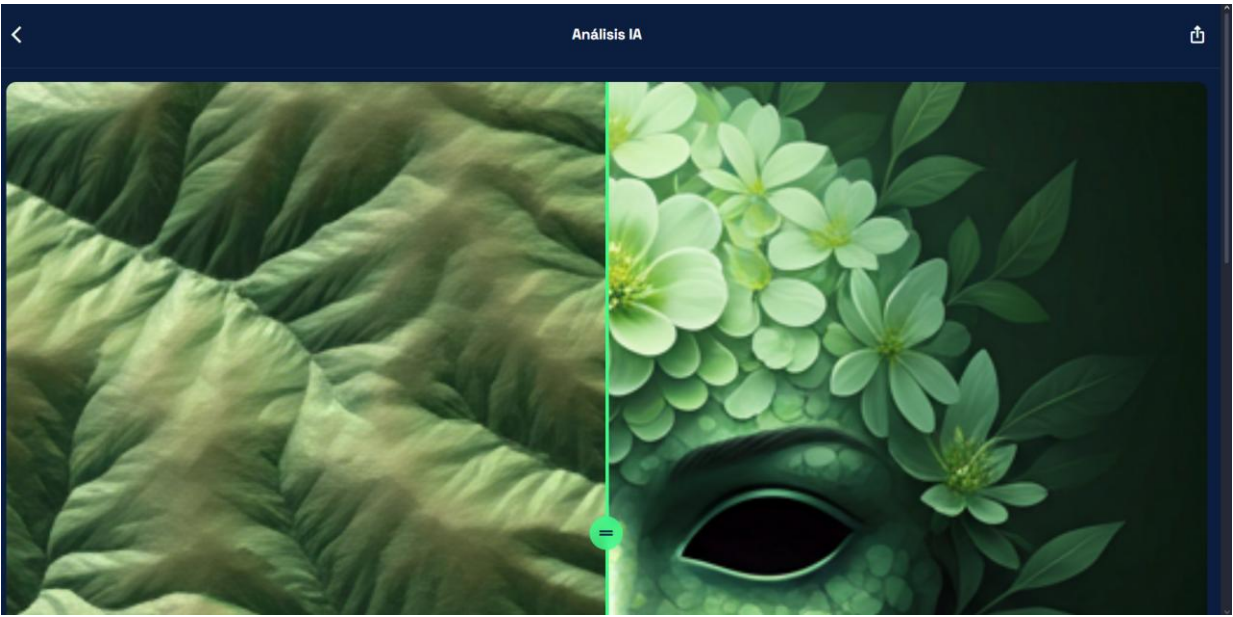
```

Paso 4: Pruebas y capturas

- Verificar cada pantalla en al menos dos tamaños de ventana (ej. escritorio y móvil).
- Tomar capturas de pantalla de cada una de las 8 pantallas en su estado principal.

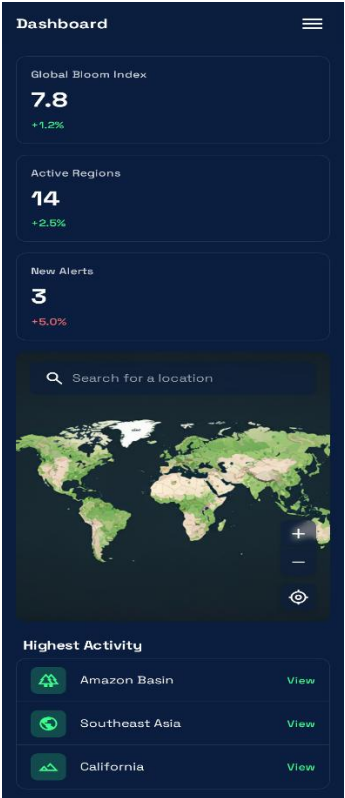
Escritorio:

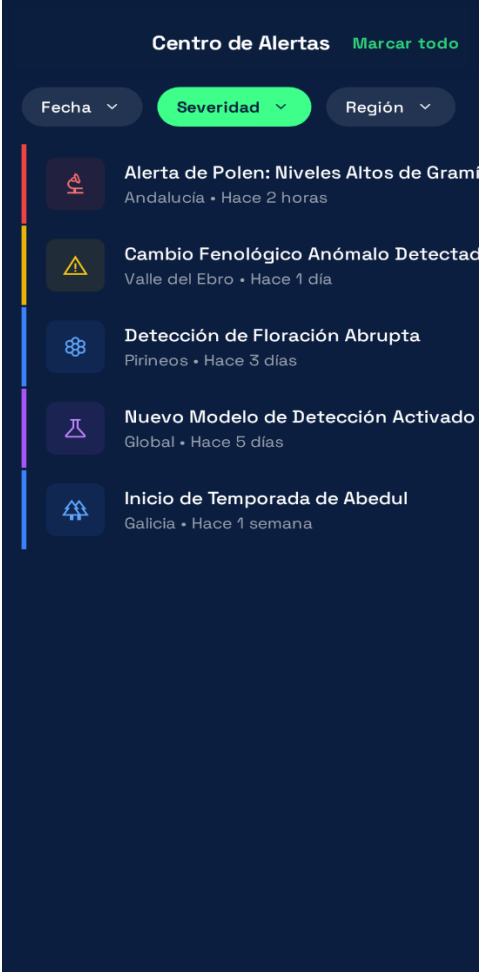
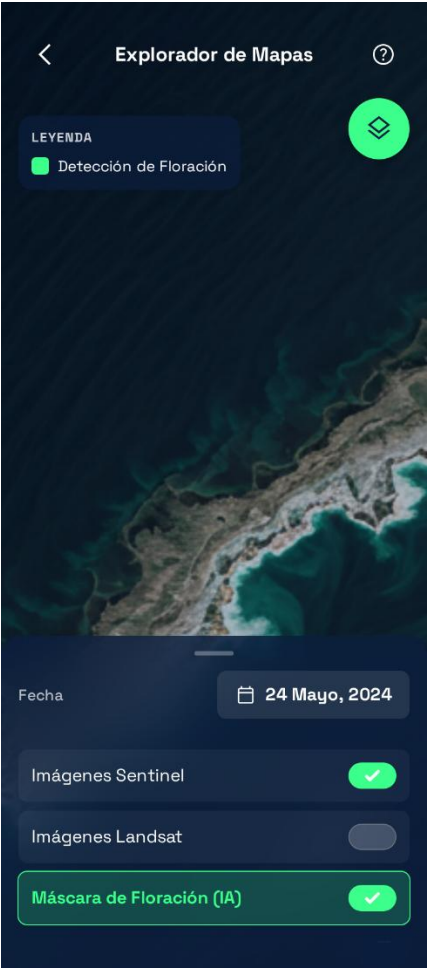


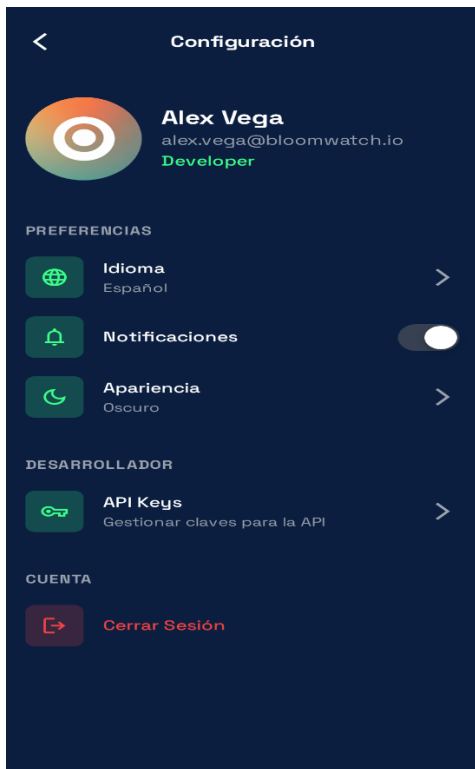




Móvil:



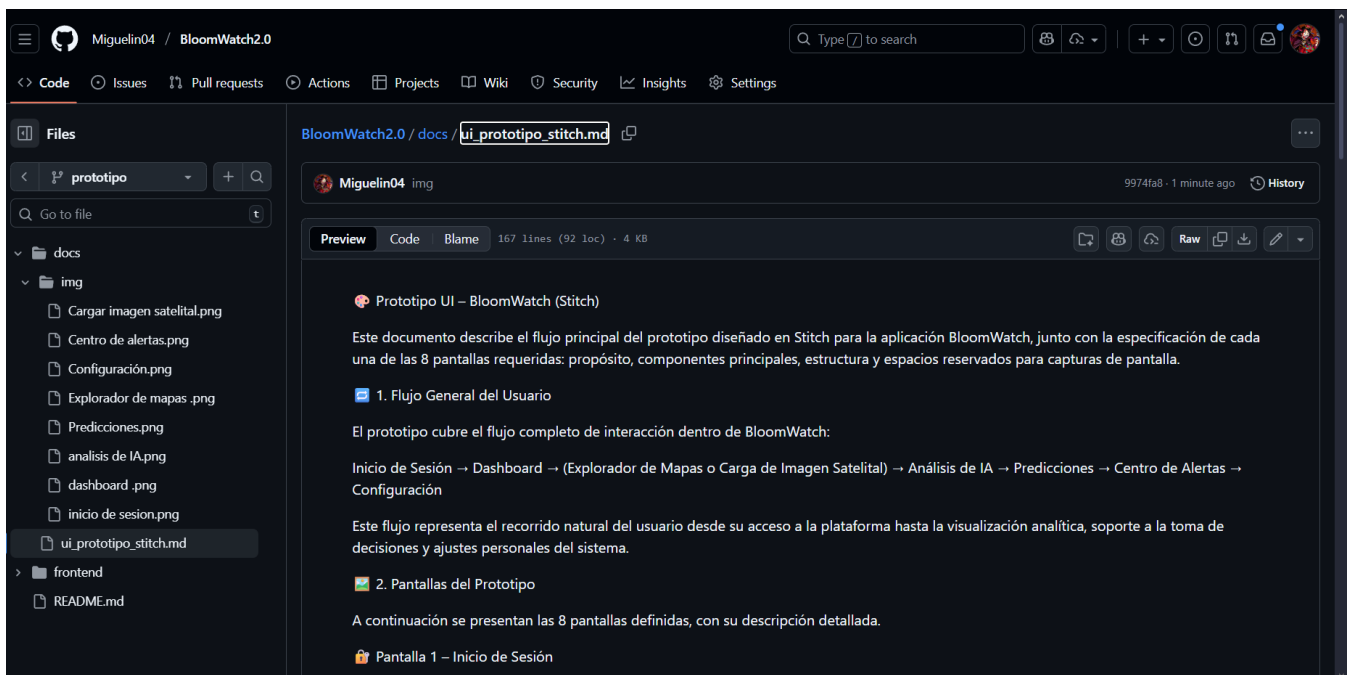




Paso 5: Documentación en el repositorio

En la carpeta /docs/ o similar, crear un archivo, por ejemplo: docs/ui_prototipo_stitch.md con:

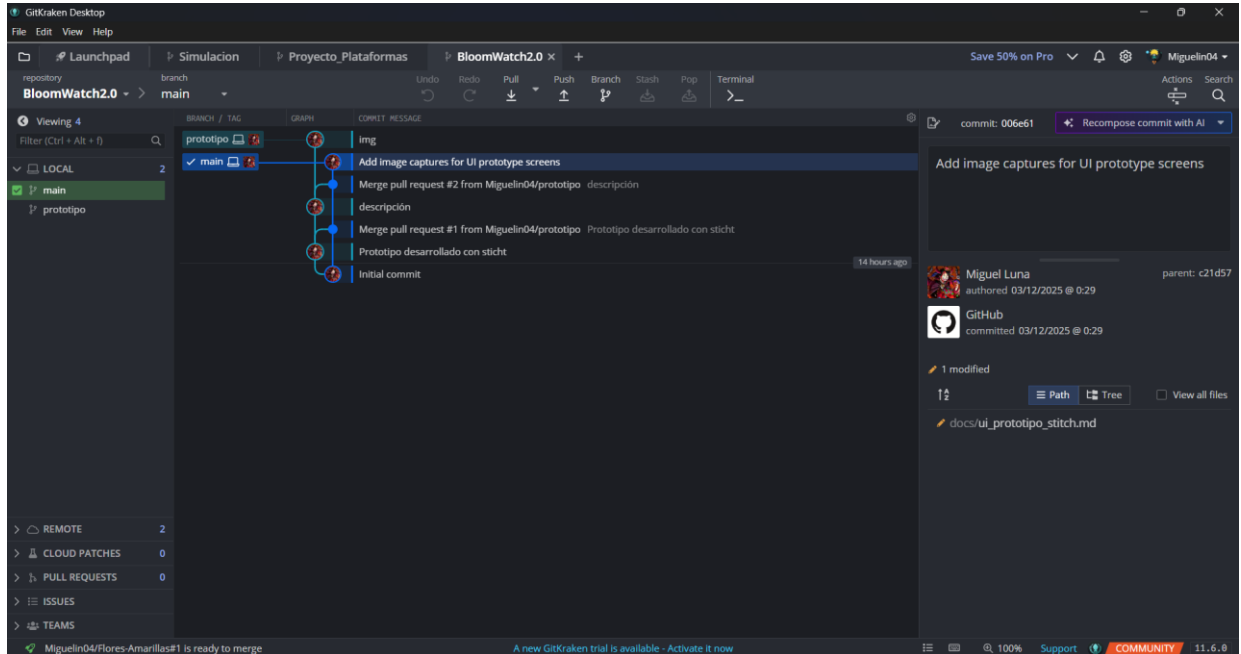
- Breve descripción del flujo que cubren las 8 pantallas.
- Una subsección para cada pantalla:
 - ✓ nombre de la pantalla,
 - ✓ propósito,
 - ✓ componentes principales (botones, formularios, listas, etc.),
 - ✓ captura de pantalla (enlazada o referenciada desde /docs/img/).



Paso 6: Commit y push

Hacer commit con un mensaje descriptivo, por ejemplo: feat(ui): prototipo

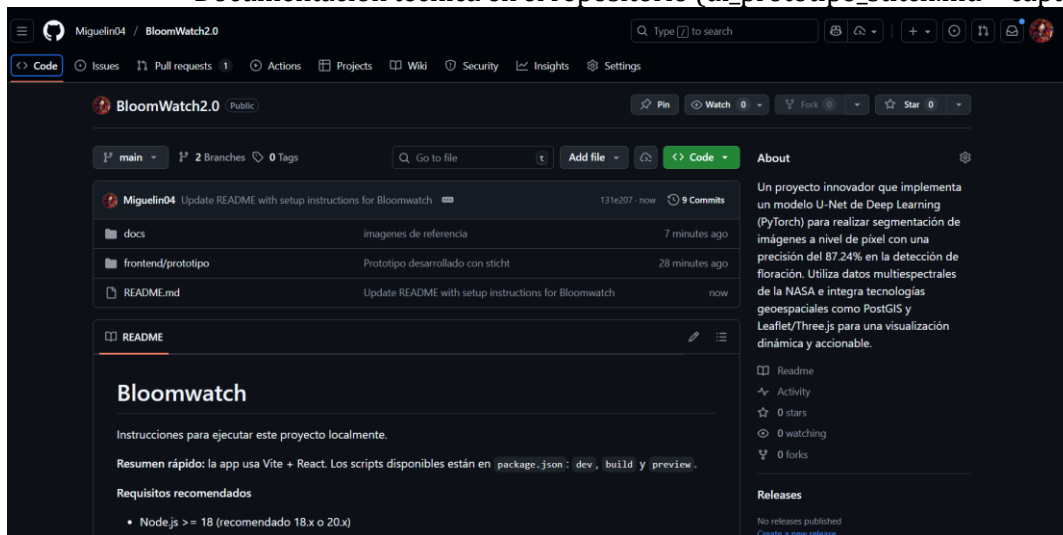
HTML/CSS basado en Stitch (8 pantallas)



5. Resultados esperados:

Prototipo de interfaz con 8 pantallas implementadas en HTML/CSS, coherentes con el flujo del proyecto.

- ✓ Prototipo en Stitch que sirvió de base para la implementación.
- ✓ Documentación técnica en el repositorio (ui_prototipo_stitch.md + capturas).



6. Preguntas de Control:

- ¿Qué ventajas encuentras al usar primero un prototipo en Stitch antes de codificar en HTML/CSS?

El uso de un prototipo inicial en herramientas como Stitch o sistemas de generación rápida de código ofrece ventajas significativas antes de escribir HTML y CSS manualmente. Permite iterar de forma visual en cuestión de segundos, validar paletas de colores como el tema dark sci-fi y ajustar el layout sin invertir tiempo en construir la interfaz desde cero. Además, facilita la validación temprana de los flujos de usuario —por ejemplo, Login → Dashboard → Mapas— sin necesidad de configurar un backend o

un sistema de rutas complejo, y promueve la creación anticipada de componentes reutilizables como Sidebar, Toast o MetricCard, evitando generar código desordenado que luego requeriría refactorización

- **¿Por qué es importante que las pantallas estén construidas con etiquetas semánticas?**

Las pantallas se construyeron utilizando etiquetas semánticas como `<nav>`, `<aside>`, `<main>`, `<header>` y `<section>` porque aportan beneficios clave. Desde el punto de vista de accesibilidad, los lectores de pantalla pueden interpretar correctamente la estructura del contenido y permitir una navegación más eficiente para usuarios con discapacidad visual. También mejoran la claridad del código, ya que un desarrollador puede comprender de inmediato la función de cada bloque, facilitando el mantenimiento. Asimismo, contribuyen a una jerarquía de información más clara, útil incluso en un dashboard privado.

- **¿Qué elementos de diseño responsivo aplicaste en, al menos, una de las pantallas?**

En cuanto al diseño responsivo, se aplicaron utilidades de Tailwind CSS siguiendo un enfoque mobile-first. En el Dashboard, por ejemplo, las tarjetas métricas utilizan una grilla adaptable (`grid-cols-1 md: grid-cols-2 lg: grid-cols-4`), lo que permite mostrar las tarjetas en una columna en móviles, dos en tablets y cuatro en pantallas grandes. Lo mismo ocurre con la distribución del mapa y el panel lateral, que pasan de una sola columna a una grilla más amplia en pantallas grandes. En la pantalla de Configuración, la estructura cambia de una columna a `md: grid-cols-3`, asignando un tercio al perfil y dos tercios al formulario en dispositivos de mayor tamaño.

- **¿Qué mejoras podrías realizar para mejorar la accesibilidad (colores, tamaños, estructura)?**

A pesar de estos avances, todavía existen mejoras de accesibilidad por implementar. El contraste entre el texto `text-bw-muted` y el fondo oscuro podría no cumplir con los estándares WCAG, por lo que sería recomendable ajustar el color para facilitar la lectura. También es importante reforzar los indicadores de foco al navegar con teclado, especialmente en inputs y botones. Los botones que contienen únicamente íconos deberían incluir etiquetas ARIA para comunicar su función a los lectores de pantalla. Finalmente, las animaciones utilizadas deberían respetar la preferencia del sistema `prefers-reduced-motion` para evitar molestias a usuarios sensibles al movimiento.

- **¿Cómo ayuda esta práctica al desarrollo posterior del frontend definitivo del proyecto?**

Esta práctica aporta un valor considerable al desarrollo del frontend definitivo. El prototipo funciona como un plano maestro: ya define tipos clave en TypeScript (como `AnalysisContext`, `Alert` o `ViewState`), establece un sistema de diseño consistente mediante tokens configurados en `tailwind.config` y organiza la lógica de estado centralizada en `App.tsx`. Al trasladar el proyecto a frameworks como Next.js, esta base permite integrar más fácilmente un gestor de estado como Context API o Redux, conectar datos reales del backend y asegurar una interfaz coherente, accesible y escalable.

7. Conclusiones

- El proyecto BloomWatch presenta una identidad visual que logra con éxito una estética científica-futurista gracias al uso de Tailwind con una paleta personalizada y efectos de Glassmorphism, lo que transmite profesionalismo y modernidad, elementos esenciales en una plataforma de análisis satelital y la experiencia de usuario es intuitiva y el flujo general desde el inicio de sesión, pasando por el dashboard y llegando al análisis de IA—es coherente y fluido.
- A nivel técnico, la arquitectura de componentes está bien estructurada: el componente `SimulatedMap` demuestra ser flexible y reutilizable, al abstraer correctamente la complejidad de Leaflet y permitir modificaciones del mapa mediante props. La gestión básica de vistas en `App.tsx` evidencia de manera efectiva cómo circulan los datos entre pantallas, especialmente al mover

información del Explorador de Mapas hacia el módulo de Análisis de IA. La selección del stack tecnológico también es acertada, integrando herramientas ligeras y eficientes como Leaflet para mapas 2D, Recharts para predicciones temporales y Lucide React para iconografía uniforme.

- Además, el prototipado de alta fidelidad, con simulación de latencia y estados de carga, proporciona una experiencia muy cercana a un sistema productivo y permite validar la usabilidad sin necesidad de un backend operativo.

8. Recomendaciones

- En cuanto a recomendaciones, la evolución hacia producción requiere implementar un sistema de rutas real mediante react-router-dom, ya que el ruteo condicional actual no es escalable y limita funciones como el deep linking. También es importante migrar la gestión global de estado hacia soluciones como Context API o Zustand para evitar el prop drilling, que se volverá complejo conforme crezca la aplicación.
- También resulta clave fortalecer la accesibilidad mediante verificaciones WCAG, mejorar la calidad del software con pruebas unitarias y E2E, y optimizar la experiencia móvil con vistas simplificadas para pantallas pequeñas, ya que la manipulación de mapas detallados en dispositivos reducidos puede resultar poco práctica.
- Finalmente, como hoja de ruta sugerida, se recomienda iniciar con la implementación del ruteo real, continuar con la conexión del **login** y dashboard a una API simulada para validar flujos completos, avanzar en la integración de GeoJSON real dentro de *SimulatedMap*, y cerrar con tareas de accesibilidad, rendimiento y carga diferida de componentes.

9. Bibliografía/Referencias

[1] N. Vlajic, X. Y. Shi, H. Roumani and P. Madani, "Rethinking the Use of Resource Hints in HTML5: Is Faster Always Better!?", in Journal of Cyber Security and Mobility, vol. 6, no. 2, pp. 195-226, May 2017, doi: 10.13052/jcsm2245-1439.625.