UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA
Escuela Profesional de Ingeniería de Sistemas

INFORME TÉCNICO DE TRABAJO ENCARGADO

Curso: PROGRAMACIÓN WEB II

Docente: Mtro. Enrique Félix Lanchipa Valencia

Ayma Choque, Erick Yoel (2021072616)
Tapia Vargas, Dylan Yariet (2021072630)

Tacna – Perú
2025

ÍNDICE

TRABAJO FINAL - I UNIDAD

I. INFORMACIÓN GENERAL

- **Objetivos:**
  - Desarrollar un sistema web que permita la administración, monitoreo y diagnóstico de dispositivos Arduino en un entorno educativo, mediante el uso de tecnologías modernas que garanticen la seguridad, accesibilidad y gestión eficiente de usuarios y datos.
- **Objetivos Específicos**:
  - Diseñar una interfaz web amigable que permita la interacción entre alumnos y administradores.
  - Implementar un sistema de autenticación y roles con Firebase Authentication.
  - Almacenar y consultar datos de diagnóstico de los dispositivos mediante Firestore Database.
  - Desplegar la aplicación web utilizando GitHub Pages para acceso público y mantenimiento remoto.
  - Garantizar la seguridad e integridad de la información mediante validaciones y reglas de acceso.
- **Equipos, materiales, programas y recursos utilizados:**

  Donde se da a conocer los equipos, materiales, programas y recursos que son utilizados en la práctica con su respectivas características / datos técnicos.

  Equipos
  - Computadora personal o laptop con conexión a internet.
  - Dispositivos Arduino UNO y sensores (para pruebas de diagnóstico).

  Software y Programas
  - Visual Studio Code: Entorno de desarrollo del proyecto.
  - Google Chrome: Navegador para ejecución y pruebas.
  - Firebase Console: Plataforma de backend en la nube.
  - Git / GitHub: Control de versiones y despliegue del sitio.
  - Figma / Canva: Diseño visual de las interfaces web.
  -

  Recursos Utilizados
  - Firebase Firestore: Base de datos NoSQL para almacenamiento de usuarios y diagnósticos.
  - Firebase Authentication: Servicio de autenticación de usuarios.
  - HTML5, CSS3 y JavaScript: Lenguajes para la construcción del frontend.
  - GitHub Pages: Servicio gratuito de hosting web.

II. MARCO TEÓRICO

El **Sistema de Administración y Diagnóstico Arduino** está basado en la integración entre el desarrollo web y el Internet de las Cosas (IoT), permitiendo gestionar y visualizar datos provenientes de microcontroladores Arduino desde una interfaz en línea.

1. Conceptos Fundamentales

- Arduino: Plataforma de hardware libre utilizada para crear proyectos electrónicos interactivos.
- Sistema Web: Aplicación accesible mediante un navegador, que permite la interacción del usuario con información almacenada en servidores remotos.
- Firebase: Plataforma de desarrollo de aplicaciones web de Google, que proporciona servicios backend como autenticación, base de datos y hosting sin necesidad de servidores propios.
- Firestore Database: Base de datos NoSQL en tiempo real, optimizada para aplicaciones escalables.
- Autenticación de Usuarios: Proceso que garantiza el acceso autorizado mediante credenciales verificadas.

2. Principios de Arquitectura

El sistema adopta una arquitectura cliente-servidor, donde el cliente (navegador web) se comunica con los servicios en la nube de Firebase.
Este enfoque facilita la escalabilidad, disponibilidad y seguridad del sistema, reduciendo la necesidad de infraestructura física.

Estructura general:

[Usuario] ⟶ [Interfaz Web HTML/CSS/JS]

↓

[Firebase SDK]

↓

[Authentication] ⟶ [Gestión de Accesos]

[Firestore] ⟶ [Datos de Usuarios y Diagnósticos]

3. Fundamentos Técnicos del Desarrollo Web

- HTML5: Define la estructura del sitio.
- CSS3: Controla la presentación y adaptabilidad visual.
- JavaScript: Gestiona la lógica del sistema, validaciones y conexión con Firebase.
- Git: Permite la trazabilidad del código y trabajo colaborativo.

III. PROCEDIMIENTO

1. Diseño del Sistema

   Se inició con el diseño de la estructura visual mediante prototipos elaborados en Figma, definiendo los componentes principales:

   - Página de inicio.
   - Módulo de login y registro.
   - Panel de alumno.
   - Panel de administrador.
   - Configuraciones generales.

   El diseño se orientó a la usabilidad, la claridad de navegación y la compatibilidad con distintos dispositivos, empleando principios de UI/UX.

2. Implementación del Frontend

   El desarrollo se realizó en Visual Studio Code, empleando HTML5, CSS3 y JavaScript organizados en carpetas modulares para un mantenimiento eficiente.

   Los archivos styles.css y main.js centralizan los estilos y la lógica del sistema.

   Se aplicaron técnicas de diseño responsivo para asegurar una correcta visualización en computadoras, tablets y smartphones.

3. Integración con Firebase

   Se creó un proyecto en Firebase Console, habilitando los servicios Authentication y Firestore Database.

   Luego, se configuró el archivo firebase-config.js con las credenciales del proyecto, permitiendo la conexión directa entre el frontend y el backend en la nube.

   Se implementaron funciones JavaScript para:

   - Registrar y autenticar usuarios.
   - Gestionar roles (administrador y alumno).
   - Guardar y consultar datos de diagnóstico.

     Además, se establecieron reglas de seguridad en Firestore para restringir accesos según los privilegios de usuario.

4. Despliegue del Sistema (enfoque ampliado)

   Una vez completadas las pruebas locales, se procedió al despliegue en la nube utilizando GitHub Pages, aprovechando su integración con Git y su capacidad de hosting estático gratuito.

Pasos realizados:

- Inicialización del repositorio con git init y configuración del control de versiones.
- Subida del proyecto a GitHub mediante comandos git add, git commit y git push.
- Configuración del archivo .gitignore para evitar subir credenciales y archivos temporales.
- Activación del servicio GitHub Pages desde la configuración del repositorio.
- Selección de la rama principal (main o gh-pages) como fuente de publicación.
- Generación automática de la URL pública para el acceso al sistema.

Ventajas técnicas del despliegue en GitHub Pages:

- Eliminación de la necesidad de un servidor físico o VPS.
- Publicación continua: cada cambio subido a GitHub se refleja automáticamente en la versión en línea.
- Integración con herramientas de versionado, facilitando el trabajo colaborativo y la trazabilidad del código.
- Seguridad mediante HTTPS gratuito proporcionado por GitHub.

Mantenimiento y actualizaciones:
Cada actualización del código se gestiona mediante commits, lo que permite mantener un registro histórico de los cambios y facilita la restauración de versiones previas.
Además, el sistema puede ser monitorizado y actualizado en tiempo real sin afectar el acceso de los usuarios finales.

5. Pruebas y Validación
   Se realizaron pruebas funcionales en entornos locales y en línea, verificando:

- Registro y autenticación correctos en Firebase.
- Acceso diferenciado a paneles según el rol.
- Sincronización de datos en Firestore.
- Correcta visualización y funcionamiento en la URL desplegada de GitHub Pages.
- Integridad y disponibilidad del sistema desde distintos dispositivos y navegadores.

IV. ANALISIS E INTERPRETACION DE RESULTADOS

Durante la fase de pruebas y despliegue se comprobó que el sistema podía funcionar íntegramente desde la nube, sin necesidad de infraestructura física.
GitHub Pages demostró ser una herramienta eficaz para la distribución del sistema y la colaboración en equipo, permitiendo una publicación inmediata de los avances.

Principales resultados del despliegue:

- El sistema es accesible desde cualquier ubicación mediante una URL pública.
- Los cambios en el código se sincronizan automáticamente tras cada commit, optimizando el mantenimiento.
- La integración con Firebase garantiza una gestión dinámica de usuarios y diagnósticos sin requerir servidor backend propio.
- La combinación de GitHub Pages + Firebase ofrece una solución escalable, gratuita y segura para proyectos académicos y educativos.

Desafíos detectados:

- Las actualizaciones de seguridad deben controlarse desde Firebase Console.
- El hosting gratuito de GitHub Pages no permite funciones de backend avanzadas (solo archivos estáticos).
- El sistema depende completamente de la conexión a internet y del correcto enlace con Firebase.

El proyecto **TareaWeb2** corresponde a un sitio web estático desarrollado con **HTML5 y CSS3**, sin dependencias externas, destinado a fortalecer los conocimientos fundamentales en diseño y estructura web. Su simplicidad permite ejecutarlo directamente en cualquier navegador moderno sin requerir un servidor dedicado.

Requisitos Previos

No se requieren configuraciones complejas. Es suficiente con contar con un navegador actualizado (Google Chrome, Edge o Firefox).
De manera opcional, puede ejecutarse localmente utilizando herramientas ligeras como:

- **Python 3:** `python -m http.server 8000`
- **Node.js + serve:** `npm install -g serve` y luego `serve -s .`
- **Live Server** (extensión en Visual Studio Code)

Estas opciones facilitan la visualización dinámica del sitio sin necesidad de subirlo a internet.

Ejecución Local

1. Abrir directamente el archivo `index.html` mediante doble clic.

2. O, desde la terminal, navegar hasta la carpeta del proyecto y ejecutar alguno de los comandos anteriores para iniciar un servidor local.

Despliegue en GitHub Pages

Para la publicación del proyecto se utilizó **GitHub Pages**, que permite alojar sitios web estáticos de forma gratuita.
El proceso de despliegue se realizó con los siguientes pasos técnicos:

Inicialización del repositorio local:

```
git init
```

Adición de archivos del proyecto:

```
git add .
```

Confirmación de cambios (commit):

```
git commit -m "Versión inicial del proyecto TareaWeb2"
```

Vinculación con el repositorio remoto:

```
git remote add origin https://github.com/usuario/TareaWeb2.git
```

Subida del proyecto a GitHub:

```
git push -u origin main
```

Activación de GitHub Pages:

- Ingresar al repositorio en GitHub.
- Abrir **Settings** → **Pages**.
- En “Source”, seleccionar la rama **main** y la carpeta raíz (`/root`).
- Guardar los cambios.

GitHub generará automáticamente una **URL pública** (por ejemplo: `https://usuario.github.io/TareaWeb2/`) para acceder al sitio web.

Estructura del Proyecto

- `index.html`: página principal.
- `css/`: hojas de estilo del sitio.
- `assets/`: carpeta con imágenes y recursos gráficos.

Beneficios Técnicos

- Despliegue gratuito y accesible desde cualquier dispositivo.
- Mantenimiento sencillo mediante control de versiones.
- Integración directa con GitHub para colaboración y actualización continua.

V. CUESTIONARIO

1. ¿Qué ventaja ofrece Firebase frente a un servidor tradicional?
   → Permite gestionar autenticación y base de datos sin infraestructura local.

2. ¿Qué función cumple GitHub Pages en el proyecto?
   → Facilita el despliegue gratuito del sistema web.

3. ¿Por qué se eligió una base de datos NoSQL?
   → Porque ofrece flexibilidad en el manejo de documentos sin estructuras fijas.

4. ¿Qué diferencia hay entre los roles de alumno y administrador?
   → El administrador gestiona usuarios y diagnósticos; el alumno solo consulta sus datos.

5. ¿Qué mejoras se proponen para futuras versiones?
   → Integración con hardware en tiempo real, generación de reportes PDF y gráficas dinámicas.

CONCLUSIONES

El desarrollo del Sistema Web de Administración y Diagnóstico Arduino permitió aplicar conceptos de ingeniería de software, bases de datos y tecnologías web modernas.
El uso de Firebase simplificó la gestión del backend, y GitHub Pages facilitó el despliegue sin requerir infraestructura adicional.
El sistema logró cumplir sus objetivos de forma eficiente, demostrando la viabilidad de emplear soluciones en la nube para la educación tecnológica.

RECOMENDACIONES

- Incorporar en futuras versiones la comunicación directa con el hardware Arduino para obtener diagnósticos automáticos.
- Implementar un módulo de exportación de reportes en formato PDF.
- Mejorar la interfaz utilizando frameworks modernos como React o Vue.js.
- Realizar copias de seguridad periódicas de Firestore para evitar pérdida de datos.

BIBLIOGRAFÍA

- Pressman, R. S. (2010). Ingeniería del Software: Un enfoque práctico. McGraw-Hill.
- Sommerville, I. (2011). Software Engineering. Pearson Education.
- Sebesta, R. W. (2014). Concepts of Web Programming. Pearson.