

**Malla/Fácil**

**Documento de Definición**

**de Arquitectura**

**Versión 1.1**

**Integrantes:** David Bravo Aravena

Javier de la Jara Vera

Ailyne Jara Sandoval

**Asignatura:** Capstone

**Sección:** 008v

**Docente:** Jorge Leopoldo Guzman Bozo

**Identificación del Documento**

| **Proyecto** | Malla Fácil |
| --- | --- |
| **Versión** | 1.1 |

| **Documento mantenido por** | Team Malla Fácil / DuocUC |
| --- | --- |
| **Fecha de última revisión** | 29-08-2024 |
| **Fecha de próxima revisión** | 30-08-2024 |

| **Documento aprobado por** | David Bravo Aravena |
| --- | --- |
| **Fecha de última aprobación** | 29-08-2024 |

**Historial de cambios**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 26-08-2024 | 1.0 | Creación del documento. | Team |
| 29-08-2024 | 1.1 | Se agregan nuevas definiciones | David Bravo |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**1. Introducción 4**](#_heading=h.mcaqriv0mule)

[Propósito 4](#_heading=h.50ujebf9l198)

[Alcance 4](#_heading=h.t7rcg77cnmci)

[**2. Arquitectura General 4**](#_heading=h.tfqzsdk2h29k)

[Vista de alto Nivel 4](#_heading=h.oyyrmadhwovd)

[Componentes Principales 4](#_heading=h.oqnxrcdl08vj)

[**3. Frontend 5**](#_heading=h.2limb9dlzrc3)

[Angular 5](#_heading=h.5in78p5boaxk)

[Estructura de Carpetas 5](#_heading=h.ydzm6hqgm6is)

[Módulos y Componentes 5](#_heading=h.jqs2hr2zawo)

[Ionic 6](#_heading=h.a15zgt7826a5)

[Integración con Angular 6](#_heading=h.d1v53wr464xr)

[**4. Backend 6**](#_heading=h.voevbpkty169)

[Node.js con Express 6](#_heading=h.g26l8j1q84hm)

[Estructura de Carpetas 6](#_heading=h.h1n350ap7s9i)

[APIs RESTful 7](#_heading=h.rst60auewqng)

[Autenticación y Seguridad 7](#_heading=h.6rxuhoq8vowe)

[**5. Base de Datos 7**](#_heading=h.y1dfqpkcoumw)

[Selección de la Base de Datos \*(Por definir) 7](#_heading=h.77qomefpuaf9)

[Diseño de la Base de Datos 7](#_heading=h.vf4ffa5guzhn)

[Estructura de la Base de Datos 7](#_heading=h.4gyy2t1mhwr9)

[**6. Integración Continua 8**](#_heading=h.tx4h67dblvp)

[Pipeline de CI/CD 8](#_heading=h.vwk694w56q5q)

[**7. Pruebas 8**](#_heading=h.pcy7akgmmc98)

[Pruebas Unitarias 8](#_heading=h.jhgqdhjmx63)

[Pruebas de Integración 8](#_heading=h.ynj88ceuy183)

[Pruebas End-to-End (E2E) 8](#_heading=h.ylp29p4725s0)

[Pruebas Automatizadas 8](#_heading=h.uds6v3v7ugkm)

[**8. Mantenimiento 9**](#_heading=h.nchaow2cvzok)

[**9. Consideraciones Finales 9**](#_heading=h.gs297o9nmy4f)

[Documentación 9](#_heading=h.74q6ld4lnocq)

# 

# **Introducción**

## Propósito

Definir la arquitectura de la aplicación que será desarrollada para web y mobile. La aplicación utilizará Node.js para el backend, Angular como framework frontend, Ionic para la conversión de la aplicación web a mobile, y APIs desarrolladas internamente. La base de datos será diseñada para soportar las operaciones de la aplicación, con un enfoque en escalabilidad, seguridad y rendimiento.

## Alcance

Se incluye la descripción de los componentes clave de la arquitectura, las tecnologías a utilizar, la interacción entre los diferentes módulos y componentes, y las decisiones de diseño que guiarán el desarrollo de la aplicación.

# **Arquitectura General**

## Vista de alto Nivel

La aplicación será diseñada con una arquitectura basada en microservicios para el backend, utilizando Node.js. El frontend será desarrollado utilizando Angular y convertiremos la aplicación web en mobile utilizando Ionic. La comunicación entre el frontend y el backend se realizará a través de APIS REST.

## Componentes Principales

* **Frontend**: Angular
* **Backend**: Node.js con Express
* **Mobile**: Ionic
* **Base de Datos**: PostgreSQL / MongoDB (según el caso (Por definir))
* **APIs**: APIS REST diseñadas internamente, API Oauth de Google
* **Autenticación**: JWT (JSON Web Tokens), API Oauth de Google
* **Almacenamiento de Archivos**: Local Storage, Cuenta de Google
* **Gestión del Estado**: NgRx (Angular Reactive Extensions)
* **Testing**: Test unitarios, Cucumber.

# **Frontend**

## Angular

El frontend será desarrollado utilizando Angular, framework para la creación de aplicaciones web. Angular proporcionará la estructura y organización del código necesario para manejar la lógica del frontend, servicios, componentes, y la gestión del estado.

## Estructura de Carpetas

* **src/**: Carpeta raíz del proyecto Angular.
  + **app/**: Contiene los módulos, componentes, y servicios de la aplicación.
  + **assets/**: Archivos estáticos como imágenes, fuentes, etc.
  + **modules/**: Agrupa un conjunto de componentes, servicios, y otros recursos relacionados con una funcionalidad específica de la aplicación.
  + **core/**: Servicios singleton, configuración de la aplicación, y otros servicios que deben estar disponibles en toda la aplicación.
  + **shared/**: Componentes, directivas, pipes y servicios reutilizables en toda la aplicación

## Módulos y Componentes

* **Módulos**: Cada funcionalidad principal será encapsulada en un módulo Angular separado.
* **Componentes**: Diseñados para ser reutilizables y siguiendo las mejores prácticas de Angular, esta estructura permite mantener el código modular y organizado.

## Ionic

Ionic se utilizará para convertir la aplicación web en una aplicación mobile híbrida. Esto permitirá desplegar la misma base de código tanto en plataformas web como móviles.

## Integración con Angular

Ionic se integrará con Angular para aprovechar la estructura y lógica existente. Esto incluye la reutilización de servicios y componentes de Angular dentro de la aplicación mobile.

# **Backend**

## Node.js con Express

El backend será desarrollado utilizando Node.js con el framework Express. Esta combinación permitirá crear un servidor eficiente y escalable para manejar las solicitudes de la aplicación.

## Estructura de Carpetas

* **src/**: Carpeta raíz del servidor.
  + **controllers/**: Contiene la lógica de los controladores de la API.
  + **models/**: Modelos de datos para interactuar con la base de datos.
  + **routes/**: Define las rutas y puntos finales de la API.
  + **services/**: Servicios de negocio y lógica de aplicación.

## APIs RESTful

Las APIS serán diseñadas siguiendo principios RESTful, con un enfoque en mantener la escalabilidad, seguridad, y rendimiento.

Se utilizarán APIS propias, para realizar consultas a la base de datos y obtener los datos de las instituciones.

## Autenticación y Seguridad

La autenticación se manejará mediante JWT y también se utilizará la API de Google para autenticación con cuentas de Google. Se implementarán políticas de seguridad para proteger la integridad de los datos y prevenir ataques como XSS, CSRF, y SQL Injection.

# **Base de Datos**

## Selección de la Base de Datos \*(Por definir)

Dependiendo de las necesidades de la aplicación, se optará por:

* **PostgreSQL**: Si se requiere una base de datos relacional con soporte para transacciones complejas.
* **MongoDB**: Si se necesita un esquema flexible y escalabilidad horizontal.

## Diseño de la Base de Datos

Se describirán las entidades principales y sus relaciones. La base de datos será diseñada para optimizar el rendimiento, asegurando integridad y consistencia.

## Estructura de la Base de Datos

* **Tablas / Colecciones**: \*.
* **Relaciones**: \*
* **Índices**: \* Estrategias para mejorar el rendimiento de las consultas.

# **Integración Continua**

## Pipeline de CI/CD

Se establecerá un pipeline de integración continua y despliegue continuo utilizando GitHub Actions.

# **Pruebas**

## Pruebas Unitarias

Las pruebas unitarias se implementarán para cada componente y módulo, utilizando frameworks de prueba específicos de la tecnología (Jasmine/Karma para Angular, Jest para Node.js).

## Pruebas de Integración

Se realizarán pruebas de integración para asegurar que los módulos trabajen correctamente juntos, especialmente la comunicación entre el frontend y backend.

## Pruebas End-to-End (E2E)

Se realizarán pruebas E2E para simular la experiencia del usuario y asegurar que la aplicación funcione correctamente en su totalidad.

## Pruebas Automatizadas

Se implementarán pruebas automatizadas utilizando **Cucumber** con **Gherkin,** definiendo escenarios con lenguaje natural. Además, se utilizará **Selenium WebDriver** para automatizar las interacciones con la interfaz de usuario, asegurando que las funcionalidades críticas funcionen correctamente.

# **Mantenimiento**

Se establecerá un plan de mantenimiento para asegurar la continuidad y actualización de la aplicación, incluyendo parches de seguridad y mejoras de rendimiento.

# **Consideraciones Finales**

## Documentación

Toda la documentación técnica, incluyendo las APIs y la base de datos, será mantenida y actualizada para facilitar el desarrollo futuro y el mantenimiento.