MOSIS - Laboratorijska vežba 4

Na prošloj vežbi je aplikacija prevedena u *SingleActivityApplication* koja koristi jedan (glavni) *Activity* i u njemu više fragmenata kojima su implementirani različiti ekrani aplikacije. Implementiran je i ViewModel podataka prema preporučenoj MVVM arhitekturi i podaci su povezani sa implementiranim fragmentima. Kako bi bila podržana osnovna funkcionalnost aplikacije, pored dodavanja novih omiljenih mesta, potrebno je omogućiti i njihovo pregledanje kao promena njihovih vrednosti.

Ciljevi ove vežbe su:

1. Zabrana dodavanja praznog omiljenog mesta.
2. Dodavanje klika na stavku u *ListView* komponenti.
3. Dodavanje fragmenta za prikaz karakteristika omiljenog mesta.
4. Omogućavanje editovanja omiljenog mesta korišćenjem postojećeg fragmenta za dodavanje novog mesta.

1. Zabrana dodavanja praznog omiljenog mesta.
    a. Trenutno je moguće dodati novu lokaciju čak i ako su oba polja za unos prazna. Da bi ovo bilo onemogućeno, potrebno je zabraniti korisniku klik na dugme za dodavanje ukoliko nikakav tekst nije unet u polje za ime lokacije.
    b. U *onViewCreated* funkciji postaviti da dugme sa id-jem *editmyplace_finished_button* inicijalno bude neaktivno.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val addButton: Button = requireView().findViewById<Button>(R.id.editmyplace_finished_button)
    addButton.isEnabled = false
    addButton.setOnClickListener{ it: View!
        val editName: EditText = requireView().findViewById<EditText>(R.id.editmyplace_name_edit)
```

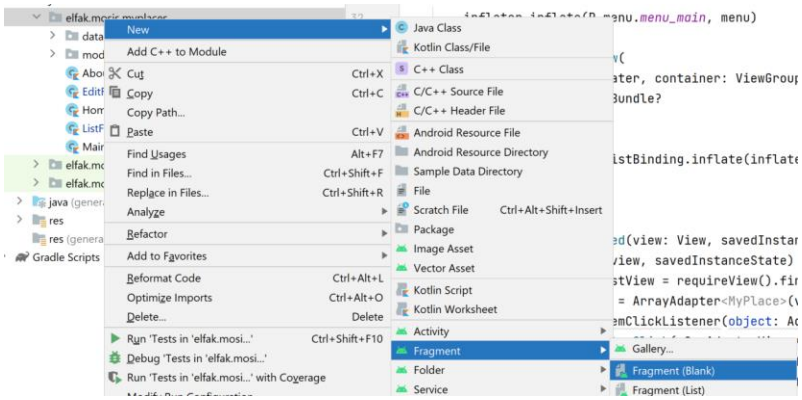
    c. Zatim pomeriti pribavljanje *EditText* instance sa id-jem *editmyplace_name_edit* van *setOnClickListenerMetoda* I dodati mu TextWatcher kako bi pratili promene nad unetim tekstom. Ako je unet bilo kakav tekst, potrebno je omogućiti klik na dugme *Add*.

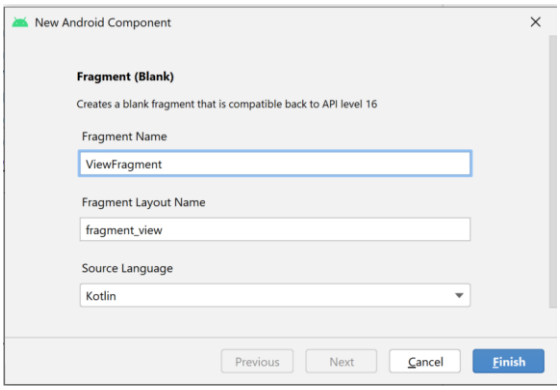
```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val editName: EditText = requireView().findViewById<EditText>(R.id.editmyplace_name_edit)
    val addButton: Button = requireView().findViewById<Button>(R.id.editmyplace_finished_button)
    addButton.isEnabled = false
    editName.addTextChangedListener(object : TextWatcher {
        override fun afterTextChanged(s: Editable?) {
            addButton.isEnabled=(editName.text.length>0)
        }
        override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {
        }
        override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
        }
    })
    addButton.setOnClickListener{ it: View!
        val name: String = editName.text.toString()
        val editDesc: EditText = requireView().findViewById<EditText>(R.id.editmyplace_desc_edit)
        val desc: String = editDesc.text.toString()
        myPlacesViewModel.addPlace( MyPlace(name, desc))
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
    val cancelButton: Button = requireView().findViewById<Button>(R.id.editmyplace_cancel_button)
    cancelButton.setOnClickListener { it: View!
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }

}
```

2. Dodavanje klika na stavku u *ListView* komponenti.
    a. Potrebno je dodati i metod koji će nakon klika na element liste prikazati *Toast* poruku u kojoj piše ime omiljenog mesta. Za ovo je potrebno postaviti *OnItemClickListener* na instancu *ListView*-a unutar *ListFragment.kt* klase.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<MyPlace>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
    myPlacesList.setOnItemClickListener(object: AdapterView.OnItemClickListener {
        override fun onItemClick(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
            var myPlace:MyPlace = p0?.adapter?.getItem(p2) as MyPlace
            Toast.makeText(view.context, myPlace.toString(), Toast.LENGTH_SHORT).show()
        }
    })
}
```

3. Dodavanje fragmenta za prikaz karakteristika omiljenog mesta.
    a. Dodati novi fragment u projekat, pod nazivom *ViewFragment.kt*





b. Počistiti sadržaj novokreiranog fragmenta kao u prethodnoj laboratorijskoj vežbi.

```kotlin
package elfak.mosis.myplaces

import ...

class ViewFragment : Fragment() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_view, container, attachToRoot: false)
    }
}
```

c. Dodati korisnički interfejs u *fragment_view.xml* tako da sadrži po jedan *TextView* za labele i po jedan za ime i opis omiljenog mesta kao jedan *Button* kojim se vraćamo u pozivajući fragment.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ViewFragment"
    android:orientation="vertical">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Name:"
        />
    <TextView
        android:id="@+id/viewmyplace_name_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Description:"
        />
    <TextView
        android:id="@+id/viewmyplace_desc_text"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        />
    <Button
        android:id="@+id/viewmyplace_finished_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/viewmyplace_finised_label"
        />
</LinearLayout>
```

d. Dodati i odgovarajući string resurs za dugme koje sklanja *ViewFragment*.

```
    <string name="editmyplace_desc_label">Description:</string>
    <string name="editmyplace_cancel_label">Cancel</string>
    <string name="editmyplace_add_label">Add</string>
    <string name="viewmyplace_finised_label">Close</string>
</resources>
```

e.

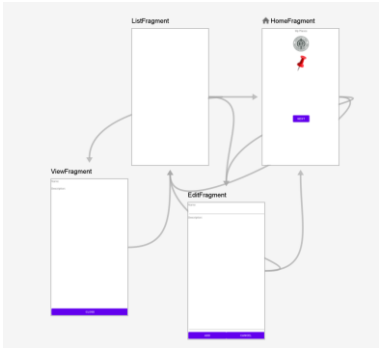
f. Potrebno je prikazati sadržaj omiljenog mesta, na koje je kliknuto u *ViewFragment*-u. Prvi korak u tom pravcu je dodavanje izabranog omiljenog mesta u *ViewModel*. Ovim će biti omogućeno da drugi fragment (*ViewFragment*) može kroz deljeni *ViewModel* da preuzme podatke o selektovanom omiljenom mestu. Dodati promenljivu *selected* u *ViewModel* ali takvu da može biti eksplicitno postavljena na *null* vrednost. Ovo je potrebno kako bi prilikom navigacije iz *ViewFragment*-a moglo deselektovati omiljeno mesto.

```
package elfak.mosis.myplaces.model

import ...

class MyPlacesViewModel: ViewModel() {
    var myPlacesList: ArrayList<MyPlace> = ArrayList<MyPlace>()
    fun addPlace(place: MyPlace){
        myPlacesList.add(place);
    }
    var selected: MyPlace? = null
}
```

g. Povezati tranzicije u *nav_graph.xml* fajlu tako da je iz *ListFragment*-a moguće preći u *ViewFragment* i vratiti se nazad.

```
    <fragment
        android:id="@+id/ListFragment"
        android:name="elfak.mosis.myplaces.ListFragment"
        android:label="My Places"
        tools:layout="@layout/fragment_list">
        <action
            android:id="@+id/action_ListFragment_to_HomeFragment"
            app:destination="@id/HomeFragment" />
        <action
            android:id="@+id/action_ListFragment_to_EditFragment"
            app:destination="@id/EditFragment" />
        <action
            android:id="@+id/action_ListFragment_to_ViewFragment"
            app:destination="@id/ViewFragment" />
    </fragment>
    <fragment
        android:id="@+id/EditFragment"
        android:name="elfak.mosis.myplaces.EditFragment"
        android:label="Edit My Place"
        tools:layout="@layout/fragment_edit">
        <action
            android:id="@+id/action_EditFragment_to_HomeFragment"
            app:destination="@id/HomeFragment" />
        <action
            android:id="@+id/action_EditFragment_to_ListFragment"
            app:destination="@id/ListFragment" />
    </fragment>
    <fragment
        android:id="@+id/ViewFragment"
        android:name="elfak.mosis.myplaces.ViewFragment"
        android:label="View My Place"
        tools:layout="@layout/fragment_view">
        <action
            android:id="@+id/action_ViewFragment_to_ListFragment"
            app:destination="@id/ListFragment" />
    </fragment>
</navigation>
```

h. Proveriti da li je implementacija u dizajneru grafa odgovarajuća.



i. Dodati instancu deljenog *ViewModel*-a u *ViewFragment.kt* i inicijalizovati korisnički interfejs na osnovu izabranog omiljenog mesta koje se nalazi u *selected* atributu. Alternativni način rada sa komponentama korisničkog interfejsa je preko *binding*-a. Na osnovu sadržaja korisničkog interfejsa fragmenta, kreira se posebna klasa preko koje je moguće pristupati elementima korisničkog interfejsa bez pojedinačnog pretraživanja tih komponenti na osnovu njihovih id-jeva.

```
ragment.kt × ViewFragment.kt × ListFragment.kt × nav_graph.xml × fragment_view.xml × themes.xml
package elfak.mosis.myplaces

import ...

class ViewFragment : Fragment() {
    private val myPlacesViewModel: MyPlacesViewModel by activityViewModels()
    private var _binding: FragmentViewBinding? = null
    private val binding get() = _binding!!
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        _binding = FragmentViewBinding.inflate(inflater, container,  attachToParent: false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.viewmyplaceNameText.text=myPlacesViewModel.selected?.name
        binding.viewmyplaceDescText.text=myPlacesViewModel.selected?.description
        myPlacesViewModel.selected = null
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

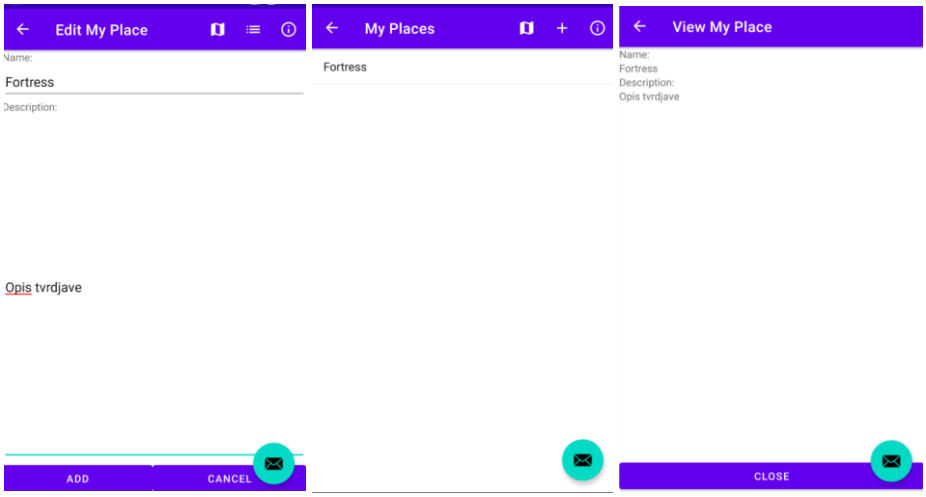
j. U *ListFragment*-u izmeniti sadržaj metode *onItemClick* tako da postavlja u *ViewModel*-u odgovarajuće izabrano omiljeno mesto a zatim vrši tranziciju na *ViewFragment*.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<MyPlace>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
    myPlacesList.setOnItemClickListener(object: AdapterView.OnItemClickListener {
        override fun onItemClick(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
            var myPlace:MyPlace = p0?.adapter?.getItem(p2) as MyPlace
            myPlacesViewModel.selected = myPlace
            view.findNavController().navigate(R.id.action_ListFragment_to_ViewFragment)
        }
    })
}
```

k. Poslednji preostali korak je dodavanje klika na *Close* dugme *ViewFragment*-a. Moguće je opet iskoristiti binding.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding.viewmyplaceNameText.text=myPlacesViewModel.selected?.name
    binding.viewmyplaceDescText.text=myPlacesViewModel.selected?.description
    binding.viewmyplaceFinishedButton.setOnClickListener {  it: View!
        myPlacesViewModel.selected = null
        findNavController().navigate(R.id.action_ViewFragment_to_ListFragment)
    }
}
```

l. Probati aplikaciju!



m.

n. Obzirom da korisnik može kliknuti na strelicu u nazad, potrebno je dodati deselekciju u metod *onDestroyView*

```
override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
    myPlacesViewModel.selected = null
}
```

4. Omogućavanje editovanja omiljenog mesta korišćenjem postojećeg fragmenta za dodavanje novog mesta.
    a. Potrebno je dodati konteksni meni *ListView* komponente i funkciju koja obrađuje dodati kontekstni meni u skladu sa rednim brojem stavke menija. Svaka od stavki treba da poziva tranziciju na određeni fragment.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<MyPlace>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
    myPlacesList.setOnItemClickListener(object: AdapterView.OnItemClickListener {
        override fun onItemClick(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
            var myPlace:MyPlace = p0?.adapter?.getItem(p2) as MyPlace
            myPlacesViewModel.selected = myPlace
            view.findNavController().navigate(R.id.action_ListFragment_to_ViewFragment)
        }
    })
    myPlacesList.setOnCreateContextMenuListener(object : View.OnCreateContextMenuListener {
        override fun onCreateContextMenu(menu: ContextMenu, v: View?, menuInfo: ContextMenuInfo) {
            val info = menuInfo as AdapterContextMenuInfo
            val myPlace:MyPlace = myPlacesViewModel.myPlacesList[info.position]
            menu.setHeaderTitle(myPlace.name)
            menu.add(0, 1, 1, "View place")
            menu.add(0, 2, 2, "Edit place")
            menu.add(0, 3, 3, "Delete place")
        }
    })
}
```

```
override fun onContextItemSelected(item: MenuItem): Boolean {
    val info = item.menuInfo as AdapterContextMenuInfo
    if (item.itemId === 1) {
        myPlacesViewModel.selected = myPlacesViewModel.myPlacesList[info.position]
        this.findNavController().navigate(R.id.action_ListFragment_to_ViewFragment)
    } else if (item.itemId === 2) {
        myPlacesViewModel.selected = myPlacesViewModel.myPlacesList[info.position]
        this.findNavController().navigate(R.id.action_ListFragment_to_EditFragment)
    } else if (item.itemId === 3) {
        Toast.makeText(this.context,  text: "Delete item", Toast.LENGTH_SHORT).show()
    }
    return super.onContextItemSelected(item)
}
```

b. Potrebno je dodati i logiku u *EditFragment* koja, u zavisnosti od toga da li postoji selektovano omiljeno mesto, omogućava funkcionalnost za editovanje ili dodavanje novog omiljenog mesta. Dodati i string resurs za *Save* labelu dugmeta

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val editName: EditText = requireView().findViewById<EditText>(R.id.editmyplace_name_edit)
    val editDesc: EditText = requireView().findViewById<EditText>(R.id.editmyplace_desc_edit)
    if(myPlacesViewModel.selected!=null){
        editName.setText(myPlacesViewModel.selected?.name)
        editDesc.setText(myPlacesViewModel.selected?.description)
    }
    val addButton: Button = requireView().findViewById<Button>(R.id.editmyplace_finished_button)
    addButton.isEnabled = false
    if(myPlacesViewModel.selected!=null)
        addButton.setText(R.string.editmyplace_save_label)
    editName.addTextChangedListener(object : TextWatcher {
        override fun afterTextChanged(s: Editable?) {
            addButton.isEnabled=(editName.text.length>0)
        }
        override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {
        }
        override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
        }
    })
    addButton.setOnClickListener{ it: View!
        val name: String = editName.text.toString()
        val desc: String = editDesc.text.toString()
        if(myPlacesViewModel.selected!=null){
            myPlacesViewModel.selected?.name = name
            myPlacesViewModel.selected?.description = desc
        }
        else
            myPlacesViewModel.addPlace( MyPlace(name, desc))
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
    val cancelButton: Button = requireView().findViewById<Button>(R.id.editmyplace_cancel_button)
    cancelButton.setOnClickListener { it: View!
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
}
override fun onDestroyView() {
    super.onDestroyView()
    myPlacesViewModel.selected = null
}
```

c. Probati aplikaciju i proveriti da li sve radi kako je očekivano. Pošto je atribut *ViewModel*-a selected zapravo referenca na omiljeno mesto, instancu klase *MyPlace*, to menjanjem preko selected reference menja se sadržaj instance koja se nalazi u listi. Alternativno je bilo moguće samo čuvati indeks izabranog omiljenog mesta.

d. Sakriti prikaz *FloatingActionButton*-a sa *Edit* i *View* fragmenata. Ovo je potrebno uraditi iz *MainActivity* klase tako što će *fab* biti sakriven na pojedinim stranicama. Potrebno je i predefinisati da *fab* prikazuje *EditFragment* u modu u kome se dodaje novo omiljeno mesto.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    setSupportActionBar(binding.toolbar)

    val navController = findNavController(R.id.nav_host_fragment_content_main)
    appBarConfiguration = AppBarConfiguration(navController.graph)
    setupActionBarWithNavController(navController, appBarConfiguration)

    navController.addOnDestinationChangedListener{controller, destination, arguments ->
        if(destination.id == R.id.EditFragment || destination.id == R.id.ViewFragment)
            binding.fab.hide()
        else
            binding.fab.show()
    }

    binding.fab.setOnClickListener { view ->
        if(navController.currentDestination?.id == R.id.HomeFragment)
            navController.navigate(R.id.action_HomeFragment_to_EditFragment)
        else if(navController.currentDestination?.id == R.id.ListFragment)
            navController.navigate(R.id.action_ListFragment_to_EditFragment)
    }
}
```

e. Počistiti kod (samostalno) tako da se prilikom dodavanja novog omiljenog mesta u zaglavlju *EditFragment*-a prikazuje *Add My Place* umesto *Edit My Place*.