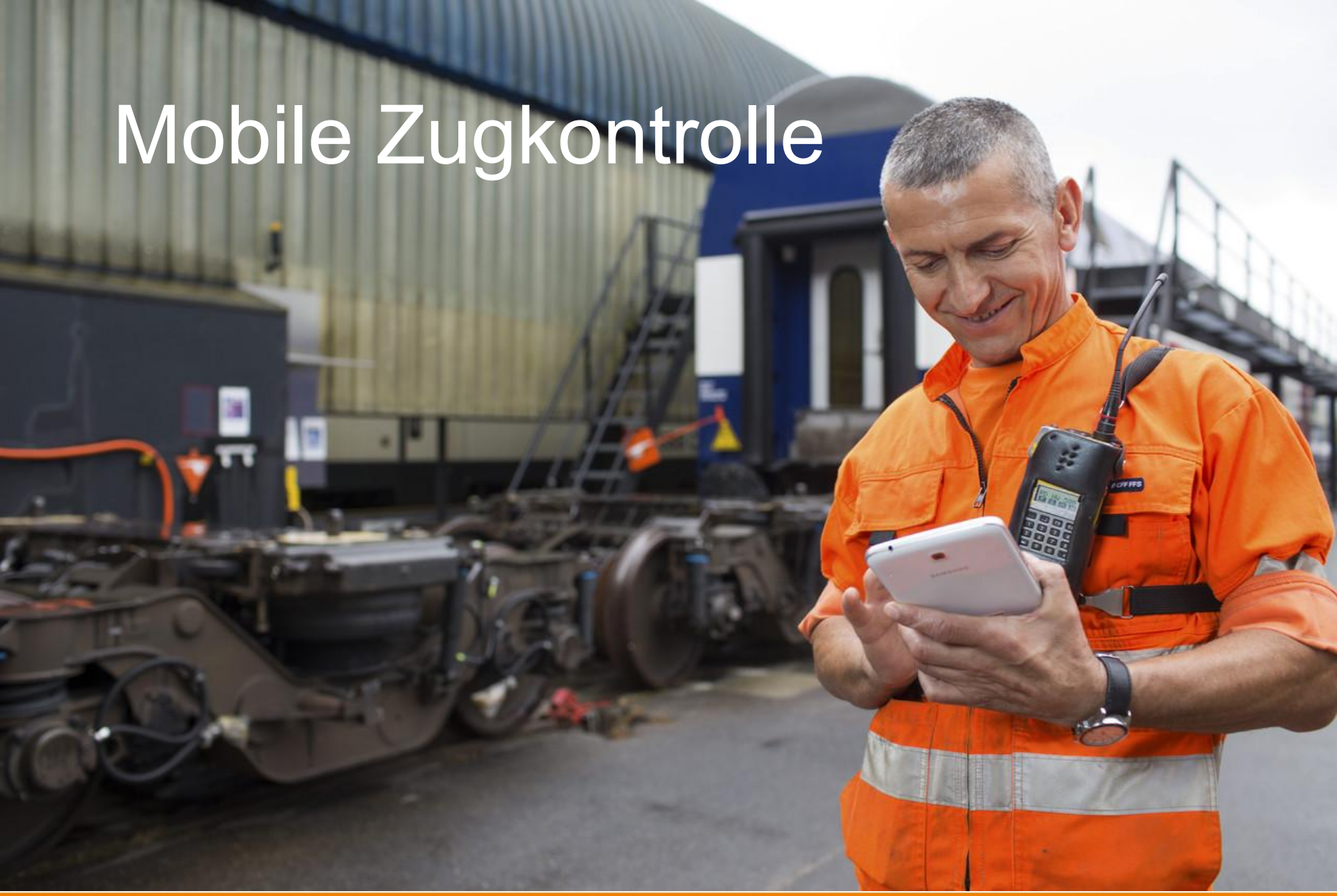
MobileTechCon

Simon Butscher | SBB CFF FFS
Gabriel Weis | BINOSYS GmbH

Android Architecture

Mobile Zugkontrolle



MobileTechCon

Migration

95% 11:07

Soll-Liste

02.02.2016 / AK Abfuhr / HABI

43017 / 5041

3385 4506 2667
A

3385 4506 5058
A

3385 4526 1970
A

3385 4556 7533
A

3385 4556 8465
A

3385 4575 1244
A

3385 4575 5294
A

3385 4575 7811
A

3385 4576 1458
A

3385 4576 2001
A

3385 4576 2357
A

3385 4576 4064
A

8385 4512 0597
A

8385 4555 5560
A

8385 4555 6048

A:	T:	LüP:	Hg:	Grp:	Bs:
11	21.0	200	9	1	Ein

P:	G:	R:	Vl:	Vb:	Vs:
50	0	999	100	120	

Kg:	Hb:	BAC:	Be:	Ba:
	1	38	1	2(K)

Von:	Nach:
Hamburg-Billwerder Ubf	Gallarate
80-10660	83-10306

Lw:

80-91 85-22

UNO G/S:	FAG:

agS:	Bb:

	C
S	55.0 t
SS	55.0 t

★★

Design in Progress

Facts

- Lines of Code: 42K
- Unit Test Coverage: 76% / 3321 Test
  - Activites: 14
  - Fragments: 72
  - Services: 2
  - Providers: 11
  - EventBus: 1

Why Architecture

- Understandability
- Testability
- Maintainability
- Reliability

Why

“A good design is easier to change than a bad design”

Dave Thomas

Why not

- Architecture is not free
  - More source code
  - Less performance
  - Needs reviews
  - Needs experienced developers

Android Design Flaws

- What makes it difficult on Android to have a good Architecture?

Android Flaws

- Android violates SOLID
  - Single responsibility:
    - Context
    - Activity
    - Service
    - Fragments
  - Dependeny inversion:
    - Concrete objects
    - Few interfaces

Android Flaws

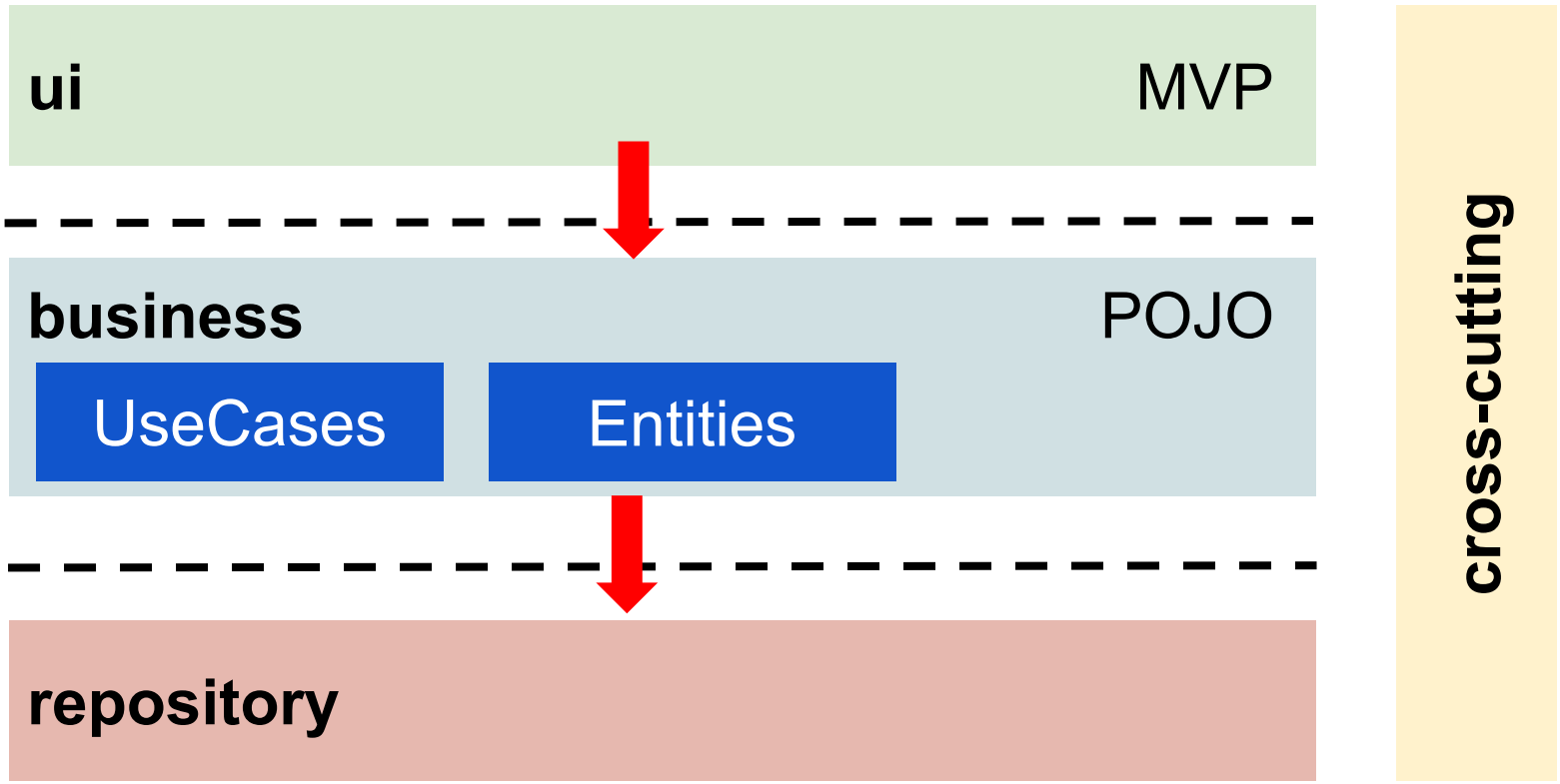
- No MVC(P)
- Degree of freedom
- Absence of concrete guidelines

Layer

© honold.ch

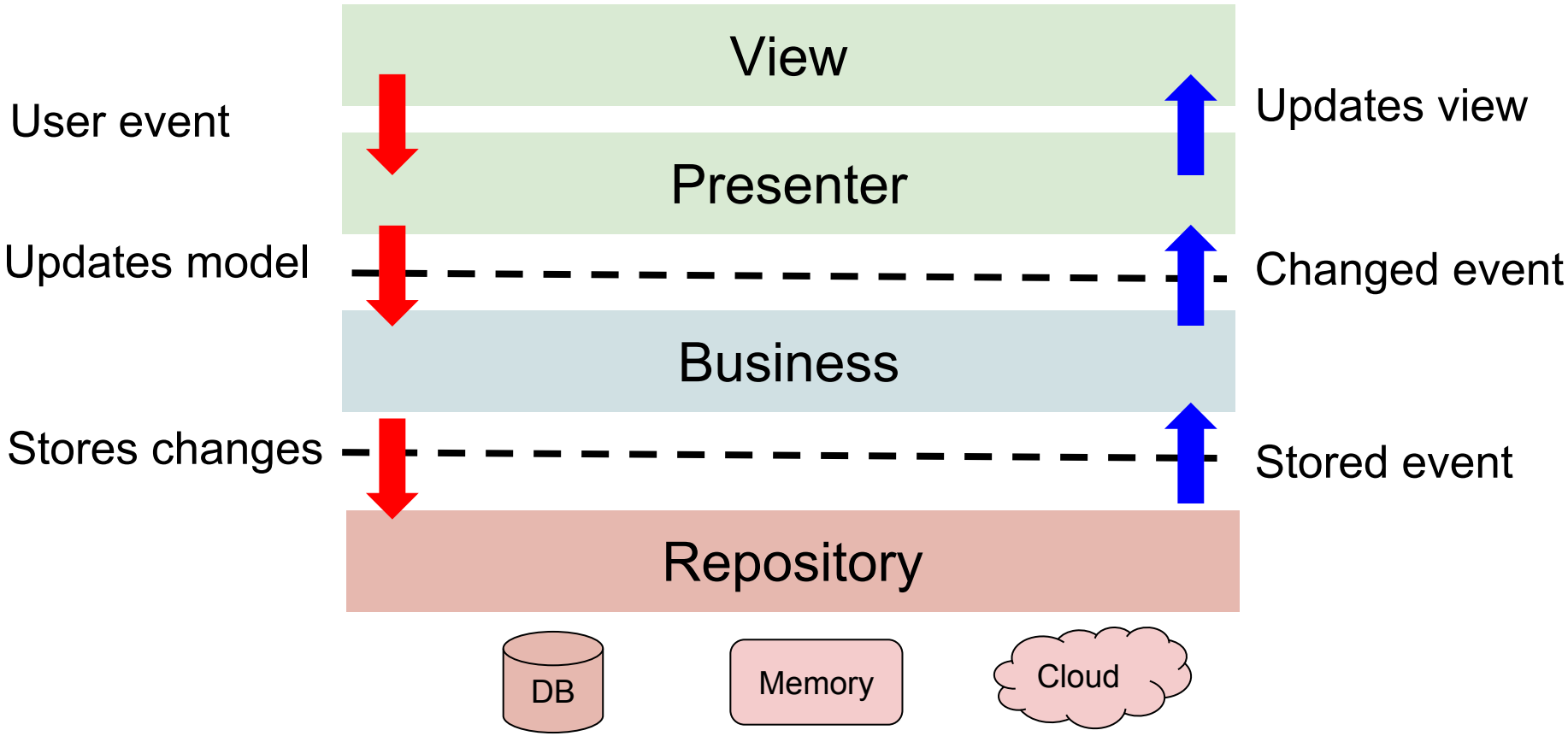
© burgerking.co.za

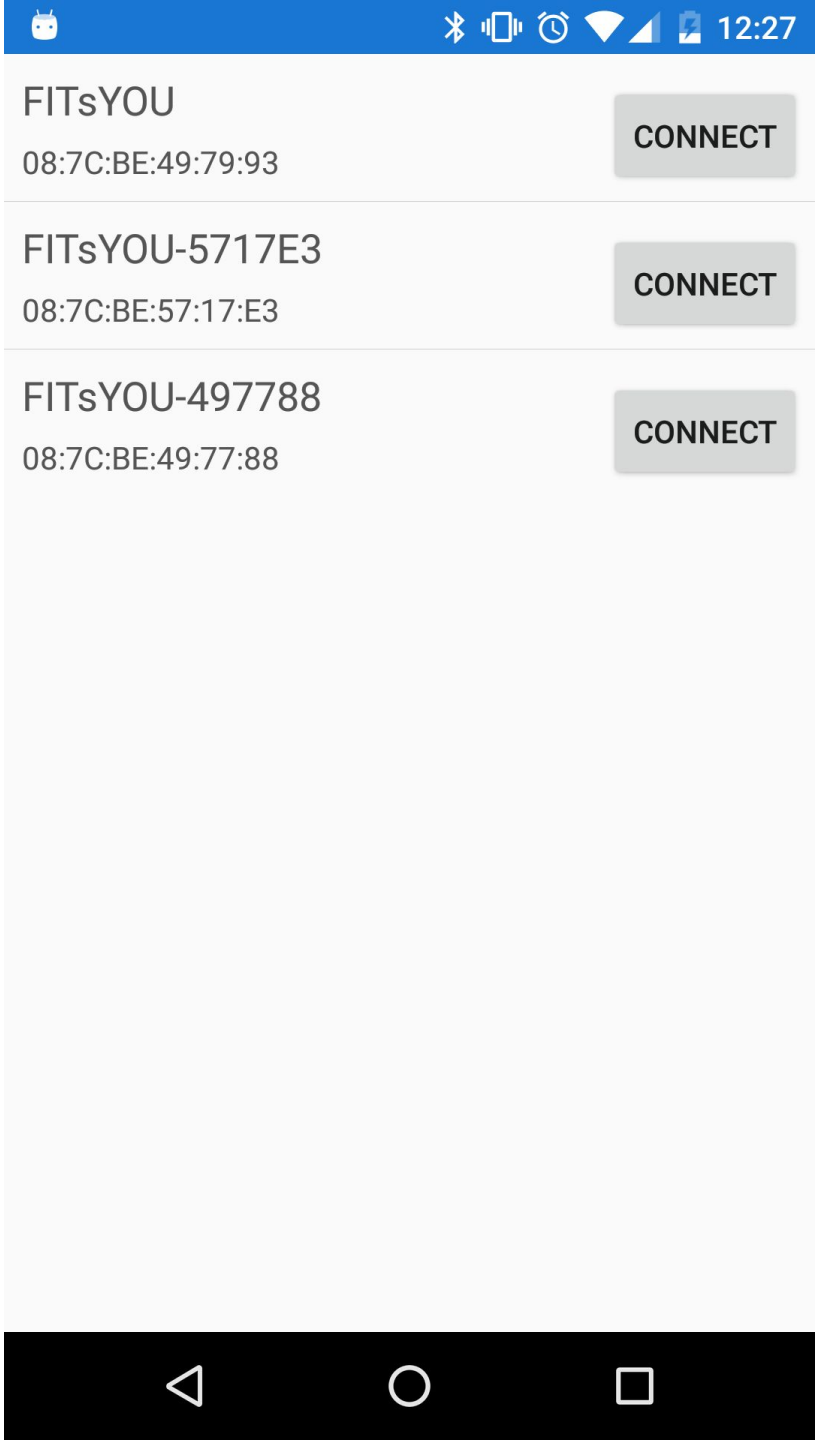
Layer



Dependency Rule

MVP and Flow


View
User event
Updates view
Presenter
Updates model
Changed event
Business
Stores changes
Stored event
Repository
DB
Memory
Cloud


12:27
FITsYOU
08:7C:BE:49:79:93
CONNECT
FITsYOU-5717E3
08:7C:BE:57:17:E3
CONNECT
FITsYOU-497788
08:7C:BE:49:77:88
CONNECT

Otto

- Easy to understand
- Decoupling & omit momory leaks
- Easy Mocking

- Intransparent
- Easy to misuse

Alternative: RXJava

More powerfull and explicit…

… less easy to learn

```java
public class BusExampleFragment extends Fragment{

    @Inject
    IBusSystem bus;
    @Inject
    ExamplePresenter presenter;

    @Override
    public void onResume() {

        super.onResume();
        bus.register(this);
        bus.register(presenter);

        bus.post(new Event());
    }


    @Override
    public void onPause() {

        super.onPause();
        bus.unregister(this);
        bus.unregister(presenter);
    }

    @Subscribe
    public void on(Event event){
        // do something
    }
}
```

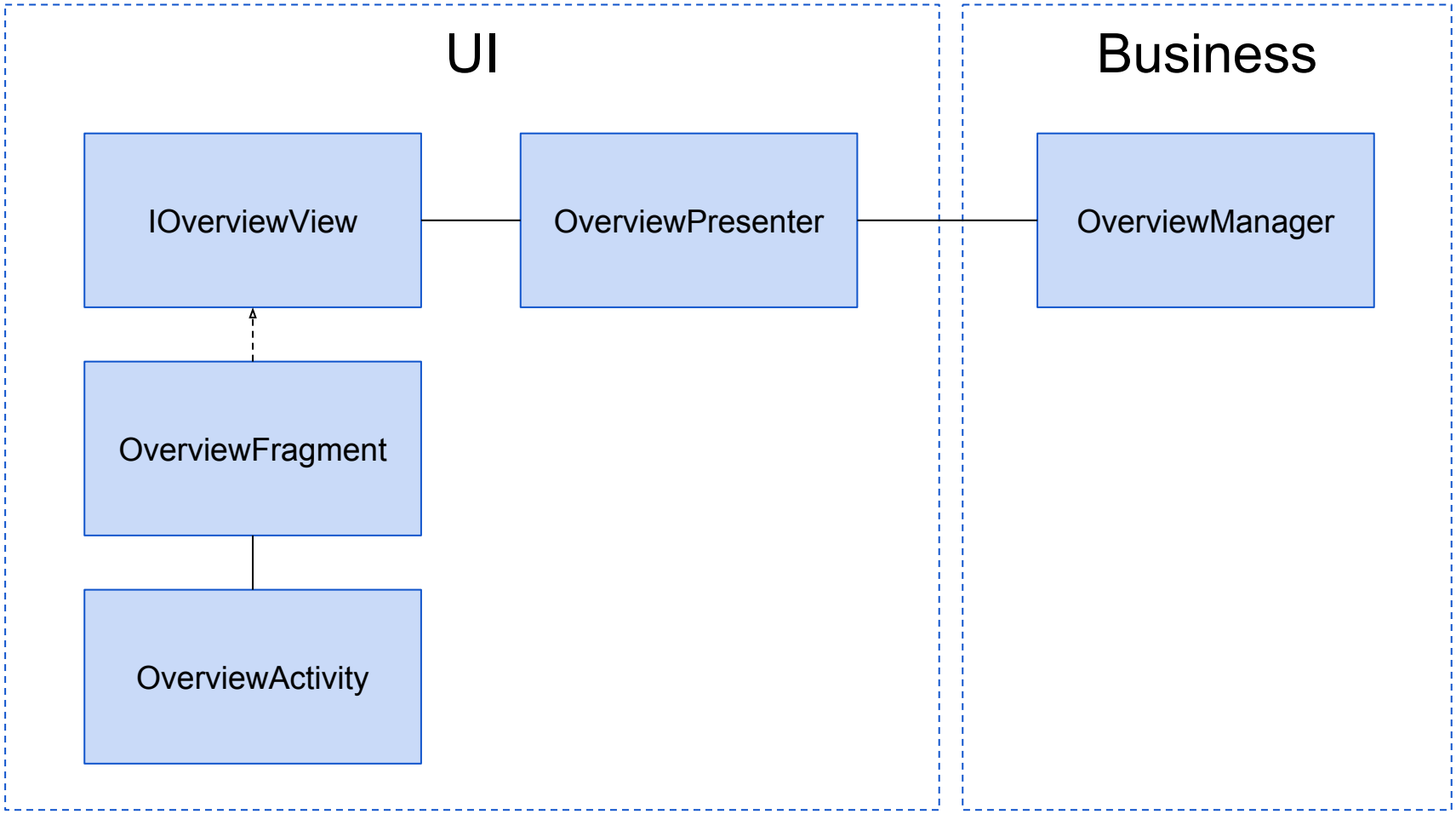
```java
public class BusExampleFragment extends Fragment{

    @Inject
    IBusSystem bus;
    @Inject
    ExamplePresenter presenter;

    @Override
    public void onResume() {

        super.onResume();
        bus.register(this);
        bus.register(presenter);

        bus.post(new Event());
    }


    @Override
    public void onPause() {

        super.onPause();
        bus.unregister(this);
        bus.unregister(presenter);
    }

    @Subscribe
    public void on(Event event){
        // do something
    }
}
```

```java
@BusObserver
public class BusExampleFragment extends Fragment{

    @Inject
    IBusSystem bus;
    @Inject
    ExamplePresenter presenter;
    @Inject
    BusRegisterer registerer;

    @Override
    public void onResume() {

        super.onResume();
        registerer.register(this);

        bus.post(new Event());
    }

    @Override
    public void onPause() {

        super.onPause();
        registerer.unregister(this);
    }

    @Subscribe
    public void on(Event event){
        // do something
    }
}
```

Activity & Fragments...

… make things ugly
(super fast)


KISS


UI
Business
IOverviewView
OverviewPresenter
OverviewManager
OverviewFragment
OverviewActivity

```java
public class OverviewActivity extends DIActivity {

    @Inject
    OverviewFragment fragment;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.overview_activity);

        showOverviewFragment();
    }

    private void showOverviewFragment() {...}

    @Override
    protected List<Object> getModules() {...}
}
```

```java
@RunWith(AndroidJUnit4.class)
@LargeTest
public class OverviewActivityTest {

    @Rule
    public ActivityTestRule<OverviewActivity> activityRule = new ActivityTestRule<>(
            OverviewActivity.class);

    @Test
    public void initialization() {
        // Arrange
        OverviewActivity testee = activityRule.getActivity();
        // Act
        FragmentManager manager = testee.getSupportFragmentManager();
        // Assert
        assertNotNull(testee.fragment);
        Fragment fragment = manager.findFragmentById(R.id.activity_overview_container);
        assertEquals(testee.fragment, fragment);
    }

    @Test
    public void getModules() {

        // Arrange
        OverviewActivity testee = activityRule.getActivity();
        // Act
        List<Object> modules = testee.getModules();
        // Assert
        assertNotNull(modules);
        assertEquals(1, modules.size());
        assertTrue(modules.get(0) instanceof OverviewModule);
    }
}
```

```java
@BusObserver
public class OverviewFragment extends Fragment implements IOverviewView {


    @Inject
    BusRegisterer registerer;
    @Inject
    OverviewPresenter presenter;
    @Inject
    OverviewListAdapter listAdapter;

    private View root;
    private ListView deviceList;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        presenter.setView(this);
    }


    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {...}

    @Override
    public void onResume() {

        super.onResume();
        registerer.register(this);
        presenter.onResume();
    }

    @Override
    public void onPause() {

        presenter.onPause();
        registerer.unregister(this);
        super.onPause();
    }
```

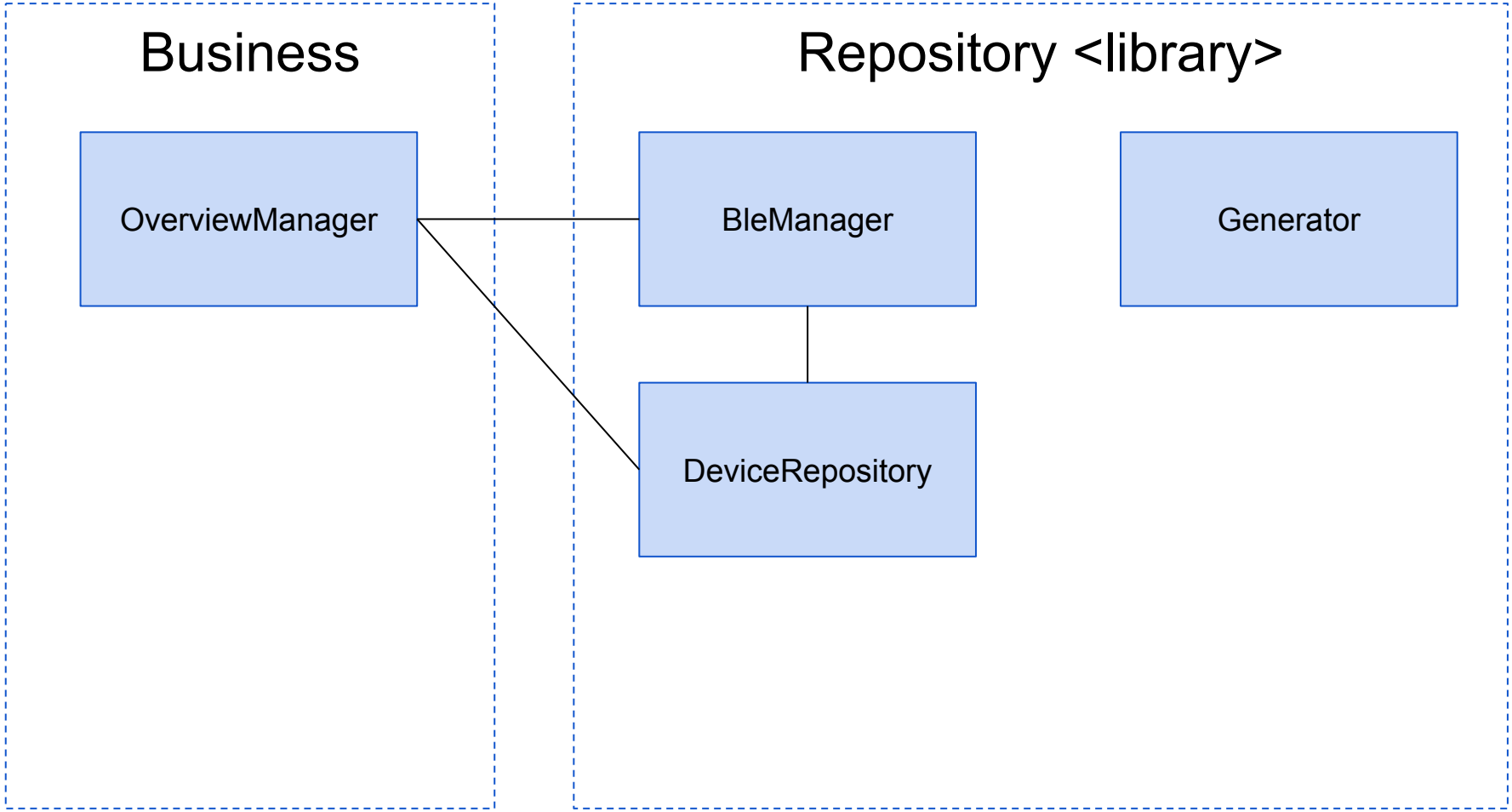
```java
@BusObserver
public class OverviewFragment extends Fragment implements IOverviewView {


    @Override
    public void addDeviceItem(BleDevice device) {

        listAdapter.add(device);
        listAdapter.notifyDataSetChanged();
    }


    @Override
    public void setDeviceItems(@NonNull List<BleDevice> devices) {

        if (devices != null) {
            listAdapter.clear();
            listAdapter.addAll(devices);
            listAdapter.notifyDataSetChanged();
        }
    }

    @Override
    public void startActivity(Class<DetailActivity> activityClass) {
        Intent intent = new Intent(getActivity(), DetailActivity.class);
        startActivity(intent);
    }
```

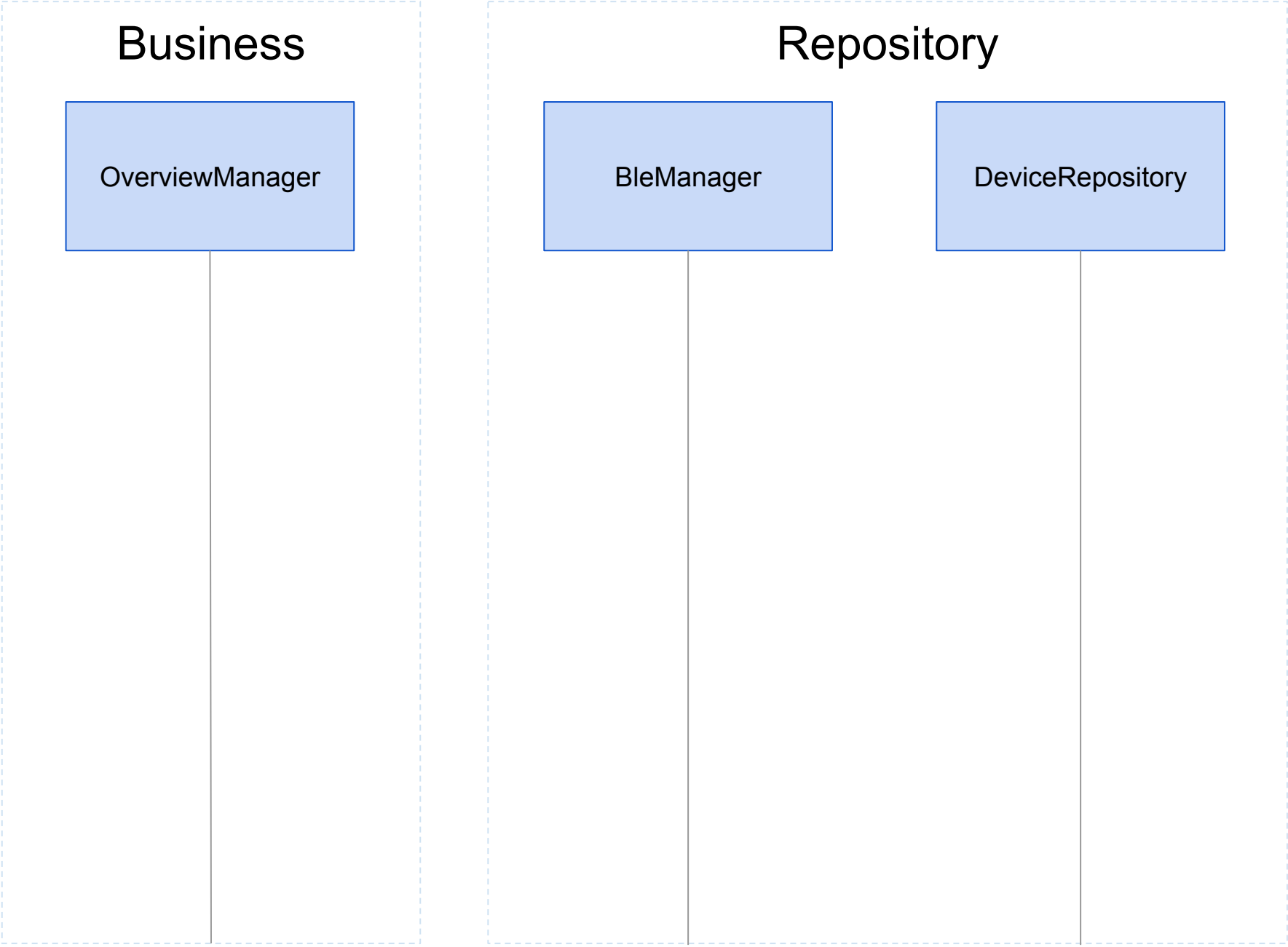
```java
public class OverviewFragmentTest {


    @Mock
    private BusRegisterer mockRegisterer;
    @Mock
    private OverviewPresenter mockPresenter;
    @Mock
    private OverviewListAdapter mockListAdapter;

    private OverviewFragment testee;

    @Before
    public void setUp(){

        MockitoAnnotations.initMocks(this);
        testee = new OverviewFragment();
        testee.presenter = mockPresenter;
        testee.registerer = mockRegisterer;
        testee.listAdapter = mockListAdapter;
    }
```

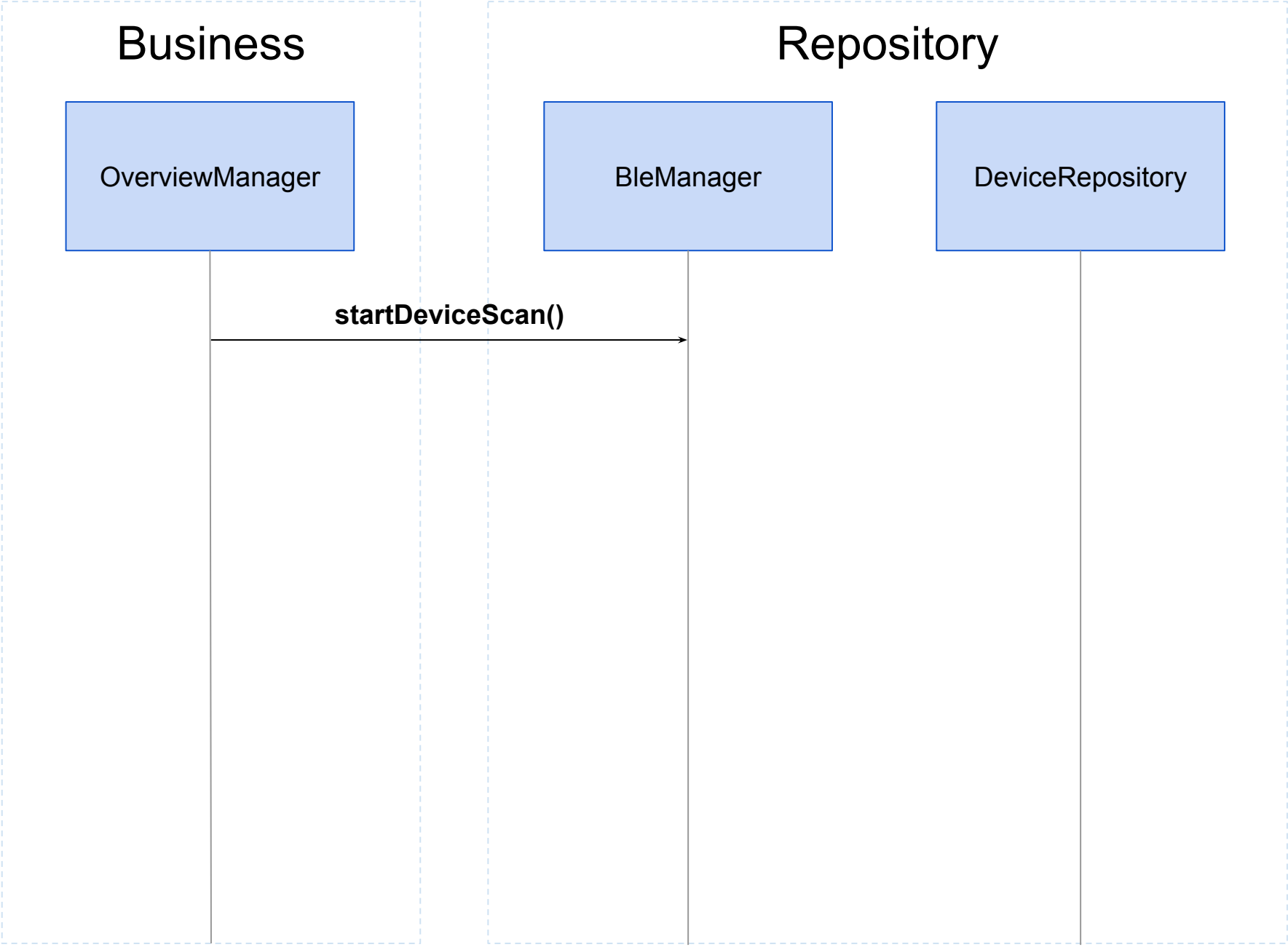
```java
public class OverviewFragmentTest {

    @Test
    public void addDeviceItem() throws Exception {
        // Arrange
        BluetoothDevice mockBluetoothDevice = mock(BluetoothDevice.class);
        // Act
        testee.addDeviceItem(mockBluetoothDevice);
        // Assert
        verify(mockListAdapter).add(mockBluetoothDevice);
        verify(mockListAdapter).notifyDataSetChanged();
    }


    @Test
    public void onResume() throws Exception {
        // Act
        testee.onResume();
        // Assert
        verify(mockRegisterer).register(this);
        verify(mockPresenter).onResume();
    }
```

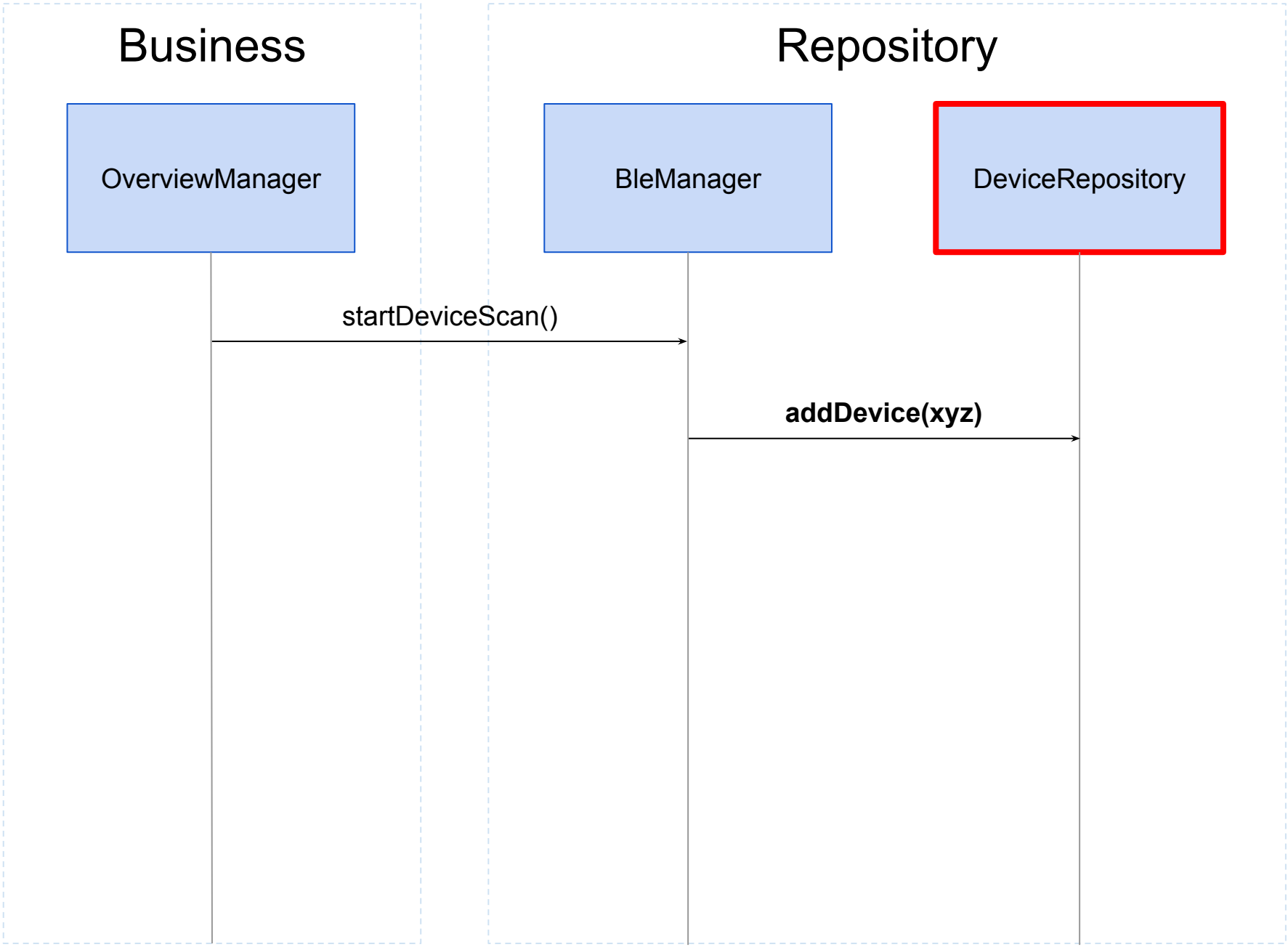
```java
@Singleton
@BusObserver
public class OverviewPresenter {

    @Inject
    OverviewManager manager;
    private IOverviewView view;

    public void onResume() {

        manager.startDeviceScan();
        initList();
    }

    public void onPause() {

        manager.stopDeviceScan();
    }

    public void setView(IOverviewView view) {

        this.view = view;
    }

    private void initList() {

        List<BleDevice> devices = manager.getAllDevices();
        view.setDeviceItems(devices);
    }
```

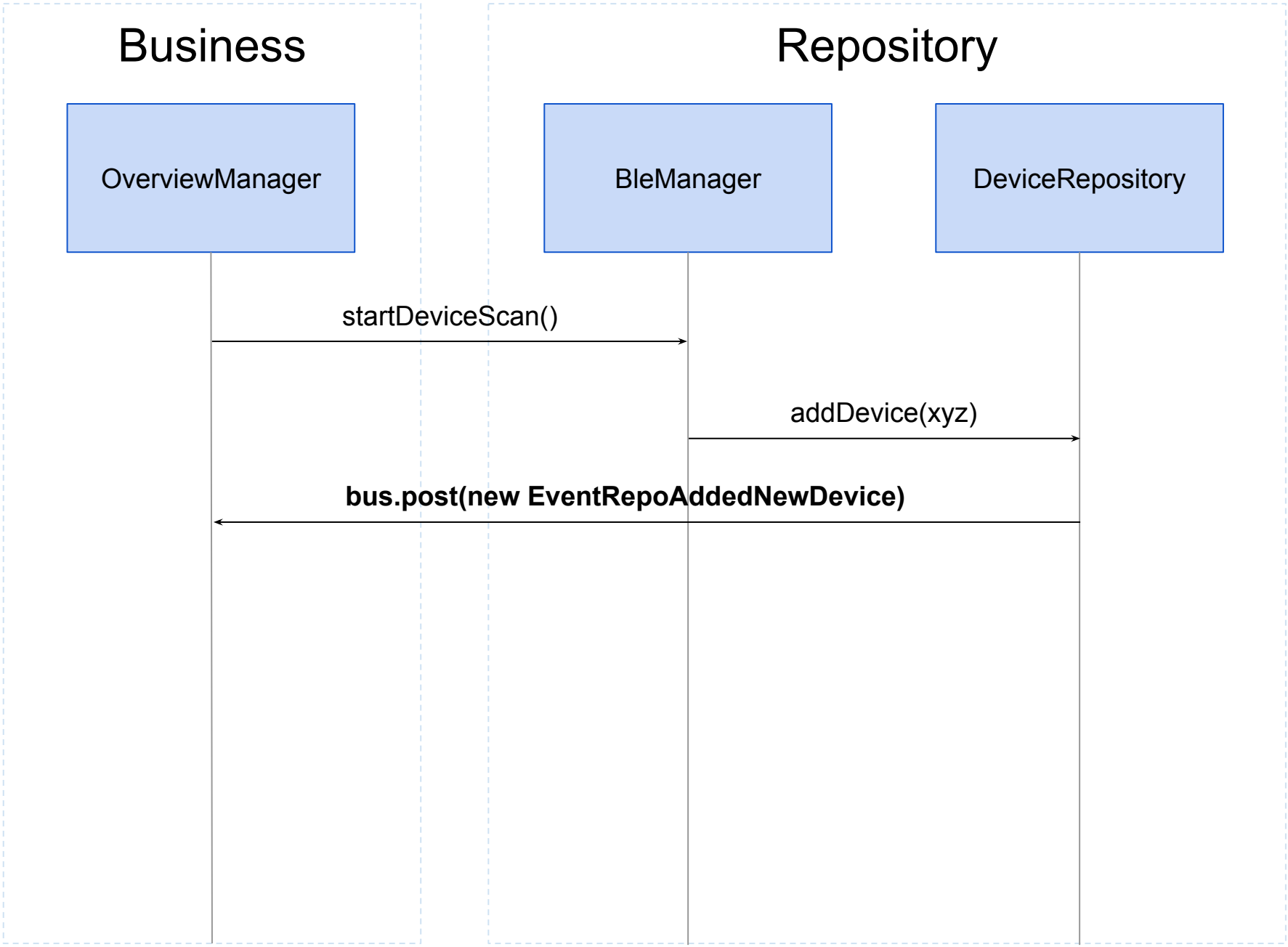
```java
@Singleton
@BusObserver
public class OverviewPresenter {

    @Subscribe
    public void on(EventBusinessFoundNewDevice event) {

        BleDevice device = manager.getDeviceWith(event.address);
        view.addDeviceItem(device);
    }

    public void onConnectButtonClick(BleDevice device) {

        manager.setSelectedDevice(device);
        view.startActivity(DetailActivity.class);
    }
```

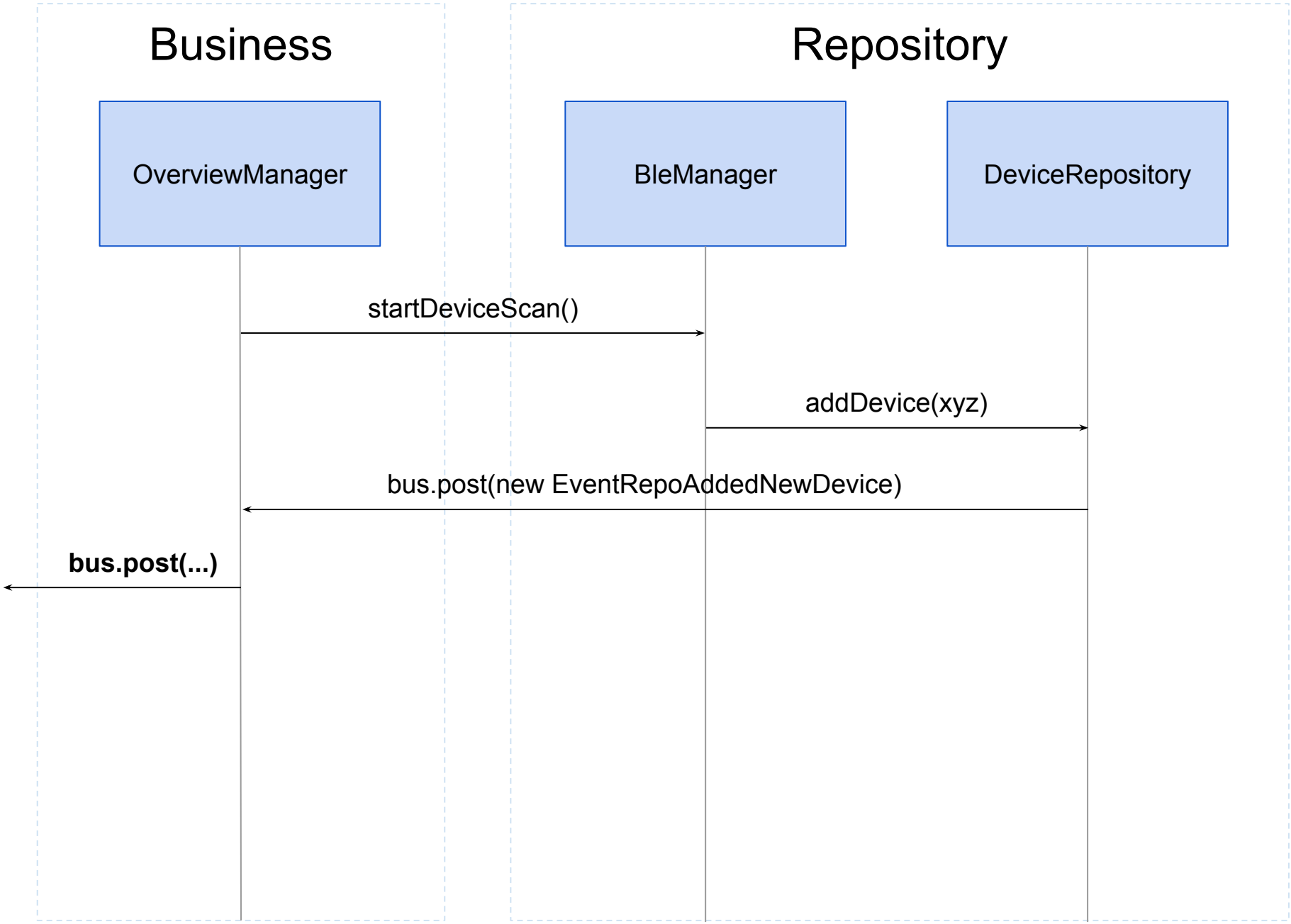

Business
Repository <library>
OverviewManager
BleManager
Generator
DeviceRepository

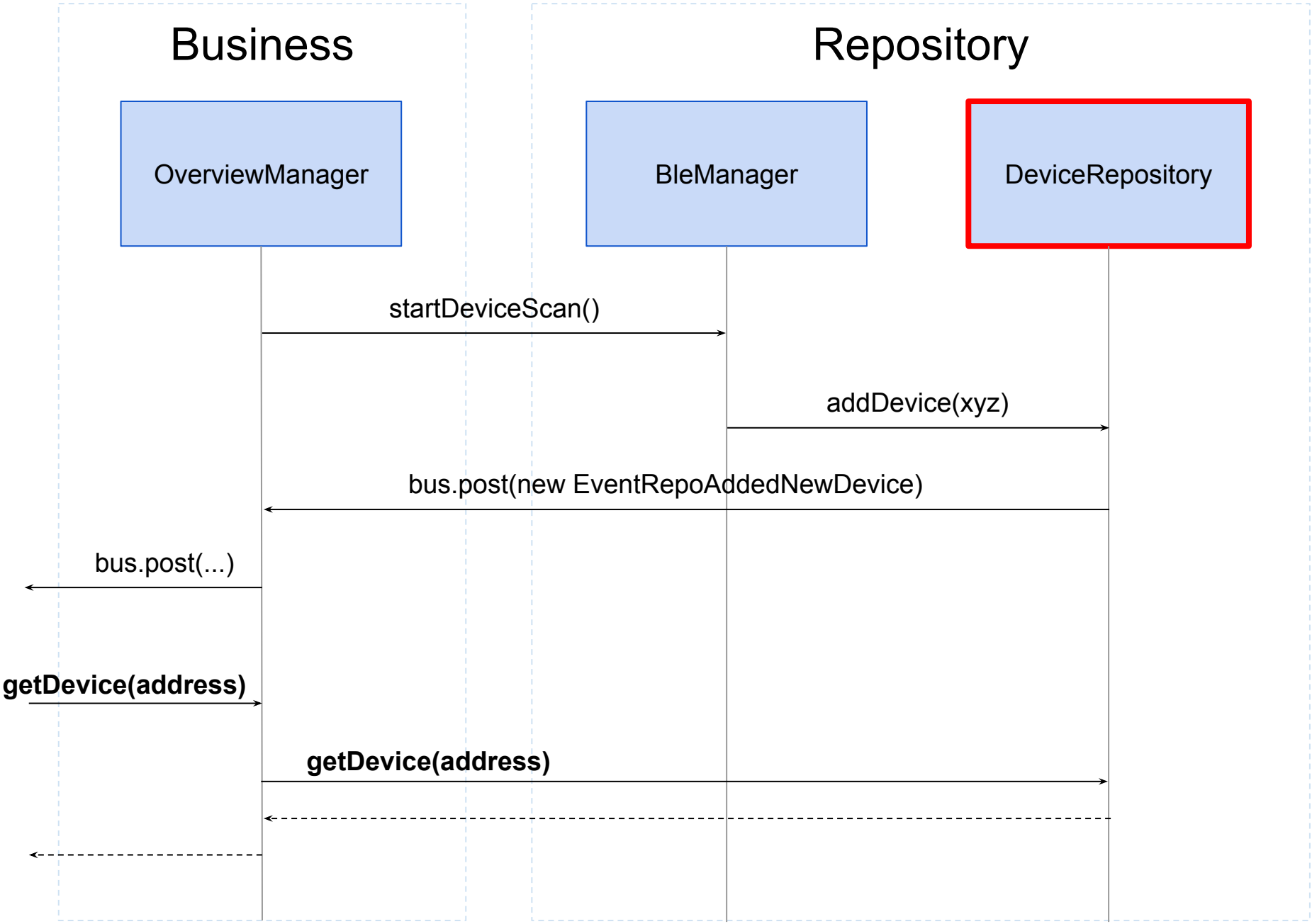

Business
Repository
OverviewManager
BleManager
DeviceRepository


Business
Repository
OverviewManager
BleManager
DeviceRepository
startDeviceScan()


Business
Repository
OverviewManager
BleManager
DeviceRepository
startDeviceScan()
addDevice(xyz)

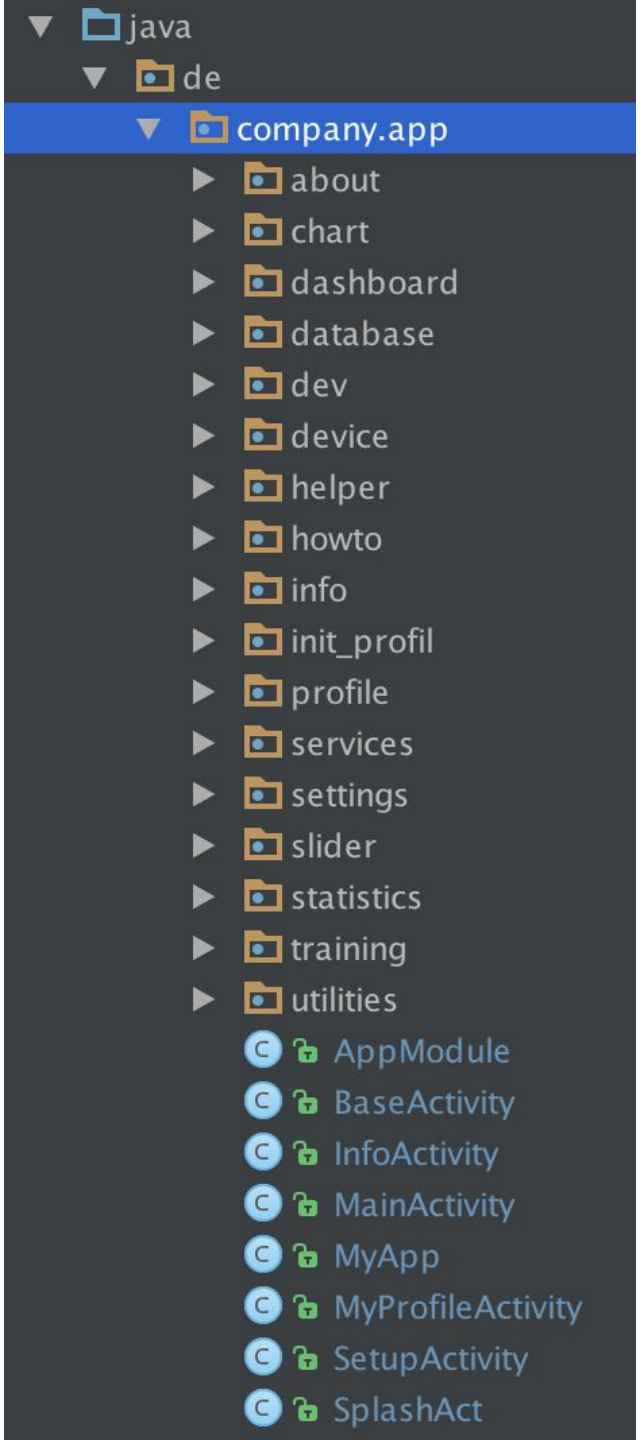

Business
Repository
OverviewManager
BleManager
DeviceRepository
startDeviceScan()
addDevice(xyz)
bus.post(new EventRepoAddedNewDevice)


Business
Repository
OverviewManager
BleManager
DeviceRepository
startDeviceScan()
addDevice(xyz)
bus.post(new EventRepoAddedNewDevice)
bus.post(...)

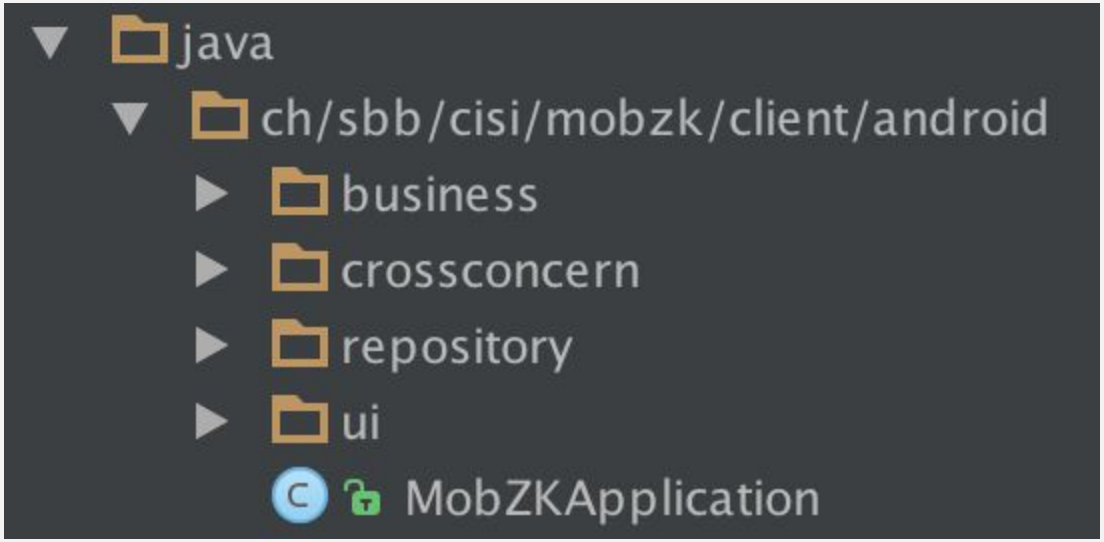

Business
Repository
OverviewManager
BleManager
DeviceRepository
startDeviceScan()
addDevice(xyz)
bus.post(new EventRepoAddedNewDevice)
bus.post(...)
getDevice(address)
getDevice(address)

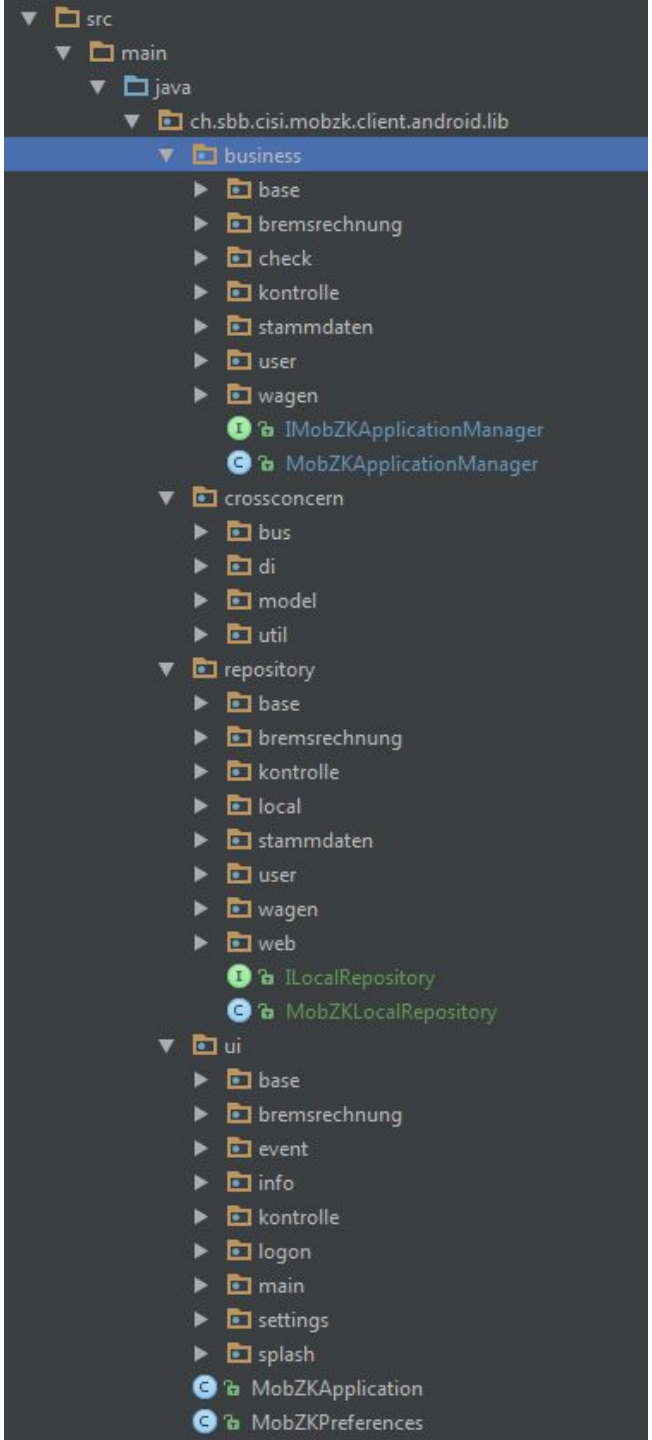
Layers start with packages...

… and packages give structure to source


java
de
company.app
about
chart
dashboard
database
dev
device
helper
howto
info
init_profil
profile
services
settings
slider
statistics
training
utilities
AppModule
BaseActivity
InfoActivity
MainActivity
MyApp
MyProfileActivity
SetupActivity
SplashAct


THE CITY OF YORK
Tourist Information
Treasurer's House
Centenary Way
Museum Gardens
Coach Rendezvous


java
ch/sbb/cisi/mobzk/client/android
business
crossconcern
repository
ui
MobZKApplication


src
main
java
ch.sbb.cisi.mobzk.client.android.lib
business
base
bremsrechnung
check
kontrolle
stammdaten
user
wagen
IMobZKApplicationManager
MobZKApplicationManager
crossconcern
bus
di
model
util
repository
base
bremsrechnung
kontrolle
local
stammdaten
user
wagen
web
ILocalRepository
MobZKLocalRepository
ui
base
bremsrechnung
event
info
kontrolle
logon
main
settings
splash
MobZKApplication
MobZKPreferences

...clarity

Summary

- KISS / SoC are Key
- Clear responsibilities
- Clear concepts & structures

- Architecture is not free but pays out

Thank you!

“No rules are universal”

(except this one)

MobileTechCon

Simon Butscher | SBB CFF FFS
Gabriel Weis | BINOSYS GmbH

Android Architecture

About

- SBB AG
  - http://www.sbb.ch/en/group/jobs-careers/experienced-employees/information-technology.html
  - Contact: simon.butscher@sbb.ch
- Binosys GmbH
  - http://binosys.de/
  - Contact: gabriel.weis@binosys.de
- Code
  - https://github.com/binosys/mtc2016_architecture_sample