UNIVERSIDAD DE ANTIOQUIA

DESARROLLO DE UNA APLICACIÓN WEB ADAPTATIVA PARA EL DESPLIEGUE DE INFORMACIÓN ASOCIADA A LA CAPA APLICATIVA DEL MODELO DE REFERENCIA DE LA ARQUITECTURA ACTUAL Y OBJETIVA

Autor

Esteban Cadavid Rodríguez

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería Electrónica y Telecomunicaciones

Medellín, Colombia

2021

Desarrollo de una aplicación web adaptativa para el despliegue de información asociada a la Capa Aplicativa del Modelo de Referencia de la Arquitectura Actual y Objetiva de la empresa Bancolombia.

Esteban Cadavid Rodríguez

Informe de práctica como requisito parcial para optar al título de:
Ingeniero electrónico

Asesores (a):
Juan Fernando Hincapié, Líder TI Bancolombia
José Edinson Aedo Cobo, Profesor UdeA

Universidad de Antioquia
Facultad de ingeniería, Departamento de ingeniería Electrónica y Telecomunicaciones.
Medellín, Colombia
2021.

Tabla de Contenido

Resumen	3
1. Introducción	4
2. Objetivos	5
2.1. Objetivo general	5
2.2. Objetivos específicos	5
3. Marco teórico	6
3.1. Mapa de capacidades de negocio	6
3.2. Aplicación web	7
3.2.1. Backend	8
3.2.1.1. JavaScript	8
3.2.1.2. Node.js	9
3.2.1.2.1. Express	9
3.2.2. Base de datos relacional	9
3.2.3. Métodos de petición HTTP	10
3.2.4. Frontend	10
3.2.4.1. Angular	11
3.2.4.1.1. Typescript	11
4. Metodología	11
4.1. Determinación de la base de datos	13
4.2. Diseño del backend	16
4.2.1. Configuración del servidor	17
4.2.2. Diagrama de árbol	17
4.3. Diseño del frontend	18
4.4. Pruebas unitarias y funcionales	18
5. Resultados y análisis	19
5.1. Base de datos	19
5.2. Diseño del aplicativo	20
5.3. Pruebas funcionales y unitarias	30
5.4. Análisis	33
6. Conclusiones	34
Referencias Bibliográficas	35

Resumen

Cuando se dirigen todas las estrategias de negocio de un área tan grande como lo es la Dirección de Estrategia, Arquitectura e Innovación de TI de Bancolombia se hace necesario encontrar métodos para hacer llegar la información de manera asertiva tanto a empleados internos como colaboradores implicados en el desarrollo de las funciones y actividades. Para este caso específico, la necesidad consistía en disponibilizar la información de la jerarquía en la que se dividen todos los productos y servicios que presta el banco y cuáles son las aplicaciones que desempeñan la solución a dichos servicios o aplicaciones, tanto actualmente como una proyección de lo esperado en el futuro.

En este trabajo se desarrolló un aplicativo web responsivo, dedicado a facilitar el acceso a esta información para todos los posibles interesados dentro del grupo Bancolombia, haciéndola aprovechable de forma ágil, ordenada, concreta y de fácil actualización, para que todas las partes que puedan requerirla se puedan ubicar e informar de cualquier factor específico relacionado con los servicios o productos que el banco despliega. Este aplicativo estructura toda su información en una base de datos relacional, que facilita la relación la información entre las tablas de la base de datos, evitando la creación de registros innecesarios en el relacionamiento. La base de datos fue implementada usando el gestor de bases de datos de "MySQL". Para administrar toda esta información se utiliza la estructura de los métodos de petición HTTP, que se encargan de recibir los pedidos realizados por URLs para así leer, escribir, editar o eliminar registros dentro de la base de datos (en el Backend) y finalmente para que el usuario pueda utilizar la aplicación y acceder a la información. se desarrolló una interfaz responsiva con Angular, donde se presenta la información en forma de listados y de un esquema, para que los usuarios puedan tener una mejor apreciación de la jerarquía y obtenga toda la información de forma clara de los productos y servicios prestados.

De esta manera, se diseño la estructura idónea para una base de datos funcional, que con su relacionamiento cumple a cabalidad con las condiciones requeridas por la solución, para ser empleada en un aplicativo que administra toda la base de datos y la lleva de manera precisa y fácil de entender a los empleados o colaboradores. El aplicativo por lo tanto genera valor agregado a la dirección de Estrategia, Arquitectura e Innovación de TI

en la forma como se suministra la información de los productos y servicios que presta el banco.

1. Introducción

La Dirección de Estrategia, Arquitectura e Innovación de TI se encarga de materializar las estrategias del negocio del grupo Bancolombia, a través de la definición y planeación del norte tecnológico para la compañía. Bancolombia en toda su trayectoria ha sido un pilar fundamental en la transformación de modelos y la innovación dentro del mercado, lo que ha hecho necesario su adaptación a metodologías ágiles que promueven la interacción con los clientes para el desarrollo de proyectos y productos, facilitando el acceso a la información de todos los servicios que se brindan y a las aplicaciones disponibles en la capa aplicativa. Aprovechando los canales virtuales, Bancolombia suministra la información a todas las personas interesadas, bien sean colaboradores o miembros del grupo.

Un medio que permite acceder de manera ágil, organizada y dinámica la información, es una aplicación web, donde se puede presentar de manera interactiva y esquemática, todo lo relacionado con la capa de aplicaciones de la empresa, la cual en este caso, se encuentra dividida en dominios, subdominios, capacidades y productos/servicios que se prestan.

Una aplicación web en general se puede dividir en dos partes: la parte que interactúa con el usuario (Frontend) y la parte interior de las aplicación, en el lado del servidor (backend), desarrollado generalmente como una API REST. En el frontend se despliega todo lo que usuario accede (las funciones) en el sitio web o aplicación. A través del frontend el usuario interactua, visualizar y acceder, de forma que se adapte a la resolución de la pantalla de cada dispositivo del que se vaya a visualizar (página adaptativa). En el backend se implementan todos los procesos y administra toda la estructura de datos de la aplicación mediante el uso de bases de datos.

Para el desarrollo de aplicaciones Web se pueden utilizar diferentes lenguajes como HTML, CSS, Laravel y JavaScript, entre otros. Estos lengujes, junto a sus APIS y frameworks, brindan un conjunto de herramientas (APIS, frameworks) que facilitan el diseño tanto del backend, como el frontend. En este trabajo se utilizará el framework node.js con su modulo express, debido a que facilita el uso de bases, tanto relacionales (SQL, MySQL, entre otras), como no relacionales (mongodb, CouchDB, entre otras).

En este proyecto se busca desarrollar la aplicación usando los lenguajes anteriormente mencionados, que permita mostrar toda la información de la capa aplicativa del banco, con todos sus dominios, subdominios, capacidades y aplicaciones de una manera organizada y de fácil acceso. Esta aplicación permitirá que cualquier persona de la organización pueda conocer las capacidades de la capa de aplicaciones de Bancolombia.

Este proyecto ha sido estructurada de la siguiente manera: en la sección 2, se describen los objetivos del trabajo; en la sección 3 se presenta el marco teórico donde se describe las diferentes técnicas para la extracción de la información y las ayudas empleadas para cumplir los requerimientos; luego en la sección 4, se describe la metodología que se empleó en el desarrollo del proyecto, la cual se complementa con los resultados y su análisis de la implementación en la sección 5; Finalmente en la sección 6 se detallan las conclusiones que se obtuvieron en el desarrollo del proyecto.

2. Objetivos

2.1. Objetivo general

Desarrollar y llevar a cabo una prueba piloto de una aplicación web adaptativa para desplegar y consultar la información detallada de cada aplicación del Modelo de Referencia de la Arquitectura Actual y Objetiva de la empresa Bancolombia, utilizando el framework Angular para el desarrollo del frontend de la aplicación y Node.js junto con su API Express para la implementación del Backend y la base de datos requerida.

2.2. Objetivos específicos

1. Identificar la estructura de la base de datos y los servicios entregados por la empresa que estarán en la aplicación web adaptativa, con la participación de empleados y miembros de la Dirección de Estrategia, Arquitectura e innovación de TI, considerando todo el mapa de capacidades de la organización con el fin de realizar el diseño del modelo, levantar los requisitos, características y funcionalidades para darle estructura a la aplicación.

2. Diseñar el modelo, la arquitectura y la estructura de datos de la aplicación web adaptativa, identificado las necesidades de información que requiere cada requisito, característica y funcionalidad

definida en el objetivo no.1, y estableciendo el diagrama entidad relación (DER) donde se definen las tablas de datos, los campos en cada tabla y las relaciones entre cada campo y cada tabla de la base de datos relacional a emplear en la aplicación.

3. Implementar la aplicación utilizando Angular, usando JavaScript/typescript para el desarrollo del frontend y node.js junto con su API express y mysql para el desarrollo del backend y llevar a cabo pruebas para verificar su funcionamiento y el cumplimiento de los requisitos definidos.

4. Realizar un piloto con miembros del equipo de Estrategia, Arquitectura e Innovación de TI, para evaluar el valor agregado de la aplicación desarrollada en el interior del grupo Bancolombia.

3. Marco Teórico

La solución al problema planteado se basa en la necesidad de visualizar del modelo de referencia de arquitectura aplicativa de Bancolombia el cual se estructura como un mapa de capacidades del negocio. En el proceso de desarrollo se diseñará una base de datos acorde a las necesidades planteadas. También se desarrollará el aplicativo web responsivo, que se divide en dos partes, una para el backend y otra para el frontend. A continuación, se describen los componentes y tecnologías usadas en el desarrollo del proyecto.

3.1. Mapa de capacidades de negocio

El mapa de capacidades es una estructura en orden jerárquico que presenta la definición de todas las actividades y servicios que la empresa es capaz de realizar en su modelo de negocio. El mapa integra lo que se quiere hacer con el negocio junto al modelo operativo de cómo se quiere realizar, lo cual hace parte central y fundamental de la arquitectura de negocio. Este tipo de mapa tiene como obejtivo mostrar las capacidades que tiene la empresa, las cuales se entienden como acciones concretas o actividades con un propósito definido para producir un resultado, pero sin incluir detalles de su ejecución. Estas capacidades pueden ser misionales, estratégicas o de funcionamiento [1].

El despliegue de esta información se puede realizar a través de un entorno web que muestre todas las aplicaciones que utiliza el banco, divididas por dominios y subdominios por su gran extensión, y así poder disponer de un directorio organizado y dinámico. Para el caso particular, obeservando la figura 2 y 3 los dominios son la primera división jerárquica que se puede apreciar (En las figuras son los grupos representados en color gris.), habiendo dos posibles casos, el primero, el dominio de "Servicio de clientes", contiene directamente las capacidades/productos prestados en el dominio (contiene los recuadros de color negro y rojo, que son los aplicativos para la solución) y la segunda opción, como el caso de "Servicios de producto", contiene subdominios (recuadros de tono verde más oscuro, como por ejemplo "Cuentas AFC"). Dentro de estos últimos, se encuentran las capacidades/soluciones (Recuadros de un tono más claro) que contienen los aplicativos que dan solución a estos, en recuadros distinguidos por colores, negros y rojos, donde los negros son aplicaciones que no tienen proyección a futuro y los rojos son aplicaciones que se tiene certeza que se continuaran.

3.2. Aplicación Web

Son aquellas aplicaciones que se codifican en lenguajes de programación que son soportadas e interpretables por navegadores web. En estas aplicaciones el usuario accede a un servidor web a través de internet o de una intranet mediante el uso de un navegador [2].

La web funciona con la arquitectura cliente (que es cualquier computador que desee consultar la web) y servidor (que es el encargado de suministra la información desde que el cliente tenga los permisos). El desarrollo de este tipo de aplicaciones normalmente se estructura en tres capas: la primera capa o (también denominada frontend) constituye la interfaz a la cual accede el usuario y donde interactúa con las funciones o servicios desplegados por la aplicación; la segunda capa está constituida por un traductor, que permite la comunicación y la administración de datos en aplicaciones distribuidas y puede utilizar una tecnología web dinámica, y por último la tercera capa o capa interna, (también denominada Backend) que se encarga de todos los procesos para que la aplicación funcione de manera adecuada, entre estos se destaca la conexión con las bases de datos y la comunicación con el servidor de hosting. De manera simplificada, el navegador web envía peticiones a la segunda capa, que ofrece servicios

de traducción al backend, el cual se vale de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario [2]

3.2.1. Backend

El backend es la parte invisible de la aplicación web, donde se da respuesta a todas las peticiones que el usuario requiere, es el interior de las aplicaciones, por lo que se le denomina el lado del servidor; es donde se encuentra almacenada la lógica de manejo de los datos, los enlaces y el acceso, y edición a la base de datos, que es donde se alberga la información que será procesada, recibida y desplegada.

Para desarrollar esta parte de las aplicaciones web existen diversos lenguajes de programación, como lo son ASP.NET, PHP, Python, Ruby, Node.js, Java, Laravel, entre otros. [3]. En este proyecto se pretende usar un solo lenguaje de programación en el desarrollo, tanto del back como del frontend, por lo cual se seleccionó JavaScript, debido a que se dispone de una mayor experiencia.

3.2.1.1. JavaScript (js)

Es un lenguaje de programación basado en prototipos, que trabaja en un solo hilo, es dinámico y soporta programas desarrollados tanto en programación orientada a objetos, como funcional. Es un lenguaje de programación que se puede aplicar a un documento HTML y crear interactividad entre sitios web. JavaScript se diseñó con una sintaxis similar a la del lenguaje de programación C, aunque adopta nombres y convenciones del lenguaje de programación Java. Al ser uno de los lenguajes más populares en la comunidad de desarrolladores, actualmente tiene una gran cantidad de documentación, de APIS y frameworks que facilita el proceso de desarrollo en el ambiente web. Teniendo en cuenta lo previamente mencionado, se utilizará Node.js, que es un entorno de ejecución JavaScipt, debido a que en el desarrollo del proyecto se quiere usar el mismo lenguaje para tener una mejor comprensión y poder mantener el código actualizado [5].

3.2.1.2. Node.js

Es un entorno de ejecución para JavaScript orientado a la creación de aplicaciones web escalables con eventos asíncronos, lo que nos permite tener muchas conexiones simultáneamente, teniendo *callbacks* por cada llamada que se hace. Permite también ejecutar más acciones en la medida que la conexión se realiza y entrar en modo de reposo si no se está realizando ninguna tarea. Como tiene tan pocas funciones tipo entrada/salida los procesos escasamente sufren un bloqueo, lo cual facilita desarrollar sistemas escalables [6].

Para facilitar más aún el desarrollo del *backend*, se puede utilizar también una API de Node.js llamada ***Express* [7],** la cual es una infraestructura de aplicaciones para Node, disponiendo de muchos métodos tanto de HTTP como de middleware lo que facilita la conexión entre el back y el *frontend*.

Una parte fundamental en el diseño de la aplicación es la determinación de la base de datos y su diseño. En este proyecto se decidió utilizar una base de datos relacional, que a continuación se describe con mayor detalle.

3.2.2. Base de datos relacional

Es tipo de base de datos permite almacenar y dar acceso a conjuntos de datos que están relacionados entre sí, pudiendo ser representado en tablas. Se debe considerar una base de datos relacional para cualquier necesidad de información cuendo los puntos de datos se relacionen entre sí y se deban administrar de una manera segura, consistente y basada en reglas. Este tipo de bases de datos suelen ser configuradas usando los lenguajes **SQL** que significa en español, lenguaje de consulta estructurada que se usa para administrar, diseñar y recuperar administración de donde está almacenada. Existen diferentes bases de datos que pueden ser empleadas con SQL Server, MySQL, PostgreSQL, entre otras. En este proyecto se seleccionó **MySQL** debido a que es una de las más populares para todos los entornos de desarrollo web [8]**.**

Una vez seleccionada la base de datos, para poder llevar a cabo las peticiones de leer, escribir o modificar la base de datos, se deben disponer

de un conjunto de métodos para que el backend interprete lo que la interfaz de usuario requiere y pueda ejecutarlo correctamente en la base de datos, por lo cual serán utilizados los métodos http.

3.2.3. Métodos de petición http

Los métodos de petición http son solicitudes que se realizan para ejecutar una acción en un recurso concreto, como puede ser el acceso a una página concreta o el envío de los datos. Estos consisten en una serie de mensajes que un cliente web realiza y que conllevan una acción en el servidor web a las que se dirigen. Cada uno de los métodos viene expresado por una palabra y se utilizan en diferentes circunstancias. En el desarrollo de la solución se utilizaron los métodos más comunes: GET (que se utiliza cuando se necesita solo consultar información), POST (Para solicitar la creación de un nuevo registro), PUT (que se emplea cuando se necesita actualizar por completo un registro existente) y DELETE (utilizado para eliminar un registro existente) [9]. El empleo de estos métodos es un factor indispensable en el funcionamiento de la arquitectura utilizada en este trabajo. En el se utiliza una arquitectura REST, que se encarga de aportar buenas prácticas para comunicar lo que desea el usuario al sistema, para que este comprenda la solicitud y la cumpla [10]. Por ejemplo el método GET mencionado solo se utilice para obtener información de la base de datos, El POST solo para crear nuevos registros y así para el resto de peticiones, constituye un práctica recomendada, la cual debe ser consecuente con el verbo de la petición y su función [16].

En el proyecto, una vez desarrollada toda la parte funcional de la aplicación, se diseña una interfaz donde el usuario puede acceder a la información y a los servicios que se pueden generar desde peticiones en el *frontend.*

3.2.4. Frontend

Como se describió previamente, el *frontend* es la parte de la aplicación donde los usuarios pueden acceder a la información y a las utilidades (funciones) que se han habilitado. En términos generales, el *frontend* constituye todo lo que el usuario puede realizar cuando accede a un sitio web. Comprende las funcionalidades de los botones, de las cajas de texto, los movimientos y todas las demás interacciones que puede tener cuando accede a un sitio o una aplicación web. Lo fundamental que se debe tener

en cuenta a la hora de diseñar el frontend de una aplicación web básica, son los conceptos de HTML (y CSS) que es un lenguaje de etiquetas que permite moldear la base de la página web y desplegar todos los elementos que sean de utilidad o de visualización con la estructura que se desee [11]. CSS, que se puede traducir como de hojas de estilo en cascada, es un lenguaje que permite realizar una presentación de documento, posibilitando así marcar la separación del contenido y la forma en la que este se presenta, ya sean colores, formas, sombras, entre otros [12].

Como se mencionó previamente, se quiere mantener el mismo lenguaje de programación para toda la aplicación y también implementar los recursos que suministran sus APIS y sus frameworks, por lo cual para el diseño del frontend de la aplicación web, se utilizó el Framework Angular [13].

3.2.4.1. Angular

Es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página. Emplea un patrón de trabajo MVC (modelo de vista y controlador) evitando escribir código repetitivo. Como la aplicación web a desarrollar es de una sola página, ésta solo se carga una vez y Angular se encarga de cambiar entre los renders (vistas) realizados en el navegador de forma dinámica y así obtener la apariencia de una web normal [13]. El lenguaje principal de programación de Angular es **Typescript** (El cual es un superconjunto de JavaScript, que extiende su sintaxis y añade objetos basados en clases y tipos estáticos.). Con esta infraestructura se garantiza que toda la sintaxis y el modo de implementación del código sea similar, lo que proporciona coherencia y consistencia en la información, permitiendo, por ejemplo, la incorporación de nuevos programadores, en caso de que sea necesario en el proyecto.

4. Metodología

Para el desarrollo del proyecto se utilizó una metodología de desarrollo en cascada (Ilustrada en la Figura 1) exceptuando el último paso que es el de mantenimiento del aplicativo, que por motivos obvios no se considera en este proyecto de corta duración. En la primera fase, se hizo un análisis a todos los requisitos pedidos para el desarrollo, entre los cuales se encuentra que fuera una aplicación que se adaptara a cualquier resolución, que

tuviese validación de campos, comprobaciones en las peticiones HTTP y comprobaciones para casos atípicos. En la parte del diseño, se llevaron a cabo todas las consideraciones necesarias para que las necesidades fueran resueltas y se tomó la decisión de con qué lenguajes de programación y con qué herramientas se trabajaría y así hacer una planeación de lo que se debía desarrollar. También se definieron las funcionalidades y como se debería presentar las pantallas al usuario. En la fase de implementación se realizó la codificación y el despliegue del aplicativo y al final se realizaron las pruebas que verificaban su correcto funcionamiento en los casos requeridos.

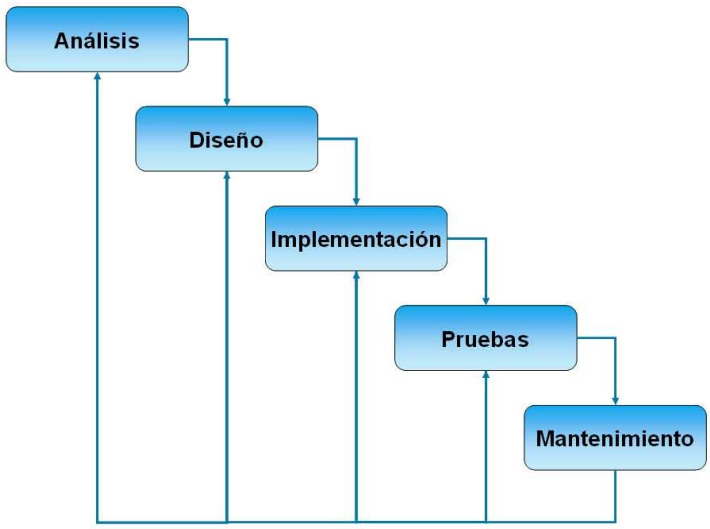

Figura 1. Ilustración de la metodología en cascada.

Como primer paso en el desarrollo de la aplicación se definieron un conjunto de historias de usuario. Cada una de estas representa un requisito funcional, servicio o parte del sistema con el que los clientes interactuarían y sirven al desarrollador para resumir el lenguaje natural de un interesado en un formato fácil de entender para cualquier persona involucrada en el desarrollo. Debido a que la definición de estos requisitos es la parte inicial del proceso de desarrollo, cada historia de usuario fue escrita en colaboración con los usuarios para así garantizar un correcto entendimiento de las necesidades de estos.

1. Como empleado quiero consultar la jerarquía de dominios en la que se dividen los productos y servicios que presta el banco para saber a donde tengo que acudir en caso de que algún usuario presente un inconveniente con uno de sus productos o servicios.

2. Como empleado quiero ver el orden de la jerarquía en forma esquemática o grafica para que así me resulte más cómodo ubicarme entre direcciones y subdirecciones sin perder el hilo al momento de hacer una búsqueda.
3. Como directivo necesito poder editar toda la información correspondiente a la capa aplicativa del banco para hacerla llegar de manera inmediata a todos los empleados y colaboradores cada que exista un cambio y así poder introducir nuevos, mejorar los existentes o borrar los que no se utilizan.

4.1. Determinación de la Base de datos.

El esquema de la base de datos proporcionada de manera inicial por Bancolombia ilustraba la jerarquía donde se encuentran ubicados los servicios y productos que presta la institución y los aplicativos que al momento de hacer dicha presentación solucionaban la problemática por la que los clientes venían al banco, y en contraste se mostraba lo que a ese día aspiraba a tener la empresa para poder dar solución a los productos y servicios que prestan, como se ilustra en la figura 2 y 3.

En la figura 2 se muestra el modelo de arquitectura aplicativa de Bancolombia (esquema de la base de datos). En ella se ilustra la jerarquía donde se encuentran ubicados los servicios y productos que presta la institución y los aplicativos para atender las necesidades de los clientes en el banco. La figura 3, muestra como el banco quiere proyectar la solución de sus servicios y productos, es la representación de qué aplicativos quiere emplear el banco para sus soluciones en un tiempo futuro.

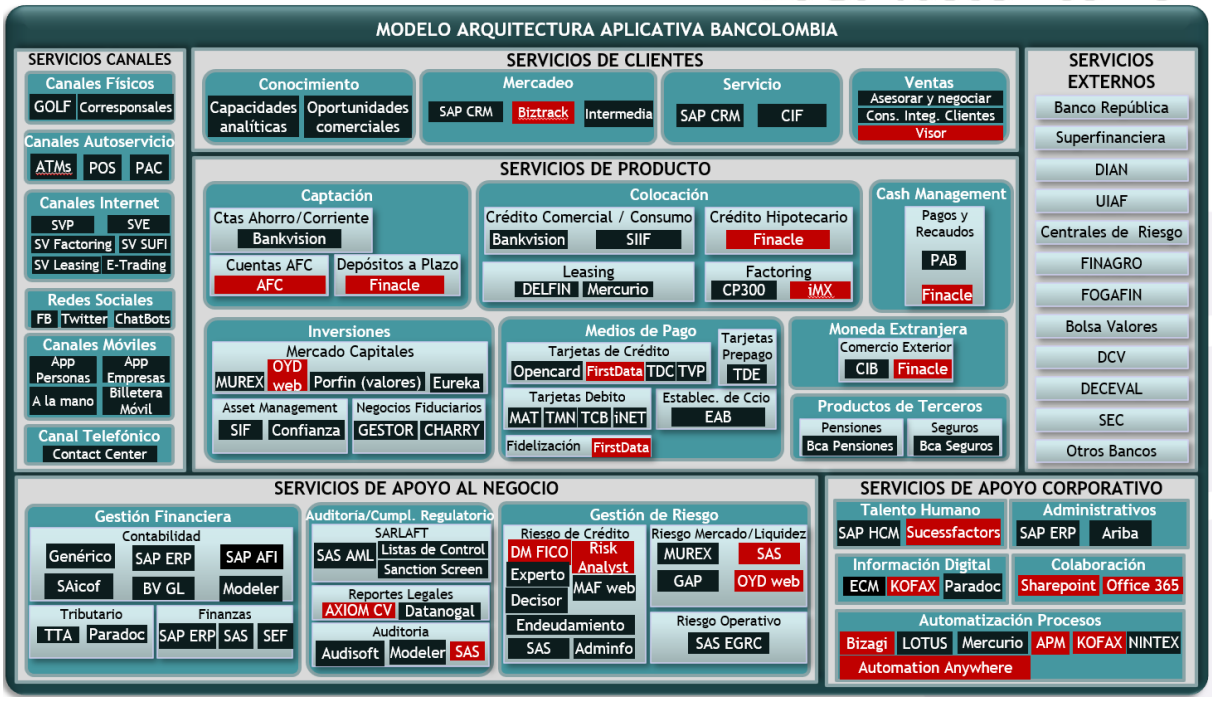

Figura 2. Modelo de arquitectura aplicativa actual suministrado por Bancolombia.

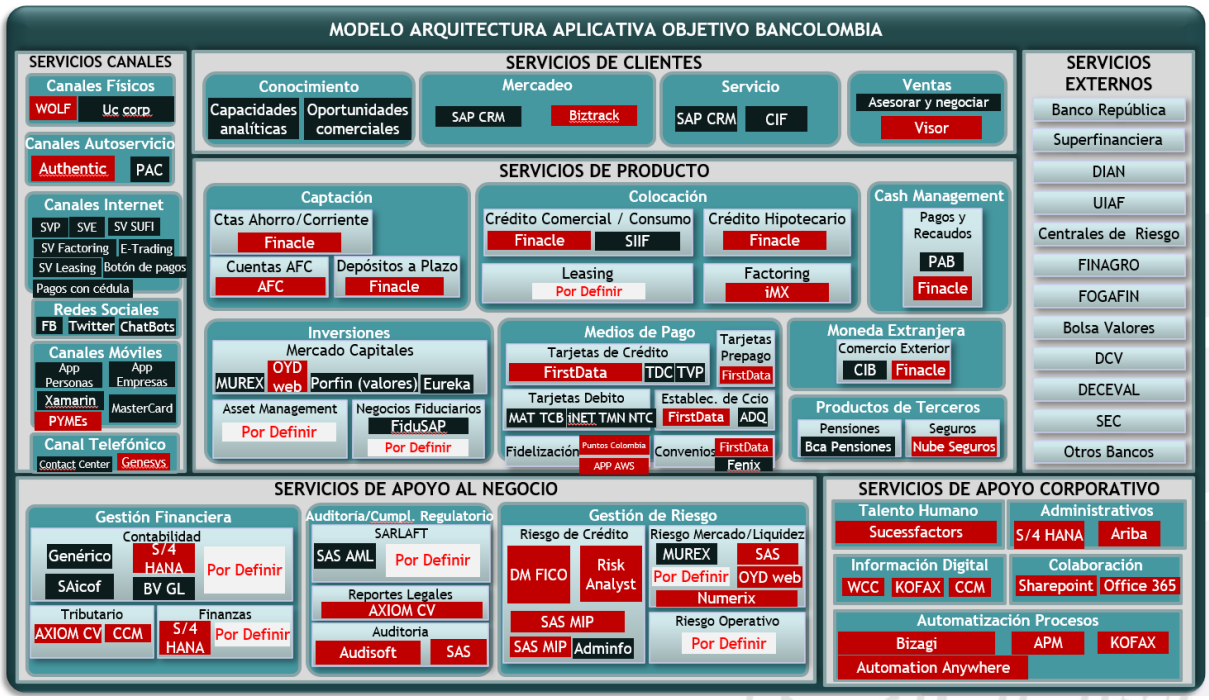

Figura 3. Modelo de arquitectura aplicativa objetivo suministrado por Bancolombia.

Se hace la aclaración de que toda esta información suministrada por la empresa fue estructurada hace algunos años (2018), por los cual se estableció que se encuentra desactualizada.

A partir de la información brindada, se estructuró la base de datos de la aplicación. Se definieron 3 tablas de información, una llamada "Servicio" que incluye los grandes dominios que contienen todos los productos y servicios que se prestan. Luego se definió otra tabla denominada "Nivel", con una relación de varios a uno con la tabla servicio y por último se definió la tabla "Aplicación" con una relación de muchos a muchos con la tabla "Nivel". También se requirió, adicionalmente una tabla intermedia para romper la relación muchos a muchos, debido a que las relaciones deben estar normalizadas (cada registro debe ser atómico, ósea no se puede tener información compuesta), buscando de este modo que las consultas SQL sean óptimas, tanto para el motor que las ejecuta, como para quien las escribe.

En la figura 4 se muestra el DER (*Diagrama Entidad Relación)* de la base de datos diseñadas para la aplicación, donde las variables identificadas con numeral ("#") se refieren a las claves primarias, las cuales son la

identificación exclusiva de cada fila de una tabla. De esta manera, se utilizan como un identificador único para cada fila de la tabla permitiendo el relacionamiento entre las tablas cuando sea necesario. Las flechas de relacionamiento que se muestran en la figura expresan la relación que tienen las tablas entre sí. La tabla de servicio se relaciona con la de nivel con una relación de uno a muchos, lo que implica que uno de los valores de la tabla "Servicio" puede contener varios de los registros de la tabla "Nivel". La tabla "Aplicación" y la tabla "Nivel" tienen una relación muchos a muchos, lo que significa que un registro de la tabla "Nivel" puede tener relación con varios de los registros de la tabla "Aplicación" y viceversa. Finalmente se crea la tabla intermedia "Nivel/Aplicación" relacionando estas dos tablas por sus claves primarias. La flecha que sale y entra de la tabla "Nivel" significa que tiene un relacionamiento con ella misma, debido a que se van a almacenar tanto los subdominios de cada dominio de la tabla servicios (Que se relacionan con la tabla "Servicio" con una clave foránea, que en este caso es "id_servicio" y la clave primaria de Servicio) y las capacidades que tiene cada uno de los subdominios. Nótese que cada subdominio puede tener varias capacidades y esta relación se da mediante el campo "Id_suplevel". Este relacionamiento tiene sentido, debido a que evita registrar demasiadas combinaciones posibles entre todos los valores que deben de ser relacionados dentro de la base de datos.

En el DER de la base de datos de la figura 4 no se incluyeron las tablas "Usuario", "Aplicación_objetivo" y "Nivel_Aplicación_objetivo", debido a que la tabla de Usuario no tiene relación con ninguna otra de las tablas de la base de datos, aquí solo se almacenan los datos de login para poder acceder a la edición de la base de datos desde el frontend (Será explicado con más detalle en la sección de login) y la lógica con las tablas de "Aplicación_objetivo" y "Nivel_Aplicación_objetivo" es la misma de la anteriormente explicada para las tablas de "Aplicación" y "Nivel_Aplicacion".

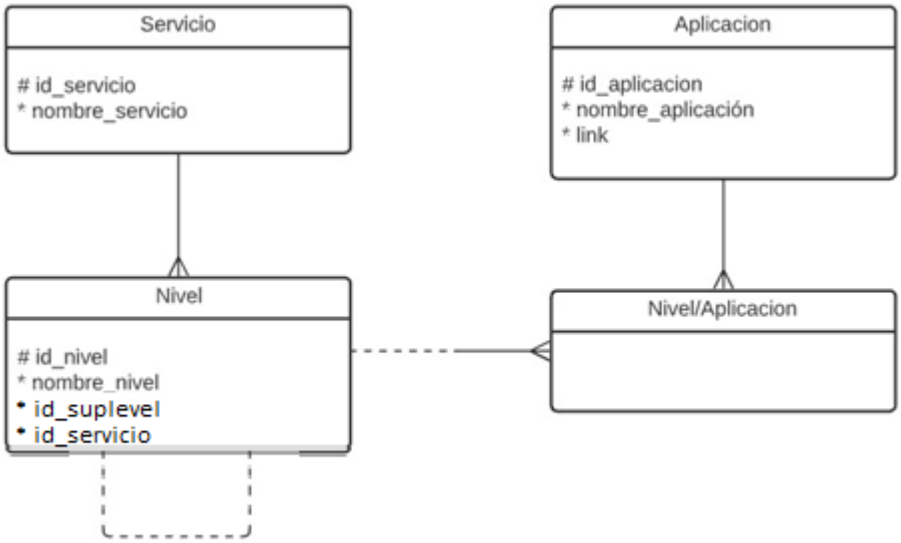

Figura 4. Diagrama entidad relación (DER) de la base de datos empleada.

4.2. Diseño del *Backend*

Luego de establecido el DER, se debe encontrar la manera de administrar toda la información contenida en la base de datos. Esta función constituye el principal proceso del *backend* de un aplicativo web, donde se busca acceder a la información que se solicita, a través de la aplicación, para luego combinarla y devolverla de acuerdo con las peticiones del usuario en *frontend*. Para el desarrollo del *backend* se usó Node.js, que es uno de los entornos de ejecución de JavaScript orientado a crear aplicaciones escalables en red [6]. Se desarrollaron dos partes fundamentales en el *backend* del aplicativo: la creación del servidor del aplicativo para atender a todos los llamados que solicite el usuario y un diagrama de árbol para representar en un esquema la jerarquía de dominios, subdominios, capacidades y aplicaciones.

Una consideración importante era el hecho de que se pudiera tener control de la ejecución de los métodos HTTP, para así poder actuar acorde a las posibles fallas, mostrando en la consola del navegador un mensaje de "Ok" cuando el pedido fue solventado con éxito y se imprimía el error y su descripción cuando este fallase.

4.2.1. Configuración del Servidor

Para interactuar con el servidor se crea una REST Api que facilite la comunicación y la generación de las respuestas a las peticiones del usuario desde el frontend. Se busca que cumpla las funciones básicas CRUD (Acrónimo en inglés para Crear, leer, actualizar y borrar registros) para administrar toda la información contenida y mostrada de la base de datos. Para facilitar el proceso de creación del servidor en Node.js se utiliza su API Express que suministra la infraestructura para construir una aplicación web y/o móviles de forma mínima y flexible, suministrando también los middlewares (que son una capa de traducción oculta que permite la comunicación y la administración de datos en las aplicaciones que se ejecutan en él [14].) necesarios para la comunicación entre el frontend y el backend. En el servidor se selecciona el puerto y el host donde se desplegará el backend, que podría ser tanto local como en la nube; se podrá escoger el motor de plantillas/templates (Texto procesado y convertido a HTML [15]) para renderizar en su ejecución. El principal aporte del servidor será la implementación de todos los métodos de petición HTTP, haciendo así una REST Api funcional que puede administrar la información de la base de datos, por medio de pedidos realizados por URLs.

4.2.2. Diagrama de árbol

Dentro de los servicios que presta el servidor a la interfaz de usuario, también se pueden renderizar *templates* de HTML. En el código del *backend* hay incluidos varios renders (pantallas), que muestran la información de la base de datos y permiten hacer modificaciones. También otro que contiene un diagrama de árbol que es invocado en el *frontend* por una etiqueta "iframe" de HTML que lo renderiza y lo lleva como un componente a una de las pantallas de visualización que tiene el aplicativo. Para su desarrollo se utilizó un motor de plantillas especificado en el servidor para su ejecución llamado "ejs" que permite tener código JavaScript dentro de un *template* HTML utilizando una sintaxis específica. Haciendo uso de la librería D3.js y de los ejemplos prácticos que brinda la documentación, se acomoda la información de la base de datos dentro de un objeto en JavaScript y en el *template*, dentro de una etiqueta script se ejecutan las funciones necesarias para desplegar un diagrama de árbol desplegable, donde cada nodo al darle *clic* se contrae y se dilata. Como se muestra en la figura 10.

4.3. Diseño del *Frontend*

Cuando se tuvo el *backend* completamente terminado y funcional, se comenzó con la realización de la interfaz para que el usuario pudiera hacer uso de las funciones definidas y servicios del aplicativo y así visualizar y administrar la información. Para el desarrollo se utilizó el *framework* Angular de JavaScript, empleando *typescript* que amplía la sintaxis de JS, agregando más funcionalidades y se enfoca en la programación orientada a objetos (POO). La elección del *framework* Angular para el desarrollo se hizo, porque facilita desarrollo de aplicaciones web *responsive* (que se adapta a cualquier resolución de pantalla) y también porque éste se especializa en la creación de aplicaciones web de una sola página, llevando así el desarrollo en este proyecto. Estructurado teniendo un componente principal como enrutador, que se encarga de todo lo que se está mostrando en el aplicativo y acorde a las peticiones URL que se hagan, controlar los componentes que se desplegarán en pantalla. La manera de desarrollar en Angular está basada en la creación de componentes, donde cada componente tiene su *template* HTML, su hoja de estilos CSS y un archivo de código para las funcionalidades en *Typescript*. Cada uno de estos componentes puede ser utilizado como una etiqueta html e insertarse en todos los *templates* realizados dentro de este aplicativo.

Para hacer las peticiones a las REST API (*backend*) desde el lado de la interfaz de usuario, se desarrolló un módulo encargado de tener las funciones para hacer las peticiones HTTP específicas a las URLs, utilizando el módulo de Angular "HttpClient" para realizar las peticiones. Este módulo de servicios se incluía en cada componente que necesitara administrar la información. Dentro de las funcionalidades del aplicativo, se agregó un acceso con usuario y contraseña para limitar el acceso a quien pudiera editar la información de la base de datos desde la interfaz de usuario.

4.4. Pruebas unitarias y funcionales

Una vez desarrollada la aplicación, se llevaron a cabo un conjunto de pruebas para validar su correcto funcionamiento. Para realizar las pruebas tanto funcionales como unitarias se siguió un formato descrito de la siguiente manera: Se realizó una tabla donde la primer columna de cada fila indica el número de prueba, la segunda columna detalla qué condición es la que se desea probar, la tercera describen los resultados que se esperan, la cuarta columna se escribe el dato de entrada al momento de realizar la prueba, la

quinta se expresa la salida obtenida al ingresar lo antes detallado y la última columna describe si la prueba fue o no exitosa. Luego de esto, se anexa tanto la evidencia como las explicaciones pertinentes de cada una de las pruebas realizadas y cómo se dio solución a las problemáticas identificadas.

5. Resultados y análisis

5.1. Base de datos

La estructura de la base de datos desarrollada fue puesta a prueba en el aplicativo funcional mostrando que su diseño fue exitoso. Se puede observar en la figura 5 qué campos tiene cada columna de las tablas de la base de datos y la manera como están relacionadas entre ellas, como fue descrito en la metodología.

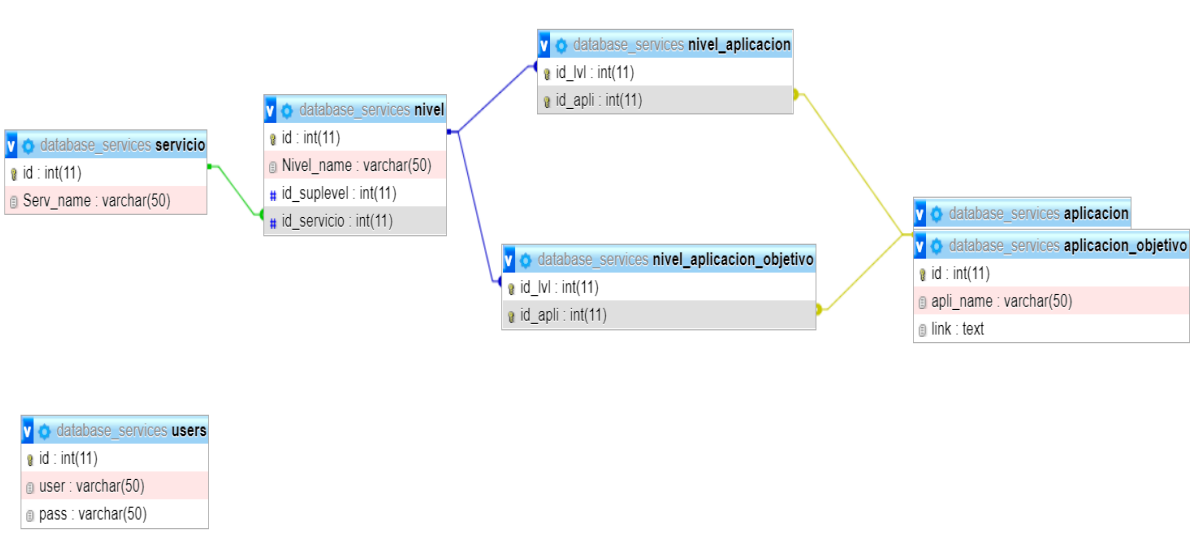

Figura 5. Estructura de la base de datos mostrada por el gestor de bases de datos de php my admin de apache.

5.2. Diseño del aplicativo

Para el desarrollo del aplicativo web responsiva se diseñaron 4 pantallas donde se visualiza información y las funcionales que tendría disponible el usuario: la pantalla principal lista los dominios en los que se dividen los servicios prestados por el banco, como ilustra la figura 6.

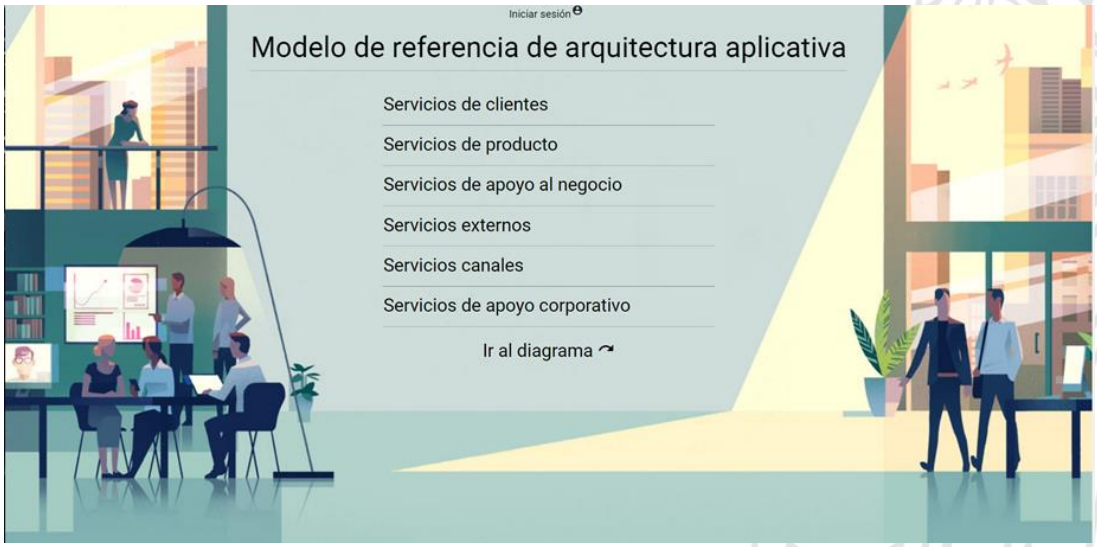

Figura 6. Pantalla principal del aplicativo.

Todos los ítems del listado en la pantalla se pueden seleccionar pasando el cursor sobre cada uno de los ítems del listado que se muestra. Al pasar el curso por los ítems que sea posible de seleccionar, producirá un aumento de su tamaño y al darle clic en cualquiera de ellos, se redirigirá a la segunda pantalla, mostrada en la Figura 8.

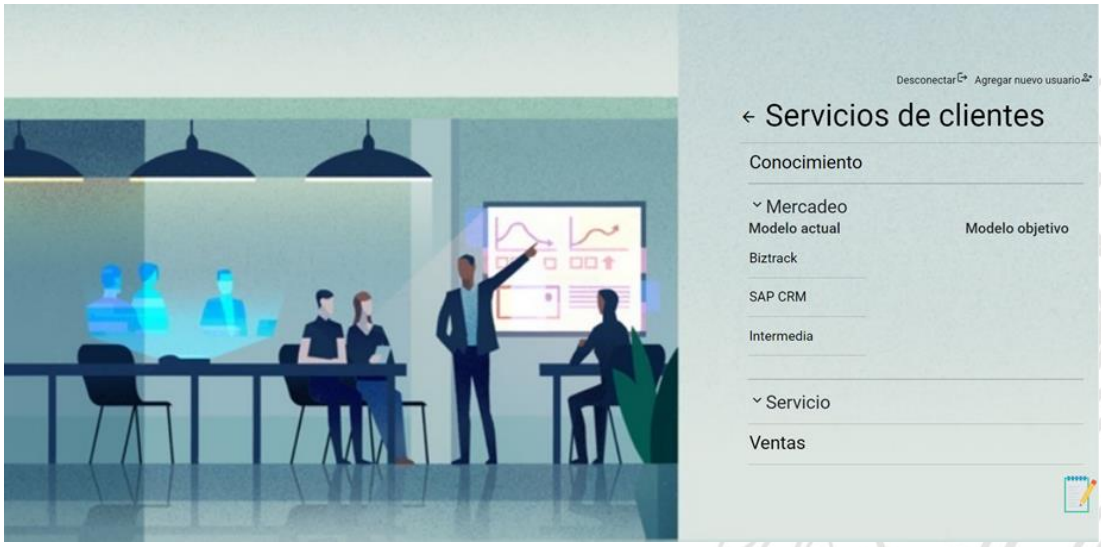

Figura 7. Segunda opción de la segunda pantalla, sin capacidades.

Si en la pantalla de subdominios se identifica que los ítems del listado están relacionados con aplicaciones que desarrollan esta capacidad, éstas serán mostradas al dar clic sobre el nombre del listado en una lista desplegable. Al seleccionar una de estas se accederá a la información pertinente del aplicativo (Referido en la figura 7 a "Biztrack", "SAP CRM" e "Intermedia") y la diferenciación vendrá dada por que al costado izquierdo del ítem del listado se visualizará una flecha desplegable.

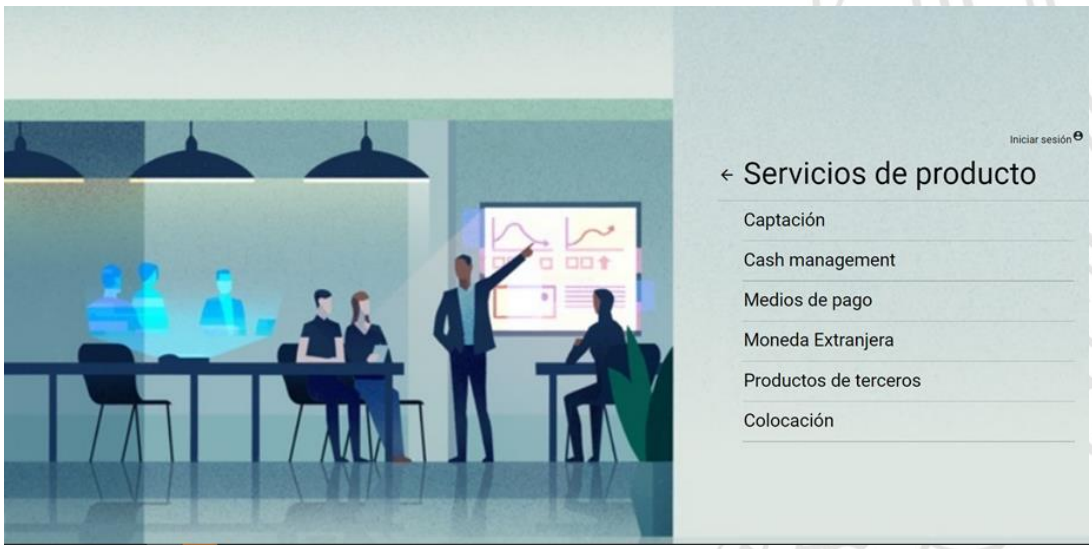

Figura 8. Segunda pantalla del aplicativo, subdominio.

En esta pantalla del aplicativo se muestra un listado similar a la primera pantalla como muestra la figura 8. Esta pantalla, dará acceso a la pantalla de capacidades al dar *clic* a los ítems del listado.

La tercera pantalla indica las capacidades se tiene la estructura de la segunda opción de la pantalla de subdominios (figura 9), donde cada uno de los ítems del listado tendrá una flecha a su costado izquierdo y desplegará las aplicaciones que desarrollan los productos y servicios prestados que allí se representan.

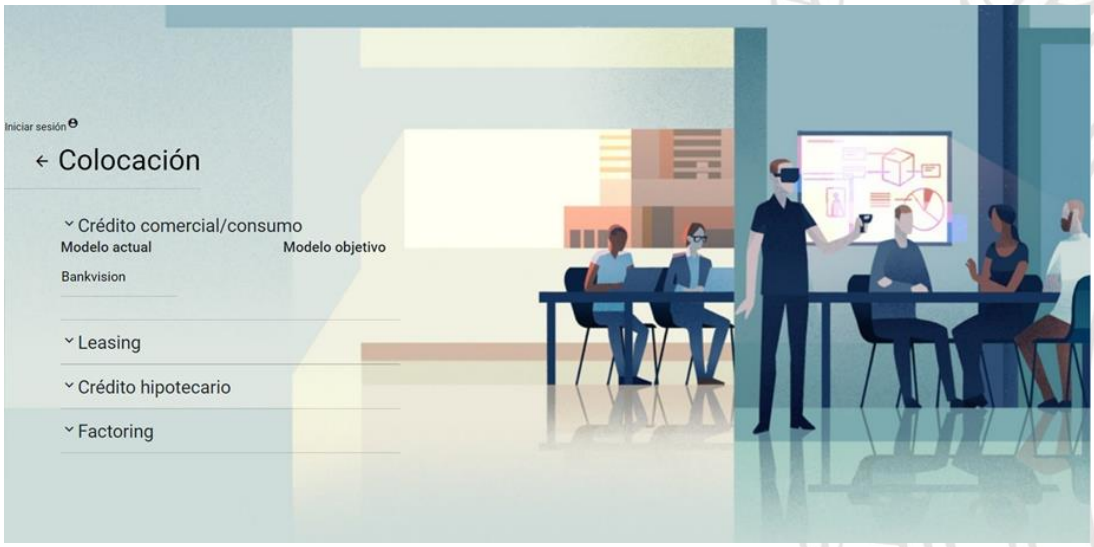

Figura 9. Tercera pantalla, capacidades.

De igual manera, como se muestra en la segunda opción de la pantalla de subdominios las aplicaciones serán mostradas al dar clic sobre la flecha al lado izquierdo del nombre en el listado y si se selecciona una de estas, sea del modelo objetivo o del modelo actual se accederá a la información pertinente del aplicativo.

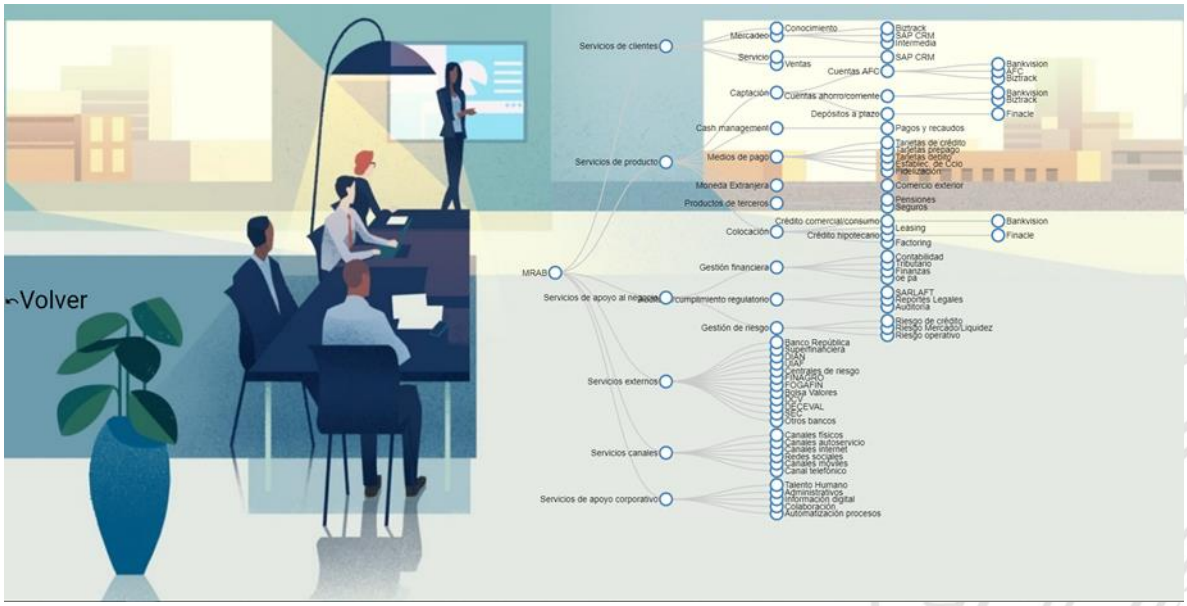

Figura 10. Diagrama de árbol del modelo de arquitectura aplicativa actual.

El acceso al esquema mostrado en la figura 10, se produce desde la primera pantalla como se indicó al dar un clic en "ir al diagrama", mostrando un diagrama de árbol desplegable (donde cada uno de los nodos mostrados se puede contraer y expandir para un mejor entendimiento del esquema y poder observar mejor la jerarquía de la capa aplicativa de la organización).

Como se mencionó anteriormente, para editar la información mostrada en los listados y en el esquema se debe iniciar una sesión, el cual se muestra en la figura 11.

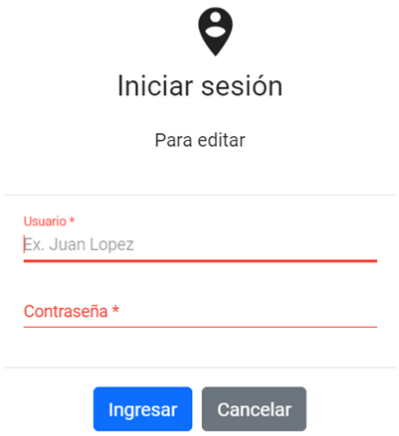

Figura 11. Pestaña de inicio de sesión.

Se debe disponer de un usuario y contraseña válidos para acceder a la opción de editar la información que se muestra en los listados. Al tener las credenciales correctas, se mostrará un ícono en la parte inferior derecha y donde se veía la opción de iniciar sesión, se aparecerá la opción de desconectarse y de agregar un nuevo usuario como lo muestra la figura 12.

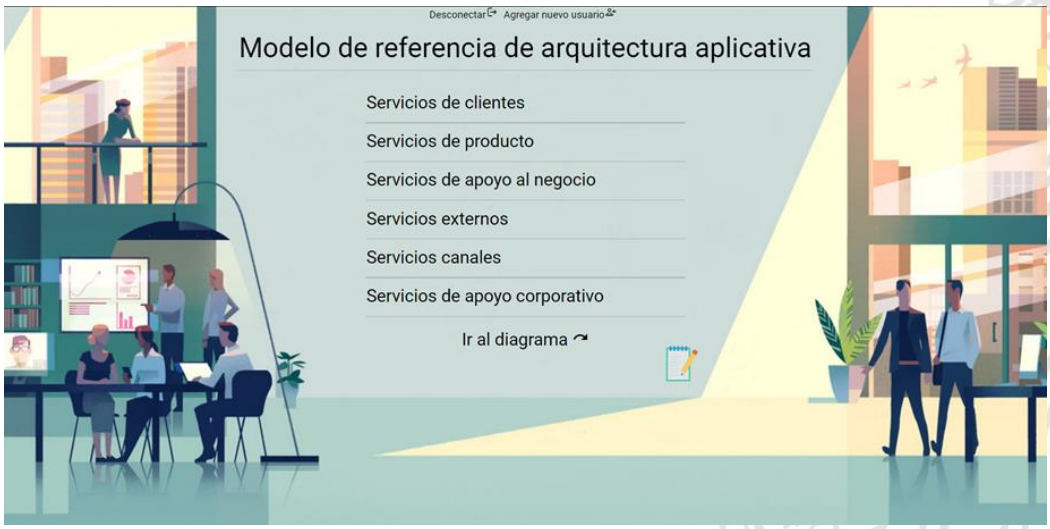

Figura 12. Pantalla inicial con el ícono de inicio de sesión.

Cada una de las pantallas principales explicadas anteriormente, tienen una opción que permite editar la información que contienen, para la pantalla inicial se ilustra a continuación en la figura 13.

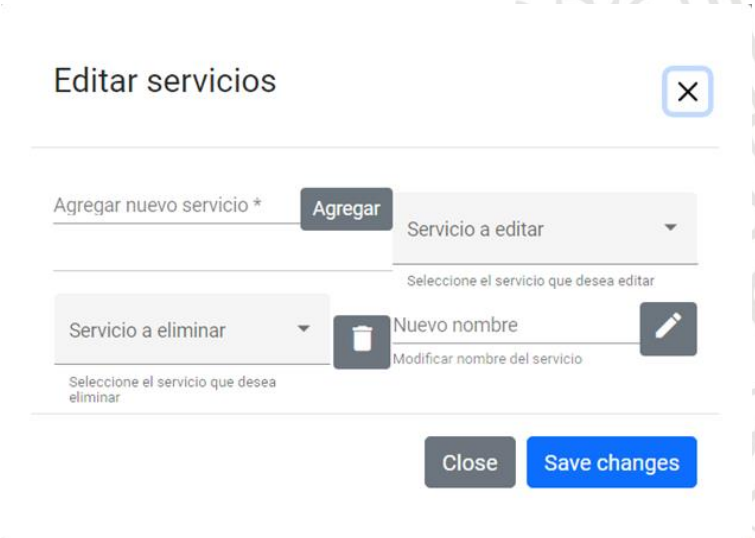

Figura 13. Pestaña de edición para la pantalla inicial.

Como se observa en la Figura 13 las opciones de la pantalla de edición en la pantalla principal son: Agregar un nuevo Servicio (Dominio), seguido a esto se tiene una lista desplegable donde se muestran todas las opciones ya creadas seleccionables para ser eliminada, luego de darle botón justamente a su lado con ícono de basura, y por último al lado derecho se tendrá el mismo listado con todas las opciones existentes y debajo de este en el campo de entrada, donde debe ser ingresado un nuevo nombre para el dominio que se cambiará por el de la opción seleccionada.

Siguiendo una lógica similar en el resto de las pantallas de edición, mediante la información pedida realizan las solicitudes HTTP y administran la información de la base de datos.

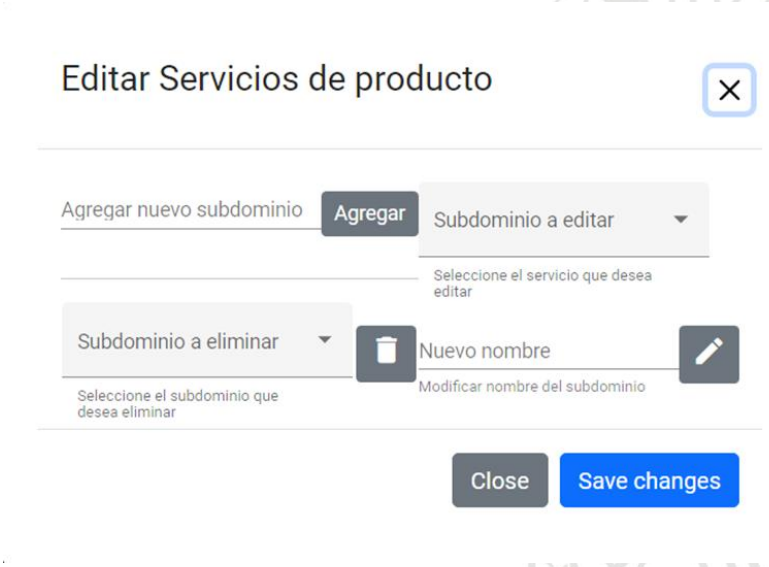

Figura 14. Dialogo de edición de la segunda pantalla.

Las opciones de edición de la información de la segunda pantalla (Subdominios), vienen descritos de la misma manera que para la pantalla principal, contiene las mimas opciones anteriormente descritas.

Se debe tener en cuenta que en la segunda y la tercera pantalla se pueden tener 2 opciones de edición, ya sea de los subdominios/capacidades o de los aplicativos, por lo tanto, se tendrá un diálogo que nos pregunte qué deseamos editar, como se muestra en la figura 15.

Figura 15. Dialogo de elección.

Acorde a lo que se elija, se redirigirá ya sea a la opción descrita por la figura 14/16 dependiendo de la pantalla donde se quiera editar o a la figura 12.

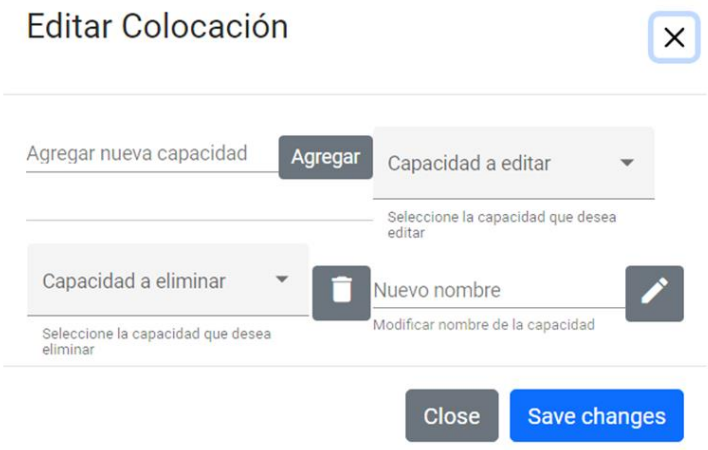

Figura 16. Dialogo de edición de la tercera pantalla.

Las opciones de edición de la información de la tercera pantalla (Subdominios), vienen descritos de la misma manera que para la pantalla principal y la segunda, contiene las mimas opciones allí descritas.

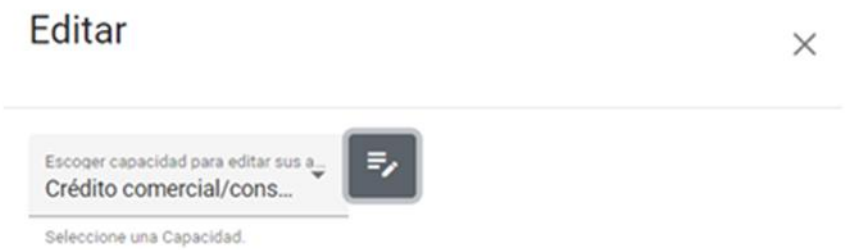

Figura 17. Dialogo de primera entrada a la edición de aplicaciones.

Al ingresar inicialmente a la pantalla de edición de aplicaciones se muestra una opción con una lista desplegable que mostrará todas las capacidades en cuestión y se deberá escoger cuál se va editar. Al escogerla, se presiona el botón a su costado derecho y se podrá acceder a la edición de las aplicaciones, como se muestra en la figura 18.

Figura 18. Dialogo de edición de aplicaciones.

Las opciones de edición de las aplicaciones que se pueden ver en la Figura 18 funcionan de la siguiente manera: primero se mostrará el diálogo solo con el listado y el botón descrito anteriormente, luego como se puede observar, las opciones son iguales, tanto para las aplicaciones actuales como para las objetivo. Por consiguiente, siguiendo este orden lo que se puede realizar es, como primero se puede escoger una aplicación que ya esté registrada y adicionarla a la capacidad en cuestión, también se podrá eliminar la relación de esta capacidad con el aplicativo en cuestión (Para estos dos casos se sigue la lógica de la lista desplegable que lo muestra y un botón a su costado derecho para confirmar). Enseguida se puede agregar una nueva aplicación, tanto su nombre como su link informativo y quedará vinculada automáticamente con la capacidad en cuestión. Como última función en esta vista, se tendrá un listado desplegable con todos los aplicativos y si se selecciona uno y se presiona el botón de su costado derecho, se eliminará el aplicativo y se quitará su relación en todas las capacidades. Como se puede observar en la parte inferior derecha de la vista, se dispone de un botón para "mover varias aplicaciones", si seleccionamos éste nos llevará al dialogo de la figura 19.

Figura 19. Dialogo de mover varias aplicaciones.

Lo primero que se debe hacer para mover varias aplicaciones es ir seleccionando una por una las aplicaciones del listado e irle dando al botón con el símbolo "+", cuando se tengan todas las deseadas seleccionadas, si se le da clic al botón con el símbolo del ojo, se podrá ver cuales aplicaciones serán movidas, y por último, se debe seleccionar a qué capacidad serán movidas y darle al botón del "+" a su lado derecho para mover de forma masiva los aplicativos.

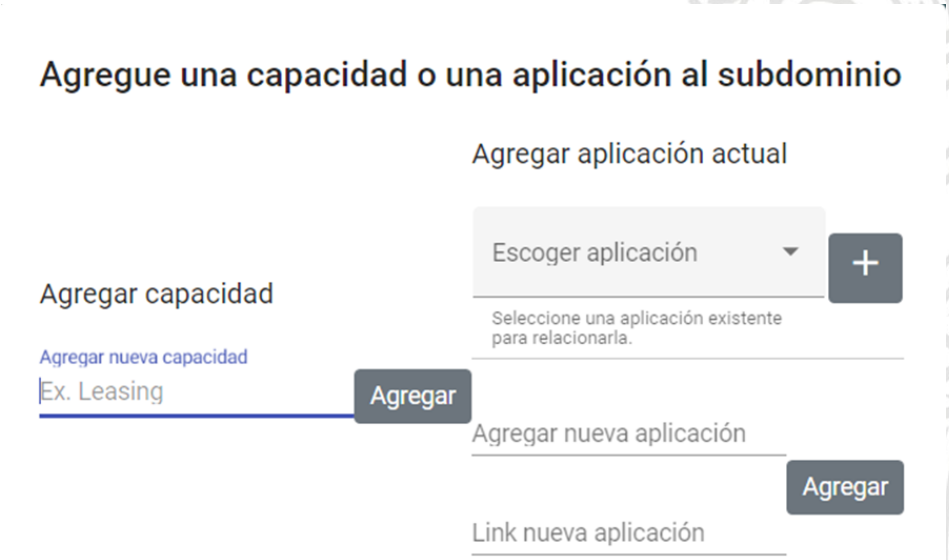

Figura 20. Dialogo mostrado al ingresar a un subdominio recién creado.

Por último, como se muestra la Figura 20, al acceder a un subdominio recién creado o que no tenga jerarquía por debajo (Darle clic a una de las opciones de la segunda pantalla) se mostrará un diálogo que pedirá, ya sea agregar una capacidad para este subdominio agregando su nombre o agregarle una relación con una aplicación actual existente o asignarle un aplicativo nuevo junto con su enlace de información.

5.3. Pruebas funcionales y unitarias

De acuerdo con la estructura establecida explicada previamente para la realización de las pruebas, se obtuvieron los resultados mostrados en tabla siguiente, lo cual permitió verificar el cumplimento de los requisitos específicos que fueron establecidos.

Consecutivo de la prueba	Condición por probar	Resultado Esperado	Dato Entrada	Dato Salida	¿Prueba exitosa? (S/N)
1	En la creación de la base de datos, se definió que los campos de los nombres de los Dominios, subdominios, capacidades y aplicaciones tuvieran un máximo de 50 caracteres.	En el campo de entrada de los datos, la cantidad de caracteres escritos no puede superar el número 50.	Se agregó 49 veces la letra a al campo y luego una letra b.	En las entradas, al haber llegado al tope de los 50 caracteres, como estaba limitado no permitía ingresar más.	S

2	En el campo donde se pide que se ingrese un link para que al dar clic sobre una aplicación ésta nos lleve a la información de la misma, debe de ingresarse una URL obligatoriamente.	En el campo de entrada del dato "Link" no se recibe ninguna entrada que no sea una URL de una página web	Se ingresó en el campo un texto normal "Prueba"	No se permitió continuar con el proceso debido a que la entrada no era una URL.	S
3	En los campos de los nombres de Dominios, subdominios, capacidades y aplicaciones no se permite que se ingresen números, ni caracteres especiales como entradas.	Al ingresar una secuencia de números no nos debe permitir continuar con el proceso sea de agregado o de edición.	Se ingresó una secuencia de números "12312" a la entrada	No se permitió continuar con el proceso debido a que la entrada contenía caracteres numéricos	S

4	Al hacer pedidos en la base de datos desde el lado del usuario, el servidor manda un mensaje que se muestra por consola indicando que la acción requerida se realizó satisfactoriamente en la base de datos	Al hacer un query a la base de datos, se debe ver reflejado en la consola un mensaje de que se realizó correctamente la acción requerida.	Se hizo un pedido a la base de datos de eliminar un dominio existente.	Se obtuvo por parte del servidor una respuesta	S
5	Al momento de agregar un subdominio nuevo si se accede a este se debe de escoger entre agregar una capacidad o una aplicación que éste utilice.	Cuando se accede al nuevo subdominio se debe abrir un cuadro que te pida agregar o una capacidad o una aplicación ya sea existente o una nueva.	Se Accedió a un subdominio recién creado.	Se visualizó un cuadro que pedía escoger entre agregar o una capacidad o una aplicación ya sea existente o una nueva.	S
6	Se alerta al usuario antes de eliminar cualquier dominio, subdominio o capacidad.	Tener una alerta que salte cada que se intenta eliminar algún valor.	Se trata de eliminar un dominio vacío.	Salta una alerta que pregunta si se está seguro o no de eliminar el dominio.	S

7	Cuando un dominio tiene subdominios por debajo en jerarquía, no se debe poder eliminar el servicio.	Debe salir una alerta indicando que no se puede eliminar el dominio	Se trata de eliminar un dominio con jerarquía.	Salta una alerta de que no se puede eliminar.	S

Tabla1. Recuento detallado de las pruebas unitarias y funcionales realizadas.

5.4. Análisis.

Efectivamente se realizó la aplicación planteada como solución a la problemática definida. La aplicación despliega toda la información de forma organizada y centralizada, tanto en forma de un listado con accesos como en forma de un esquema. La aplicación es responsiva lo que implica que se adaptará a cualquier resolución. La estructura ideada para la base de datos cumplió a cabalidad con la necesidad establecida, haciendo un uso óptimo en memoria evitando tener registros innecesarios a la hora de relacionar las diferentes tablas de información de la base de datos. Finalmente, todas las pruebas realizadas en el aplicativo desarrollado, resultaron exitosas, como se detalla en la figura 14. También se desarrollaron un informe y los manuales para su manejo e instalación.

6. Conclusiones

Se desarrollo una aplicación que facilita el acceso a la información relacionada a la capa de aplicaciones de la empresa. En esta aplicación los usuarios tienen una manera más fácil de orientarse referente a los servicios. La aplicación permite visualizar la jerarquía a manera de un esquema o de listado seleccionable, lo cual facilita acceder a la información de cada aplicativo, brindando una perspectiva más grande, permitiendo ampliar un el entendimiento sobre como Bancolombia brinda los servicios y productos a sus usuarios.

La administración de la base de datos es eficiente porque al ser relacional evita que se creen muchos registros de más, permite crear relaciones entre las tablas sin tener que hacer una gran cantidad de registros y al relacionar la tabla de "Nivel" con ella misma se puede tener de una manera más compacta la información, pero sin perder la lógica y el orden de la jerarquía.

El uso de Angular y Node.js resultó conveniente para el desarrollo del aplicativo web responsivo, ya que con Node se facilita mucho la creación de un servidor que preste servicios para realizar todos métodos de petición HTTP correspondientes para la administración de la información de las bases de datos. Por otro lado, Angular posee la ventaja de que solo por ser programado en este framework, el aplicativo del frontend ya es responsivo y presta muchas facilidades a la hora de darle orden y funcionalidad a los objetos(etiquetas) dentro de los template(pantallas) a los que el usuario accede.

El aporte del aplicativo a los servicios/productos de Bancolombia tiene un gran impacto porque los empleados y colaboradores que necesitan acceder a toda la información de los servicios que se prestan y el mecanismo de los aplicativos que corresponden estos servicios, con el fin de llegar a las soluciones de una manera fácil, concreta, dinámica, y sencilla de actualizar. Para así no perder el foco de transformación ni el hilo conductor del desarrollo de las soluciones brindadas. Con base en las pruebas realizadas con la aplicación, se concluye que se lograron los objetivos planteados en la propuesta inicial.

Referencias Bibliográficas

[1] Villalobos, J (2020, 1 Febrero). El mapa de capacidades del negocio. https://www.researchgate.net/publication/331608341_El_Mapa_de_Capacidades_de_Negocio_-_Lecturas_para_Arquitectos_de_Negocio.

[2] EcuRed. (s. f.). Aplicación web - EcuRed. https://www.ecured.cu/Aplicaci%C3%B3n_web#Estructura_de_las_aplicaciones_web.

[3] Platzi: Cursos online profesionales de tecnología. (s. f.). Platzi. https://platzi.com/blog/que-es-frontend-y-backend/.

[4] Stefaniak, P. (2019, 10 septiembre). ¿Qué es Backend y Frontend? Descubre Comunicación. https://descubrecomunicacion.com/que-es-backend-y-frontend/#:%7E:text=El%20back%20end%20del%20sitio,para%20que%20todo%20funcione%20correctamente.

[5] JavaScript | MDN. (s. f.). JavaScript. https://developer.mozilla.org/es/docs/Web/JavaScript.

[6] Node.js – Entorno de ejecución para JavaScript. (s. f.). https://nodejs.org/

[7] Express - Infraestructura de aplicaciones web Node.js. (s. f.). Express. https://expressjs.com/es/

[8] fw_error_www. (s. f.-b). oracle. https://www.oracle.com/co/database/what-is-a-relational-database/

[9] Developer.mozilla.org. 2021. Métodos de petición HTTP - HTTP | MDN. [online] Available at: https://developer.mozilla.org/es/docs/Web/HTTP/Methods.

[10] Red Hat. 2021. ¿Qué es una API de REST?. [online] Available at: https://www.redhat.com/es/topics/api/what-is-a-rest-api.

[11] Qué es HTML. (s. f.). CódigoFacilito. https://codigofacilito.com/articulos/que-es-html.

[12] CSS | MDN. (s. f.). CSS Tutoriales. https://developer.mozilla.org/es/docs/Web/CSS.

[13] Angular. (s. f.). Angular. https://angular.io.

[14] Azure.microsoft.com. 2021. Qué es middleware: definición y ejemplos | Microsoft Azure. [online] Available at: https://azure.microsoft.com/es-es/overview/what-is-middleware/.

[15] Apuntes.de. 2021. Motores de templates. [online] Available at: https://apuntes.de/nodejs/templates/#gsc.tab=0.

[16] Souza, I., 2021. API Rest: ¿qué es y cómo funciona ese recurso?. [online] Rock Content - ES. Available at: https://rockcontent.com/es/blog/api-rest/.