

Informe de HCI

Implementación de Interfaz Móvil: Última Iteración funcional

Grupo 6

Bezdjian, Alejandro	abezjdia@itba.edu.ar	52108
Carmona, Lucas	lucarmon@itba.edu.ar	53051
Hirschowitz, Kevin	khirscho@itba.edu.ar	52539
Hurmuz, Agop	ahurmuz@itba.edu.ar	53248

Índice

Introducción	3
Inicio	4
Búsqueda	5
Productos	9
Producto	10
Pedidos	11
Notificaciones	15
Decisiones de Usabilidad	16
Incompatibilidad de idiomas en la Api	17
Instructivo de instalación	19
Conclusión	20

Introducción

En este informe describiremos el proceso que conllevó la implementación de la aplicación para celulares con sistema operativo Android. Si bien realizamos previamente una serie de prototipos no todas las ideas y pantallas quedaron idénticas al mismo, esto se debió a dificultades en la implementación y al darnos cuenta mientras las implementamos que otras disposiciones eran más convenientes para el usuario.

Inicio

RFB 1: Mostrar el listado de categorías de productos utilizando los siguientes filtros:

- o Género (femenino y masculino).

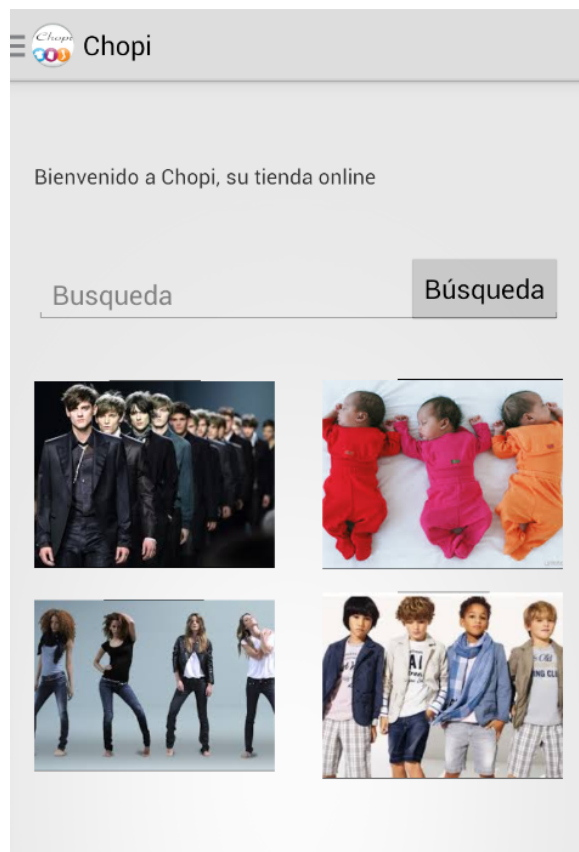
- o Edad (infantil y adulto).

RFB 2: Mostrar el listado de sub-categorías de productos utilizando los siguientes filtros:

- o Género (femenino y masculino).

- o Edad (infantil y adulto).

A continuación podemos ver la pantalla de inicio, donde se cuenta con las posibilidades de acceder a los productos en principio realizando un filtrado según género(Masculino y femenino) o por edad(Bebé e infantil). Si bien en el prototipo habíamos propuesto otro tipo de display de los productos nos pareció que para que el usuario que accede al sitio web sería más amigable la aplicación si presentaba imágenes, disposiciones y opciones similares a la misma. Por eso agregamos las 4 fotos indicando así la forma en que accedemos a los productos.



A continuación mostramos el escenario con el que nos enfrentamos en caso de seleccionar la foto del género “masculino”. La aplicación nos dirige a una nueva actividad que presenta una *expandableList* la cual tiene entre sus opciones principales las categorías y si seleccionamos determinada categoría se mostrará en modo de lista, todas las subcategorías pertinentes a la misma. Podemos destacar que no se cargan la totalidad de las categorías y subcategorías, sino solo las que están disponibles para el género o edad en cuestión. Mencionaremos y contrastamos esta diferencia más adelante cuando mostremos otro ejemplo relacionado con categorías y subcategorías. Por ejemplo, como accedimos mediante la foto del género “Masculino” dentro de accesorios sería inconsistente si apareciese la opción de aros o carteras. Decidimos agregar también la opción para ver todos los productos de la categoría, nos pareció una opción interesante para la navegación del usuario.

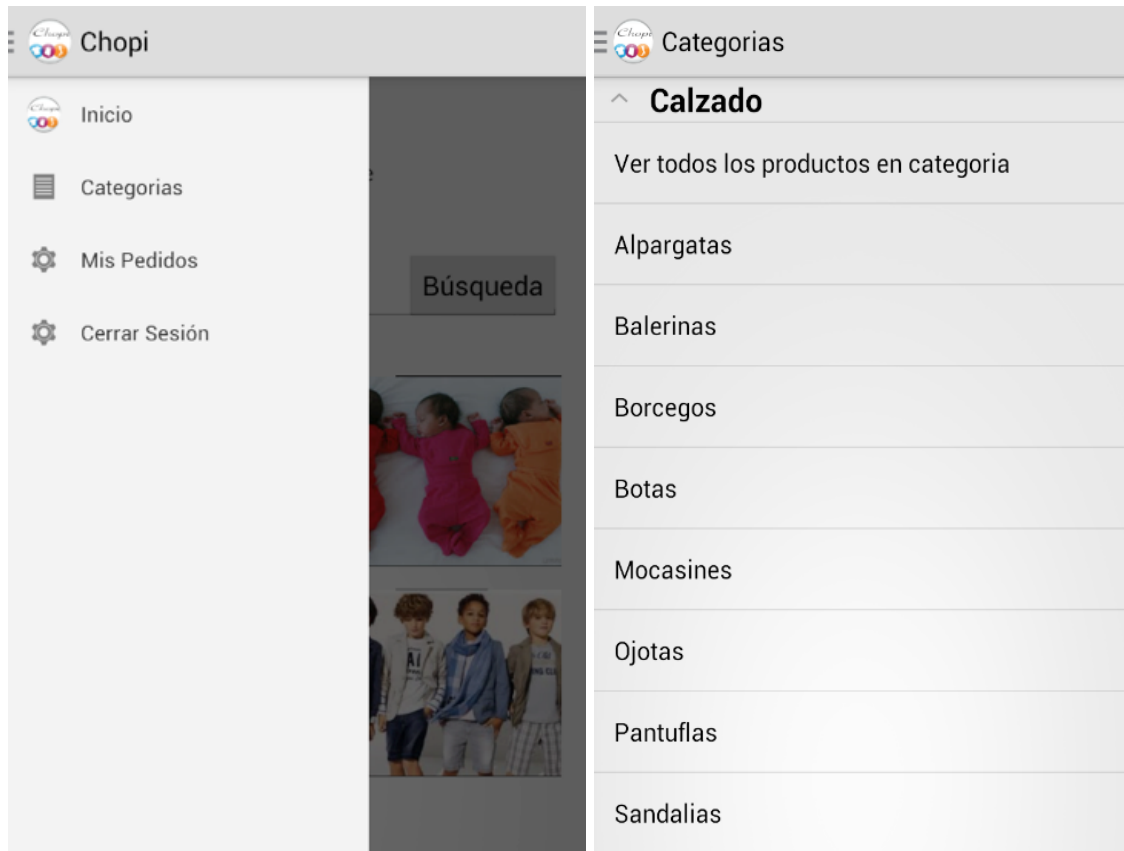


Búsqueda

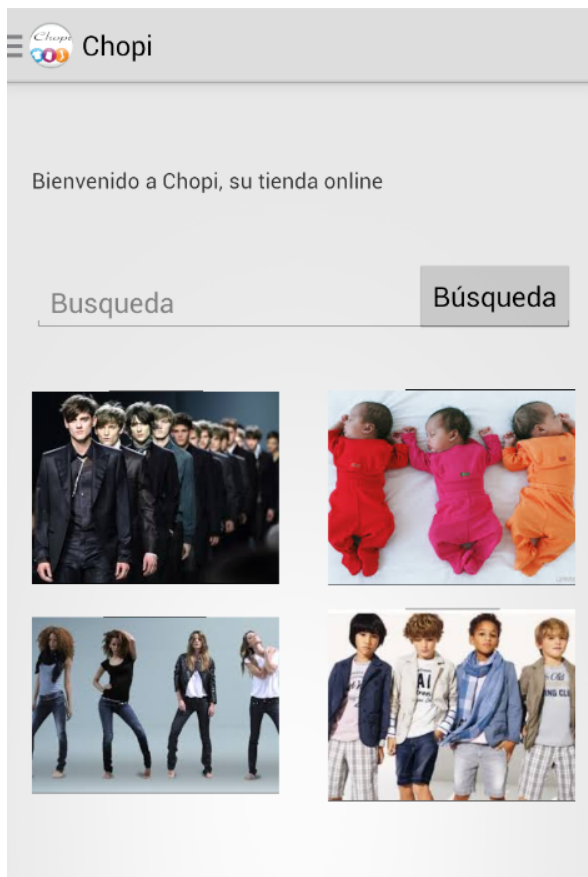
RFB 3: Buscar productos utilizando los siguientes filtros:

- o Categoría.
- o Sub-categoría.
- o Nombre.

Mostramos a continuación otra forma para acceder a los productos mediante el *Drawer*. En este último se encuentra en todas las actividades para que el usuario pueda acceder de cualquier lugar al inicio, una nueva búsqueda, a sus pedidos y cerrar su sesión. Dentro de las opciones del *drawer* podemos observar que la segunda opción nos lleva a categorías, sin embargo, las mismas no es exactamente la misma actividad la cual mostramos en el ejemplo anterior. Estas categorías y subcategorías que se muestran son generales a todos los géneros y edades. En las capturas de pantallas siguientes mostramos los pasos para poder acceder a la pantalla deseada:



Una vez seleccionada la opción de categorías se mostrará una *expandableList* como la que mencionamos en el ejemplo anterior, en la captura mostramos como es la vista cuando se abre una de las categorías. Tenemos la opción de *colapsar* las subcategorías para poder navegar de forma más fácil a través de las categorías.










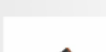
Si el usuario desea acceder a los productos por búsqueda puede hacerlo desde la pantalla principal de la aplicación. Cuando se presiona el botón de búsqueda la aplicación realizará un *textToSpeech* esto permite que la aplicación sea más amigable para usuarios con visibilidad reducida pudiendo así saber si tipearon bien el nombre del producto que buscaban.

Productos

RFB 4: Mostrar los siguientes datos para cada uno de los resultados de la búsqueda de productos:

- o Foto.
- o Marca.
- o Nombre.
- o Precio.

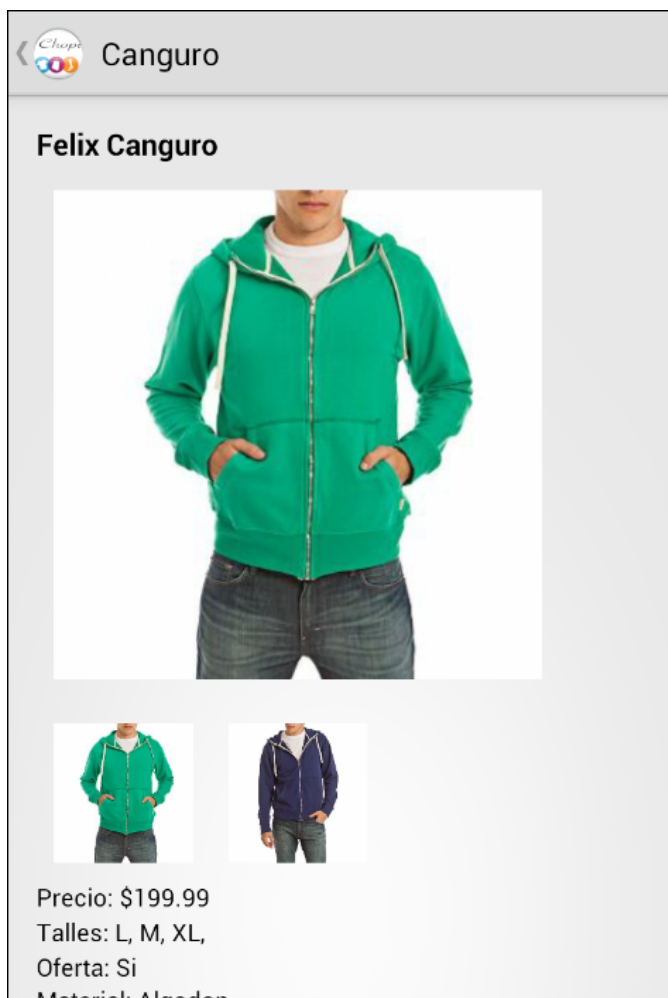
Una vez seleccionado el filtrado correspondiente se lleva a una nueva actividad donde el usuario puede observar todos los productos en una lista. Cada producto tiene la información básica como el nombre, marca, foto principal y precio; ya que si el usuario tiene interes en algun producto en particular accede al mismo el cual lo lleva a otra actividad mostrando mas fotos, y todos sus detalles.

 Productos	
	Pancha Beige Hush Puppies Seattle - Roi \$470.0
	Pancha Blanca Late Miano - Coca-Cola Shoes \$598.0
	Pancha Blanca Patagonia - Red Horn \$630.0
	Pancha Combinada Azul Late Xadrez - Coca-Cola Shoes \$470.0
	Pancha Coral - Levi's \$458.0
	Pancha Gris Late Sanded - Coca-Cola Shoes \$598.0
	Pancha Negra Late Milano - Coca-Cola Shoes

Producto

RFB 5: Mostrar los siguientes datos al seleccionar cualquiera de los resultados de la búsqueda de productos:

- o Fotos.
- o Marca.
- o Nombre.
- o Precio.
- o Colores.
- o Talles.
- o Detalles.



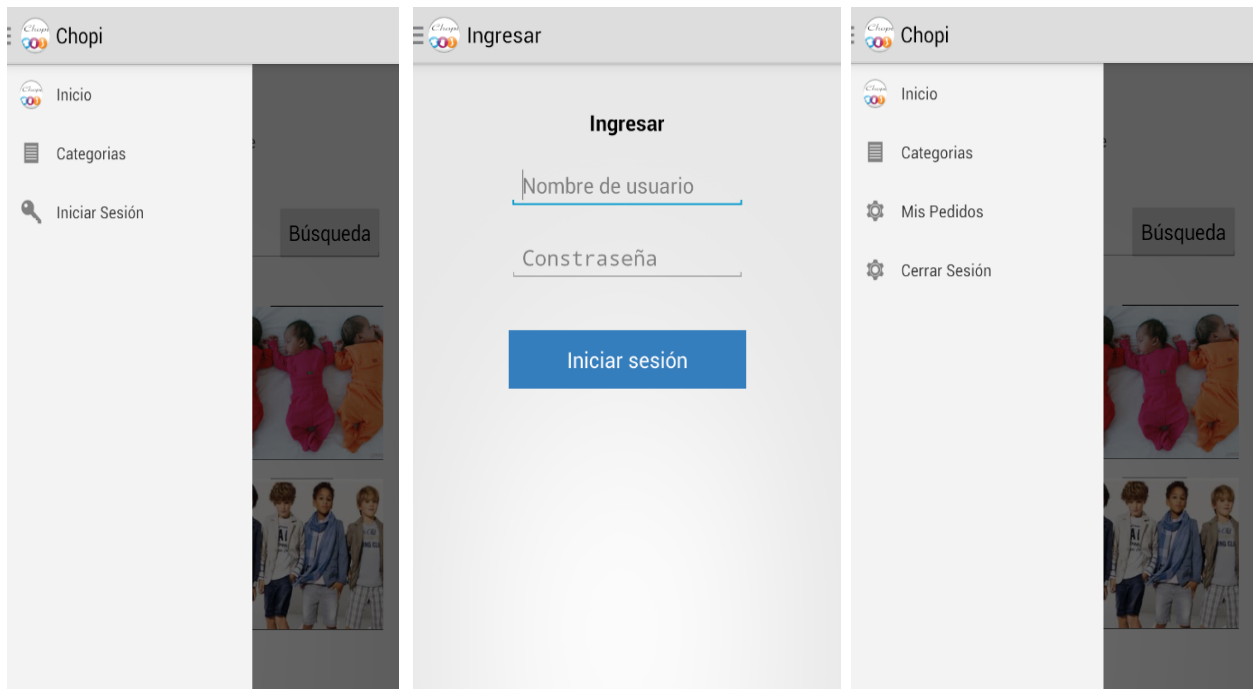
Mostramos el prototipo contrastado con el producto, la principal diferencia entre ambos es que el prototipo proponía un carrusel para las fotos, el mismo no fue implementado debido a su complejidad a la hora de aplicarlo. Decidimos agregar las fotos del producto de forma tal que el usuario pueda verlas todas a la vez, así, el usuario puede en menos tiempo tener una idea sobre como es el producto sin perder tiempo haciendo click para la siguiente foto. Esta vista es similar a la presentada en la página web para que el usuario se sienta familiarizado con el sitio y para navegarlo con más comodidad.

Pedidos


RFB 6: Mostrar los siguientes datos para cada uno de los pedidos del listado de pedidos.

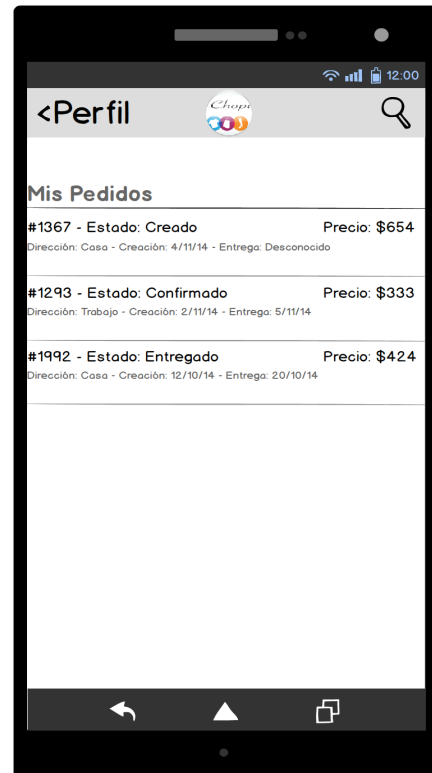
- o Número.
- o Estado (creado, confirmado, despachado o entregado).
- o Dirección de envío.
- o Fecha de creación.
- o Fecha de entrega.
- o Precio total.

Para poder mostrar los pedidos de un usuario fue necesario primero verificar quien era el usuario en cuestión. Decidimos agregar en nuestro *Drawer* la opción para que el usuario inicie y cierre sesión, y, si el usuario había ingresado al sistema se le agregaría la opción extra para visualizar sus pedidos.



Si seleccionamos la opción para ver nuestros pedidos nos dirigirán a una nueva actividad, en la misma tendremos un listado con las distintas órdenes, mostrando sus direcciones y estados. Para mostrar estos los datos mencionados en esta actividad nos alcanzó con ejecutar la query *GetAllOrders* de un determinado usuario, sin embargo, cuando quisimos obtener el precio total, vimos que en el array de orders que traía esta consulta este dato no estaba. Por lo tanto, decidimos que era mejor hacer una nueva actividad que mostraría los datos completos de la orden, teniendo que hacer una query específica para esta nueva actividad con *GetOrderById*. Mostramos a continuación la actividad que muestra todos los pedidos asociados a un determinado usuario, realizando click en alguno de ellos veremos más información sobre la orden en cuestión:

 Mis Pedidos
Numero de orden: 1477
Direccion: Facu
Estado: Confirmado
Numero de orden: 1129
Direccion: kevin
Estado: En Transito
Numero de orden: 1130
Direccion: aa
Estado: Entregado
Numero de orden: 1133
Direccion: aa
Estado: Confirmado
Numero de orden: 1215
Direccion: Trabajo
Estado: Confirmado



La imagen de la izquierda se corresponde a la implementación en nuestra app y la de la derecha la propuesta en el prototipo, podemos mencionar que la misma es prácticamente igual con la salvedad que la de la implementación no informa el precio total de la orden. Esto se debió principalmente a la razón mencionada anteriormente sobre la consulta a la API que no informaba este dato. Lo que se le agregó a esta vista fue también una lista de los productos para que el usuario pueda ver con mayor comodidad y detalles su compra. Mostramos a continuación la vista que obtenemos en caso de clicar alguna orden:



Pedido: 1129

Precio total: \$6838

Latitud:2

Longitud:-180

Estado:En Transito

Direccion: kevin



Airborn Point

Precio: \$999.99 Cantidad: 6



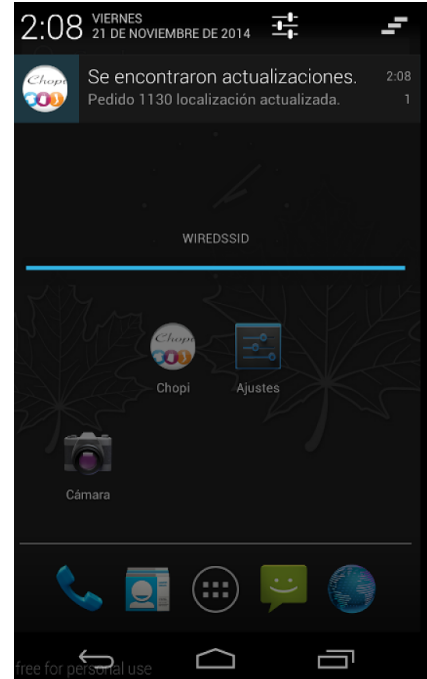
Active Padded

Precio: \$839.99 Cantidad: 1

Notificaciones

RFB 7: Alertar sobre cambios en el estado (despachado o entregado) o en la posición geográfica (latitud y/o longitud) que puedan sufrir ciertos pedidos de interés mediante notificaciones en la barra de estado del dispositivo.

Podemos observar que a pesar de no nos encontramos con la aplicación abierta, si se realiza alguna modificación en la orden se muestra en una notificación de todas formas.



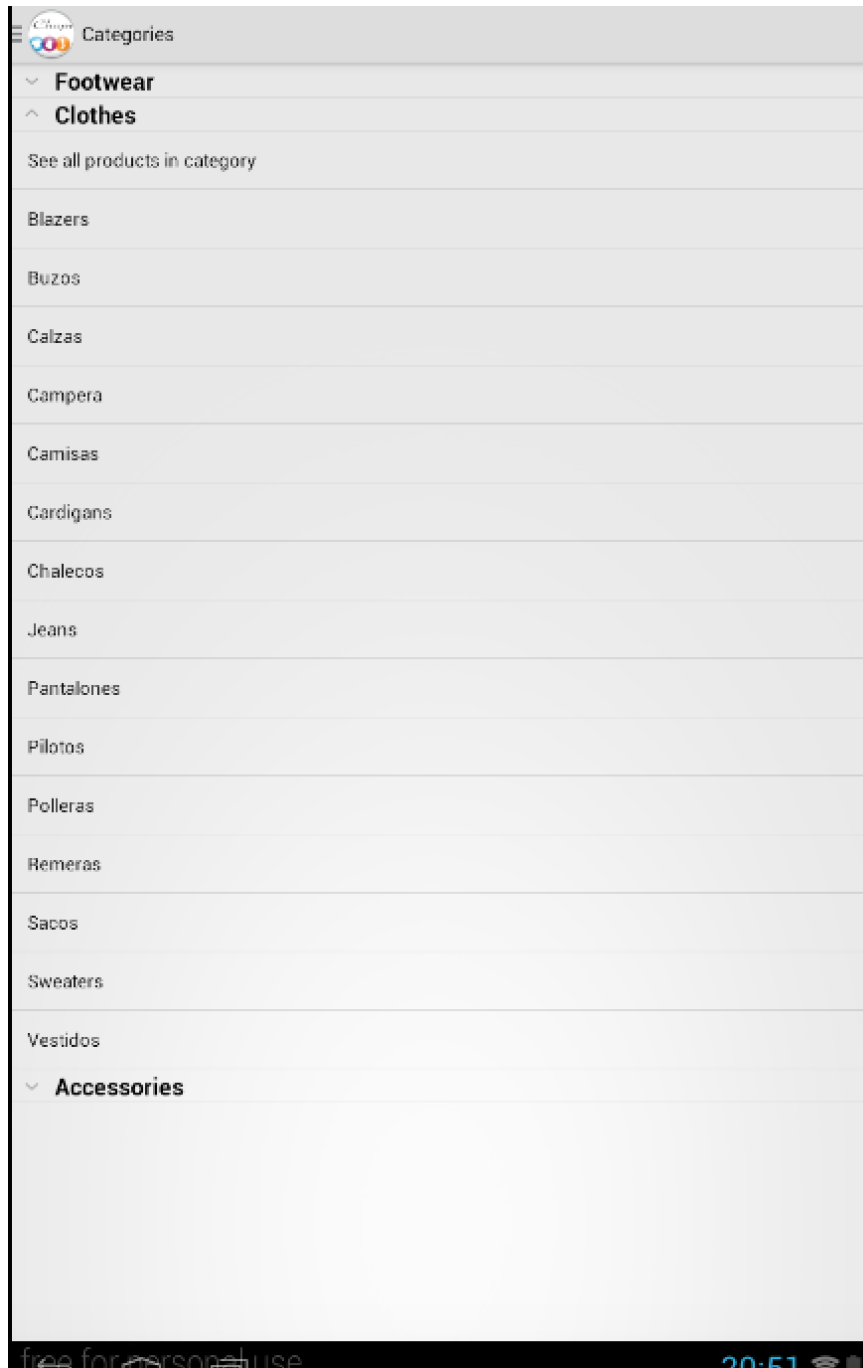
Decisiones de usabilidad



Decidimos no implementar algunas de las funcionalidades que habíamos incluido en el prototipo ya que no eran obligatorias tal como los filtros y organizar los resultados por marca, color y ocasión. Por otra parte, nos pareció que si incluimos en el drawer las subcategorías la información quedaría muy colapsada ya que podría haber 10 subcategorías o más. Además, tener 2 drawers que se desplazan de izquierda a derecha y que su contenido varíe radicalmente consideramos que era confuso para los usuarios más inexpertos.

Incompatibilidad de idiomas en la api

A pesar de que se pedía que tuviera versiones en castellano e inglés hubo partes de la aplicación que decidimos que no tenga este tipo de traducción ya que ensuciaría el código, tal es el caso de las subcategorías. Si bien pudimos realizar la traducción de las categorías ya que eran solo 3, las subcategorías eran demasiadas y no era mantenible en el tiempo si a futuro se agregara una nueva subcategoría. Para poder brindar el comportamiento deseado sería necesario realizar una consulta a la Api y que la misma tenga la opción mediante un filtro extra por ejemplo de devolver los resultados en inglés. Mostramos una captura a continuación sobre el problema mencionado anteriormente y cómo decidimos dejar nuestra implementación:



Instructivo de instalación

Desarrollamos la aplicación utilizando Android Studio para facilitar el agregado de librerías que facilitan la realización de consultas a la API, con el fin de ejecutar de modo más simple la aplicación sin la necesidad de la instalación y agregado de todas estas librerías

externas(*Retrofit* y *Robospice* entre otras) generamos el archivo : .apk ubicado en la carpeta bin dentro de TPE2.

Basta con pasarse el archivo al celular con sistema operativo Android, ya sea por cable USB o por drive/mail e instalar la aplicación, posteriormente usted podrá utilizar todas las funcionalidades de la misma.

Si uno lo desea puede instalar android Studio y agregar las siguientes librerías externas:

- gson-2.3 (Parseo de json a objetos java)
- picasso-2.4.0 (Manejo de imágenes)
- retrofit-1.7.1
- robospice-1.4.14
- robospice-cache-1.4.14
- robospice-retrofit-1.4.14
- support-annotations-20.0.0

y conectando el dispositivo por usb podría bootear la app en su celular android.

Conclusión

Si bien la implementación de la aplicación no es idéntica a el prototipo que nos planteamos en principio presenta las mismas funcionalidades básicas. Algunas de las razones por las que se diferencian fue que no sabíamos la dificultad que implicaba realizar algunas funciones y disposición de objetos, tal es el caso del carrusel. Decidimos entonces elegir otras opciones para mostrar la misma información. Por otra parte, en el prototipo habíamos incluido funcionalidades que no eran obligatorias, decidimos por un tema de tiempo no implementarlas.