Understanding Common Android Architectural Patterns

Omri Erez

SOFTWARE ENGINEER

@innovationMaze | https://www.linkedin.com/in/omrierez/

God Class / Object

- Contains a high number of components
- Components are coupled
- A very lengthy class
- Avoid them at all cost

Activity

CoinModel

CryptoCoinEntity

Tracker:
- Activity lifecycle
- Location

bindViews

MyCryptoAdapter

Network Logic for API request

Persist data to local storage

RecyclerView

EntityToModelMapperTask

Runtime permission logic

Read data from local storage

Common Architectural Patterns

MVC: Model - View - Controller

MVP: Model - View - Presenter

MVVM: Model - View - ViewModel

MVC: Model View Controller

MVC

Model

- Represents the data models
- Manages the data’s states
- Includes the business logic of our application
- Often used across different parts of our app

MVC

View

- Essentially it’s our layouts and views
- The way we represent the data
- Renders the user interface

Controller

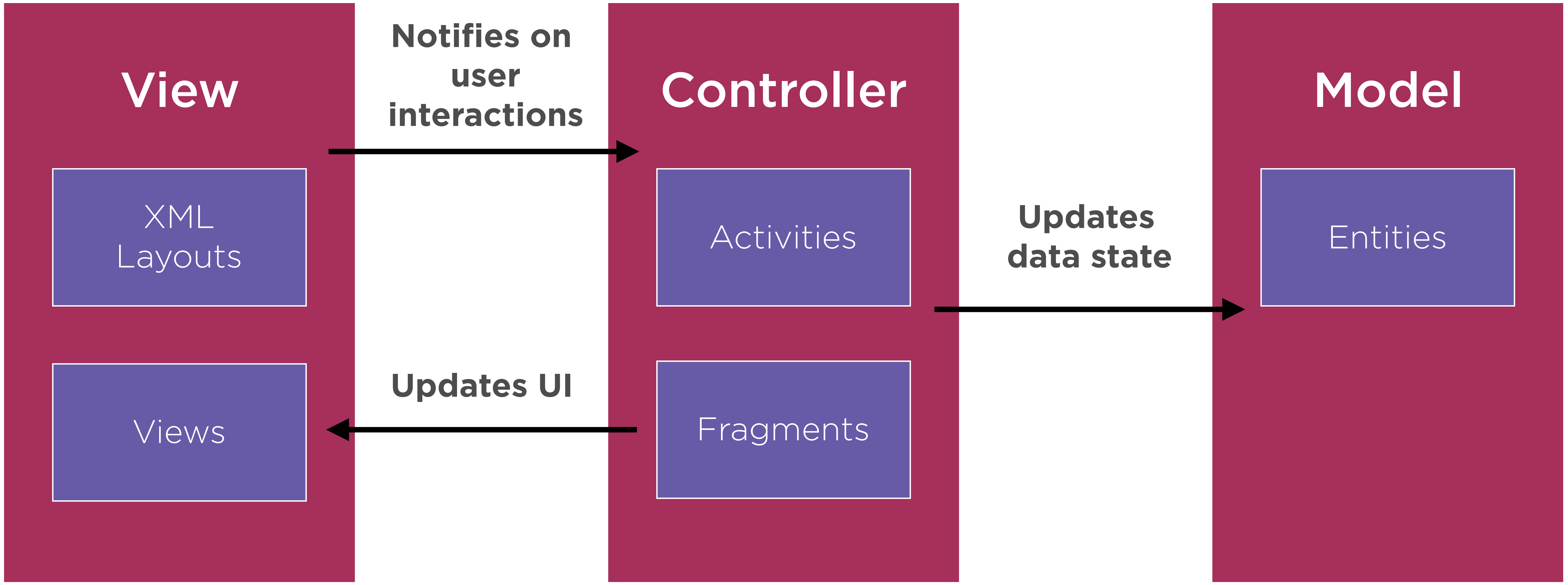
MVC

- Essentially it’s our activities and fragments

- Includes user’s interactions with our app

- The communication channel between our views and models

MVC Diagram


View
XML Layouts
Views
Notifies on user interactions
Updates UI
Controller
Activities
Fragments
Updates data state
Model
Entities

MVP: Model View Presenter

MVP

Model

- Same as in the MVC pattern

MVP

View

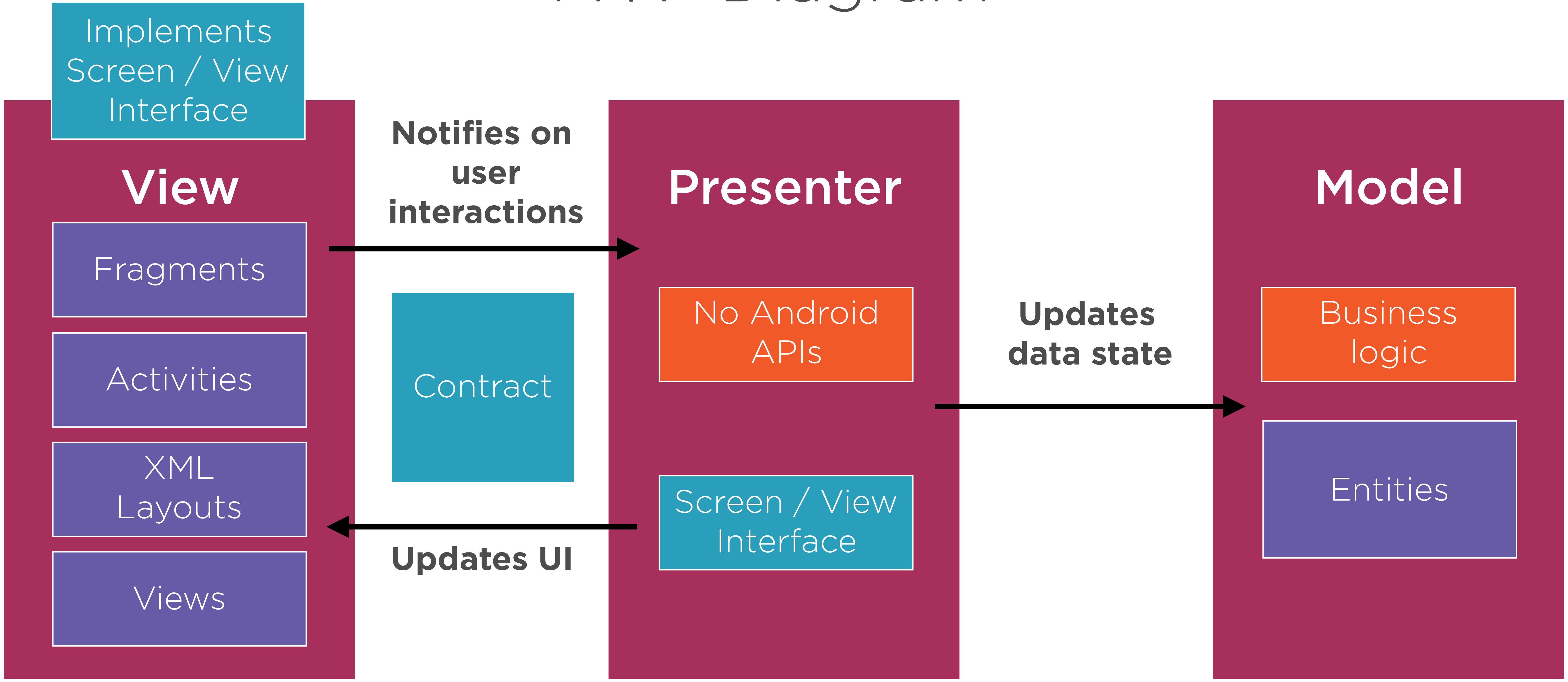
- Our XML Layouts and views
- Our activities and fragments:
  - In the Android world the two are strongly bonded with the views
  - Will implement an interface for the presenters actions

Presenter

MVP

- Has no relation to our views (Unlike MVC)
- Operations are invoked by our view (Activities or fragments)
- Views update is done via the view’s interface

MVP Diagram


Implements Screen / View Interface
View
Fragments
Activities
XML Layouts
Views
Notifies on user interactions
Contract
Updates UI
Presenter
No Android APIs
Screen / View Interface
Updates data state
Model
Business logic
Entities

MVVM: Model View ViewModel

MVVM

- Using the Data Binding library from Google
- Minimize views binding code
- Views bindings logic is implemented in the XML layout

Without Data Binding

```
class CoinViewHolder extends RecyclerView.ViewHolder {

    TextView tvNameAndSymbol;
    TextView tvPriceAndVolume;
    ImageView ivIcon;

    public CoinViewHolder(View itemView) {
        super(itemView);
        tvNameAndSymbol = itemView.findViewById(R.id.tvNameAndSymbol);
        tvPriceAndVolume = itemView.findViewById(R.id.tvPriceAndVolume);
        ivIcon = itemView.findViewById(R.id.ivIcon);
    }
}
```

Without Data Binding

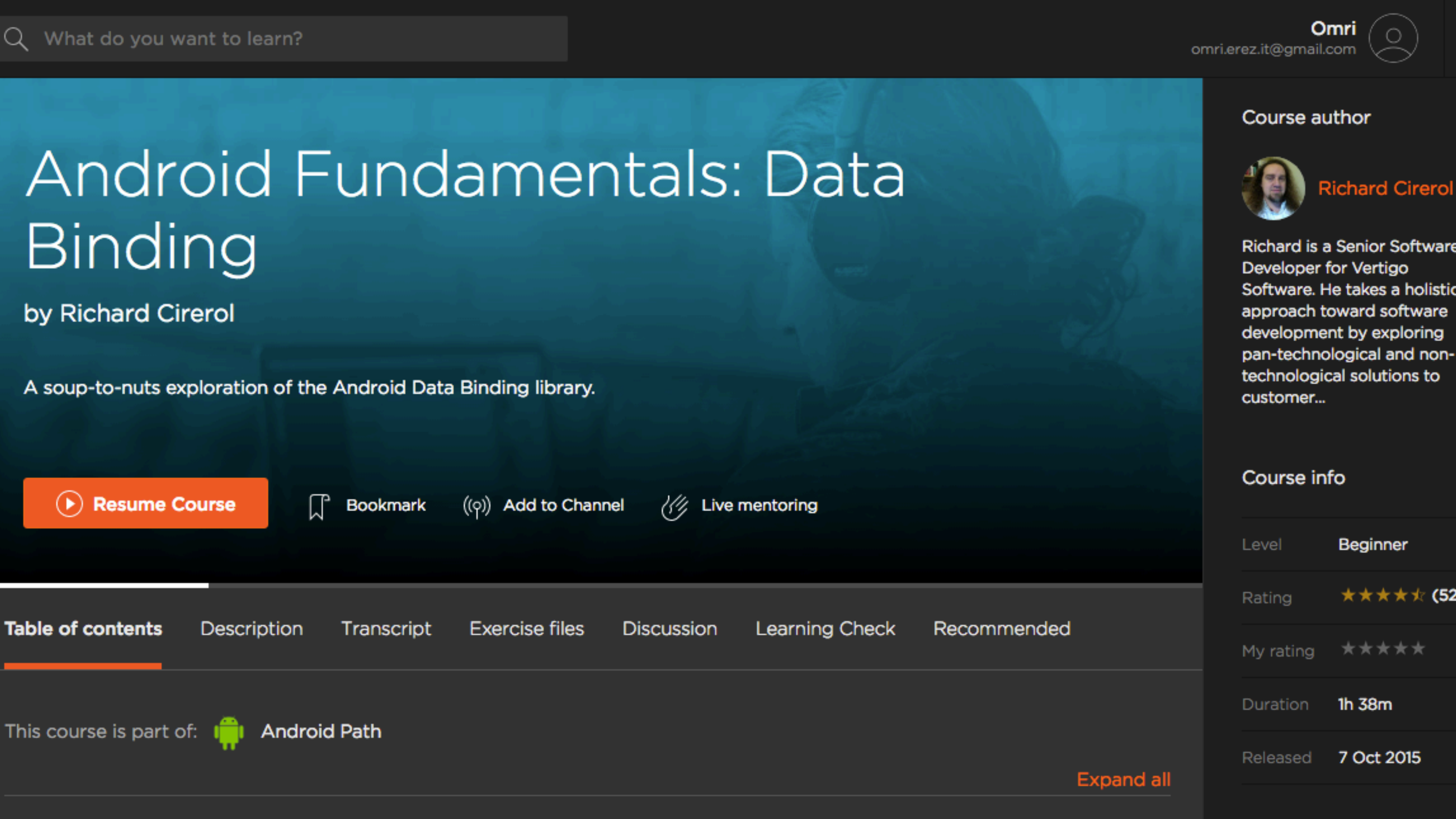
```
@Override
public void onBindViewHolder(CoinViewHolder holder, int position) {

    final CoinModel model = mItems.get(position);
    holder.tvNameAndSymbol.setText(model.name);
    holder.tvPriceAndVolume.setText(model.priceUsd);

}
```

With Data Binding

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable
            name="coin"
            type="com.pluralsight.cryptobam.MainActivity.CoinModel" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:orientation="vertical">
        <TextView
            android:id="@+id/tvNameAndSymbol"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{coin.name}" />
        <TextView
            android:id="@+id/tvPriceAndVolume"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{coin.priceUsd}"
            />
    </LinearLayout>
</layout>
```



What do you want to learn?

Omri
omri.erez.it@gmail.com

Android Fundamentals: Data Binding

by Richard Cirerol

A soup-to-nuts exploration of the Android Data Binding library.

Resume Course Bookmark Add to Channel Live mentoring

Table of contents Description Transcript Exercise files Discussion Learning Check Recommended

This course is part of: Android Path

Expand all

Course author

Richard Cirerol

Richard is a Senior Software Developer for Vertigo Software. He takes a holistic approach toward software development by exploring pan-technological and non-technological solutions to customer...

Course info

Level	Beginner
Rating	★★★★☆ (52
My rating	★★★★★
Duration	1h 38m
Released	7 Oct 2015

MVVM

Model and View

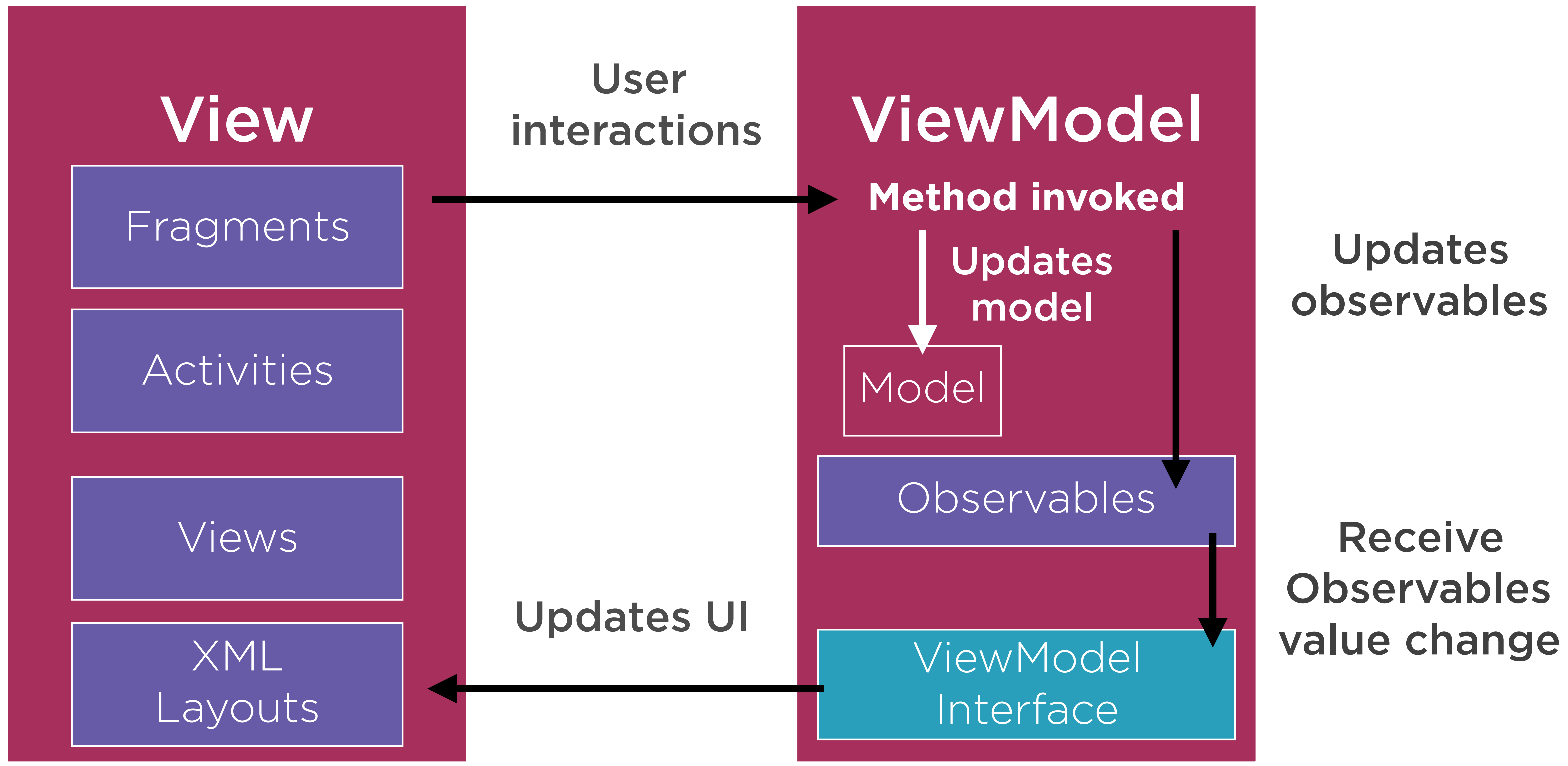
- Same as in the MVP pattern

MVVM

ViewModel

- Contains the Model
- Uses observable variables for update values
- On each value update, the relevant views will be updated

MVVM Diagram



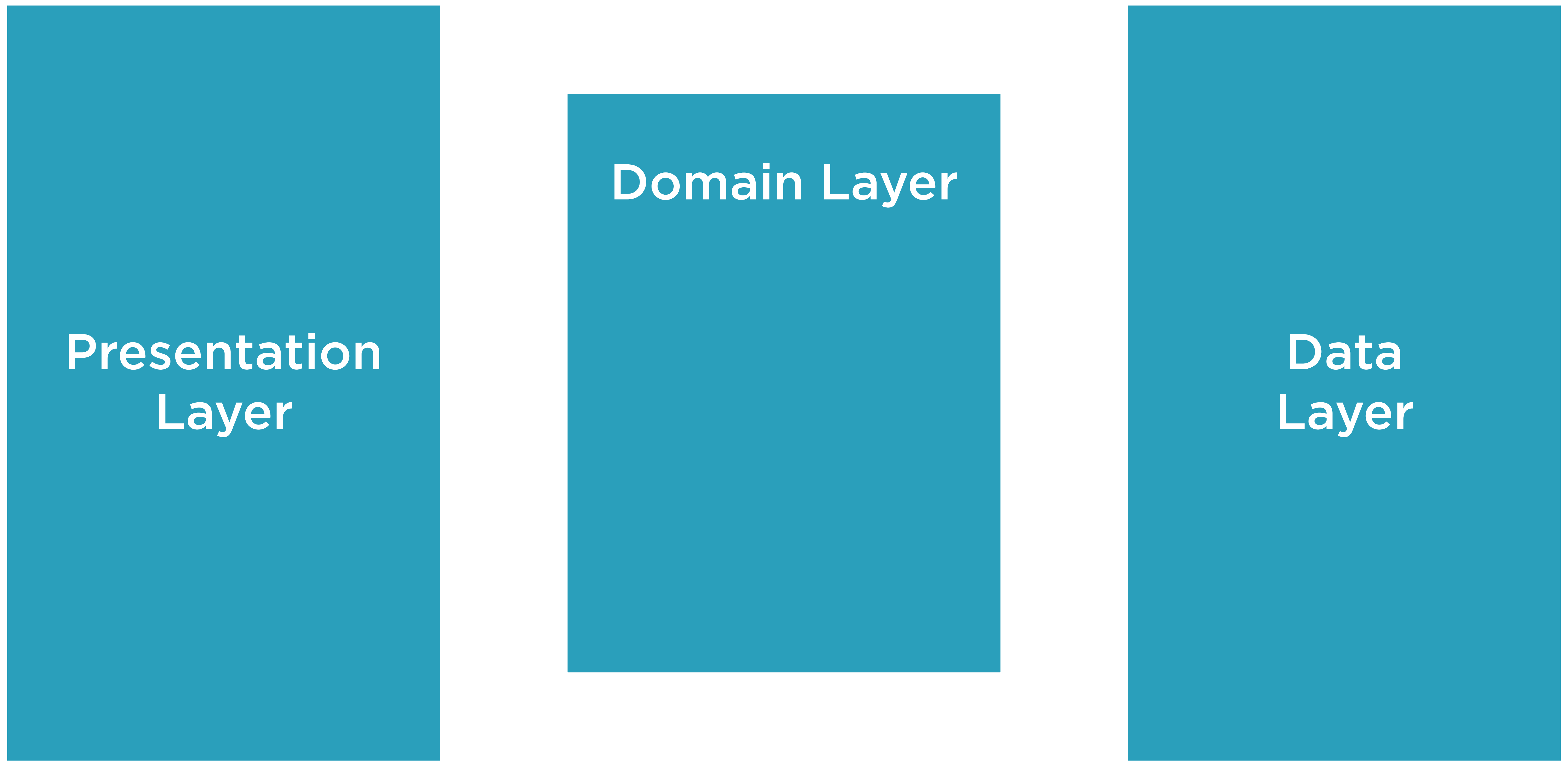
Comparison

Primitive Types and Objects Size

PATTERN	Dependency on Android APIS	XML Complexity	Unit testability	Modular & SRP
MVC Controller	High	Low	Difficult	No
MVP Presenter	Low	Low	Good	Yes
MVVM ViewModel	Low - No dependency	Medium	Great	Yes

The Clean Architecture

The Clean Architecture



Presentation Layer

- Android components: activities, services, fragments etc.
- Custom views
- Presenters

Domain Layer

- Entities

- Screen interfaces

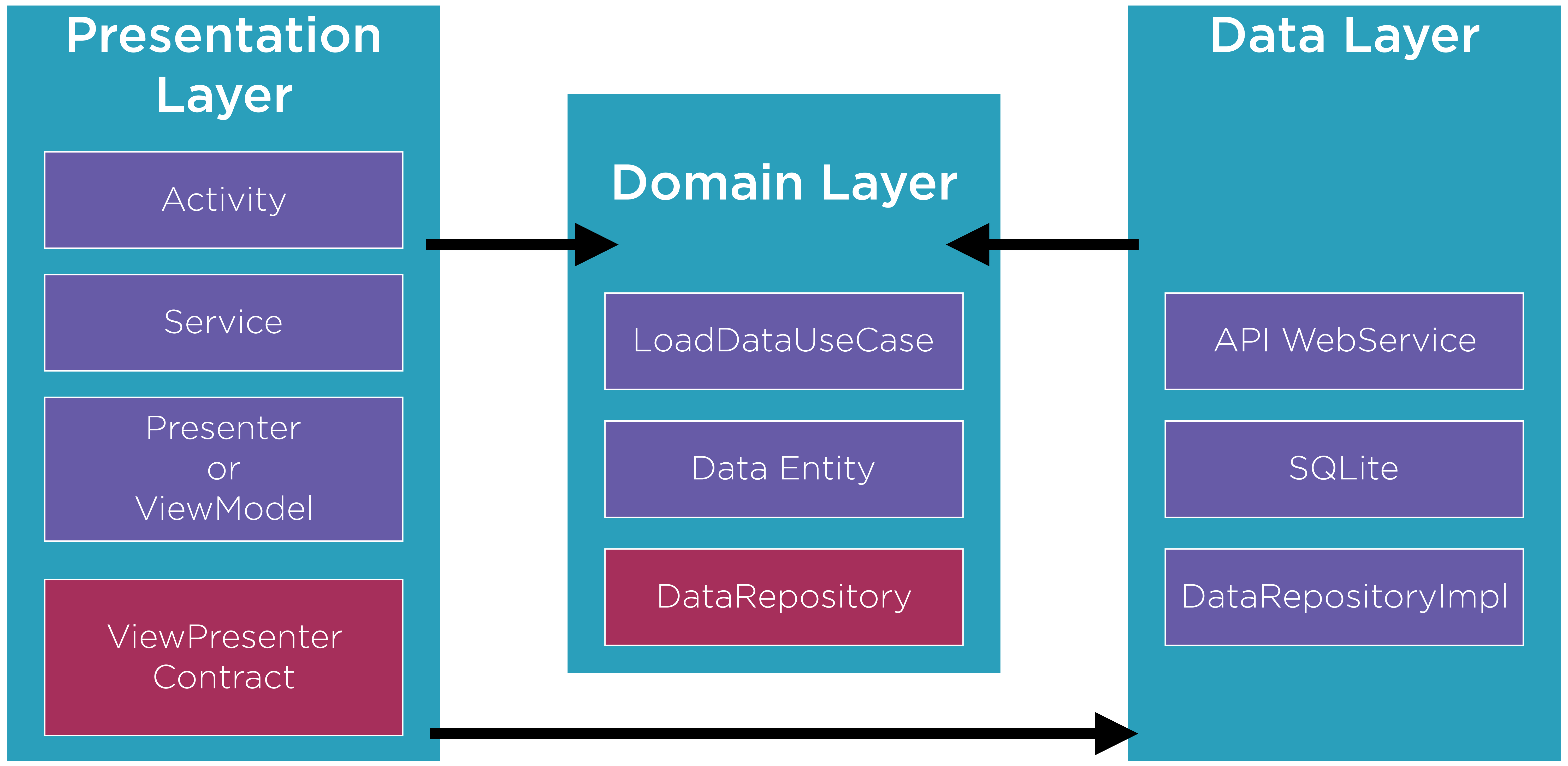
- Use cases

- Pure Java / Kotlin - Android free

Data Layer

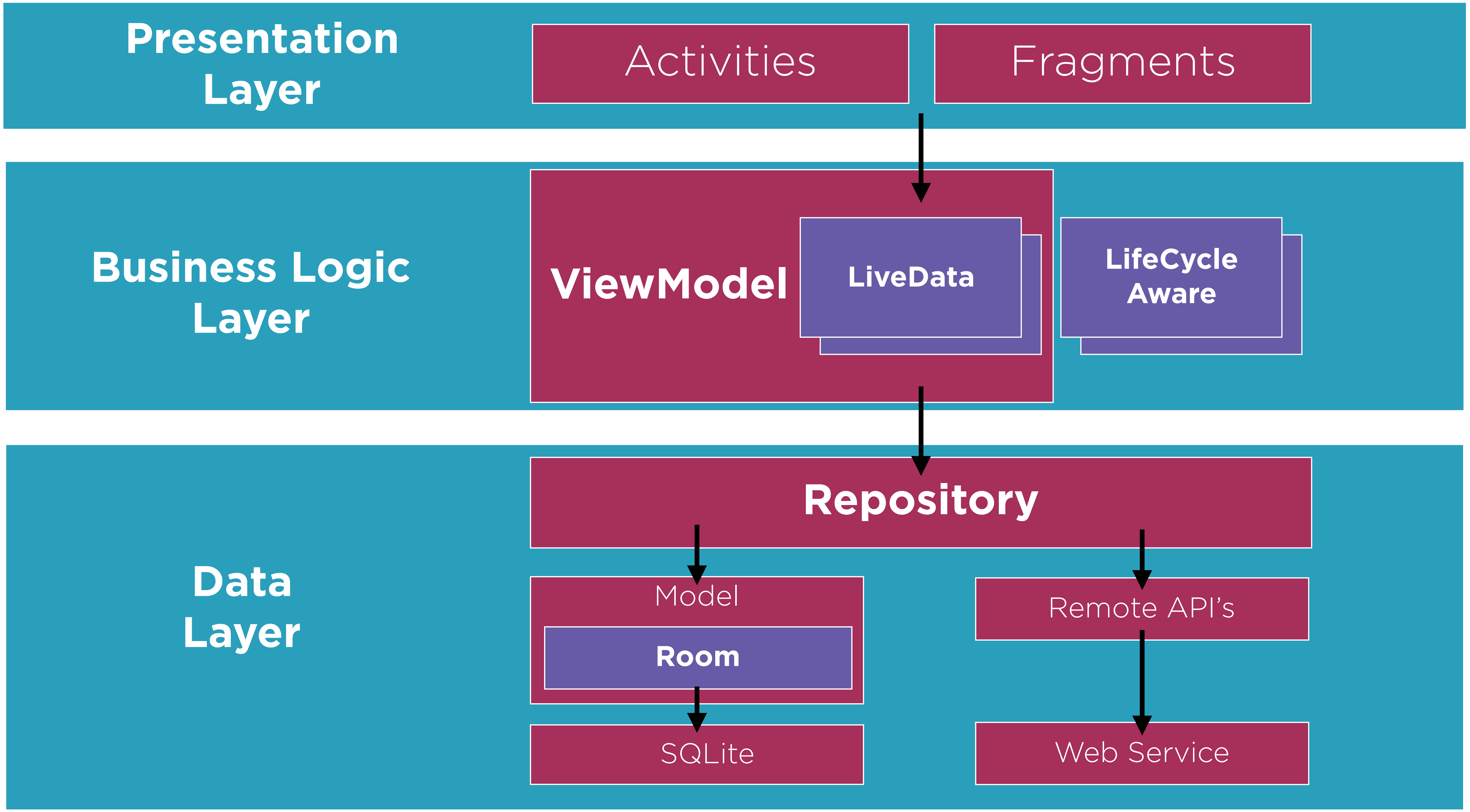
- External Apis access
- Local storage components
- Memory & disk caches

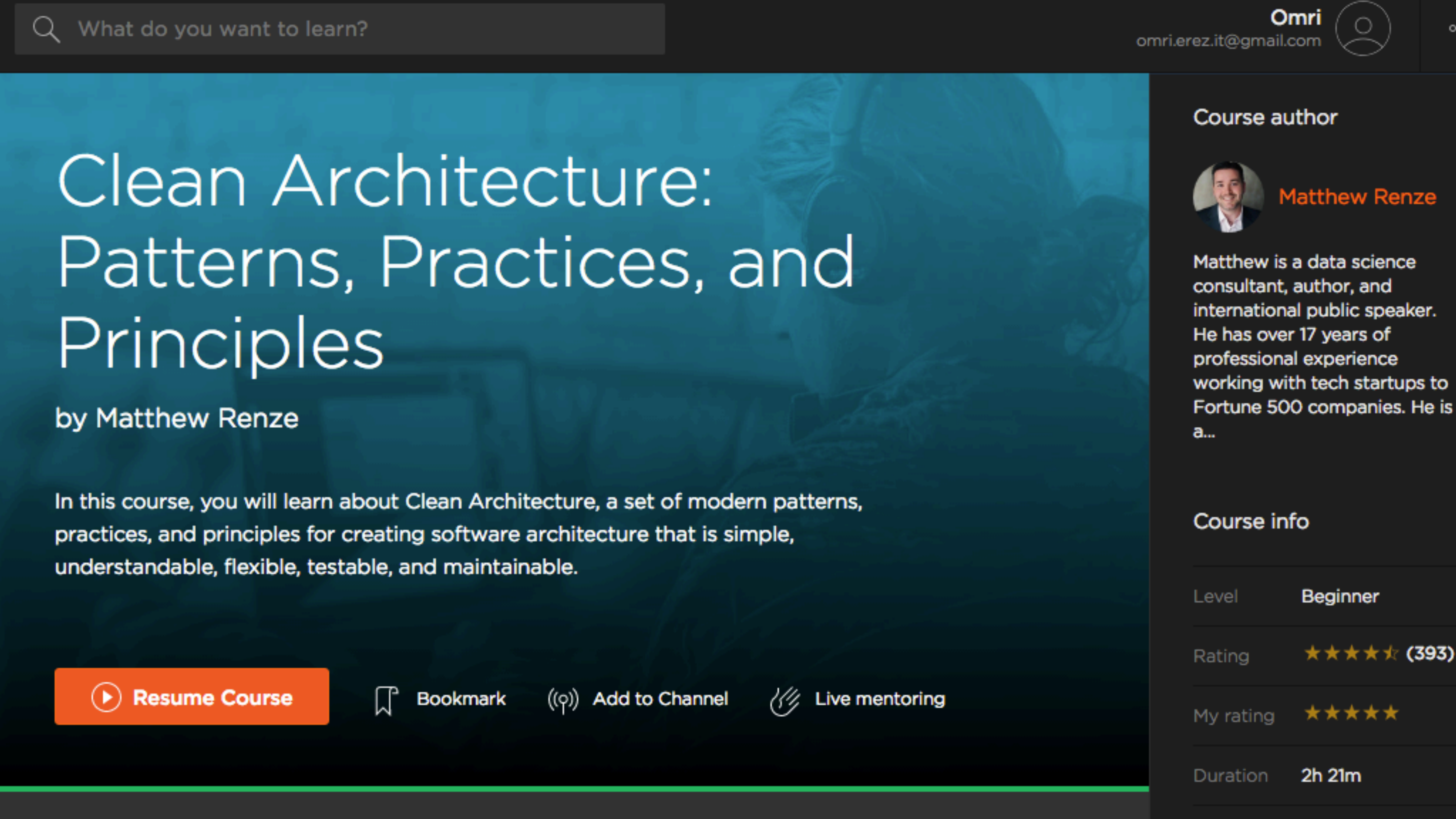
The Clean Architecture



Advantages

- Modular approach
- Each module can be tested separately
- Domain layer can be reused for other JVM applications


Presentation Layer
Activities
Fragments
Business Logic Layer
ViewModel
LiveData
LifeCycle Aware
Data Layer
Repository
Model
Room
SQLite
Remote API's
Web Service



What do you want to learn?

Omri
omri.erez.it@gmail.com

Clean Architecture: Patterns, Practices, and Principles

by Matthew Renze

In this course, you will learn about Clean Architecture, a set of modern patterns, practices, and principles for creating software architecture that is simple, understandable, flexible, testable, and maintainable.

Course author

Matthew Renze

Matthew is a data science consultant, author, and international public speaker. He has over 17 years of professional experience working with tech startups to Fortune 500 companies. He is a...

Course info

Level	Beginner
Rating	★★★★⯪ (393)
My rating	★★★★★
Duration	2h 21m

Summary

Common architectural patterns

- God classes
- Differences between MVC MVP and MVVM
- The Clean architecture

Understanding Common Android Architectural Patterns

Omri Erez

SOFTWARE ENGINEER

@innovationMaze | https://www.linkedin.com/in/omrierez/

God Class / Object

- Contains a high number of components

- Components are coupled

- A very lengthy class

- Avoid them at all cost

Activity

CoinModel

CryptoCoinEntity

Tracker:
- Activity lifecycle
- Location

bindViews

MyCryptoAdapter

Network Logic for API request

Persist data to local storage

RecyclerView

EntityToModelMapperTask

Runtime permission logic

Read data from local storage

Common Architectural Patterns

MVC: Model - View - Controller

MVP: Model - View - Presenter

MVVM: Model - View - ViewModel

MVC: Model View Controller

MVC

Model

- Represents the data models
- Manages the data’s states
- Includes the business logic of our application
- Often used across different parts of our app

MVC

View

- Essentially it’s our layouts and views

- The way we represent the data

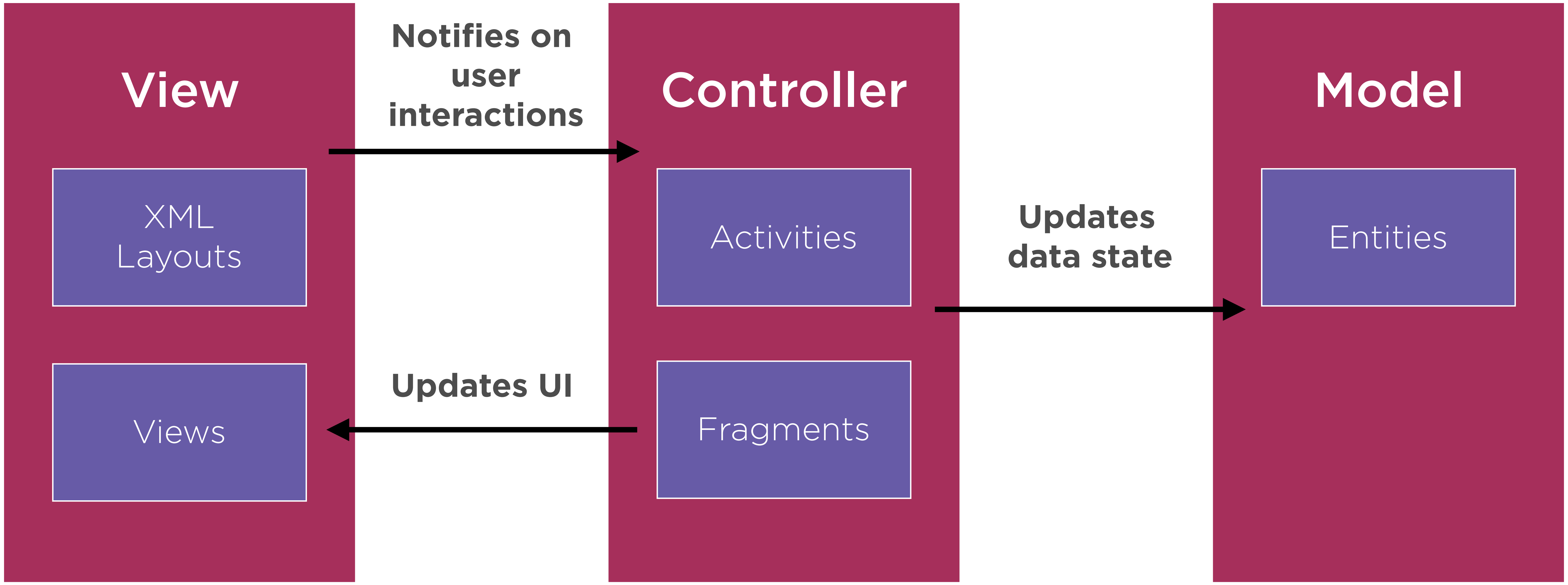
- Renders the user interface

MVC

Controller

- Essentially it’s our activities and fragments
- Includes user’s interactions with our app
- The communication channel between our views and models

MVC Diagram


View
XML Layouts
Views
Notifies on user interactions
Updates UI
Controller
Activities
Fragments
Updates data state
Model
Entities

MVP: Model View Presenter

MVP

Model

- Same as in the MVC pattern

MVP

View

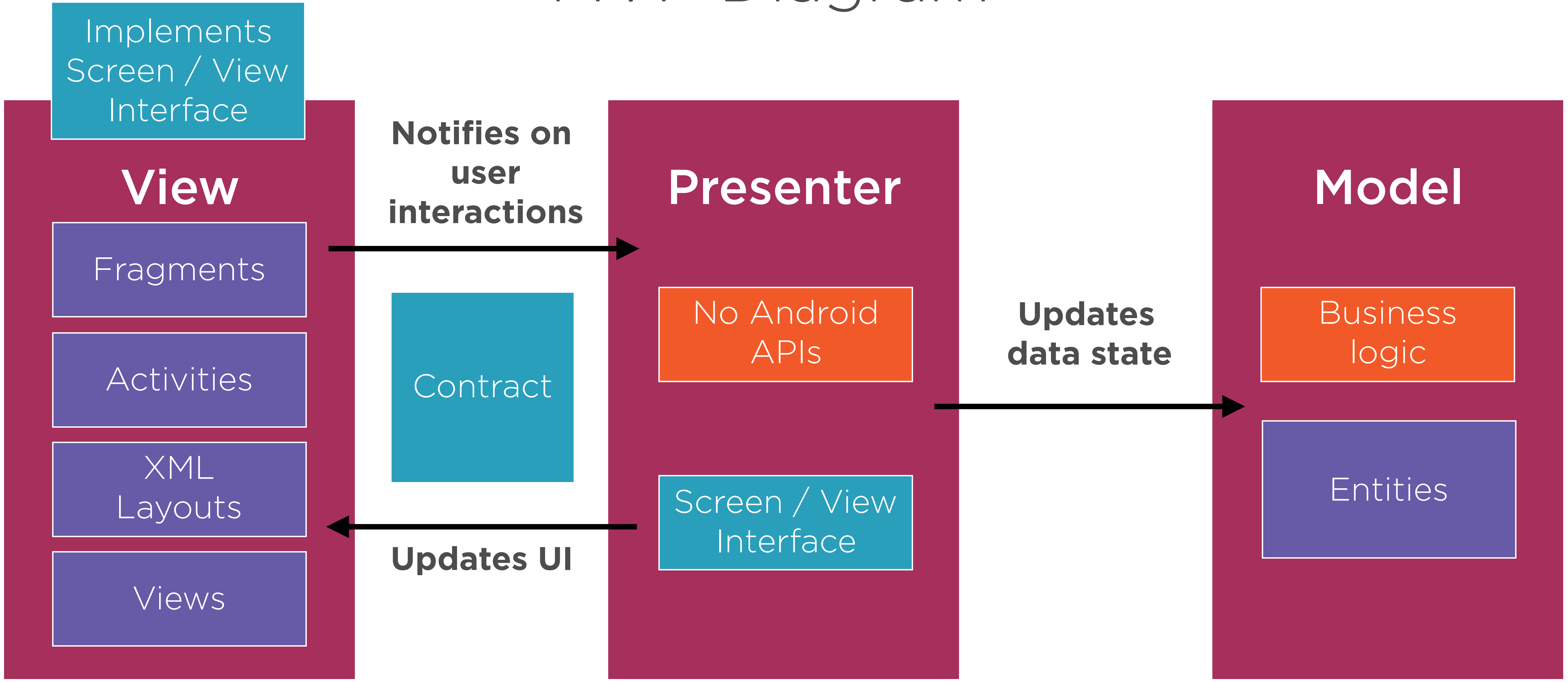
- Our XML Layouts and views
- Our activities and fragments:
  - In the Android world the two are strongly bonded with the views
  - Will implement an interface for the presenters actions

Presenter

MVP

- Has no relation to our views (Unlike MVC)
- Operations are invoked by our view (Activities or fragments)
- Views update is done via the view’s interface

MVP Diagram

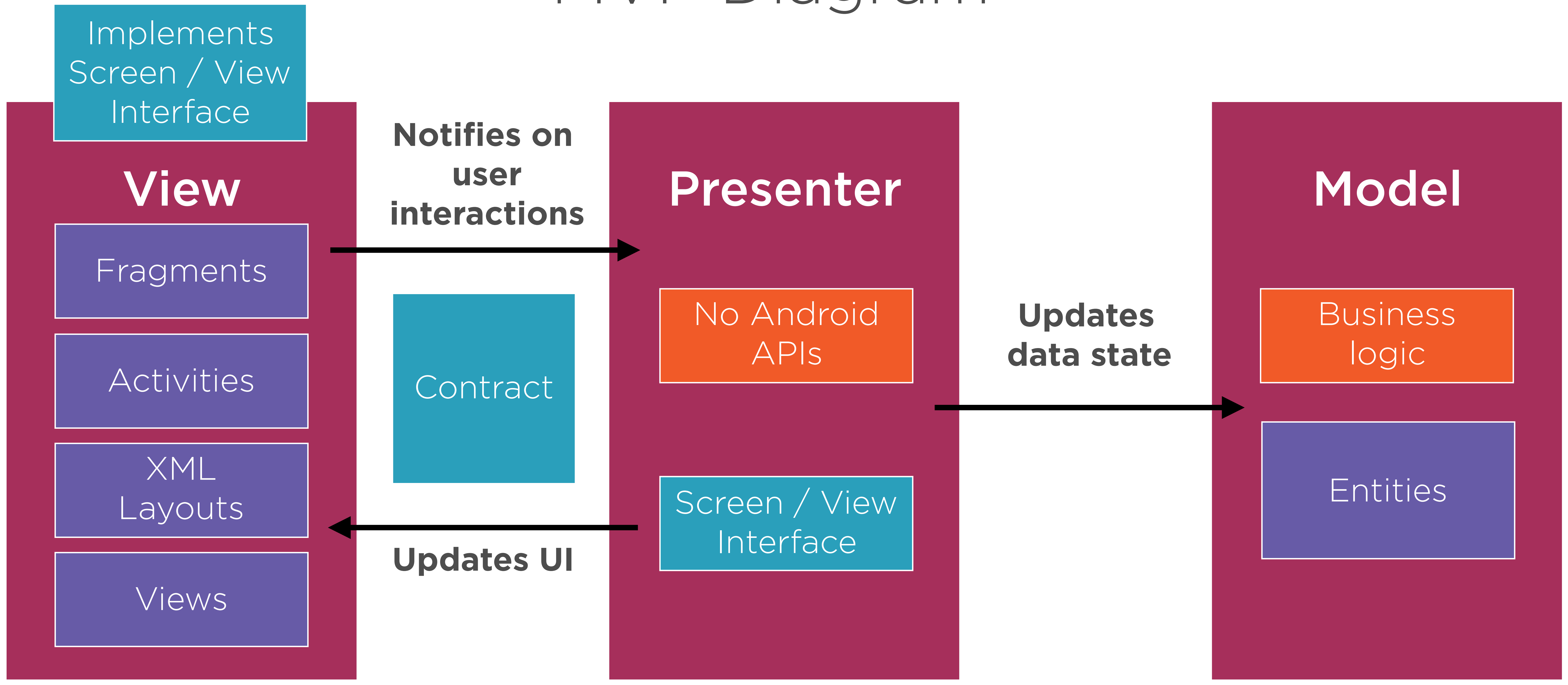

Implements Screen / View Interface
View
Fragments
Activities
XML Layouts
Views
Notifies on user interactions
Contract
Updates UI
Presenter
No Android APIs
Screen / View Interface
Updates data state
Model
Business logic
Entities

No Android Dependencies

=

More Testable and Decoupled

MVP Diagram


Implements Screen / View Interface
View
Fragments
Activities
XML Layouts
Views
Notifies on user interactions
Contract
Updates UI
Presenter
No Android APIs
Screen / View Interface
Updates data state
Model
Business logic
Entities

MVVM: Model View ViewModel

MVVM

- Using the Data Binding library from Google
- Minimize views binding code
- Views binding’s logic is implemented in the XML layout

Without Data Binding

```java
class CoinViewHolder extends RecyclerView.ViewHolder {

    TextView tvNameAndSymbol;
    TextView tvPriceAndVolume;
    ImageView ivIcon;

    public CoinViewHolder(View itemView) {
        super(itemView);
        tvNameAndSymbol = itemView.findViewById(R.id.tvNameAndSymbol);
        tvPriceAndVolume = itemView.findViewById(R.id.tvPriceAndVolume);
        ivIcon = itemView.findViewById(R.id.ivIcon);
    }
}
```

Without Data Binding

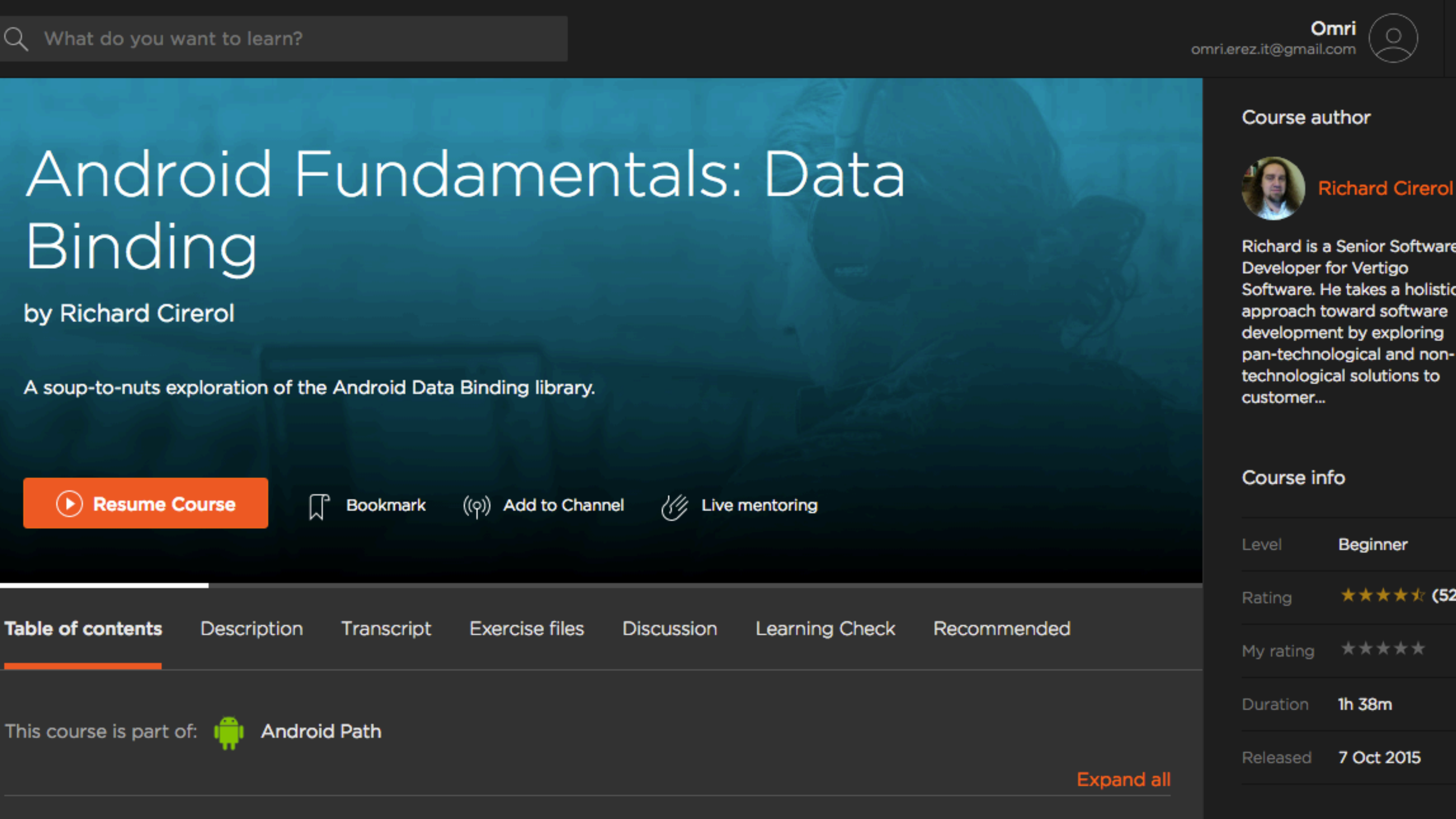
```
@Override
public void onBindViewHolder(CoinViewHolder holder, int position) {

    final CoinModel model = mItems.get(position);
    holder.tvNameAndSymbol.setText(model.name);
    holder.tvPriceAndVolume.setText(model.priceUsd);

}
```

With Data Binding

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable
            name="coin"
            type="com.pluralsight.cryptobam.MainActivity.CoinModel" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:orientation="vertical">
        <TextView
            android:id="@+id/tvNameAndSymbol"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{coin.name}" />
        <TextView
            android:id="@+id/tvPriceAndVolume"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{coin.priceUsd}"
            />
    </LinearLayout>
</layout>
```



What do you want to learn?

Omri
omri.erez.it@gmail.com

Android Fundamentals: Data Binding

by Richard Cirerol

A soup-to-nuts exploration of the Android Data Binding library.

Resume Course   Bookmark   Add to Channel   Live mentoring

Table of contents   Description   Transcript   Exercise files   Discussion   Learning Check   Recommended

This course is part of: Android Path

Expand all

Course author

Richard Cirerol

Richard is a Senior Software Developer for Vertigo Software. He takes a holistic approach toward software development by exploring pan-technological and non-technological solutions to customer...

Course info

Level	Beginner
Rating	★★★★☆ (52
My rating	★★★★★
Duration	1h 38m
Released	7 Oct 2015

MVVM

Model and View

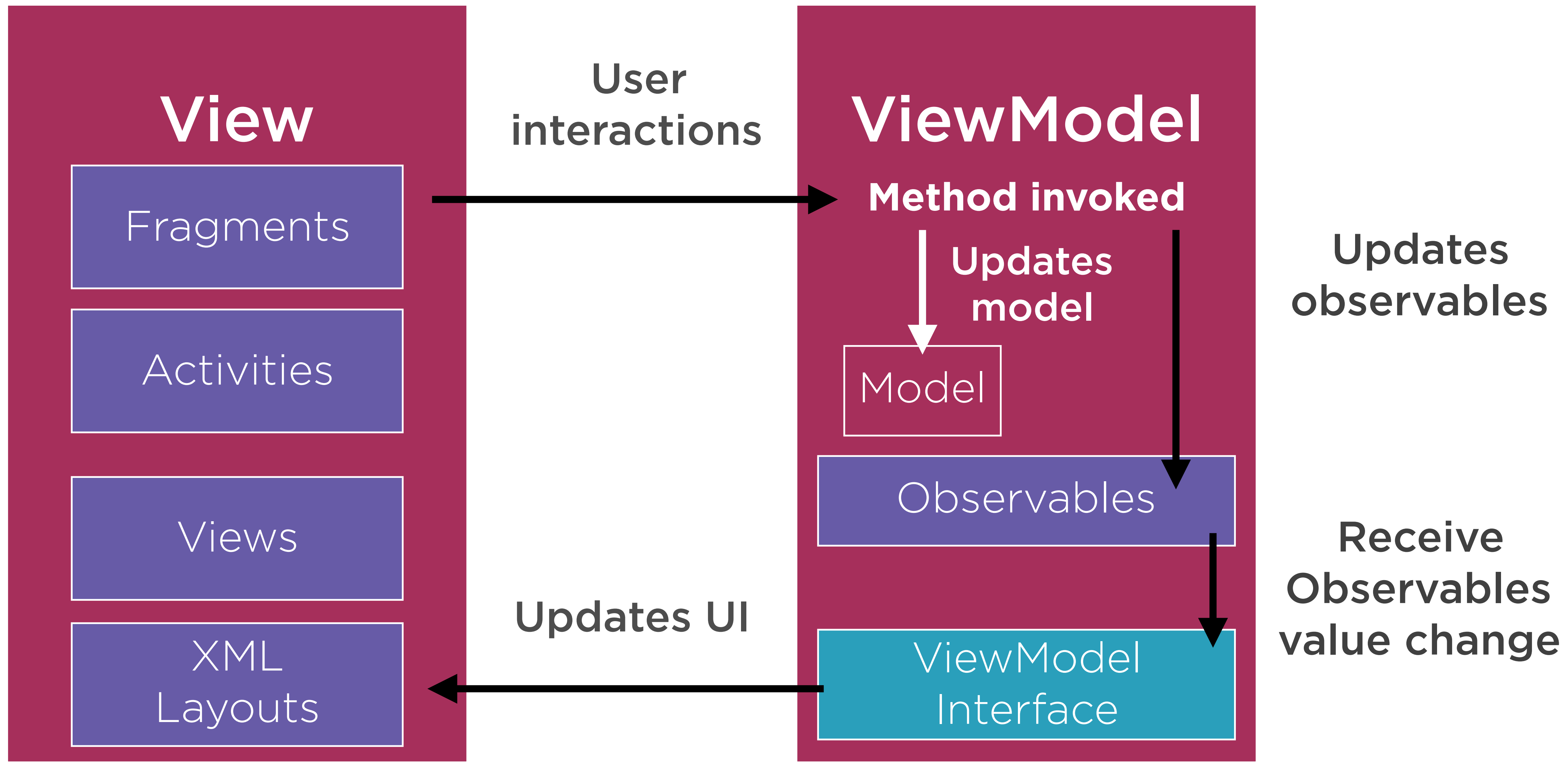
- Same as in the MVP pattern

MVVM

ViewModel

- Contains the Model
- Uses observable variables for update values
- On each value update, the relevant views will be updated

MVVM Diagram



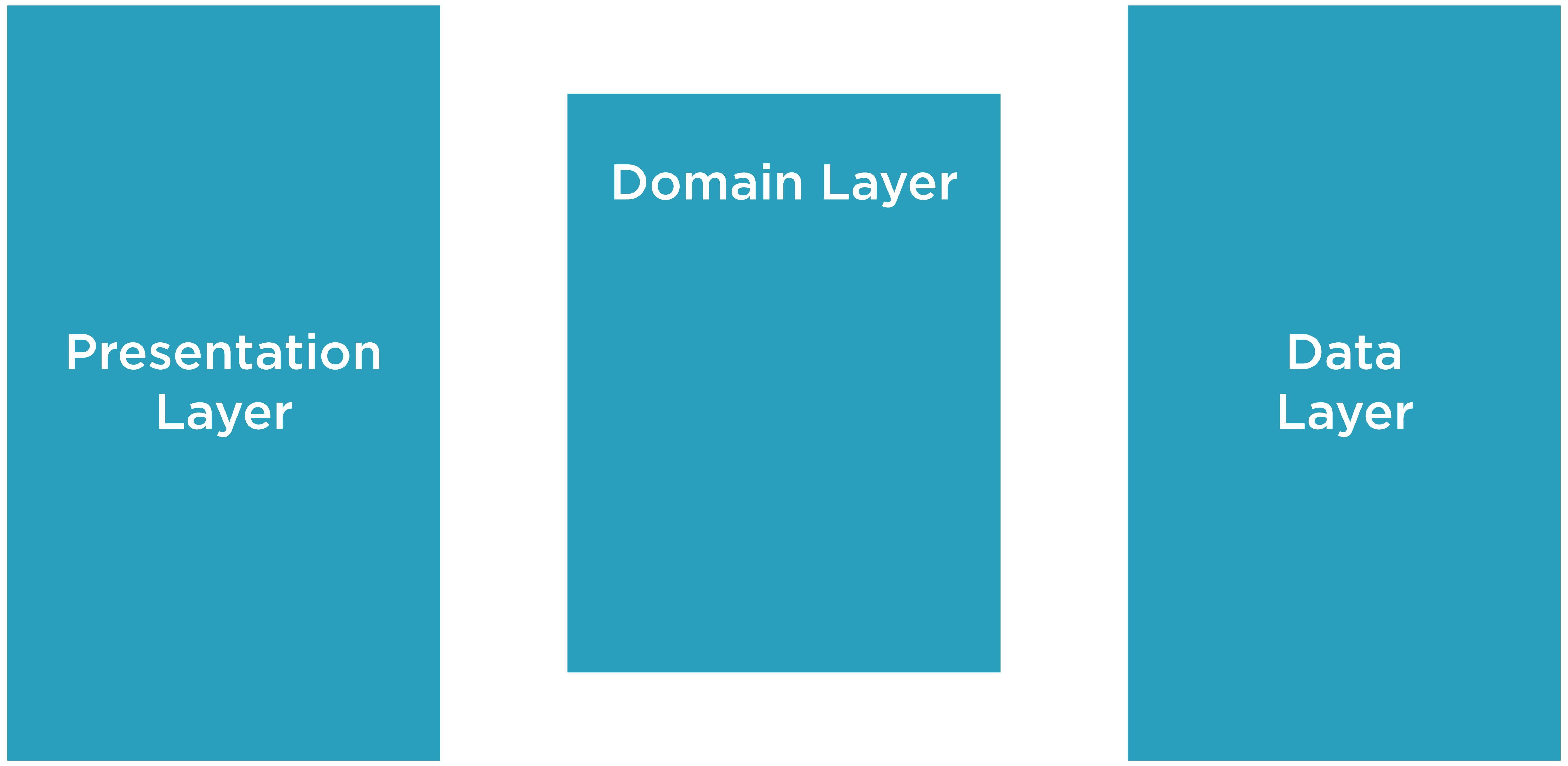
Comparison

Primitive Types and Objects Size

PATTERN	Dependency on Android APIS	XML Complexity	Unit Testability	Modular & SRP
MVC Controller	High	Low	Difficult	No
MVP Presenter	Low	Low	Good	Yes
MVVM ViewModel	Low or No dependency	Medium or High	Great	Yes

The Clean Architecture

The Clean Architecture



Presentation Layer

- Android components: activities, services, fragments etc.
- Custom views
- Presenters

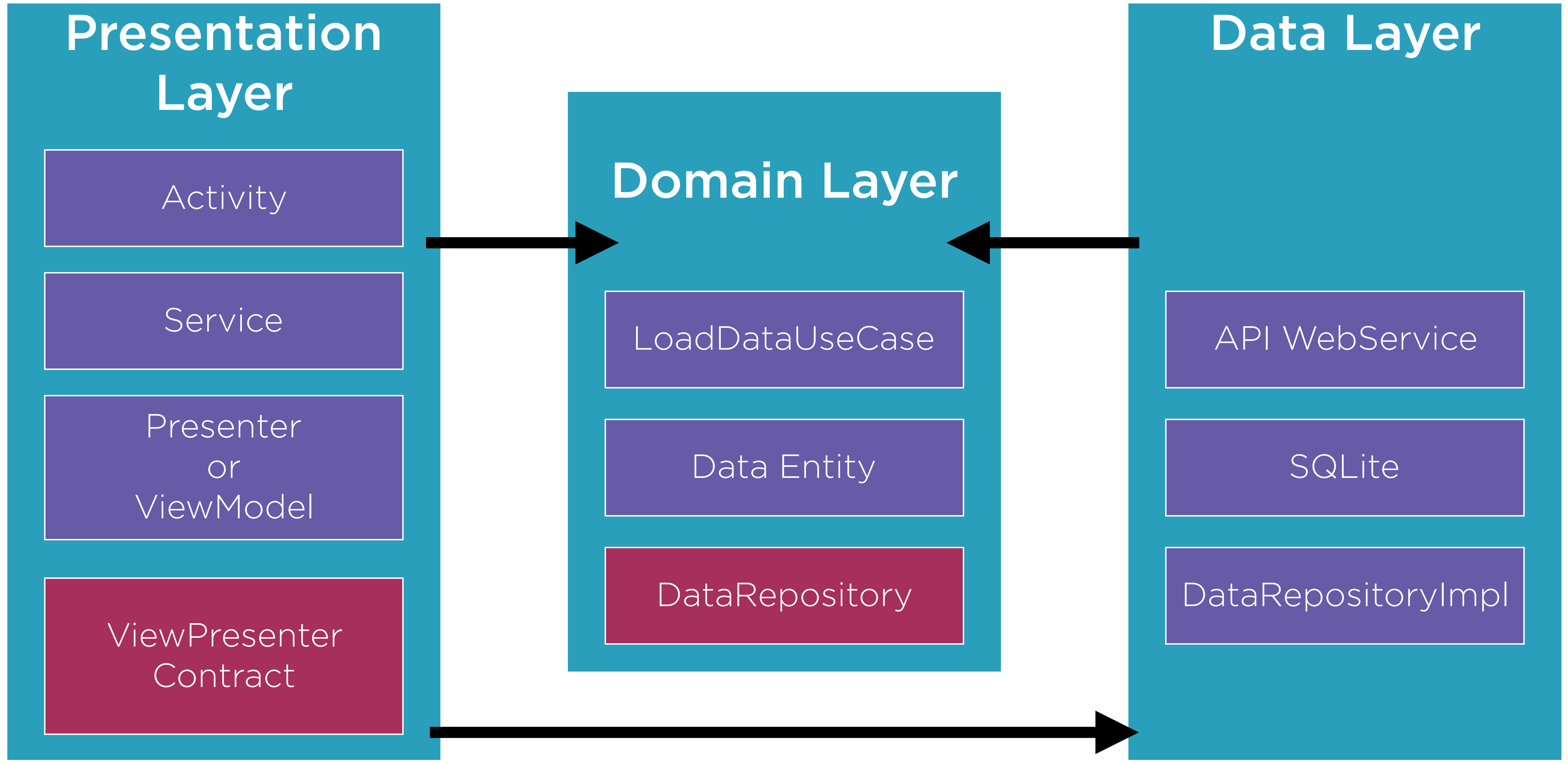
Domain Layer

- Entities
- Screen interfaces
- Use cases
- Pure Java / Kotlin - Android free

Data Layer

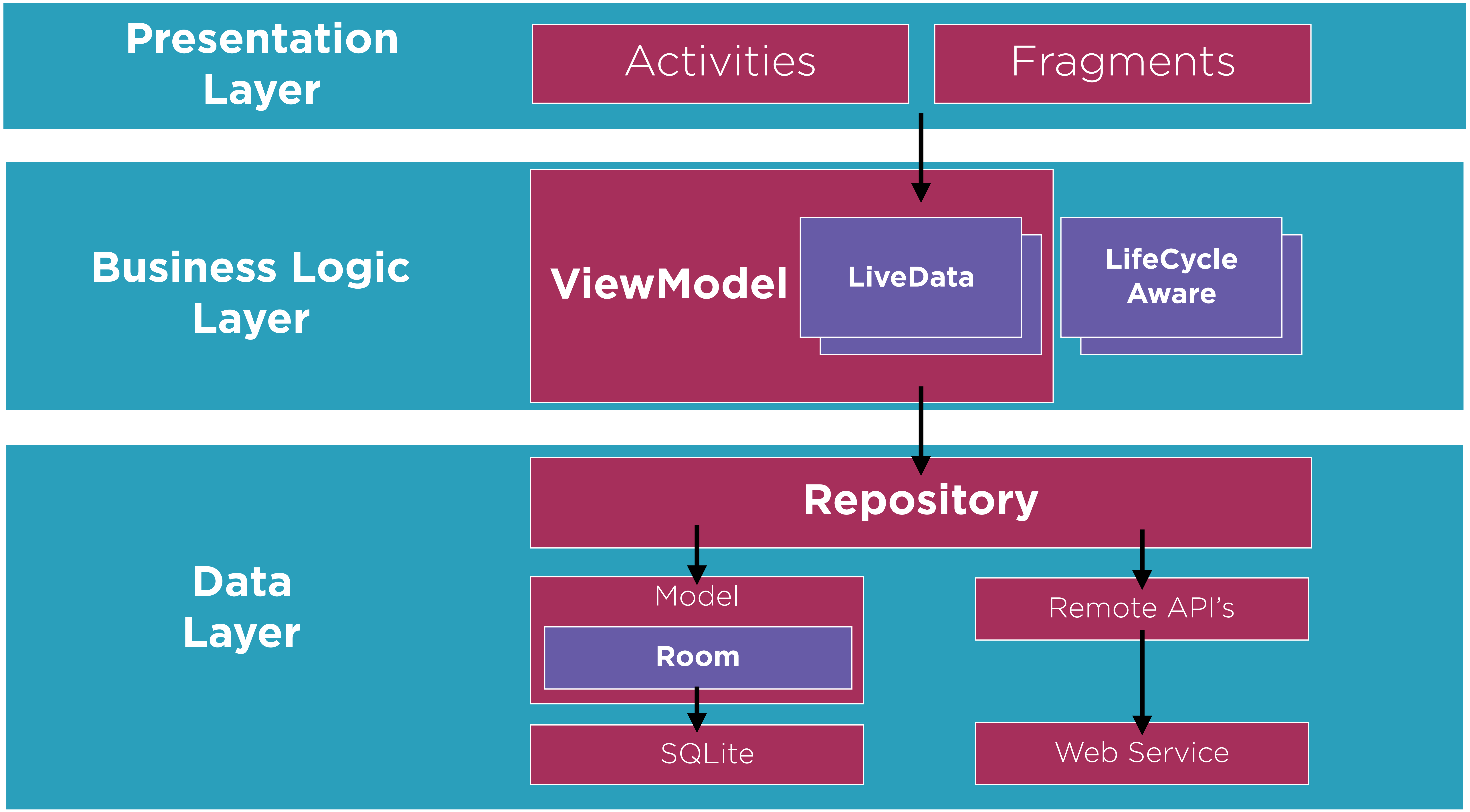
- External APIs access
- Local storage components
- Memory & disk caches

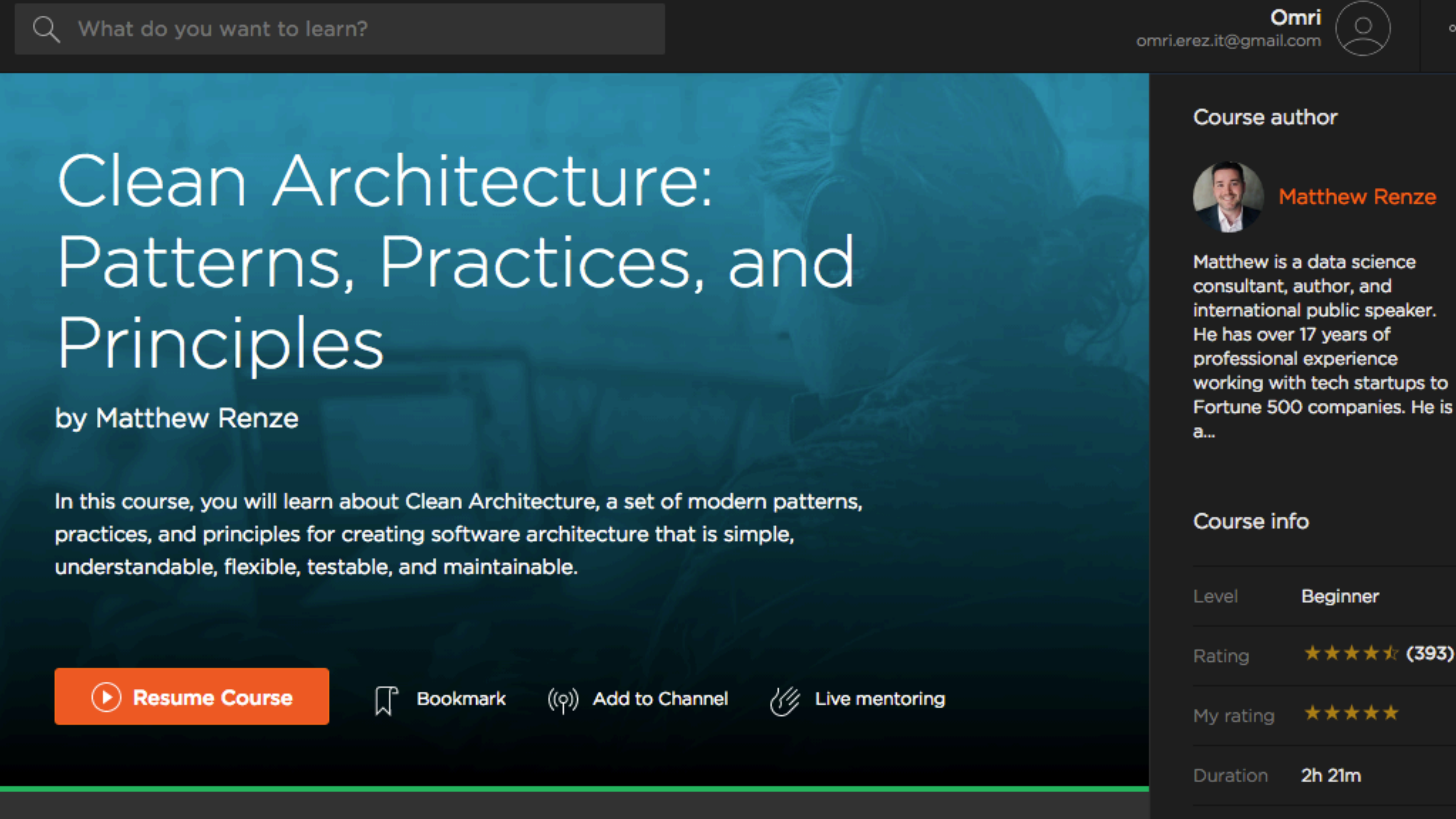
The Clean Architecture



Advantages

- Modular approach
- Each module can be tested separately
- Domain layer can be reused for other JVM applications


Presentation Layer
Activities
Fragments
Business Logic Layer
ViewModel
LiveData
LifeCycle Aware
Data Layer
Repository
Model
Room
SQLite
Remote API's
Web Service



What do you want to learn?

Omri
omri.erez.it@gmail.com

Clean Architecture: Patterns, Practices, and Principles

by Matthew Renze

In this course, you will learn about Clean Architecture, a set of modern patterns, practices, and principles for creating software architecture that is simple, understandable, flexible, testable, and maintainable.

Resume Course | Bookmark | Add to Channel | Live mentoring

Course author

Matthew Renze

Matthew is a data science consultant, author, and international public speaker. He has over 17 years of professional experience working with tech startups to Fortune 500 companies. He is a...

Course info

Level	Beginner
Rating	★★★★⯪ (393)
My rating	★★★★★
Duration	2h 21m

Summary

Common architectural patterns

- God classes
- Differences between MVC, MVP, and MVVM
- The Clean Architecture