Getting Started with REST development

01. Wah! What is REST?

REPRESENTATIONAL STATE TRANSFER

An Architectural Pattern to develop scalable software system with ease of maintenance and high performance.

02. But Why REST ?

Easy to Scale
High Reliability
Less Complex

And that’s not it !

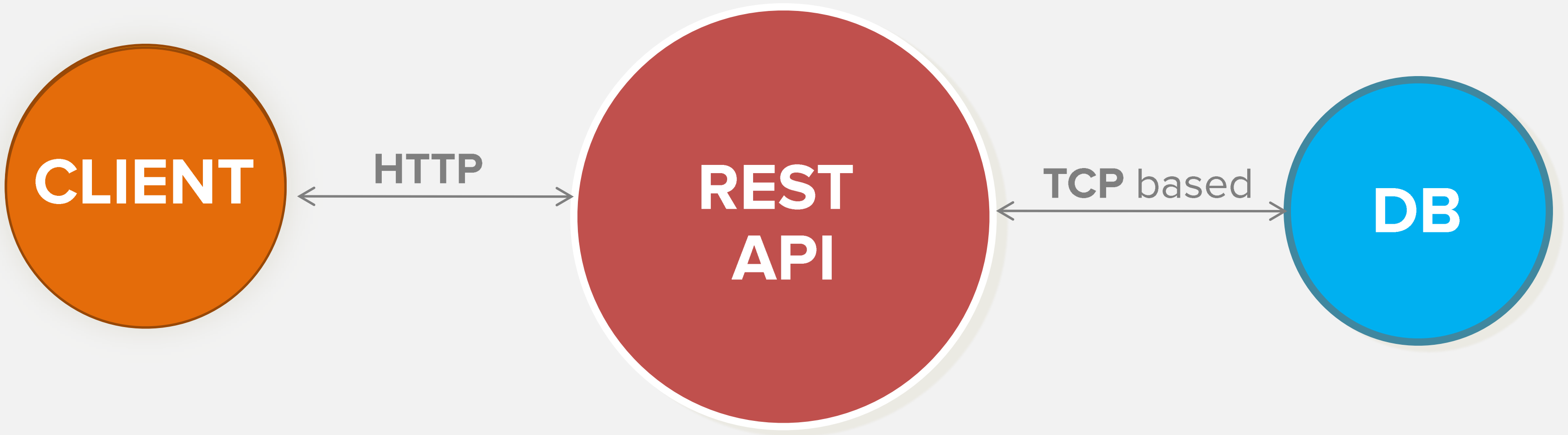
1. Its simple. Literally.

2. Scale up in seconds.

3. Uses HTTP

03. Let’s break down REST

1. Client communicates with REST server through HTTP.
2. REST server invokes necessary controller and process data.
3. REST server sends the response in form of JSON / XML



04. Properties of REST

1. CLIENT – SERVER MODEL

2. STATELESS

3. LATENCY (CACHE)

4. UNIFORM INTERFACE

5. CODE-ON-DEMAND

05. REST Best Practices

1. Use OAuth or JWT for HTTP Authorization
2. Filter all requests before invoking any controllers
3. Never use passwords to maintain state
4. Use proper status codes in response
5. Use External Key Storage like Redis
6. Use versioning for API

06. REST Behavior

1. **GET –** Fetch a resource
2. **POST –** Add a resource (not idempotent)
3. **PUT –** Add / Update a resource (idempotent)
4. **DELETE –** Delete a resource
5. **HEAD –** Get the header w/o body

Enough Speech !

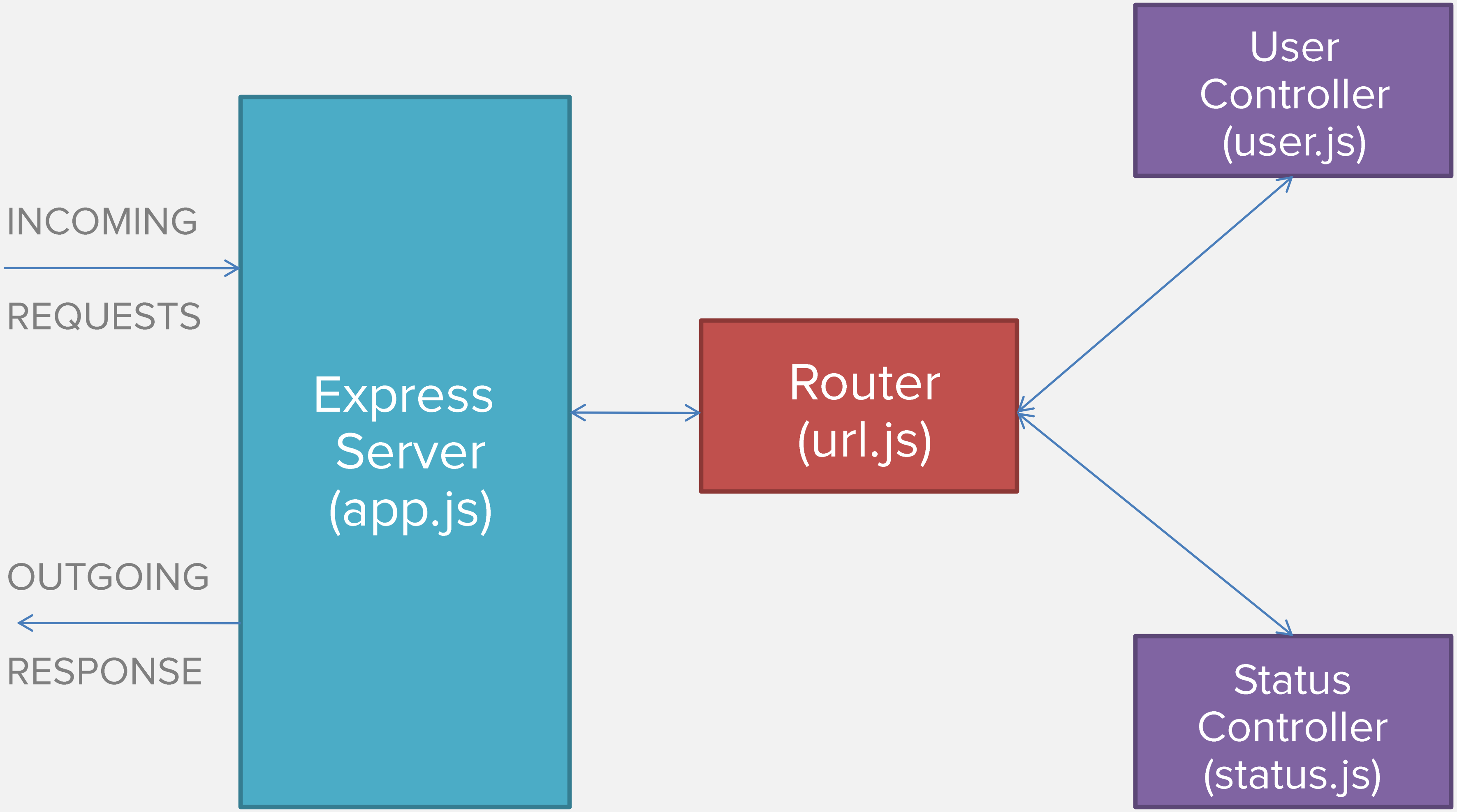
Let’s Code.

We’re going to build a Social Network.

08. Components to build

1. Create User Account **(POST)**
2. Search Users **(GET)**
3. Delete an User **(DELETE)**
4. Share a Status **(POST)**
5. Retrieve all statuses **(GET)**

09. Architecture



User Controller

- Users -> [ ] Stores all user information

- Each user has following structure:

```
{
    username  : String,
    fullName   : String,
    age             : Integer
}
```

User Controller: Filter Use - Cases

- **CREATE ACCOUNT** – Check for empty content

  Already registered account

- **SEARCH ACCOUNT –** Check for username availability

  Check for all username

- **DELETE ACCOUNT –** Check for username availability

  Check for null value

Status Controller

- Statuses -> [ ] Stores all status information
- Each status has following structure,

```
{
          id          :  String (uuid),
          username  :  String,
          content     : String,
          time          : Integer (Timestamp)
}
```

Testing out our REST Server

- Test all the API endpoints using Postman
- Test all use cases
- Use automated testing wherever possible

QUESTIONS ?

I am anticipating for one at least

THANK YOU