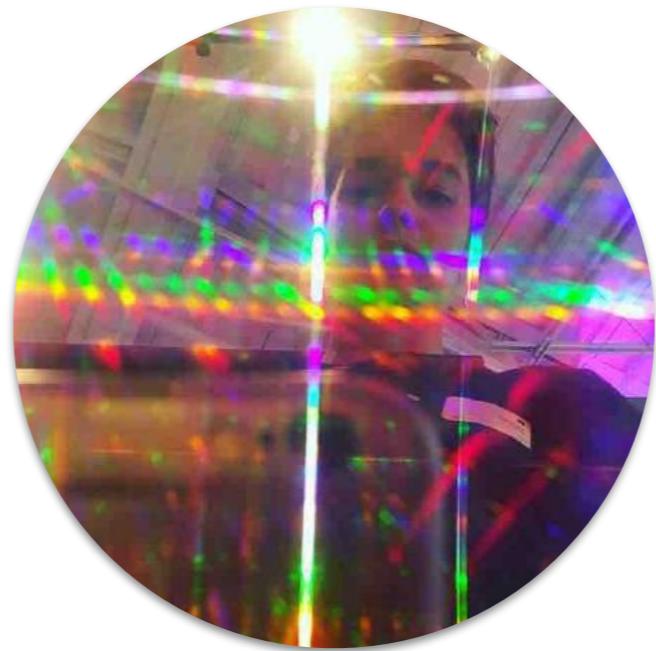


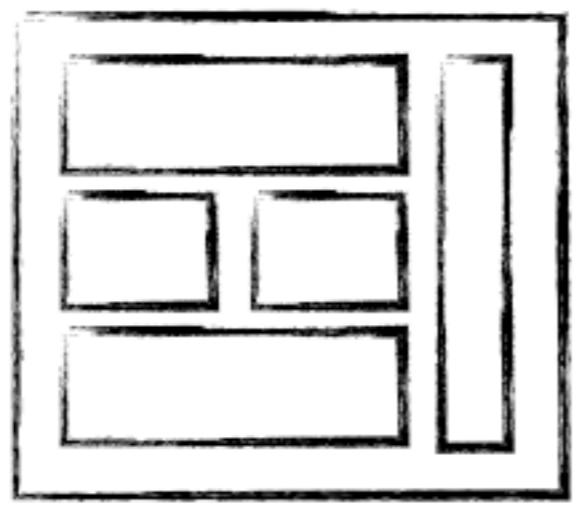
Microservices in Vapor

Building Big Projects



@tanner0101

<http://tanner.xyz>



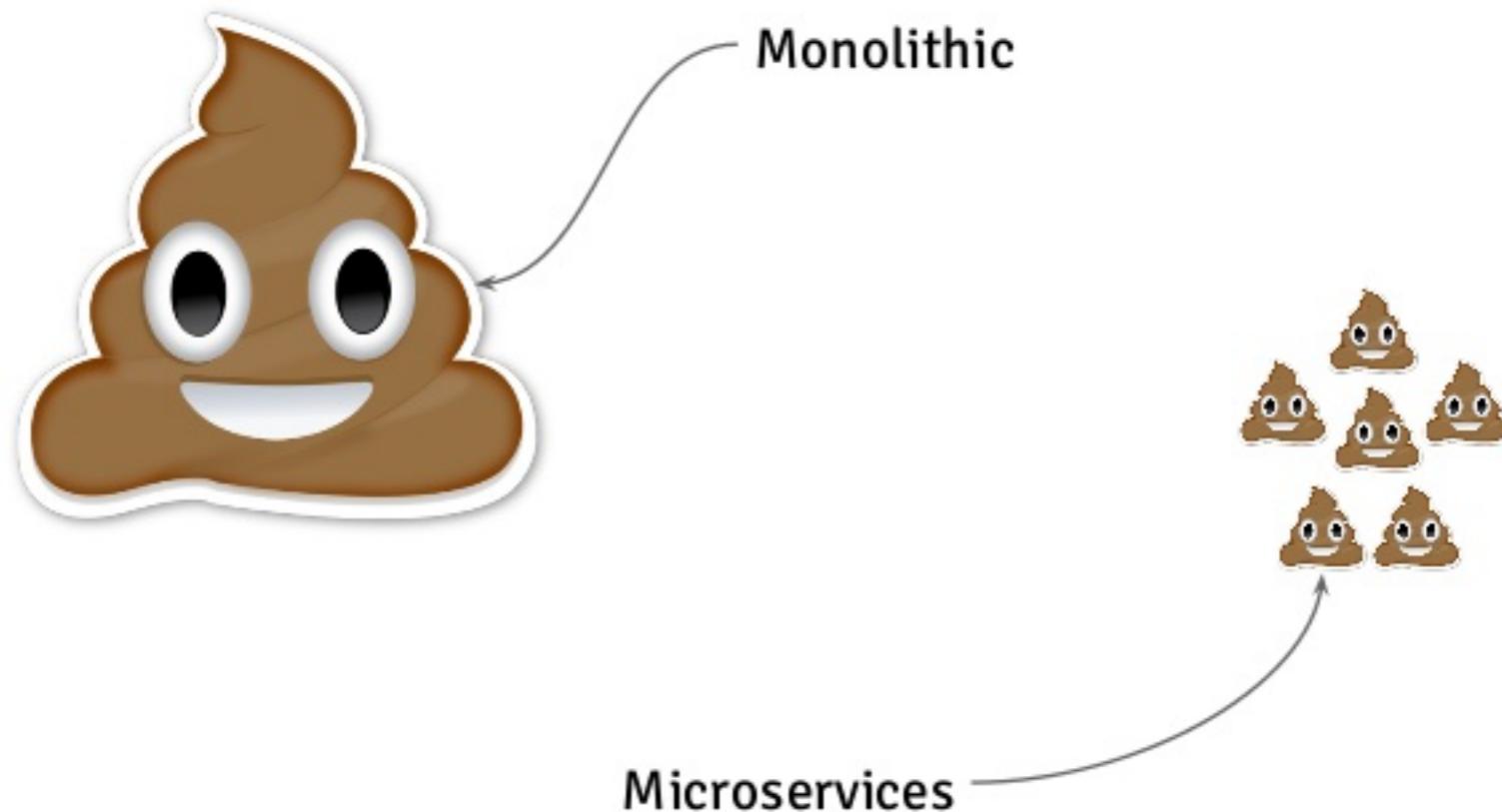
MONOLITHIC/LAYERED



MICRO SERVICES

*<http://eugenedvorkin.com/seven-micro-services-architecture-advantages/>

Monolithic vs Microservices

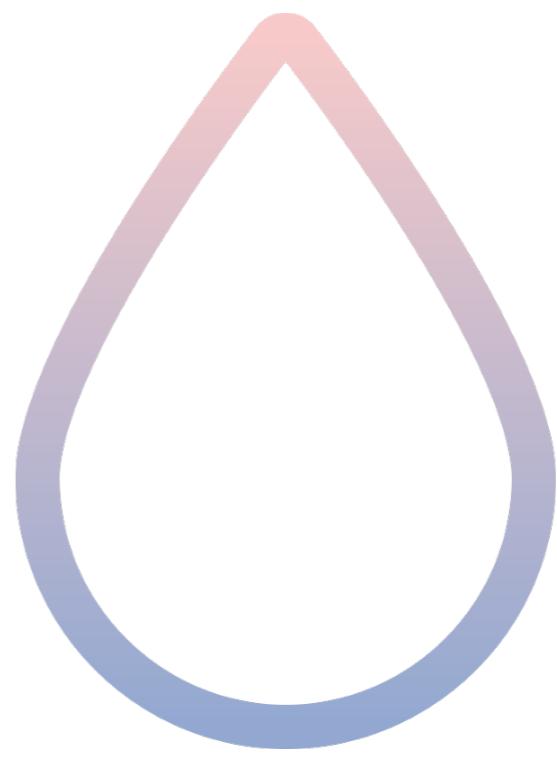
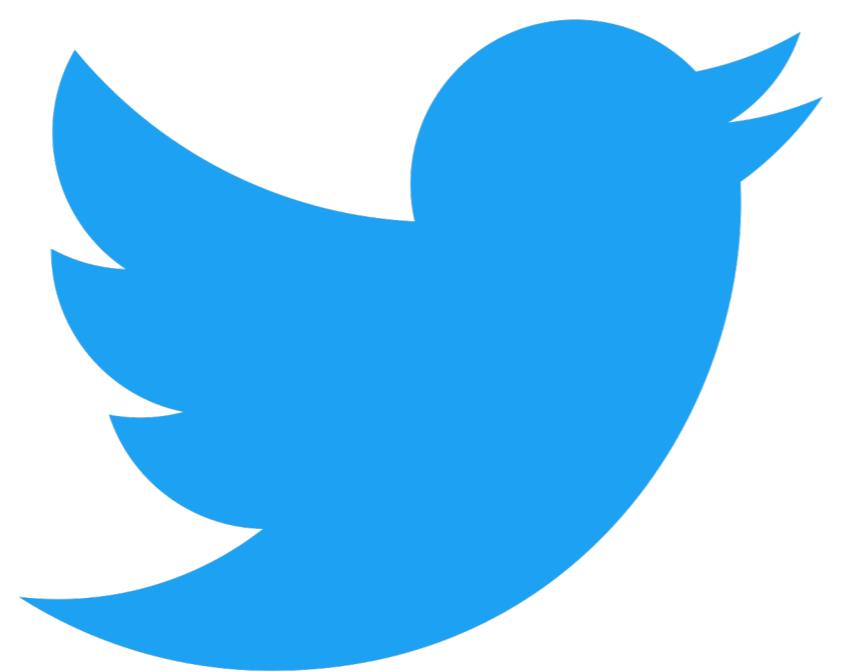


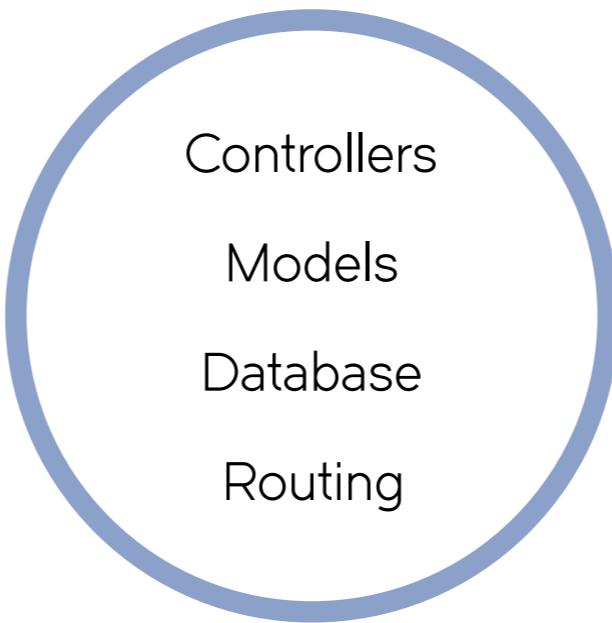
 @alvaro_sanchez



```
1 import Vapor
2
3 public class User: Model {
4     ...
5 }
```

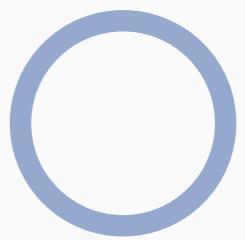
```
1 import Models  
2  
3 extension Models.User {  
4     func sayHello() {  
5         print("Hi, my name is \(self.name)")  
6     }  
7 }
```





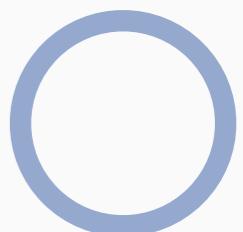
Tweets API

```
1 import Vapor  
2  
3 class Tweet: Model {  
4     var userId: String  
5     var message: String  
6     ...  
7 }
```



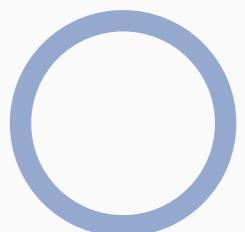
Tweets API

```
1 import Vapor
2
3 class TweetController {
4     func store(_ req: Request) throws -> ResponseRepresentable {
5         guard
6             let message = req.data["message"]?.string,
7             let userId = req.data["userId"]?.string
8         else {
9             throw Abort(.badRequest)
10        }
11
12        let tweet = Tweet(userId: userId, message: message)
13        try tweet.save()
14        return tweet
15    }
16 }
```



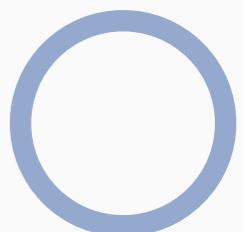
Tweets API

```
1 import Vapor
2
3 let drop = try Droplet()
4
5 let tweetController = TweetController()
6 drop.post("tweets", handler: tweetController.store)
7
8 try drop.run()
```



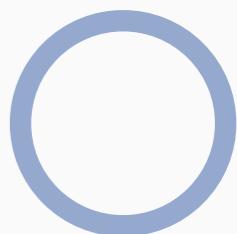
Tweets API

```
1 import Vapor
2
3 class TweetController {
4     func store(_ req: Request) throws -> ResponseRepresentable {
5         guard
6             let message = req.data["message"]?.string,
7             let userId = req.data["userId"]?.string
8         else {
9             throw Abort(.badRequest)
10        }
11
12        let tweet = Tweet(userId: userId, message: message)
13        try tweet.save()
14        return tweet
15    }
16 }
```

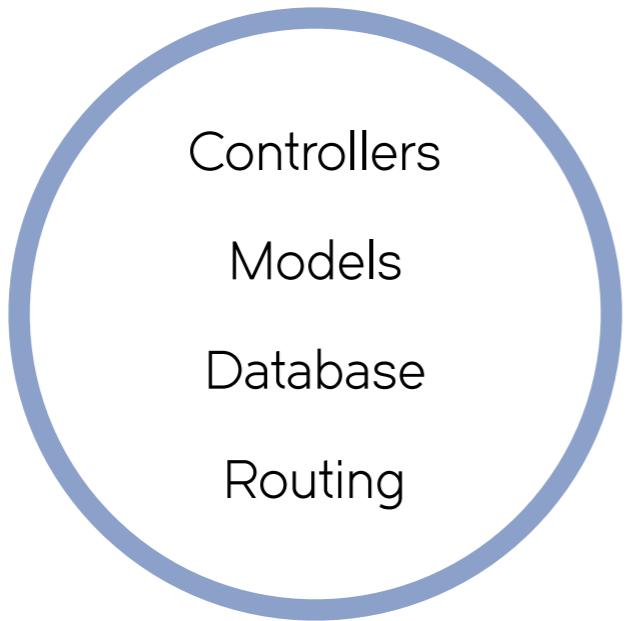


Tweets API

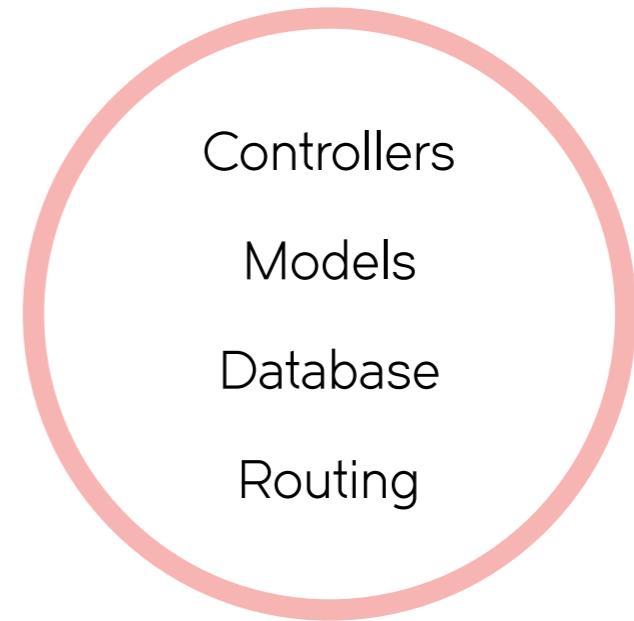
```
6     let message = req.data["message"]?.string,  
7     let userId = req.data["userId"]?.string  
8 else {  
9     throw Abort(.badRequest)  
10 }  
11  
12 let res = try drop.client.get("http://api.twitter.com/users/\\(userId)")  
13 guard  
14     let isBanned = res.data["is_banned"]?.bool,  
15     res.status == .ok  
16 else {  
17     throw Abort(.internalServerError)  
18 }  
19  
20 guard !isBanned else {  
21     throw Abort(.unauthorized)  
22 }  
23  
24 let tweet = Tweet(userId: userId, message: message)  
25 try tweet.save()  
26 return tweet  
27 }  
28 }
```



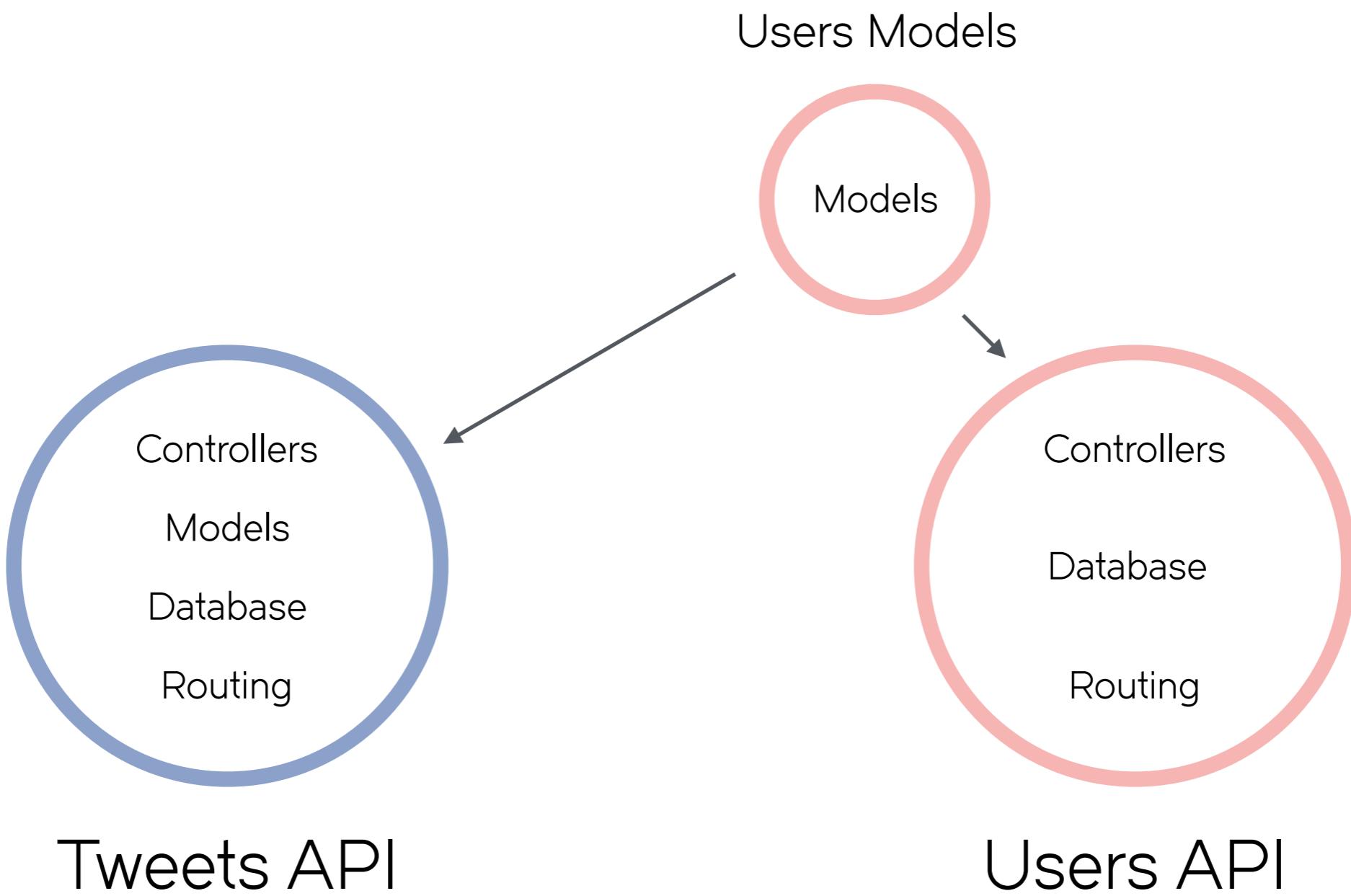
Tweets API



Tweets API



Users API

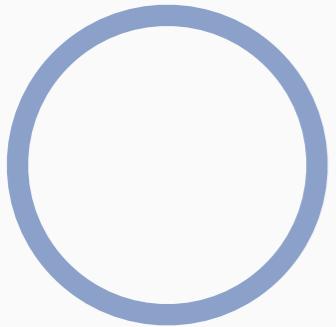


Users Models



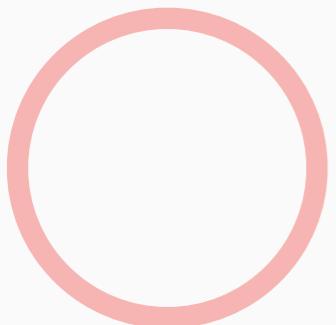
```
1 import PackageDescription  
2  
3 let package = Package(  
4     name: "UsersModels",  
5     dependencies: [  
6         .Package(url: "https://github.com/vapor/json.git", majorVersion: 2)  
7     ]  
8 )
```

Tweets API



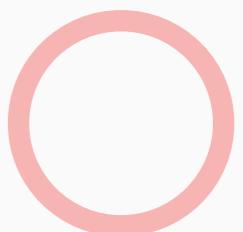
```
1 import PackageDescription  
2  
3 let package = Package(  
4     name: "TweetsAPI",  
5     dependencies: [  
6         .Package(url: "https://github.com/vapor/vapor.git", majorVersion: 2),  
7         .Package(url: "https://github.com/twitter/users-models.git", majorVersion: 1)  
8     ]  
9 )
```

Users API



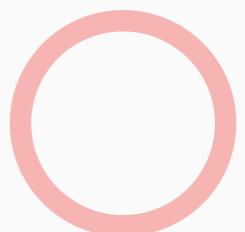
```
1 import PackageDescription  
2  
3 let package = Package(  
4     name: "UsersAPI",  
5     dependencies: [  
6         .Package(url: "https://github.com/vapor/vapor.git", majorVersion: 2),  
7         .Package(url: "https://github.com/twitter/users-models.git", majorVersion: 1)  
8     ]  
9 )
```

```
1 public final class User {  
2     public var id: String  
3     public var name: String  
4     public var isBanned: Bool  
5 }
```



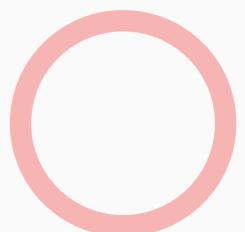
Users Models

```
1 import JSON
2
3 extension User: JSONConvertible {
4     public init(json: JSON) throws {
5         id = try json.get("id")
6         name = try json.get("name")
7         isBanned = try json.get("isBanned")
8     }
9
10    public makeJSON() throws -> JSON {
11        var json = JSON()
12        try json.set("id", id)
13        try json.set("name", name)
14        try json.set("isBanned", isBanned)
15        return json
16    }
17 }
```



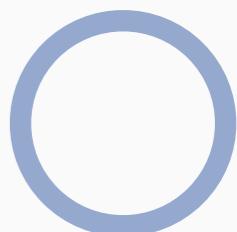
Users Models

```
1  {
2    "id": "a8ccc26e-2c9d-4105-b69b-686f9b0b09f3",
3    "name": "Bob",
4    "isBanned": false
5 }
```



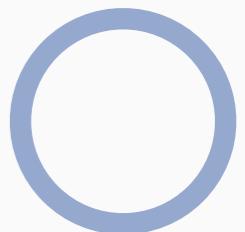
Users Models

```
6     let message = req.data["message"]?.string,  
7     let userId = req.data["userId"]?.string  
8 else {  
9     throw Abort(.badRequest)  
10 }  
11  
12 let res = try drop.client.get("http://api.twitter.com/users/\\(userId)")  
13 guard  
14     let isBanned = res.data["is_banned"]?.bool,  
15     res.status == .ok  
16 else {  
17     throw Abort(.internalServerError)  
18 }  
19  
20 guard !isBanned else {  
21     throw Abort(.unauthorized)  
22 }  
23  
24 let tweet = Tweet(userId: userId, message: message)  
25 try tweet.save()  
26 return tweet  
27 }  
28 }
```

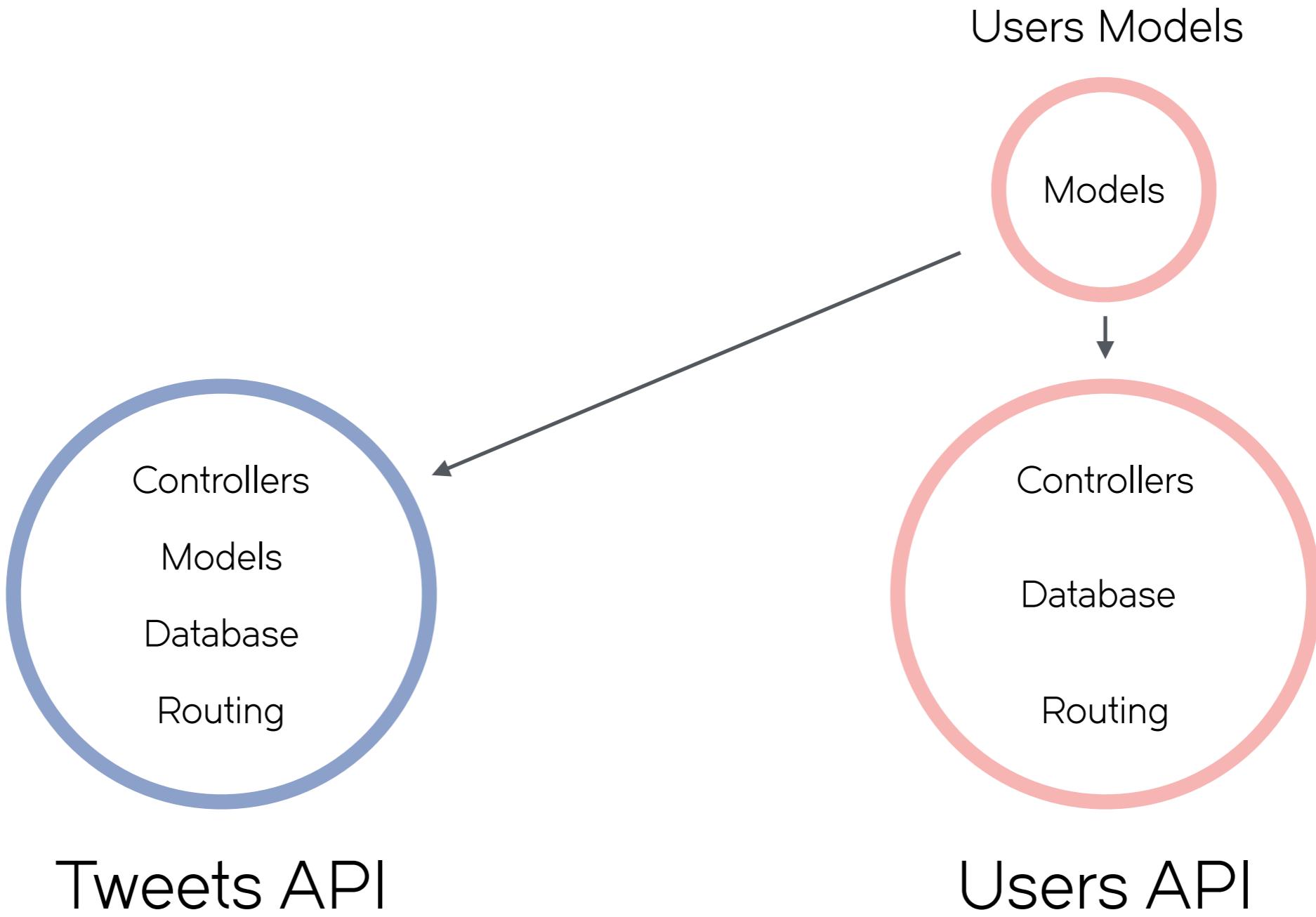


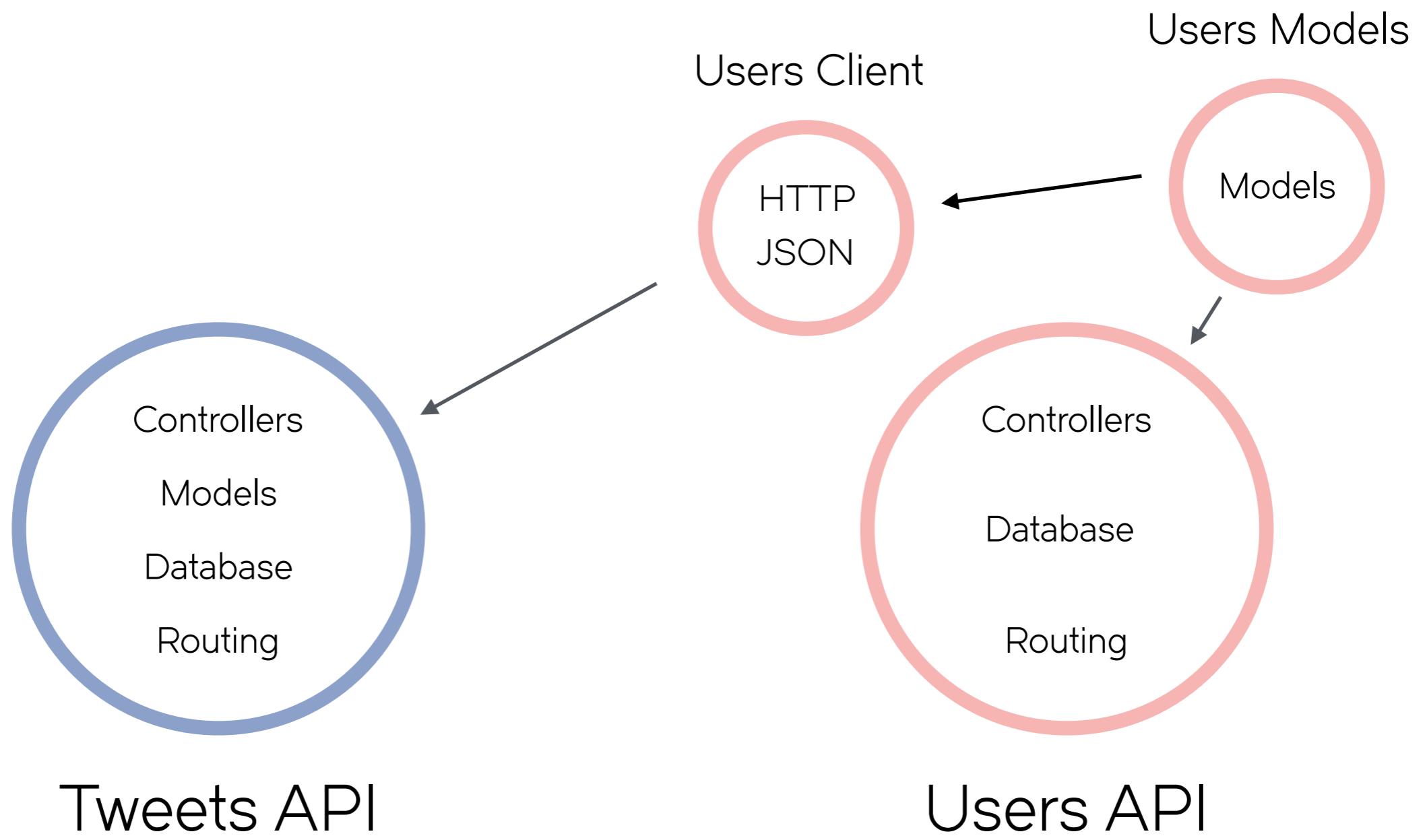
Tweets API

```
4 func store(_ req: Request) throws -> ResponseRepresentable {  
5     guard  
6         let message = req.data["message"]?.string,  
7         let userId = req.data["userId"]?.string  
8     else {  
9         throw Abort(.badRequest)  
10    }  
11  
12    let res = try drop.client.get("http://api.twitter.com/users/\(userId)")  
13    let user = try User(json: res.json)  
14  
15    guard !user.isBanned else {  
16        throw Abort(.unauthorized)  
17    }  
18  
19    let tweet = Tweet(userId: userId, message: message)  
20    try tweet.save()  
21    return tweet  
22}  
23}
```

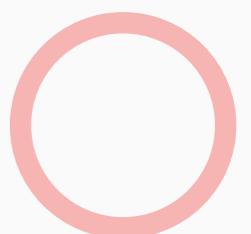


Tweets API



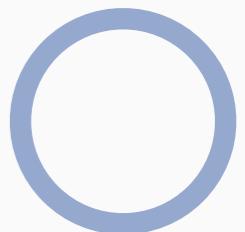


```
1 public final class UserClient {  
2     ...  
3  
4     public static func getUser(withId id: String) throws -> User {  
5         let res = client.get("/users/\\(id)")  
6         return try User(json: res.json)  
7     }  
8     ...  
9  
10 }
```



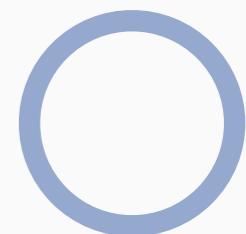
Users Client

```
4 func store(_ req: Request) throws -> ResponseRepresentable {  
5     guard  
6         let message = req.data["message"]?.string,  
7         let userId = req.data["userId"]?.string  
8     else {  
9         throw Abort(.badRequest)  
10    }  
11  
12    let res = try drop.client.get("http://api.twitter.com/users/\(userId)")  
13    let user = try User(json: req.json)  
14  
15    guard !user.isBanned else {  
16        throw Abort(.unauthorized)  
17    }  
18  
19    let tweet = Tweet(userId: userId, message: message)  
20    try tweet.save()  
21    return tweet  
22}  
23}
```

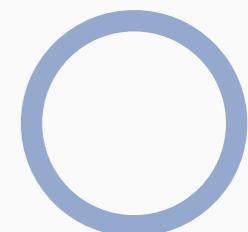


Tweets API

```
5
6     let message = req.data["message"]?.string,
7     let userId = req.data["userId"]?.string
8 else {
9     throw Abort(.badRequest)
10}
11
12 let user = try UserClient.getUser(withId: userId)
13 guard !user.isBanned else {
14     throw Abort(.unauthorized)
15}
16
17 let tweet = Tweet(userId: userId, message: message)
18 try tweet.save()
19 return tweet
20}
21}
```

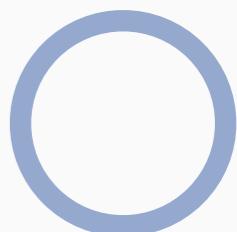


```
3 class TweetController {  
4     func store(_ req: Request) throws -> ResponseRepresentable  
5     guard  
6         let message = req.data["message"]?.string,  
7         let userId = req.data["userId"]?.string  
8     else {  
9         throw Abort(.badRequest)  
10    }  
11  
12    try UserClient.assertUserNotBanned(withId: userId)  
13  
14    let tweet = Tweet(userId: userId, message: message)  
15    try tweet.save()  
16    return tweet  
17 }  
18 }
```



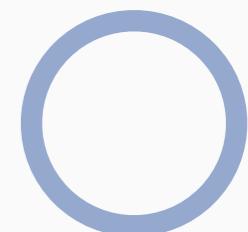
Tweets API

```
6     let message = req.data["message"]?.string,  
7     let userId = req.data["userId"]?.string  
8 else {  
9     throw Abort(.badRequest)  
10 }  
11  
12 let res = try drop.client.get("http://api.twitter.com/users/\\(userId)")  
13 guard  
14     let isBanned = res.data["is_banned"]?.bool,  
15     res.status == .ok  
16 else {  
17     throw Abort(.internalServerError)  
18 }  
19  
20 guard !isBanned else {  
21     throw Abort(.unauthorized)  
22 }  
23  
24 let tweet = Tweet(userId: userId, message: message)  
25 try tweet.save()  
26 return tweet  
27 }  
28 }
```

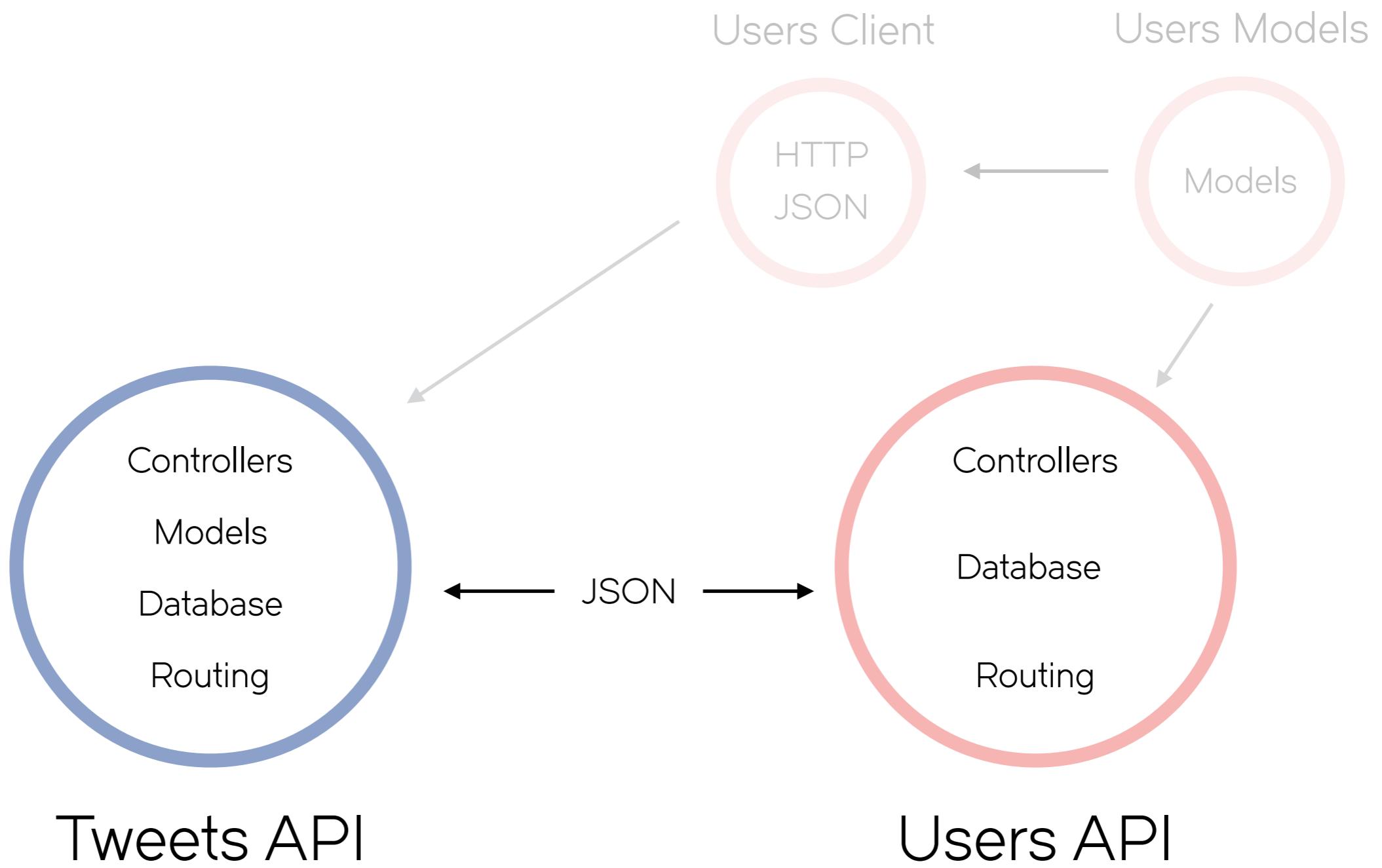


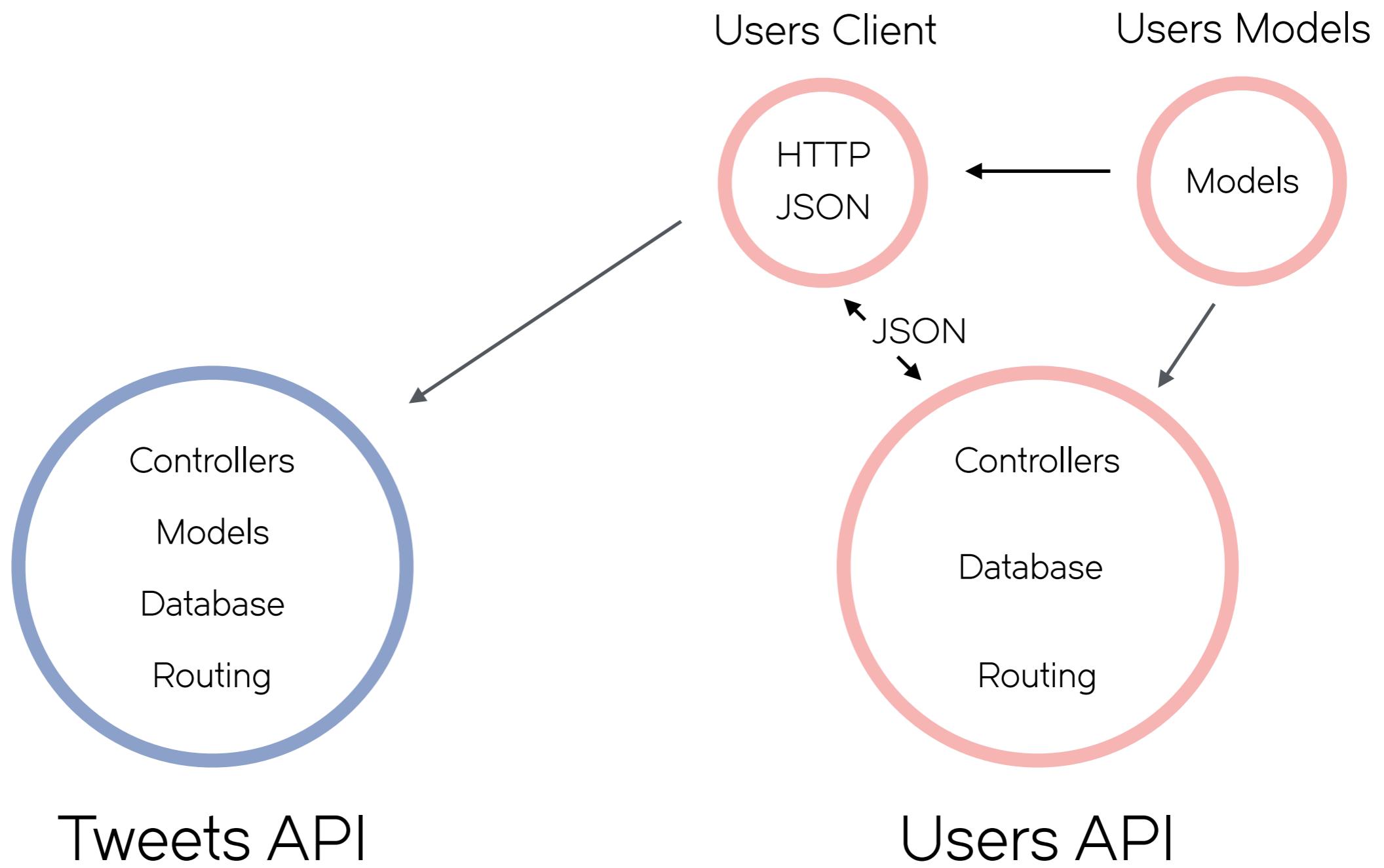
Tweets API

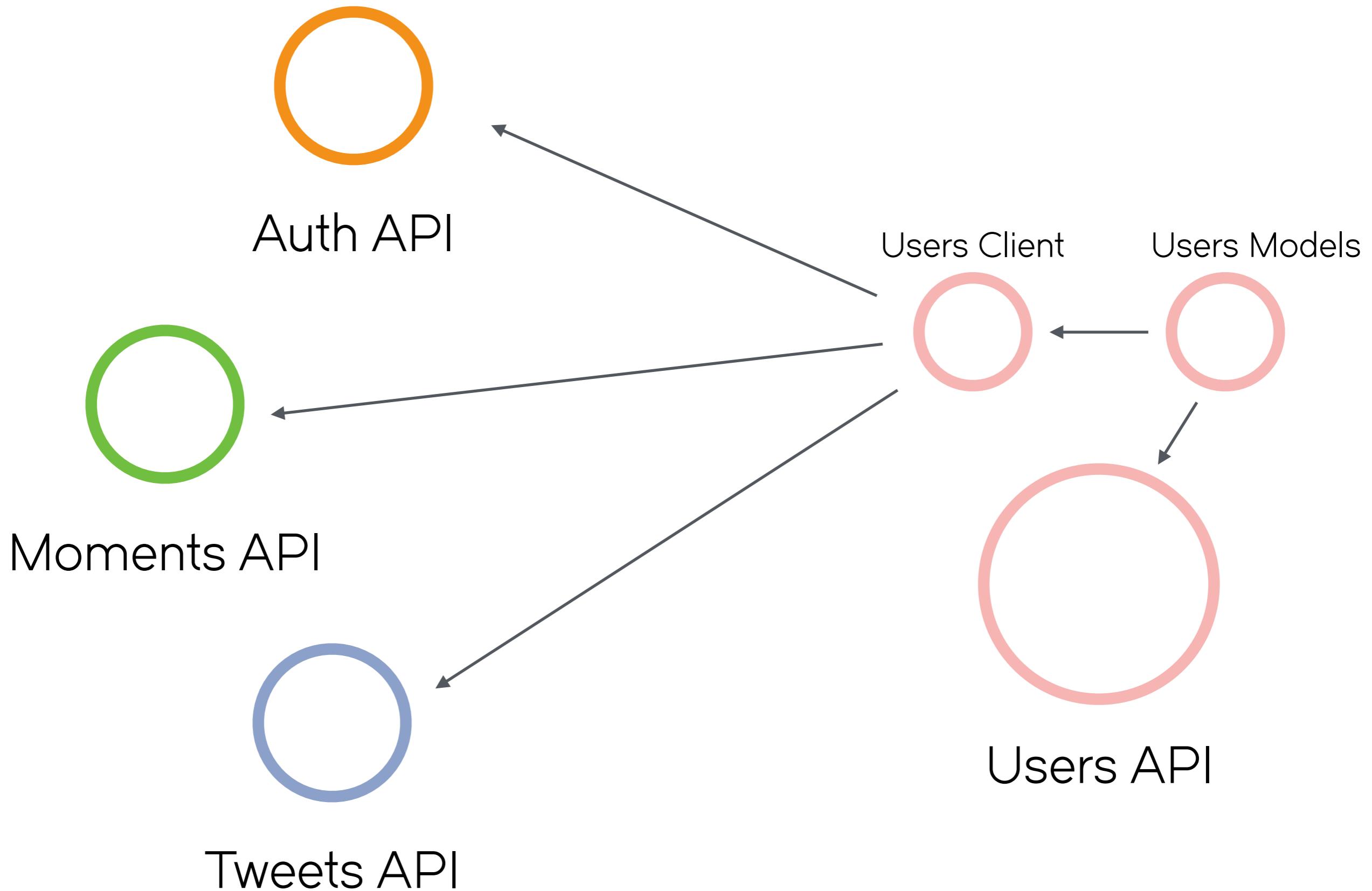
```
3 class TweetController {  
4     func store(_ req: Request) throws -> ResponseRepresentable  
5     guard  
6         let message = req.data["message"]?.string,  
7         let userId = req.data["userId"]?.string  
8     else {  
9         throw Abort(.badRequest)  
10    }  
11  
12    try UserClient.assertUserNotBanned(withId: userId)  
13  
14    let tweet = Tweet(userId: userId, message: message)  
15    try tweet.save()  
16    return tweet  
17 }  
18 }
```



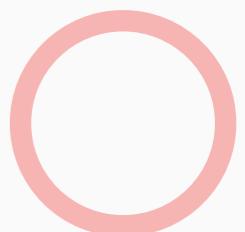
Tweets API







```
1 public final class UserClient {  
2     ...  
3  
4     @available(*, deprecated: 2.0, message: "No longer necessary.")  
5     public static func assertUserNotBanned(withId id: String) throws {  
6         ...  
7     }  
8     ...  
9     ...  
10 }
```



Users Client

The screenshot shows the Xcode IDE with a dark theme. The top bar displays the project name "TweetsAPI", a build status of "Build Succeeded", and a warning icon with the number "1". The file path in the sidebar is "TweetsAPI > Sources > App > Controllers > TweetController.swift". The main editor area contains the following Swift code:

```
1 import Vapor
2 import HTTP
3
4 class TweetController {
5     func store(_ req: Request) throws -> ResponseRepresentable {
6         guard
7             let message = req.data["message"]?.string,
8             let userId = req.data["userId"]?.string
9         else {
10             throw Abort(.badRequest)
11         }
12
13         try UserClient.assertUserNotBanned(withId: userId)
14             // 'assertUserNotBanned(withId:)' is deprecated: No longer necessary.
15         let tweet = Tweet(userId: userId, message: message)
16         try tweet.save()
17         return tweet
18     }
19 }
20
21
22
```

A yellow warning bar highlights the line "try UserClient.assertUserNotBanned(withId: userId)". A tooltip above the bar states: "'assertUserNotBanned(withId:)' is deprecated: No longer necessary."



Tweets API

Building Big Projects with Vapor

DRY, use modules and extensions

DRY, use modules and extensions

```
1 drop.get("profile") {  
2     drop.view.make("profile.leaf", [  
3         "name": "\\" + user.firstName + user.lastName" + "  
4     ])  
5 }
```

DRY, use modules and extensions

```
1 extension User {  
2     var fullName: String {  
3         return "\$(user.firstName) \$(user.lastName)"  
4     }  
5 }  
6  
7 drop.get("profile") {  
8     drop.view.make("profile.leaf", [  
9         "name": user.fullName  
10    ])  
11 }
```

Building Big Projects with Vapor

DRY, use modules and extensions

Avoid direct JSON / Strings

Avoid direct JSON / Strings



```
1 let user = try User(json: json)
2 guard !user.isBanned else {
3     ...
4 }
```



```
1 guard let isBanned = json["isBanned"]?.bool else {
2     ...
3 }
4
5 guard !isBanned else {
6     ...
7 }
```

Building Big Projects with Vapor

DRY, use modules and extensions

Avoid direct JSON / Strings

The compiler is your friend

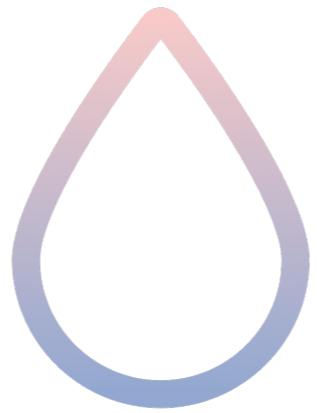
The compiler is your friend



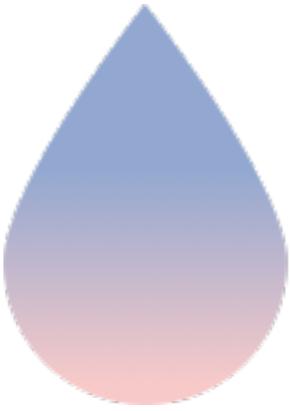
```
1 try UsersClient.assertUserNotBanned(withId: userId)
```



```
1 let res = try drop.client.get("https://api.twitter.com/users/\(userId)")  
2 let user = try User(json: res.json)  
3 guard !user.isBanned else {  
4     throw Abort(.unauthorized)  
5 }
```



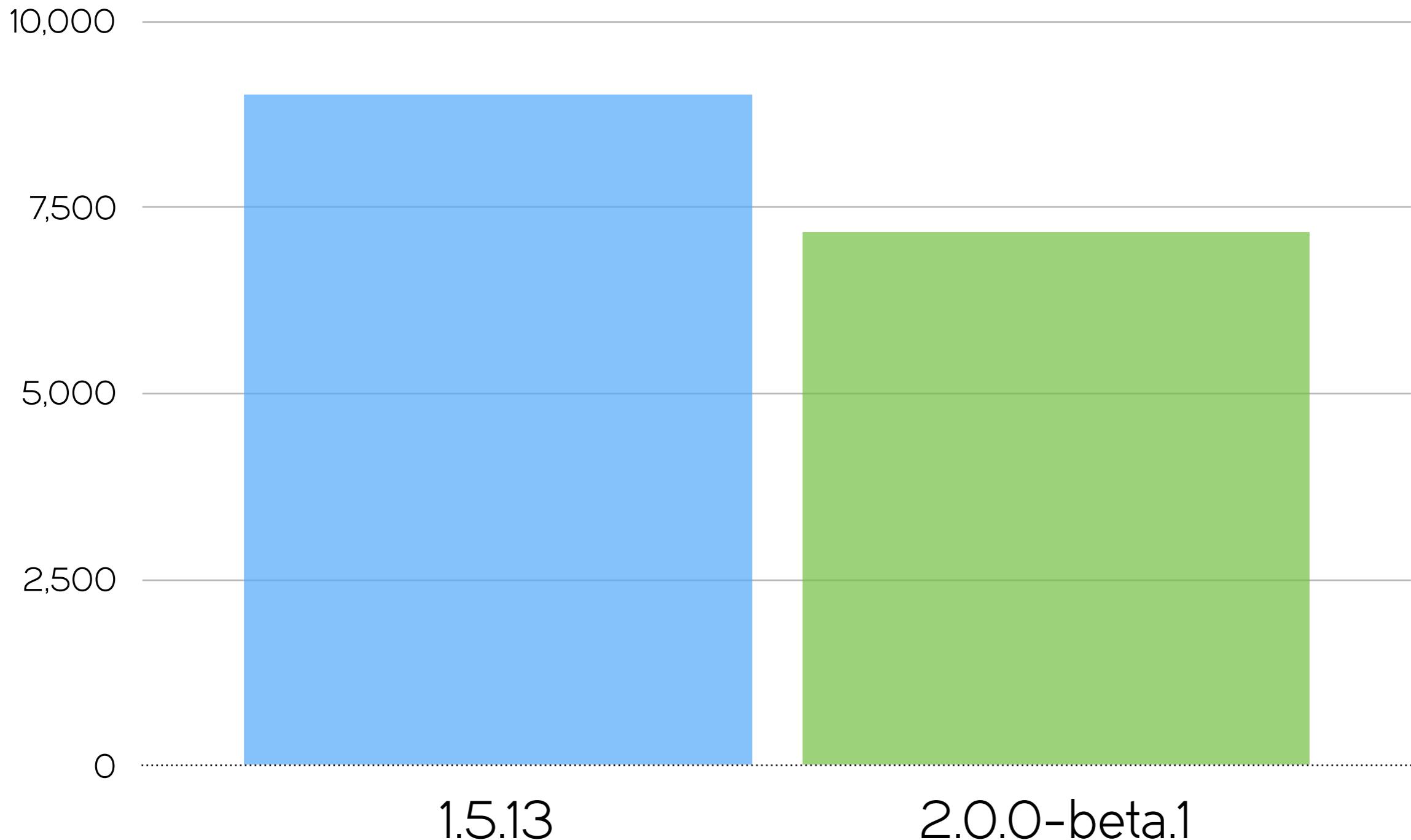
VAPOR



Vapor²

Less Code

Vapor 2 is lighter and more modular



More Features

>50 Alpha Iterations

The screenshot shows a GitHub repository page for the project `vapor / vapor`. The top navigation bar includes links for `Pull requests`, `Issues`, and `Gist`. Below the header, there are buttons for `Unwatch` (379), `Unstar` (9,096), `Fork` (515), and `Settings`. The main content area displays a pre-release named `Vapor 2.0 Alpha 24`, released by `LoganWright` 6 days ago. The release notes mention three commits to master since the release. A bulleted list highlights new features: Validation will be provided, Providers can now include resources, and Easier config load. Below the release, there are links for `#921`, `#922`, and `#920`. The `Downloads` section offers options to download the source code as a `zip` or `tar.gz`.

vapor / vapor

Code Issues 50 Pull requests 6 Projects 1 Wiki Pulse Graphs Settings

Releases Tags Draft a new release

Pre-release

2.0.0-alpha.24 e5ddced

Vapor 2.0 Alpha 24 Edit

LoganWright released this 6 days ago · 3 commits to master since this release

New:

- Validation will be provided
- Providers can now include resources
- Easier config load

#921 #922 #920

Downloads

Source code (zip)

Source code (tar.gz)

Debuggable

Multiple Validation Errors

Soft Delete

Generic Relations

Mail

Raw

JWT

Swift 3.1

Authorization

Pagination

Improved Modularity

Fluent Storage

UUID

Row

BCrypt

Testing

Faster Compilation

_method

OpenSSL

Cache Expiration

Custom ID Keys

5x Faster Redis

Route List

Node

Double Pivot

SQLite Memory DB

Connections API

Timestamps

Beautiful New Docs

Vapor / Middleware

beta.docs.vapor.codes

Search

GitHub 29 Stars · 79 Forks

Vapor Docs

- Overview
- Getting started
- Vapor
 - Folder Structure
 - Droplet
 - Views
- Controllers
- Middleware
- Validation
- Provider
- Hash
- Log
- Commands
- Settings
- JSON
- Routing
- Fluent
- Auth
- Sessions
- HTTP
- Leaf
- Advanced

Table of contents

- Version Middleware
 - Breakdown
 - Request
 - Errors
 - Route Groups
 - Configuration
 - Advanced
 - Extensions
 - Middleware
 - Response
 - Usage
 - Response Representable

Version Middleware

As an example, let's create a middleware that will add the version of our API to each response. The middleware would look something like this:

```
final class VersionMiddleware: Middleware {
    func respond(to request: Request, chainingTo next: Responder) throws -> Response {
        let response = try next.respond(to: request)

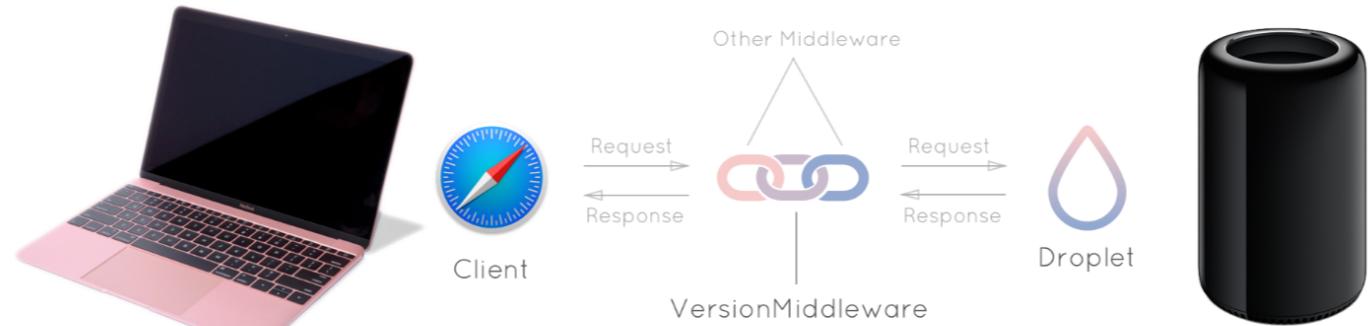
        response.headers["Version"] = "API v1.0"

        return response
    }
}
```

We then supply this middleware to our [Droplet](#).

```
let drop = Droplet()
drop.middleware.append(VersionMiddleware())
```

You can imagine our `VersionMiddleware` sitting in the middle of a chain that connects the client and our server. Every request and response that hits our server must go through this chain of middleware.



Beautiful New Docs

Vapor / Middleware

bcrypt

6 search results

BCryptHasher

BCrypt is a password hashing function that automatically incorporates salts and offers a configurable work factor. The work factor can be...

Manual

You can manually assign a BCryptHasher to drop.hash. let drop = try Droplet() drop.hash = BCryptHasher(workFactor: ...)

Configuration

To use the BCryptHasher change the "hash" key in the droplet.json configuration file.

```
Config/droplet.json { ..., hash : ... }
```

Hash

Hash Hashing is a one way method of converting arbitrary data into a fixed size format. Unlike ciphers, data that is hashed cannot be...

Included

Here is a list of all the packages and modules included with Vapor. Tip While these packages are included in Vapor by default, they can...

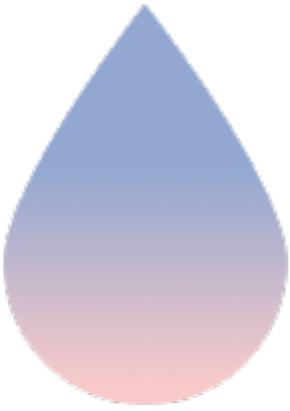
Overview

Vapor Documentation This is the documentation for Vapor, a Web Framework for Swift that works on iOS, macOS, and Ubuntu; and all of the...

Table of contents

- Version Middleware
- Breakdown
- Request
- Errors
- Route Groups
- Configuration
- Advanced
- Extensions
- Middleware
- Response
- Usage
- Response Representable

The screenshot shows a web browser window with the URL `beta.docs.vapor.codes`. The page title is "Vapor / Middleware". A search bar at the top contains the query "bcrypt". Below the search bar, it says "6 search results". The first result is titled "BCryptHasher" with a brief description. Subsequent results include "Manual", "Configuration", "Hash", "Included", and "Overview". On the right side of the main content area, there is a vertical sidebar titled "Table of contents" containing a list of Vapor components. At the bottom of the page, there is a diagram illustrating the middleware stack. It shows a "Client" (represented by a laptop icon) interacting with "Other Middleware" (represented by a red and blue chain icon). This "Other Middleware" then interacts with "VersionMiddleware" (represented by a blue teardrop icon). Finally, "VersionMiddleware" interacts with a "Droplet" (represented by a black cylinder icon).



Vapor²



#vapor2

BETA

Available Today
beta.docs.vapor.codes



#beta



vapor.codes



vapor.team



@codevapor
#vaporswift



docs.vapor.codes



Questions?



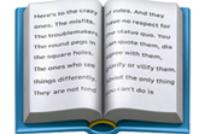
vapor.codes



vapor.team



@codevapor
#vaporswift



docs.vapor.codes

