INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Application Development For Mobile Devices

Material Design

Emiliano González López

19/junio/2020

Contenido

Material Desing

Material design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma. Fue desarrollado por Google y anunciado en la conferencia Google I/O celebrada el 25 de junio de 2014. Ampliando la interfaz de tarjetas vista por primera vez en Google Now.

Material se integró en Android Lollipop como reemplazo de Holo, anteriormente utilizado desde Android 4 y sucesores. La filosofía también se aplicó en Google Drive y Google Docs, Sheets y Slides, y se irá extendiendo progresivamente a todos los productos de Google (incluyendo Google Search, Gmail y Google Calendar), proporcionando una experiencia consistente en todas las plataformas. Google también lanzó APIs para que los desarrolladores externos incorporaran Material Design a sus aplicaciones.

Características

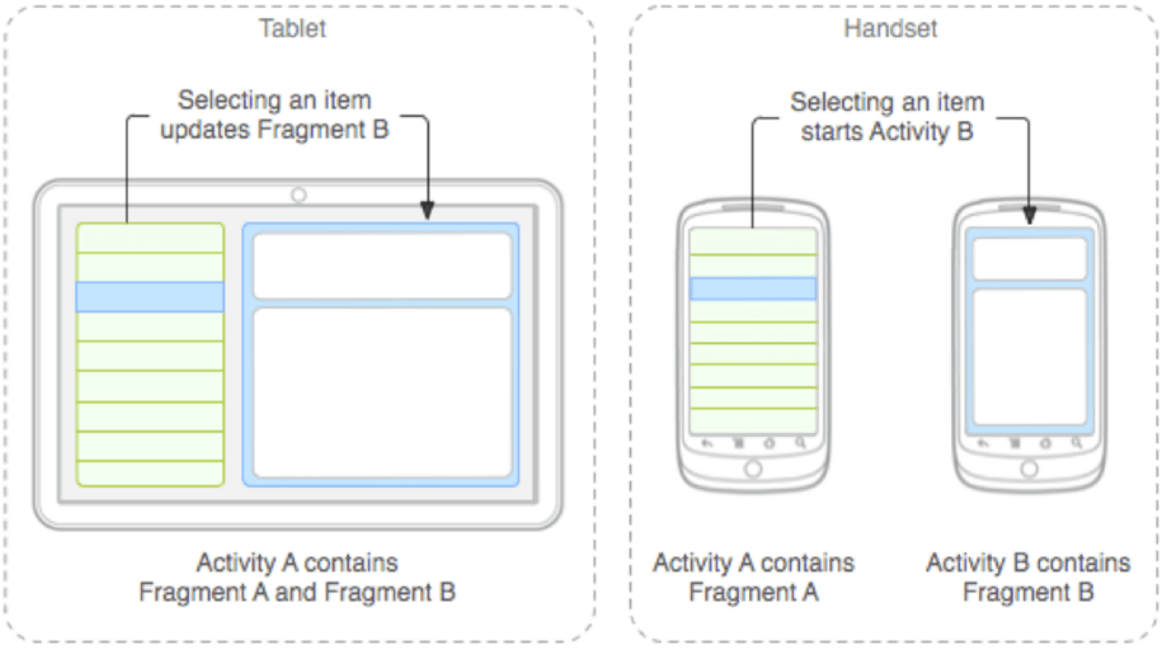
- Es un lenguaje visual de diseño común, que posee sus propias normas para casi todos los detalles y se mantienen independientemente del tamaño de pantalla, donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal.
- Incluye una renovación de la tipografía Roboto para ésta que pueda adaptarse correctamente a todas las plataformas.
- Es amplio, gráfico e intuitivo. Los elementos fundamentales de diseño se basan en tipografía de imprenta, mallas, espacio, escala, color y uso de imágenes.
- Los tratamientos visuales se hacen más agradable a la vista. Opciones adecuadas de color, imágenes de borde a borde, tipografía a gran escala y el espacio, crean una interfaz gráfica amplia para sumergir al usuario en la experiencia.
- Con el movimiento se respeta y refuerza al usuario como el motor principal.
- El diseño de material proporciona un conjunto de propiedades para personalizar el tema de diseño de materiales en color.

Tambien se ocupa uno de los conceptos más difíciles de usar en Android que son los fragmentos.

Los Fragmentos en Android, son secciones reutilizables de la interfaz de usuario, estos fragments tienen su propio ciclo de vida tal cual lo tienen las actividades.

Dentro de una actividad pueden haber varios fragmentos, siempre van asignados a la actividad y su ciclo de vida esta ligada al ciclo de vida de la Actividad o Activity en el que se insertan, pueden agregarse o quitarse mientras el Activity esta activo.

Un fragment, representa un comportamiento o una parte de la interfaz de usuario que se implementa en el Activity, así como tiene su propio ciclo de vida, también tienes sus propios eventos.



Desarrollo

En este caso esta práctica es más el caso de un desarrollador de UX/UI, desarrollando un ejercicio completo del estándar de diseño para las aplicaciones de Android, Material design.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mToolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayShowHomeEnabled(true);
    drawerFragment = (FragmentDrawer)
            getSupportFragmentManager().findFragmentById(R.id.fragment_navigation_drawer);
    drawerFragment.setUp(R.id.fragment_navigation_drawer, (DrawerLayout)
            findViewById(R.id.drawer_layout), mToolbar);
    drawerFragment.setDrawerListener(this);
    // display the first navigation drawer view on app launch
    displayView(0);
}
```

En la archivo Main solo declararemos la ToolBar, y desplegamos el fragment principal del cual dependejran los demas

```
public class FragmentDrawer extends Fragment {
    private static String TAG = FragmentDrawer.class.getSimpleName();
    private RecyclerView recyclerView;
    private ActionBarDrawerToggle mDrawerToggle;
    private DrawerLayout mDrawerLayout;
    private NavigationDrawerAdapter adapter;
    private View containerView;
    private static String[] titles = null;
    private FragmentDrawerListener drawerListener;
    public FragmentDrawer() {
    }
    public void setDrawerListener(FragmentDrawerListener listener) {
        this.drawerListener = listener;
    }
    public static List<NavDrawerItem> getData() {
        List<NavDrawerItem> data = new ArrayList<>();
        // preparing navigation drawer items
        for (int i = 0; i < titles.length; i++) {
            NavDrawerItem navItem = new NavDrawerItem();
            navItem.setTitle(titles[i]);
            data.add(navItem);
        }
        return data;
    }
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // drawer labels
        titles = getActivity().getResources().getStringArray(R.array.nav_dra
wer_labels);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflating view layout
        View layout = inflater.inflate(R.layout.fragment_navigation_drawer,
container,
                false);
        recyclerView = (RecyclerView) layout.findViewById(R.id.drawerList);
        adapter = new NavigationDrawerAdapter(getActivity(), getData());
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(getActivity())
);
        recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getAct
ivity(),
                recyclerView, new ClickListener() {
            @Override
            public void onClick(View view, int position) {
                drawerListener.onDrawerItemSelected(view, position);
                mDrawerLayout.closeDrawer(containerView);
            }
            @Override
            public void onLongClick(View view, int position) {
            }
        }));
        return layout;
    }

    public void setUp(int fragmentId, DrawerLayout drawerLayout, final Toolb
ar toolbar) {
        containerView = getActivity().findViewById(fragmentId);
        mDrawerLayout = drawerLayout;
        mDrawerToggle = new ActionBarDrawerToggle(getActivity(), drawerLayou
t, toolbar,
                R.string.drawer_open, R.string.drawer_close) {
            @Override
            public void onDrawerOpened(View drawerView) {
                super.onDrawerOpened(drawerView);
                getActivity().invalidateOptionsMenu();
            }
```

```
            @Override
            public void onDrawerClosed(View drawerView) {
                super.onDrawerClosed(drawerView);
                getActivity().invalidateOptionsMenu();
            }
            @Override
            public void onDrawerSlide(View drawerView, float slideOffset) {
                super.onDrawerSlide(drawerView, slideOffset);
                toolbar.setAlpha(1 - slideOffset / 2);
            }
        };
        mDrawerLayout.setDrawerListener(mDrawerToggle);
        mDrawerLayout.post(new Runnable() {
            @Override
            public void run() {
                mDrawerToggle.syncState();
            }
        });
    }
    public static interface ClickListener {
        public void onClick(View view, int position);
        public void onLongClick(View view, int position);
    }
    static class RecyclerTouchListener implements RecyclerView.OnItemTouchLi
stener {
        private GestureDetector gestureDetector;
        private ClickListener clickListener;
        public RecyclerTouchListener(Context context, final RecyclerView rec
yclerView,
                                     final ClickListener clickListener) {
            this.clickListener = clickListener;
            gestureDetector = new GestureDetector(context, new
                    GestureDetector.SimpleOnGestureListener() {
                        @Override
                        public boolean onSingleTapUp(MotionEvent e) {
                            return true;
                        }
                        @Override
                        public void onLongPress(MotionEvent e) {
                            View child = recyclerView.findChildViewUnder(e.g
etX(), e.getY());
                            if (child != null && clickListener != null) {
                                clickListener.onLongClick(child,
                                recyclerView.getChildPosition(child));
                            }
```

```
                        }
                    });
        }
        @Override
        public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e)
 {
            View child = rv.findChildViewUnder(e.getX(), e.getY());
            if (child != null && clickListener != null && gestureDetector.on
TouchEvent(e))
            {
                clickListener.onClick(child, rv.getChildPosition(child));
            }
            return false;
        }
        @Override
        public void onTouchEvent(RecyclerView rv, MotionEvent e) {
        }
        @Override
        public void onRequestDisallowInterceptTouchEvent(boolean disallowInt
ercept) {
        }
    }
    public interface FragmentDrawerListener {
        public void onDrawerItemSelected(View view, int position);
    }
}
```

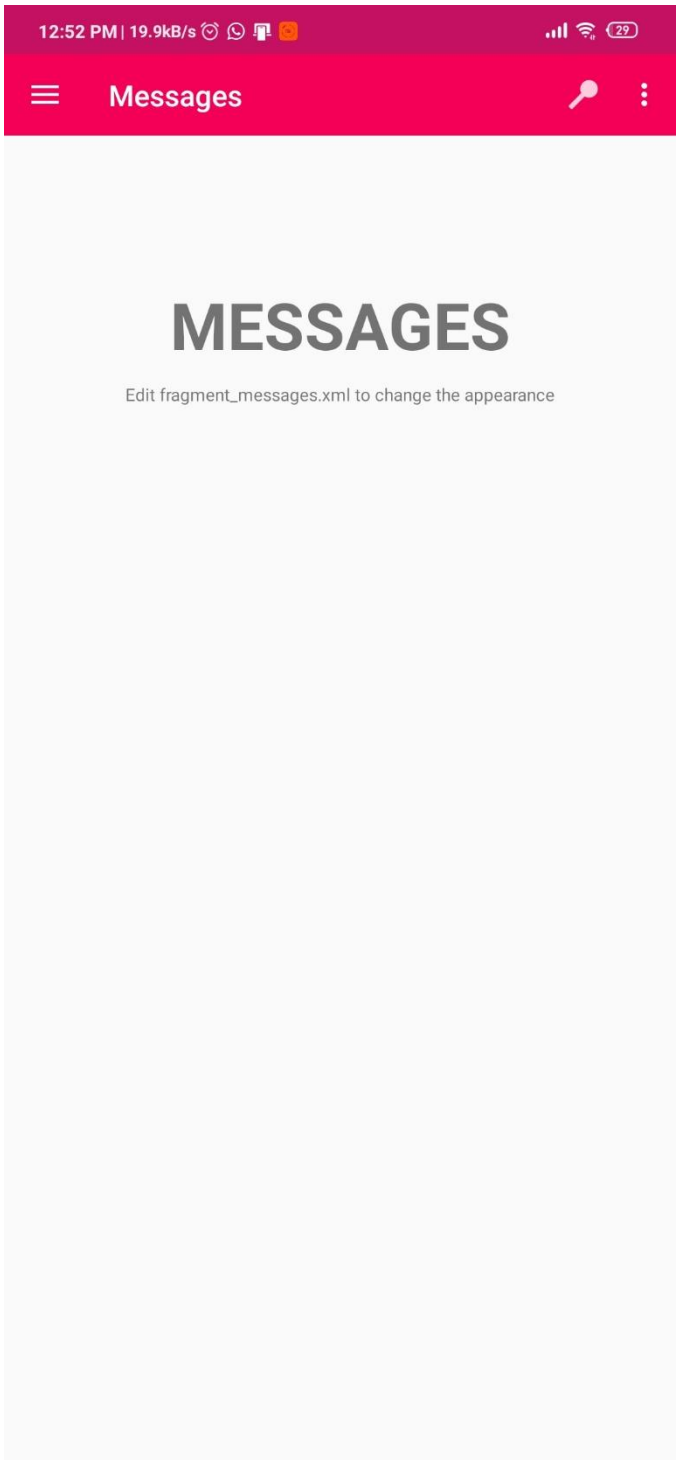
Tambien tenemos una clase NavigationDrawerAdapter que será la que permita los cambios de fragmentos.

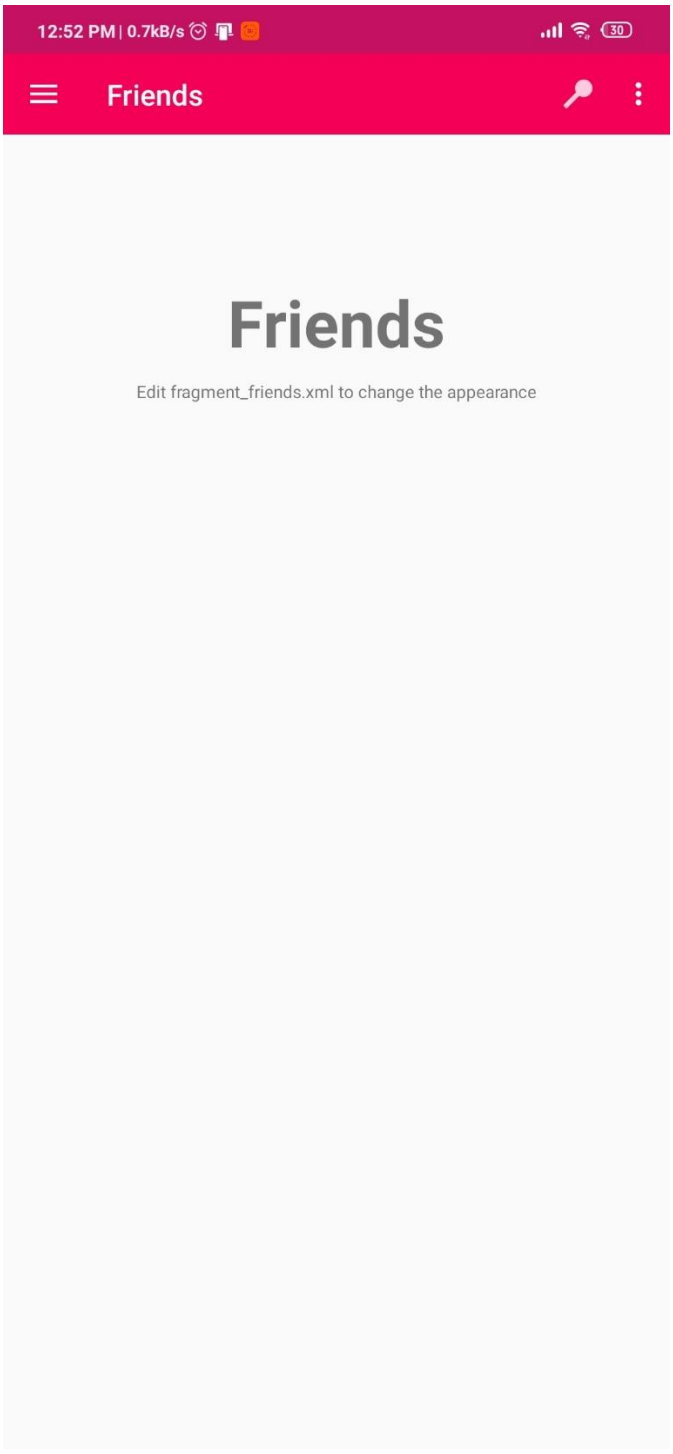
Pero la que gestiona los cambios con el menú lateral es la clase FragmenDrawer

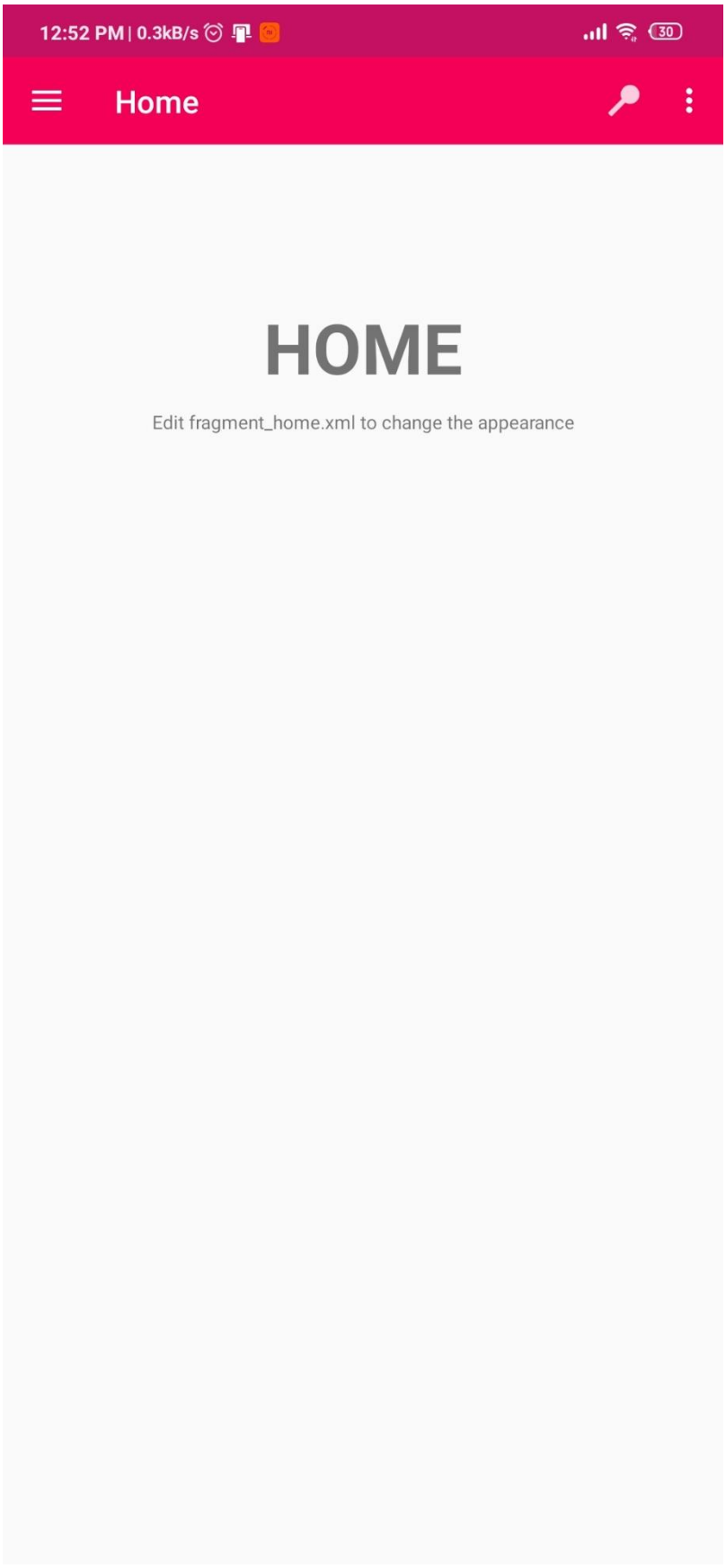
```
public class NavigationDrawerAdapter extends
        RecyclerView.Adapter<NavigationDrawerAdapter.MyViewHolder> {
    List<NavDrawerItem> data;
    private LayoutInflater inflater;
    private Context context;
    public NavigationDrawerAdapter(Context context, List<NavDrawerItem> data
) {
        this.context = context;
        inflater = LayoutInflater.from(context);
        this.data = data;
    }
    public void delete(int position) {
        data.remove(position);
```

```
        notifyItemRemoved(position);
    }
    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = inflater.inflate(R.layout.nav_drawer_row, parent, false)
;
        MyViewHolder holder = new MyViewHolder(view);
        return holder;
    }
    @Override
    public void onBindViewHolder(MyViewHolder holder, int position) {
        NavDrawerItem current = data.get(position);
        holder.title.setText(current.getTitle());
    }
    @Override
    public int getItemCount() {
        return data.size();
    }
    class MyViewHolder extends RecyclerView.ViewHolder {
        TextView title;
        public MyViewHolder(View itemView) {
            super(itemView);
            title = (TextView) itemView.findViewById(R.id.title);
        }
    }
}
```

Pruebas


12:52 PM | 19.9kB/s
Messages
MESSAGES
Edit fragment_messages.xml to change the appearance


12:52 PM | 0.7kB/s
Friends
Friends
Edit fragment_friends.xml to change the appearance


12:52 PM | 0.3kB/s
30
Home
HOME
Edit fragment_home.xml to change the appearance