Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA

Fernando Josué Flores Valdez
Asesorado por el Ing. Daniel Oswaldo Pérez Ramírez

Guatemala, febrero de 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

FERNANDO JOSUÉ FLORES VALDEZ
ASESORADO POR EL ING. DANIEL OSWALDO PÉREZ RAMÍREZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, FEBRERO DE 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADORA	Inga. Floriza Felipa Ávila Pesquera de Medinilla
EXAMINADOR	Ing. Sergio Leonel Gómez Bravo
EXAMINADOR	Ing. Carlos Alfredo Azurdia Morales
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DESARROLLO E IMPLEMENTACIÓN DE UNA APLCIACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas con fecha 2 de febrero de 2022

Fernando Josué Flores Valdez

Guatemala, 09 de enero de 2023

Ingeniero
Oscar Argueta Hernández
Director de la Unidad de EPS
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero Argueta:

Por medio de la presente doy por **finalizado satisfactoriamente** el proyecto de EPS titulado: “DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA”. Además, hago de su conocimiento que he **revisado y aprobado el informe final del mismo.**

Dicho proyecto e informe fue elaborado por el estudiante: FERNANDO JOSUÉ FLORES VALDEZ quien se identifica con registro académico 201504385 y código único de identificación 2726 70499 0101, de la carrera en Ciencias y Sistemas de la Facultad de Ingeniería en la Universidad de San Carlos de Guatemala.

Sin otro particular me despido.

Atentamente,

Ing. Daniel Oswaldo Pérez Ramír.

Ing. Daniel Oswaldo Pérez Ramírez
Colegiado No.7467
Asesor de EPS
Escuela de Ingeniería en Ciencias y Sistemas

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 12 de enero de 2023.
REF.EPS.DOC.02.01.2023.

Ing. Oscar Argueta Hernández
Director Unidad de EPS
Facultad de Ingeniería
Presente

Estimado Ingeniero Argueta Hernández:

Por este medio atentamente le informo que como Supervisora de la Práctica del Ejercicio Profesional Supervisado, (E.P.S) del estudiante universitario de la Carrera de Ingeniería en Ciencias y Sistemas, **Fernando Josué Flores Valdez, Registro Académico 201504385 y CUI 2726 70499 0101** procedí a revisar el informe final, cuyo título es **DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA.**

En tal virtud, **LO DOY POR APROBADO**, solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,

"Id y Enseñad a Todos"

Inga. Floriza Felipa Ávila Pesquera de Medinilla
Supervisora de EPS
Área de Ingeniería en Ciencias y Sistemas

FFAPdM/RA

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 12 de enero de 2023.
REF.EPS.D.11.01.2023.

Ing. Carlos Gustavo Alonzo
Director Escuela de Ingeniería Ciencias y Sistemas
Facultad de Ingeniería
Presente

Estimado Ingeniero Alonzo:

Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA,** que fue desarrollado por el estudiante universitario **Fernando Josué Flores Valdez, Registro Académico 201504385 y CUI 2726 70499 0101** quien fue debidamente asesorado por el Ing. Daniel Oswaldo Pérez Ramírez y supervisado por la Inga. Floriza Felipa Ávila Pesquera de Medinilla.

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte del Asesor y la Supervisora de EPS, en mi calidad de Director apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,
"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
DIRECCIÓN
Unidad de Prácticas de Ingeniería y EPS
Facultad de Ingeniería

Ing. Oscar Argueta Hernández
Director Unidad de EPS

/ra

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 19 de enero de 2023

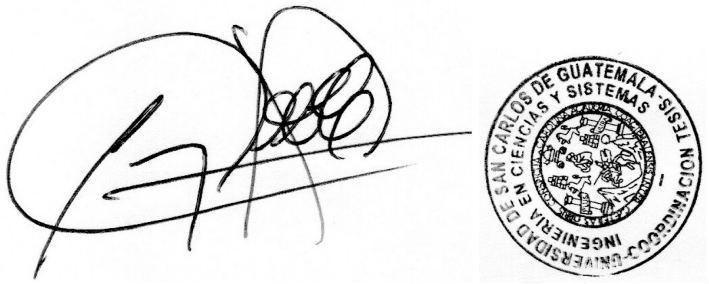
Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación-EPS del estudiante **FERNANDO JOSUÉ FLORES VALDEZ** carné **201504385** y CUI **2726 70499 0101,** titulado: **"DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,



Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LNG.DIRECTOR.043.EICCSS.2023

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador de área y la aprobación del área de lingüística del trabajo de graduación titulado: **DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA**, presentado por: **Fernando Josué Flores Valdez** , procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Msc. Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, febrero de 2023

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.211.2023

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN PWA (PROGRESSIVE WEB APP), PARA LA DESCENTRALIZACIÓN DE LA INFORMACIÓN CONTENIDA EN EL SOFTWARE DATAFORG DEL INSTITUTO NACIONAL DE BOSQUES DE GUATEMALA,** presentado por: **Fernando Josué Flores Valdez,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, febrero de 2023

AACE/gaoc

ACTO QUE DEDICO A:

Mis padres Magdalena Raquel Valdez Duarte y Jorge Ernesto Flores Morales, por su amor incondicional, con el cual me brindaron en todo momento su apoyo moral, emocional y económico a través de muchos esfuerzos y sacrificios.

Mis hermanos Sara Elizabeth y Alejandro José, por su cariño, ejemplo y apoyo moral que siempre me han brindado.

Mis abuelas María Morales y Magda Duarte (q. e. p. d.), que con su sabiduría siempre procuraron guiarme y aconsejarme para no darme por vencido y culminar mis estudios universitarios.

Mi familia Que siempre me demostraron su interés y disposición para apoyarme en cualquier momento de mi carrera.

Mis amigos Quienes dentro de los salones del T3, oficinas de la FAUSAC y a las afueras de la USAC, me brindaron sus consejos y apoyo siempre que fue necesario.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Casa de estudios que me brindo la oportunidad de acceder a mi formación académica universitaria.
Instituto Nacional de Bosques de Guatemala	Institución que me abrió sus puertas y siempre estuvo en completa disposición para apoyarme durante el desarrollo de mi trabajo de graduación.
Mis amigos	Quienes estuvieron en los buenos y malos momentos de mi vida universitaria, por su ayuda y apoyo moral para superar todos los obstáculos que se presentaron durante mi carrera.
Ingeniero	Daniel Pérez por su amistad, tiempo, apoyo y asesoría técnica en este trabajo de graduación y cualquier otro trabajo de mi carrera universitaria.
Licenciado	Luis Siney, por brindarme todas las herramientas y apoyo para desarrollar de la mejor manera mi trabajo de graduación dentro del INAB.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES VII
LISTA DE SÍMBOLOS IX
GLOSARIO XI
RESUMEN XV
OBJETIVOS XVII
INTRODUCCIÓN XIX

1. FASE DE INVESTIGACIÓN 1
1.1. Antecedentes de la empresa 1
1.1.1. Reseña histórica 1
1.1.2. Misión 1
1.1.3. Visión 1
1.1.4. Objetivos 2
1.1.4.1. Objetivo General 2
1.1.4.2. Objetivo Estratégico 1. Ambiental 2
1.1.4.3. Objetivo Estratégico 2. Económico 2
1.1.4.4. Objetivo Estratégico 3. Social 3
1.1.4.5. Objetivo Estratégico 4. Institucional 3
1.1.4.6. Credo Institucional 3
1.1.5. Servicios que brinda 4
1.1.5.1. Sistemas informáticos que pone a disposición 5
1.2. Descripción de las necesidades 6
1.3. Priorización de las necesidades 6
1.4. Diagnóstico FODA del proyecto 8

1.4.1. Análisis Interno 8
1.4.1.1. Fortalezas 8
1.4.1.2. Debilidades 8
1.4.2. Análisis Externo 9
1.4.2.1. Fortalezas 9
1.4.2.2. Debilidades 9

2. FASE TÉCNICO PROFESIONAL 11
2.1. Ingeniería y Software 11
2.1.1. Ingeniería 11
2.1.2. Software 11
2.1.3. Ingeniería de software 11
2.2. Detalles de Dataforg 12
2.2.1. Tecnologías utilizadas en Dataforg 12
2.3. Aplicaciones de software 13
2.3.1. Aplicaciones móviles 13
2.3.1.1. Aplicaciones móviles nativas 14
2.3.1.2. Aplicaciones móviles hibridas 15
2.4. Aplicaciones Web progresivas 17
2.4.1. Características mínimas de una PWA 17
2.4.1.1. Contexto seguro HTTPS 17
2.4.1.2. Service workers 18
2.4.1.3. Ciclo de vida de un Service Worker 19
2.4.1.4. Archivo Manifest 20
2.4.2. Ventajas de PWA 21
2.4.2.1. Reconocible 21
2.4.2.2. Instalable 22
2.4.2.3. Enlazable 22
2.4.2.4. Independiente de la red 22

2.4.2.5. Compatibilidad de mejora progresiva .. 22
2.4.2.6. Reconectable 23
2.4.2.7. Adaptable 23
2.4.2.8. Segura 23
2.5. WebAssembly 24
2.6. Blazor Framework 24
2.6.1. Componentes 24
2.6.2. Blazor Server 27
2.6.3. Blazor WebAssembly 28
2.7. Herramientas seleccionadas 30
2.7.1. Blazor WebAssembly 30
2.7.2. SQL Server 32
2.7.3. C# y .NET core 33
2.7.4. Entity Framework 33
2.8. Arquitectura de PWA Dataforg 34
2.8.1. Arquitectura PWA Dataforg cliente 34
2.8.2. Arquitectura Dataforg server 35
2.9. Cliente del proyecto 37
2.9.1. Diseño de la aplicación 37
2.9.1.1. Filtros 38
2.9.1.2. Búsquedas 41
2.9.1.3. Vista individual de una especie 43
2.9.1.4. Terminología forestal 46
2.9.1.5. Referencias bibliográficas 47
2.9.1.6. Directorio de empresas 48
2.9.2. Estructura de archivos aplicación cliente 50
2.9.3. Distribución de componentes 52
2.9.3.1. Componentes de uso compartido 53
2.9.3.2. Componentes específicos 54

2.9.4. Información en cache 54
2.9.4.1. Administración de cache en service worker 55
2.9.5. Service Worker 55
2.9.5.1. Función onFetch 55
2.9.5.2. Separación de caches 56
2.9.6. Impresión de reportes 56
2.10. Servidor del proyecto 57
2.10.1. Estructura de archivos servidor 57
2.10.2. Estructura del directorio Server 59
2.10.3. Estructura del directorio Shared 62
2.10.4. Controladores 63
2.10.5. Servicios 65
2.10.6. Sitio Administrativo CRUD 65
2.11. Base de datos 66
2.11.1. Modelo entidad relación de la base de datos 66
2.12. Ambientes de la aplicación 66
2.12.1. Desarrollo 66
2.12.2. Pruebas y producción 67
2.13. Atención a los requerimientos 67
2.13.1. Evaluación de objetivos 68
2.13.2. Ventajas del proyecto 72

3. FASE DE ENSEÑANZA – APRENDIZAJE 75
3.1. Estrategia de enseñanza 75
3.2. Fase de pruebas 75

CONCLUSIONES 77
RECOMENDACIONES 79

REFERENCIAS 81
APÉNDICES 85

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Organigrama del INAB 4
2. Arquitectura anterior de Software Dataforg 12
3. Funcionamiento del contexto seguro HTTPS 18
4. Ciclo de vida de un Service Worker 19
5. Estructura básica de un archivo Manifest 21
6. Estructura y sintaxis básica de un componente Razor. 26
7. Funcionamiento de Blazor Server y SignalR 27
8. Comportamiento de Blazor y WebAssembly 29
9. Arquitectura PWA Dataforg cliente 35
10. Arquitectura Dataforg server 37
11. Diseño de filtros del prototipo 38
12. Diseño de filtros de la aplicación 39
13. Diseño de filtros de Dataforg existente 39
14. Diseño de resultados de una búsqueda del prototipo 41
15. Diseño de resultados de una búsqueda de la aplicación 41
16. Diseño de resultados de una búsqueda de Dataforg existente 42
17. Diseño de la vista de una especie del prototipo 43
18. Diseño de la vista de una especie de la aplicación 44
19. Diseño de la vista de una especie de Dataforg existente 44
20. Diseño de tarjeta colapsable para una característica especie 46
21. Diseño de la vista de un término forestal 47
22. Diseño de la vista de una referencia bibliográfica 48
23. Diseño de la vista de todas las empresas 49

24. Diseño de la vista de una empresa 50
25. Estructura de archivos aplicación cliente 51
26. Estructura de archivos en el servidor 58
27. Configuración de API para aplicación cliente 60
28. Configuración de Blazor 61
29. Configuración de servicios internos CRUD 61
30. Configuración de aplicación administrativa 62
31. Visualización de filtrado y búsqueda de información 70

TABLAS

1. Categorización de necesidades 7
2. Ventajas y desventajas de aplicaciones móviles nativas 15
3. Ventajas y desventajas de aplicaciones móviles hibridas. 16

LISTA DE SÍMBOLOS

Símbolo	Significado
GB	Gigabyte
Ghz	Giga Hertz
TB	Terabytes

GLOSARIO

API	Mecanismos utilizados para establecer comunicación entre dos componentes de software, a través de un conjunto de definiciones y protocolos.
Blazor	Es un framework de desarrollo para Single Page Application, su nombre es una combinación de los términos Browser y Razor, este tipo de aplicaciones permiten ejecutar las vistas en el lado cliente haciendo uso de WebAssembly, reduciendo la carga del servidor.
Google Meet	Herramienta de videoconferencia desarrollada por Google, esta permite agendar reuniones en Google Calendar, realizarlas de forma instantánea y si el proveedor de correo lo permite, grabar y almacenar la grabación de cada llamada.
INAB	Instituto Nacional de Bosques de Guatemala.
Kanban	Es un método de gestión de flujo de trabajo, se utiliza para definir, gestionar y mejorar los servicios. Su objetivo es maximizar la eficiencia y procurar la mejora continua.
MANEFOR	Sistema de Manejo Forestal.

Modelo ER	Es un modelo que permite describir a detalle conceptos interrelacionados dentro de un dominio conocido. El modelo básico está compuesto por dos cosas: entidades y relaciones. En base de datos estas entidades representan conceptos u objetos que pertenecen a un sistema, y sus relaciones nos indican qué tipo de interacción tienen con las demás entidades en el sistema.
MVP	Producto Mínimo Viable, se refiere al producto que contiene las suficientes características que cubren la mayor cantidad de requerimientos iniciales, estas funciones pueden ser consideradas como el mínimo para poder lanzar el producto al público.
PINPEP	Programa de incentivos forestales para poseedores de pequeñas extensiones de tierra de vocación forestal o agroforestal.
PROBOSQUE	Sistema de información de bosques proporcionado por el INAB.
Product Backlog	Listado de todas las tareas derivadas de los requerimientos iniciales y que deben ser completadas durante todo el periodo de desarrollo acordado. Este debe ser visible para todo el equipo para facilitar la visión del proyecto y dimensionar de una mejor forma el esfuerzo necesario para concluir cada tarea.

PWA	Es una aplicación desarrollada con tecnologías web básicas, HTML, CSS, JavaScript, y WebAssembly, el objetivo es que pueda ser utilizada en cualquier plataforma. Independientemente del dispositivo o sistema operativo, siempre que utilice un navegador web, puede ser utilizada, la diferencia con una aplicación web estándar, es que esta permite ser instalada y añadida a la pantalla de inicio tanto en escritorio como en dispositivos móviles. Al ser una aplicación que se despliega de igual manera que una aplicación web normal, permite ahorrar tiempos de publicación, desarrollo y aprobación de las tiendas.
Scrum	Metodología ágil que utiliza sesiones y artefactos que permiten llevar a cabo un desarrollo constante y flexible, de esta manera podemos satisfacer las necesidades del usuario y definir prioridades.
SEINEF	Sistema Electrónico de Información de Empresas Forestales.
SIFGUA	Sistema de información general de Guatemala.
Single Page Application	Son páginas web que dan una experiencia de usuario distinta, dando la apariencia de que todo ocurre dentro de la misma página inicial, esto permite hacer desarrollo de aplicaciones web que dan la apariencia de ser ejecutadas en un dispositivo de forma nativa.

	Esto es muy útil al realizar desarrollo de aplicaciones cross-platform con tecnologías web comunes.
WebAssembly	Es un tipo de código que puede ser ejecutado en navegadores web modernos, trabaja a bajo nivel y en un formato binario compacto, su rendimiento es casi igual al nativo, por lo cual brinda mayor versatilidad. Es capaz de proporcionar lenguajes como c/c++, c# y rust. Este puede ser ejecutado con JavaScript, por lo cual no existe limitante en cuanto a cuál utilizar, podemos elegir cuál de los dos aporta una mejor solución en un momento dado.
WEBMAIL	Plataforma de correos del INAB.

RESUMEN

La institución ya cuenta con un software para distribución desarrollado en java, el cual opera de forma local y debe ser instalado en la computadora del usuario que desea tener acceso a la base de datos en cuestión, esto representa una limitación para todos los usuarios que no cuentan con una computadora personal o que no cumplen con los requerimientos mínimos para poder operar con este sistema. Se pretende que esta Base de datos se pueda publicar de forma que pueda ser accedida desde cualquier dispositivo en cualquier momento, descentralizando así la consulta de la información. Para ello se utilizará la información y base de datos que la institución provea, y de no existir se implementará una base de datos que almacene toda la información relacionada a las especies forestales de Guatemala que el Instituto Nacional de Bosques proporcione, posterior a ello se deberá definir, desarrollar y desplegar los servicios web necesarios para establecer el acceso y consulta a base de datos y por último el desarrollo y despliegue de la aplicación web progresiva a través de los cuales el usuario final pueda interactuar con la información ya existente.

Esta solución tomará las funcionalidades, módulos y características que el software ya existente posee, y las adaptará al formato adecuado que permita a los usuarios interactuar con la información de una manera fácil y rápida. La aplicación contará con un diseño responsive, de tal forma que su despliegue sea independiente al tamaño de pantalla desde el cual se está accediendo, es decir que se adapte a cualquier dimensión y la visualización e interacción no se vea comprometida, afectando el uso para el usuario. El diseño de la aplicación se enfocará en brindar una estética que se asemeje a los sistemas de consulta de

datos e información como lo son las redes sociales, para evitar el rechazo del usuario final o una curva de adaptación demasiado lenta que limite su uso.

OBJETIVOS

General

Desarrollar una aplicación multiplataforma que permita la descentralización de la información de especies forestales de Guatemala, contenida en la herramienta Dataforg del Instituto Nacional de Bosques de Guatemala.

Específicos

1. Diseñar los modelos entidad relación a partir de los cuales se crearán las bases de datos, que se utilizarán para almacenar la información con la que los usuarios van a interactuar.

2. Desarrollar versión web y móvil (PWA) de los módulos del software Dataforg: búsquedas y filtrado de especies forestales, empresas forestales, términos forestales y referencias bibliográficas. Haciendo uso de los frameworks .NET y Blazor.

3. Implementar uso de memoria cache para permitir al usuario final guardar de forma local resultados de búsquedas en el sistema, y posteriormente consultar esta información aún sin conexión a internet.

4. Desarrollar una plantilla de documento que permita al usuario final visualizar y exportar los resultados de búsquedas en la aplicación, en formato PDF.

5. Programar la lógica de la fórmula para el cálculo de carbono de especies forestales e integración de esta como un nuevo módulo dentro del software Dataforg.

INTRODUCCIÓN

En repetidas ocasiones vemos como sistemas y aplicaciones que fueron diseñados en el pasado, cumplieron con todas las expectativas y fueron bastante útiles; sin embargo, con el pasar de los años una gran cantidad de este software va quedando obsoleto, esto puede deberse a diferentes factores, siendo uno de los factores más comunes el poco o nulo mantenimiento y actualización del software.

Esta es la situación a la cual se enfrenta el Software de Dataforg, desarrollado por el Instituto Nacional de Bosques de Guatemala. Actualmente cuentan con una primera versión de su producto, el cual está orientado especialmente a todas las personas que realizan consultas e investigación acerca de las especies forestales en el territorio nacional, en ella se pueden visualizar y filtrar amplios catálogos de información recolectada y clasificada por la misma institución, sin embargo esta herramienta está disponible únicamente para sistema operativo Windows en versión de escritorio, esto dificulta la promoción y el uso de la misma ya que las necesidades actuales son distintas a las que en su momento pudo presentar el grupo objetivo.

Hoy en día, una aplicación de software que no está disponible en versión web o móvil, prácticamente es inútil ya que su uso se vuelve sumamente limitado. Como respuesta y haciendo uso de la tecnología actual, se desarrollará una PWA, la cual es ensamblada haciendo uso de tecnologías web comunes tales como HTML, CSS, JavaScript y WebAssembly. Un PWA nos permite utilizar la aplicación desde el navegador móvil y como valor agregado le da al usuario la posibilidad de poder instalar esta aplicación en su dispositivo personal, como si

se tratara de cualquier aplicación nativa, permitiendo así hacer uso del almacenamiento nativo para poder guardar información importante y consultarla posteriormente aun sin conexión a internet.

El uso de PWA facilitará el acceso del software al usuario final, y también el lanzamiento de las actualizaciones de la aplicación a los desarrolladores de la institución. La herramienta proporcionará las mismas funcionalidades que la aplicación de escritorio presentaba anteriormente, también contará con dos nuevos módulos relacionados a la temática del software.

Esta aplicación permitirá a la institución brindar un mejor servicio a sus usuarios y a la población en general, ya que podrá ser instalada por cualquier persona en un dispositivo portátil sin requerimientos mínimos elevados; por otro lado, la facilidad de poder guardar información para su consulta sin conexión internet, brindará un gran valor agregado, ya que en la mayoría de los casos esta información se debe consultar en campo abierto, donde la conexión a internet puede ser inestable o inexistente.

1. FASE DE INVESTIGACIÓN

1.1. Antecedentes de la empresa

1.1.1. Reseña histórica

Institución comprometida a trabajar para el Sector Forestal de Guatemala desde el año 1996. El INAB es una entidad estatal, autónoma, descentralizada, con personalidad jurídica, patrimonio propio e independencia administrativa, y es el órgano de dirección y autoridad competente del sector público agrícola en materia forestal.

1.1.2. Misión

Ejecutar y promover los instrumentos de política forestal nacional, facilitando el acceso a los servicios forestales que presta la institución a los actores del sector forestal, mediante el diseño e impulso de programas, estrategias y acciones, que generen un mayor desarrollo económico, ambiental y social del país.

1.1.3. Visión

El Instituto Nacional de Bosques es una institución líder y modelo en la gestión de la política forestal nacional, reconocida nacional e internacionalmente por su contribución al desarrollo sostenible del sector forestal en Guatemala, propiciando mejora en la economía y en la calidad de vida de su población, y en la reducción de la vulnerabilidad al cambio climático.

1.1.4. Objetivos

1.1.4.1. Objetivo General

Promover el desarrollo forestal del país y contribuir al desarrollo rural integral, a través del fomento al manejo sostenible y restauración de los bosques y tierras forestales, el fortalecimiento de la gobernanza forestal y la vinculación bosques-industria-mercado.

1.1.4.2. Objetivo Estratégico 1. Ambiental

Promover el manejo de los bosques del país, fomentando y regulando su uso sostenible, protección y restauración, como mecanismo para garantizar su permanencia, recuperación y mejora de su productividad, incrementando la provisión de bienes y servicios para garantizar los medios de vida a la sociedad y contribuir con la reducción de la vulnerabilidad del país a los efectos del cambio climático.

1.1.4.3. Objetivo Estratégico 2. Económico

Contribuir al desarrollo económico y social del país, impulsando la vinculación del bosque a la industria forestal y el mercado, como mecanismo para lograr mayor valor agregado de los productos forestales e incrementar la inversión y generación de empleo, y que se reconozca el aporte del sector forestal a la economía nacional.

1.1.4.4. Objetivo Estratégico 3. Social

Fortalecer la gobernanza forestal consolidando alianzas con los gobiernos y organizaciones locales, para promover el vínculo de los bienes y servicios del bosque con el desarrollo social, fomentando la cultura forestal, incrementando la legalidad y reduciendo la conflictividad en torno al uso del bosque.

1.1.4.5. Objetivo Estratégico 4. Institucional

Fortalecer la modernización institucional y las competencias del recurso humano, orientado en un modelo de gestión de calidad basado en resultados, que garanticen la eficiencia institucional para satisfacer la demanda social, y la prestación de un servicio de calidad al usuario.

1.1.4.6. Credo Institucional

Creemos en:

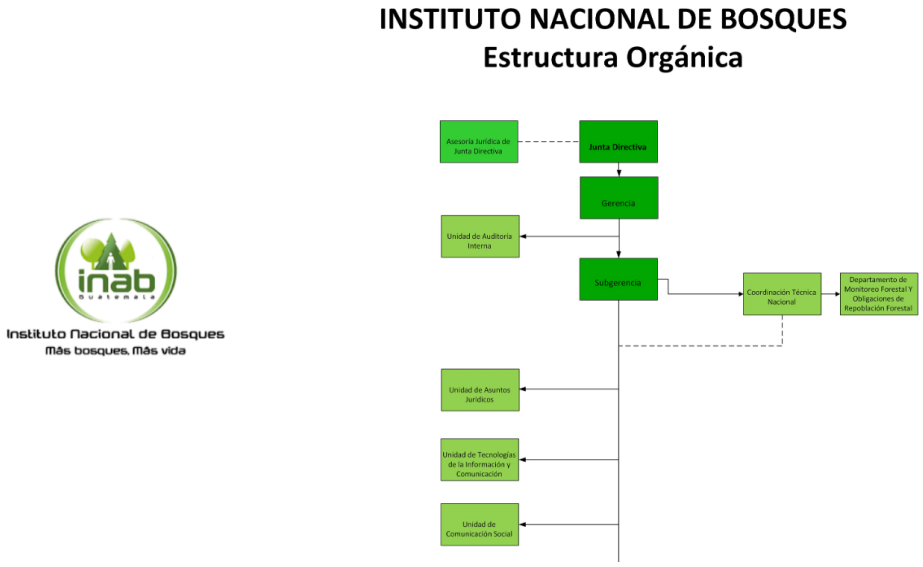
- La importancia del bosque como generador de bienes y servicios ambientales para la sociedad guatemalteca.
- La incorporación del bosque a la actividad productiva, bajo el principio de manejo sostenible, constituye la mejor alternativa para su valorización y conservación.
- La aplicación de normas y procedimientos claros facilita la inversión para promover el desarrollo del sector forestal y la generación de empleo.
- El recurso humano capacitado y sus valores de transparencia, honestidad, responsabilidad, disciplina, creatividad, innovación, dinamismo y perseverancia son bases fundamentales para alcanzar el cumplimiento de nuestra misión.

Las ventajas comparativas de nuestro país son elementos clave para convertir al sector forestal en motor de la economía nacional, contribuyendo al desarrollo rural integral.

1.1.5. Servicios que brinda

El Instituto nacional de bosques de Guatemala se encuentra organizado de la siguiente forma:

Figura 1. **Organigrama del INAB**



Fuente: Instituto Nacional de Bosques (2022). *Organigrama*. Consultado el 03 de mayo de 2022. Recuperado de https://www.inab.gob.gt/images/acercadeinab/organigrama/Propuesta_Organigrama.jpg

Este proyecto pertenece a la unidad de Tecnologías de la Información y Comunicación, quien es la encargada de brindar soporte y realizar el desarrollo

de proyectos de software dentro del instituto con el objetivo de sistematizar la publicación y acceso a información relacionadas a dicha institución.

1.1.5.1. Sistemas informáticos que pone a disposición

- SEINEF
- Manejo forestal
- Información al usuario sobre licencias de aprovechamiento forestal
- PROBOSQUE
- Especies forestales
- Registro nacional forestal
- PINPEP
- MANEFOR - SIFGUA
- Geo portal SIG
- Estado de conservación de los bosques
- Calendario forestal
- Mercado forestal guatemalteco
- Calculadora de carbono
- Sistema de información basado en parcelas permanentes
- Consulta nota de envió
- Mercados forestales
- Sistema de obligaciones de repoblación forestal
- Investigación forestal
- WEBMAIL
- Centro de información forestal

1.2. Descripción de las necesidades

Con el objetivo de identificar de forma correcta las necesidades de la institución se tuvieron reuniones constantes con el encargado del área de tecnología del instituto nacional de bosques INAB, al concluir el proceso de entrevista y detección de necesidades se pudieron encontrar las siguientes:

- Una aplicación que permita ser utilizada en cualquier sistema operativo, sin la necesidad de ser descargada de alguna tienda oficial de distribución de aplicaciones como App Store o Google Play Store.
- Descentralizar los datos ya existentes acerca de especies forestales y datos relacionados a ellas como empresas y terminología.
- Migrar por completo la base de datos local de especies a una base de datos en una máquina virtual dentro de un servidor on premise.
- Desarrollar y publicar los servicios web necesarios para que el usuario final, a través de la aplicación móvil pueda consultar y filtrar la información que desee.
- Generar informes en formato PDF, generados a partir del filtrado de información que el usuario final realice y solicite.

Añadir nuevos módulos al software Dataforg, los cuales están estrechamente relacionados con el estudio de especies forestales: calculadora de carbono y paquetes tecnológicos.

1.3. Priorización de las necesidades

Con el objetivo de tener una mejor categorización y priorización se hizo uso de la técnica MoSCoW, esta técnica utiliza 4 categorías con las cuales podemos clasificar los requisitos y ordenarlos de mayor a menor:

- Must: estos requisitos no son negociables, deben estar cubiertos por el producto final ya que de lo contrario no se considerará un producto que pueda ser utilizado por el usuario final.
- Sould: son requisitos que deben estar en el producto final ya que le garantizan una buena experiencia de usuario a todos los que hagan uso de la herramienta.
- Could: la ausencia de estos requisitos no compromete la usabilidad, rendimiento o experiencia de usuario del producto, sin embargo, su desarrollo aportará valor agregado a la herramienta.
- Would/Won´t: estos requisitos son de muy baja prioridad, su aporte al producto o herramienta es muy poco y sin ellas el producto ya puede considerarse de utilidad para el usuario final.

Las necesidades generales se categorizan de la siguiente forma:

Tabla I. **Categorización de necesidades**

Necesidad	**Categoría**
Migración de base de datos existente	Must.
Visualización de Especies forestales	Must.
Visualización de Empresas forestales	Must.
Visualización de Referencias Bibliográficas	Must.
Visualización de Terminología Forestal	Must.
Funcionamiento de la aplicación sin conexión a internet	Must.
Documentación de aplicación (técnica y de usuario)	Must.
Generación de Reportes PDF	Should.
Integración de Calculadora de carbono	Should.
Integración de Paquetes Tecnológicos	Should.
Iconos y animaciones dinámicas y atractivas	Could.

Fuente: elaboración propia.

1.4. Diagnóstico FODA del proyecto

1.4.1. Análisis Interno

1.4.1.1. Fortalezas

- Disposición de infraestructura física lista para su uso, proporcionada por la institución
- Base de datos existente, con información y catálogos detallados previamente relacionados a través de una propuesta de modelo entidad relación.
- Herramienta de software ya existente, a pesar de ser una versión que funciona localmente en versión de escritorio, nos permite obtener una visión más completa de la idea inicial y hacia donde se desea evolucionar.
- Apertura a la descentralización de la información por parte de los interesados dentro de la institución.
- La institución ya cuenta con servicios en línea, es decir que se encuentran familiarizados con este tipo de productos y lo más importante, una infraestructura mínima dedicada a la distribución de estos.

1.4.1.2. Debilidades

- La infraestructura ya existente podría no estar a completa disposición.
- Aún no se cuenta con ningún servicio en plataforma Móvil.
- Podría existir un conocimiento limitado en cuanto a la administración y distribución de servicios en dispositivos móviles.

1.4.2. Análisis Externo

1.4.2.1. Fortalezas

Alta disponibilidad de sistemas de software en el sitio oficial de la institución.

1.4.2.2. Debilidades

Resistencia al cambio por parte de los usuarios finales de la herramienta de software.

2. FASE TÉCNICO PROFESIONAL

2.1. Ingeniería y Software

2.1.1. Ingeniería

Es la aplicación de conocimientos técnicos y científicos para diseñar e implementar de soluciones a problemáticas de cualquier tipo.

2.1.2. Software

Herramienta conformada por uno o varios conjuntos de programas y rutinas, que le permiten a la computadora llevar a cabo determinadas tareas.

2.1.3. Ingeniería de software

La ingeniería de software es la aplicación de conocimientos técnicos y científicos en la construcción y desarrollo de herramientas de computación, así también en la elaboración de la documentación necesaria para utilizar y dar mantenimiento a estos programas.

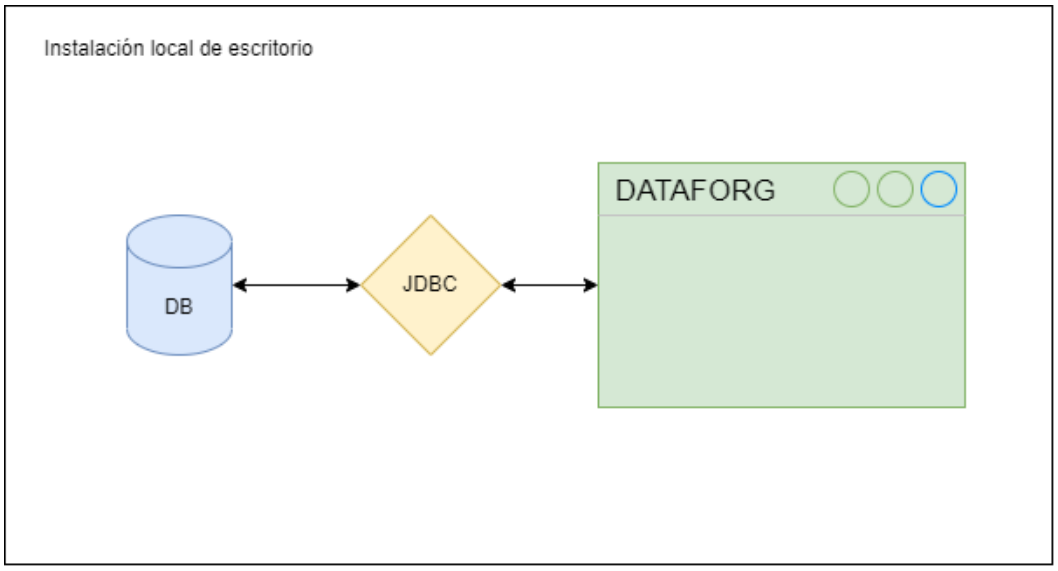
2.2. Detalles de Dataforg

2.2.1. Tecnologías utilizadas en Dataforg

JAVA: lenguaje de programación orientado a objetos utilizado para el desarrollo de la lógica y formulario de escritorio para visualización, filtrado y consulta de datos.

HSQLDB: sistema de base de datos relacional, escrito en java. Entre sus ventajas se pueden mencionar, su velocidad multihilo y que es una base de datos transaccional con opción a ser utilizada en memoria tipo volátil y RAM o en disco duro.

Figura 2. **Arquitectura anterior de Software Dataforg**



Fuente: elaboración propia, realizado con diagrams.net.

2.3. Aplicaciones de software

Una aplicación de software es una herramienta de computación diseñada específicamente para dar solución a una necesidad o problemática en concreto; por lo regular estas aplicaciones están dirigidas al usuario final. Estas son diseñadas y comercializadas por organizaciones ajenas al fabricante del sistema operativo principal del ordenador, se instalan por el usuario y son totalmente opcionales; en otras palabras, el correcto funcionamiento del ordenador no se ve afectado por la ausencia de estas.

Muchas veces es común encontrar aplicaciones preinstaladas en las computadoras junto al sistema operativo principal, aun cuando se trata de un equipo totalmente nuevo. La razón por la que esto sucede es que los fabricantes de software pueden tener acuerdos con los desarrolladores del sistema operativo o fabricantes del equipo; algunos ejemplos de aplicaciones son los Procesadores de Texto y Hojas de cálculo.

Al analizar detenidamente se puede concluir, por ejemplo, que un procesador de texto no es indispensable para que un ordenador funcione correctamente, pero por otro lado es necesario para un usuario que en su día laboral debe redactar documentos y reproducirlos.

2.3.1. Aplicaciones móviles

Una aplicación móvil, comúnmente conocida como app es una herramienta de software diseñada para equipos portátiles, tales como teléfonos inteligentes, tabletas inteligentes y en algunos casos ordenadores portátiles. La razón por la que existen y su demanda aumenta con mayor frecuencia, es que el uso de dispositivos portátiles es más común cada día, y a pesar de que su

capacidad de computación, memoria y procesamiento aumenta de forma exponencial, aun no se puede comparar con la capacidad de un ordenador de escritorio o portátil para ejecutar varias tareas simultaneas.

Otra desventaja de los dispositivos móviles que es bastante evidente, es el reducido espacio en pantalla con el que el usuario cuenta; a diferencia de una pantalla o monitor con el que regularmente se interactúa con un ordenador de escritorio, el espacio para interactuar con las herramientas en los celulares y tabletas es bastante reducido, esto obliga a los desarrolladores a pensar en soluciones que permitan aumentar el uso de los dispositivos sin dejar de lado la accesibilidad y facilidad para manipular las aplicaciones de software.

Cuando se habla de aplicaciones móviles podemos agruparlas en dos conjuntos, las aplicaciones nativas y las aplicaciones web.

2.3.1.1. Aplicaciones móviles nativas

Una aplicación nativa es desarrollada y compilada para ser ejecutada por un sistema operativo en específico, esto permite garantizar un mejor rendimiento y fluidez de la herramienta. Por otro lado, es importante considerar que este tipo de aplicaciones requieren de un mayor uso de tiempo y recursos para poder ser desarrolladas, ya que se debe diseñar y elaborar una versión del mismo producto para cada tipo de sistema operativo, en lenguajes y herramientas de programación diferentes.

Tabla II. **Ventajas y desventajas de aplicaciones móviles nativas**

Ventajas	Desventajas
Las aplicaciones nativas tienden a ser más rápidas y ofrecer un rango más amplio de funcionalidades	Lanzar las mismas funciones en todas las plataformas al mismo tiempo es más difícil y consume mucho tiempo.
Proporcionan un mejor rendimiento de software y tienden a tener un mejor manejo de las caídas de conexión a internet.	Involucran un mayor costo como lo son las distintas habilidades con las que los desarrolladores deben contar para desarrollar la misma aplicación en todas las distintas plataformas.
Permiten tener una apariencia y sensación más reconocible.	Se consume mucho tiempo y esfuerzo de forma separada en el desarrollo para cada una de las plataformas.
Aplicaciones de juego o con una fuerte carga de animaciones tienen un mejor rendimiento	Cada desarrollo cuenta con su propio ciclo de lanzamiento y actualizaciones, lo que eventualmente repercutirá en un alto costo de tiempo de desarrollo.
Mantiene las proporciones para una mejor calidad de imagen y gráficos junto a la aplicación	Las aplicaciones nativas pueden tomar un mayor tiempo de descarga, lo que podría causar la perdida de potenciales usuarios.
Tienen algunas dependencias de plataforma, lo cual hace que sea más sencillo lidiar con requerimientos.	La flexibilidad es limitada comparada con las aplicaciones hibridas.

Fuente: Cynoteck, Subodh Dharmwan (2021). Aplicaciones híbridas frente a aplicaciones nativas: la lista de verificación para descubrir cuál es la adecuada para usted. Consulta el 4 de mayo de 2022. Recuperado de https://cynoteck.com/wp-content/uploads/2021/03/native-app-492x1024.png

2.3.1.2. Aplicaciones móviles hibridas

Las aplicaciones Web, son todas aquellas que se acceden y utilizan desde el navegador, por lo regular están elaboradas con HTML, CSS y Javascript, lo cual permite garantizar el bajo consumo de recursos y memoria del sistema operativo. Este tipo de aplicación puede ser accesible desde cualquier dispositivo

que soporte un navegador web, sin embargo, si la aplicación no cuenta con las configuraciones y módulos especializados, es común que se requiera una conexión estable a internet y su interacción con el hardware del dispositivo sea muy limitada.

Tabla III. **Ventajas y desventajas de aplicaciones móviles hibridas.**

Ventajas	Desventajas
Utilizan un solo código base para todas las plataformas. El código debe escribirse una sola vez.	Aplicaciones con gráficos avanzados, como 3D, juegos HD, podrían verse comprometidos.
En la mayoría de las ocasiones el rendimiento de hardware pude ser similar al de una aplicación nativa.	Por la naturaleza de estas aplicaciones, la apariencia de esta puede variar de un usuario a otro.
Pueden ahorrar tiempo y dinero, aun produciendo distintas versiones de la misma aplicación.	Algunas funciones de los dispositivos más actuales pueden no estar disponibles para ser utilizadas debido a que el lanzamiento de plugins híbridos no es inmediato.
Las aplicaciones hibridas ofrecen una experiencia de usuario casi idéntica en las distintas plataformas.	Contienen muchas dependencias de frameworks y librerías.
Se pueden acceder con o sin conexión a internet	En ocasiones es necesario desarrollar en distintas ramas, debido a que las distintas plataformas también tienen distintos componentes de hardware, puede incrementar los costos de tiempo y desarrollo
Están basadas en tecnologías web, por ello algunas de ellas pueden ser accedidas desde un navegador web	

Fuente: Cynoteck, Subodh Dharmwan (2021). Aplicaciones híbridas frente a aplicaciones nativas: la lista de verificación para descubrir cuál es la adecuada para usted. Consulta el 4 de mayo de 2022. Recuperado de https://cynoteck.com/wp-content/uploads/2021/03/Hybrid-apps-492x1024.png

2.4. Aplicaciones Web progresivas

Una aplicación web progresiva es una aplicación web que incorporan de APIs y funciones del navegador web, y que acompañada del uso de estrategias de mejora progresiva buscan ofrecer el comportamiento de una aplicación nativa. Aunque no existe un estándar formalizado para el desarrollo de este tipo de aplicaciones, podemos definirlo como un patrón de diseño.

Este patrón de diseño se puede comparar con AJAX, ya que ambos hacen uso de un conjunto de atributos de aplicación, incluyendo tecnologías y técnicas web específicas.

Para considerar una aplicación web como una PWA, debe cumplir con un conjunto de características específicas, las cuales deben ser como mínimo:

- Contexto seguro HTTPS
- Uno o más Service Workers
- Un archivo Manifest

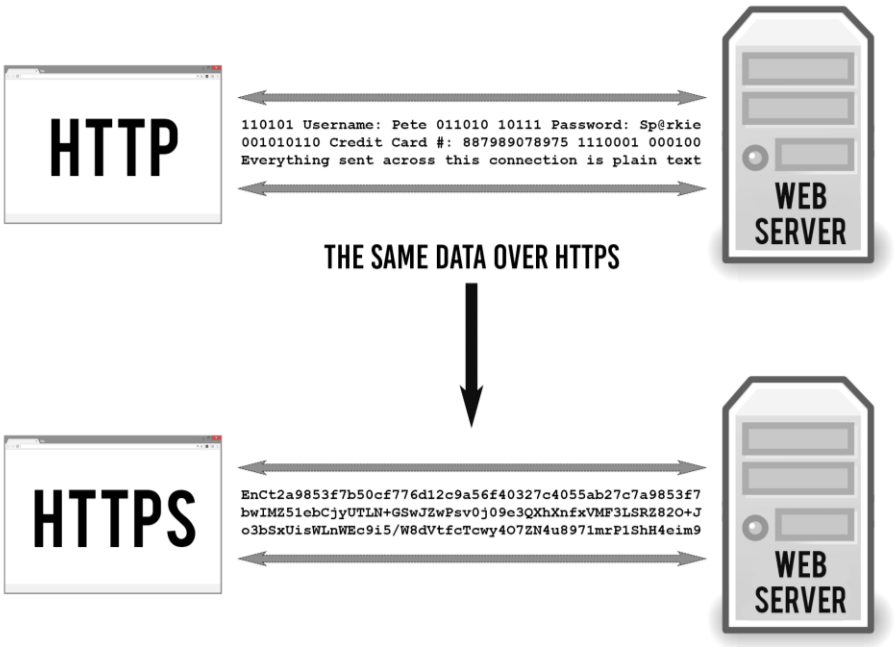
2.4.1. Características mínimas de una PWA

2.4.1.1. Contexto seguro HTTPS

Las aplicaciones web progresivas deben ser servidas a través de una red segura, poner a disposición una aplicación de forma segura, a través de un protocolo como HTTPS no solo es una buena práctica, también garantiza a los usuarios finales que el sitio al que están accesando es confiable y que todas sus transacciones se realizan de una forma segura.

La razón por la que una PWA debe ser publicada en un contexto seguro es que con frecuencia las funciones que proporciona pueden contener datos e información sensible, un ejemplo es la geolocalización. Por otro lado, al hacer uso de service workers es indispensable que la aplicación se haya cargado mediante HTTPS, de otra forma estos no estarán disponibles y el navegador no podrá ejecutarlos.

Figura 3. **Funcionamiento del contexto seguro HTTPS**



Fuente: TipTopSecurity (2017). *How Does HTTPS Work? RSA Encryption Explained*. Consultado el 05 de mayo de 2022. Recuperado de https://tiptopsecurity.com/wp-content/uploads/2017/06/WhatIsHTTPSEncryption.png

2.4.1.2. Service workers

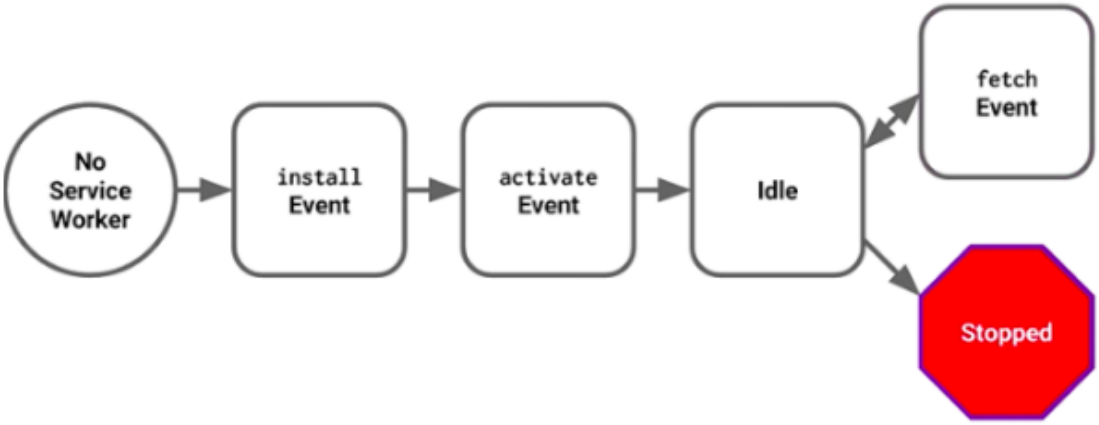
Son scripts con los que se pueden administrar las solicitudes de red, por ejemplo, interceptar o controlar su comportamiento y almacenar respuestas de

contenido en caché. A través del uso de estos, los desarrolladores pueden crear sitios web rápidos, fiables y muy importante, añadir la funcionalidad fuera de línea. Algunas funciones que permite administrar los service workers son:

- Controlar el tráfico y solicitudes de red.
- Almacenar archivos y respuestas en caché.
- Manejar notificaciones push.
- Hacer uso de la aplicación fuera de línea.

2.4.1.3. Ciclo de vida de un Service Worker

Figura 4. **Ciclo de vida de un Service Worker**



Fuente: Ceyhun Keklink, Medium (2017). *What is a Service Worker?*. Consultado el 05 de mayo de 2022. Recuperado de https://medium.com/commencis/what-is-service-worker-4f8dc478f0b9

En el ciclo de vida de un Service Worker podemos distinguir tres fases principales:

- Registro: hace referencia al evento en el cual el servicio es registrado en el dispositivo, este proceso puede iniciarse en todo momento en el que la

pagina es cargada, este proceso no se ejecuta todo el tiempo ni todas las veces que la aplicación es cargada, solamente se iniciara cuando reconozca que existe un nuevo servicio o una actualización de este.

- Instalación: este se ejecuta en dos escenarios, no existe un service worker o existe una actualización, en este evento es donde se especifica y almacena en caché el contenido estático.
- Activación: es iniciado al finalizar el evento de instalación, en el podemos eliminar contenido estático.

2.4.1.4. Archivo Manifest

Un archivo Manifest, es un archivo en formato JSON en el cual se describe como se desea presentar la aplicación al usuario final, la razón de su existencia esta específicamente relacionado con los PWAs, con este archivo garantizamos que la aplicación web progresiva sea detectable. Algunos datos que se describen en su contenido son: nombre de aplicación, URL de inicio, iconos, entre otros detalles, necesarios para llevar la aplicación de un formato web, a uno similar al de una aplicación móvil nativa.

Figura 5. **Estructura básica de un archivo Manifest**

```
{
  "name": "Google I/O 2015",
  "short_name": "I/O 2015",
  "start_url": "./?utm_source=web_app_manifest",
  "display": "standalone",
  "icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
  }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
  }],
  "related_applications": [{
    "platform": "web"
  }, {
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?
id=com.google.samples.apps.iosched"
  }]
}
```

Fuente: Mdn Web Docs (2022). *Web App Manifest*. Consultado el 05 de mayo de 2022. Recuperado de https://developer.mozilla.org/es/docs/Web/Manifest

2.4.2. Ventajas de PWA

2.4.2.1. Reconocible

El objetivo es que las aplicaciones puedan contar con metadatos utilizables por los navegadores web, en un PWA se pueden interpretar como

metadatos, la información contenida en el archivo Manifest, donde se detallan características como nombres de aplicación, iconos y colores de tema.

2.4.2.2. Instalable

Se debe garantizar la experiencia de usuario, dar la sensación de que se está interactuando con una aplicación nativa, parte de ello es poder contar con iconos en la pantalla de inicio, los cuales puedan acceder a la aplicación y lanzarla su propio contenedor nativo.

2.4.2.3. Enlazable

Es importante reconocer que es una de las ventajas más relevantes de las PWAs, ya que se encuentran vinculadas a una sola URL especifica, desde la cual son servidas, dejando de lado los procesos complejos de instalación desde una tienda de aplicaciones como App Store o Google Play.

2.4.2.4. Independiente de la red

La independencia de red hace referencia a la capacidad de una aplicación de poder operar con mala conexión e incluso cuando exista una ausencia de esta. Para poder llevar a cabo estas funciones, las PWAs combinan tecnologías como Service Workers, API caché, almacenamiento Web e Indexed DB.

2.4.2.5. Compatibilidad de mejora progresiva

Cuando se desarrolla una PWA, hay que tener en cuenta que esta aplicación será descargada y accedida desde distintos tipos de dispositivos, con lo cual se deben implementar estrategias capaces de incorporar flujos alternos

en los casos donde las características y tecnologías del PWA no sean compatibles por completo con el navegador.

2.4.2.6. Reconectable

Una de las ventajas de las PWA es la opción con la cual los usuarios pueden interactuar con el contenido nuevo y actualizaciones sin necesidad de estar dentro de la aplicación o utilizando su dispositivo. Al incorporarse Service workers y API Web Push, el desarrollador es capaz de poder enviar actualizaciones a los usuarios sin necesidad que ellos accedan al nuevo contenido.

2.4.2.7. Adaptable

El UI de este tipo de aplicaciones es capaz de ajustarse a cualquier dimensión de pantalla, independientemente del dispositivo, ya sea computador personal, de escritorio, celular o tableta; los componentes visuales cuentan con la suficiente configuración y estilos para poder adaptarse a las necesidades del dispositivo.

2.4.2.8. Segura

El medio en el cual se sirve este tipo de aplicaciones proporciona un mecanismo de entrega seguro, el cual garantiza que evita la mayoría de las veces cualquier intento de espionaje o de manipulación de contenido, para esto implementa el protocolo HTTPS

2.5. WebAssembly

WebAssembly es un tipo de código ejecutable por los navegadores más recientes, su lenguaje trabaja a bajo nivel, de una forma similar al del lenguaje ensamblador. Gracias a un formato compacto y binario, su ejecución proporciona un rendimiento casi nativo, una característica muy importante es que fue diseñado para poder funcionar a la par de JavaScript, permitiendo hacer uso de ambos al mismo tiempo.

La gran ventaja de poder funcionar paralelo a JavaScript es que cualquier desarrollador con conocimientos de JS puede desarrollar en WebAssembly, incluso sin saber escribir código WebAssembly.

2.6. Blazor Framework

Blazor es una framework para el desarrollo de interfaces de usuario web, accesibles desde lado cliente. Con esta plataforma de trabajo se puede sustituir por completo el uso de Javascript por C#, lo cual facilita la interacción con el servidor cuando este se encuentra alojado en un ecosistema de .NET. Al desarrollar una aplicación web con Blazor la lógica de la aplicación se comparte con el cliente y el servidor.

2.6.1. Componentes

Al igual que frameworks que actualmente han tomado auge como Angular, React o VueJS, Blazor funciona con un patrón muy similar, basado en componentes. Un componente no es nada más que cualquier elemento que está presente en la interfaz de usuario, puede ser tan grande como una página completa, o tan pequeño como el cuadro de entrada de texto de un formulario.

Un componente en Blazor es una clase escrita en C# de .NET y cada uno tiene las siguientes propiedades:

- Define su propia lógica de representación de la interfaz de usuario
- Puede controlar las acciones del usuario
- Tiene la capacidad de anidarse y ser reutilizados
- Pueden ser compartidos y distribuidos como una biblioteca de clases o paquete NuGet.

Normalmente un componente de Blazor es escrito en formato Razor, los cuales están contenidos en archivos de extensión Razor (*.razor). Razor es la sintaxis que C# implementa para combinar lenguaje de etiquetas HTML con código C# en un mismo archivo. Los componentes se usan única y específicamente para definir la lógica de la interfaz de usuario en el lado cliente.

La utilización de lenguaje de etiquetas HTML para definir la interfaz de usuario hace que el desarrollador pueda sentirse más cómodo y familiarizado con el flujo y lógica de desarrollo; el uso de código C# está contemplado para interpretar las interacciones que el usuario tiene con la aplicación y el componente directamente, en esta porción de código es donde la manipulación de datos, eventos e información ocurre, pudiendo o no devolver información y desencadenar más eventos.

Figura 6. **Estructura y sintaxis básica de un componente Razor.**

```razor
<div class="card" style="width:22rem">
    <div class="card-body">
        <h3 class="card-title">@Title</h3>
        <p class="card-text">@ChildContent</p>
        <button @onclick="OnYes">Yes!</button>
    </div>
</div>

@code {
    [Parameter]
    public RenderFragment? ChildContent { get; set; }

    [Parameter]
    public string? Title { get; set; }

    private void OnYes()
    {
        Console.WriteLine("Write to the console in C#! 'Yes' button selected.");
    }
}
```

Fuente: Microsoft Docs (2022). *Blazor de ASP.NET Core*. Consultado el 07 de mayo de 2022. Recuperado de https://docs.microsoft.com/es-es/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

En la porción de código anterior se detalla la estructura más común y básica de un componente Razor.

Atributo Title: es un atributo que se puede pasar a través de una etiqueta en el componente, el valor proporcionado para este atributo se asigna a la propiedad Title del componente.

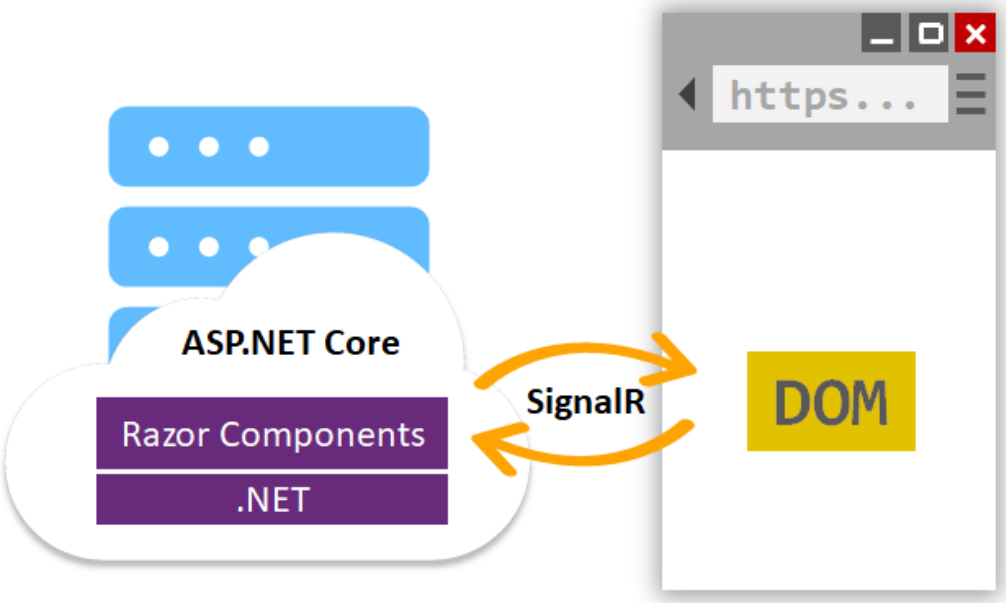
Contenido ChildContent: es una porción de HTML la cual es accedida y asignada al componente en el momento en que se anida más contenido HTML dentro de la etiqueta. De la misma forma en que se pueden anidar más etiquetas también se puede anidar más componentes Razor.

Método OnYes: es un método escrito en lenguaje C#, que es desencadenado por el evento onclick del botón del componente. Es importante resaltar que este código C# es escrito dentro del mismo archivo Razor, y puede

acceder a las propiedades del componente de la misma forma que lo hacen los métodos y funciones en una clase c#. Su llamada se hace desde otros métodos o funciones e incluso a través de los eventos convencionales de elementos HTML.

2.6.2. Blazor Server

Figura 7. **Funcionamiento de Blazor Server y SignalR**



Fuente: Microsoft Docs (2022). *Blazor de ASP.NET Core*. Consultado el 07 de mayo de 2022. Recuperado de https://docs.microsoft.com/es-es/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

Razor con SignalR es una de las formas de publicar y proveer un PWA con .NET, la principal característica es que el grafico de componentes, vistas y paginas residen en un Servidor Blazor, sin excepción alguna.

De forma resumida, cada línea de código de Razor emite HTML en texto. Posterior a la renderización, el servidor se encarga de desechar la instancia de la página o vista en cuestión, junto a cualquier estado que la misma haya desencadenado. Si se produce otra solicitud de esta página, entonces toda la página se vuelve a renderizar en HTML y es enviada al cliente.

La forma en que se actualiza el sitio, componentes y acciones en el cliente es la siguiente: con cada solicitud el servidor Blazor Server, produce un grafo de componentes, similar al DOM que significa modelo de objetos del documento. Este grafo incluye dentro de su estado las propiedades y campos del componente. Blazor evalúa el grafo y produce una representación binaria, la cual envía al cliente para ser renderizada. Posterior a la conexión realizada entre el cliente y el servidor, los elementos estáticos del componente se reemplazan por elementos interactivos.

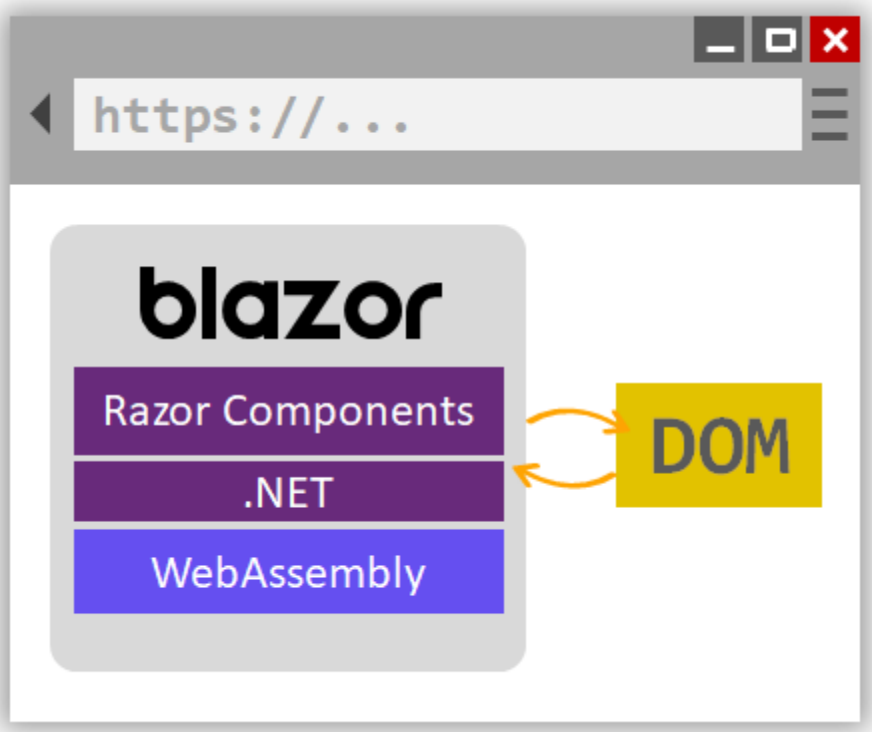
Cuando una interacción del usuario o un evento en la aplicación produce una actualización en la interfaz, el grafo de componentes calcula un valor de diferencia de interfaz de usuario, esta diferencia nos d a como resultado el conjunto más pequeño de ediciones DOM necesarias para producir un cambio en la interfaz del lado cliente. Esta diferencia se recibe como respuesta del servidor y en formato binario, una vez recibida el explorador se encarga de aplicar los cambios.

2.6.3. Blazor WebAssembly

Otra alternativa para publicar nuestra aplicación y salir de los estándares aplicación Cliente-Servidor, es Blazor WebAssembly, este es un framework para SPA o aplicaciones de una sola página. Blazor WebAssembly es capaz de proveer aplicaciones interactivas del lado cliente desarrolladas con .NET, utiliza

estándares web sin la implementación de plugins extra o recompilando código en otro lenguaje. Una de sus ventajas es el soporte, ya que funciona en todos los navegadores web modernos incluyendo los navegadores de dispositivos móviles. WebAssembly nos permite ejecutar código .NET gracias a que utiliza formato de bytecode, el cual es optimizado para su fácil descarga y una ejecución rápida.

Figura 8. **Comportamiento de Blazor y WebAssembly**



Fuente: Microsoft Docs (2022). *Blazor de ASP.NET Core*. Consultado el 07 de mayo de 2022. Recuperado de https://docs.microsoft.com/es-es/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

Cuando una aplicación Blazor es compilada y ejecutada por un navegador ocurre lo siguiente:

- Los archivos Razor y C# se compilan en archivos de ensamblado .NET
- Los archivos de ensamblado .NET son descargados directamente al navegador
- Blazor WebAssembly configura el entorno para cargar los archivos de ensamblado .NET, para poder manipular el DOM y realizar llamadas al API, WebAssembly utiliza JavaScript interop.

2.7. Herramientas seleccionadas

Tomando en cuenta los lenguajes de programación y bases de datos utilizados en la institución se analizaron las distintas opciones que existen para elaborar aplicaciones móviles, decidiendo así, utilizar las siguientes tecnologías.

2.7.1. Blazor WebAssembly

Debido a que dentro de la institución el único lenguaje para desarrollo, soporte y mantenimiento de aplicaciones de software interno es .NET, se evaluaron las distintas alternativas que .NET pone a disposición para desarrollo de aplicaciones móviles, considerando las necesidades y requerimientos de software realizado por la institución se concluyó que una PWA con Blazor WebAssembly era la mejor solución para cumplir con los objetivos en el tiempo establecido.

Las razones por las que un PWA con Blazor WebAssembly es una solución óptima para este proyecto son:

- Punto de vista de la institución:
    - Herramienta con soporte de Microsoft, garantizando soporte y documentación a largo plazo

- C# es el lenguaje utilizado en todos los proyectos de software de la institución.
- La institución requiere que la aplicación pueda instalada en cualquier dispositivo móvil, independiente del sistema operativo, un PWA da solución a este tipo de requerimiento.
- El software no contempla interacciones especificas con hardware, por lo tanto, no es requerido un framework con funciones, módulos o plugins sumamente elaborados y Blazor WebAssembly nos brinda las herramientas necesarias para cumplir los objetivos planteados.

- Punto de vista del desarrollador:
  - Al ser un framework desarrollado por Microsoft, su documentación y soporte es bastante amplio, suficiente para poder desarrollar cualquier proyecto, incluso para cualquier desarrollador con poca experiencia con el lenguaje y framework.
  - El desarrollo y curva de aprendizaje para desarrollar nuevos componentes, personalizados, es relativamente sencillo, ya que su comunidad y documentación es sumamente amplia.
  - Permite desarrollar aplicaciones en corto tiempo, al estar orientada a una programación y diseño por componentes facilita la interpretación del código para cualquier desarrollador.
  - Actualmente es muy común ver que el desarrollo de aplicaciones adopta con frecuencia un diseño por componentes, como lo hace React Native; Blazor no es la excepción, ya que utiliza esa misma lógica y le permite a cualquier desarrollador, con experiencia en proyectos estructurados por componentes, adaptarse de forma sencilla a soluciones de software escritas en Blazor Framework.
  - Debido a que está orientado a un diseño de componentes, permite reutilizar bastante código, no solo el código asociado a la lógica de la aplicación, sino también para la visualización de datos.

- Punto de vista del usuario:
    - Blazor utiliza Bootstrap como hoja de estilos, una de las más utilizadas y con las que el usuario final se encuentra familiarizado. Su adaptabilidad a cualquier dimensión de pantalla es muy buena con lo que permite que cualquier dispositivo pueda hacer uso de ella sin ningún problema.
    - Al proveer el software como un PWA el usuario no debe ingresar a una tienda virtual para descargar la aplicación, simplemente debe acceder desde el navegador y añadirla al inicio de su dispositivo.
    - El PWA puede ser instalado o no, es una función totalmente opcional y el usuario es quien tiene la decisión final.
    - Una aplicación web progresiva brinda una experiencia de usuario casi idéntica a la que una aplicación nativa puede dar.

2.7.2. SQL Server

Al igual que la elección del framework para desarrollar la lógica del software, la decisión de que motor de base de datos utilizar se ve limitada por la suite de herramientas que la institución utiliza; la institución indica que por facilidad, compatibilidad y mantenimiento se utiliza SQL Server en todos sus softwares.

Entre las ventajas que nos proporciona para utilizar con .NET están:

- Compatibilidad al 100%, permitiendo utilizar paquetes, plugins y frameworks Microsoft con total seguridad.
- La conexión a base de datos es sumamente sencilla y con amplia documentación.

- Permite la integración con Entity Framework, lo cual facilita la migración y manipulación de datos en cualquier momento a cualquier otro proveedor del mismo motor de base de datos.
- Creación de vistas y controladores específicos para cada entidad en la base de datos utilizando paquetes especializados en dicha tarea.

2.7.3. C# y .NET core

Para la comunicación entre el cliente y el servidor, se utiliza el lenguaje de programación C#, el cual permite desarrollar la lógica necesaria para manipular o consultar los datos contenidos en la base de datos.

Específicamente, se hará uso de C# para elaborar el API con la cual la aplicación cliente se pueda comunicarse con la base de datos.

2.7.4. Entity Framework

Es una herramienta que permite al desarrollador mapear una base de datos a Objetos, definidos en C# como Clases.

Las relaciones de estos objetos permiten realizar consultas de tipo LINQ, actualizaciones y migraciones de esquemas de datos completos. Este framework permite realizar todas estas operaciones con distintos tipos de base datos, y por supuesto siendo Microsoft el proveedor, permite realizarlo con SQL Server.

2.8. Arquitectura de PWA Dataforg

Definidas las tecnologías y frameworks a utilizar se definen las siguientes arquitecturas, que en conjunto permitirán la consulta y mantenimiento de la información.

2.8.1. Arquitectura PWA Dataforg cliente

El cliente es el encargado de proveer una interfaz entre usuario y servidor, permite la visualización y manipulación de la información a través de filtros especializados, debe ser intuitiva para el usuario ya que la mayoría del grupo objetivo no es especializada en el uso de tecnologías y aplicaciones de software.

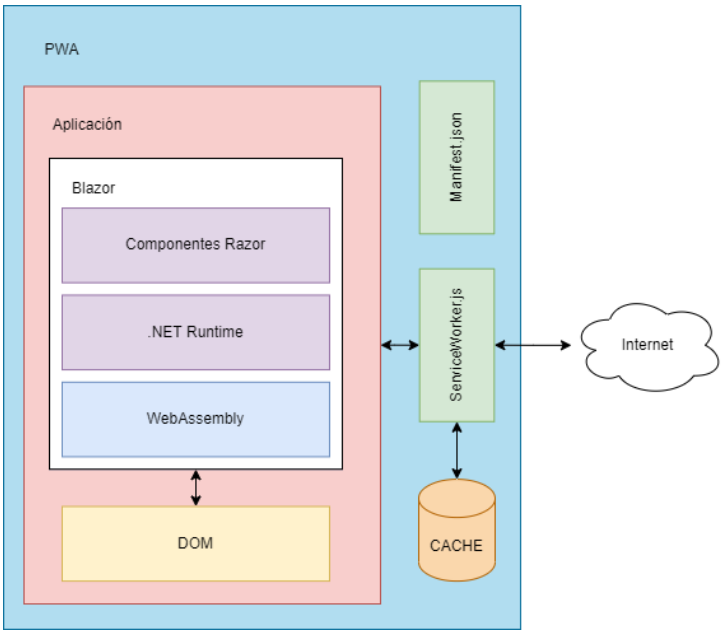
Al ser un PWA, la aplicación cliente no solo cumple las funciones tradicionales de interactuar con el servidor de forma transaccional obteniendo y enviando información a la base de datos; es también la encargada de administrar las actualizaciones del código fuente, almacenamiento en cache y direccionamiento de las peticiones web que los eventos o funciones desencadenen.

Como se puede observar en la Figura 9, el PWA no interactúa directamente con el internet, toda petición es filtrada por el archivo Service Worker, este es un archivo escrito en Java Script encargado de filtrar las peticiones, como desarrolladores podemos administrar estos filtros, por ejemplo, detectar el contenido de la URL a la que se realiza la petición, el tipo de la petición.

Debido a que el cliente es un PWA, también se cuenta con un archivo Manifest, el cual define contenido multimedia extra, los cuales son utilizados al instalar la aplicación en el dispositivo.

Finalmente podemos observar la estructura de Blazor WebAssembly, la cual interactúa con los archivos y componentes adicionales que componen un PWA tradicional.

Figura 9. **Arquitectura PWA Dataforg cliente**



Fuente: elaboración propia, realizado con diagrams.net

2.8.2. Arquitectura Dataforg server

En el servidor web es el encargado de proveer la información al cliente. Dentro de él se encuentran los servicios y lógica del acceso a datos, a través de un API, la aplicación cliente puede interactuar con la base de datos, con cada

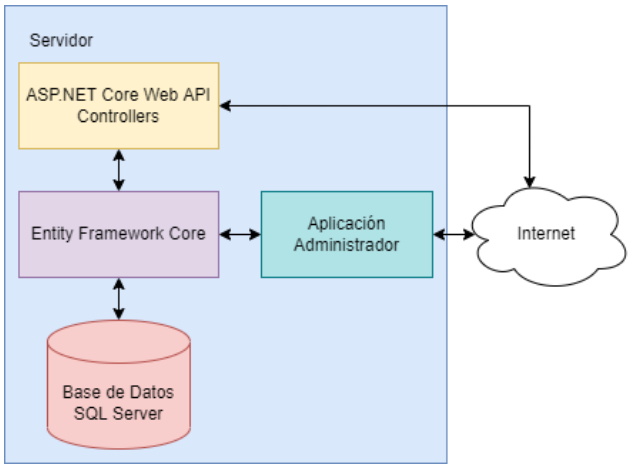
uno de los servicios del API el servidor procesa el contenido de la petición y devuelve una respuesta.

Cada ruta de acceso del API está programada con C#, dependiendo del tipo de petición el servidor sabe que lógica aplicar y a que datos accesar. Los datos son consultados y manipulados a través de Entity Framework core, la razón por la cual se utiliza este framework es la facilidad para acceder a cada entidad, proporciona métodos y funciones específicas para cada una de las acciones que la base de datos puede procesar.

Al utilizar Entity Framework no solo garantizamos la concurrencia y consistencia de la base de datos, sino también una migración sin alteración alguna, esto gracias a que, al utilizar un Contexto de Base de datos, todos los nombres de índices, llaves y triggers son exactamente iguales al migrar la información.

Junto a la lógica de los servicios también se encuentra la aplicación administrativa, o como comúnmente se conoce las vistas CRUD de las entidades relacionadas a cada especie almacenada en la base de datos, esta aplicación se provee como una aplicación web tradicional en .NET, sin hacer uso de WebAssembly o archivos adicionales para funcionar como un PWA.

Figura 10. **Arquitectura Dataforg server**



Fuente: elaboración propia, realizado con diagrams.net

2.9. Cliente del proyecto

2.9.1. Diseño de la aplicación

Para diseñar la aplicación móvil se utilizó el framework de CSS Bootstrap, este es el framework que por defecto propone e integra a sus aplicaciones Blazor. Las hojas de estilos y archivos de JavaScript que Bootstrap proporciona simplifican la manera en que se desarrolla un sitio web responsive, este término hace referencia a que todas las pantallas de una aplicación puedan adaptarse a cualquier dispositivo móvil sin importar las dimensiones de la pantalla. Es importante que un PWA cumpla con esta característica, ya que de lo contrario la accesibilidad al contenido y funciones del software se verían comprometidas, limitando al usuario a utilizar la herramienta únicamente en pantallas grandes, como la de un ordenador.

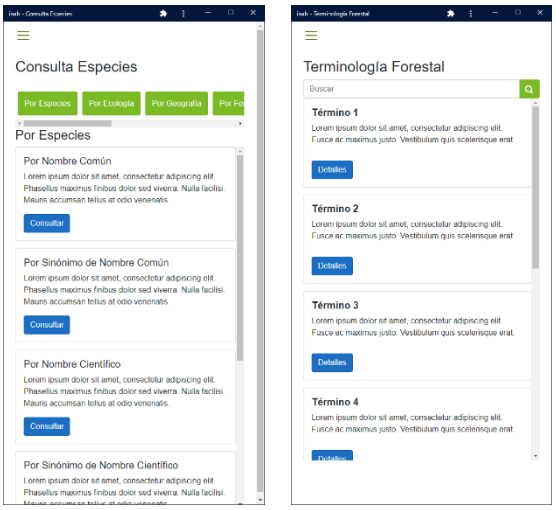
Para el diseño de la nueva aplicación, se utilizó como referencia, las pantallas y filtros existentes en el Dataforg en versión escritorio. A continuación, se muestran las comparaciones de elementos ya existentes en el Dataforg, los prototipos para la nueva aplicación y el diseño final con el que se elaboró la nueva herramienta de software.

2.9.1.1. Filtros

Figura 11. **Diseño de filtros del prototipo**

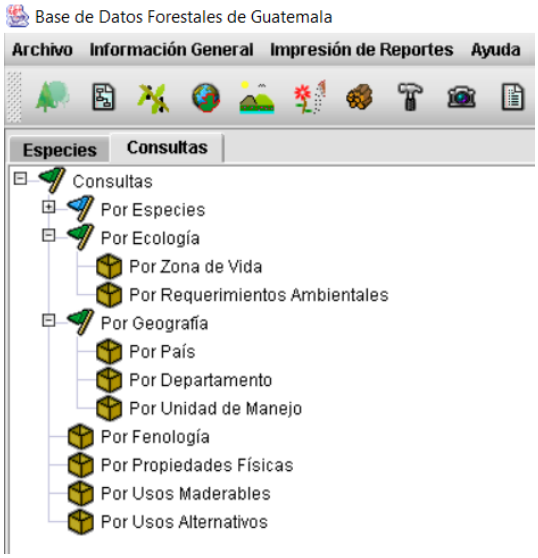
Fuente: prototipo desarrollado para PWA Dataforg

Figura 12. **Diseño de filtros de la aplicación**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Figura 13. **Diseño de filtros de Dataforg existente**



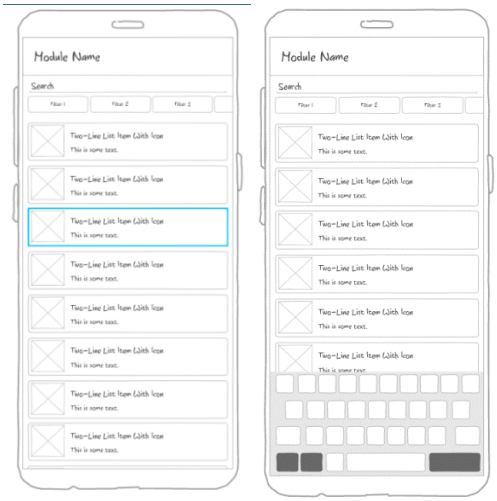
Fuente: aplicación desarrollada para el INAB para la consulta de la base de datos forestales de Guatemala.

Como podemos observar en las Figuras 11 y 12, los filtros del PWA se presentan de una forma más amigable e intuitiva, utilizando como referencia la forma en que aplicaciones como Instagram o Pinterest presenta los filtros disponibles al usuario final.

Los filtros y categorías disponibles en la aplicación de escritorio existente, se trasladan en su totalidad a la nueva aplicación móvil, estos filtros podemos visualizarlos en la Figura 13; como es de esperarse, esta información se presenta de una forma bastante accesible para cualquier usuario que haya tenido contacto con una computadora y aplicaciones de escritorio básicas, sin embargo es esa la mayor problemática de la herramienta actual, esta forma de presentar los filtros es muy antigua y presenta muchas limitantes al momento de intentar adaptarla a dispositivos más pequeños. Para resolver esta problemática se utilizó un menú colapsable para las categorías, y un scroll horizontal infinito para los filtros asociados a cada categoría. Al dar solución de esta forma, permitimos que se puedan agregar más categorías y filtros a la aplicación con mucha facilidad, esto sin comprometer la usabilidad y accesibilidad de las funciones de la herramienta.

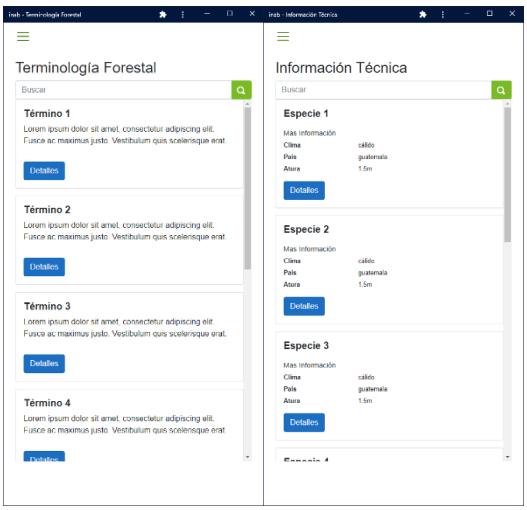
2.9.1.2. Búsquedas

Figura 14. **Diseño de resultados de una búsqueda del prototipo**



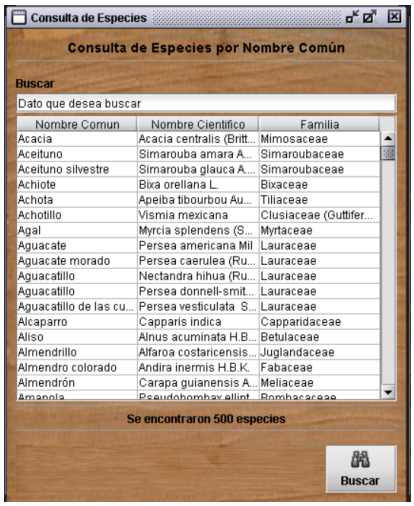
Fuente: prototipo desarrollado para PWA Dataforg

Figura 15. **Diseño de resultados de una búsqueda de la aplicación**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Figura 16. **Diseño de resultados de una búsqueda de Dataforg existente**



Fuente: aplicación desarrollada para el INAB para la consulta de la base de datos forestales de Guatemala.

En las figuras anteriores, Figuras 14, 15 y 16, se observa la comparación de cómo se presentan los datos al usuario, posterior a realizar una búsqueda. En el caso de las búsquedas en la aplicación de Dataforg actual, existen ligeras variaciones en las tablas de resultados, dependiendo del tipo de búsqueda y filtro aplicado, sin embargo, estas columnas no son de mucha utilidad ya que en la mayoría de los casos estos datos que acompañan a cada resultado solamente contaminan la vista, y no son tomados en cuenta al filtrar con búsqueda por texto dentro de los mismos resultados.

Si un filtro por texto no toma en cuenta los atributos que acompañan a los resultados, no deberían ser presentados debido a que esto solo causa confusión y molestias al usuario final. Por esta razón se aplicaron cambios en los datos que acompañan a las búsquedas, siendo el titulo y parte de la descripción general, los únicos datos que se presentan por cada entidad.

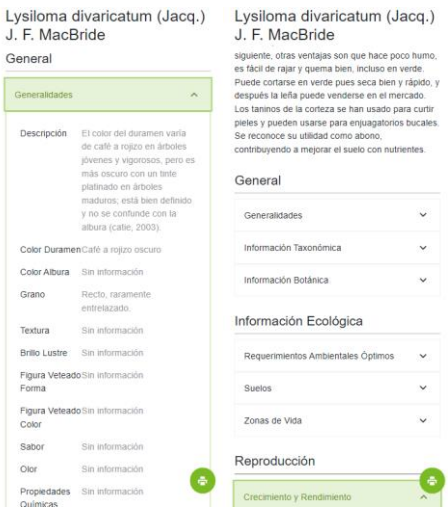
Además de proporcionar una visualización de información más limpia para el usuario, también se agiliza el filtrado por texto, ya que la aplicación no debe realizar comparaciones innecesarias ni procesado de datos adicionales.

2.9.1.3. Vista individual de una especie

Figura 17. **Diseño de la vista de una especie del prototipo**

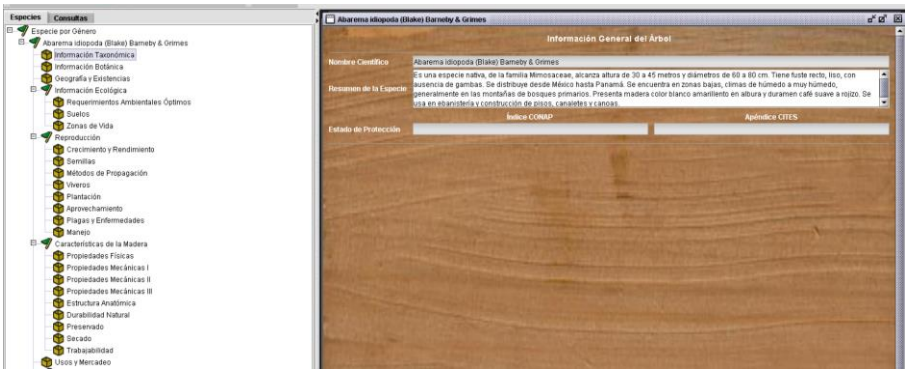
Fuente: prototipo desarrollado para PWA Dataforg

Figura 18. **Diseño de la vista de una especie de la aplicación**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Figura 19. **Diseño de la vista de una especie de Dataforg existente**



Fuente: aplicación desarrollada para el INAB para la consulta de la base de datos forestales de Guatemala.

Al igual que el diseño de las demás vistas, se procuró conservar la forma en la que el usuario interactúa con la información, sin embargo, al ser la vista que

contiene más información en comparación a las demás, requirió un mayor análisis y diseño para poder brindar una buena experiencia de usuario.

Para poder decir que este diseño brinda una buena experiencia de usuario, se acudió a un debate con los encargados del área a la que el proyecto pertenece, cuestionando cuales deberían ser los requisitos ideales con los que la aplicación tendría que contar, para poder ofrecer un servicio que se adaptara a la mayoría de los usuarios del software. Concluyendo así que las características principales de esta vista tenían que ser:

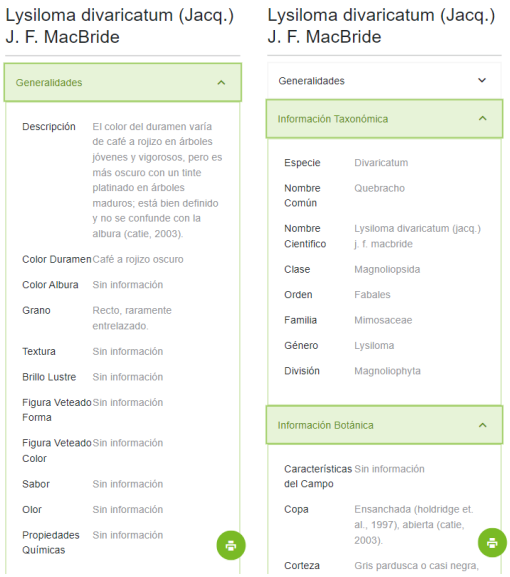
- Permitir el enfoque en un grupo de información especifico por cada especie, con la finalidad de poder dirigir la atención a un segmento de características sin obstruir la vista con distracciones innecesarias.
- Facilitar al usuario ocultar información que no sea de su interés y mostrar solamente las características que le serían útiles en algún momento dado.
- Los elementos con los que el usuario puede interactuar deben ser intuitivos y de fácil acceso.

Tomando en cuenta estos requerimientos se realizó un diseño basado en tarjetas colapsables, las cuales contienen una tabla con la información asociada. Esta agrupación permite que la visualización de los datos sea ordenada, limpia y que exista un estándar para que la consulta de información sea más sencilla una vez comprendido el formato.

Debido a la cantidad de información que se asocia a las características de una especie, se optó por el formato de una tabla vertical, donde las columnas se muestran en forma de filas y la información de una fila en las columnas. Este formato permite que sin importar la cantidad de información que se desee mostrar en pantalla, la interacción y consulta de datos no sea tediosa ni complicada,

especialmente en pantallas de tamaño reducido o en formato vertical, como la de los dispositivos móviles.

Figura 20. **Diseño de tarjeta colapsable para una característica especie**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

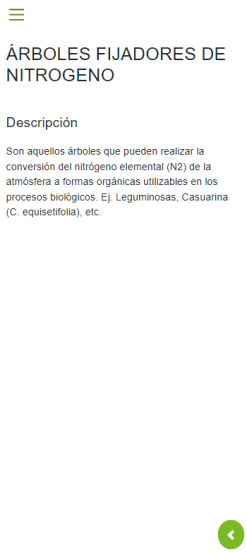
En la Figura 20 se muestra el funcionamiento del componente antes mencionado, cuando se consulta una especie por defecto todas sus categorías se encuentran expandidas, a la espera de la interacción que el usuario desee tener con la información presentada.

2.9.1.4. Terminología forestal

Al acceder a uno de los elementos en el listado de resultados de términos forestales, se muestra el detalle completo de la descripción de dicho termino. Para desplegar la información se optó por una vista sumamente simple, resaltando el título de lo que se muestra en pantalla, seguido por la descripción

completa disponible en la base de datos. En el momento en que la información exceda el número de líneas disponibles en el espacio de la pantalla, el usuario podrá deslizarse hacia abajo para continuar con la lectura.

Figura 21. **Diseño de la vista de un término forestal**

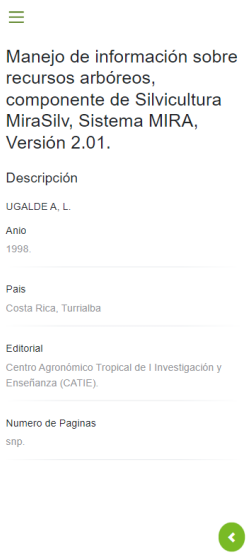


Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

2.9.1.5. Referencias bibliográficas

En el caso de las referencias bibliográficas, se utiliza un diseño muy similar a las tablas verticales que se usan para la mostrar la información de una especie; a pesar de que los datos relacionados a las referencias bibliográficas no son muy extensos, es importante considerar que pueden existir casos puntuales, en los que la información ocupe todo o más del espacio disponible en pantalla del dispositivo, por ello se implementó este tipo de vista, en el cual el usuario puede interactuar de manera fácil e intuitiva con la tabla.

Figura 22. **Diseño de la vista de una referencia bibliográfica**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

2.9.1.6. Directorio de empresas

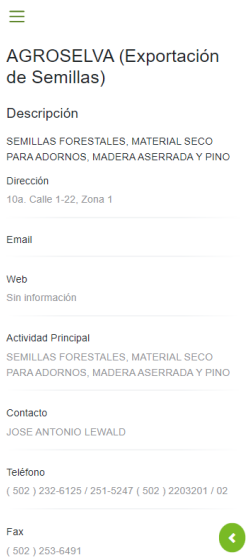
Para el listado de empresas se diseñó una variante del componente que se utiliza para mostrar los elementos disponibles; se consideró de esta manera debido a que por lo regular al buscar un contacto, los usuarios buscan dos datos importantes, dirección de correo electrónico y número de teléfono. Para facilitar las búsquedas se muestran ambos datos en el listado de elementos, así, el usuario no necesariamente debe acceder a toda la información para obtener los datos más importantes de cada empresa.

Figura 23. **Diseño de la vista de todas las empresas**

Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Sin embargo, si el usuario lo desea también puede acceder a la información completa y detallada que se encuentra almacenada en la base de datos, como se muestra a continuación en la Figura 24.

Figura 24. **Diseño de la vista de una empresa**

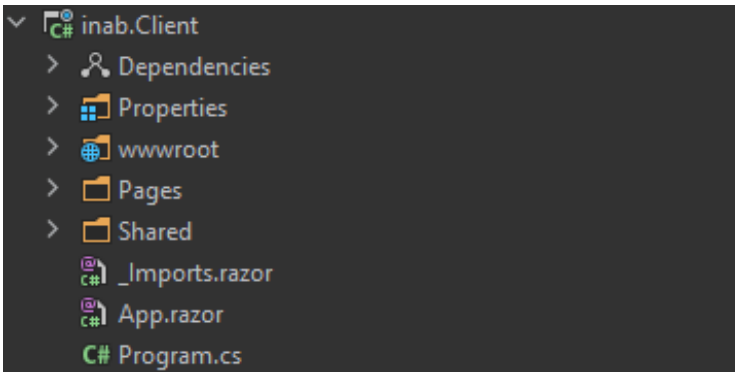


Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

2.9.2. Estructura de archivos aplicación cliente

Al utilizar la plantilla de PWA Blazor WebAssembly se genera automáticamente una estructura de directorios y archivos básicos para comenzar con el desarrollo, dentro de los cuales debemos destacar el que contiene el subdominio Client, este directorio es el que contiene los archivos y subdirectorios donde podemos agregar y editar archivos que están relacionados específicamente con la aplicación lado cliente.

Figura 25. **Estructura de archivos aplicación cliente**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Como se observa en la Figura 10, el directorio inab.Client, contiene varios archivos y directorios importantes, los cuales se explicarán a continuación:

- wwwroot: en esta carpeta se encuentran todos los archivos estáticos de la aplicación web, tales como hojas de estilo, archivos JSON de configuración e iconos de la aplicación. Es también, dentro de este directorio donde se almacenan los archivos necesarios para poder convertir una aplicación web en PWA, estos archivos son: manifest.json y service-worker.js.
- Pages: este directorio es el que contiene los archivos de extensión razor, los cuales son asociados a una página o componente dentro de la aplicación. Dentro de este directorio encontraremos un folder llamado Components, el cual contiene solamente archivos Razor relacionados a componentes.
- Shared: en esta carpeta se encuentran componentes que todas las páginas en nuestra aplicación comparten, NavMenu.razor es el menú de la aplicación y el cual es accesible desde cualquier página, a su vez este

menú está contenido en MainLayout.razor, quien él es el padre de todas las páginas de nuestra aplicación.

- _Imports.razor: en este archivo se describen todas las sentencias de import que comparten todas las páginas en la aplicación.
- App.razor: es el archivo que contiene las etiquetas necesarias para configurar el Router y la lógica para desplegar el Layout indicado, dependiendo del resultado de evaluar la ruta solicitada por el usuario.
- Program.cs: contiene las configuraciones y declaraciones iniciales necesarias para instanciar toda la aplicación y servicios web utilizados por la aplicación.

2.9.3. Distribución de componentes

Debido a que la aplicación cuenta con diversos módulos de información, se realizó un análisis exhaustivo de las funciones de búsquedas y filtrados con los que cada apartado cuenta; una vez realizada esta inspección se concluyó que existían algunos componentes bastante similares entre sí, y que por ende pueden compartir la misma estructura base con ligeros cambios modificables según el contexto en el que se utilicen.

A los componentes que se pueden observar repetidas veces en distintos apartados de la aplicación les llamaremos de uso compartido, ya que sin importar donde los utilicemos la única información que varía es su contenido, mas no su estructura interna. Por otro lado, encontraremos a los componentes que solo existen y se utilizan en una situación específica, no pueden ser llamados en otros módulos o apartados de la aplicación, a estos les llamaremos Componentes específicos.

2.9.3.1. Componentes de uso compartido

Estos componentes los encontraremos en el directorio Components, dentro de la carpeta padre Pages, que pertenecen al proyecto inab.Client. En el folder Components encontraremos únicamente componentes que pueden ser compartidos por todos los demás componentes y páginas de la aplicación cliente, en caso contrario si la porción de código no puede ser utilizada por todos los elementos de la aplicación, deberá colocada en otro directorio.

Los componentes que podemos encontrar en este directorio son:

- Collapsable, este componente permite alojar información agrupada a través de un título el cual permite al usuario mostrar u ocultar su contenido al interactuar con el encabezado.
- FAB: es un botón flotante que se encuentra en la esquina inferior derecha al cual se le puede asociar diferentes funciones y asignar un icono especifico.
- ListViewSearchBar: junto a ListViewSearchBarItem, es uno de los componentes más relevantes y vitales para el funcionamiento de la aplicación, en él se listan todos los elementos resultantes de una búsqueda, contiene una barra de búsqueda por texto el cual brinda al usuario limitar aún más los resultados de su búsqueda.
- ListViewSearchBarItem: se encarga de mostrar la información de un elemento asociado al resultado de una consulta, los datos que se pueden visualizar en este componente es un resumen por lo que es información limitada.
- Loader: esta es una vista que cubre al componente principal actual mientras existe una llamada a un servicio, mientras el servicio está procesando la solicitud esta capa le da retroalimentación al usuario de lo que está sucediendo; esto es bastante útil en situaciones donde la

conexión a internet es bastante débil o si la solicitud está tomando más tiempo de lo habitual.

2.9.3.2. Componentes específicos

Son componentes asociados a tareas más delimitadas, se encuentran apartados y agrupados por una carpeta con el mismo nombre del componente. Por ejemplo, en el directorio Consultas, podemos encontrar más directorios que a su vez contienen a los componentes relacionados a esa consulta.

Agrupar a los componentes en directorios facilita el mantenimiento del software, una mejor comprensión de lo que cada pieza de código realiza y una vista más simple de la estructura y elementos que lo componen.

2.9.4. Información en cache

Para que una PWA pueda funcionar sin conexión a internet es necesario elegir una estrategia para almacenar la información en memoria, esta información no puede ser almacenada en memoria dinámica por lo que el uso de RAM no es una opción. Los estándares para el desarrollo de un PWA sugieren varias opciones para realizar dicha acción, en el caso de esta aplicación se eligió administrar la información en cache, esta opción brinda ventajas a largo plazo, ya que no se requieren configuraciones complejas y la curva de aprendizaje para administrar la cache es relativamente corta.

En un PWA es necesario configurar la cache a través de un archivo service worker, este archivo es el único que permite realizar este tipo de acciones en segundo plano y mantenerse escuchando por los cambios y solicitudes que el usuario realiza.

2.9.4.1. Administración de cache en service worker

Configurar la caché en un service worker es muy sencillo, basta con brindar un nombre y realizar la apertura de dicho espacio en la caché, para ello se utiliza el objeto caché, dentro del cual se encuentra la función open, esta función recibe como argumento el nombre del espacio que deseamos reservar en la caché.

Para guardar o sobre escribir un elemento en la cache se utiliza la respuesta de la función open, el objeto que da como respuesta permite acceder a la función put, a través de ella se puede almacenar las respuestas de los servicios para poder dar prioridad a lo que se encuentra en memoria antes de salir a internet.

En este proyecto se utilizó una estrategia en la que la respuesta inmediata siempre es la que se encuentra en memoria, y si existe conexión a internet se procede a consultar en segundo plano la información para mantener siempre actualizada la caché.

2.9.5. Service Worker

2.9.5.1. Función onFetch

En esta parte del ciclo es donde ocurre toda la lógica para administrar los datos en memoria cache, en ella se verifica el contenido de la URL consultada, dependiendo de las características con las que cumpla es como se administrará la respuesta del servidor. Dentro de las características que se analizan es el tipo de petición y su origen, así también si contiene palabras clave que hagan referencia a un grupo de servicios específicos.

2.9.5.2. Separación de caches

Dentro del proyecto se configuraron dos caches, una de ellas es donde se almacenan todos los archivos estáticos que componen la interfaz de usuario de la aplicación y la otra es donde se guardan las respuestas de los distintos endpoints a los que la herramienta hace solicitudes cada vez que consulta información.

Esta separación es útil ya que en algunos escenarios es importante mantener actualizada la información de una parte del software, mientras que en otros se espera hasta tener un cambio de versión; por ejemplo, los archivos estáticos de estilos e iconos que se actualizan cada vez que el número de versión de código cambia, y por otro lado, las respuestas de los servicios que es conveniente mantenerlas al día, ya que si existiera información nueva y no se ha actualizado al versión de la aplicación, estos datos serian inaccesibles al usuario hasta que ocurra lo contrario.

2.9.6. Impresión de reportes

Para extender reportes e imprimirlos o guardarlos, se utilizó el mismo administrador de documentos que los navegadores en su actualidad implementan. Gracias a la versatilidad y continuo mejoramiento de los buscadores, es muy sencillo configurar este tipo de capturas de información; con una simple llamada de JavaScript y funciones asociadas, el propio navegador genera un documento en formato PDF, con base en lo que el usuario tiene en pantalla al momento de hacer la llamada a esta función.

Aprovechar esta ventaja no solo hace más rápido y liviano el tamaño de la aplicación, sino también que el mantenimiento y soporte de compatibilidad de

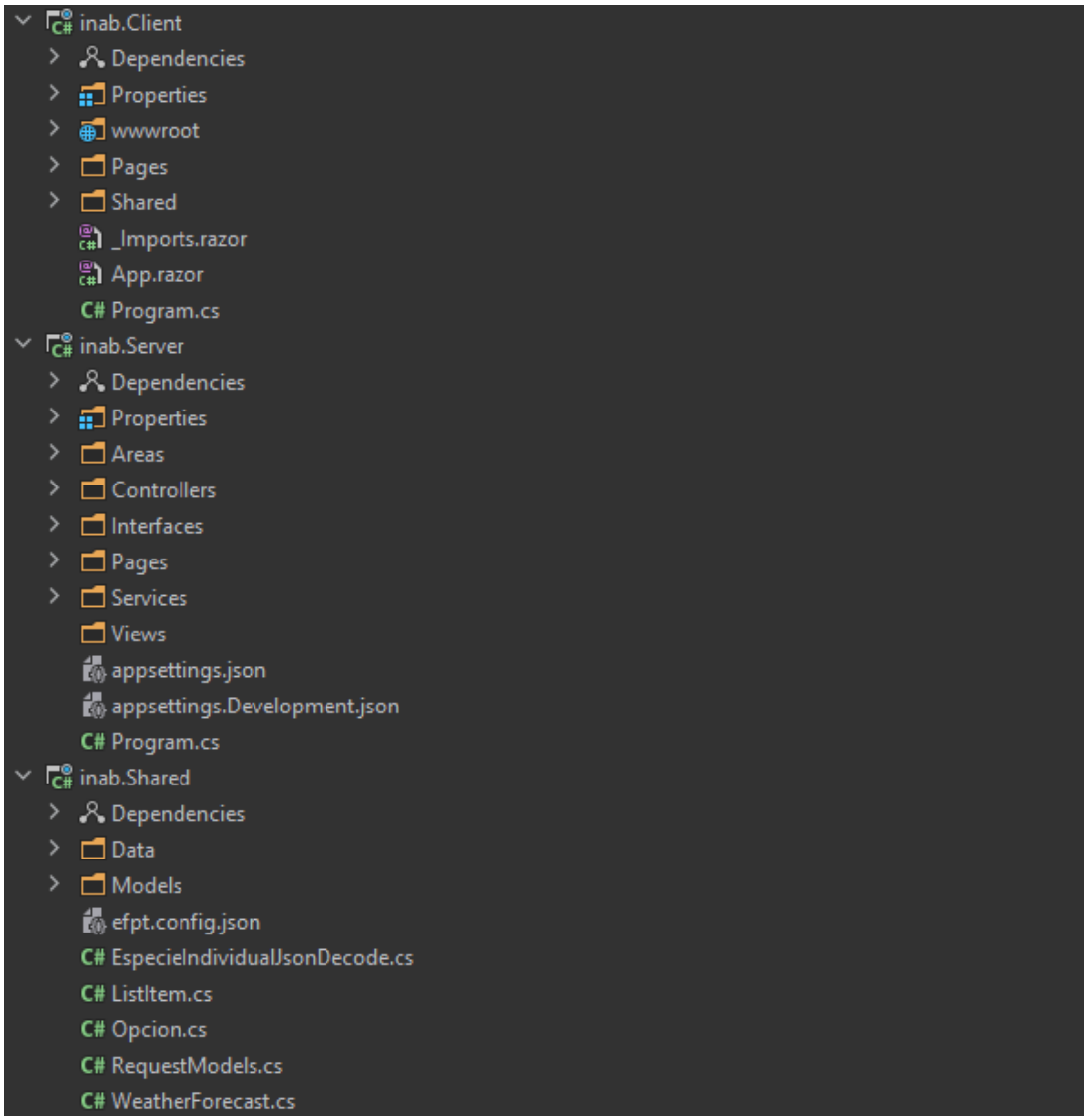
esta característica sea más accesible a cualquier desarrollador. Una de las ventajas más notorias, es la facilidad con la que el usuario puede elegir que páginas desea en el reporte y cuáles no. También hay que mencionar la sensación nativa que transmite, ya que se interactúa directamente con el sistema operativo del dispositivo

2.10. Servidor del proyecto

2.10.1. Estructura de archivos servidor

Como se menciona anteriormente en la estructura de archivos de la aplicación cliente, al utilizar una plantilla de Blazor WebAssembly, automáticamente se genera una estructura mínima de directorios, los cuales se pueden observar en la siguiente figura.

Figura 26. **Estructura de archivos en el servidor**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Como se observa en la Figura 12, un proyecto de Blazor WebAssembly para PWA se compone de 3 directorios principales: Client, Server y Shared.

- Client: contiene todo los archivos, configuraciones y directorios necesarios para proveer al usuario final de una aplicación PWA instalable en cualquier dispositivo.
- Server: en el encontramos todas las clases y directorios con los que el API de la aplicación funciona; de ser necesario es aquí donde se configuran las aplicaciones web tradicionales, como en este caso la aplicación web administrativa.
- Shared: dentro de esta carpeta se almacena cualquier tipo de archivo que comparta el cliente y el servidor, como lo son modelos, interfaces, etc. Todos los archivos y directorios que se crean en esta carpeta son accesibles por cualquiera de los dos folders principales listados anteriormente.

2.10.2. Estructura del directorio Server

Áreas: en este directorio se definen los subdirectorios relacionados a aplicaciones individuales en el servidor, en este caso se puede localizar el subdirectorio Amin, el cual contiene las páginas, componentes, configuraciones y lógica necesaria para administrar el CRUD de las entidades en la base de datos.

Controllers: dentro de este directorio se encuentran las distintas clases asociadas a los servicios del API a los que la aplicación cliente hace referencia en sus funciones y eventos. Cada controlador hereda de BaseController, esta es una clase que contiene una propiedad de tipo DBContext y hereda de Controller, de esta forma podemos acceder a la base de datos en cada controlador sin generar duplicidad de código.

Interfaces: este folder contiene las interfaces asociadas a los servicios internos de la aplicación administrativa; debido a que esta aplicación se

encuentra contenida en el mismo servidor, no es necesario crear servicios aislados en un API, en este caso el servicio es interno y permite asociarse a las acciones de los formularios y tablas que los CRUD utilizan.

Services: en este folder se almacenan las clases que implementan las interfaces contenidas en el directorio Interfaces, en cada implementación se describe la lógica para cada ruta en base al tipo de petición realizada. Estos son los servicios asociados a los CRUD.

Appsettings.json: archivo en formato JSON utilizado para declarar y asignar valores a variables que pueden utilizarse al compilarse la aplicación, en este caso se utiliza para definir la cadena de conexión a la base de datos, de esta forma se descentraliza el alojamiento de la base de datos; al estar contenido en este archivo facilitamos cambiar el host y credenciales de la base de datos sin necesidad de recompilar el proyecto completo y configurarlo en IIS.

Program.cs: es el archivo que utiliza el servidor para levantar todas las configuraciones, páginas y servicios de la aplicación, a continuación, se presentan y detallan fragmentos de las configuraciones utilizadas:

Figura 27. **Configuración de API para aplicación cliente**

```
builder.Services.AddHttpClient( name: "crud", configureClient: c :HttpClient  =>
{
    IConfigurationRoot configuration = new ConfigurationBuilder()
        .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)
        .AddJsonFile("appsettings.json") // IConfigurationBuilder
        .Build();

    var baseurl :string?  = configuration["URLS:baseCRUDURL"];
    c.BaseAddress = new Uri(baseurl);
});
builder.Services.AddDbContext<Dataforgv2Context>();
```

Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

En la Figura 13 se puede observar cómo se asocia el archivo JSON de configuraciones appsettings.json el cual contiene la cadena de conexión a la base de datos, seguido de ello en la última línea de esta figura se observa la forma en la que se relaciona el contexto de base de datos a la aplicación.

Figura 28. **Configuración de Blazor**

```
builder.Services.AddControllersWithViews();
builder.Services.AddServerSideBlazor();
builder.Services.AddRazorPages();
builder.Services.AddControllers().AddJsonOptions(x :JsonOptions =>
    x.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles);

var app :WebApplication = builder.Build();
```

Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

La Figura 14 detalla la manera en que se describe la forma en que habilitamos el uso de Blazor y paginas Razor dentro del servidor, es necesario realizar esta modificación ya que inicialmente la aplicación está configurada para que Blazor sea compilado e interpretado por WebAssembly dentro del navegador del usuario final, sin embargo, la aplicación administrativa también se encuentra escrita desarrollada con páginas y componentes Razor.

Figura 29. **Configuración de servicios internos CRUD**

```
builder.Services.AddTransient<IEspecies, EspeciesManager>();
builder.Services.AddTransient<IZonasDeVida, ZonasDeVidaManager>();
builder.Services.AddTransient<IUsosAlternativos, UsosAlternativosManager>();
builder.Services.AddTransient<IUsosMaderables, UsosMaderablesManager>();
builder.Services.AddTransient<IMigrate, MigrateManager>();
```

Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Para poder utilizar los servicios del CRUD en la aplicación administrativa, se necesita añadir los servicios dentro de las configuraciones iniciales, como podemos verlo en la Figura 15, esta acción se realiza indicando la interfaz y la clase que define la lógica para una de las rutas de los servicios.

Figura 30. **Configuración de aplicación administrativa**

```
app.UseEndpoints(endpoints :IEndpointRouteBuilder =>
{
    endpoints.MapBlazorHub( path: "~/admin/_blazor");
    endpoints.MapFallbackToAreaPage( pattern: "~/admin/{*clientroutes:nonfile}", page: "/_AdminHost", area: "Admin");

    app.MapFallbackToFile("index.html");
});
```

Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Por último, se debe definir el subdirectorio en el cual el usuario va a acceder a las aplicaciones internas del lado servidor; para esto se debe mapear: el archivo de configuración Blazor de la aplicación, la ruta relativa que el usuario debe visitar y en qué área del servidor se encuentra la aplicación; esto se puede observar en la Figura 16.

2.10.3. Estructura del directorio Shared

Data: folder que contiene archivos de clases dedicadas al acceso y manipulación de datos.

Models: en este directorio se almacena todos los archivos generados por Entity Framework, entre ellos las clases que definen las relaciones y atributos de los objetos en los que se mapean las entidades del esquema de datos.

efpt.config.json: archivo en formato JSON que contiene las definiciones para generar el contexto de base de datos y modelos relacionados a una base de datos ya existente.

EspecieIndividualJsonDecode.cs: clase encargada de mapear los atributos devueltos por la consulta de una especie individual, al definir esta clase se facilita el proceso para poder leer una respuesta de tipo especie. La razón por la cual se define una clase para este tipo de respuesta es que la consulta es personalizada por lo que no existe un modelo en el contexto de base de datos que pueda mapear la respuesta de este tipo.

ListItem.cs: es una clase encargada de construir un objeto del tipo esperado por el componente ListViewSearchBarItem en la aplicación cliente, el objetivo es reutilizar el código y componentes al máximo, de esta forma cualquier tipo de resultado de una consulta puede ser traducido e interpretado por el componente universal ListViewSearchBar.

Opción.cs: esta clase es la que define el cuerpo de una opción para poder ser utilizada en el menú de opciones disponibles para filtrar información en la aplicación cliente.

RequestModel.cs: dentro de esta clase se crea una instancia de tipo HttpClient, en ella centralizamos el uso de una única URL base para realizar peticiones al API.

2.10.4. Controladores

En Razor un controlador o ApiController es una clase que extiende de la clase padre Controller la cual provee de los métodos, funciones y propiedades

necesarias para poder interactuar con las rutas y métodos HTTP para servicios RESTful.

Para definir un control es necesario realizar algunas anotaciones, en este caso ApiController, EnableCors y principalmente Route. Cada una de estas anotaciones es necesaria ya que sin ella no podríamos indicar que la clase declarada será utilizada como un punto de acceso del API del sistema.

A continuación, se describen las anotaciones utilizadas en los controladores del API:

ApiController: indica que la clase será utilizada como un controlador o punto de acceso al API del sistema.

EnableCors: permite que el punto de acceso definido sea accesible desde cualquier punto, incluso desde rutas y sitios ajenos al directorio en el que la aplicación se encuentra desplegada, de lo contrario cualquier servicio definido será única y exclusivamente accesible desde el servidor y directorio de la aplicación.

Route: en ella se define con una cadena de texto, cual es la ruta a la que el controlador se encuentra asociado.

Dentro de la clase es necesario agregar una anotación por cada función declarada, esta indica el método HTTP que la ruta espera. Es en esta misma anotación en la cual se incluyen los argumentos que una ruta GET toma, por ejemplo, si se desea pasar un id en la ruta, es necesario definir el nombre de la variable y su tipo de dato; esta información se define con una sintaxis especifica y dentro de una cadena de texto.

2.10.5. Servicios

Similar a los controladores, los servicios son clases que definen métodos y funciones para interactuar con la base de datos, la diferencia es que estos servicios están diseñados para ser utilizados por clases y vistas del sitio administrativo.

Al ser pocas las acciones que se realizan desde el sitio administrativo, el número de servicios es más reducido. Separar la lógica y diseño de la forma en que se interactúa con la base de datos permite un mejor control sobre el código y la base de datos, facilitando a largo plazo el mantenimiento y desarrollo de nuevas funciones.

2.10.6. Sitio Administrativo CRUD

Debido a la gran cantidad de información que el sistema administra y a la continua variación a la que algunos datos se ven expuestos, se solicitó desarrollar una aplicación web administrativa capaz de visualizar, agregar, editar y eliminar datos asociados a las especies que se encuentran almacenadas en la base de datos.

Con base a los requerimientos se decidió que una solución cliente servidor era suficiente para poder satisfacer las necesidades planteadas por la institución, de esta forma se podría mantener la simplicidad de un sistema de administración de datos capaz de soportar y permitir el desarrollo de futuras funcionalidades relacionadas a dicho sistema.

Para poder servir esta aplicación se realizó una configuración especial en el servidor, esta configuración se llevó a cabo con el objetivo de agrupar todos

los subsistemas que conforman el nuevo DATAFORG en un solo lugar; para ello se utilizó un middleware en el archivo principal de la capa servidor del sistema, a través de esto se le brinda las instrucciones necesarias al sistema para que dependiendo de la ruta consultada, la aplicación pueda decidir a qué subsistema debe de redirigir al usuario.

2.11. Base de datos

2.11.1. Modelo entidad relación de la base de datos

El modelo entidad relación describe cada tabla junto a sus atributos, índices y llaves existentes en la base de datos. Permite documentar la base de datos de una forma gráfica, en él se listan los atributos junto a su tipo de dato, valores máximos, mínimos, por defecto, si puede ser nulo, entre otros. También se puede especificar si un atributo es índice o llave de la tabla.

En el diagrama también se describen las relaciones en forma de conexiones entre tablas, cada tipo de relación contiene su propia notación; al representarse de esta forma, la interpretación y consulta de la base de datos es más sencilla para cualquier otro desarrollador, especialmente en bases de datos donde el número de entidades es bastante grande.

2.12. Ambientes de la aplicación

2.12.1. Desarrollo

Durante el desarrollo de la aplicación, se hizo uso de equipo personal, con los siguientes recursos:

- Asus Vivobook 8GB de RAM, Intel core i5 7ma generación, con Linux Mint 20.
- Instancia de base de datos SQL Server en GCP
- Blazor Framework
- .NET Core versión 6

2.12.2. Pruebas y producción

Para las pruebas de despliegue y publicación de la aplicación se utilizó una instancia de servidor virtualizada, la cual fue proveída por la institución, al ser un espacio virtual los recursos son variables y pueden ser modificados en cualquier momento, pudiendo ser adaptados a cualquier otra circunstancia.

Debido a la facilidad con la que la institución puede acceder a estos recursos, las pruebas fueron realizadas en la instancia que a futuro será la que alojará a la aplicación de producción dirigida a usuarios finales, eso facilito la configuración y despliegue ya que no se requerirá de posteriores ajustes o configuraciones por parte del personal de desarrollo y mantenimiento de la institución.

2.13. Atención a los requerimientos

Para iniciar con el desarrollo de este proyecto, el cual tenía como objetivo principal rediseñar las herramientas de consulta actual, se realizó una toma de requerimientos inicial; a partir de esta toma de requerimientos se comenzó con las bases de la aplicación, sin embargo, como en cualquier otro proyecto de software complejo o sencillo, es común que existan cambios, solicitudes de modificaciones e incluso añadir o sustituir algunas partes del software inicialmente planteado y acordado. Tomando en cuenta esta situación se optó

por adoptar una metodología de desarrollo y toma de requerimientos donde el cambio constante no fuera una limitante para continuar con la construcción del proyecto, pero tampoco para la institución quien era la interesada en tener un producto de calidad capaz de cubrir todas las necesidades planteadas.

En el momento en que los requerimientos sufrieran cambios, se realizaba un debate acerca de las nuevas prioridades y funciones pendientes de desarrollo, con el objetivo de satisfacer las necesidades en orden de prioridad y sin dejar de lado los requerimientos inicialmente contemplados. Para que cada cambio o requerimiento adicional pudiera ser aceptado, se debía llegar a acuerdo por ambas partes teniendo en cuenta la forma en que dichas solicitudes pudieran repercutir en tiempos de entrega proyectados en la planificación; esto no significaba no entregar el producto final sino una alteración en las proyecciones que anteriormente se tenían, por ejemplo algunos requerimientos aumentaron en carga y esfuerzo, mientras que otros los redujeron, por lo que la entrega del producto final no se vería retrasada.

2.13.1. Evaluación de objetivos

Durante la evaluación de objetivos, se buscó analizar uno por uno comprobando con evidencia, la correcta implementación de la solución planteada para cada uno de ellos.

Para la base de datos se optó por continuar utilizando el modelo de datos que previamente existía, esto permite una mejor adaptación por parte del equipo de soporte técnico de la institución, sin la necesidad de tener que tomar un proyecto completamente nuevo. Posterior a un análisis extenso de los requerimientos, se concluye que los paquetes tecnológicos están estrechamente relacionados con el modelo de datos asociado a las especies forestales;

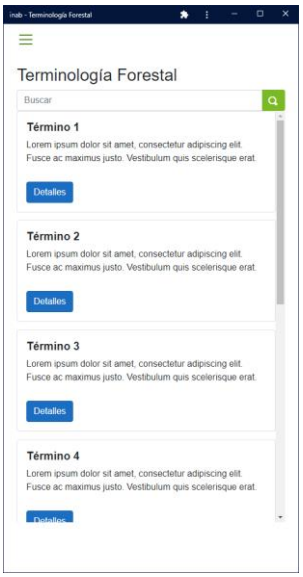
contando únicamente con leves diferencias, por ejemplo, en algunos casos la única diferencia son datos inexistentes en la base de datos, con lo que la solución a ello es habilitar la edición de atributos en las tablas correspondientes.

La integración de algunos nuevos campos y actualización de información relacionada a especies forestales da solución por completo al requerimiento de añadir paquetes tecnológicos a la aplicación DATAFORG.

El modelo entidad relación conserva la definición de datos ya existente, añadiendo al nuevo modelo relaciones entre las entidades correspondientes, un modelo de datos SQL debe contar con relaciones de llaves primarias y foráneas para brindar el máximo rendimiento al momento de consultar información. Debido a la estructura del modelo de datos en este proyecto, es indispensable la configuración de las relaciones, ya que son muchas las entidades existentes y los tiempos de respuesta se verían comprometidos al no indexar las tablas.

Sobre el desarrollo e implementación de las búsquedas y filtrado de la información contenida en el DATAFORG se muestra el componente principal implementado por todos los módulos del software que cuentan con este tipo de interacción.

Figura 31. **Visualización de filtrado y búsqueda de información**



Fuente: PWA Dataforg desarrollada para sustituir la aplicación Dataforg actual.

Para buscar dentro de un módulo de información, se utiliza la barra de búsqueda superior, la cual se encuentra fijada en el mismo lugar todo el tiempo para facilitar su acceso desde cualquier punto del listado de resultados. Sin necesidad de volver a la parte superior de la pantalla, el usuario puede filtrar los resultados, esto es sumamente útil en aplicaciones donde la cantidad de información manipulada es muy grande, con ello se facilita el uso y accesibilidad a las herramientas sin comprometer la experiencia del usuario.

Los filtros específicos por módulo varían, en este caso cada módulo y sus categorías cuentan con pantallas especificas las cuales están especializadas en un tipo de búsqueda en concreto. Cada pantalla adicional para filtrar esta desarrollada en base al software existente por lo que únicamente se adaptó visualmente al nuevo tipo de aplicación.

Sobre la implementación y administración de memoria caché, se realiza a través del service worker del PWA, es la forma más eficiente de integrar el uso de cache en una aplicación de este tipo, además de facilitar su configuración e interpretación por cualquier desarrollador con conocimiento intermedio a cerca de esta herramienta. Para contar con un mejor manejo de los datos en memoria se utilizan espacios independientes, dependiendo de los datos que se desean almacenar, los dos espacios principales son los que se relacionan al contenido estático e indispensable para que la parte visual de la aplicación pueda mostrarse sin conexión a internet, y por otro lado el espacio donde se guarda los datos que se asocian a los servicios del API.

Al contar con espacios en memoria separados y con identificadores únicos podemos realizar modificaciones en cualquier momento sin necesidad de ajustar todo el sistema; dependiendo de donde es que se desea realizar el cambio o actualización sabemos a qué apartado dirigirnos y sin afectar las configuraciones realizadas para las demás funciones existentes.

A cerca de los informes PDF que el usuario puede extender con base a sus búsquedas, se integró la captura de información del documento HTML en el navegador y se trasladó a un documento en formato PDF, esta integración facilito el desarrollo y garantiza su continuo funcionamiento sin la necesidad de añadir módulos o plugin adicionales al proyecto, además es la única forma en la que el usuario puede extender reportes aun sin contar con una conexión a internet.

La renderización de la información en el documento PDF se llevó a cabo a través de la toma de datos en base a contenedores y etiquetas div dentro del documento HTML servido por la aplicación, utilizando esta técnica podemos dividir la información agrupada por subcategorías y mostrarla en páginas separadas respectivamente, lo cual además de presentar la información de forma

ordenada en el documento, también permite y facilita al usuario seleccionar que páginas y subcategorías de su búsqueda desea conservar en el documento y cuales no, todo ello de una forma nativa utilizando el administrador de archivos PDF del navegador en el cual se accede a la aplicación.

2.13.2. Ventajas del proyecto

- Descentralización de la consulta y modificación de información de la base de datos actual de especies forestales de la institución.
- Acceso a la información desde cualquier lugar y desde cualquier dispositivo con conexión a internet.
- Consulta de información previamente visualizada en la aplicación, gracias a la administración de información en caché.
- Visualización de datos más intuitiva.
- Actualización de datos para todos los usuarios sin necesidad de proporcionar un instalable de la herramienta por cada nueva función o característica.
- Extensión de reportes PDF de forma dinámica, habilitando al usuario la opción de seleccionar que características desea añadir o quitar del reporte.
- Control sobre la información que los usuarios pueden visualizar, anteriormente los datos podrían ser de versiones pasadas y no estar al día.
- Fácil control sobre la publicación de la herramienta, al no ser una aplicación nativa no es necesario configurar la distribución a través de las tiendas virtuales de los sistemas operativos de los dispositivos.
- Control sobre los filtros y características a los que el usuario puede acceder.
- Consulta de datos filtrados y actuales desde cualquier dispositivo con un navegador web.

- Permite al usuario decidir si desea utilizar una aplicación instalada en su dispositivo o si solamente desea consultarla desde su navegador.

3. FASE DE ENSEÑANZA – APRENDIZAJE

3.1. Estrategia de enseñanza

Para trasladar el conocimiento necesario a los usuarios se utilizó una estrategia en la cual, se capacitaría en cuanto al funcionamiento y flujos del sistema a los usuarios a cargo del área y el proyecto en el INAB, para que posterior a ello, fueran ellos mismos los encargados de producir el material adecuado para transmitir la información necesaria y adecuada a sus usuarios finales. Esta estrategia permite asegurarnos que el conocimiento a cerca de la nueva herramienta será trasladado con claridad al usuario final, esto debido a que nuestro contacto directo es únicamente con quienes administraran el proyecto, y no con los usuarios finales. Además, es el personal del INAB quien ha tratado con sus usuarios por años, por lo que conocen la mejor forma para poder comunicar y trasladar el conocimiento a los interesados dentro de y fuera de su institución.

3.2. Fase de pruebas

Al terminar el proyecto, se realizaron pruebas por parte de los usuarios de la institución, esto para garantizar que el funcionamiento de la herramienta era correcto y cumplía por completo con las expectativas del proyecto, dando solución a los requerimientos planteados inicialmente. Aproximadamente se utilizó un periodo de quince días, durante los cuales se atendió cada una de las observaciones de los interesados, en su mayoría relacionadas con modificaciones visuales, las cuales ayudarían a mejorar la experiencia de usuario y la usabilidad de la herramienta.

CONCLUSIONES

1. La migración de la información a una sola base de datos publica dentro de la nube, da solución por completo a la problemática de la distribución de datos a los usuarios de DATAFORG, esto garantiza que la información consultada por cualquier persona estará actualizada en todo momento. Anteriormente podían existir muchas versiones del mismo software utilizando una base de datos local desactualizada.

2. Gracias al tipo de aplicación desarrollado, se amplió el alcance que el software tiene con los usuarios actuales y potenciales. Al permitir el acceso a la información desde cualquier tipo de dispositivo, se incrementará el número de usuarios interesados en utilizar el software.

3. Con la solución implementada para conservar información en memoria cache se extiende el soporte de la herramienta a los usuarios que no cuentan con una conexión a internet estable en todo momento. Con frecuencia los usuarios que utilizan la herramienta hacen estudio de campo, por lo que la conexión a internet casi siempre es una limitante, no contando con ella, o si cuentan con acceso a la red, es altamente probable que el enlace a internet no sea estable y limite el uso de herramientas para consultar datos.

4. Adicional al soporte sin conexión a internet con el que la aplicación cuenta, la exportación de información en formato PDF facilita almacenar y compartir datos contenidos en la base de datos de especies del INAB, en ocasiones es necesario compartir información con más personas, sin

embargo, no todos están dispuestos a descargar aplicaciones en sus dispositivos, por lo que la alternativa más común para compartir datos son archivos de texto plano.

5. Los encargados del proyecto ahora pueden modificar y añadir información a su catálogo de datos gracias al software administrativo que complementa todo el sistema. Este módulo ayudara a cualquier tipo de usuario con un mínimo de experiencia en software a poder dar mantenimiento a la base de datos, y con ello mantener la información actualizada en todo momento.

RECOMENDACIONES

1. Mantener una comunicación constante y efectiva con todos los interesados del proyecto debe ser indispensable, ya que algunas veces durante el desarrollo de la herramienta no se sostuvo dicha comunicación, provocando así atrasos y acumulación de modificaciones en módulos y componentes del software.

2. Contar con un historial de las modificaciones, observaciones y solicitudes de los interesados en el proyecto ayudaría a poder trasladar lo discutido y trabajado durante el desarrollo de la herramienta a nuevos encargados de área. Mientras se llevaba a cabo el proyecto, personal del área a cargo del software fueron sustituidos, lo que ocasionaba retroceder e invertir tiempo explicando y trasladando esta información a los nuevos miembros de la institución.

3. Tomar en cuenta el tiempo de holgura, facilitará aceptar nuevos requerimientos o realizar modificaciones necesarias en las partes del proyecto que se estén desarrollando, estimar tiempo extra ayudará a cumplir con los tiempos de entrega acordados y satisfacer todas las necesidades de los interesados.

4. Adoptar una metodología de desarrollo continuo agilizará y ayudará a mantener un proyecto en constante cambio y desarrollo, pudiendo atender las solicitudes iniciales de los clientes en su totalidad y también tomar e integrar nuevos objetivos que surjan durante el desarrollo del software.

REFERENCIAS

1. Group, T. H. (2022). Hsqldb. Obtenido de https://hsqldb.org/

2. MDN, c. (2022). MDN Web Docs. Obtenido de https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

3. MDN, C. (2022). MDN Web Docs. Obtenido de https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs

4. MDN, C. (2022). MDN Web Docs. Obtenido de https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen

5. MDN, C. (2022). MDN Web Docs. Obtenido de https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

6. MDN, C. (2022). MDN Web Docs. Obtenido de https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

7. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-

us/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

8. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/es-es/aspnet/core/blazor/?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0

9. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/blazor/progressive-web-app?view=aspnetcore-6.0&tabs=visual-studio

10. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/blazor/blazor-server-ef-core?view=aspnetcore-6.0

11. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/blazor/webassembly-lazy-load-assemblies?view=aspnetcore-6.0

12. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/existing-database

13. Microsoft, C. (2022). Microsoft. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/razor-pages/razor-pages-conventions?view=aspnetcore-6.0

14. WebAssembly. (s.f.). Web Assembly. Obtenido de https://docs.microsoft.com/en-us/aspnet/core/razor-pages/razor-pages-conventions?view=aspnetcore-6.0

APÉNDICES

Apéndice 1. **Listado de requerimientos iniciales**

- Especies forestales
  - Listado de especies forestales: debe permitir visualizar el listado completo de especies, ordenado alfabéticamente, al interactuar con una de ellas se accede a cada una vista de las propiedades asociadas a esa especie, cada propiedad puede o no tener datos divididos en subcategorías.
  - Filtros de especies forestales: el listado de especies forestales debe poder ser filtrado a través de distintos parámetros, estos parámetros ya se encuentran establecidos por la aplicación Dataforg existente, las categorías de filtros disponibles son:
    - Por especies
    - Por ecología
    - Por geografía
    - Por fenología
    - Por propiedades físicas
    - Por usos maderables
    - Por usos alternativos
  - Filtros por especies: dentro de estos filtros se contemplan todos aquellos relacionados con la descripción general de la especie, los cuales son:
    - Nombre común
    - Sinónimo de nombre común
    - Nombre científico

Continuación Apéndice 1.

- Sinónimo de nombre científico
- Familia
- Género

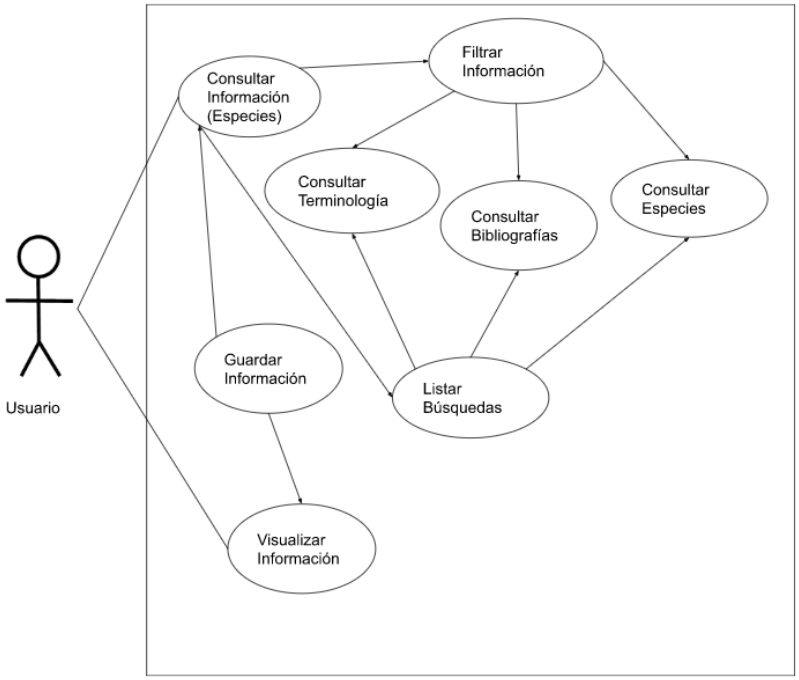
- Filtros por ecología: en esta categoría se encuentran las características relacionadas al hábitat de la especie:
  - Zona de vida
  - Requerimientos ambientales
- Filtros por geografía: filtros relacionados al área geográfica que con frecuencia ocupa una especie:
  - País
  - Departamento
  - Unidad de manejo
- Filtros por fenología: información relacionada a la flora y fructificación de una especie:
  - Fechas en que inicia
  - Fechas en que termina
- Filtros por propiedades físicas: en esta categoría se puede filtrar por el peso específico de una especie.
- Filtros por usos maderables: dentro de este filtro se listan los posibles usos maderables que una especie puede tener, al ingresar a uno se desplegaran las entidades que cumplen con el uso seleccionado.
- Filtros por usos alternativos: similar al filtro por usos maderables, contemplando el uso alternativo que se le puede dar a una especie.
- Visualización de la información de una especie: por cada especie existente en la base de datos, el usuario deberá poder consultar las siguientes categorías:
  - Información general

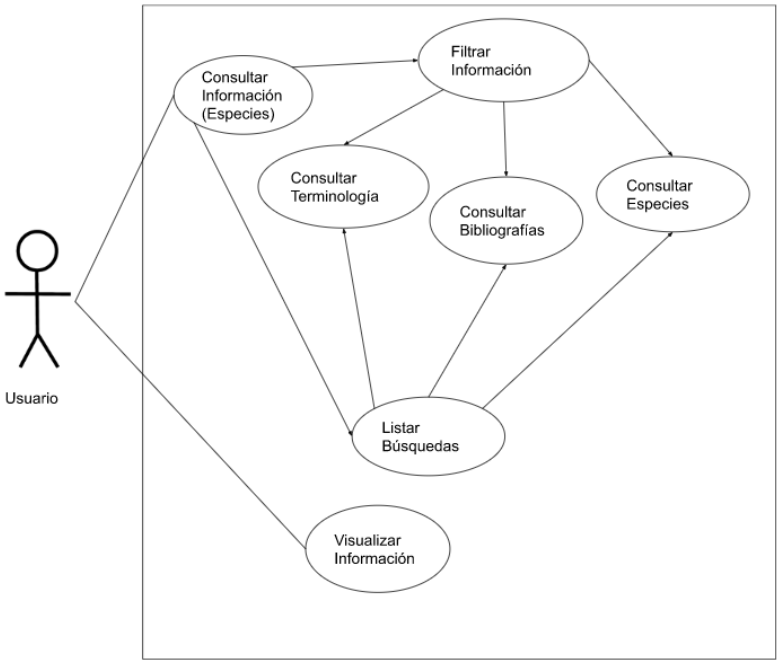
Continuación Apéndice 1.

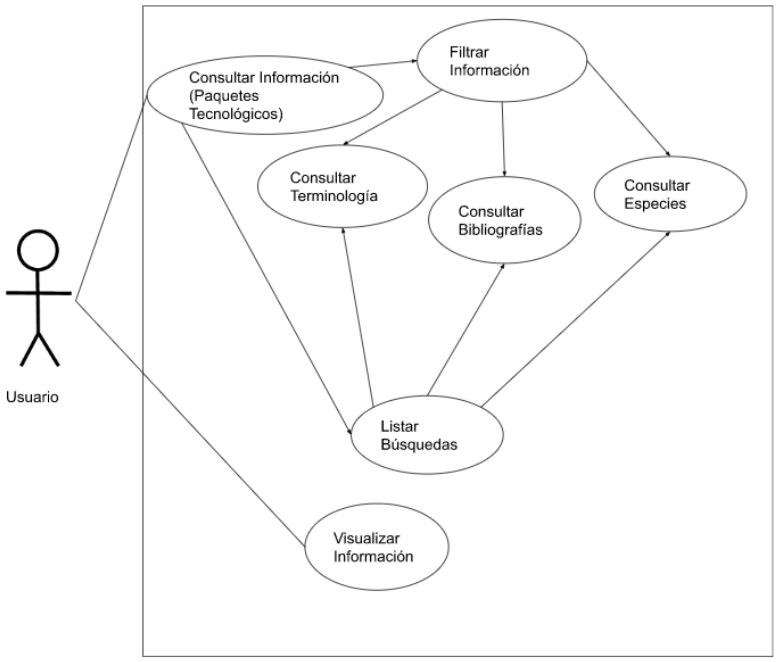
- Información taxonómica
- Información botánica
- Información geográfica y existencias
- Información ecológica
- Reproducción
- Características de la madera
- Usos y mercadeo
- Imágenes

- Consulta de fuentes bibliográficas: Apartado que permite listar y buscar referencias bibliográficas de las distintas fuentes utilizadas para alimentar la base de datos.
- Consulta de empresas forestales: Módulo que permite listar y buscar el contacto de las distintas empresas forestales registradas en la base de datos de la institución.
- Reportes: los reportes que se pueden extender son relacionados a los listados que el usuario en un momento determinado haya filtrado o buscado, estos se exportan en formato PDF el cual podrá ser descargado desde la aplicación y posteriormente almacenado o impreso por el usuario.

Apéndice 2. **Diagramas de casos de uso**







Fuente: Elaboración propia, realizado con diagrams.net

Apéndice 3. **Diagrama entidad relación de base de datos**

El diagrama entidad relación de la base de datos es muy grande, que al minimizarlo no se nota ninguna descripción, por lo cual comparto el enlace: https://ibb.co/Fxm4Czp