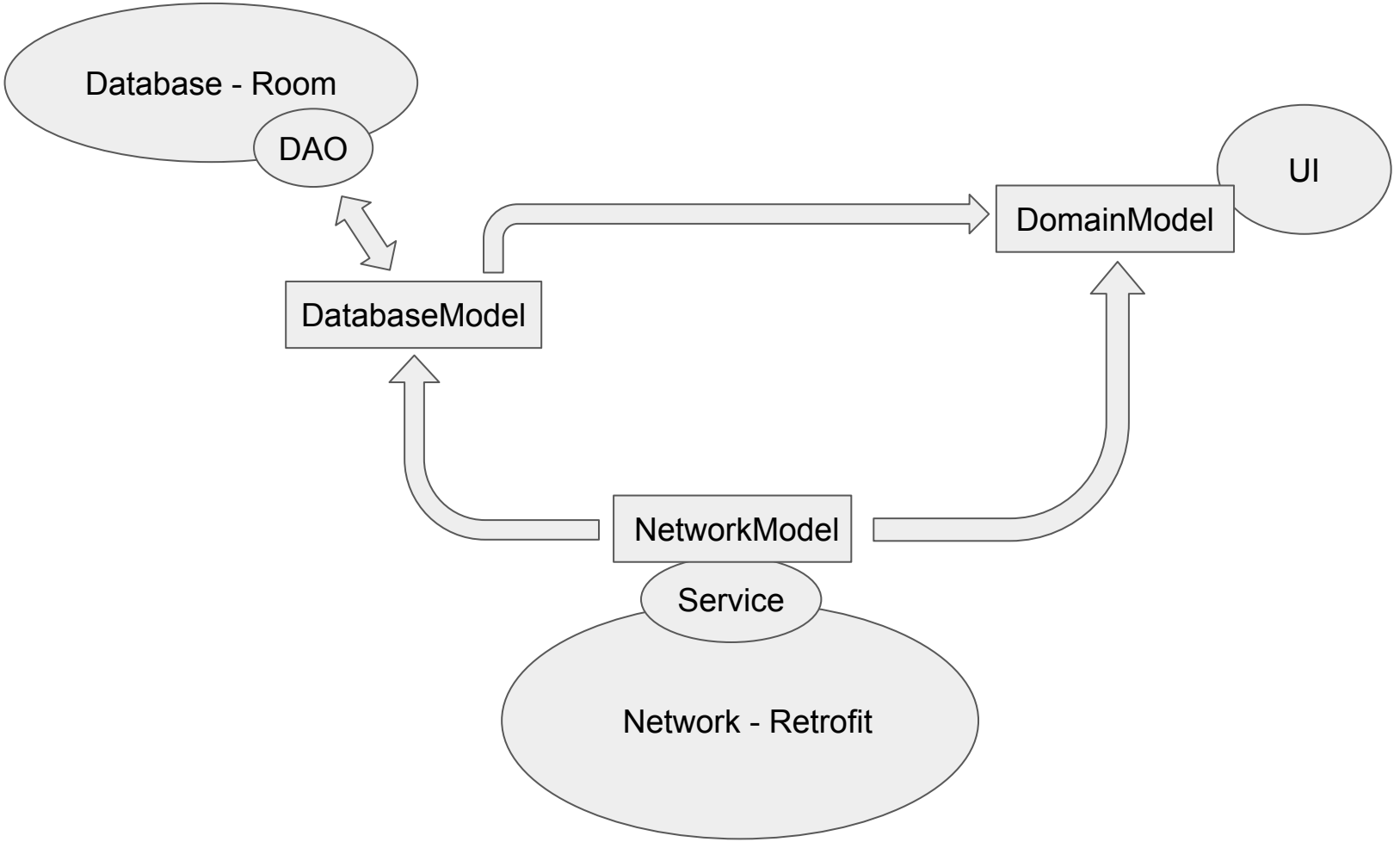
Android

Internet, Datenbank, RecyclerView


Database - Room
DAO
UI
DomainModel
DatabaseModel
NetworkModel
Service
Network - Retrofit

Network - Retrofit

Komponenten

DTO: Data Model

Service: Interface mit HTTP-Annotationen

Retrofit: Implementierung der Service-Interfaces

Berechtigung

```
<uses-permission android:name="android.permission.INTERNET" />
```

DTO zum Datenempfang

```
class NetworkPerson (
    val name: String,
    val username: String
)
```

Container zum Managen der Collection

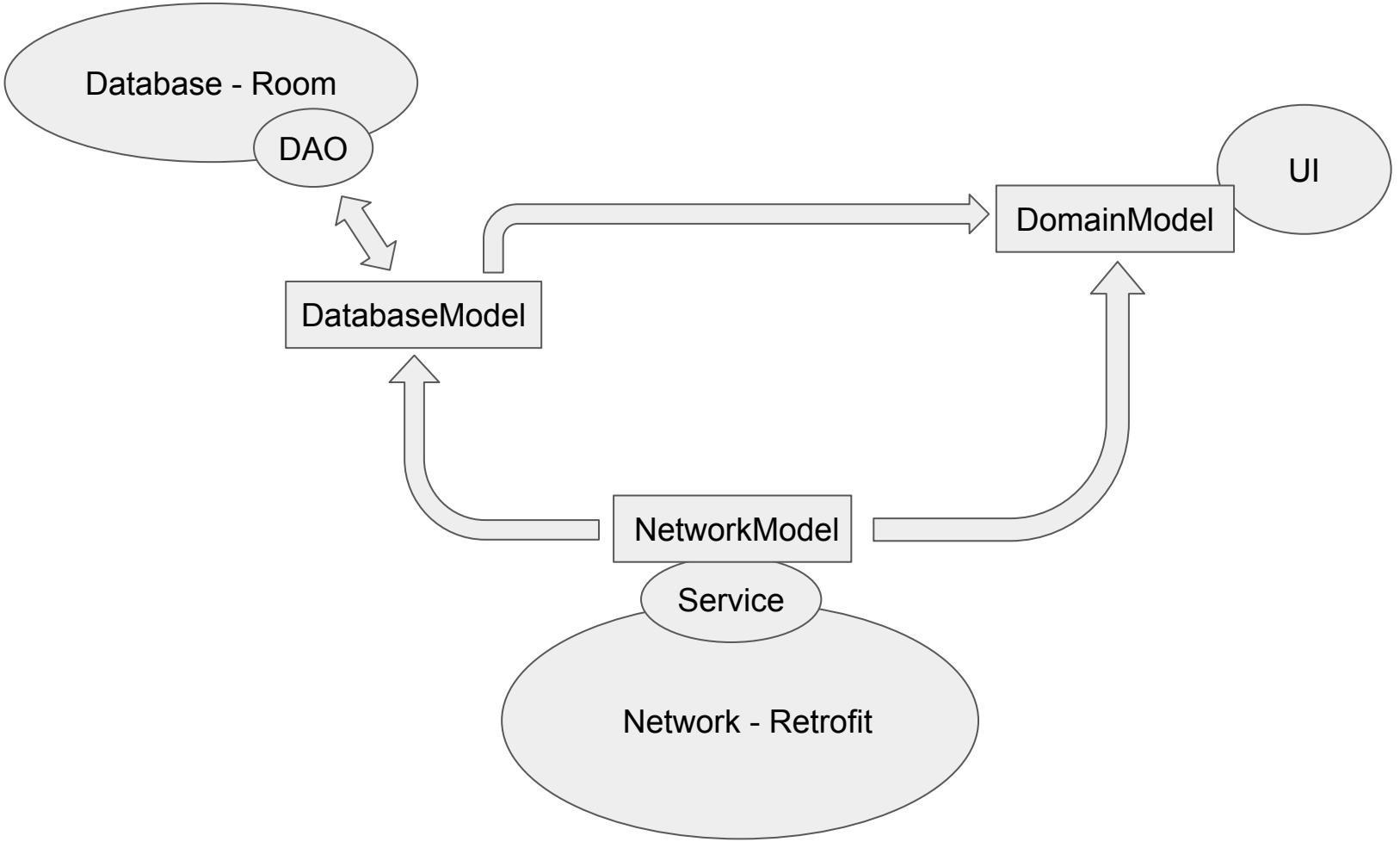
```
class NetworkPersonContainer(
    val persons: List<NetworkPerson>
) {
    fun asDomainModel(): List<Person> {
        return persons.map { it: NetworkPerson
            Person(
                name = it.name,
                username = it.username
            )
        }
    }

    fun asDatabaseModel(): Array<DatabasePerson> {
        return persons.map { it: NetworkPerson
            DatabasePerson(
                name = it.name,
                username = it.username
            )
        }.toTypedArray()
    }
}
```

Service

```
interface PersonService {
    @GET( value: "persons")
    fun getPersons(): Deferred<NetworkPersonContainer>
}
```

Network-Object mit Retrofit und Adaptern

```
object Network {
    private val moshi = Moshi.Builder()
        .add(KotlinJsonAdapterFactory())
        .build()

    private val retrofit = Retrofit.Builder()
        .baseUrl( baseUrl: "https://jsonplaceholder.typicode.com")
        .addConverterFactory(MoshiConverterFactory.create(moshi))
        .addCallAdapterFactory(CoroutineCallAdapterFactory())
        .build()

    val persons = retrofit.create(PersonService::class.java)
}
```


Database - Room
DAO
UI
DomainModel
DatabaseModel
NetworkModel
Service
Network - Retrofit

Database - Room

Komponenten

Entity: Database Table

DAO: Database access

Database: Database holder

Datenbank-Entität und Umwandlung

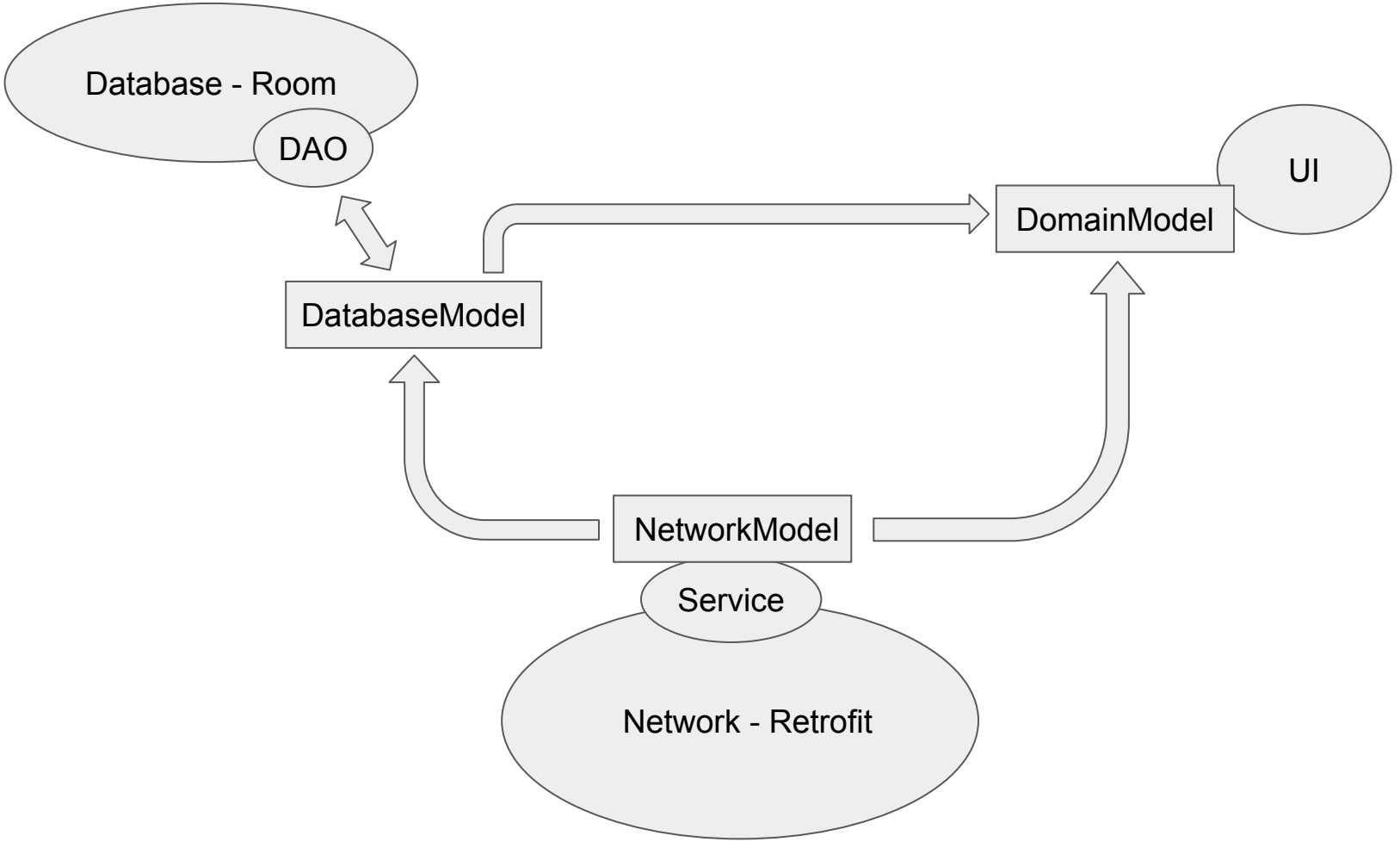
```
@Entity
class DatabasePerson (
    val name: String,
    @PrimaryKey val username: String
)

fun List<DatabasePerson>.asDomainModel(): List<Person> {
    return map { it: DatabasePerson
        Person(
            name = it.name,
            username = it.username
        )
    }
}
```

DAO

```
@Dao
interface PersonDao {
    @Query( value: "select * from databaseperson")
    fun getPersons(): LiveData<List<DatabasePerson>>
    @Insert
    fun insertAll(vararg persons: DatabasePerson)
    @Query( value: "delete from databaseperson")
    fun deleteAll()
}
```

Datenbank-Singleton

```
@Database(entities = [DatabasePerson::class], version = 2)
abstract class PersonDatabase : RoomDatabase() {
    abstract val personDao: PersonDao
}

private lateinit var INSTANCE: PersonDatabase

fun getDatabase(context: Context): PersonDatabase {
    synchronized(PersonDatabase::class.java) {
        if (!::INSTANCE.isInitialized) {
            INSTANCE = Room
                .databaseBuilder(context.applicationContext, PersonDatabase::class.java,
                    name: "persons")
                .fallbackToDestructiveMigration()
                .build()
        }
    }
    return INSTANCE
}
```


Database - Room
DAO
UI
DomainModel
DatabaseModel
NetworkModel
Service
Network - Retrofit

UI

Domain Model

```
class Person (
    val name: String,
    val username: String
)
```

Repository zum Refreshen der Daten

```
class PersonRepository(private val database: PersonDatabase) {

    val persons: LiveData<List<Person>> = Transformations.map(database.personDao.getPersons()) {  it: List<DatabasePerson>!
        it.asDomainModel()
    }

    suspend fun refreshPersons() {
        withContext(Dispatchers.IO) {  this: CoroutineScope
            //val persons = Network.persons.getPersons().await()
            val persons = NetworkPersonContainer(listOf(...))
            database.personDao.insertAll(*persons.asDatabaseModel())
        }
    }

}
```

ViewModel benutzt Repository

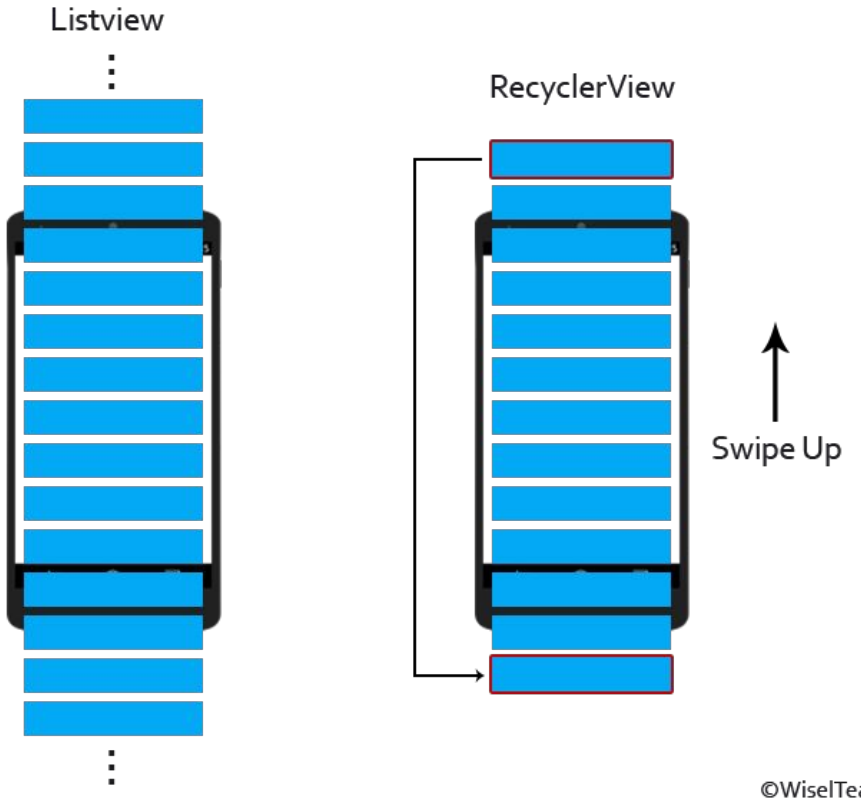
```
class DemoViewModel(application: Application) : AndroidViewModel(application) {

    private val viewModelJob = SupervisorJob()
    private val viewModelScope = CoroutineScope( context: viewModelJob + Dispatchers.Main)
    private val database = getDatabase(application)
    private val personRepository = PersonRepository(database)

    init {
        viewModelScope.launch { this: CoroutineScope
            personRepository.refreshPersons()
        }
    }

    val persons = personRepository.persons

    class Factory(private val app: Application) : ViewModelProvider.Factory {
        override fun <T : ViewModel?> create(modelClass: Class<T>): T {
            if (modelClass.isAssignableFrom(DemoViewModel::class.java)) {
                @Suppress( ...names: "UNCHECKED_CAST")
                return DemoViewModel(app) as T
            }
            throw IllegalArgumentException("Unable to construct viewmodel")
        }
    }

}
```

RecyclerView im Fragment

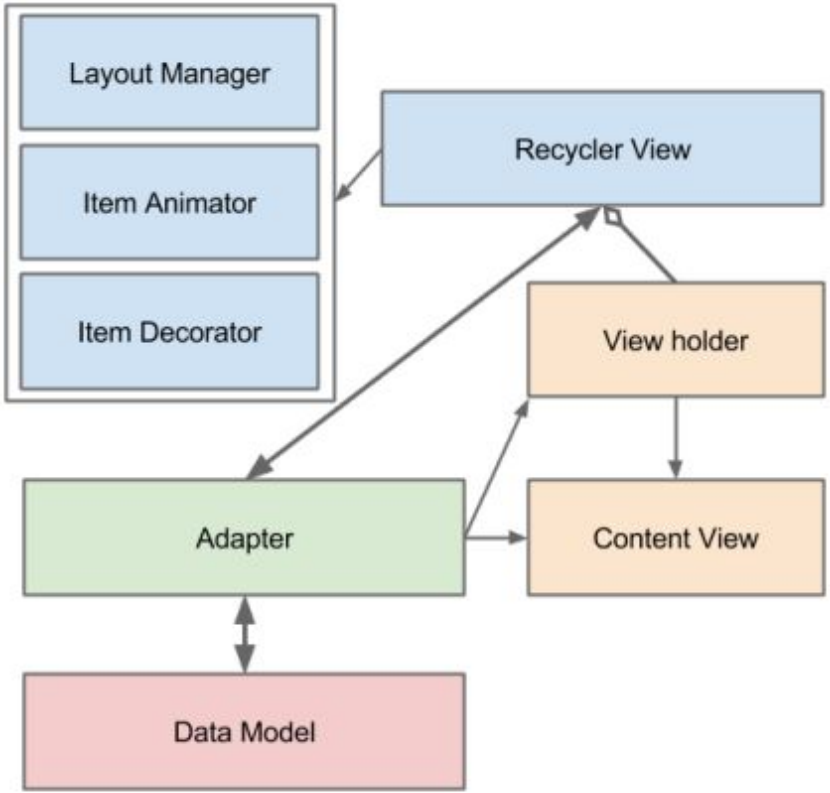
```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".ui.DemoFragment">

    <data>
        <variable
            name="viewModel"
            type="at.htl.demoapplication.viewmodels.DemoViewModel" />
    </data>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycler_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:listitem="@layout/demo_item" />

    </FrameLayout>

</layout>
```

Item der RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="person"
            type="at.htl.demoapplication.domain.Person" />
    </data>

    <com.google.android.material.card.MaterialCardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:cardElevation="5dp"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="16dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <androidx.constraintlayout.widget.Guideline...>

            <androidx.constraintlayout.widget.Guideline...>

            <TextView...>

            <TextView...>

        </androidx.constraintlayout.widget.ConstraintLayout>

    </com.google.android.material.card.MaterialCardView>

</layout>
```



Quelle: https://android.jlelse.eu/understanding-recyclerview-components-part-2-1fd43001a98f



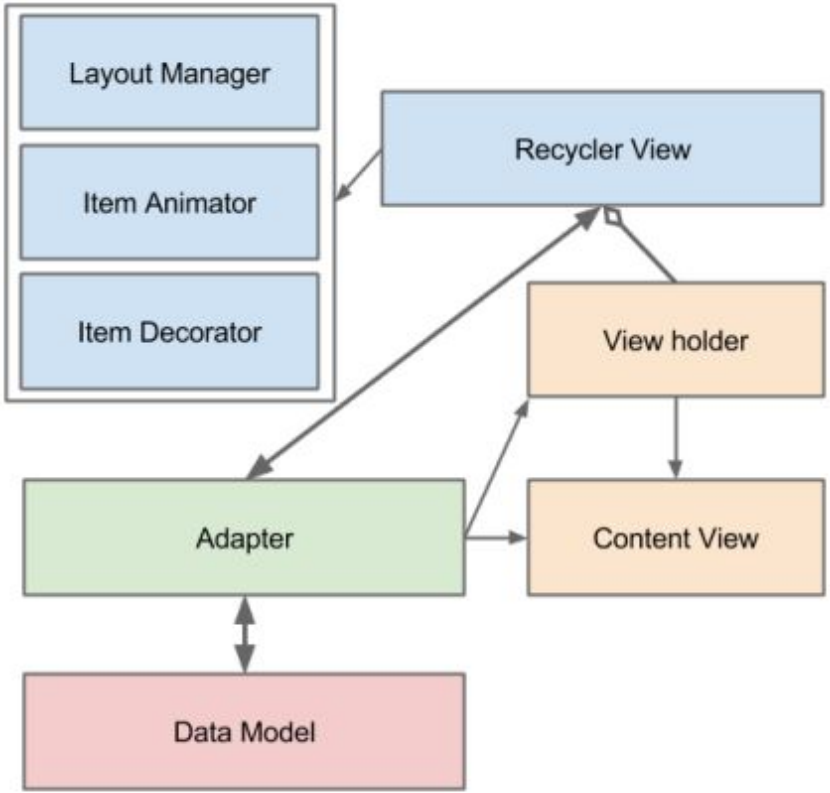
Quelle: https://www.android4dev.com/how-to-display-list-of-data-into-recyclerview-android-kotlin/

ViewHolder

```
class DemoViewHolder(val viewDataBinding: DemoItemBinding)
    : RecyclerView.ViewHolder(viewDataBinding.root) {
    companion object {
        @LayoutRes
        val LAYOUT = R.layout.demo_item
    }
}
```

ViewModelAdapter managed Item-Befüllung

```
class DemoAdapter : RecyclerView.Adapter<DemoViewHolder>() {

    var persons: List<Person> = emptyList()
        set(value) {
            field = value

            notifyDataSetChanged()
        }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): DemoViewHolder {
        val withDataBinding: DemoItemBinding = DataBindingUtil.inflate(
            LayoutInflater.from(parent.context),
            DemoViewHolder.LAYOUT,
            parent,
            attachToParent: false
        )
        return DemoViewHolder(withDataBinding)
    }

    override fun getItemCount() = persons.size

    override fun onBindViewHolder(holder: DemoViewHolder, position: Int) {
        holder.viewDataBinding.also { it: DemoItemBinding
            it.person = persons[position]
        }
    }

}
```



Quelle: https://www.android4dev.com/how-to-display-list-of-data-into-recyclerview-android-kotlin/

Fragment

```
private var viewModelAdapter: DemoAdapter? = null

override fun onActivityCreated(savedInstanceState: Bundle?) {
    super.onActivityCreated(savedInstanceState)
    viewModel.persons.observe(viewLifecycleOwner, Observer<List<Person>> { persons ->
        persons.apply { this: List<Person>!
            viewModelAdapter?.persons = persons
        }
    })
}

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val binding: FragmentDemoBinding = DataBindingUtil.inflate(
        inflater,
        R.layout.fragment_demo,
        container,
        attachToParent: false
    )
    binding.lifecycleOwner = viewLifecycleOwner

    binding.viewModel = viewModel

    viewModelAdapter = DemoAdapter()
    binding.root.findViewById<RecyclerView>(R.id.recycler_view).apply { this: RecyclerView!
        layoutManager = LinearLayoutManager(context)
        adapter = viewModelAdapter
    }

    return binding.root
}
```

Android-Demo

Leanne Graham
Bret

Ervin Howell
Antonette

Clementine Bauch
Samantha

Patricia Lebsack
Karianne

Chelsey Dietrich
Kamren

Mrs. Dennis Schulist
Leopoldo_Corkery

Kurtis Weissnat
Elwyn.Skiles

Nicholas Runolfsdottir V
Maxime_Nienow

Glenna Reichert
Delphine