NETWORKING WITH URLSESSION

PART 4: URLSESSION COOKBOOK 1

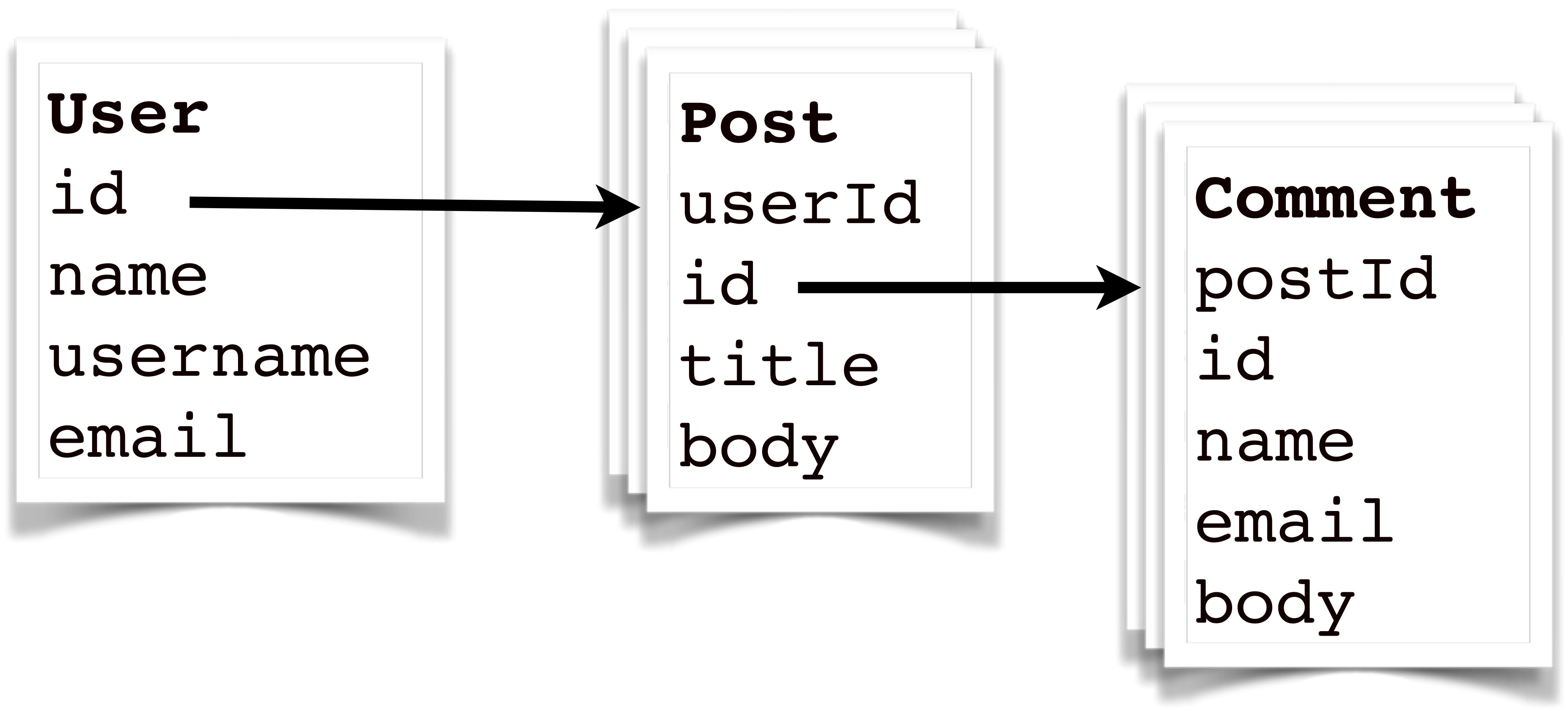
REST API

REST API

REpresentational State Transfer

REST API

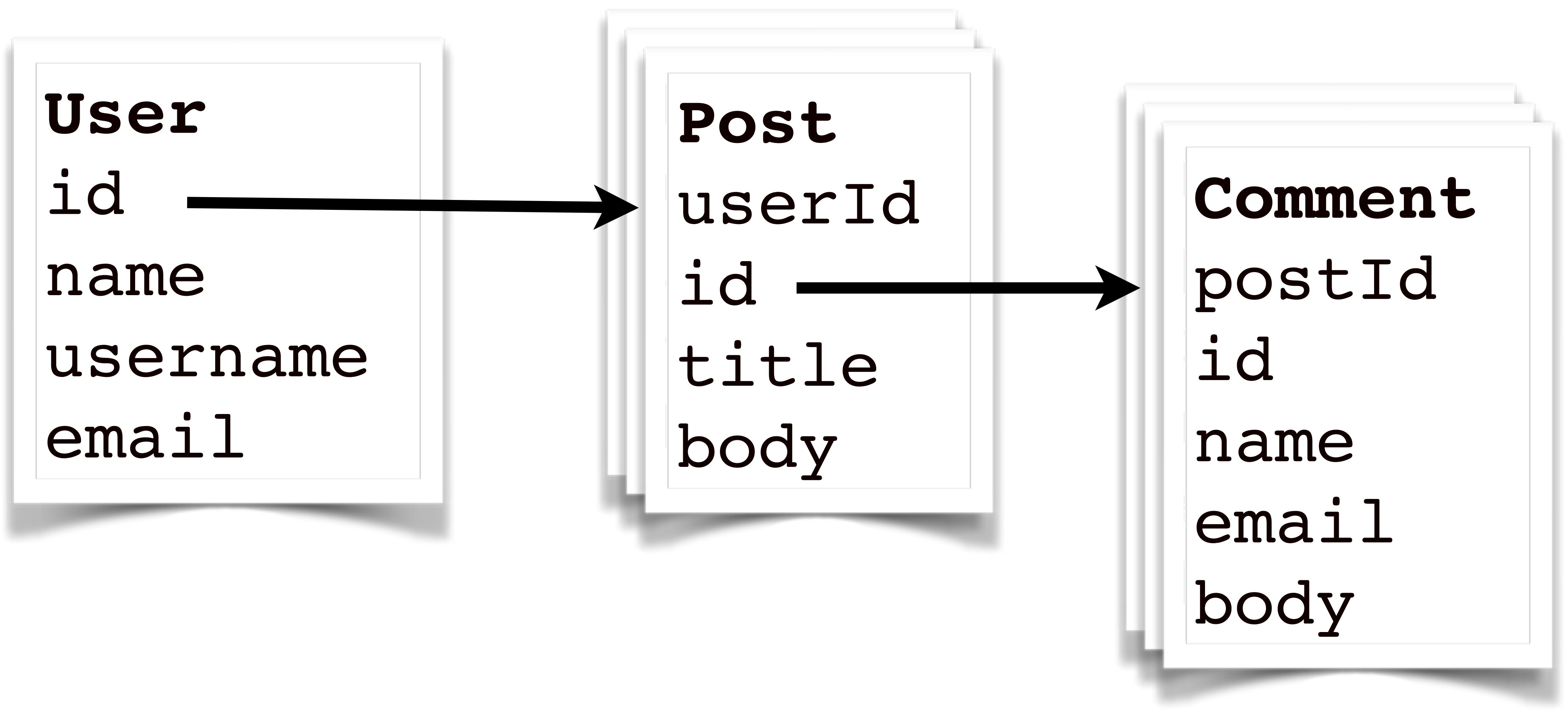
REpresentational State Transfer



REST API

`GET /posts`

REpresentational State Transfer

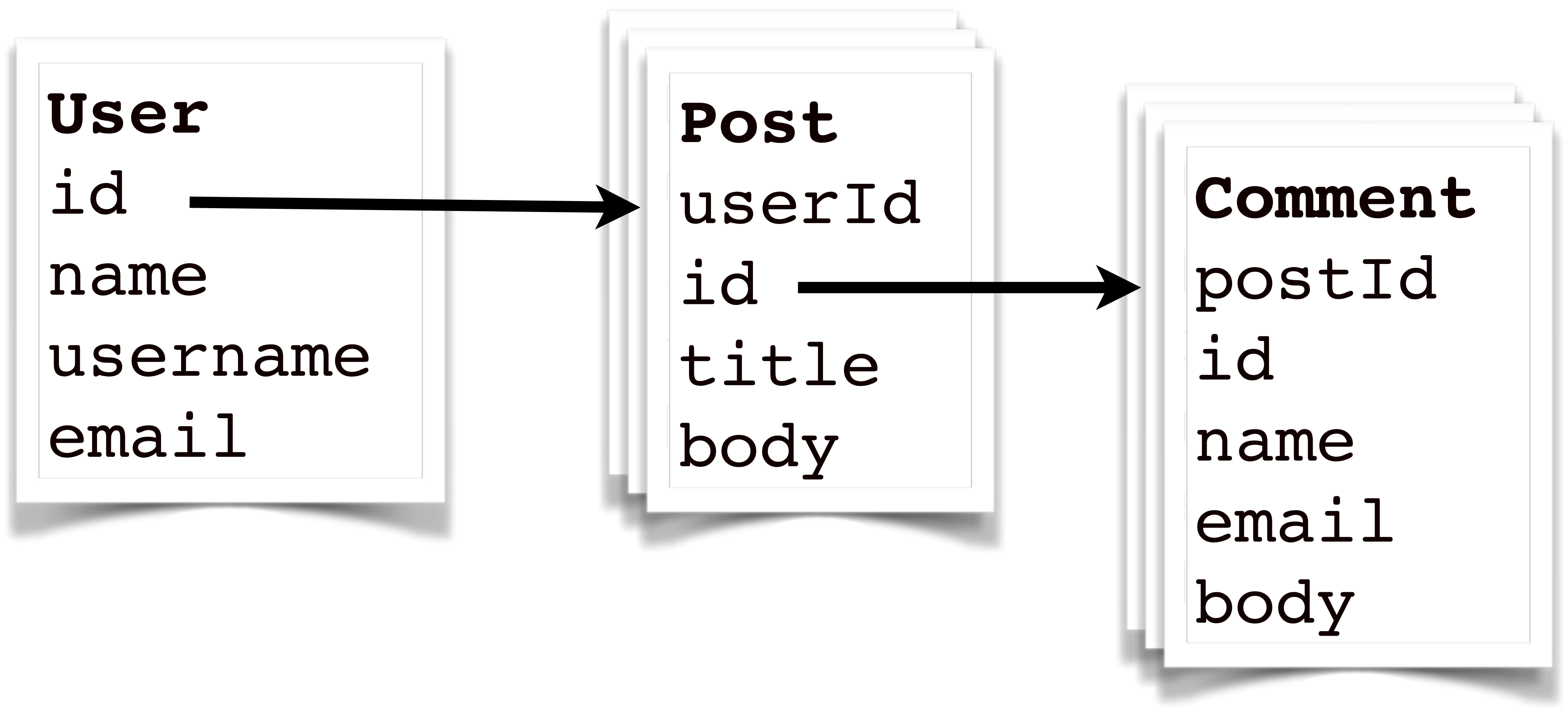


REST API

GET /posts

GET /posts/1

REpresentational State Transfer



REST API

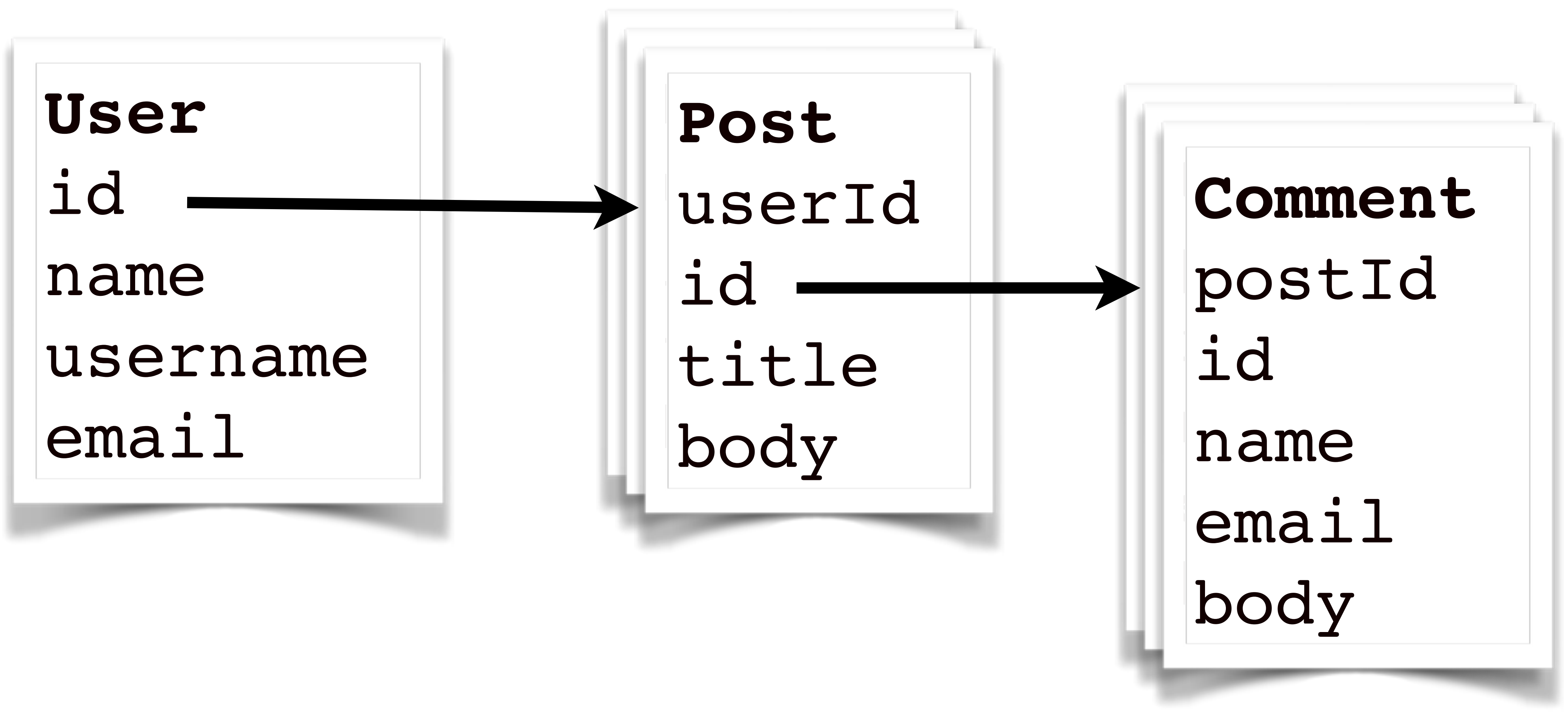
```
GET /posts
```

```
GET /posts/1
```

```
GET /posts/1/comments
```

```
GET /comments?postId=1
```

REpresentational State Transfer



REST API

```
GET /posts
```

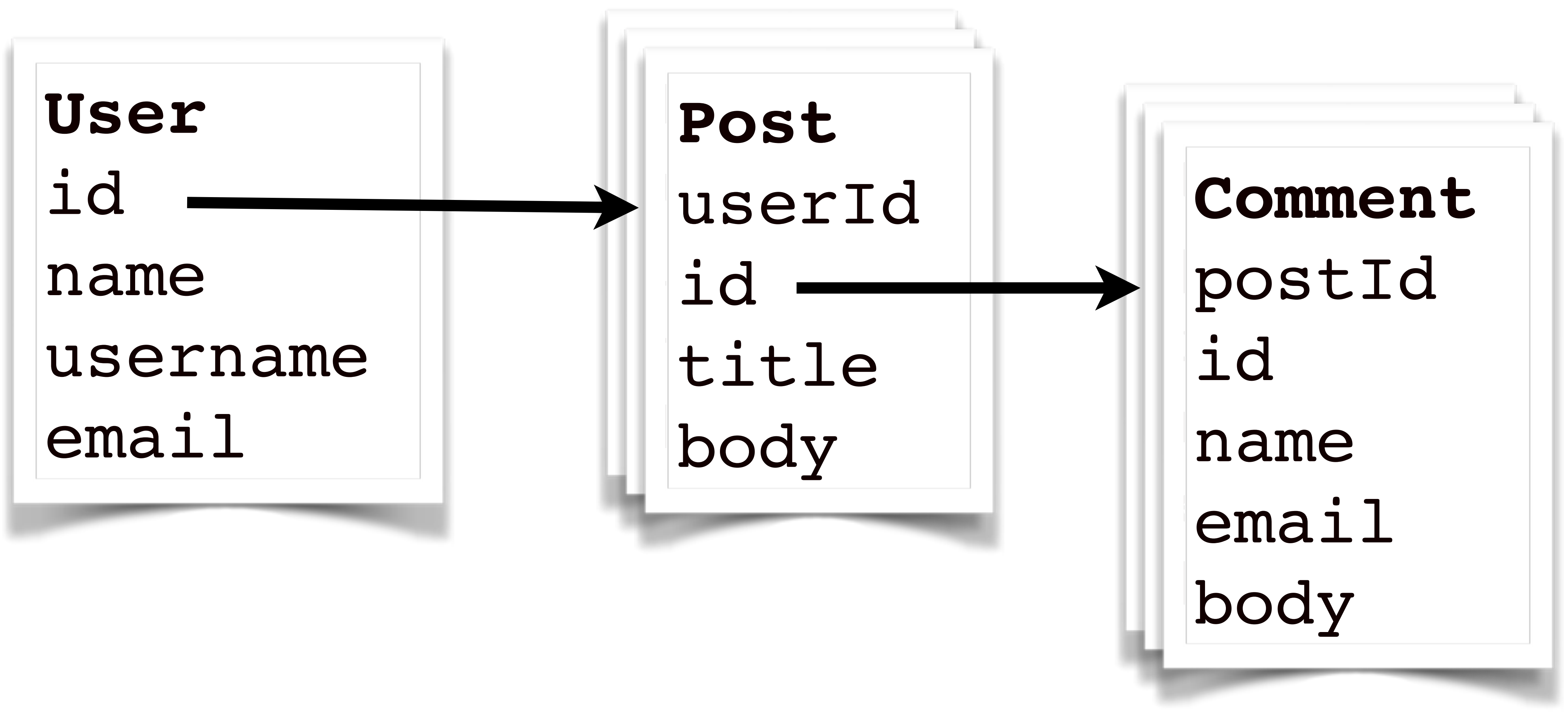
```
GET /posts/1
```

```
GET /posts/1/comments
```

```
GET /comments?postId=1
```

```
GET /posts?userId=1
```

REpresentational State Transfer



REST API

```
GET  /posts
```

```
GET  /posts/1
```

```
GET  /posts/1/comments
```

```
GET  /comments?postId=1
```

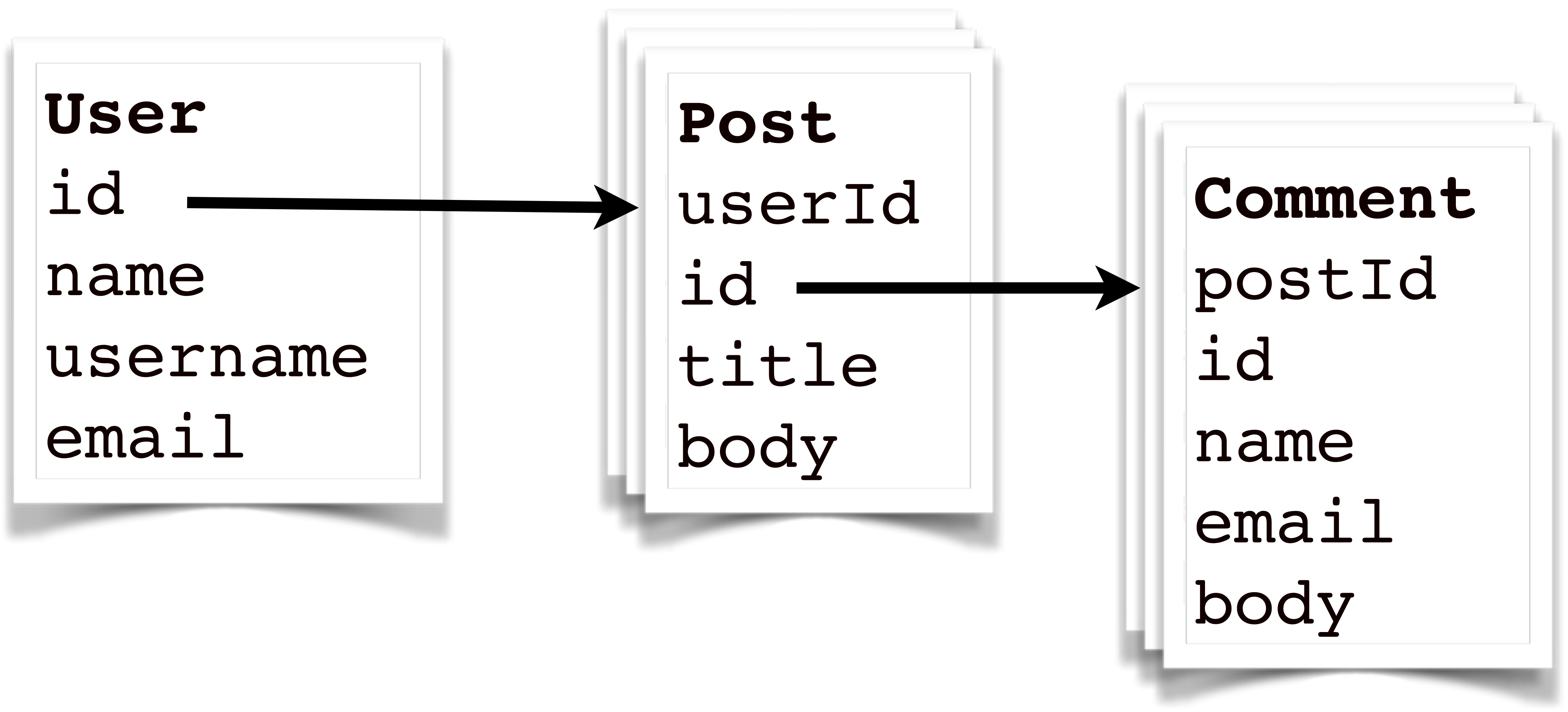
```
GET  /posts?userId=1
```

```
POST  /posts
```

```
PUT  /posts/1
```

```
DELETE  /posts/1
```

REpresentational State Transfer



JSONSerialization

JavaScript Object Notation

JSONSerialization

JavaScript Object Notation

```
string, boolean, array, object/dictionary, null and number
```

JSONSerialization

JavaScript Object Notation

```
string, boolean, array, object/dictionary, null and number
```

```
class func jsonObject(with: Data,
  options: JSONSerialization.ReadingOptions = [])

class func data(withJSONObject: Any,
  options: JSONSerialization.WritingOptions = [])

class func isValidJSONObject(Any)
```

Which Error is That?

```
let task = session.dataTask(with: url!) { data, response, error in
  if error != nil {
    // Used for client-side errors
  }
  // Used for server-side errors
  (response as? NSHTTPURLResponse)?.statusCode
}
```

Which Error is That?

```
let task = session.dataTask(with: url!) { data, response, error in
  if error != nil {
    // Used for client-side errors
  }
  // Used for server-side errors
  (response as? NSHTTPURLResponse)?.statusCode
}
```

Which Error is That?

```
let task = session.dataTask(with: url!) { data, response, error in
  if error != nil {
    // Used for client-side errors
  }
  // Used for server-side errors
  (response as? HTTPURLResponse)?.statusCode
}
```

URLSessionDataTask

URLSessionDataTask

```swift
let session = URLSession.shared
```

URLSessionDataTask

```swift
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
```

URLSessionDataTask

```
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
let dataTask = session.dataTask(with: url) { data, response, error in
```

URLSessionDataTask

```
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
let dataTask = session.dataTask(with: url) { data, response, error in
  // Check client-side error and response from server
```

URLSessionDataTask

```
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
let dataTask = session.dataTask(with: url) { data, response, error in
  // Check client-side error and response from server
  // Process data
```

URLSessionDataTask

```swift
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
let dataTask = session.dataTask(with: url) { data, response, error in
  // Check client-side error and response from server
  // Process data
  DispatchQueue.main.async {
    // Update UI
  }
}
```

URLSessionDataTask

```swift
let session = URLSession.shared
let url = URL(string: “https://jsonplaceholder.typicode.com/posts/1")
let dataTask = session.dataTask(with: url) { data, response, error in
  // Check client-side error and response from server
  // Process data
  DispatchQueue.main.async {
    // Update UI
  }
}
dataTask?.resume()
```

Demo

Challenge Time!

```
let urlString = "http://localhost:3000/posts/"
```

```
let array = try JSONSerialization.jsonObject(with: data, options: [])
    as? [JSONDictionary]
```

```
posts.append(Post(id: id, author: author, title: title))
```

```
struct Post {
  let id: Int
  let author: String
  let title: String
}
```