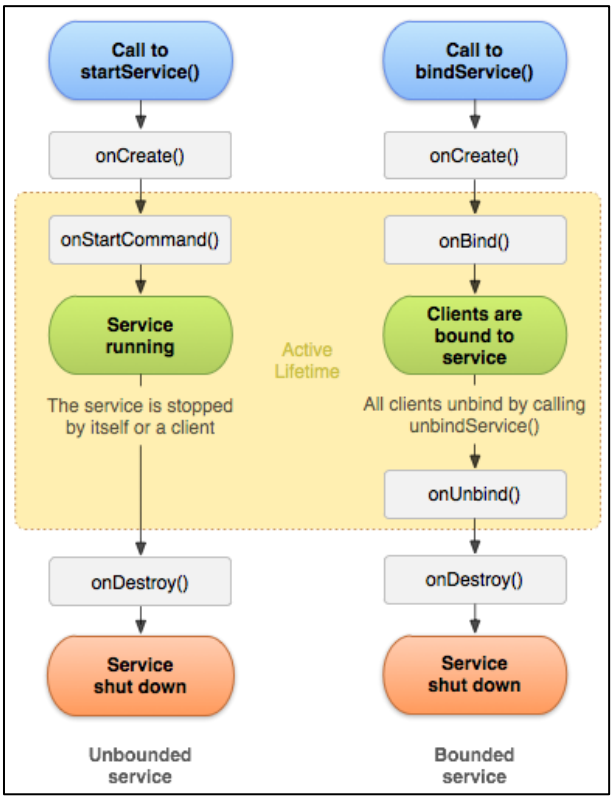
Povezani servis

Vezani servis je sličan pokrenutom servisu, osim što pokrenuti servis ne omogućava komunikaciju sa komponentom koja ga pokreće. Sa druge strane, vezani servis omogućava komunikaciju sa komponentom koja ga pokreće na principu server-klijent. Implementacijom komunikacije međuprocesima (IPC) ova interakcija može da se vrši i van granica procesa.

Za razliku od pokrenutih servisa, na vezane servise može biti povezano više komponenata klijenata. Kada su povezane, ove komponente komuniciraju sa servisom preko različitih mehanizama.

Vezani servis se implementira kao podklasa klase Service. On mora da implementira metodu `onBind` koja se poziva po povezivanju komponente na servis. Klijenti se na servis povezuju pozivanjem metode `bindService`. Takođe, klijenti moraju da implementiraju interfejs `ServiceConnection` i implementiraju njegove metode `onServiceConnected` i `onServiceDisconnected` koje se pozivaju kada se veza sa servisom kreira, odnosno uništava. Klijent se odvezuje sa servisa pozivom metode `unbindService`. S obzirom da više klijenata može biti povezano na servis, tek kada se svi klijenti odvežu sa servisa, servis se završava (životni vek servisa).

Takođe, vezani servis može biti pokrenut kao pokrenuti (pozivom metode `startService`). Međutim, u tom slučaju servis ostaje pokrenut čak i kada se svi klijenti odvežu sa njega, dok se ne pozove metoda `stopService`, odnosno `stopSelf`.



Slika 1 - Životni vek servisa

Binder

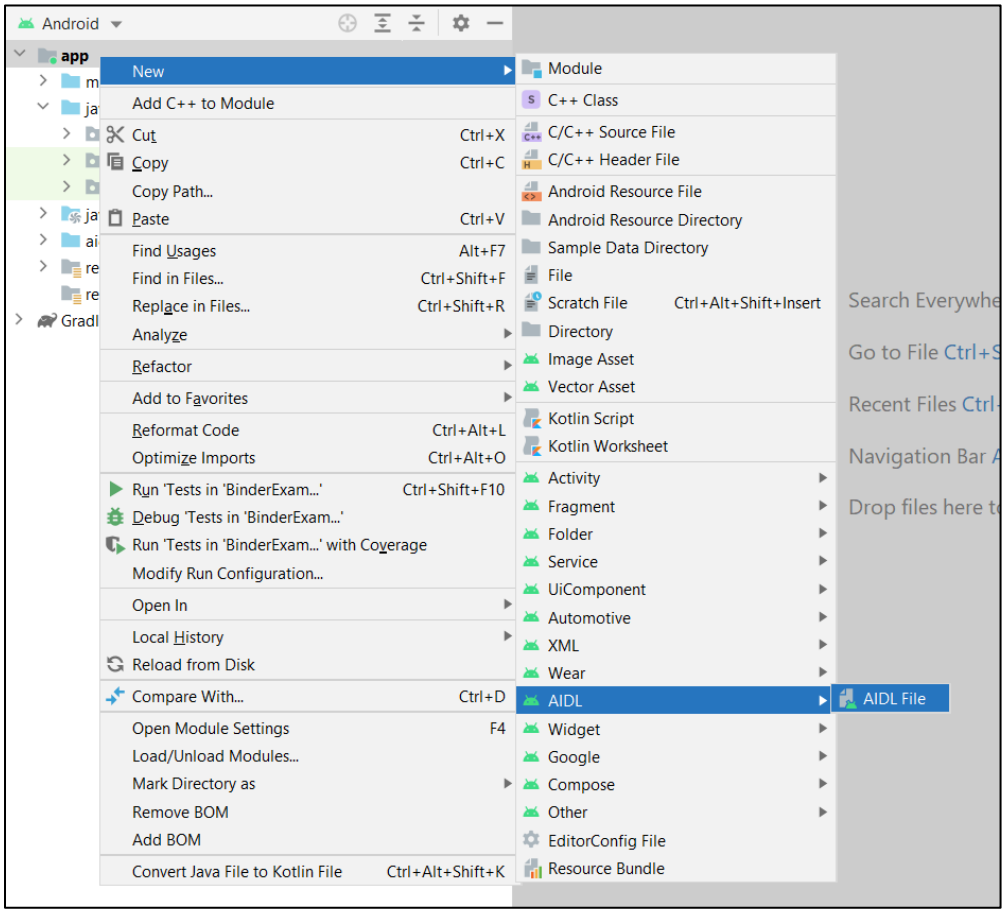
Jedan od mehanizama za komunikaciju sa povezanim servisom je **Binder** mehanizam. Ovaj mehanizam pruža informacije potrebne za komunikaciju sa servisom. Metoda `onBind` vraća objekat Binder, koji će klijenti koristiti za direktno pristupanje metodama unutar servisa. Metodi `onServiceDisconnected` se prosleđuje IBinder objekat.

AIDL - Android Interface Definition Language

Na nivou Java programskog jezika Binder komunikacija se izvršava time što postoji alat za generisanje celokupne komunikacije, a na korisniku je samo da upotrebi dobijeno rešenje. Definisan je programski jezik AIDL (*AIDL - Androd Interface Definition Language*) kojim se definiše programska sprega servisa.

Datoteke koje sadrže AIDL programski kod imaju ekstenziju **.aidl**. Sledeći primer ilustruje jedan primer AIDL opisa programske sprege servisa.

AIDL interfejs se kreira u okviru **project/app** foldera kao što je prikazano na slici - Slika 2.



Slika 2 - Kreiranje AIDL datoteke

U interfejsu se definišu metode koje će predstavljati funkcionalnosti Binder klase. Primer

implementacije:

```
interface IBinderExample {

   int getValue();

   void setValue(int value);

}
```

Nakon kreiranja ovog interfejsa potrebno je prevesti projekat, a zatim se uveriti da se na putanji:

app/build/generated/aidl_source_output_dir/debug/out/imepaketa[1]

nalazi izgenerisan source file aidl interfejsa koji je kreiran u prethodnom koraku. Da bi se videla ova putanja potrebno je odabrati **Project** pregled aplikacije. U ovoj datoteci treba da se nalazi apstraktna klasa **Stub** koja sadrži metodu *asInterface* pomoću koje kreiramo objekat Binder-a.

Nakon uspešnog kreiranja aidl interfejsa, kreira se klasa koja predstavlja Binder. Ova klasa nasleđuje apstraktnu klasu Stub iz prethodno generisane datoteke:

```
public class BinderExample extends IBinderExample
```

Ova klasa mora da implementira metode definisane u interfejsu.

Klasa servisa

Kada se završi implementacija metoda interfejsa potrebno je kreirati servis koji se u *onBind* metodi povezuje sa prethodno kreiranim Binder-om.

[1] U zavisnosti od verzije Android Studio alata, ova putanja može da se razlikuje.

```
public class BindService extends Service {

   private BinderExample mBinderExample = null;

    @Override

    public IBinder onBind(Intent intent) {

        if (mBinderExample == null) {

            mBinderExample = new BinderExample();

        }

        return mBinderExample;

    }

    @Override

    public boolean onUnbind(Intent intent) {

        mBinderExample.stop();

        return super.onUnbind(intent);

    }

}
```

Komunikacija sa Binder-om

U aktivnosti u kojoj se pozivaju metode Binder-a potrebno je implementirati interface *ServiceConnection* i njegove metode *onServiceConnected* i *onServiceDisconnected*.

```
    public void onServiceConnected(ComponentName componentName, IBinder
iBinder) {

        Log.d(LOG_TAG, "onServiceConnected");

        mService = IBinderExample.Stub.asInterface(iBinder);

        try {

            mService.setCallback(new CallbackExample());

        } catch (RemoteException e) {

            Log.e(LOG_TAG, "setCallback failed", e);

        }

    }

    @Override

    public void onServiceDisconnected(ComponentName componentName) {

        Log.d(LOG_TAG, "onServiceDisconnected");

        mService = null;

    }
```