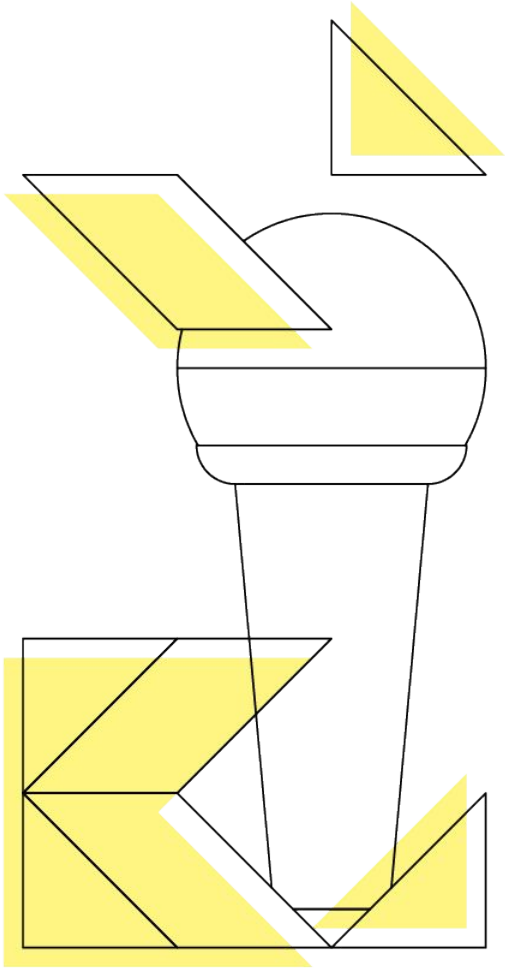
Connecting application with Ktor

David Córdova Cardoza
Android Dev, GDG, KUG
@ChubyAvodroc

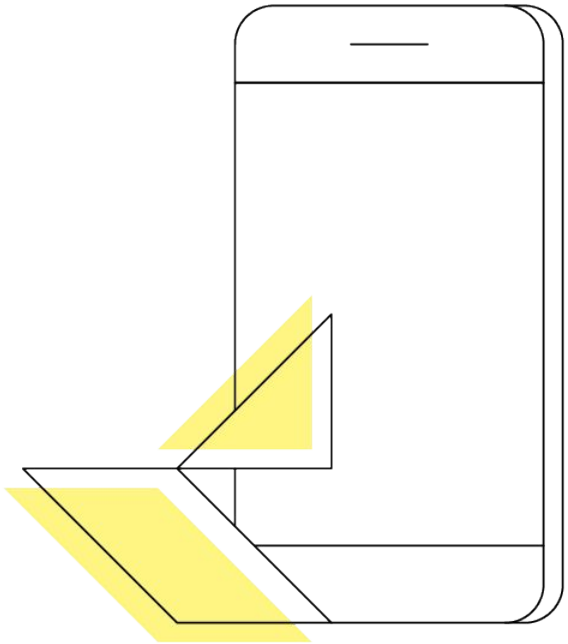


So, what is Ktor?

Ktor is a framework, created by JetBrains, for building asynchronous servers and clients in connected systems using the powerful Kotlin programming language. ¯_(ツ)_/¯

It uses a DSL and Coroutines, by using coroutines Ktor is very performance, you know, coroutines stuff FTW, by using a DSL is very easy to read and understand FTW x2

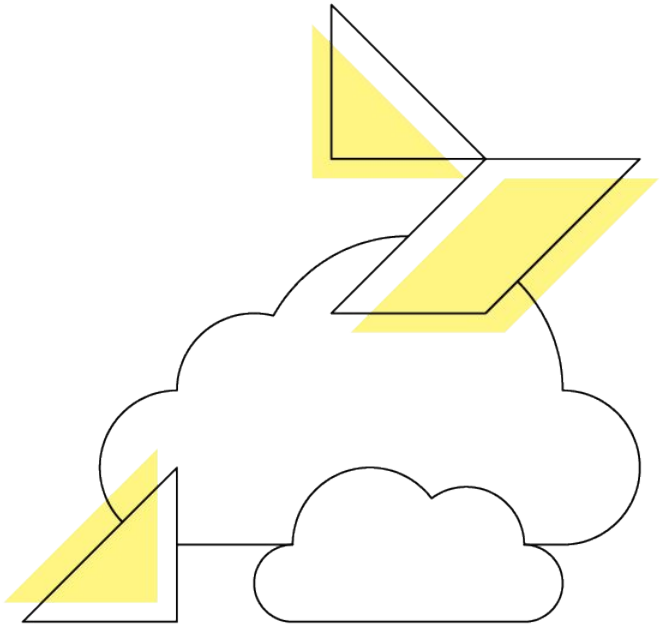
Current version: 1.2.3 ¯_(ツ)_/¯ x2



How to... Ktor

There are several ways, but 3 of them are very easy.

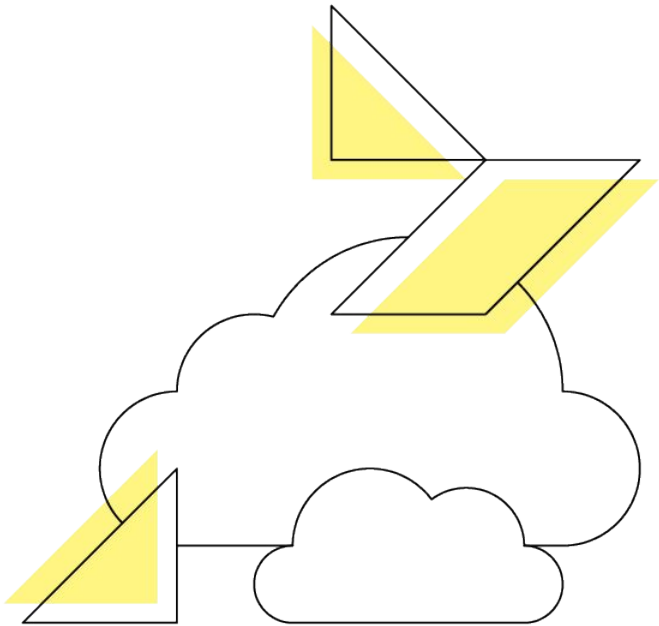
1. Web



How to... Ktor

There are several ways, but 3 of them are very easy.

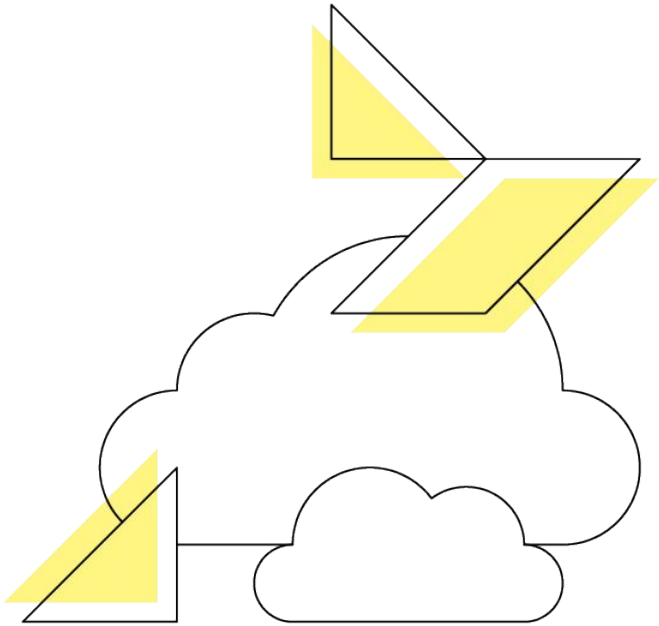
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#



How to... Ktor

There are several ways, but 3 of them are very easy.

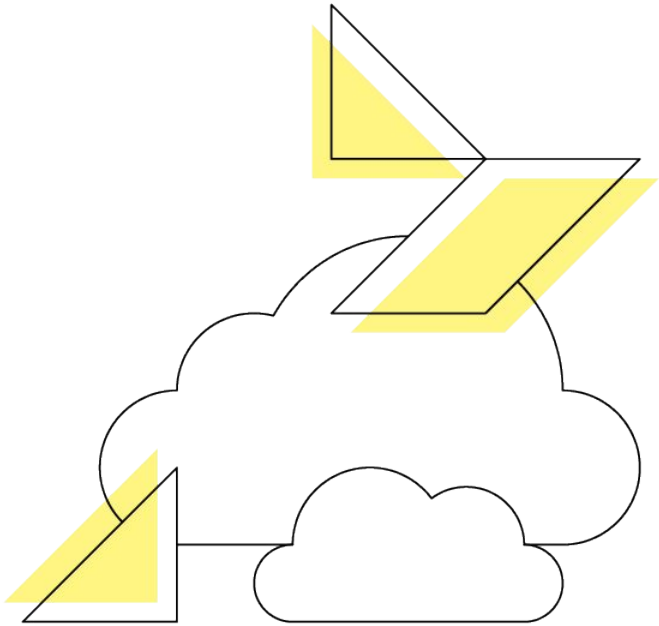
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client



How to... Ktor

There are several ways, but 3 of them are very easy.

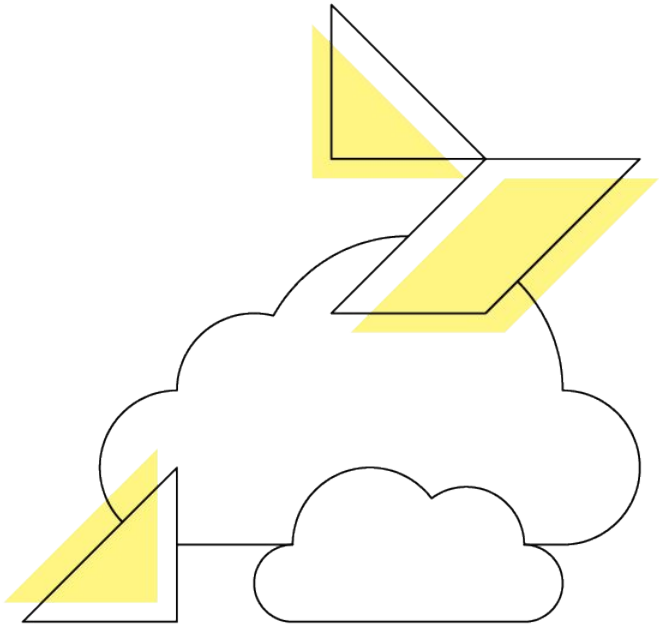
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client
    c. Build project and unzip it



How to... Ktor

There are several ways, but 3 of them are very easy.

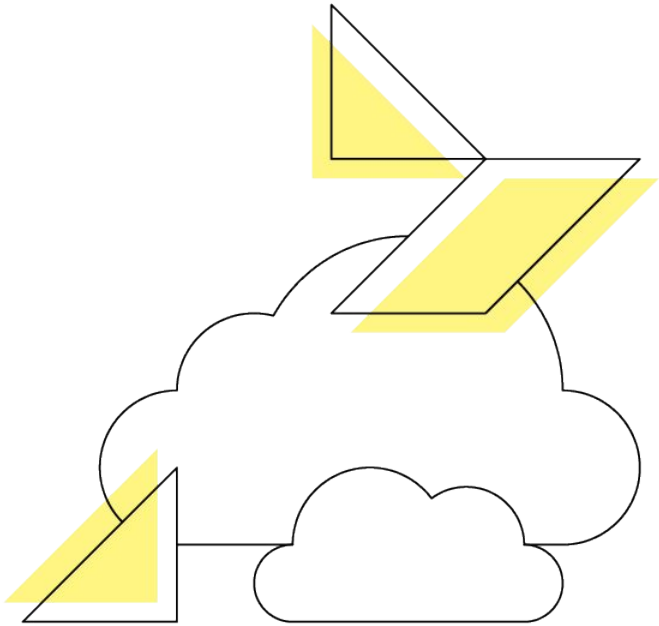
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client
    c. Build project and unzip it
    d. Open project



How to... Ktor

There are several ways, but 3 of them are very easy.

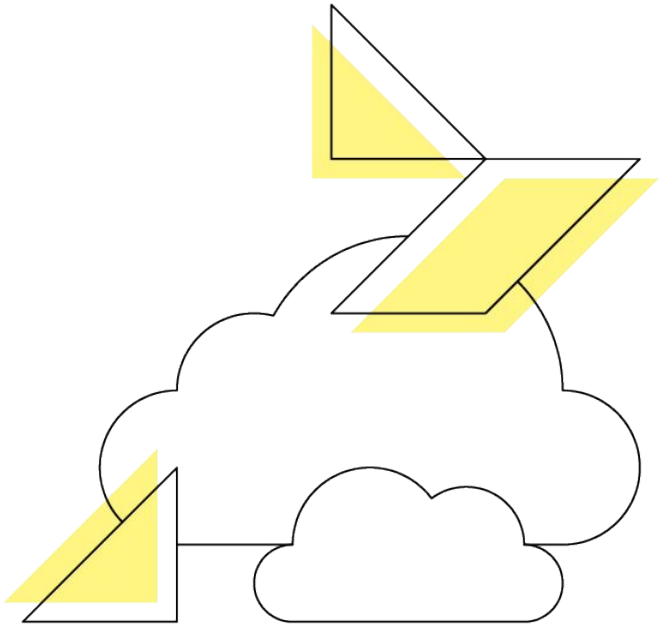
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client
    c. Build project and unzip it
    d. Open project
    e. Run… Be happy you already have your server :raised_hands:



How to... Ktor

There are several ways, but 3 of them are very easy.

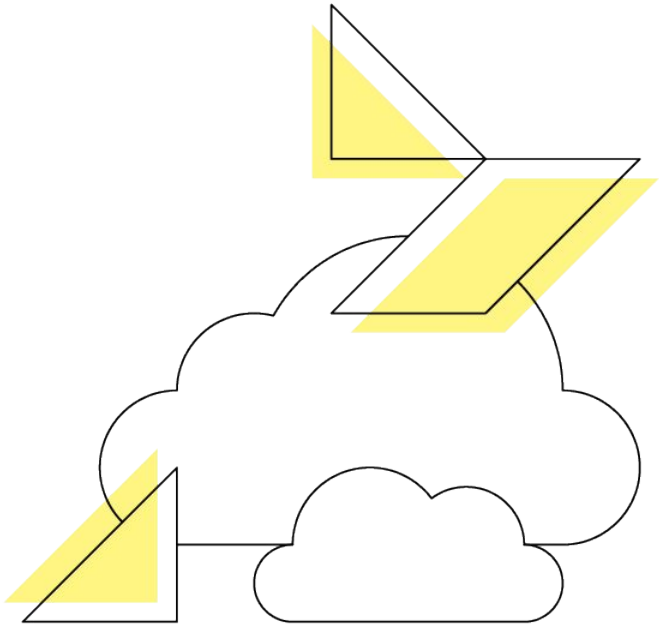
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client
    c. Build project and unzip it
    d. Open project
    e. Run… Be happy you already have your server :raised_hands:
2. Idea Plugin



How to... Ktor

There are several ways, but 3 of them are very easy.

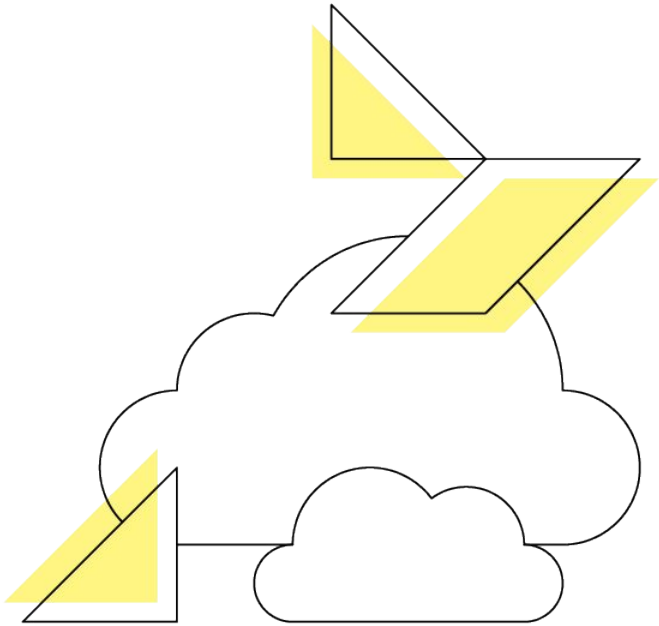
1. Web
   a. Go to https://ktor.io/quickstart/generator.html#
   b. Set project properties, features for Server and Client
   c. Build project and unzip it
   d. Open project
   e. Run… Be happy you already have your server :raised_hands:
2. Idea Plugin
   a. Install Ktor plugin on IntelliJ, it does not work on AS : -(



How to... Ktor

There are several ways, but 3 of them are very easy.

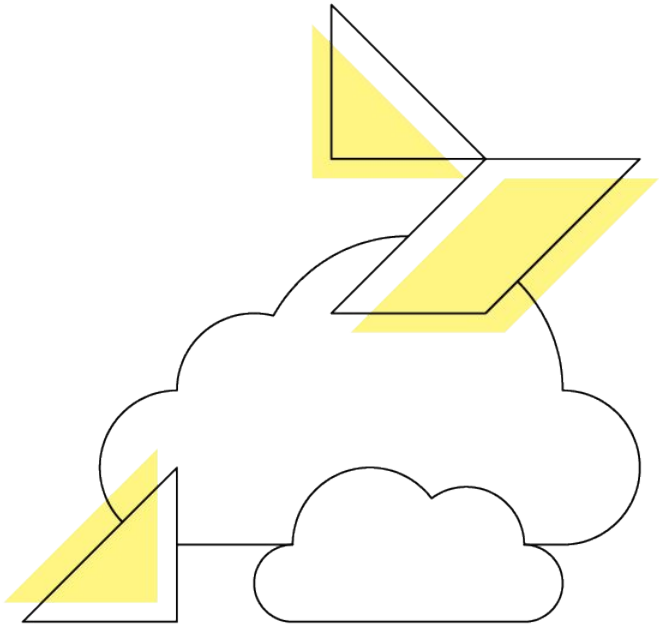
1. Web
    a. Go to https://ktor.io/quickstart/generator.html#
    b. Set project properties, features for Server and Client
    c. Build project and unzip it
    d. Open project
    e. Run… Be happy you already have your server :raised_hands:
2. Idea Plugin
    a. Install Ktor plugin on IntelliJ, it does not work on AS : -(
    b. Follow web steps, seriously, they are the same…



How to... Ktor

There are several ways, but 3 of them are very easy.

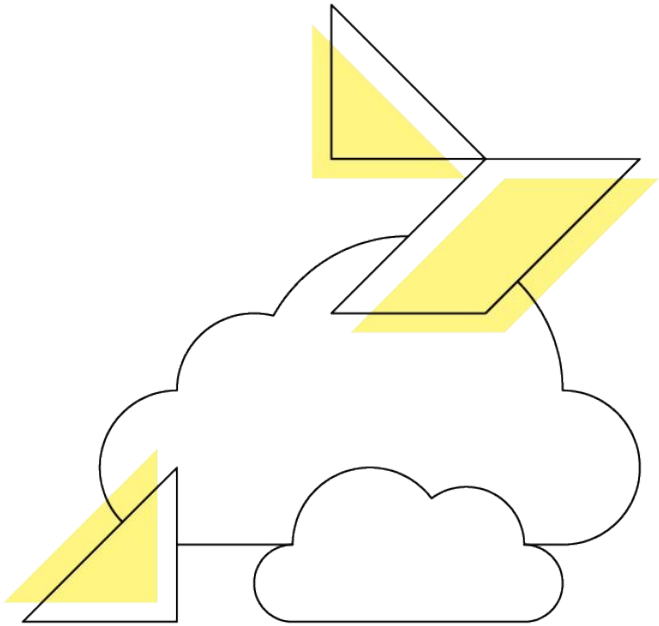
1. Web
   a. Go to https://ktor.io/quickstart/generator.html#
   b. Set project properties, features for Server and Client
   c. Build project and unzip it
   d. Open project
   e. Run… Be happy you already have your server :raised_hands:
2. Idea Plugin
   a. Install Ktor plugin on IntelliJ, it does not work on AS : -(
   b. Follow web steps, seriously, they are the same…
3. From scratch



How to... Ktor

There are several ways, but 3 of them are very easy.

1. Web
   a. Go to https://ktor.io/quickstart/generator.html#
   b. Set project properties, features for Server and Client
   c. Build project and unzip it
   d. Open project
   e. Run… Be happy you already have your server :raised_hands:
2. Idea Plugin
   a. Install Ktor plugin on IntelliJ, it does not work on AS : -(
   b. Follow web steps, seriously, they are the same…
3. From scratch
   a. Go and pick one of the two above, its 2019 already

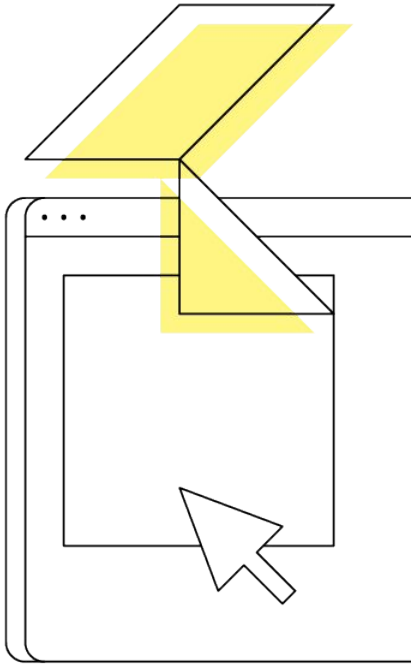


Servers

First things first... the application object.

It's the main actor in our server, basically the guy who is in charge to accept request and return responses.

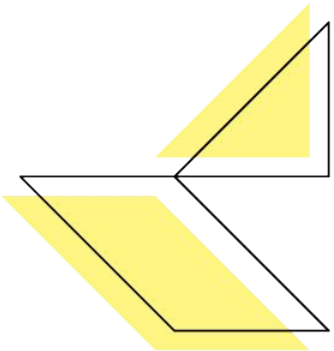
```
fun main(args: Array<String>) {
    val server = embeddedServer(Netty, port = 8080) {
        //application
    }
    server.start(wait = true)
}
```



Engines

We can use any of the following

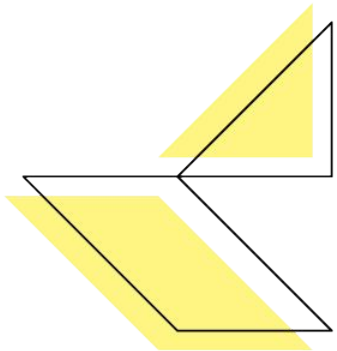
1. Tomcat



Engines

We can use any of the following

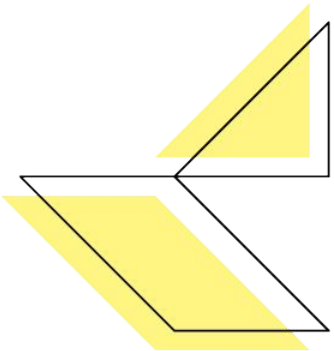
1. Tomcat
2. Jetty



Engines

We can use any of the following

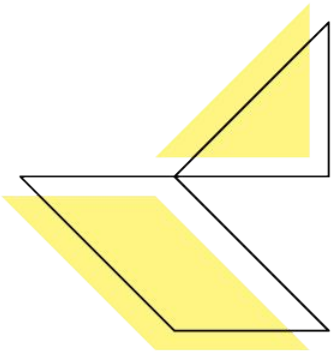
1. Tomcat
2. Jetty
3. Netty



Engines

We can use any of the following

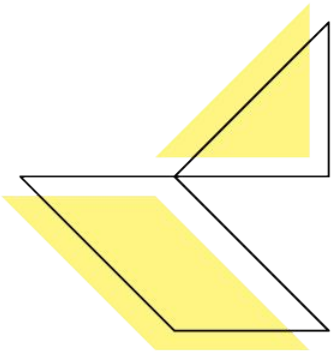
1. Tomcat
2. Jetty
3. Netty
4. CIO



Engines

We can use any of the following

1. Tomcat
2. Jetty
3. Netty
4. CIO
5. Mock

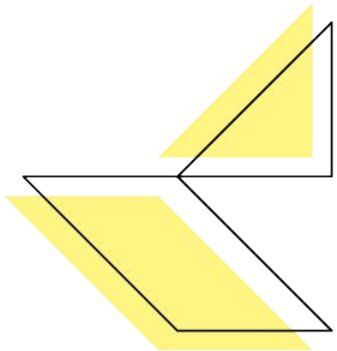


Engines

We can use any of the following

1. Tomcat
2. Jetty
3. Netty
4. CIO
5. Mock

Each engine can be configured in their own way thanks to final extension function on embeddedServer. Also it can be configured through a module extension function on Application

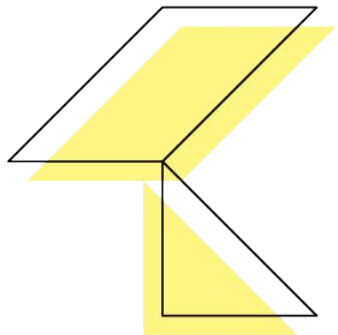


Features

Basically add functionality to the server, there are a lot of them already built in and other needs to be added as a dependency... I'm looking at you Gradle

Some features are:

- Routing

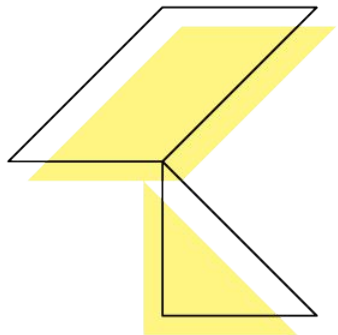


Features

Basically add functionality to the server, there are a lot of them already built in and other needs to be added as a dependency... I'm looking at you Gradle

Some features are:

- Routing
- Logging

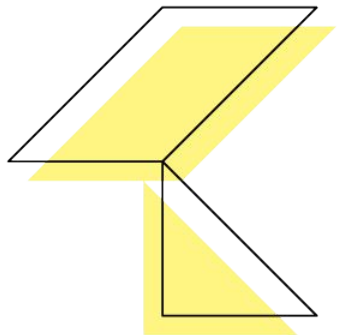


Features

Basically add functionality to the server, there are a lot of them already built in and other needs to be added as a dependency... I'm looking at you Gradle

Some features are:

- Routing
- Logging
- Status Pages

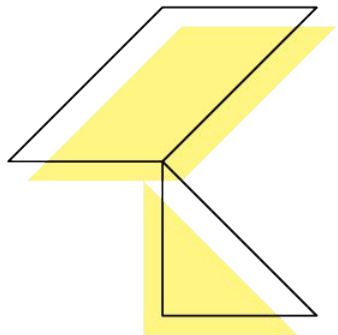


Features

Basically add functionality to the server, there are a lot of them already built in and other needs to be added as a dependency... I'm looking at you Gradle

Some features are:

- Routing
- Logging
- Status Pages
- Authentication



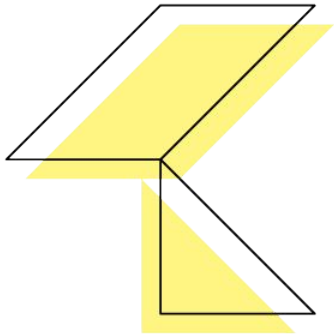
Features

Basically add functionality to the server, there are a lot of them already built in and other needs to be added as a dependency... I'm looking at you Gradle

Some features are:

- Routing
- Logging
- Status Pages
- Authentication

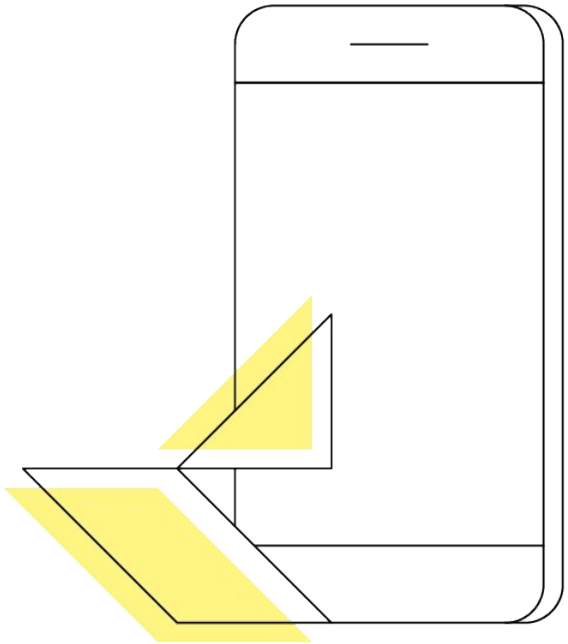
```
install(DefaultHeaders)
    install(CallLogging)
    install(Routing) {
        get("/") {
            call.respondText("Hello, World!")
        }
    }
```



Request

When the application is handling routes gets access to an ApplicationCall “call” as part of the application context, this call is in a nutshell a wrapper for both request and responses.

From the call we get access to the actual request and its properties such as headers, cookies, body, url etc

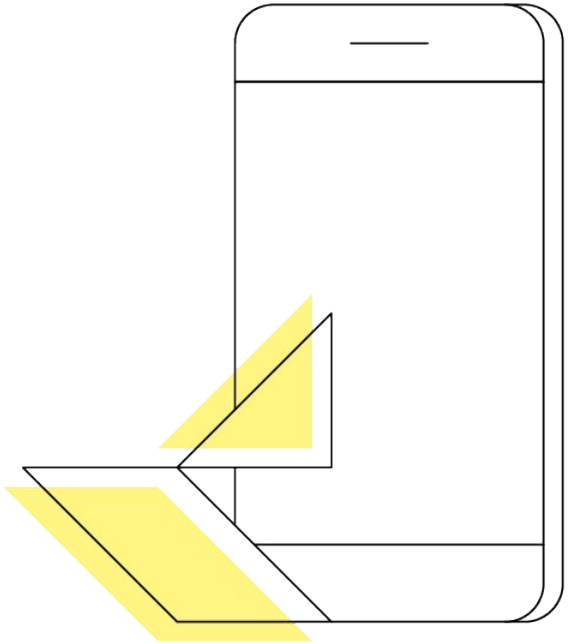


Request

Some properties are part of the call itself, for example parameters or attributes.

To handle HTTP verbs there are several extension functions that makes really easy to create an API.

```
get("/attendees") {
        val request = call.request
        println(request.headers)
```

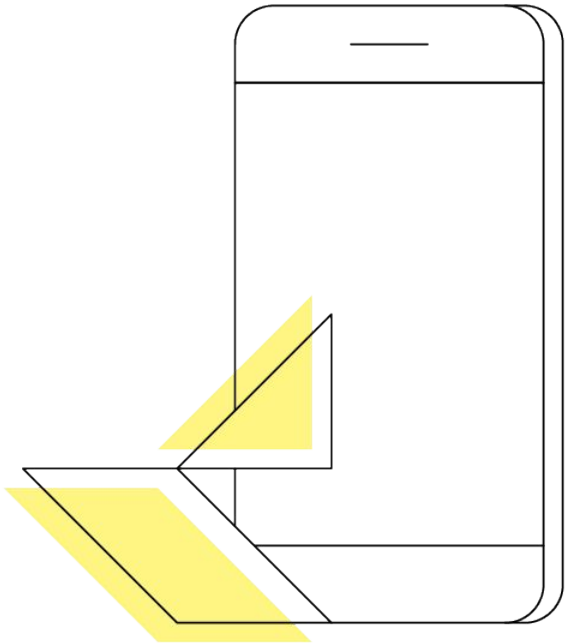


Request

For Https verbs such as PUT, PATCH and POST we can retrieve the body with one of the following methods

```
val channel: ByteReadChannel = call.receiveChannel()
val text: String = call.receiveText()
val inputStream: InputStream = call.receiveStream()
// NOTE: InputStream is synchronous and blocks the thread
val multipart: MultiPartData = call.receiveMultipart()
val postParameters: Parameters = call.receiveParameters()
val obj: T = call.receive<T>()
val obj: T? = call.receiveOrNull<T>()
```

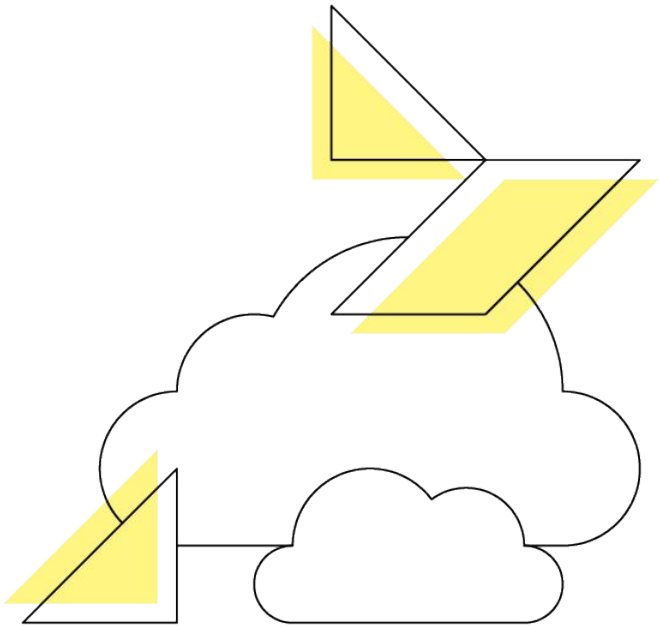
Worth mention that the last two ones need a custom Typed request (Content Negotiation)



Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

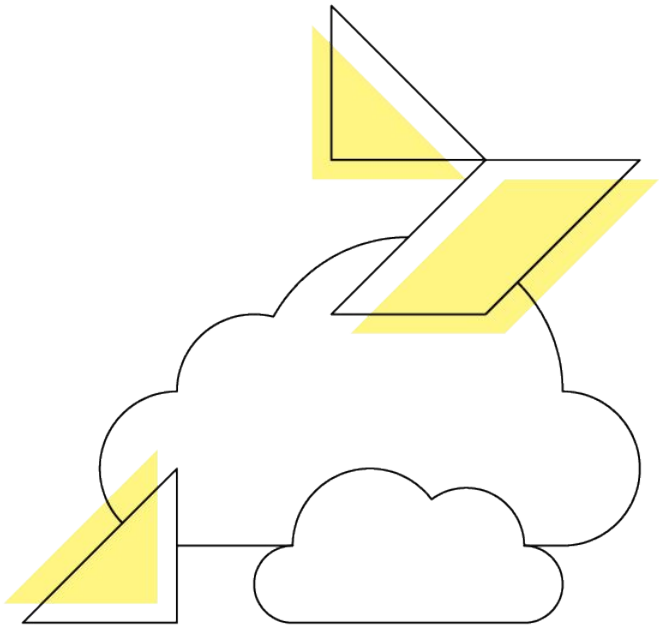


Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

- Jackson

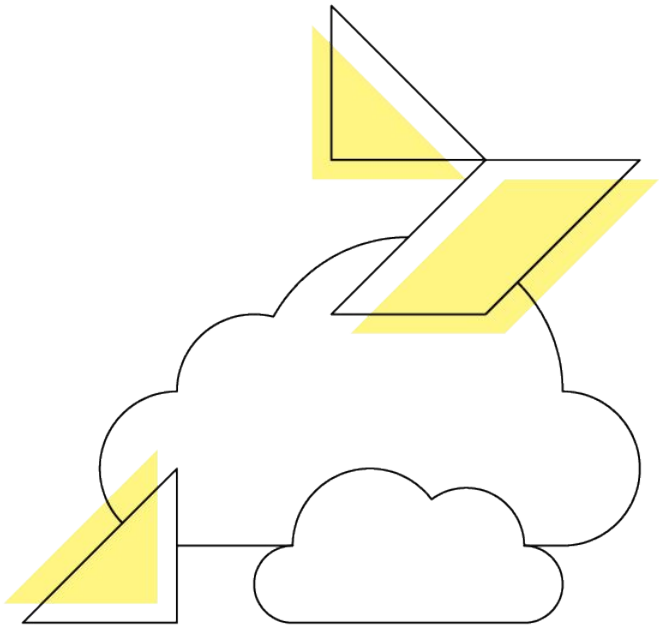


Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

- Jackson
- Moshi

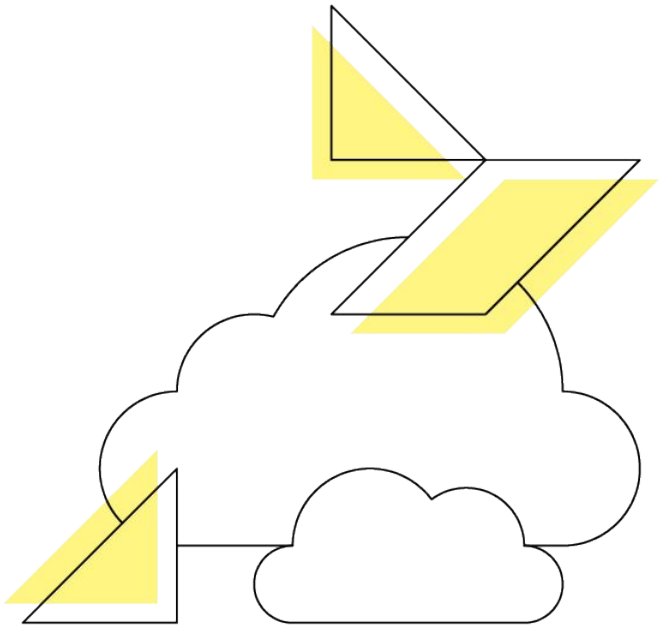


Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

- Jackson
- Moshi
- Gson

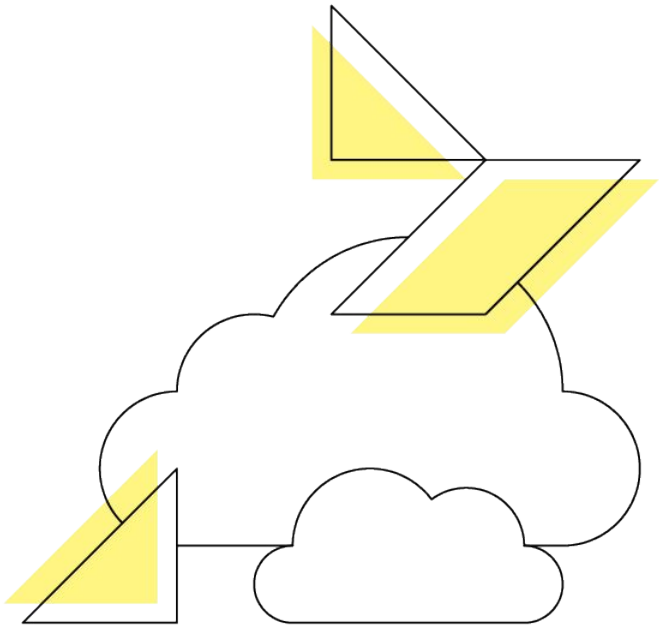


Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

- Jackson
- Moshi
- Gson
- Kotlinx Serializer



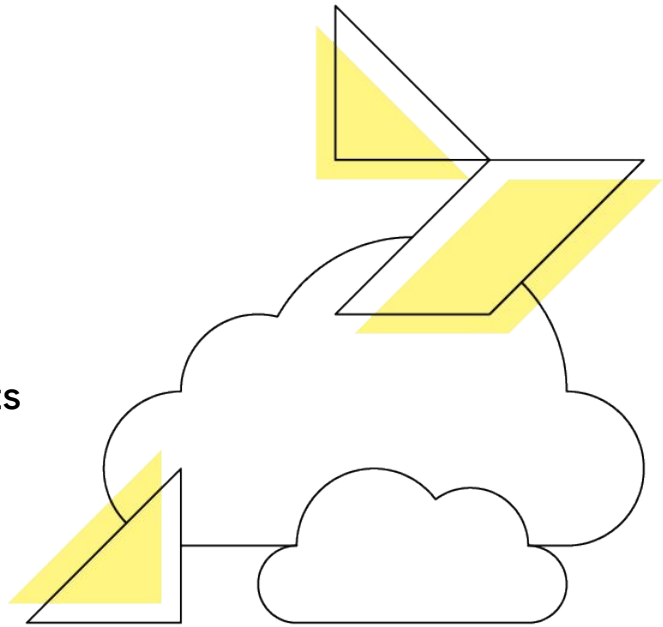
Content Negotiation

It is a feature that will allow us to specify a way in which the server knows how to serialize our objects, as all features it needs to be installed and configured.

Some options are

- Jackson
- Moshi
- Gson
- Kotlinx Serializer

As every feature each one of them can be configured in differents ways, just remember to annotate your objects to be serialized

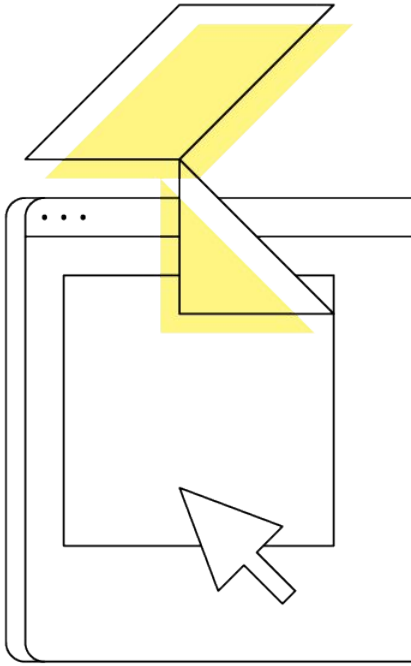


Responses

As the same as Request we can get access to a Response object from the ApplicationCall “call” object that is available to us when managing routes

```
val response = call.response
response.status(HttpStatusCode.OK) - Sets the HttpStatusCode to a predefined standard one
response.status(HttpStatusCode(418, "I'm a tea pot")) - Sets the HttpStatusCode to a custom status code
response.contentType(ContentType.Text.Plain.withCharset(Charsets.UTF_8))
response.header("X-My-Header", "my value") - Appends a custom header
response.header("X-My-Times", 1000) - Appends a custom header
```

Just notice that once the response is sent there’s no way to change its values.

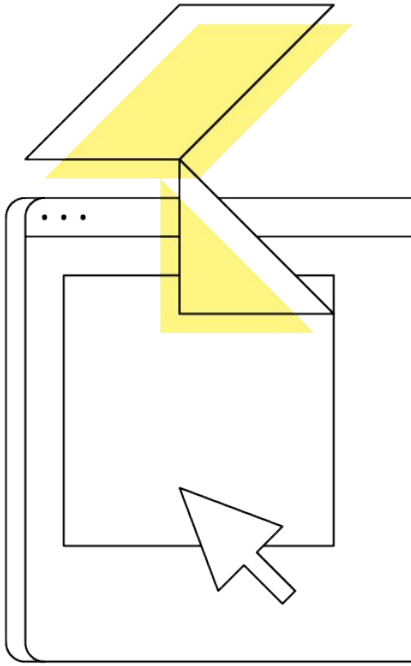


Responses

So how to actually send the response?

Well the “call” object has, again, some convenient methods to do this. Lets look at them.

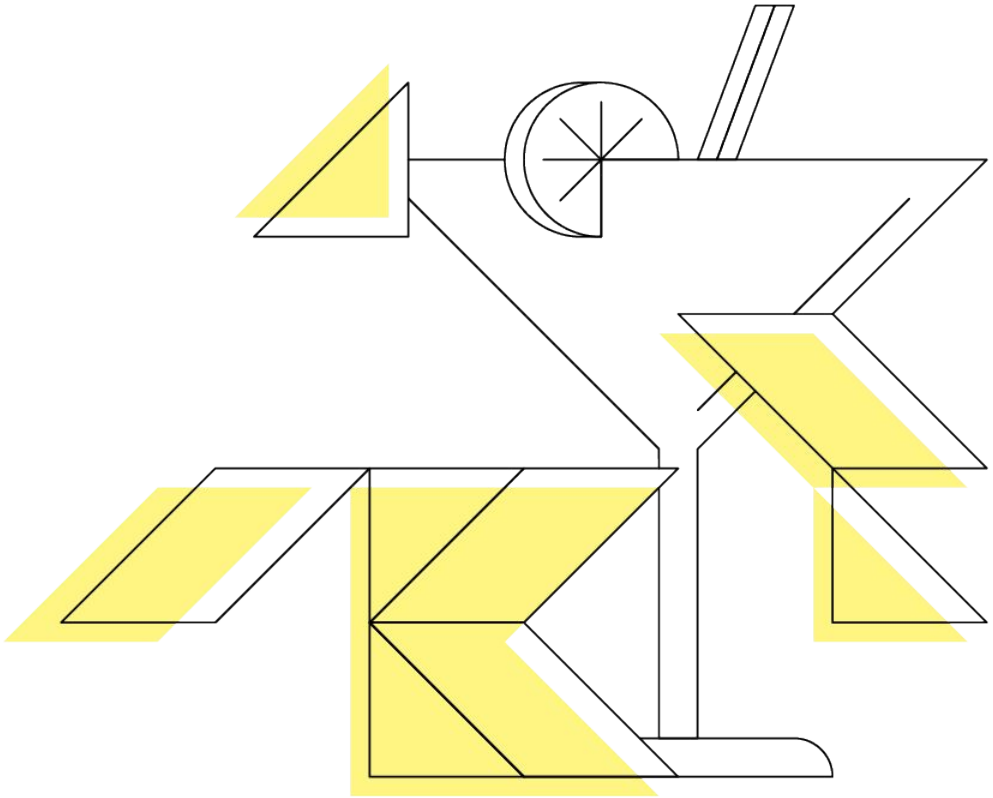
```
call.respond(MyDataClass("hello", "world"))
- Specifies a status code, and sends a payload in a single call.
call.respond(HttpStatusCode.NotFound, MyDataClass("hello", "world"))
call.respondText("text") - Just a string with the body
call.respondText { "string" } - Responding a string with a suspend provider
call.respondText(contentType = ..., status = ...) { "string" }
call.respondBytes(byteArrayOf(1, 2, 3))
call.respondFile(File("/path/to/file"))
```



Demo time

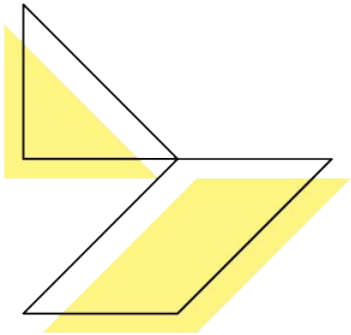
I really, like really, hope this works…….

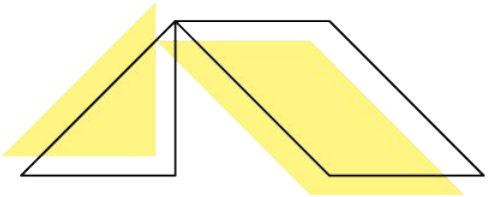
¯_(ツ)_/¯



References

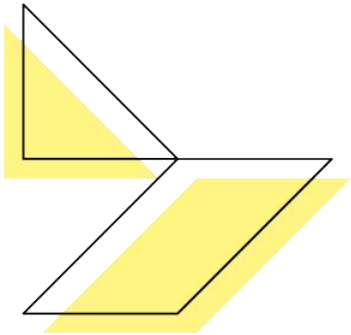
- https://ktor.io/

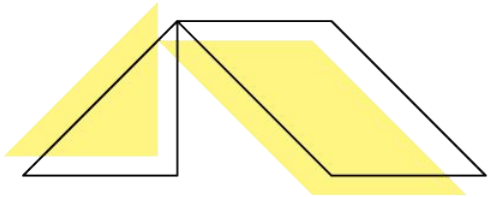




References

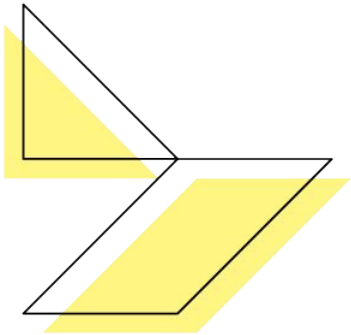
- https://ktor.io/
- https://ktor.io/quickstart/index.html

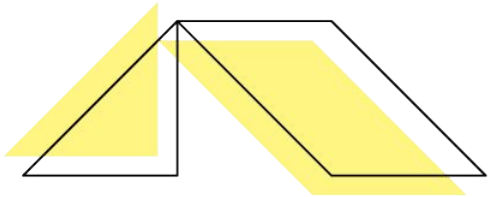




References

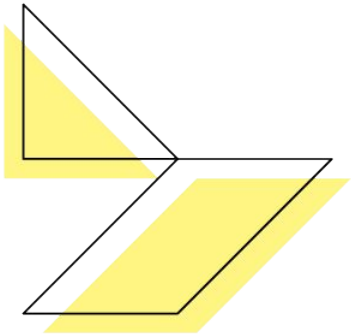
- https://ktor.io/
- https://ktor.io/quickstart/index.html
- https://ktor.io/servers/index.html

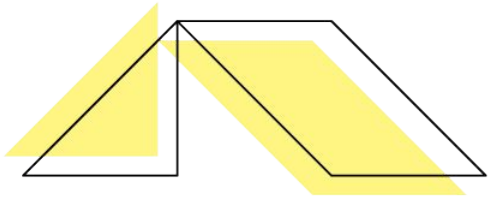




References

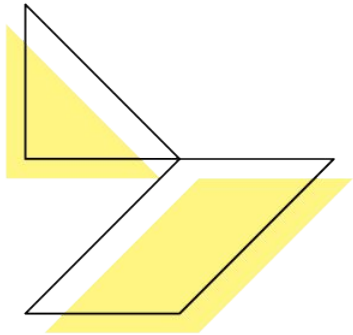
- https://ktor.io/
- https://ktor.io/quickstart/index.html
- https://ktor.io/servers/index.html
- https://ktor.io/clients/index.html

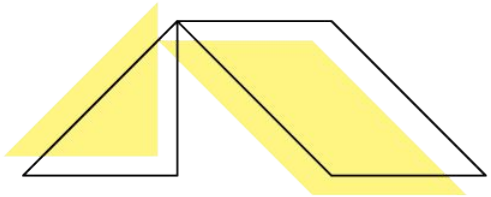




References

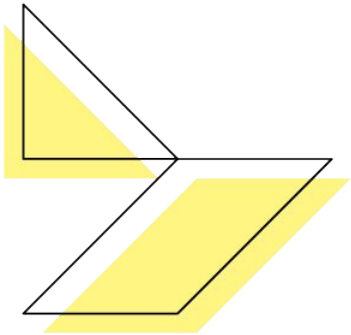
- https://ktor.io/
- https://ktor.io/quickstart/index.html
- https://ktor.io/servers/index.html
- https://ktor.io/clients/index.html
- https://www.youtube.com/watch?v=V4PS3IjIzlw

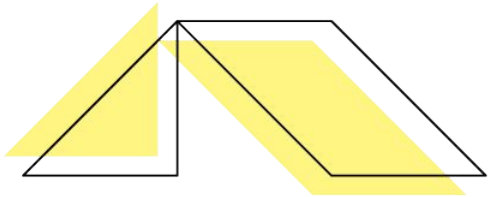




References

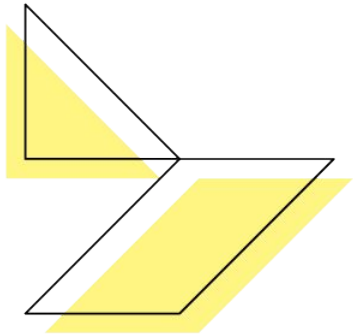
- https://ktor.io/
- https://ktor.io/quickstart/index.html
- https://ktor.io/servers/index.html
- https://ktor.io/clients/index.html
- https://www.youtube.com/watch?v=V4PS3IjIzlw
- https://github.com/JetBrains/kotlinconf-app

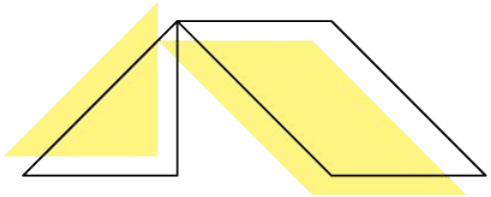




References

- https://ktor.io/
- https://ktor.io/quickstart/index.html
- https://ktor.io/servers/index.html
- https://ktor.io/clients/index.html
- https://www.youtube.com/watch?v=V4PS3IjIzlw
- https://github.com/JetBrains/kotlinconf-app
- https://github.com/ChubyAvodroc/Kotlin-Everywhere





That’s all Folks

THANK YOU