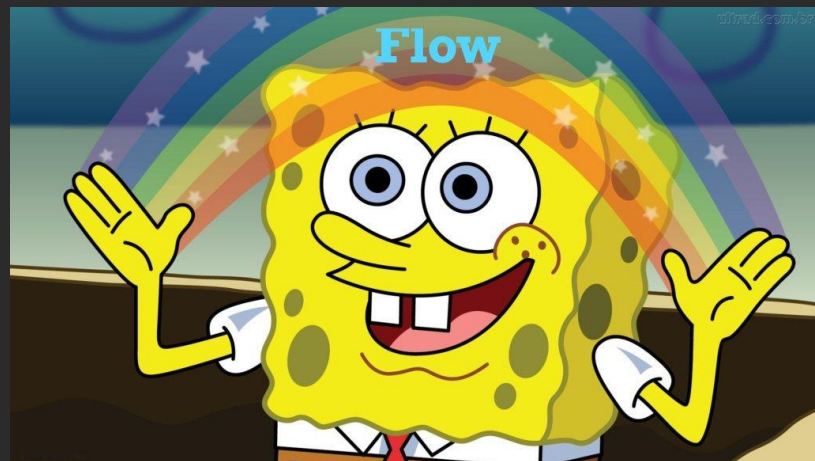


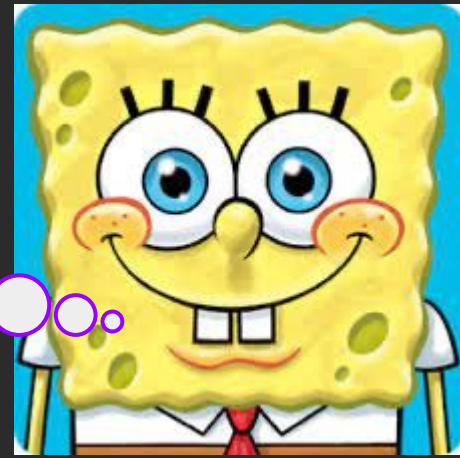


# Flow

*Under the water*



*Happy to show a  
code using  
Flow...*



*Flow... I don't  
have this  
flavor*



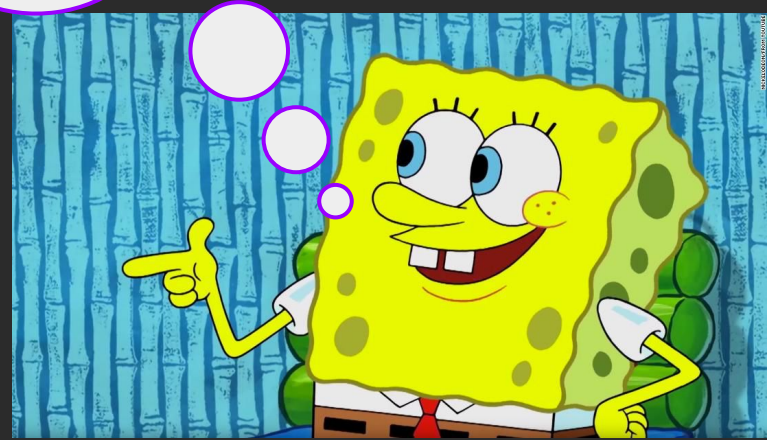
*Patrick it is  
not a flavor 🧊  
I will explain  
better*

```
data class Article(  
    val id: UUID,  
    val title: String,  
    val abstract: String,  
    val authorName: String,  
    val publishedAt: Date  
)
```

```
data class Author(  
    val id: UUID,  
    val name: String,  
    val researchArea: String,  
)
```

*I'm doing an app to  
show all articles we  
have created here in  
Bikini Bottom.*

*These are the model  
classes...*



```
class ArticlesProducer {  
    fun articles(): Flow<List<Article>> {...}  
}  
  
class AuthorProducer {  
    fun authors(): Flow<List<Author>> {...}  
}
```

Creating the  
**producers** to  
fetch the  
information...

**Producer**

Intermediary

Consumer

```
class ArticlesIntermediary {  
  
    private val articlesProducer = ArticlesProducer()  
    private val authorsProducer = AuthorProducer()  
  
    fun articles(): Flow<List<ArticleFullInformation>> {  
        return articlesProducer.articles().combine(authorsProducer.authors()) { articles, authors →  
            articles.map { article →  
                ArticleFullInformation(  
                    article = article,  
                    author = authors.first { author → author.name == article.authorName }  
                )  
            }  
        }  
    }  
}
```



*I created an intermediary  
because I need the full  
information about the  
author, the API doesn't  
provide.*

```
class ArticlesIntermediary {  
  
    private val articlesProducer = ArticlesProducer()  
    private val authorsProducer = AuthorProducer()  
  
    fun articles(): Flow<List<ArticleFullInformation>> {  
        return articlesProducer.articles().combine(authorsProducer.authors()) { articles, authors →  
            articles.map { article →  
                ArticleFullInformation(  
                    article = article,  
                    author = authors.first { author → author.name == article.authorName }  
                )  
            }  
        }  
    }  
}
```

```
data class ArticleFullInformation(  
    val article: Article,  
    val author: Author  
)
```



```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        lifecycleScope.launch { this: CoroutineScope
            ArticlesIntermediary().articles().collect { it: List<ArticleFullInformation>
                Log.d(TAG, msg: "-----")
                it.forEach { articleFullInfo →
                    Log.d(
                        TAG,
                        msg: "Article: " +
                            "${articleFullInfo.article.title}\n, " +
                            "Author: " +
                            "${articleFullInfo.author.id} ${articleFullInfo.author.name}"
                    )
                }
            }
        }
    }
}

```

Producer


Intermediary

Consumer



}



A cartoon illustration of SpongeBob SquarePants and Patrick Star in Bikini Bottom. Patrick is inside a large, dark brown, round object that looks like a giant chocolate chip or a large bug. SpongeBob is standing on the sandy ocean floor, looking at Patrick. There are some small pieces of food and a green can on the sand. The background is a blue gradient representing the ocean. Two thought bubbles are present: one from Patrick and one from SpongeBob.

Hey Spongebob,  
I found a bug  
in your app  
yesterday.

Patrick, oh no!  
Explain please!



I had the app opened.  
Squidward said you wrote a  
new article, but the app  
didn't show when I opened  
it.



Yes, the app is  
not working

I will fix. Give  
me few seconds!



```

lifecycleScope.launch { this: CoroutineScope
    repeatOnLifecycle(Lifecycle.State.STARTED) { this: CoroutineScope
        launch { this: CoroutineScope
            ArticlesIntermediary().articles().collect { it: List<ArticleFullInformation>
                Log.d(TAG, msg: "-----")
                it.forEach { articleFullInfo →
                    Log.d(
                        TAG,
                        msg: "Article: " +
                            "${articleFullInfo.article.title}\n, " +
                            "Author: " +
                            "${articleFullInfo.author.id} ${articleFullInfo.author.name}"
                    )
                }
            }
        }
    }
}

```

*Restartable behavior  
implemented!*

