Uniwersytet Wrocławski

Rafał Lewandków

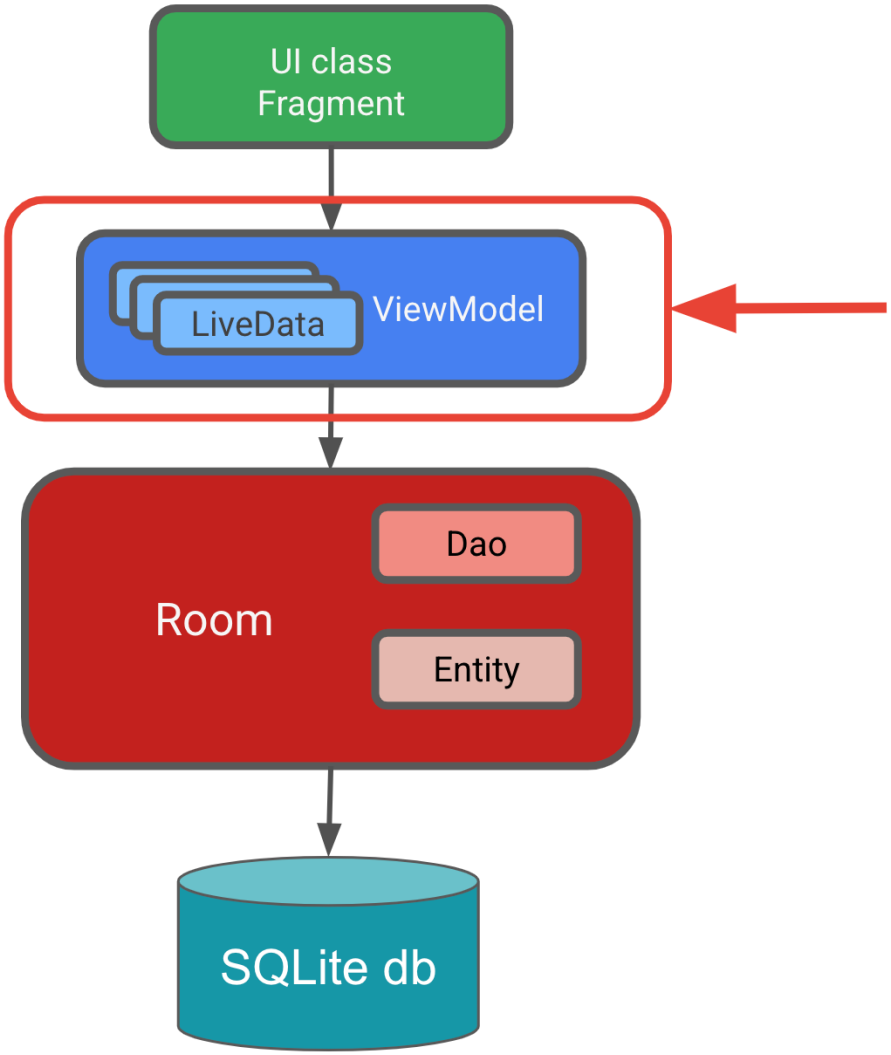
PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

WYKŁAD 11

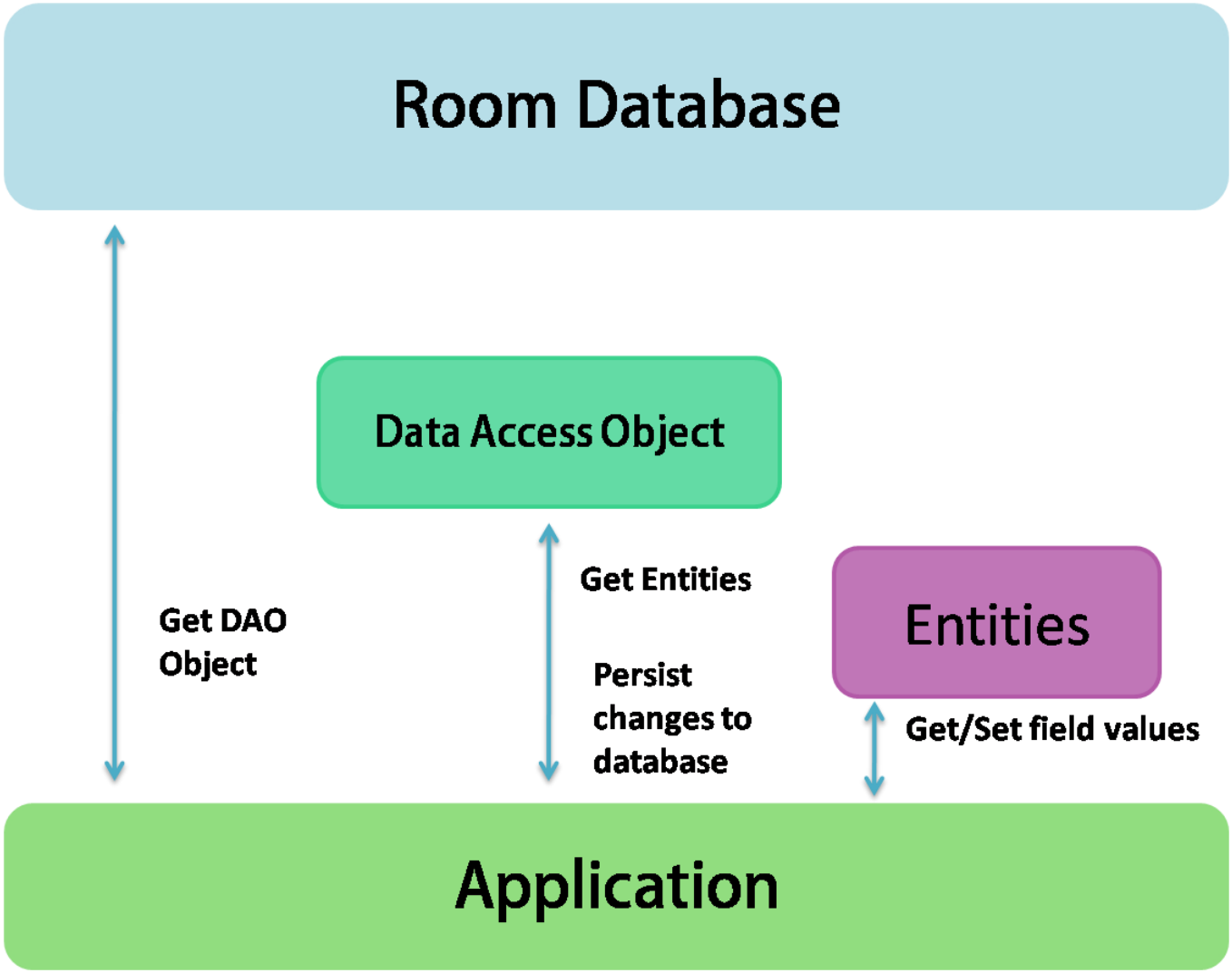
Bazy danych ROOM

- Dao
- Entity
- SQL

ROOM

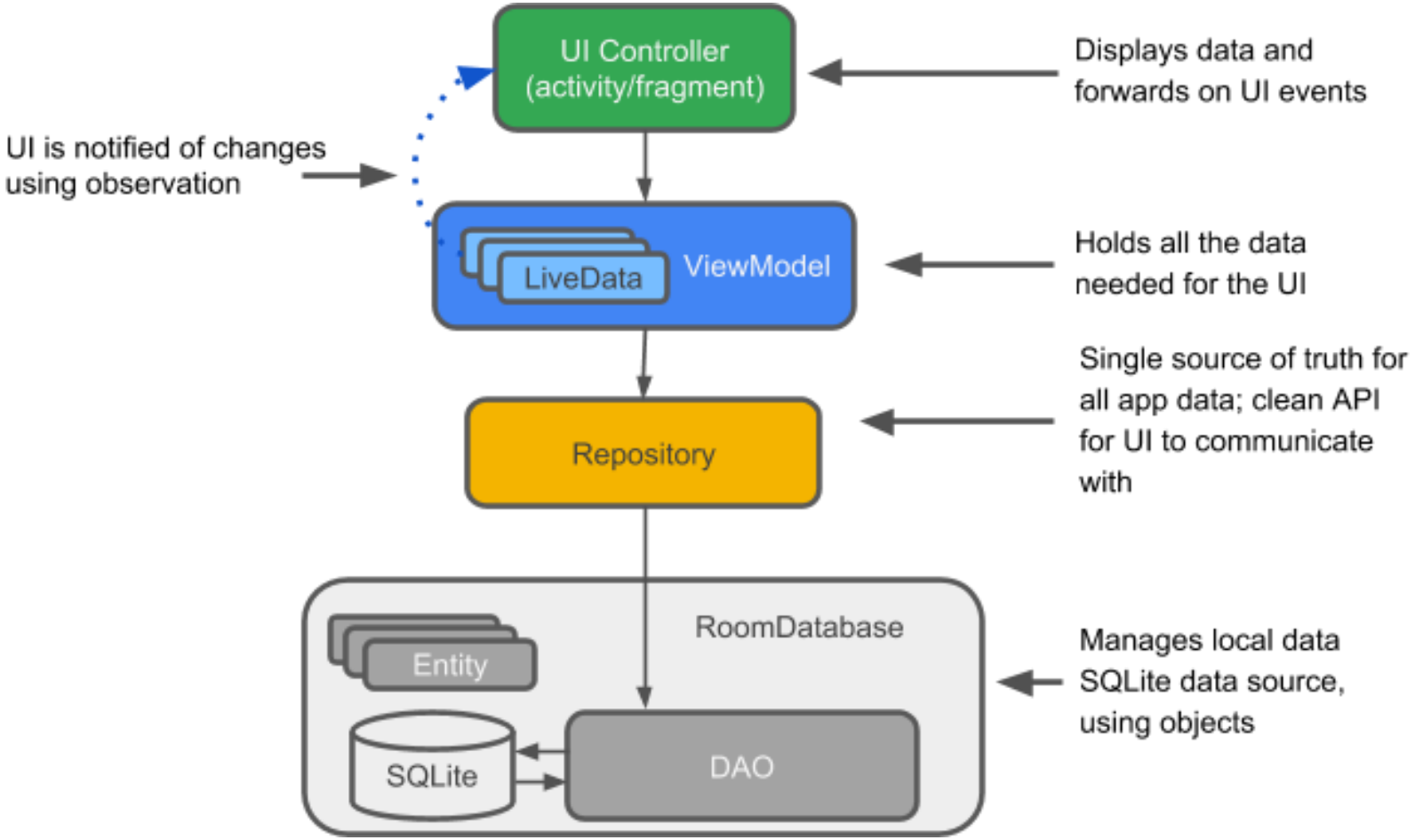

UI class
Fragment
LiveData
ViewModel
Room
Dao
Entity
SQLite db

ROOM

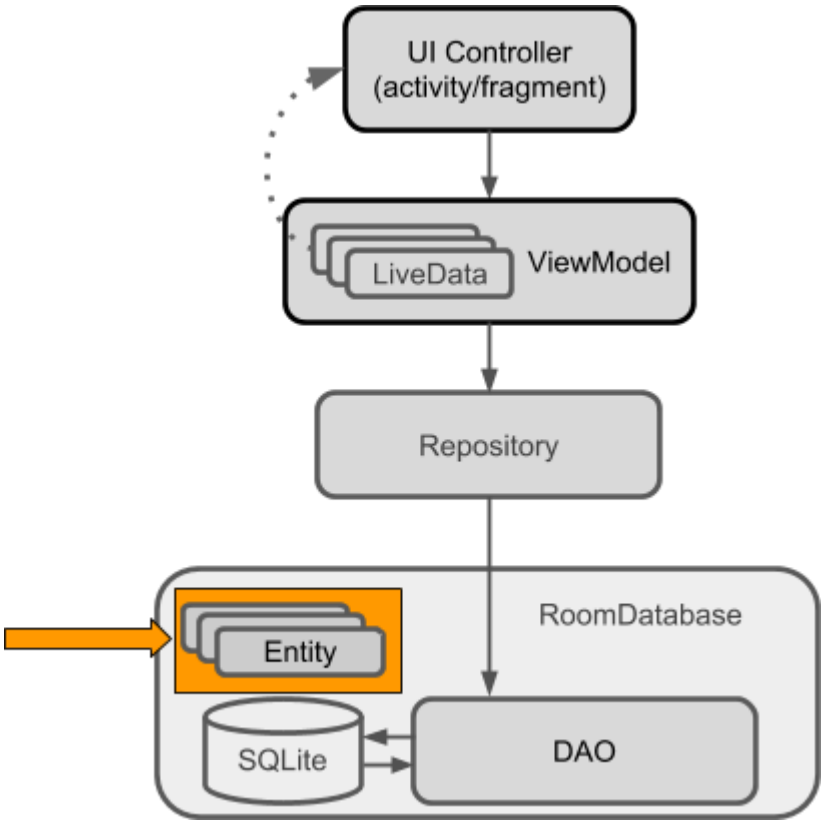


https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40kuldeepmakwana3977%2Froom-database-using-android-kotlin-8cd3cd854b15

ROOM


UI Controller
(activity/fragment)
Displays data and
forwards on UI events
UI is notified of changes
using observation
LiveData
ViewModel
Holds all the data
needed for the UI
Repository
Single source of truth for
all app data; clean API
for UI to communicate
with
RoomDatabase
Entity
SQLite
DAO
Manages local data
SQLite data source,
using objects

Entity



https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html

Uniwersytet Wrocławski

Entity

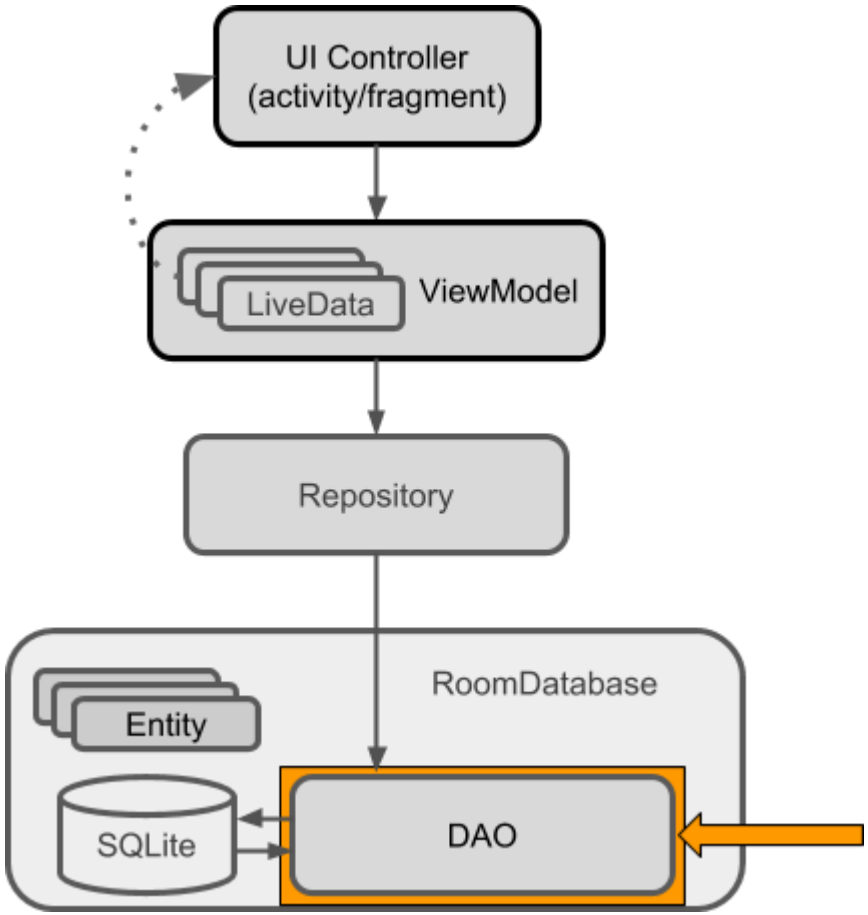
ID	First name	Last name	...
12345	Aleks	Becker	...
12346	Jhansi	Kumar	...

```
@Entity
public class Person {
    @PrimaryKey (autoGenerate=true)
    private int uid;

    @ColumnInfo(name = "first_name")
    private String firstName;

    @ColumnInfo(name = "last_name")
    private String lastName;
```

https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html

Uniwersytet Wrocławski

Dao



https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html

Dao

```
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table ORDER BY lastName ASC, firstName ASC")
    fun getUsers(): Flow<List<User>>

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insert(user: User)

    @Query("DELETE FROM user_table")
    suspend fun deleteAll()
}
```

Implementing Data Access Object

@Dao

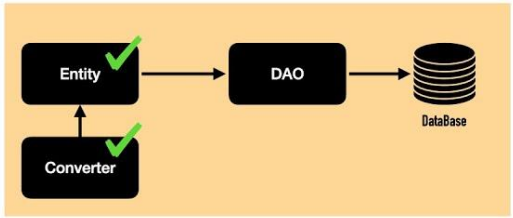
Used to mark a class as Data Access Object (DAO)

DAO abstracts data base operations

Select	@Query	
Insert	@Insert	@Query
Delete	@Delete	@Query
Update	@Update	@Query

DAO annotation must be used with interface of abstract class

For each table / entity there must me separate Dao



https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html

SQLite

Główne cechy SQLite:

- **Działa jako biblioteka dostępna w postaci pliku w aplikacji:** że nie wymaga oddzielnego procesu – aplikacja może bezpośrednio komunikować się z bazą
- **Jest samowystarczalna:** cała baza danych jest przechowywana w jednym pliku – brak konieczności konfigurowania zasobów
- **Transakcyjność:** umżliwia grupowanie operacji bazodanowych jako pojedyncze, atomowe działania
- **Wsparcie dla standardowego SQL:** obsługuje większość standardowego języka SQL

ROOM

Główne składniki biblioteki **ROOM**:

- **Entity:** reprezentuje tabelę w bazie danych. Każda klasa oznaczona adnotacją `@Entity` może być mapowana do jednej tabeli w bazie danych, pola klasy odpowiadają kolumnom tej tabeli

- **DAO:** definiuje metody dostępowe do danych w bazie. Definiujemy za pomocą adnotacji `@Dao`

- **Database:** Klasa bazowa reprezentująca bazę danych. Miejsce gdzie definiujemy wszystkie *encje*, które mają zostać użyte w aplikacji. `Room` automatycznie tworzy implementację bazy danych w oparciu o tą klasę

Uniwersytet Wrocławski

ROOM + MVVM

```
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val firstName: String,
    val lastName: String
)
```

Uniwersytet Wrocławski

ROOM + MVVM

```kotlin
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val firstName: String,
    val lastName: String
)
```

```kotlin
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table ORDER BY lastName ASC, firstName ASC")
    fun getUsers(): Flow<List<User>>

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insert(user: User)

    @Query("DELETE FROM user_table")
    suspend fun deleteAll()
}
```

Uniwersytet Wrocławski

ROOM + MVVM

```kotlin
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val firstName: String,
    val lastName: String
)
```

```kotlin
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table ORDER BY lastName ASC, firstName ASC")
    fun getUsers(): Flow<List<User>>

    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insert(user: User)

    @Query("DELETE FROM user_table")
    suspend fun deleteAll()
}
```

```kotlin
@Database(entities = [User::class], version = 1, exportSchema = false)
abstract class UserDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao

    companion object {
        @Volatile
        private var Instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return Instance ?: synchronized(this) {
                Room.databaseBuilder(context, UserDatabase::class.java, "user_database")
                    .build()
                    .also { Instance = it }
            }
        }
    }
}
```