KMM + Compose: Multiplatform Love

BEER
{ }
TECH GROUP

PTER !

Brief introduction

A few words about the technologies

- Kotlin Multiplatform
- Jetpack Compose UI
- Compose Multiplatform

What we're going to see

KMM + **Compose Multiplatform** app

- Setup
- Shared code examples
- Final result

Prerequisites

- Android Studio
- Xcode (yes, a Mac too)
- JDK
- Kotlin Multiplatform Mobile plugin
- Kotlin plugin

Suggestion: use **KDoctor** to check all prerequisites for KMM are met.

Template project

- Create a new **Kotlin Multiplatform Mobile** project inside **Android Studio**
- Add **iOS** module and **Xcode** project manually
- Add **Compose Multiplatform** dependencies

or, easier:

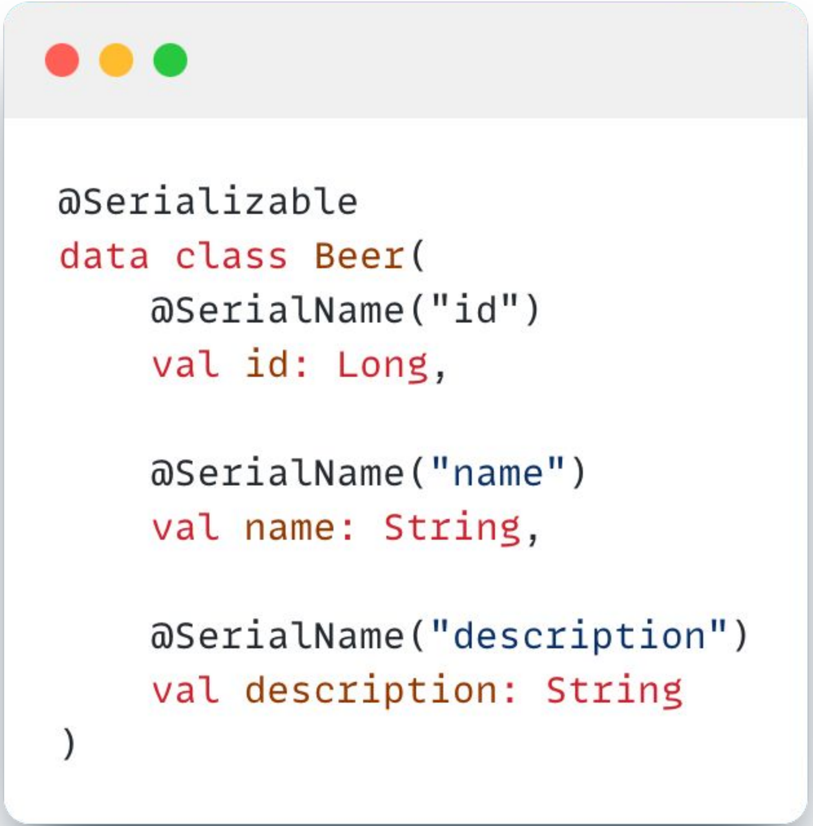
- **Clone** https://github.com/JetBrains/compose-multiplatform-ios-android-template
- Profit

Project structure

- **Android** app module (**/androidApp**)
- **iOS** app native project folder (**/iosApp**)
- **Shared** code module (**/shared**)
    - ***/androidMain*** sub-module
    - ***/iosMain*** sub-module
    - ***/commonMain*** sub-module

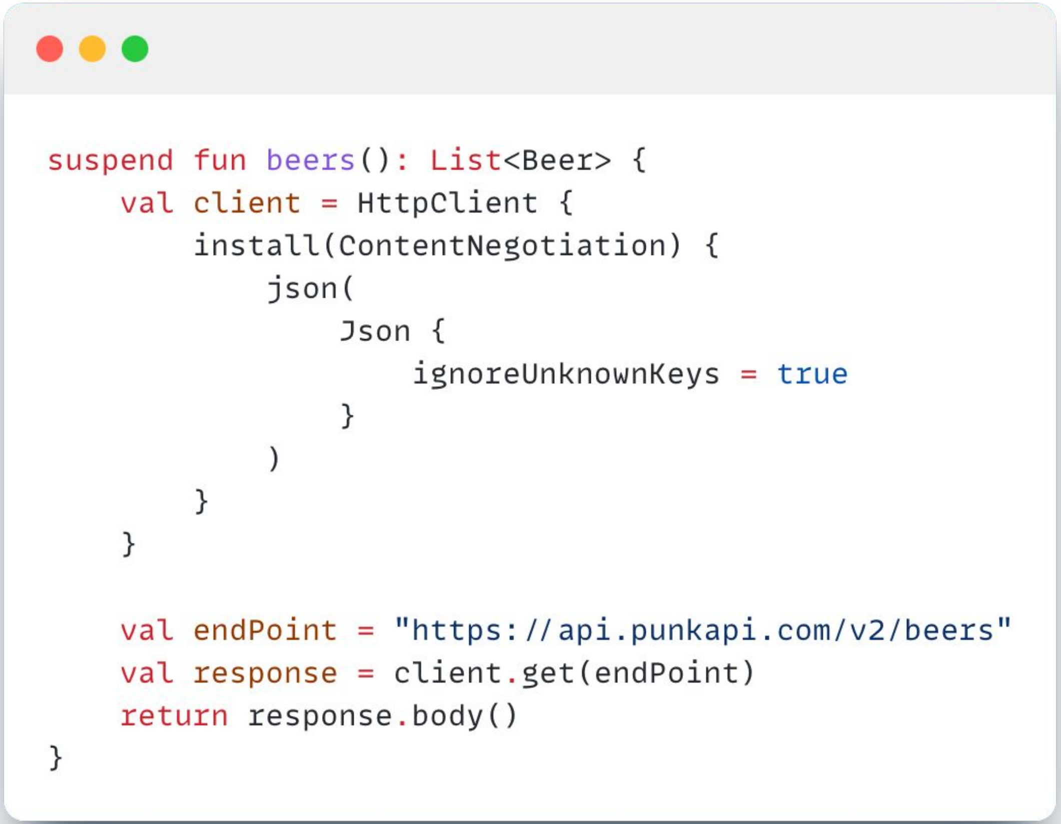
Shared networking

- Punk API
- KotlinX Serialization
- Ktor



```
@Serializable
data class Beer(
    @SerialName("id")
    val id: Long,

    @SerialName("name")
    val name: String,

    @SerialName("description")
    val description: String
)
```

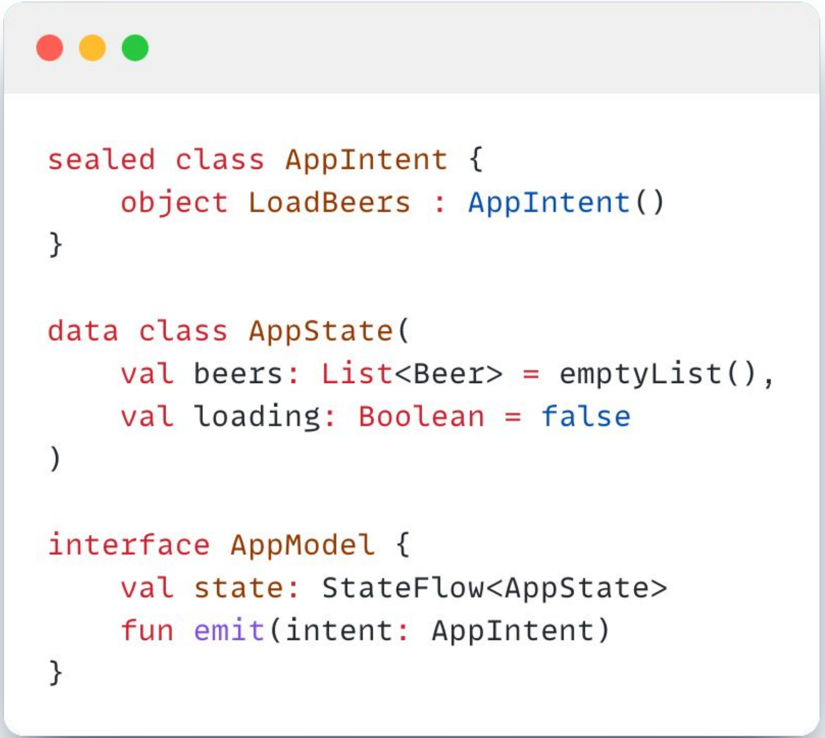
- Ktor **HttpClient** setup
- Network call
- Response deserialization



```
suspend fun beers(): List<Beer> {
    val client = HttpClient {
        install(ContentNegotiation) {
            json(
                Json {
                    ignoreUnknownKeys = true
                }
            )
        }
    }

    val endPoint = "https://api.punkapi.com/v2/beers"
    val response = client.get(endPoint)
    return response.body()
}
```

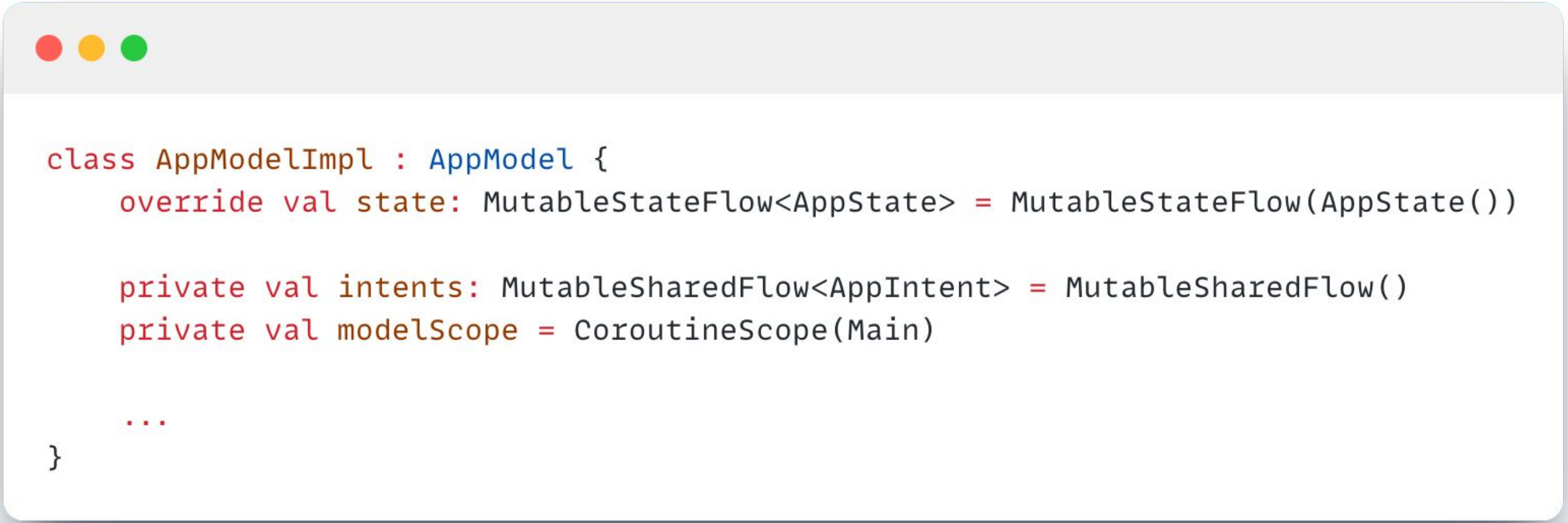
Shared business logic

- Intent definition
- State definition
- AppModel MVI contract



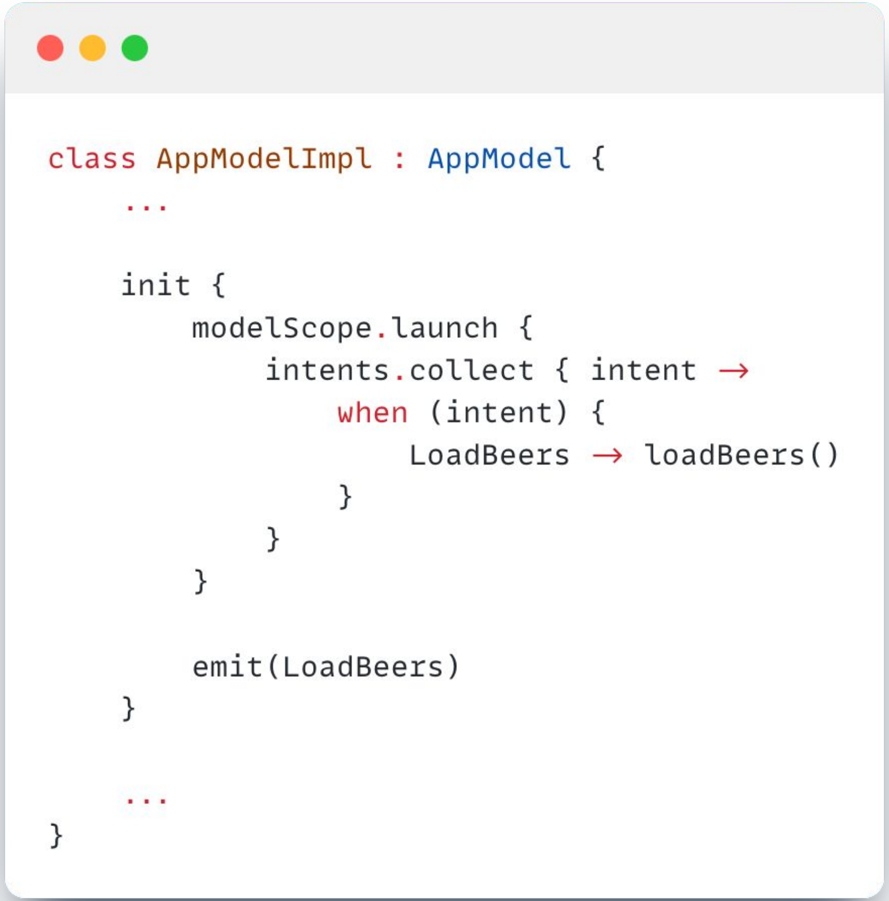
```
sealed class AppIntent {
    object LoadBeers : AppIntent()
}

data class AppState(
    val beers: List<Beer> = emptyList(),
    val loading: Boolean = false
)

interface AppModel {
    val state: StateFlow<AppState>
    fun emit(intent: AppIntent)
}
```

- **State** and **Intent** Flows initialization
- **CoroutineScope** creation



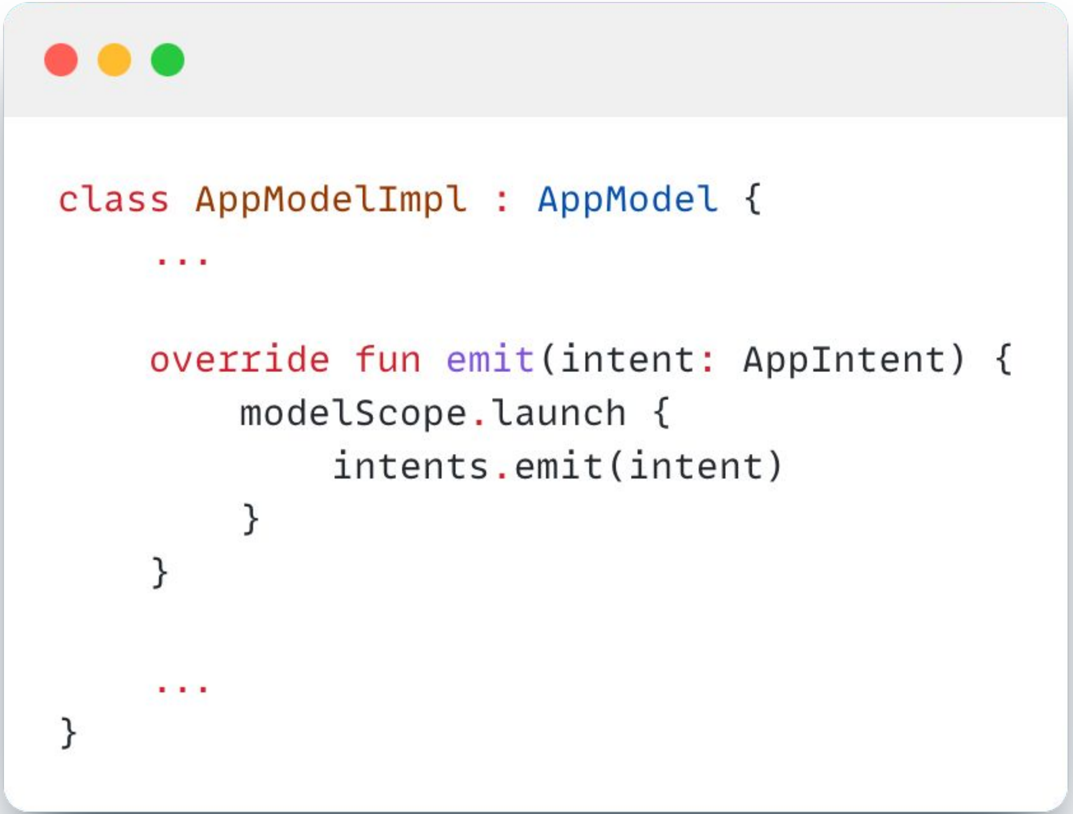
```
class AppModelImpl : AppModel {
    override val state: MutableStateFlow<AppState> = MutableStateFlow(AppState())

    private val intents: MutableSharedFlow<AppIntent> = MutableSharedFlow()
    private val modelScope = CoroutineScope(Main)


    ...
}
```

- **Collect** Intents and respond to each
- Emit **LoadBeers** intent



```
class AppModelImpl : AppModel {
    ...

    init {
        modelScope.launch {
            intents.collect { intent →
                when (intent) {
                    LoadBeers → loadBeers()
                }
            }
        }

        emit(LoadBeers)
    }

    ...
}
```

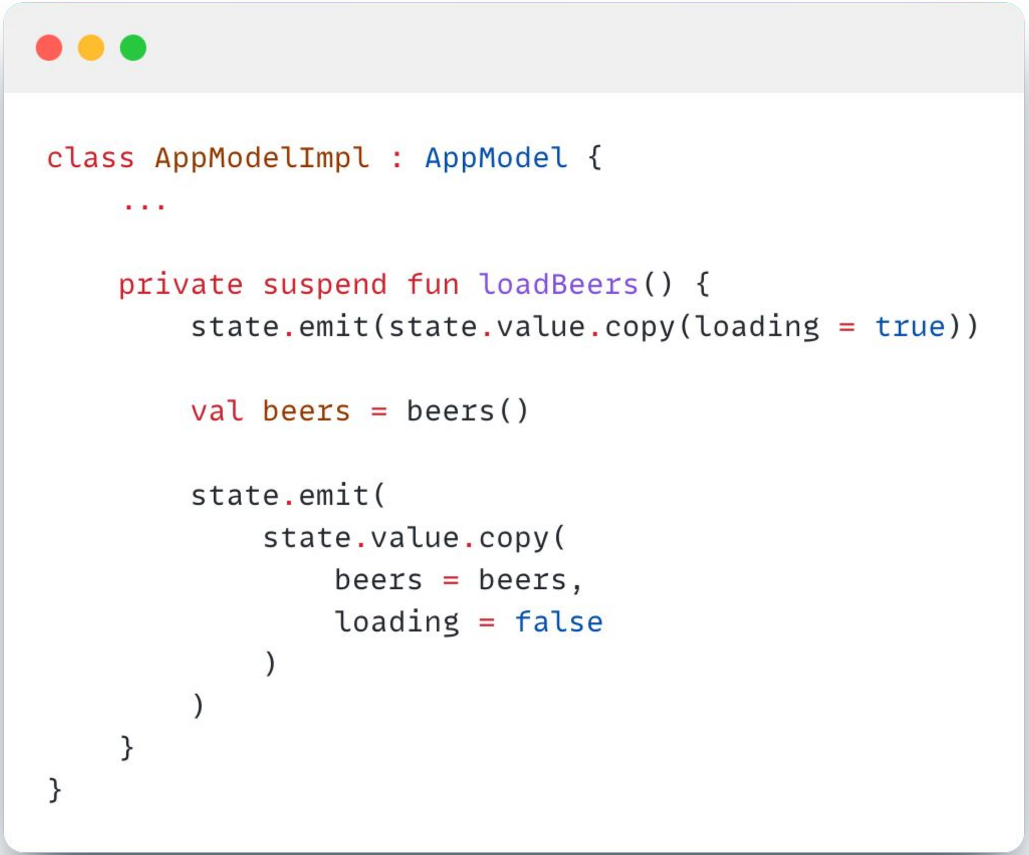
Intent emit implementation



```
class AppModelImpl : AppModel {
    ...

    override fun emit(intent: AppIntent) {
        modelScope.launch {
            intents.emit(intent)
        }
    }

    ...
}
```

loadBeers implementation:

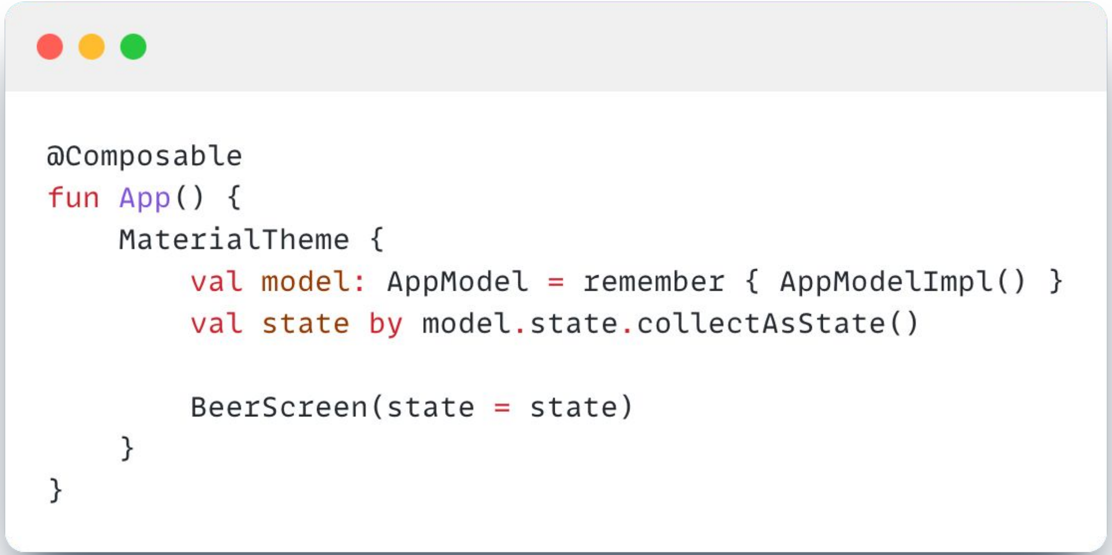
- **Emit State** with loading true
- **Request beers** from the API
- **Emit State** with loaded beers and loading false



```kotlin
class AppModelImpl : AppModel {
    ...

    private suspend fun loadBeers() {
        state.emit(state.value.copy(loading = true))

        val beers = beers()

        state.emit(
            state.value.copy(
                beers = beers,
                loading = false
            )
        )
    }
}
```

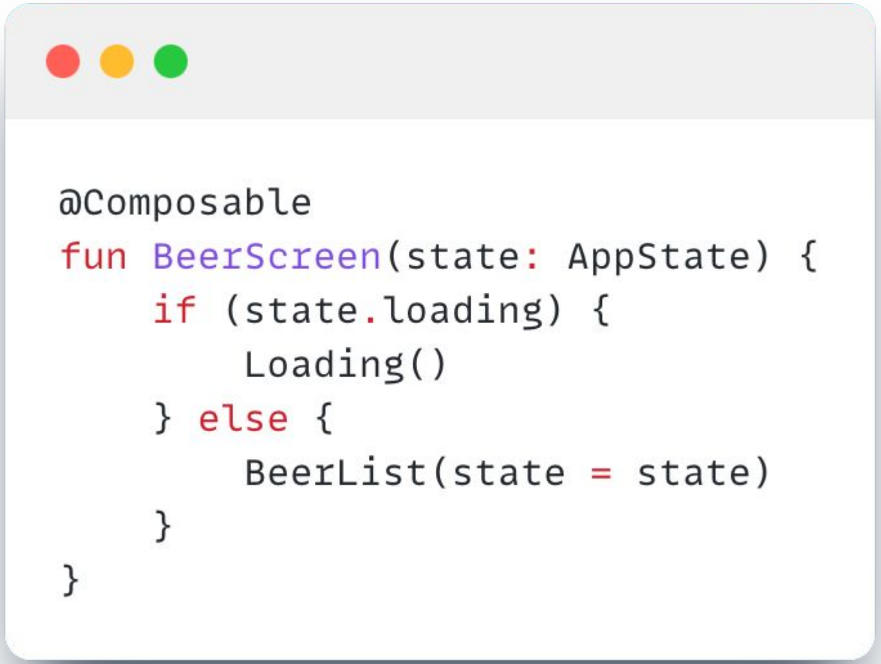
Shared UI

- Initialize and *remember* **AppModel**
- Listen to **State changes**
- Compose **BeerScreen** with latest State



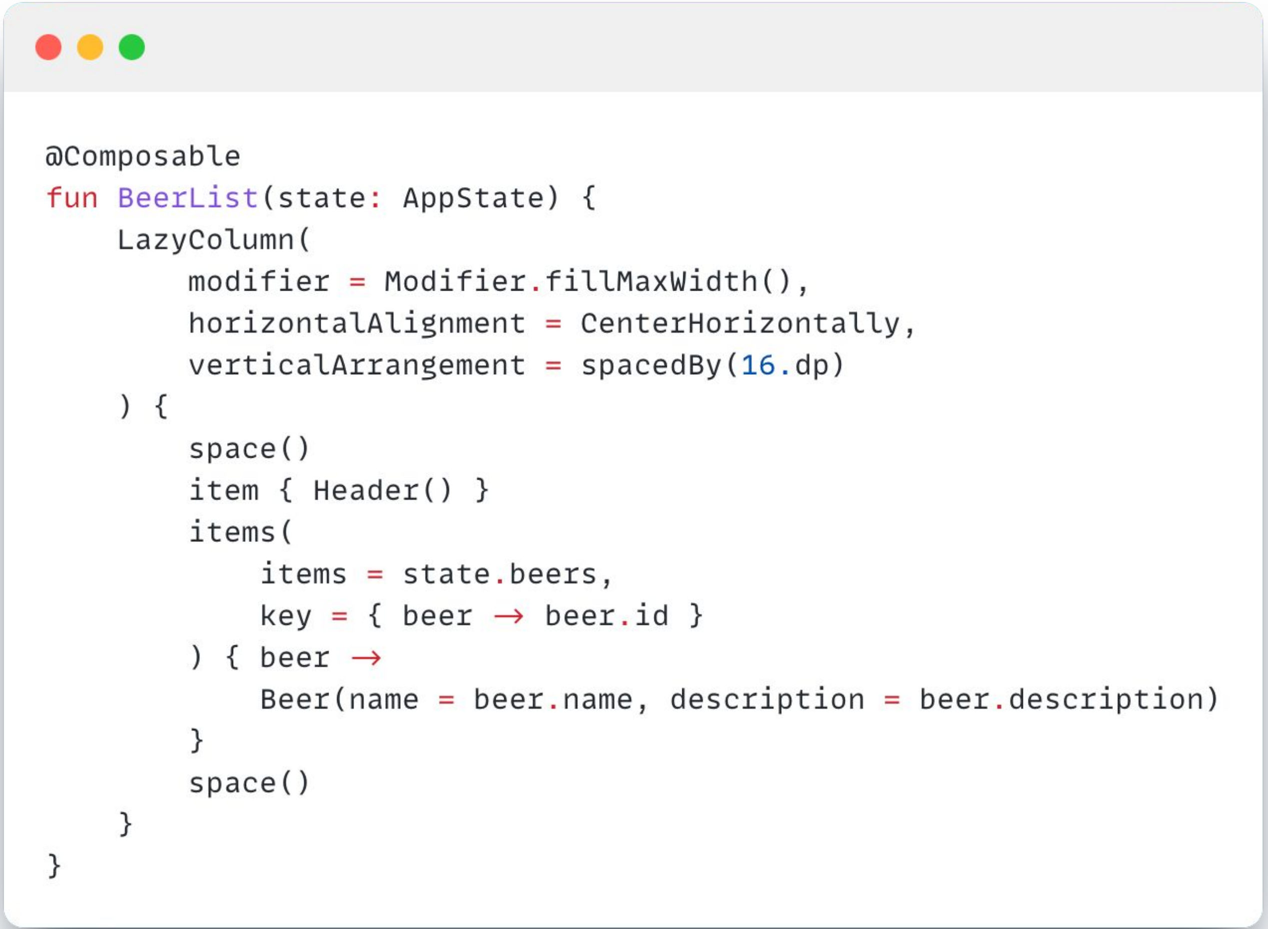
```kotlin
@Composable
fun App() {
    MaterialTheme {
        val model: AppModel = remember { AppModelImpl() }
        val state by model.state.collectAsState()

        BeerScreen(state = state)
    }
}
```

Change composition based on the **loading** state:
if loading is **true**, compose a simple circular loading bar,
if loading is **false**, compose our actual beer list.



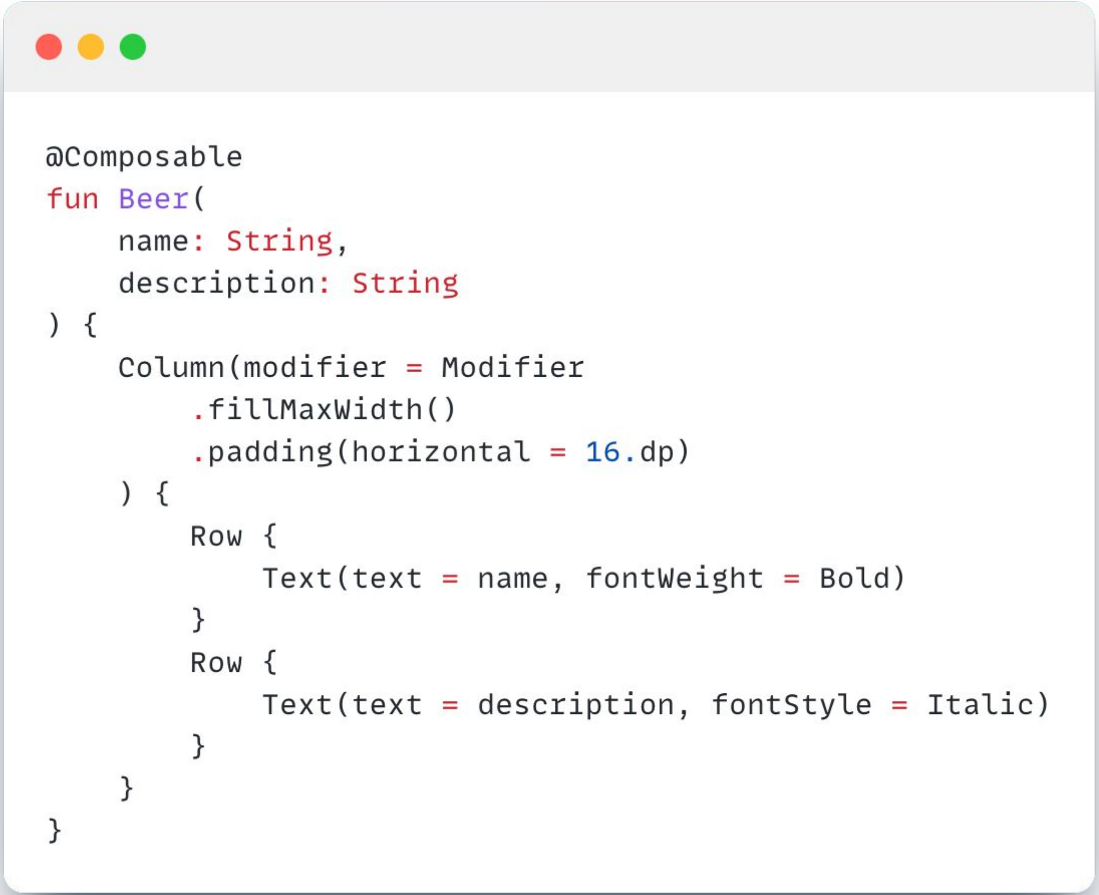
```kotlin
@Composable
fun BeerScreen(state: AppState) {
    if (state.loading) {
        Loading()
    } else {
        BeerList(state = state)
    }
}
```

- Beer list composition
- Decorations



```
@Composable
fun BeerList(state: AppState) {
    LazyColumn(
        modifier = Modifier.fillMaxWidth(),
        horizontalAlignment = CenterHorizontally,
        verticalArrangement = spacedBy(16.dp)
    ) {
        space()
        item { Header() }
        items(
            items = state.beers,
            key = { beer → beer.id }
        ) { beer →
            Beer(name = beer.name, description = beer.description)
        }
        space()
    }
}
```

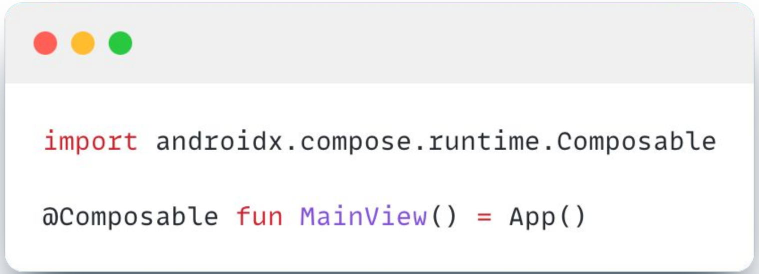
- **Beer item** composition
- **Text** composables



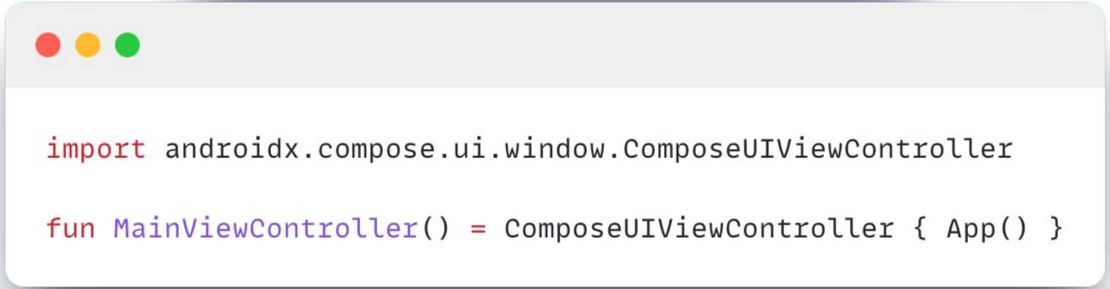
```kotlin
@Composable
fun Beer(
    name: String,
    description: String
) {
    Column(modifier = Modifier
        .fillMaxWidth()
        .padding(horizontal = 16.dp)
    ) {
        Row {
            Text(text = name, fontWeight = Bold)
        }
        Row {
            Text(text = description, fontStyle = Italic)
        }
    }
}
```

Run and result

Use the ***App*** composable in the platform modules **androidMain** and **iosMain**, inside their entry points.



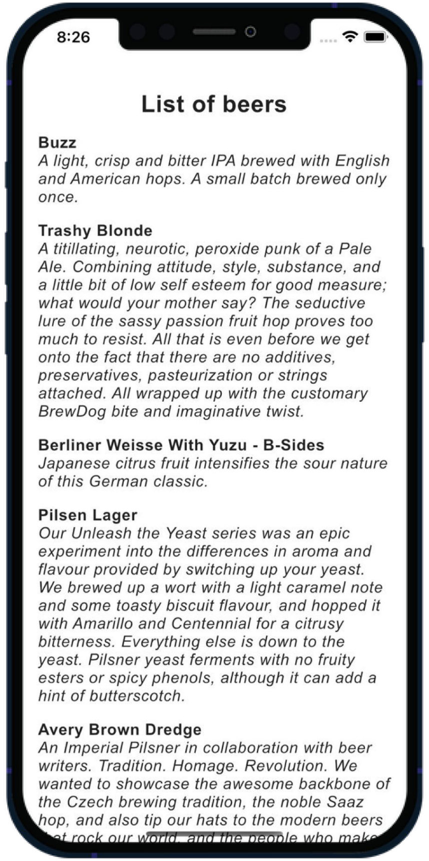
```kotlin
import androidx.compose.runtime.Composable

@Composable fun MainView() = App()
```



```kotlin
import androidx.compose.ui.window.ComposeUIViewController

fun MainViewController() = ComposeUIViewController { App() }
```

Our first, simple (*very simple*) multiplatform app for **iOS** and **Android**, but:

- Shared business logic
- Shared networking
- Shared UI



8:26

List of beers

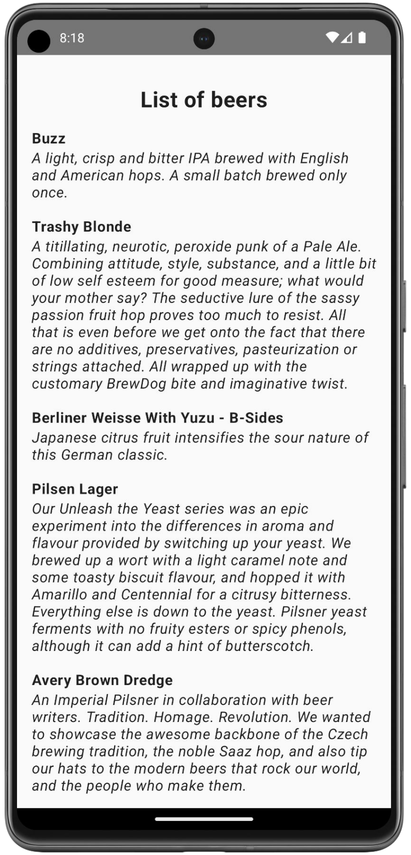
Buzz
A light, crisp and bitter IPA brewed with English and American hops. A small batch brewed only once.

Trashy Blonde
A titillating, neurotic, peroxide punk of a Pale Ale. Combining attitude, style, substance, and a little bit of low self esteem for good measure; what would your mother say? The seductive lure of the sassy passion fruit hop proves too much to resist. All that is even before we get onto the fact that there are no additives, preservatives, pasteurization or strings attached. All wrapped up with the customary BrewDog bite and imaginative twist.

Berliner Weisse With Yuzu - B-Sides
Japanese citrus fruit intensifies the sour nature of this German classic.

Pilsen Lager
Our Unleash the Yeast series was an epic experiment into the differences in aroma and flavour provided by switching up your yeast. We brewed up a wort with a light caramel note and some toasty biscuit flavour, and hopped it with Amarillo and Centennial for a citrusy bitterness. Everything else is down to the yeast. Pilsner yeast ferments with no fruity esters or spicy phenols, although it can add a hint of butterscotch.

Avery Brown Dredge
An Imperial Pilsner in collaboration with beer writers. Tradition. Homage. Revolution. We wanted to showcase the awesome backbone of the Czech brewing tradition, the noble Saaz hop, and also tip our hats to the modern beers at rock our world, and the people who make



8:18

List of beers

Buzz
A light, crisp and bitter IPA brewed with English and American hops. A small batch brewed only once.

Trashy Blonde
A titillating, neurotic, peroxide punk of a Pale Ale. Combining attitude, style, substance, and a little bit of low self esteem for good measure; what would your mother say? The seductive lure of the sassy passion fruit hop proves too much to resist. All that is even before we get onto the fact that there are no additives, preservatives, pasteurization or strings attached. All wrapped up with the customary BrewDog bite and imaginative twist.

Berliner Weisse With Yuzu - B-Sides
Japanese citrus fruit intensifies the sour nature of this German classic.

Pilsen Lager
Our Unleash the Yeast series was an epic experiment into the differences in aroma and flavour provided by switching up your yeast. We brewed up a wort with a light caramel note and some toasty biscuit flavour, and hopped it with Amarillo and Centennial for a citrusy bitterness. Everything else is down to the yeast. Pilsner yeast ferments with no fruity esters or spicy phenols, although it can add a hint of butterscotch.

Avery Brown Dredge
An Imperial Pilsner in collaboration with beer writers. Tradition. Homage. Revolution. We wanted to showcase the awesome backbone of the Czech brewing tradition, the noble Saaz hop, and also tip our hats to the modern beers that rock our world, and the people who make them.

Thanks!

@movement.speed

@movement.speed@mastodon.uno