UNIVERSIDAD DE ANTIOQUIA

DESARROLLO DE COMPONENTES REUTILIZABLES PARA LA CONSTRUCCIÓN DE APLICATIVOS MÓVILES BANCOLOMBIA

Ricardo Tangarife González

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería Electrónica y Telecomunicaciones

Medellín, Colombia

2020

DESARROLLO DE COMPONENTES REUTILIZABLES PARA LA CONSTRUCCIÓN DE APLICATIVOS MÓVILES BANCOLOMBIA

Ricardo Tangarife González

Informe de práctica como requisito para optar al título de:
Ingeniero Electrónico.

Asesores

Ricardo Andrés Velásquez Vélez – Ingeniero Electrónico
Paulo Andrés Arias Esguerra – Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería Electrónica y Telecomunicaciones.
Medellín, Colombia
2020.

Resumen

Para Bancolombia es relevante optimizar el tiempo de entrega de los desarrollos de soluciones software para los canales digitales que ofrece. Dentro de estos canales se encuentran los aplicativos móviles, así que para reducir estos tiempos se implementó un API basada en el sistema de diseño Bancolombia, en la cual se contienen todos los componentes móviles transversales a los canales.

Durante esta práctica se desarrollaron 5 de los componentes móviles transversales que componen esta API. Cada componente se documentó para que el desarrollador a utilizarlo pueda identificar cómo hacerlo, cuáles son sus parámetros, características y alcances. Así mismo, los componentes cuentan con pruebas unitarias que corroboran su funcionamiento. También se desarrolló una aplicación demostrativa de los componentes disponibles dentro del API, allí se muestran diferentes escenarios de cada componente.

La disponibilidad de componentes dentro del API, permitió mejorar los tiempos de desarrollo de aplicativos móviles al interior del banco, ahora los desarrolladores solo deberán armar un rompecabezas tomando las piezas disponibles en el API del sistema de diseño, poniéndolas a trabajar en conjunto para completar la aplicación requerida.

Introducción

El Grupo Bancolombia es un ente bancario con presencia nacional e internacional; en el cual el desarrollo tecnológico ha sido un pilar fundamental que ha acompañado la transformación e innovación en el mercado durante sus 145 años de trayectoria, ofreciendo sostenibilidad, seguridad y disponibilidad de cada uno de sus servicios.

En la búsqueda de mejora continua, el banco ha implementado desarrollos y practicas tecnológicas tanto propias como de terceros, método que ha funcionado de manera eficaz dando solución a los requerimientos presentados hasta el momento.

En miras de optimizar los desarrollos en los canales digitales ofrecidos por el banco para sus clientes, dentro de los cuales se encuentran los aplicativos móviles para las diferentes plataformas, Android e iOS. Se encuentra una falencia en las metodologías de desarrollo utilizadas, y es que la estructuración tecnológica no tiene una arquitectura centralizada, lo que hace que cada desarrollo se tome como un ente aislado de los demás, ralentizando los desarrollos y no permitiendo dar una mantenibilidad suficiente.

Ahora, a través del desarrollo de un API centralizada y tokenizada siguiendo los lineamientos percibidos desde el banco y enfocada en los componentes móviles que

se requieren para los diferentes aplicativos, se podrá tener una biblioteca de componentes reutilizables y documentados.

Esta API será el punto de partida para la configuración front-end de las aplicaciones móviles y proyectos que se presenten en adelante, ahorrando tiempo de desarrollo, y aumentando la robustez en los productos, pues ahora se podrá dar mantenibilidad constante al ser un insumo propio del banco.

Esta práctica consistió en desarrollar cinco componentes como aporte a esta API centralizada, componentes transversales y reutilizables para las diferentes aplicaciones móviles del banco y proyectos móviles posteriores.

Objetivo General

Desarrollar cinco componentes móviles front-end reutilizables de los 250 componentes con los que cuenta el API del Sistema de Diseño Mobile Bancolombia, utilizando el framework Flutter e implementando las estrategias de atomic y material design. Los cinco componentes que se desarrollarán están asociados con objetos de la interfaz gráfica.

Objetivos Específicos

Analizar los requerimientos establecidos para los componentes a desarrollar en el API, según los lineamientos adoptados por las necesidades del banco.

Desarrollar las clases reutilizables dentro del API, asociadas a los componentes móviles buscando la mayor versatilidad e implementado las metodologías de material y atomic design.

Evaluar cada uno de los componentes desarrollados mediante la aplicación de pruebas unitarias e implementando un caso de uso, con el fin de verificar el correcto funcionamiento.

Marco Teórico

Diseño de Aplicativos Móviles Modulares

Existen muchas aplicaciones para dispositivos móviles que comparten similitudes, y la implementación de nuevas aplicaciones puede resultar en código duplicado dentro del mismo proyecto o en proyectos similares. Además, el desarrollo móvil actual sigue siendo una tarea propensa a errores que puede producir aplicaciones difíciles de mantener. Como resultado, los desarrolladores necesitan nuevas herramientas que les ayuden a desarrollar software móvil de manera eficiente.[1]

Una aplicación modular, mantiene una funcionalidad aislada de otra, pero a la vez permite integrarse fácilmente. En estas se pueden elegir cuáles son las funcionalidades requeridas a implementar simplificando el desarrollo y agilizando la implementación. Así mismo, la modularidad agrega flexibilidad a los posibles cambios que puedan surgir. Por tanto, el desarrollo modular de aplicativos móviles consta de módulos configurables con funcionalidades preestablecidas que se pueden reutilizar como base para generar una aplicación móvil en días y/o semanas; en lugar de meses como ocurre en un clásico proyecto de desarrollo móvil.

Con esta metodología se tiene a disposición módulos con funcionalidades listas para integrar, que permiten reaccionar rápidamente a los cambios de mercado y/o diseño, reducir costos de desarrollo, optimizar recursos de pruebas y sumar flexibilidad. [2] [3]

Sistema de Diseño

Un sistema de diseño se establece como el conjunto de patrones y practicas que se comparten en este caso en el Banco de forma coherente y organizada. Buscando la centralización de los desarrollos de los canales digitales móviles se planteó el Sistema de Diseño Mobile Bancolombia, con el fin de obtener una mayor flexibilidad y mantenibilidad a largo plazo.[4] Este sistema está basado en la metodología atomic design.

Atomic design

Esta metodología está compuesta por cinco etapas distintas que trabajan juntas para crear sistemas de diseño de interfaces de una manera más deliberada y jerárquica. Las cinco etapas del diseño atómico son: Átomos, Moléculas, Organismos, Plantillas y Páginas.

En este sentido los átomos serán aquellos componentes fundamentales que comprenden todas las interfaces, cada átomo tiene sus propias propiedades definidas tales como tamaños, fuentes y tipografías, como átomos se tienen elementos tipo botones, switch, checkbox, iconos, entre otros. En química las moléculas son grupos de átomos unidos entre sí que adquieren distintas propiedades nuevas, siguiendo esta lógica, aquí las moléculas son grupos relativamente simples de elementos UI (Interfaz de Usuario) que funcionan juntos como una unidad, ejemplos de estos serán: buscadores, alertas, acordeones, notificaciones, entre otros. Los organismos son componentes de UI relativamente complejos compuestos por grupos de moléculas y/o átomos y/u otros organismos, en aplicativos móviles estos comprenderán elementos como navegaciones y encabezados o barra de aplicación.

Ahora, las plantillas son objetos a nivel de pantalla que hacen uso de componentes en un diseño y articulan la estructura del diseño. Por último, las páginas corresponden a plantillas instanciadas según el contenido e interacciones necesarias para cada pantalla.[5]

Material design

Es el sistema de diseño de Google, el cual es un sistema adaptable de pautas, componentes y herramientas que respaldan las mejores prácticas de diseño de interfaces de usuario. Respaldado por código de fuente abierta, Material agiliza la colaboración entre diseñadores y desarrolladores. En este se enmarcan las paletas de colores, el diseño de las cuadriculas y las tipografías. Estos lineamientos son los preestablecidos dentro del framework Flutter. [6]

Flutter

Es un framework de desarrollo para aplicaciones móviles, cuenta con un conjunto de herramientas de interfaz de usuario multiplataforma de código abierto creado por Google en 2017; está diseñado para permitir la reutilización de código en sistemas operativos como iOS y Android, al mismo tiempo que permite que las aplicaciones interactúen directamente con los servicios nativos de cada una de las plataformas mencionadas. El objetivo es permitir que los desarrolladores entreguen aplicaciones de alto rendimiento que se sientan naturales en diferentes plataformas, esto porque las herramientas anteriormente prestadas para el desarrollo transversal de aplicativos móviles se realizaba de forma interpretada a cada sistema operativo y no de forma nativa, teniendo esta ventaja Flutter sobre tecnologías como React Native o Xamarin.[7]

Flutter utiliza Dart como lenguaje de programación, también desarrollado por Google, diseñado con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y hacer el proceso de desarrollo lo más cómodo y rápido posible para los desarrolladores. Por eso, viene con un conjunto bastante extenso de herramientas integrado, como su propio gestor de paquetes, varios compiladores, un analizador y formateador. Además, la máquina virtual de Dart y la compilación Just-in-Time hacen que los cambios realizados en el código se puedan ejecutar inmediatamente.[8]

Los widgets en Flutter se definen como una unidad de composición, siendo los componentes básicos de la interfaz de usuario de una aplicación Flutter, y cada widget es una declaración inmutable por parte de la interfaz de usuario. A través de estos se forma una jerarquía en la que cada widget se anida dentro de su padre y puede recibir contexto del mismo. Esta estructura llega hasta el widget raíz que es el contenedor de la aplicación. Por lo anterior toda clase en Flutter es considerada en

sí un Widget que corresponde a un componente móvil. De allí que todos los componentes que se creen dentro de nuestro sistema de diseño serán y estarán compuestos por widgets. Así pues, el framework contiene widgets base tanto para ajuste de propiedades: relleno, alineación, filas, columnas, cuadrículas; como para representaciones visuales: Container, Text, Image, entre otros. Un ejemplo trivial de lo anterior es el observado en la Figura 1. En éste todas las clases instanciadas corresponden a Widgets y se encuentran anidados cada uno dentro de su padre hasta llegar al widget raíz en este caso y comúnmente utilizado el MaterialApp de la librería Material propia de Google.[7].

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('My Home Page')),
        body: Center(
          child: Builder(
            builder: (BuildContext context) {
              return Column(
                children: [
                  Text('Hello World'),
                  SizedBox(height: 20),
                  RaisedButton(
                    onPressed: () {
                      print('Click!');
                    },
                    child: Text('A button'),
                  ),
                ],
              );
            },
          ),
        ),
      ),
    );
  }
}
```

Figura 1. Ejemplo implementación widget Flutter.

TDD (Test Driven Development o Desarrollo Dirigido por Pruebas)

Es un proceso de desarrollo que consiste en codificar pruebas, desarrollar y refactorizar de forma continua el código construido. La idea principal de esta metodología es realizar de forma inicial las pruebas unitarias para el código que tenemos que implementar. Es decir, primero codificamos la prueba y, posteriormente, se desarrolla la lógica. Así consistirá en desarrollar a miras de cumplir con los test, basados en los criterios de aceptación de cada componente.[9]

Metodología

Como procedimiento general para los componentes desarrollados dentro del API inicialmente se tomó el diseño entregado en el sistema de diseño Bancolombia por parte de UX para cada uno de los componentes, para con este insumo analizar las características y complejidad. Posteriormente se evaluaron cuáles eran las

propiedades transversales para los componentes de interfaz front-end, esto apoyado de los lineamientos dados por el banco respecto a las necesidades previstas desde el negocio.

Una vez claras todas las propiedades y características de los componentes se procedió a su codificación en lenguaje Dart y a través del framework de desarrollo flutter haciendo uso de la librería material con los elementos básicos para construcción de widgets.

Finalmente codificados los componentes se realizaron las correspondientes pruebas unitarias en las cuáles se verificó el correcto renderizado de cada componente en sus diferentes escenarios, estas pruebas se realizaron mediante el mismo framework pues éste ofrece la bondad de realizar los test del lenguaje trabajado.

Como paso último se codificó una implementación de casos de uso de los componentes mostrando diferentes combinaciones de los parámetros transversales, a través de una aplicación beta interna denominada showcase donde se muestran los diferentes componentes disponibles dentro del API del sistema de diseño Bancolombia.

Componente 1 – Colors Dark

El componente de colores corresponde a la paleta de colores utilizada para todo el sistema de diseño Bancolombia en el diseño de los componentes de la API, por tanto corresponden a tokens dentro del atomic design; así mismo por esto se define la API como tokenizada, ya que se definen este tipo de lineamientos como contantes intrínsecas del diseño, conteniendo en este caso todos los posibles colores que se pueden utilizar según determinaciones del banco para utilizar dentro de sus aplicativos móviles.

En este caso la paleta de colores dark corresponde a la homologación de los colores para los componentes cuando la aplicación se encuentra en modo Dark, que es utilizado para oscurecer la luminosidad de la aplicación. Algunos de los colores de la paleta para este componente de tokens definida desde UX fueron los contemplados en la Figura 2.

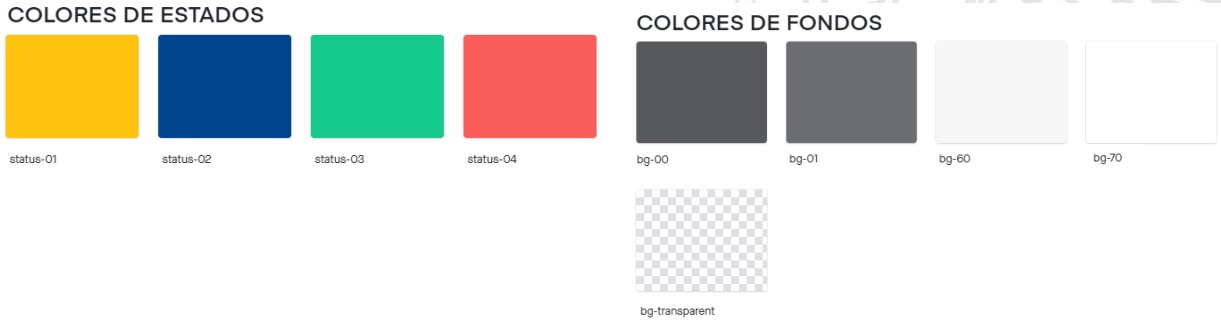

Figura 2. Definición de diseño componente colors Dark.

Al llevar esto a código, se implementó a través de una clase que representa el componente de tokens de colores dark, en ésta se instanciaron como constantes los nombres de referencia para cada uno de los colores de la paleta asignados a su correspondiente valor en el sistema numérico hexadecimal RGB homologados para el tema dark.

Con este componente se pueden acceder a cada uno de estos tokens de color dark para ser asignados ya sea en otros componentes o por el desarrollador mismo al momento de elaborar la aplicación.

Para realizar la prueba este componente se utilizaron estos tokens de colores en diferentes componentes del API corroborando la accesibilidad a cada token del componente y que se renderizará el color especifico instanciado.

Componente 2 – Tags

Los tags son componentes que se usan para los elementos que necesitan ser categorizados u organizados usando palabras clave que los describen, dentro del sistema de diseño Bancolombia se establecieron cinco tipos de tags; el diseño para estos recibido por UX fue el observado en la Figura 3.

Figura 3. Definición de diseño componente Tags.

Por su complejidad simple los tags se categorizan dentro del atomic design como átomos, pues únicamente están compuestos por tokens, como se puede observar en el diseño, un token de color de fondo y un token de texto corto.

Para este caso se analizó cómo debía manejarse los diferentes tipos de tags dentro del componente, así que como parámetros de entrada se identificaron el tipo de tag; que se manejó a través de un enumerable, y como segundo parámetro el texto que debe ir dentro del tag.

En la elaboración de las pruebas para este componente se verifica el correcto renderizado de los diferentes tipos de tags y evalúa la correcta asignación del texto dentro del mismo.

Componente 3 – Etiquetas

Este componente de acuerdo a su complejidad se encontró que dentro del atomic design corresponde a una molecula, pues se compone tanto de un texto como de un icono, que son átomos.

Se evaluaron cuáles eran las características transversales para este componente de interfaz front-end, encontrando que era necesario que tuviera un icono accionable, así mismo que se pudiera elegir si la etiqueta se encontraba activa o inactiva, que se le pudiera configurar el texto que contendría y que tuviera dos iconos de izquierda y derecha opcionales.

El diseño de UX para este componente era el siguiente constatado en la Figura 4.

Figura 4. Definición de diseño componente Etiquetas.

Se realizaron las correspondientes pruebas unitarias en las cuáles para este componente se verificó el correcto renderizado en los escenarios activo o inactivo, con icono a la izquierda o derecha, además de probar que detectara el icono accionable izquierdo.

Componente 4 – Layout

Para este componente con el diseño tomado desde UX se encontró que la definición estaba un poco abierta y difería de cómo funcionaban las estructuras de pantallas en dispositivos móviles, pues se tenía pensado que el layout se podría trabajar como un sistema de grillas como se hace en las tecnologías web, el diseño era el observado en la Figura 5.

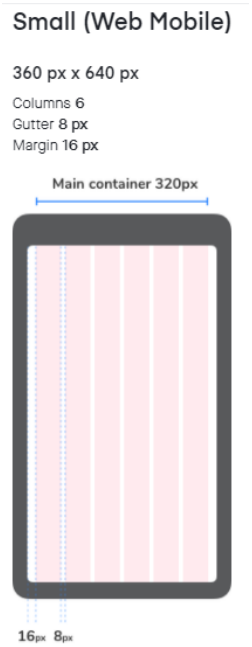

Figura 5. Definición de diseño componente Layout.

Por tanto, para llevar esta idea de correctos espaciados en el mundo móvil y que fuera responsivo para todos los dispositivos se ideo el componente como una clase la cual permite obtener tamaños responsivos para valores estáticos basados en los tamaños del diseño, que se adapten a todas las pantallas.

Así este componente layout es un componente que ofrece métodos públicos para obtener tamaños responsivos respecto al dispositivo partiendo de tamaños estáticos del diseño pasados como parámetros.

Se definieron para este componente que debía contar con métodos tanto para espaciados de alto como también de ancho, así mismo el componente a partir del contexto del dispositivo debía identificar si el dispositivo se encontraba en posición vertical u horizontal para adaptarse también en esos casos.

Para realizar las pruebas a este componente, se utilizaron diferentes dispositivos de tamaños cambiantes en su alto y ancho, así como también intercambiando la posición vertical y horizontal, así al utilizarse el componente layout responsivo los elementos debían adaptarse de acuerdo al tamaño de la pantalla del dispositivo y su orientación.

Componente 5 – Header navigation bar

El header navigation bar corresponde a la barra de aplicación superior ubicada en las pantallas, con elementos accionables e información acerca de la navegación dentro de cada pantalla de la aplicación.

Por las anteriores características y al contener otros elementos de tipo moléculas dentro del atomic design se considera un organismo. El diseño proporcionado desde el equipo de UX para este componente en el sistema de diseño Bancolombia fue el siguiente observado en la Figura 6.

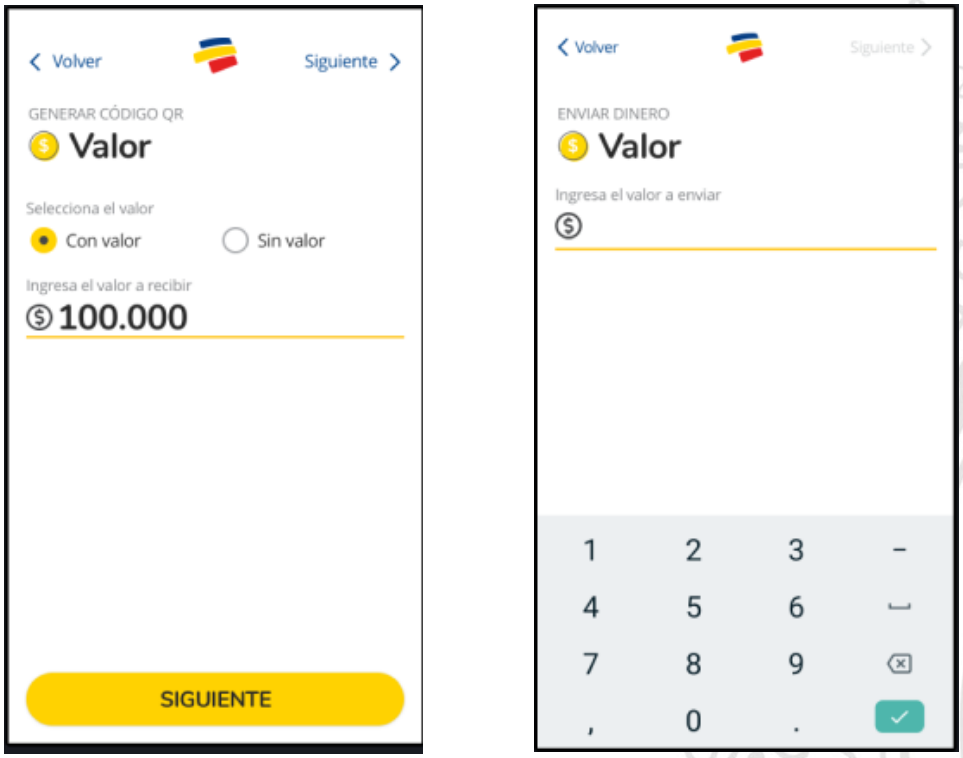

Figura 6. Definición de diseño componente Header Navigation Bar.

Basado en el anterior diseño de la Figura 6, se analizaron las propiedades con las cuales debía contar el componente y se identificaron las siguientes:

El componente debía permitir ingresar los ítems derecho e izquierdo por parámetros, tanto su texto, como el icono y si se encontraba activo o inactivo. Los anteriores siendo todos opcionales para el desarrollador.

La imagen central del header, podría variar por lo tanto también se podía pasar una imagen como parámetro, sino se pasaba tenía la imagen isotipo del banco por defecto.

Se identificó también que se requería realizar una animación al hacer scroll hacia arriba sobre la pantalla, en este caso la imagen central del header debía desvanecerse y en cambio aparecer el título de la pantalla que también es ingresado como parámetro. En este mismo escenario los textos laterales debían desaparecer y únicamente quedar los iconos. Si el usuario devolvía el scroll hacia abajo el header debía volver a su estado natural.

Para su codificación se utilizó un widget base ofrecido por el framework llamado Appbar de la librería material design, este ya por defecto está diseñado para ubicarse en la parte superior de la pantalla, así que se agregaron las propiedades y definiciones necesarias para la nueva clase correspondiente al componente Header deseado, con los tamaños, espaciados y colores definidos desde diseño.

En las pruebas de renderizado para este componente se cubrieron ítems inactivos, elementos opcionales, respuesta a tap en ítems y la detección correcta del cambio al realizar scroll.

Resultados y análisis

Componente 1 -Colors Dark

Para este componente se presenta su utilización en diferentes componentes implementados en la aplicación de muestra showcase, así mismo se realiza la comparación de cambio de la paleta de colores cuando no se encuentra en modo dark el tema del dispositivo. Observado en la Figura 7.

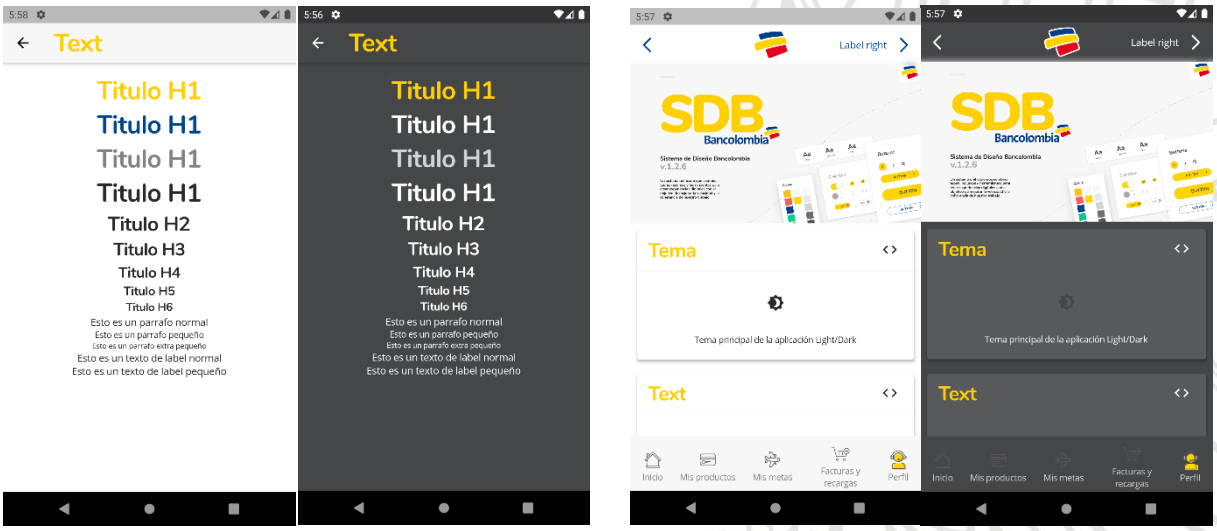

Figura 7. Pantallas Showcase resultado componente colors Dark.

Las cuatro pantallas expuestas en la Figura 7, corresponden a vistas de la aplicación showcase de muestra para los desarrolladores emulándose en un dispositivo Android. En estos pantallazos se muestran dos ejemplos de cómo cambian los respectivos colores de los elementos cuando al validarse el modo del tema en el que se encuentra configurado el dispositivo se le asignan los tokens de colores dark establecidos en el componente desarrollado.

Para acceder al uso de estos tokens de colores dark, se hace el llamado a través de la clase de la siguiente forma observada en la Figura 8.

```
background: BcColorsDark.BG_PRIMARY,
```

Figura 8. Fragmento de código implementación colors Dark.

En la Figura 8, background es la propiedad a la cuál se le desea asignar el color, BcColorsDark es el nombre del componente y BG_PRIMARY es el color que se desea asignar.

Componente 2 - Tags

En el caso de este componente para la implementación solo se requiere ingresar dos parámetros al constructor de la clase: el tipo de tag que se desea, esto a través de un enumerable, y el texto que se desea contenga el tag, como ejemplo:

```
    BcTag(
        text: "Información",
        alertType: BcTagType.Info,
    ),
```

Figura 9. Fragmento de código implementación Tags.

En el caso de la Figura 9 se implementa un tag de tipo info con el texto "Información".

En la Figura 10 se muestran los casos de uso en la aplicación de muestra showcase.

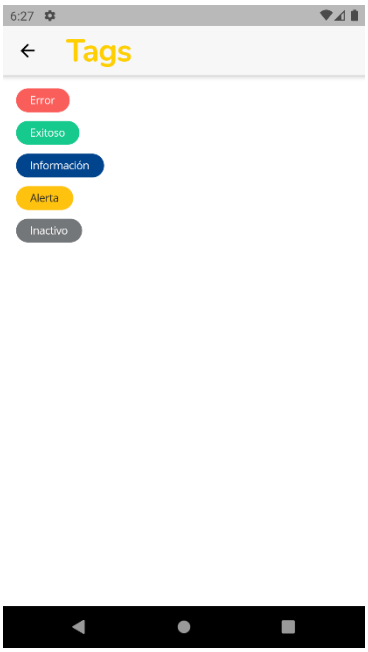

Figura 10. Pantalla Showcase resultado componente Tags.

En la Figura 10 se visualizan los 5 tipos de tags, Error, Exitoso, Información, Alerta e Inactivo.

Componente 3 – Etiquetas

La molécula de etiquetas tiene por su parte tiene 5 parámetros de entrada en su constructor para la implementación; 2 de estos son requeridos que corresponden un string del texto que contendrá la etiqueta y el otro es un boleano que indica si está activa o inactiva. Por parte de los opcionales son el icono de la izquierda, el icono de la derecha y la función callback del icono de la derecha al dar tap sobre el mismo.

Un ejemplo de su implementación es el mostrado en la Figura 11.

```
BcFilterLbl(
  text: 'Active',
  active: true,
  prefixIcon: BcFunctionalIcons.FILE_ALT,
  suffixIcon: BcFunctionalIcons.TIMES_CIRCLE,
),
```

Figura 11. Fragmento de código implementación Etiquetas.

En la Figura 11 se pasan ambos iconos, prefixIcon que corresponde al icono de la izquierda y suffixIcon que corresponde al de la derecha. El boleano que indica que está activa la etiqueta en verdadero y el texto que contendrá la etiqueta en este caso 'Active'.

Los diferentes casos de uso para las etiquetas se presentan en la aplicación showcase de la manera observada en la Figura 12.

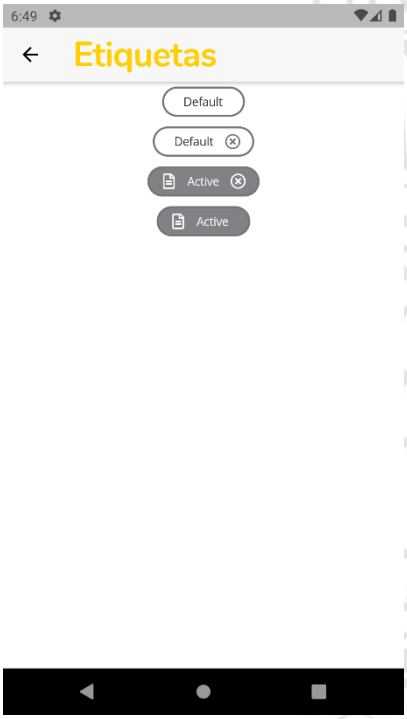

Figura 12. Pantalla Showcase resultado componente Etiquetas.

En la Figura 12 la primera etiqueta corresponde a una etiqueta inactiva sin iconos laterales, la segunda etiqueta corresponde a una etiqueta inactiva únicamente con icono a la derecha, la tercera etiqueta activa con ambos iconos laterales, y la cuarta etiqueta activa únicamente con icono a la izquierda.

Componente 4 – Layout

El componente Layout consta de una clase a la cual en su constructor se debe pasar como parámetro el contexto en el cual se desea utilizar, para así a través de éste identificar los tamaños y propiedades del dispositivo y pantalla donde se instancie el elemento.

Un ejemplo para declarar su implementación es el mostrado en la Figura 13.

```
BcResponsiveDesign _responsiveDesign = BcResponsiveDesign(context);
```

Figura 13. Fragmento de código implementación Layout.

Y para su utilización se pueden acceder a los métodos del componente, donde se les debe pasar el valor que desean sea responsivo respecto al tamaño de la pantalla, los métodos del componente son tanto para el ancho, el alto y el texto.

Ejemplos de utilización de los métodos son los observados en las Figuras 14 y 15.

```
double containerWidth = _responsiveDesign.widthMultiplier(46.0);
double containerHeight = _responsiveDesign.heightMultiplier(26.0);
```

Figura 14. Fragmento de código implementación Layout.

En la Figura 14 se utilizan las propiedades de tamaños para asignarse posteriormente a un contenedor.

```
fontSize: _responsiveDesign.textMultiplier(16.0);
```

Figura 15. Fragmento de código implementación Layout.

En la Figura 15 se le asigna a una propiedad de un componente el tamaño de la fuente de forma responsiva a través del uso del componente Layout.

Para mostrar su funcionamiento, se muestran en las Figuras 16 y 17 pantallas emuladas de la aplicación showcase, donde la disposición de los elementos y los componentes hacen uso del componente Layout responsivo.

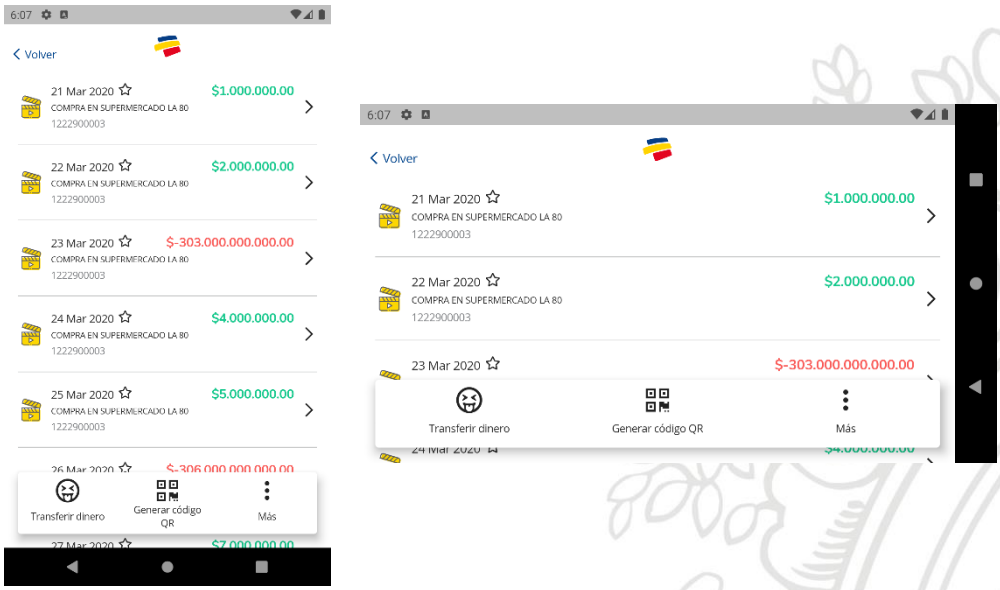

Figura 16. Pantallas Showcase resultado componente Layout.

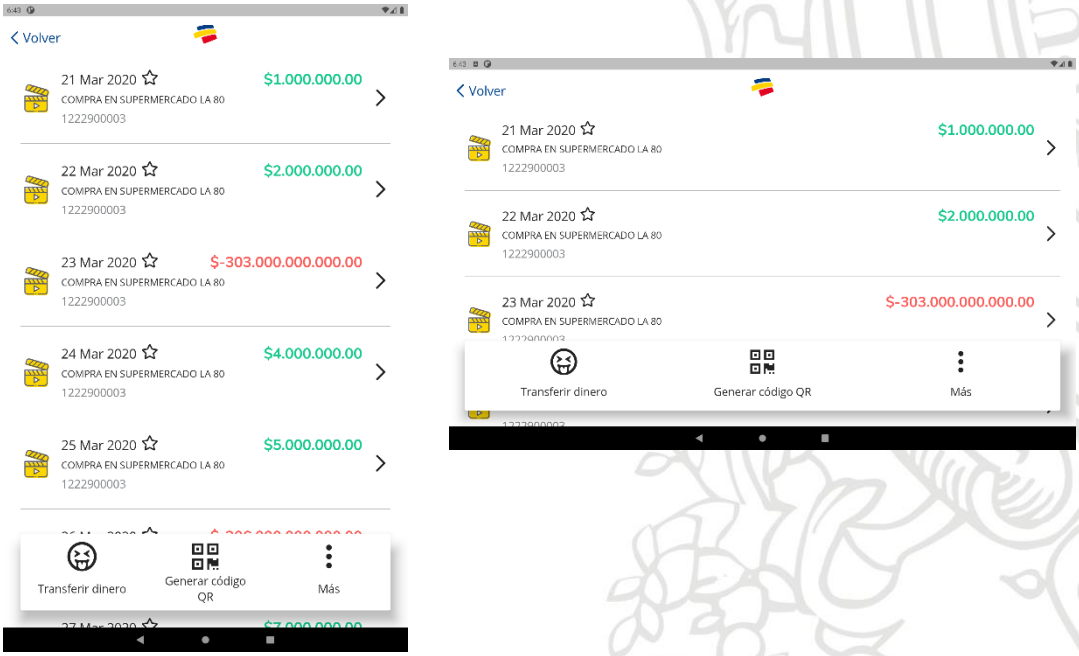

Figura 17. Pantallas Showcase resultado componente Layout.

La Figura 16 corresponde a un dispositivo móvil Pixel 2 en orientación vertical y horizontal; mientras que la Figura 17 corresponde a un dispositivo tipo tableta Nexus 10 también en orientación vertical y horizontal.

Como se observa en las Figuras 16 y 17 la configuración de la pantalla se conserva guardando las dimensiones apropiadas para cada contexto en el que se encuentra la pantalla, adaptando mediante el componente responsivo de Layout, los textos, altos y anchos.

Componente 5 – Header navigation bar

El organismo Header navigation bar cuenta con 6 parámetros en su constructor, uno único requerido correspondiente al string del texto que lrá como titulo de la pantalla cuando se realiza scroll sobre la pantalla.

Los 5 parámetros opcionales corresponden a una función callback para cuando se realiza tap sobre la imagen central, la imagen central la cual si no se pasa al constructor se utiliza el logotipo del banco por defecto, el controlador para el scroll y el los 2 item de la izquierda y derecha los cuales se les realizó un modelo para simplicidad el cual contiene el icono del item, el texto, un boleano que indica si está activo o inactivo y por último la función callback de cuando se realiza tap sobre el item.

Un ejemplo de su implementación en código es la siguiente mostrado en la Figura18.

```
BcHeaderNavigationBar(
        title: 'HEADER',
        image: BcIconSvg.assetLogo('Bancolombia/logo_primario_horizontal'),
        leftItem: BcHeaderNavigationBarItem(
          icon: BcFunctionalIcons.CHEVRON_LEFT,
          label: 'Label left',
          enabled: false,
          onButtonTap: _onTapped),
        rightItem: BcHeaderNavigationBarItem(
          icon: BcFunctionalIcons.SIGN_OUT_ALT,
          label: 'Cerrar sesión',
          onButtonTap: _onTapped)
      );
```

Figura 18. Fragmento de código implementación Header Navigation Bar.

En la Figura 18 se instancia el Header navigation bar con un string "HEADER" como título, se le pasa una imagen diferente del logo del Banco para ser ubicada en la parte central, como item de la izquierda se pasa el modelo con icono de flecha hacia la izquierda, con el string 'Label left' como texto del item, el boleano enable en falso haciendo que el item de la izquierda esté deshabilitado y la función callback

_onTapped asignada a la propiedad onButtonTap para ser ejecutada cuando este item sea presionado, para el item de la derecha se pasa el icono de salir, el string "Cerrar sesión" como label para el item e igualmente la función _onTapped asignada a la propiedad onButtonTap, como no se define la propiedad enabled por defecto el item estará habilitado.

A continuación, en la Figura 19 se presentan los casos de uso del componente implementados en la aplicación showcase.

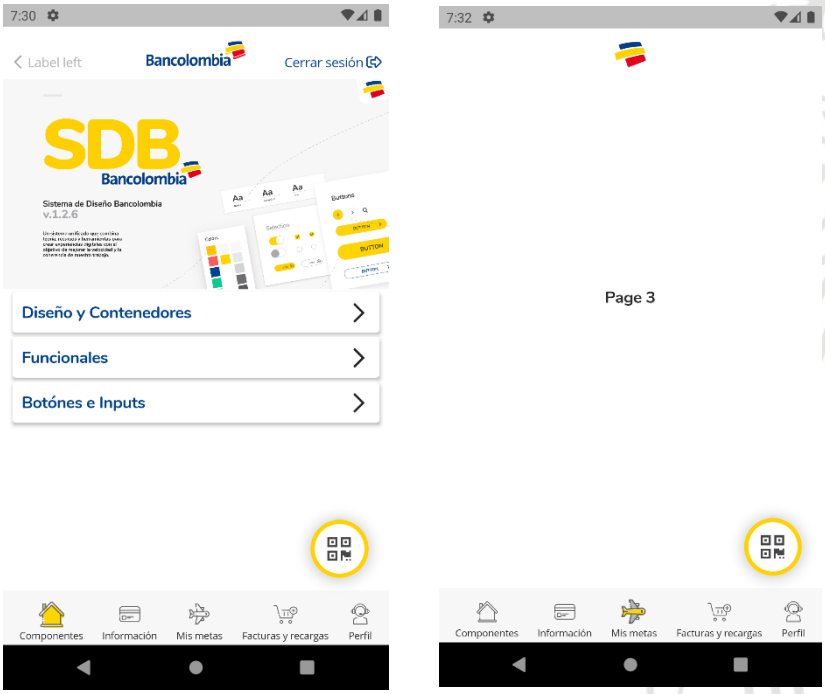

Figura 19. Pantallas Showcase resultado componente Header Navigation Bar.

En la Figura 19 la primera pantalla corresponde al ejemplo de implementación mencionado en el código, mientras que la segunda pantalla corresponde a un header únicamente con la imagen por defecto.

Cuando se agrega la propiedad del scroll controller al header navigation bar se permite tener la funcionalidad de animación y cambio del header al realizar scroll, esta característica se muestra en las siguientes pantallas de la Figura 20.

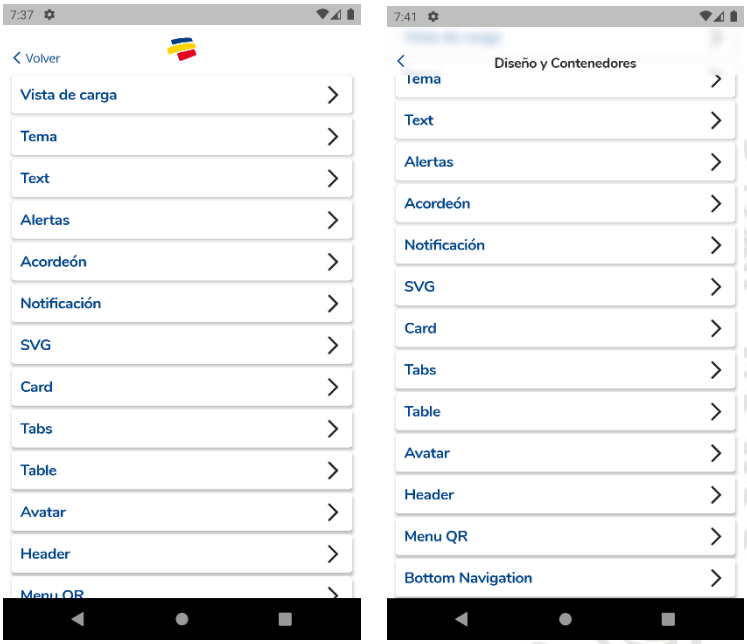

Figura 20. Pantallas Showcase resultado componente Header Navigation Bar.

Se observa en la Figura 20 cuando el usuario realiza scroll hacia arriba se detecta mediante el controller desde el componente header y realiza este cambio, dando efecto de blur o desenfoque sobre el fondo, intercambiando la imagen central por el título del header y quitando los label de los item de los lados únicamente dejando el icono; cuando el usuario vuelve a la posición inicial del scroll el header retoma su configuración inicial.

Conclusiones

1. Al tenerse una visión transversal de los componentes desarrollados ayudó a definir cuáles eran esos parámetros necesarios configurables para el desarrollador final, teniendo un componente con una estructura definida desde su diseño reutilizable para toda la gama de aplicativos móviles con la que cuenta el banco y posteriores.

1.1 Los componentes desarrollados hacen parte de una de las primeras API de consumo para el desarrollo front-end de aplicativos móviles; una idea adaptada desde el mundo web donde existen múltiples librerías para el desarrollo front-end de las vistas. Con esta API se agilizó el desarrollo de las aplicaciones móviles pues como se mostró en los resultados, son componentes de simplicidad en su implementación los cuales ya traen toda la configuración de visualización de acuerdo a los lineamientos de diseño del banco.

2. El desarrollo del sistema de diseño Bancolombia para aplicativos móviles utilizando las metodologías material design y atomic design hicieron que la implementación de este API fuera estructurada y limpia, de tal forma que se tengan tipados todos los componentes. Teniendo una jerarquía clara y permitiendo saber cuáles componentes están compuestos por otros.

2.1 El uso del framework flutter para la implementación de estos componentes reutilizables afianzó la transversalidad de los mismos, pues este framework de desarrollo es agnóstico al sistema operativo del dispositivo (iOS/Android) realizando la compilación a lenguaje nativo para ambos sistemas operativos.

3. Se encontró que al tener todos los componentes del API expuestos con sus diferentes casos de uso en el aplicativo de muestra showcase, agiliza el proceso para que el desarrollador a usar el API visualice los alcances del componente, cuáles sus son características principales y cómo podría implementarlo.

3.1 Como se realizaron pruebas unitarias de renderizado a cada componente dentro del API, logra minimizar el tiempo en la etapa de testeo en los aplicativos que hagan uso de esta API; ya que, al estar compuesta con estos componentes previamente testeados, muchas pruebas de renderizado no serán necesarias dentro del aplicativo final, optimizando los tiempos de desarrollo.

Referencias Bibliográficas

[1] Vazquez, Mabel & Vincent, Pierre & Nieto, Juan & Sanchez Lopez, Juan de Dios. (2012). Applying a Modular Framework to Develop Mobile Applications and Services. Journal of Universal Computer Science. 18. 704-727. 10.3217/jucs-018-05-0704.

[2] NSX. 2020. Que Ventajas Tienen Las Aplicaciones Móviles Modulares - NSX. [online] Available at: <http://www.e-nsx.com/2018/01/29/ventajas-apps-modulares/> [Accessed 11 September 2020].

[3] Medium. 2020. Build A Modular Android App Architecture. [online] Available at: <https://proandroiddev.com/build-a-modular-android-app-architecture-25342d99de82> [Accessed 11 September 2020].

[4]Product Design Handbook. [Online]. Disponible en: https://designhandbook.mendesaltaren.com/. [Accessed: 01- Sep- 2020].

[5]"Atomic Design Methodology | Atomic Design by Brad Frost", Atomicdesign.bradfrost.com, 2020. [Online]. Disponible en: https://atomicdesign.bradfrost.com/chapter-2/. [Accessed: 01- Sep- 2020].

[6]"Material Design", Material Design, 2020. [Online]. Disponible en: https://material.io/design/foundation-overview. [Accessed: 01- Sep- 2020]

[7]"Flutter architectural overview", Flutter.dev, 2020. [Online]. Disponible en: https://flutter.dev/docs/resources/architectural-overview. [Accessed: 01- Sep- 2020].

[8]"¿Por qué Flutter usa Dart?", Medium, 2020. [Online]. Disponible en: https://medium.com/comunidad-flutter/por-qu%C3%A9-flutter-usa-dart-8e7703650def. [Accessed: 01- Sep- 2020].

[9] Paradigmadigital.com. 2020. TDD, Una Metodología Para Gobernarlos A Todos. [online] Available at: <https://www.paradigmadigital.com/techbiz/tdd-una-metodologia-gobernarlos-todos/> [Accessed 11 September 2020].