**Primer Proyecto Práctico.**

**2ª Evaluación:**

**AgriWeb App - Plataforma Inteligente de Agricultura Local.**

**ÍNDICE**

[**Requerimientos Funcionales:** 3](#_Toc188910621)

[**Requerimientos Técnicos:** 4](#_Toc188910622)

[**Estructura Propuesta de la Aplicación:** 4](#_Toc188910623)

[**Resultados Esperados:** 5](#_Toc188910624)

[**Criterios de Evaluación:** 6](#_Toc188910625)

**Descripción General:**

El proyecto será una **SPA (Single Page Application)** enfocada en brindar herramientas útiles a agricultores y ganaderos mediante tecnologías modernas del entorno cliente. La aplicación incluirá funcionalidades de autenticación, geolocalización, visualización de datos ambientales y análisis de imágenes para apoyar en la toma de decisiones sobre actividades agrícolas.

**Requerimientos Funcionales:**

**1. Autenticación de Usuarios**

* Implementar autenticación por **email** y **contraseña** utilizando **Firebase Authentication**.
* Permitir registro, inicio de sesión, cierre de sesión y recuperación de contraseñas.

**2. Geolocalización**

* Una vez autenticado, el usuario podrá:
  + Obtener su **ubicación geográfica** (latitud, longitud).
  + Mostrar su posición actual sobre un mapa interactivo utilizando la librería **Leaflet.js** o similar.

**3. Información Meteorológica (API OpenWeatherMap)**

* Integrar la API de **OpenWeatherMap** para consultar el estado del tiempo y pronósticos basados en la ubicación del usuario.
* Visualizar datos como:
  + Temperatura actual.
  + Humedad.
  + Precipitaciones.
  + Velocidad del viento.
* Presentar esta información en tarjetas o gráficos interactivos.

**4. Información Gráfica sobre el Mapa**

* Mostrar sobre el mapa:
  + Información ambiental de la zona (e.g., niveles de humedad o lluvias recientes).
  + Agregar capas adicionales en el mapa, como límites de zonas agrícolas relevantes.

**5. Análisis Masivo de Datos**

* Consultar y obtener datos masivos de la API (e.g., condiciones meteorológicas de múltiples ubicaciones o histórico de datos).
* Presentar los datos al usuario en tablas interactivas utilizando **DataTables.js**, con funcionalidades como:
  + Filtros.
  + Ordenación.
  + Exportación a CSV/Excel.

**6. Captura y Análisis de Imágenes (TensorFlow.js)**

* Permitir al usuario tomar fotos desde su dispositivo móvil, seleccionando entre la cámara frontal o trasera.
* Analizar las imágenes utilizando **TensorFlow.js** y modelos preentrenados para fines agrícolas, como:
  + Identificar el estado de salud de una planta (e.g., detección de enfermedades).
  + Clasificar tipos de cultivos.
  + Detectar niveles de madurez de frutos.

**Requerimientos Técnicos:**

1. **Tecnologías de Desarrollo**:
   * **HTML5**, **CSS3**, **JavaScript (ES6)**.
   * Framework opcional para SPA: **React.js**, **Vue.js** o **Angular** (se recomienda React.js para simplicidad).
   * Librerías:
     + **Firebase Authentication** (autenticación).
     + **Leaflet.js** o similar (mapas interactivos).
     + **OpenWeatherMap API** (datos meteorológicos).
     + **DataTables.js** (tablas interactivas).
     + **TensorFlow.js** (análisis de imágenes).
2. **Dependencias Específicas**:
   * **Firebase**: para la autenticación.
   * **Axios**: para consumir la API de OpenWeatherMap.
   * **Leaflet.js**: para mapas interactivos.
   * **TensorFlow.js**: para análisis de imágenes.
3. **Diseño y UX/UI**:
   * Interfaz responsiva (compatible con dispositivos móviles).
   * Uso de frameworks de diseño como **Tailwind CSS** o **Bootstrap**.
   * Experiencia de usuario fluida e intuitiva.
4. **Herramientas Opcionales**:
   * **Chart.js** o **D3.js**: para gráficos visuales de datos meteorológicos.
   * **Dexie.js**: para almacenamiento local de datos (opcional).

**Estructura Propuesta de la Aplicación:**

**Páginas/Componentes:**

1. **Inicio de Sesión / Registro**:
   * Formulario para autenticarse y/o registrarse.
2. **Dashboard Principal**:
   * Bienvenida al usuario autenticado.
   * Visualización de datos meteorológicos actuales.
   * Acceso a funciones clave: mapa, tablas de datos, análisis de imágenes.
3. **Mapa Interactivo**:
   * Mostrar ubicación actual del usuario.
   * Cargar información adicional (e.g., capas de humedad o clima).
4. **Tablas Interactivas**:
   * Listar datos masivos obtenidos de la API.
5. **Cámara y Análisis**:
   * Componente para capturar imágenes y ejecutar el modelo de TensorFlow.js.

**Resultados Esperados:**

Una aplicación web interactiva y funcional que permita a los estudiantes integrar conocimientos de:

* Autenticación de usuarios.
* Geolocalización y visualización de mapas.
* Consumo de APIs y visualización de datos.
* Uso de inteligencia artificial en el cliente.

/agriweb-app-daw2xx

├── /css

│ ├── style.css # Estilos principales

│ └── responsive.css # Estilos responsivos

├── /js

│ ├── app.js # Lógica principal

│ ├── auth.js # Funcionalidades de autenticación

│ ├── maps.js # Funciones del mapa interactivo

│ ├── tables.js # Configuración de tablas dinámicas

│ └── tensorflow.js # Análisis de imágenes

├── /assets

│ ├── /images # Imágenes de la aplicación

│ └── /icons # Iconos usados en la interfaz

├── index.html # Página principal

└── dashboard.html # Página del panel principal

**Flujo de Navegación**

1. **Inicio:**
   * El usuario accede a la Landing Page e inicia sesión.
2. **Dashboard:**
   * Al autenticarse, el usuario es redirigido al panel principal.
   * Desde aquí puede acceder a las funcionalidades principales:
     + Consultar información meteorológica.
     + Visualizar mapas interactivos.
     + Consultar tablas de datos masivos.
     + Capturar imágenes para análisis.
3. **Exploración:**
   * El usuario navega entre las secciones del sistema según sus necesidades.

**Criterios de Evaluación:**

| **Criterio** | **Indicadores de Logro** | **Peso (%)** | **Puntos Máximos (100)** |
| --- | --- | --- | --- |
| **a) Evaluación de ventajas e inconvenientes de comunicación asíncrona.** | - El informe o documentación incluye una explicación sobre los beneficios y limitaciones de usar comunicación asíncrona (e.g., interactividad, velocidad vs complejidad técnica). | 1% | 1 |
| **b) Análisis de mecanismos disponibles para comunicación asíncrona.** | - Se describen y justifican los mecanismos usados en el proyecto (e.g., fetch, Axios, Firebase). | 1% | 1 |
| **c) Uso de objetos relacionados con comunicación asíncrona.** | - Uso correcto de objetos como fetch o XMLHttpRequest para consumir APIs. | 10% | 10 |
| **d) Identificación de propiedades y métodos de los objetos.** | - Se evidencia el uso de propiedades y métodos relevantes de los objetos utilizados (e.g., métodos de fetch, manejo de respuestas JSON, etc.). | 1% | 1 |
| **e) Comunicación asíncrona en la actualización dinámica del documento Web.** | - Implementación de actualizaciones dinámicas en el DOM basadas en datos de APIs (e.g., OpenWeatherMap, tablas interactivas). | 10% | 10 |
| **f) Uso de distintos formatos en el envío y recepción de información.** | - Consumo de APIs en formatos como JSON o XML (al menos uno). | 10% | 10 |
| **g) Programación de aplicaciones asíncronas para diferentes navegadores.** | - La aplicación es funcional en navegadores modernos como Chrome, Firefox o Edge (se comprueba en la documentación o en la demo). | 32% | 32 |
| **h) Clasificación y análisis de librerías para tecnologías de actualización dinámica.** | - Se justifican las librerías utilizadas en el proyecto (e.g., Firebase, Leaflet.js, DataTables.js) con una breve comparación en la documentación. | 5% | 5 |
| **i) Creación y depuración de programas que utilicen librerías de actualización dinámica.** | - Se implementan correctamente librerías como Leaflet.js, DataTables.js, o TensorFlow.js y se evidencian procesos de depuración en el código entregado. | 30% | 30 |

**Distribución por Funcionalidades del Proyecto:**

Basándonos en esta nueva distribución, los puntos para las funcionalidades específicas del proyecto serán ajustados proporcionalmente para reflejar el peso del criterio **g (32%)** y **i (30%)**, ya que tienen el mayor impacto en la evaluación. Las funcionalidades serían evaluadas de la siguiente manera:

**1. Autenticación de Usuarios (Firebase Authentication)**

* Correcto manejo del flujo de registro, inicio de sesión y cierre de sesión.
* **Puntos máximos:** 5.

**2. Geolocalización y Visualización de Mapas (Leaflet.js)**

* Consumo de la API de geolocalización para obtener latitud y longitud del usuario.
* Implementación de mapas interactivos con Leaflet.js.
* Uso de capas dinámicas para mostrar datos en el mapa.
* **Puntos máximos:** 20 (10 puntos ajustados a funcionalidad + 10% actualización dinámica).

**3. Información Meteorológica (OpenWeatherMap API)**

* Consumo de la API de OpenWeatherMap para obtener y visualizar datos meteorológicos.
* Presentación de datos en tarjetas interactivas.
* **Puntos máximos:** 10.

**4. Tablas Interactivas (DataTables.js)**

* Integración de tablas interactivas para visualizar datos masivos (e.g., históricos meteorológicos).
* Funcionalidades como filtros, ordenación y exportación de datos.
* **Puntos máximos:** 10.

**5. Captura y Análisis de Imágenes (TensorFlow.js)**

* Uso de TensorFlow.js para implementar modelos de análisis agrícola.
* Captura de imágenes desde dispositivos móviles (frontal o trasera).
* Resultados procesados mostrados al usuario.
* **Puntos máximos:** 25 (incluye 15% del criterio i).