Ballerina
Swan Lake

# Modern Compiler Trends: Insights from the Ballerina Compiler

Hasitha Aravinda | hasitha@wso2.com
Architect - WSO2


Why Are You Studying Compiler Theory?
Ballerina
Swan Lake


Why Might You Build or Work on a Compiler in the Future?
Ballerina
Swan Lake

# What Defines a Compiler Today?

Ballerina
Swan Lake

# Modern Compilers Are No Longer Standalone Applications

# Compiler Ecosystem

- IDEs. i.e. VSCode, IntelliJ Idea
- Code Scanners
  - Security Scanners:
  - Linters, Spot Bug, Sonar Scan, Check Style
- Tools
  - Code Formatters, Code Visualizers, Artifact Generators, Report Generators
- Package Management System
- Cross-Compilers

and many more…

# Modern Compiler Trends

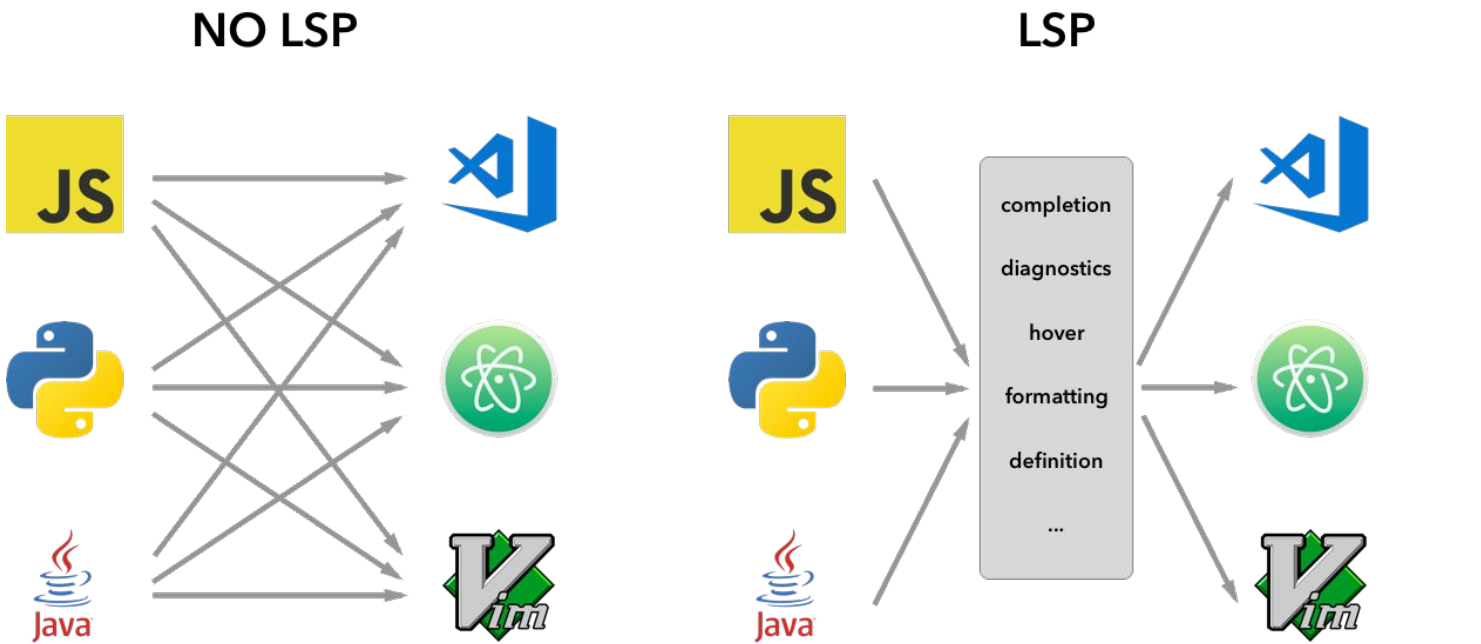
# LSP
# Language Server Protocols

# LSP

- Created by Microsoft
- Standardize the protocol for how language development tools communicate with each other.
- Encourage reusability.
- Features
  - Auto complete
  - Go to definition
  - Documentation on hover
  - Refactoring

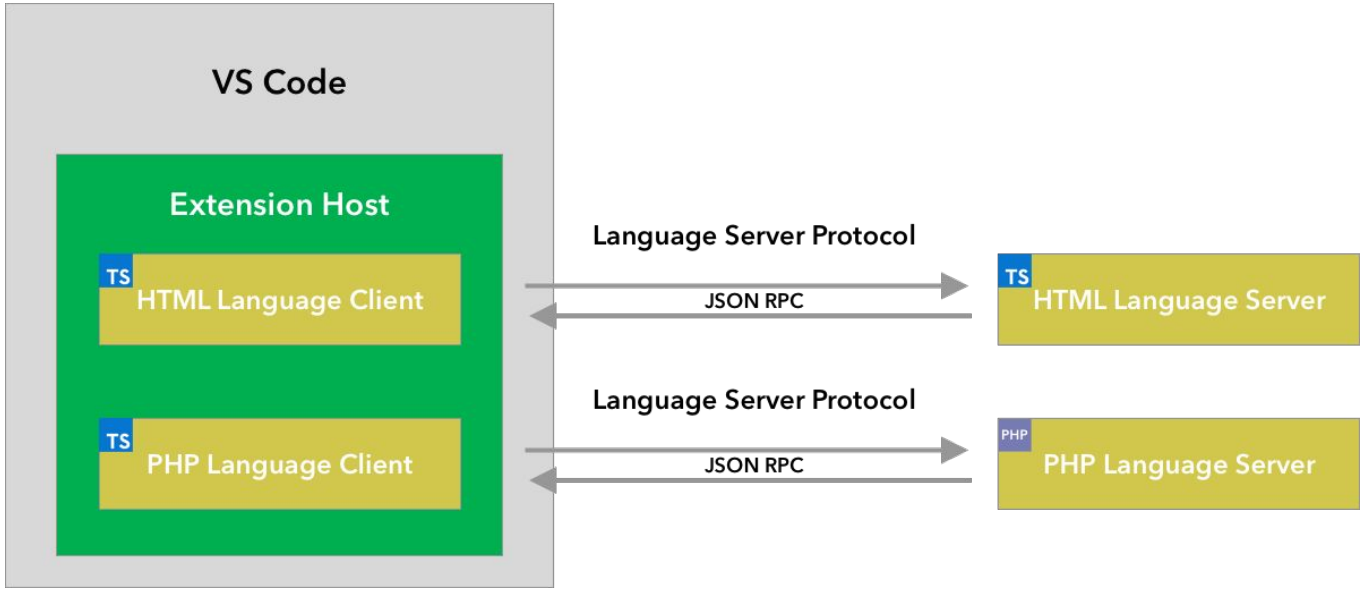
  and many more...

# LSP



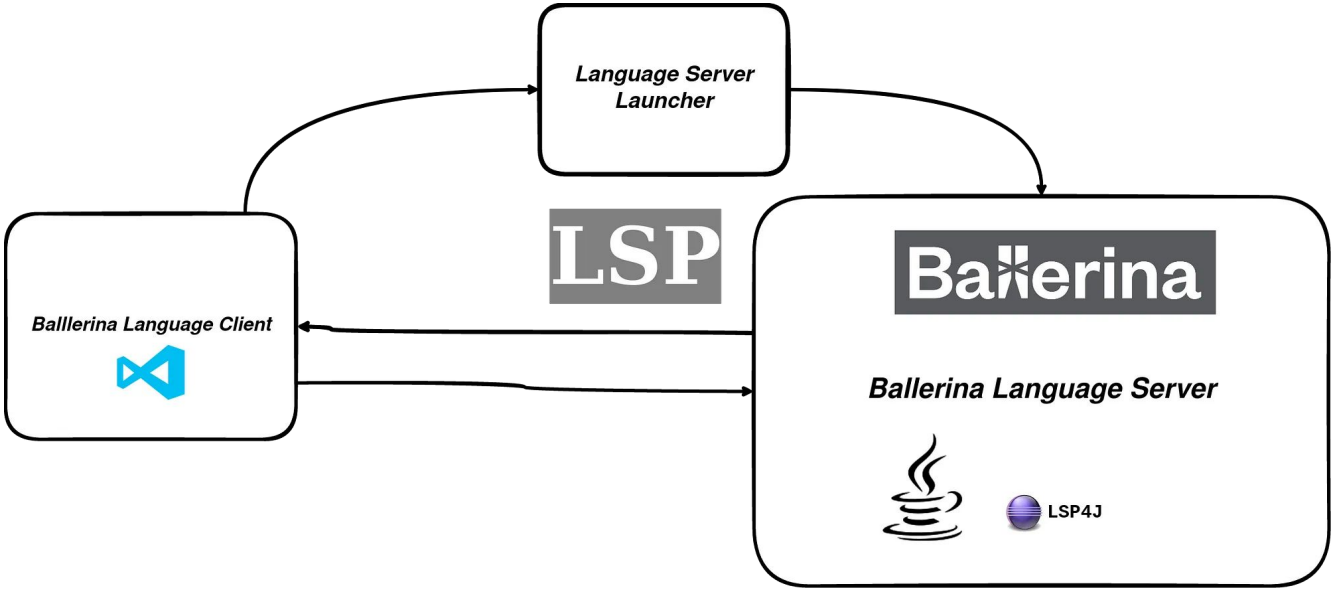

Credits: https://code.visualstudio.com/api/language-extensions/language-server-extension-guide

# LSP




Credits: https://code.visualstudio.com/api/language-extensions/language-server-extension-guide

# Ballerina LSP




Credits: https://medium.com/ballerina-techblog/practical-guide-for-the-language-server-protocol-3091a122b750

# WebAssembly (Wasm)

# Wasm

- **Binary Instruction Format**: Wasm is a binary instruction format designed for a stack-based virtual machine.
- **Compilation Target**: It serves as a portable target for compiling high-level languages like C, C++, and Rust, enabling them to run on the web.
- **Performance**: Provides near-native execution speed, making it suitable for performance-critical applications such as games and complex UIs.
- **Security**: Designed to maintain the security guarantees of the web, ensuring safe execution within a sandboxed environment in web browsers.
- **Platform Independence**: Wasm is platform-independent, facilitating consistent behavior across different systems and devices.

# Examples

- Run Linux or other Operating Systems in your browser!
  - https://bellard.org/jslinux/
- Web AutoCad
  - https://web.autocad.com/
- RustPython
  - https://rustpython.github.io/demo/
- More Applications: https://madewithwebassembly.com/

# AI and Machine Learning

# AI and Machine Learning

- Multiple Areas
  - Context aware completion
  - Bug detection
  - Code optimization
  - Natural language to Code
  - Personalization

Tools

  - VSCode Copilot
  - https://cursor.sh/
  - OpenAI's ChatGPT and APIs

Many more

# Cloud Based Development

# Cloud Based Development

- Developer platform as a service
- Cloud compilers as a service

- Advantages
  - Accessibility, Collaboration, Scalability, Reduced Hardware Cost
- Disadvantages
  - Internet dependency,  Security, Limited features, Performance, Cost over time.
- Products
  - Github Actions
  - Github Codespaces
  - AWS Cloud9
  - WSO2 Choreo

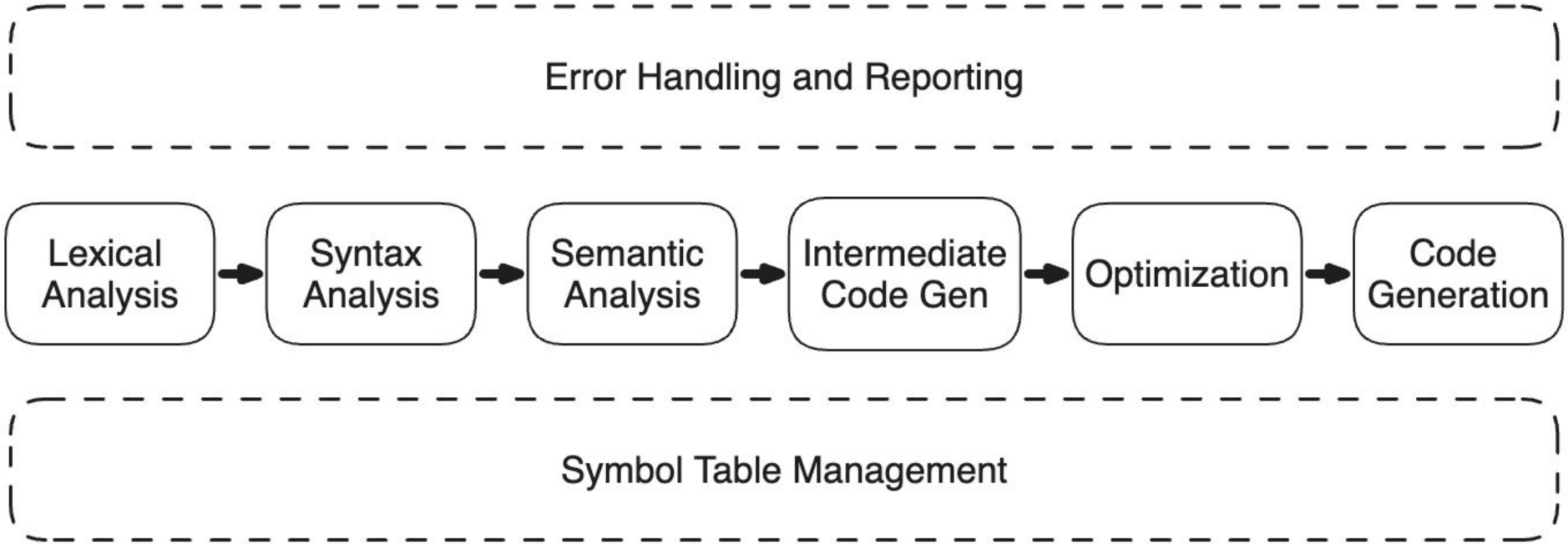
# Other Trends

- Cross-Language Interoperability
- DSL
- Quantum Computing Language Development
- Adaptive Optimization
- Energy Efficient
- Secure Compilation

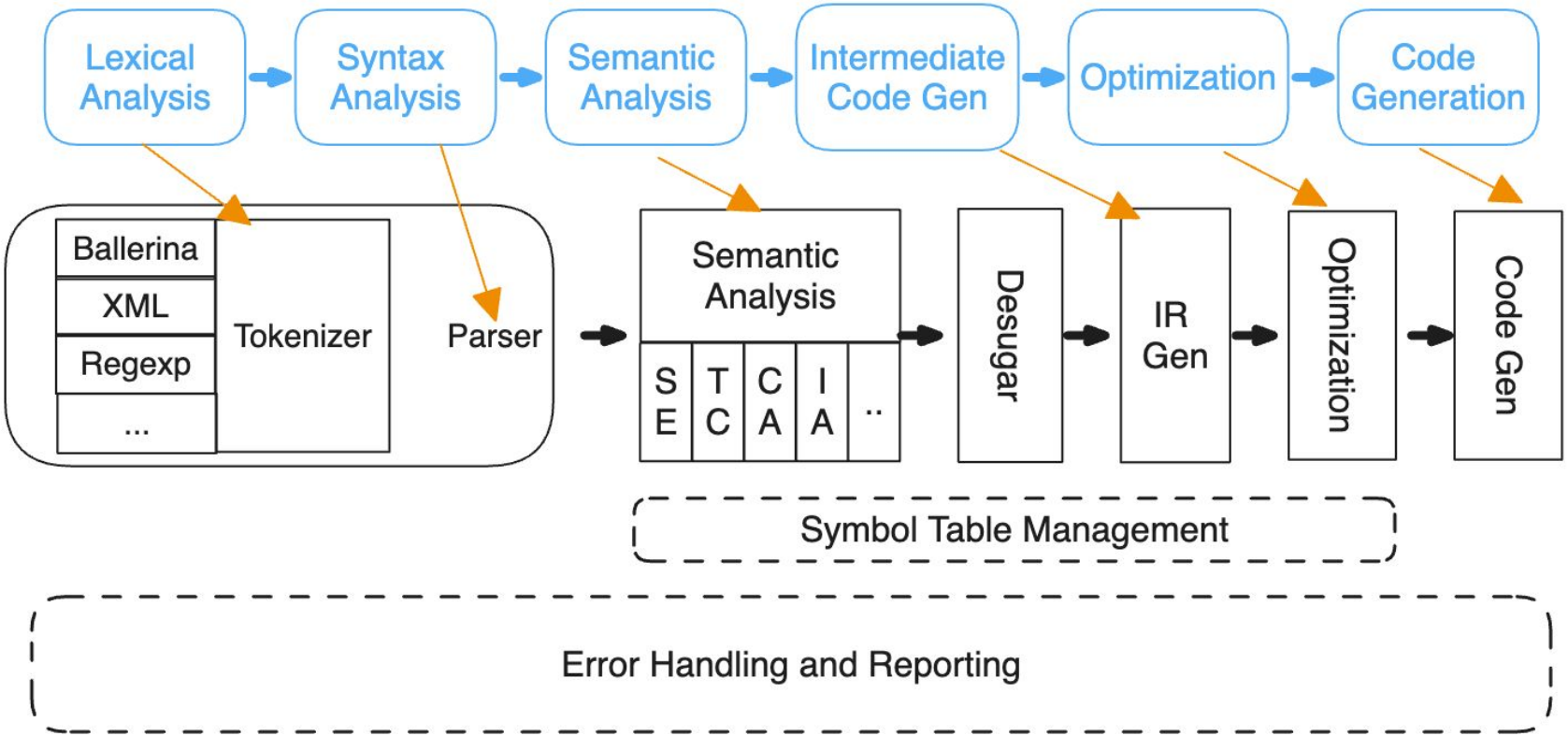
and more …

# Extending Ballerina Compiler

# Compilation Phases As Discussed In The Dragon Book



# Ballerina Compiler - Compilation Phases



# Feature Aware Mini Phases

- Rather than having single monolithic phase, we break phases into small phases based on requirements
- Semantic Analysis
  - Constant Analysis - resolving compile time constants
  - Symbol Enter - resolving module level constructs, types, variables, functions, etc.
  - Worker Analysis - resolving worker(concurrent constructs) interactions
  - Semantic Analysis* - resolving statements
  - Type Checker - resolving expressions
  - Code Analysis - resolving reachability
  - Isolate Analysis - checking concurrency.

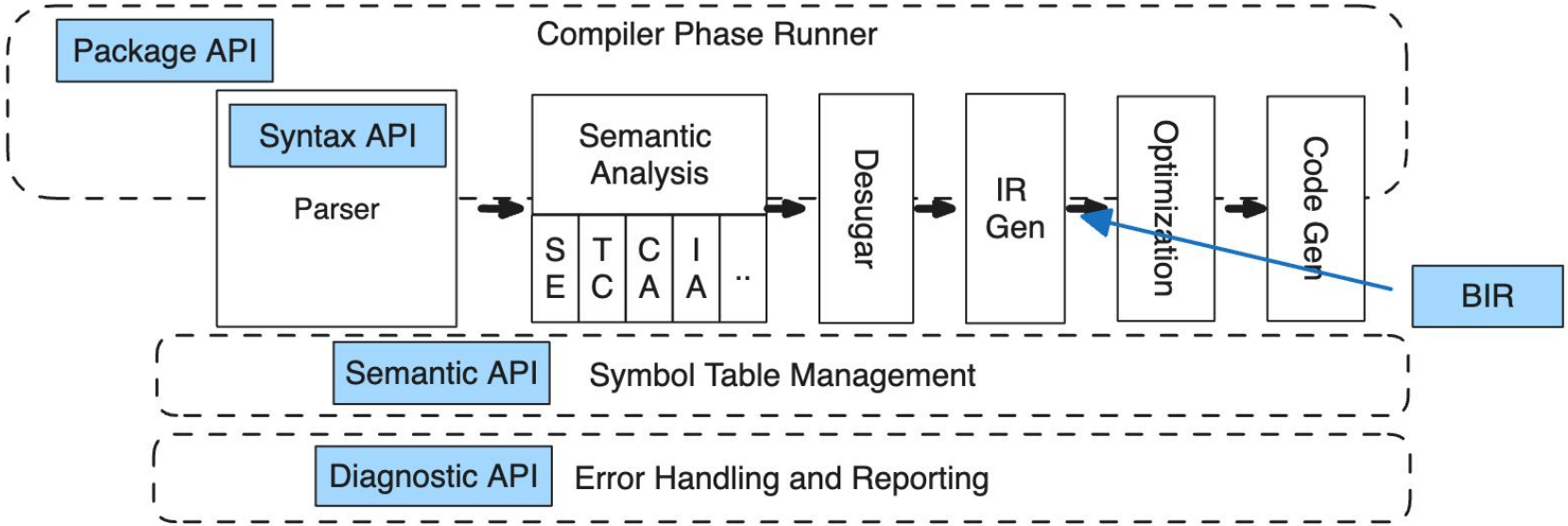
  and more.

# Interutable Phases

- Redesign Phases such that, we can tell upto which point compilation should run.
- Introduced a compilation phase runner to manage the compilation requests.

# Reusable Phases and Results

- Redesign phases such that, result of each phase can be reuse.
- Introduced Compiler APIs.
- Use Generated intermediate code (**BIR**) as the lowered version.

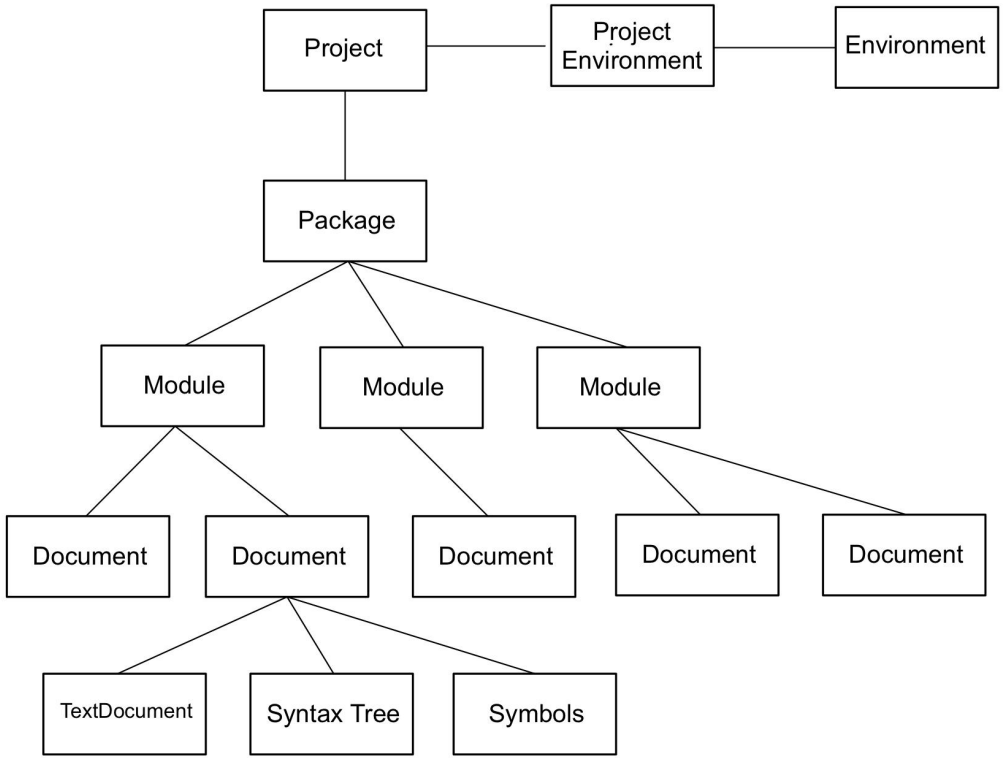


# Use Cases

# Use Case 1: Supporting Multiple Modules

- Package is a logical source code grouping.
- Developers compiles a project (using CLI).
- A package contains one or more ballerina modules.
- Modules are reusable components. i.e `import ballerina/http`
- A module is Ballerina's unit of compilation (CompilationUnit).
  - Compiler phases run against a module.
- Package Structure
  - ➢ Parent module
  - ➢ Zero or More sub Modules
- Modules can have inter-dependencies, but no cyclic.
- One executable per package, that is for the parent module.

# Use Case 1: Supporting Multiple Modules



# Use Case 1: Supporting Multiple Modules

Example: `ballerina/graphql` package

- graphql
- graphql.dataloader
- graphql.subgraph

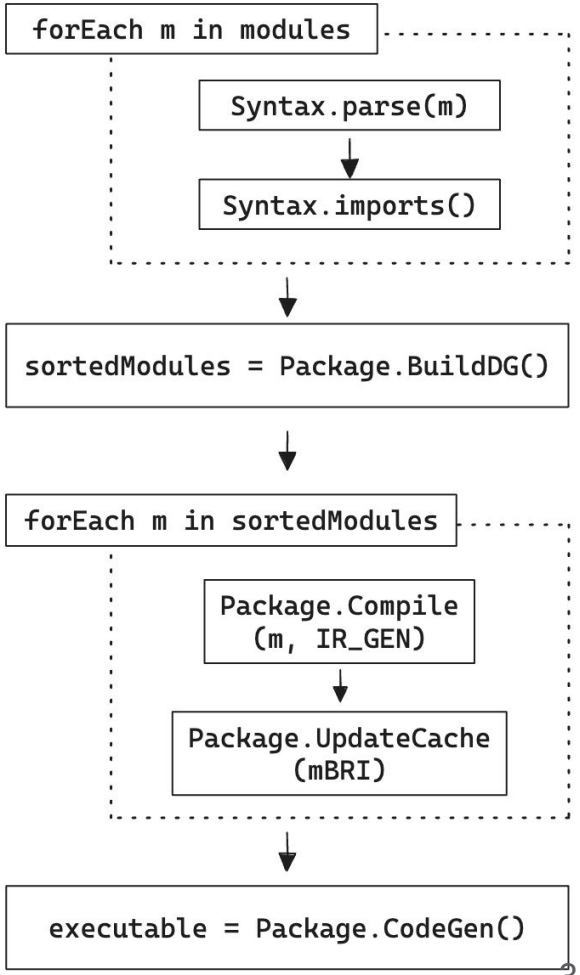
Problems we have to solve.

- Multiple compilation units ⇒ Multiple compilations
- Decide compilation order
- Reusable Symbol Table Entries
- Single executable ⇒ Merge compilation results.

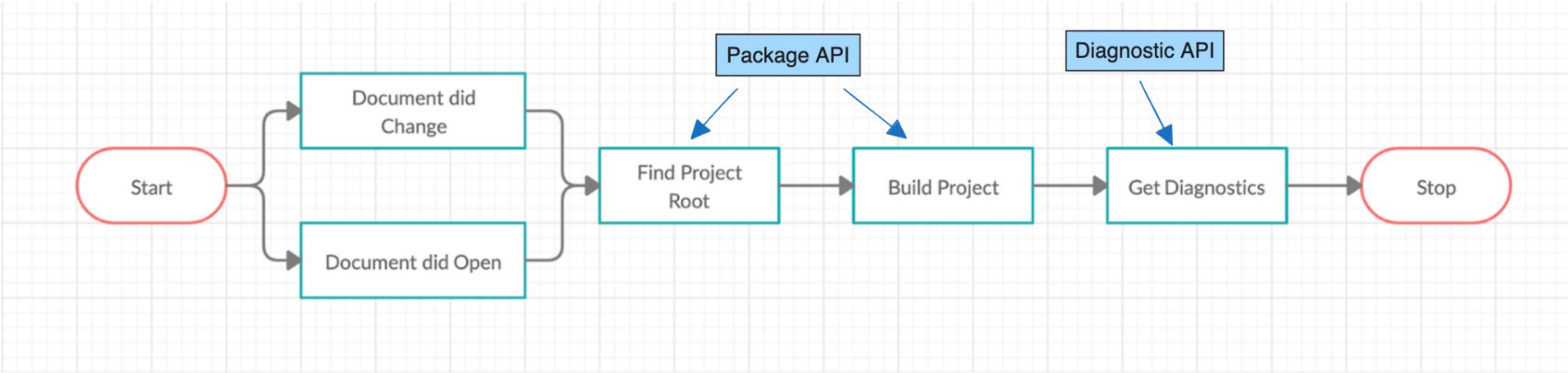
# Use Case 1: Supporting Multiple Modules

## Our Solution

- Use the syntax API to get list of import statements for each module.
- Use this data build the dependency graph
- Execute the compilation phases upto BIR generation, from bottom to top of the dependency graph.
- Reuse BIRs of compiled modules
- At the parent module, execute full compilation to get the executable.

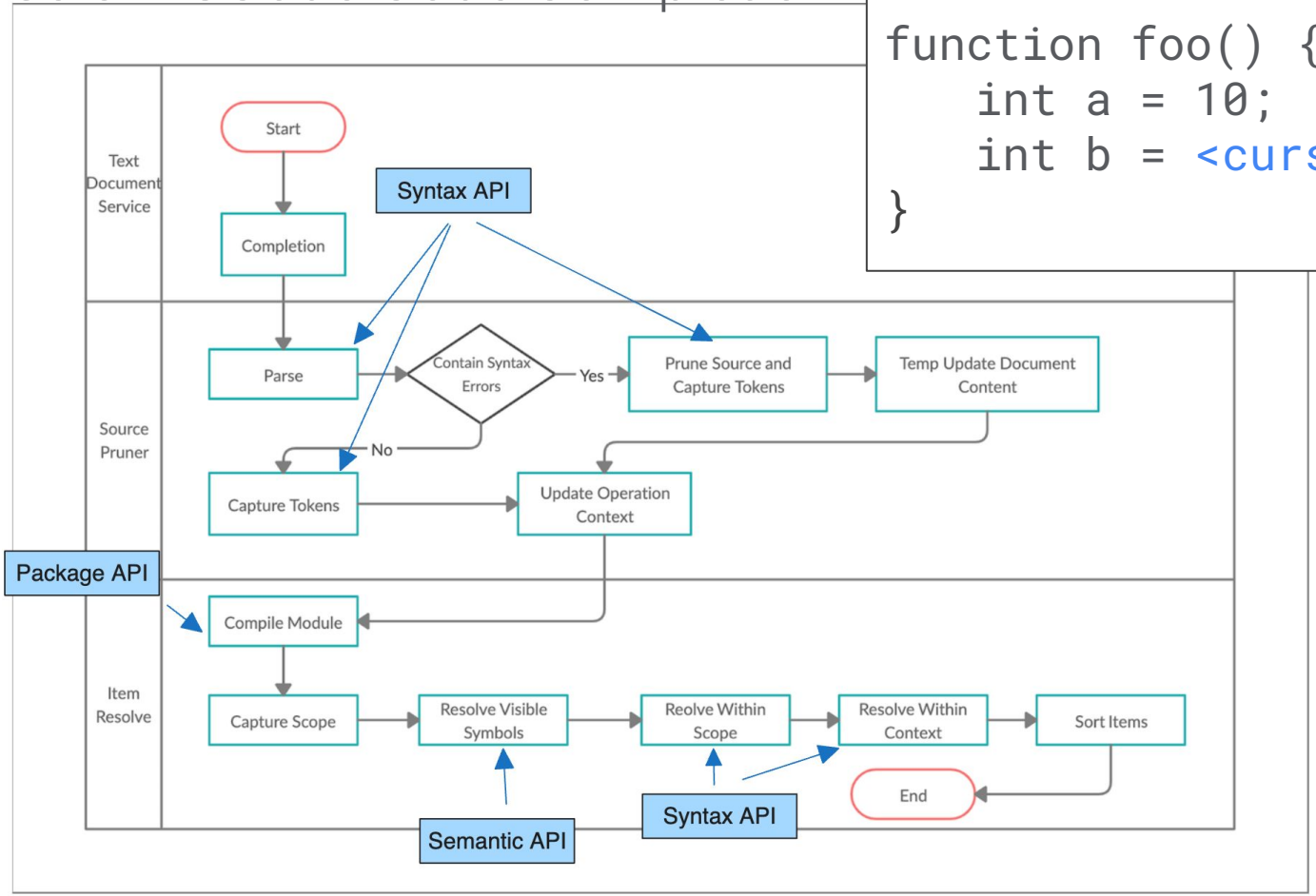


# Use Case 2: VSCode Show Diagnostics



# Use Case 3: VSCode Code Completion

```
function foo() {
    int a = 10;
    int b = <cursor>
}
```

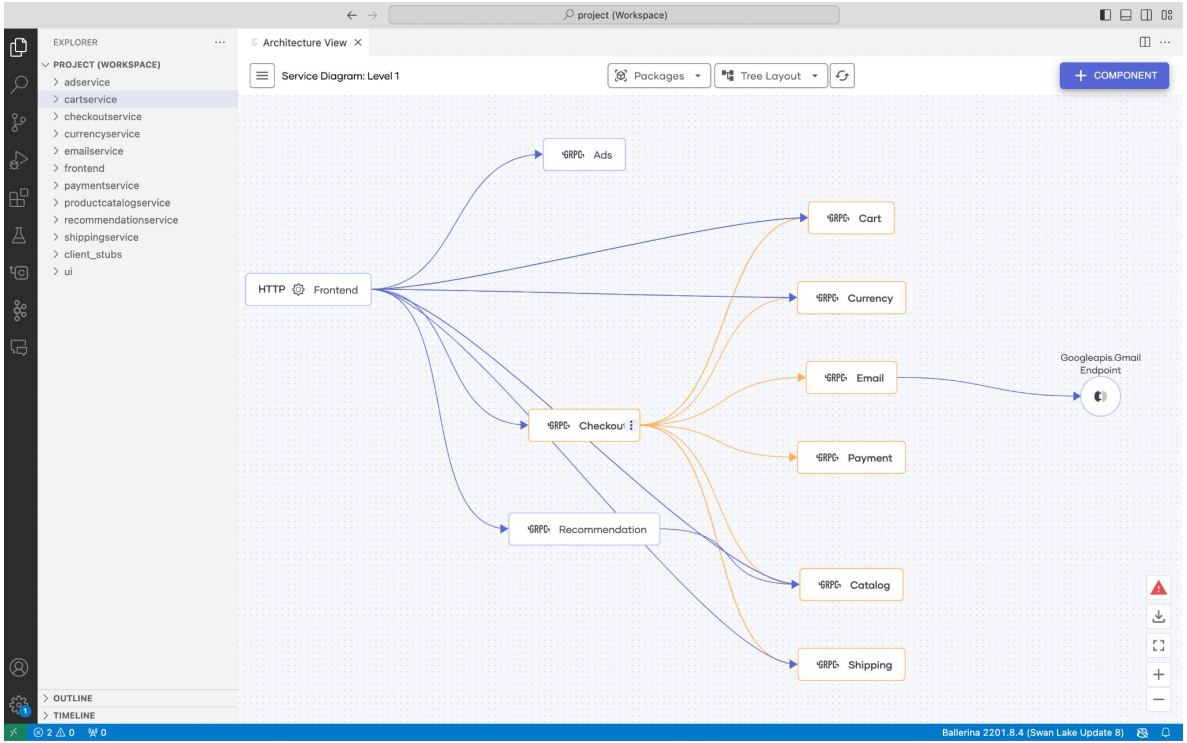


# Use Case 4: Compiler Extensions

- Extend language semantics
- Kinds
  - Validate Code
  - Modify Code
  - Generate New Code and Artifacts
- Examples
  - Code Validation
    - Http resource function can have only certain parameters.
    - Validate annotation attachments.
  - Code Modifications
    - Attached OpenAPI specification as attachment.
  - Code Generation
    - Generate docker and k8s artifacts.

# Use Case 5: Visual Features

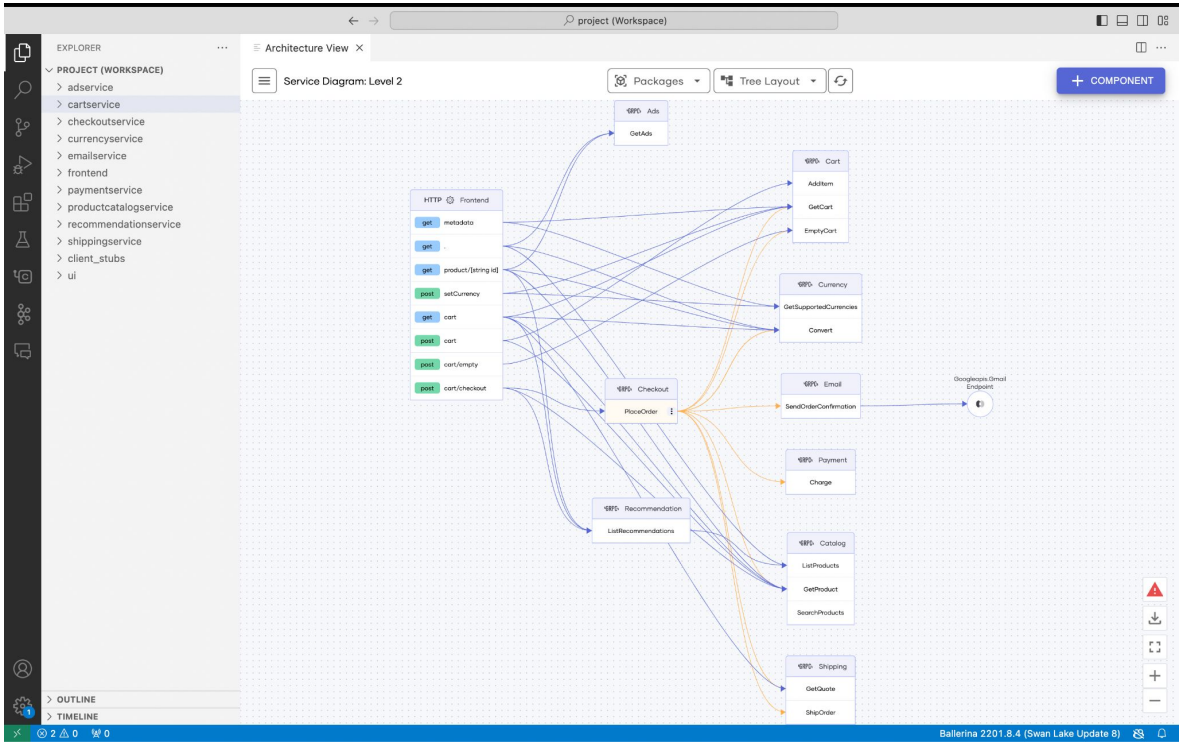
## Architectural design view - Services




Ballerina

# Use Case 5: Visual Features

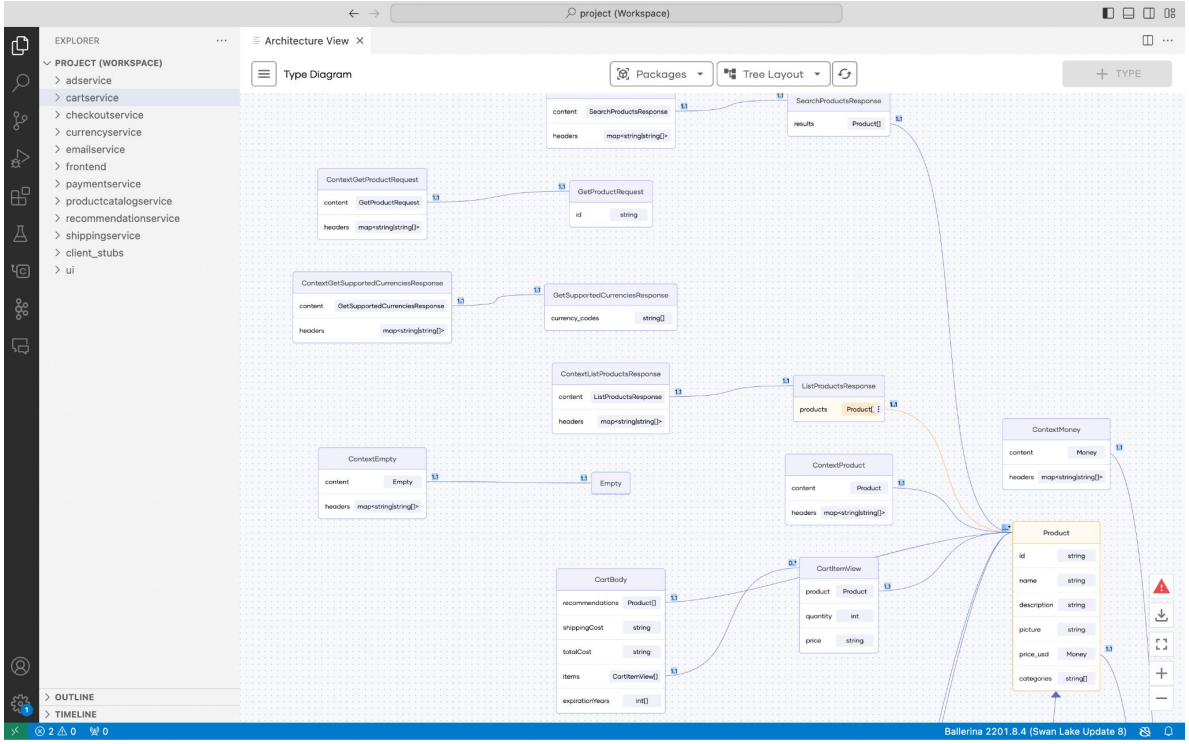
## Architectural design view - Resources




Ballerina

# Use Case 5: Visual Features

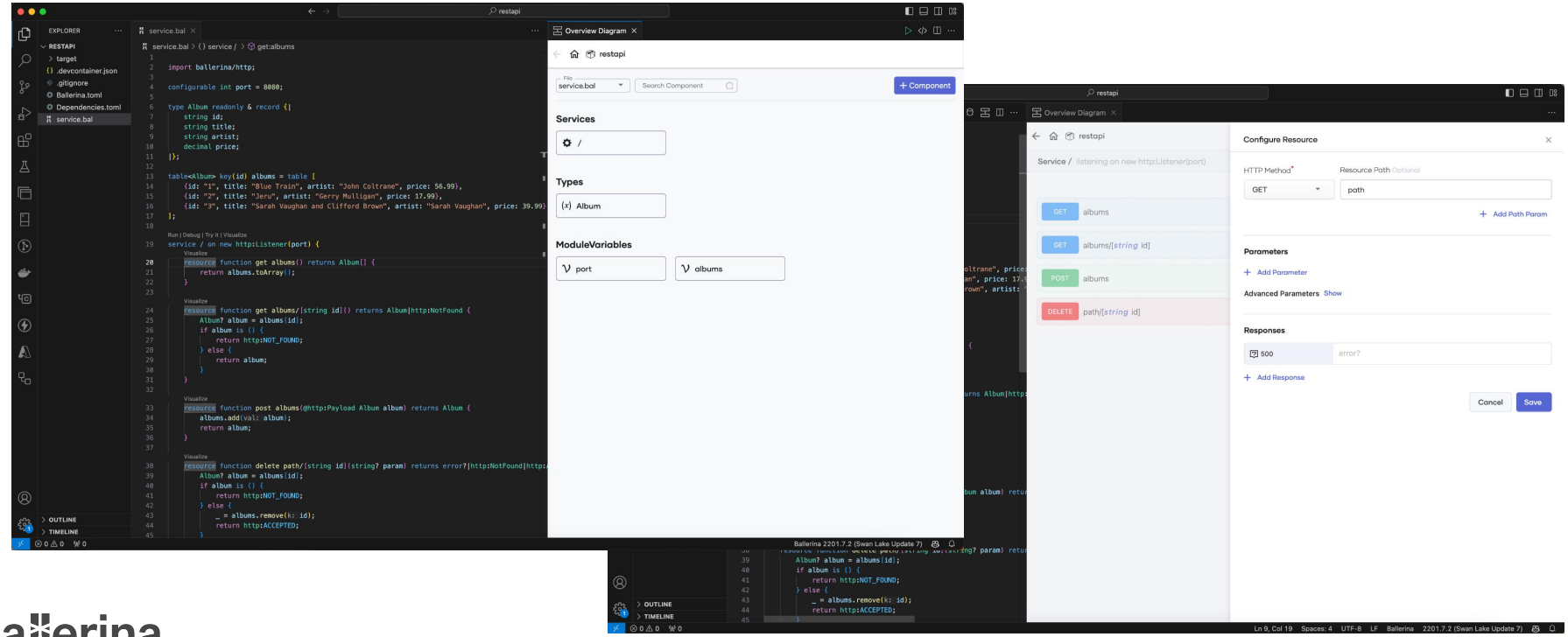
## Architectural design view - Types




Ballerina

# Use Case 5: Visual Features

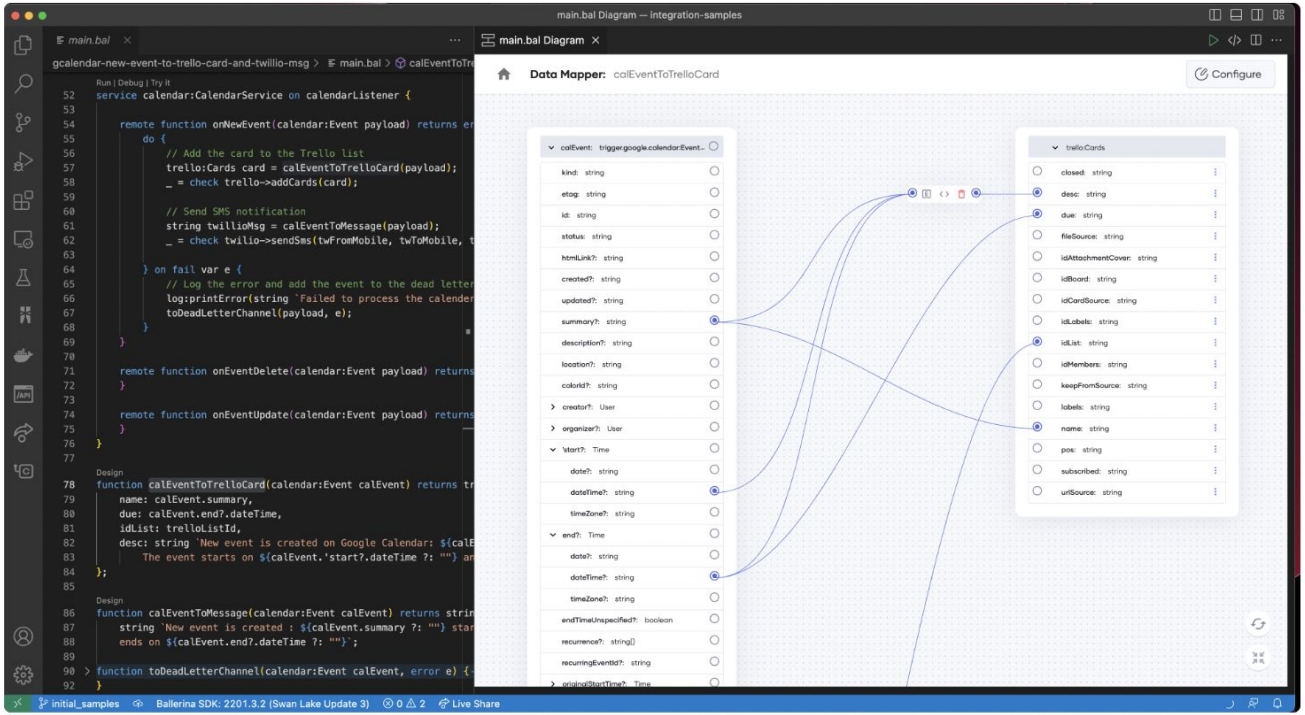
## Service designing




Ballerina

# Use Case 5: Visual Features

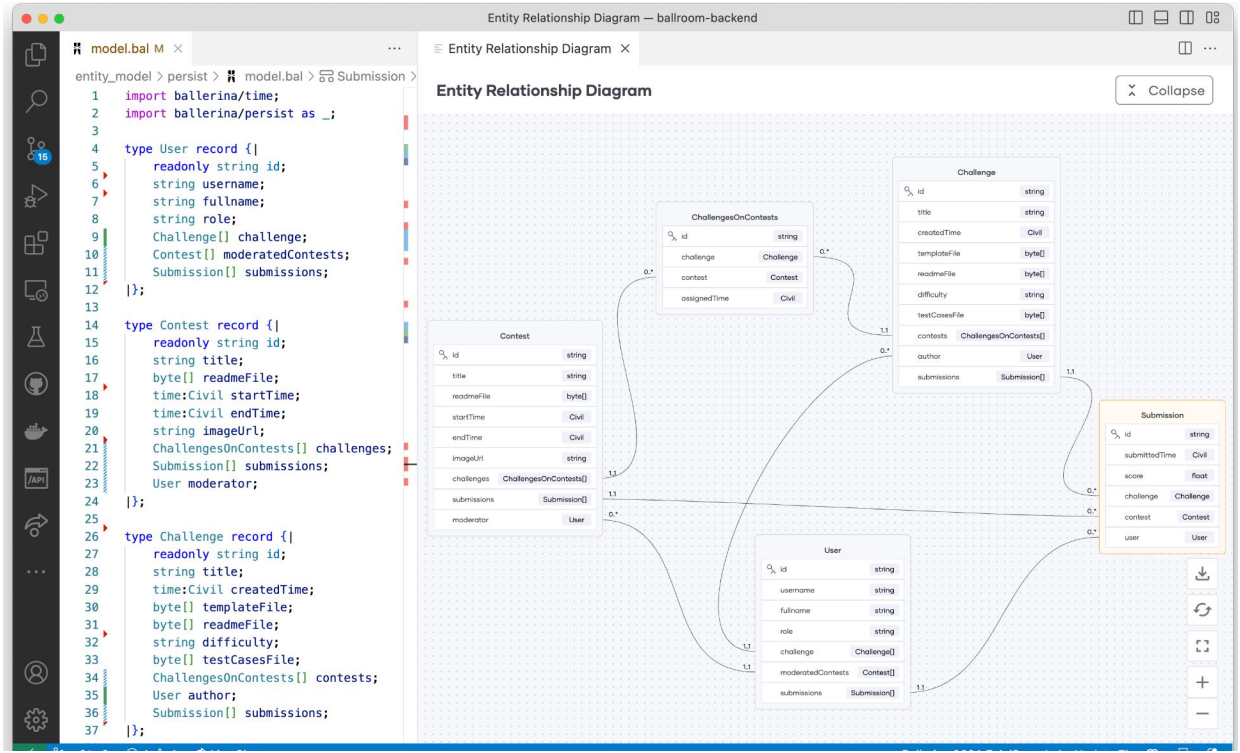
## Data mapping




Ballerina

# Use Case 5: Visual Features

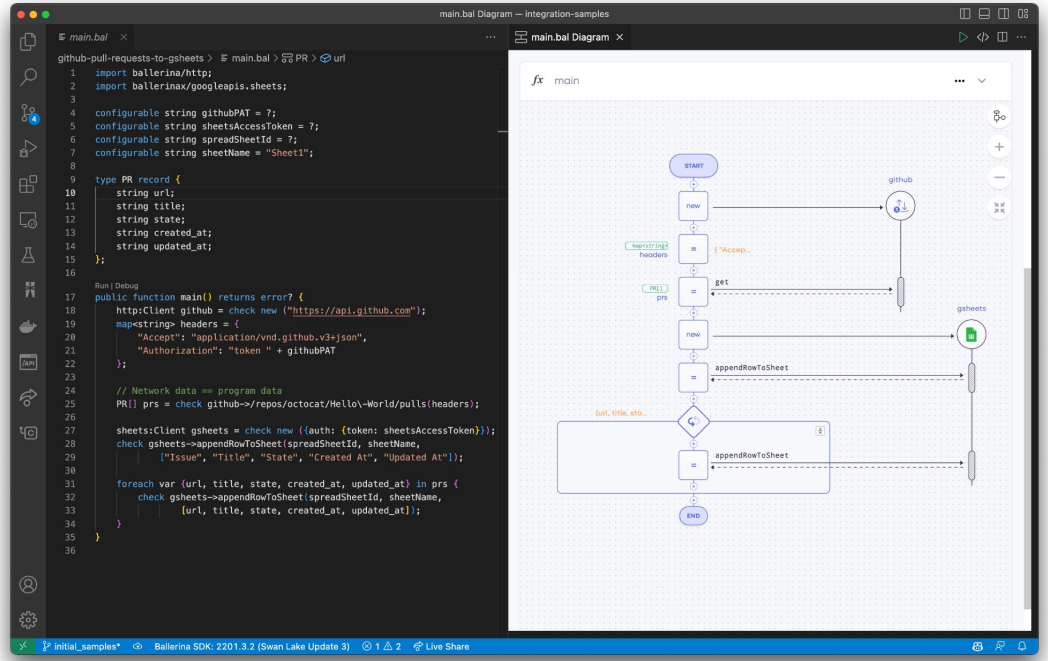
## Data persistence




Ballerina

# Use Case 5: Visual Features

## Text and graphical syntax parity




Ballerina

# Key Lessons from Developing Ballerina Lang

- It is a Platform
- Continuous Adaptation
- Incremental vs. Radical Change
- User-Centric Development

# Find out more…

- Learn Ballerina
  - Learn pages
  - Ballerina by example
  - Ballerina VS Code extension
  - Ballerina training video series
  - Ballerina certification

- Join the Ballerina community

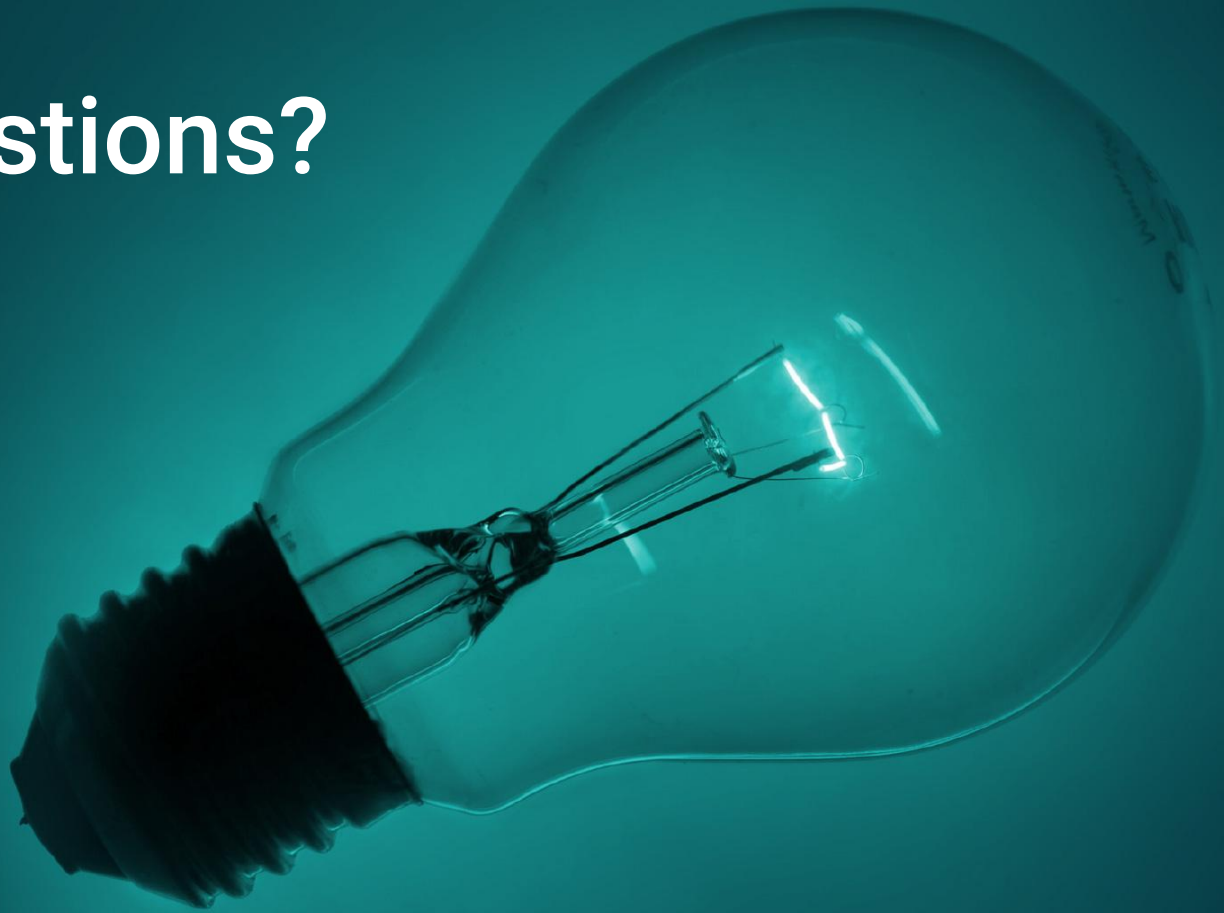

ballerinalang


Tag : ballerina

@ballerinalang


ballerina-lang

# Questions?



Ballerina
Swan Lake