Ballerina for integration

Heshan Padmasiri

May 2024

In this presentation:

- Anatomy of modern web application
- Data oriented programming
- Network communication
- How to get started

Anatomy of modern web application

Everything needs a backend service

- Almost all modern applications needs some sort of network backend
- This even includes applications that you mostly use offline
  - Text editor : setting synchronization and plugin management
  - Single player games : DRM, save file synchronization, OTA updates
- Most modern applications has multiple user facing “clients”
  - Mobile client
  - Web app
  - There may be additional integrations with other “smart gadgets” like smart watches, smart speakers, etc.
- Users expect a unified “state” across all clients

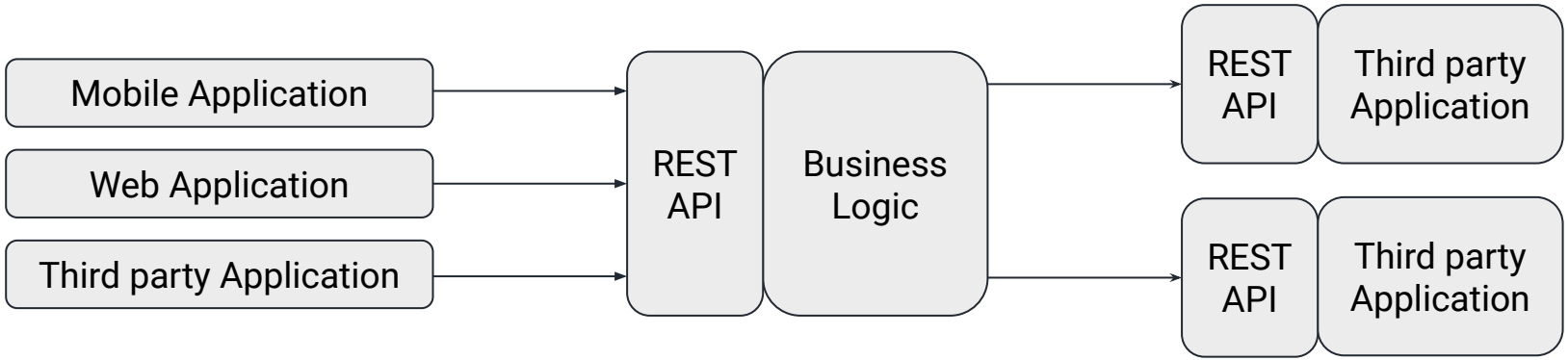
HTTP

- Hypertext Transfer Protocol (HTTP) underpins most of our network interactions
- It's a stateless protocol following the client server architecture
- Message consists of a header and body
- Body is usually,
  - HTML (web sites)
  - JSON
  - XML

REST service

- Extend the HTTP on top of Representational State Transfer (REST) architecture
  - Uniform interface
  - Client server decoupling
  - Stateless
  - Cacheability
  - Layered system architecture.
  - Code on demand (optional)
- Use the HTTP request methods like GET, POST

REST service



Other API architectures

- While REST is the most commonly used architecture is not the best option all the time
- Depending on use case
  - WebSocket : two way interactive communication
  - GraphQL: give client control over what data it receives
  - gRPC : remote procedure calls

What makes Ballerina the best options for writing network APIs

- Data oriented programming
- Network abstractions as first class citizens
- Effortless concurrency
- First class tooling support
- Connector ecosystem

Data oriented programming

Importance of data

- All APIs do is essentially move data from place to another
  - Get requests from client
  - Request additional data from other services
  - Enrich existing data with additional data
  - Remove parts of existing data
  - Send response to client
- Therefore how we represent data makes a huge difference in language ergonomics

Rich set of built in types

- In addition to usual basic types common resource representations such as `json` and `xml` are also treated as first class types.
- `record` as a universal data type.
- Comparison operations works on all value types

```
json jsonValue = { id: 5, value: "hello"};
xml xmlValue =
xml`<root><id>5</id><value>hello</value></root>`;
```

```
type User record {
   int id;
   string name;
};
User[] users = check io:fileReadCsv("users.csv");
User[] httpUsers = check httpClient->/users();
```

```
io:println([1, 2, 3] == [1, 2, 3]);
io:println([1, 2, 3] < [1, 5, 10, 20]);
```

Rich set of built in types

- `table` type when you need lookup tables.
- SQL like query expressions for declarative sequence creation

```
type User record {|
   readonly record {|
       string firstName;
       string lastName;
   |} name;
   int age;
|};
table<User> key(name) users = getUsers();
User? user = users[{firstName: "John", lastName:
"Doe"}];
```

```
Address[] user1Addresses =
from var address in addresses
join var user in users
   on address.ownerId equals user.id
where user.name == "user1"
select address;
```

Ballerina
Swan Lake

Structural typing

- Type system based on set theory
  - Types are sets of values and subtype is just subset
- Define new types using set operations
- Type relations are inferred not explicit

```
type IntValue record {
   int value;
};

type ByteValue record {
   byte value;
};

type IntValueWithMetadata record {
   int value;
   string metadata;
};

public function main() {
   IntValueWithMetadata x = { value: 5, metadata:
"hello" };
   ByteValue y = { value: 5 };
   IntValue z = x;
   IntValue w = y;
}
```

Network abstractions

Adding network abstractions to language

- We need a way to represent network abstractions in the programming language
  - HTTP clients
  - REST service
  - Marshalling and unmarshalling
- In most languages this is done by libraries
  - Dependency management
  - Difficult to optimize
  - Weak tooling support

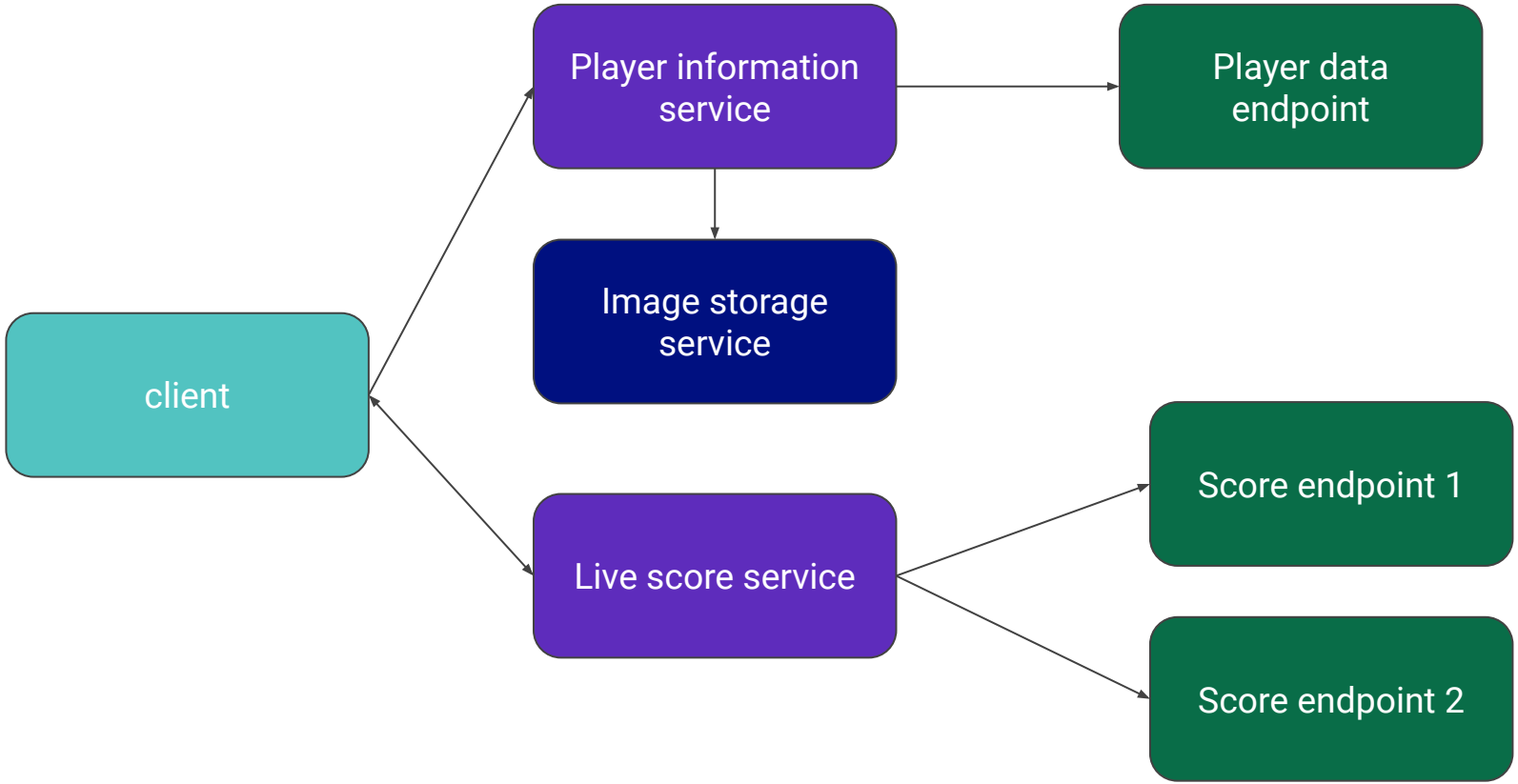
Mapping network abstractions to language abstractions

- But the biggest problem is how to map them to constructs provided by the language
  - Meta programming in languages like Rust (macros)
  - Syntactic metadata in languages like Java (annotations)
  - DSL
- This means you have to learn two “languages”. Your programming language + “language” of you library
  - Adds unwanted complexity
- You may have to learn multiple libraries
  - One for GraphQL
  - Another for REST
  - Serializing and deserializing data
  - HTTP client

Network constructs as first class citizens

- Instead of trying to retrofit network constructs to language, make them first class citizens like classes or functions
- Since directly integrated to the language
  - All the dependencies are core language libraries
  - No need for any extra tools
  - Since compiler and runtime knows what you are doing better chances to optimize
- Ballerina has first class support for many standards
  - https://ballerina.io/learn/by-example/ Network libraries section
  - Also has tools to that can generate code given spec (Ex. GraphQL, OpenAPI)
- Comes with all the bells and whistles
  - Constraint validation
  - Authentication
  - Mocking, etc.

Demo REST API + WebSocket


Player information service
Player data endpoint
Image storage service
client
Live score service
Score endpoint 1
Score endpoint 2

Source code

How to get started

How to get started

- Download ballerina at https://ballerina.io/downloads/
- For the best experience install the VSCode extension
- Learn Ballerina: https://ballerina.io/learn
- Ballerina student engagement program: https://ballerina.io/community/student-program/
- Join the Ballerina community

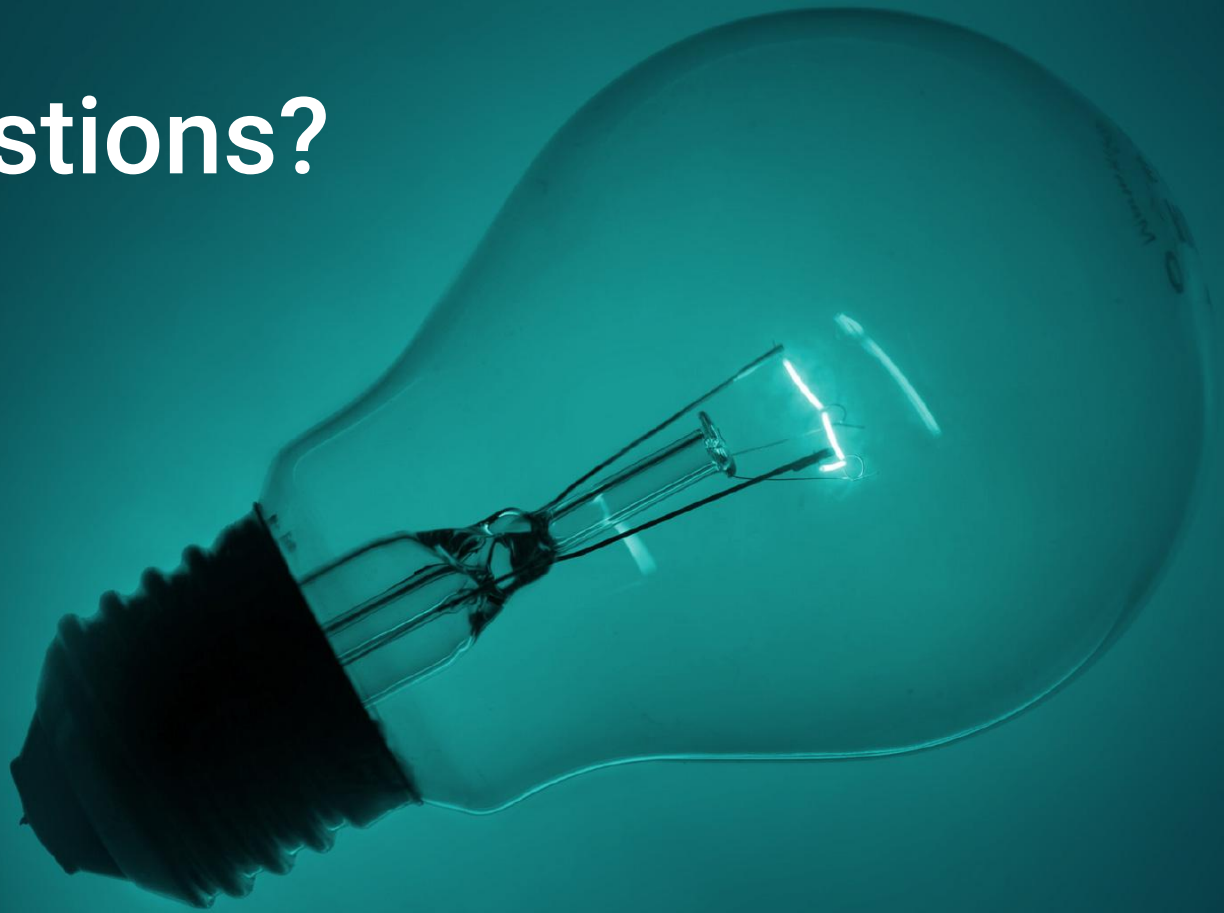
Discord
ballerinalang

COLLECTIVES™ on stackoverflow
WSO2 Collective

twitter
@ballerinalang

GitHub
ballerina-lang

Questions?



Ballerina
Swan Lake

Thank you!

If you have any further questions, please email **contact@ballerina.io** or raise them in the **Ballerina Discord server.**