Servers ❤️ Kotlin

Ryan Harter
@rharter

Ktor

Ktor

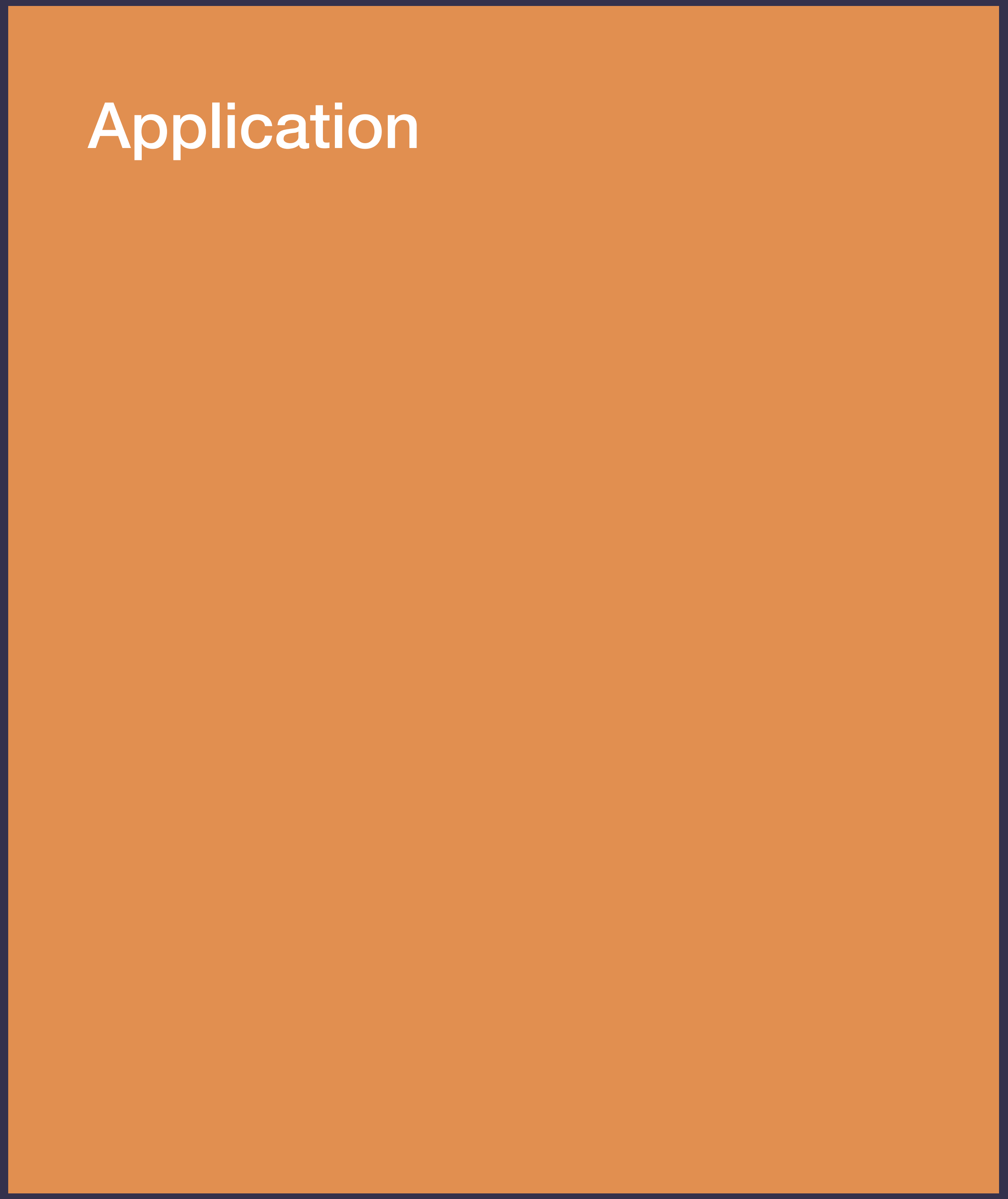
~~Easy to use, fun and asynchronous.~~

Ktor

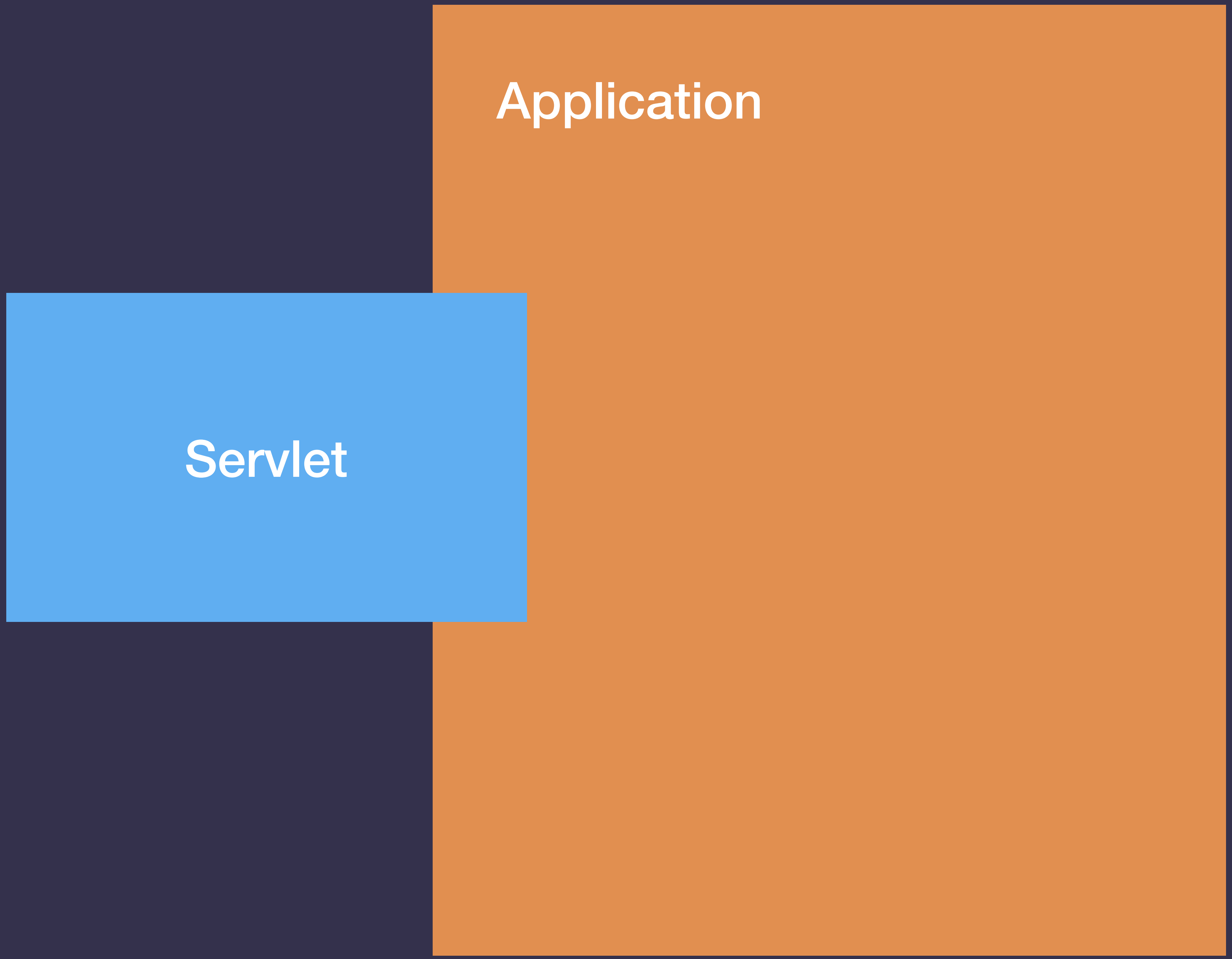
~~Easy to use, fun and asynchronous.~~

Composable, DSL based web services in Kotlin

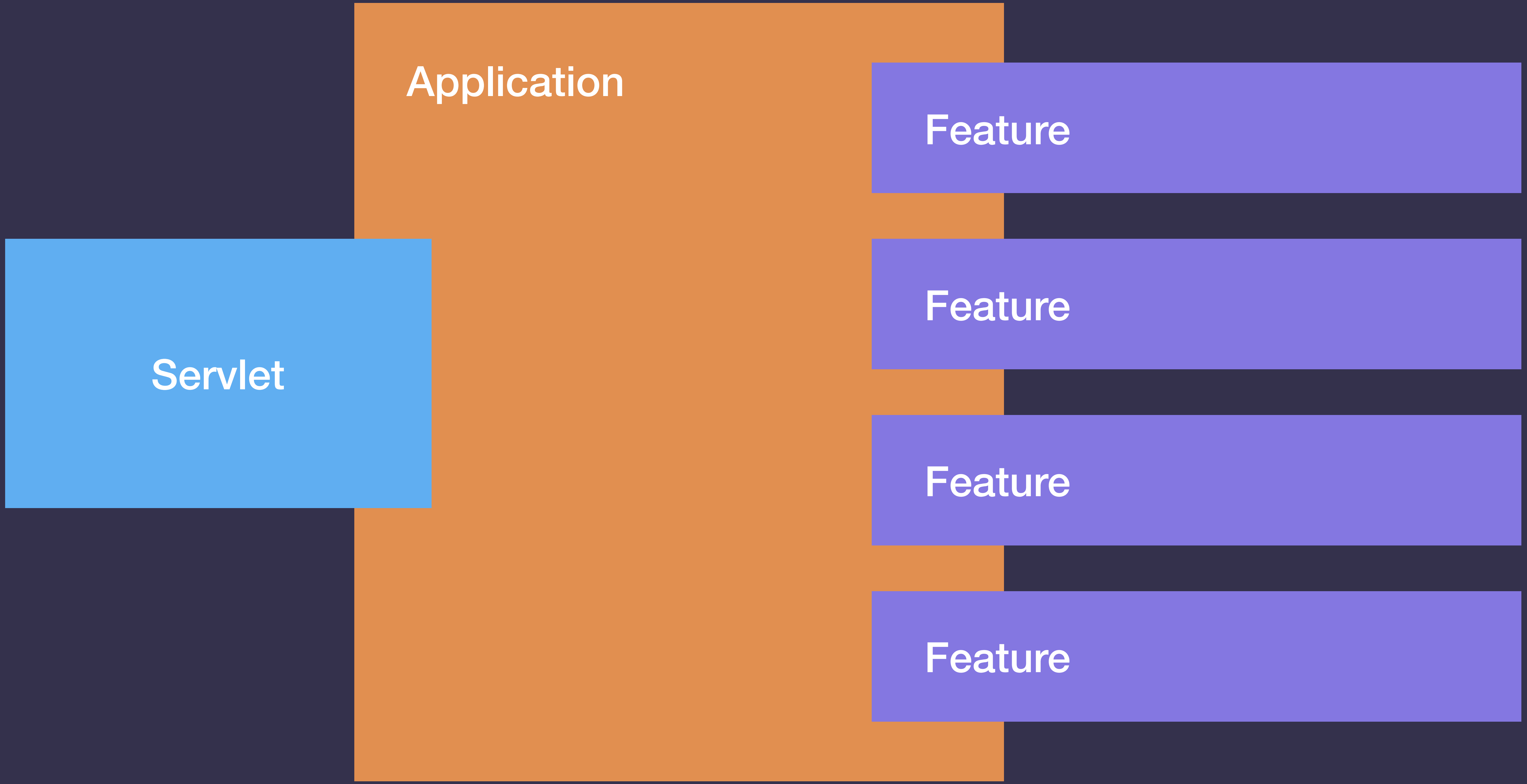
Ktor



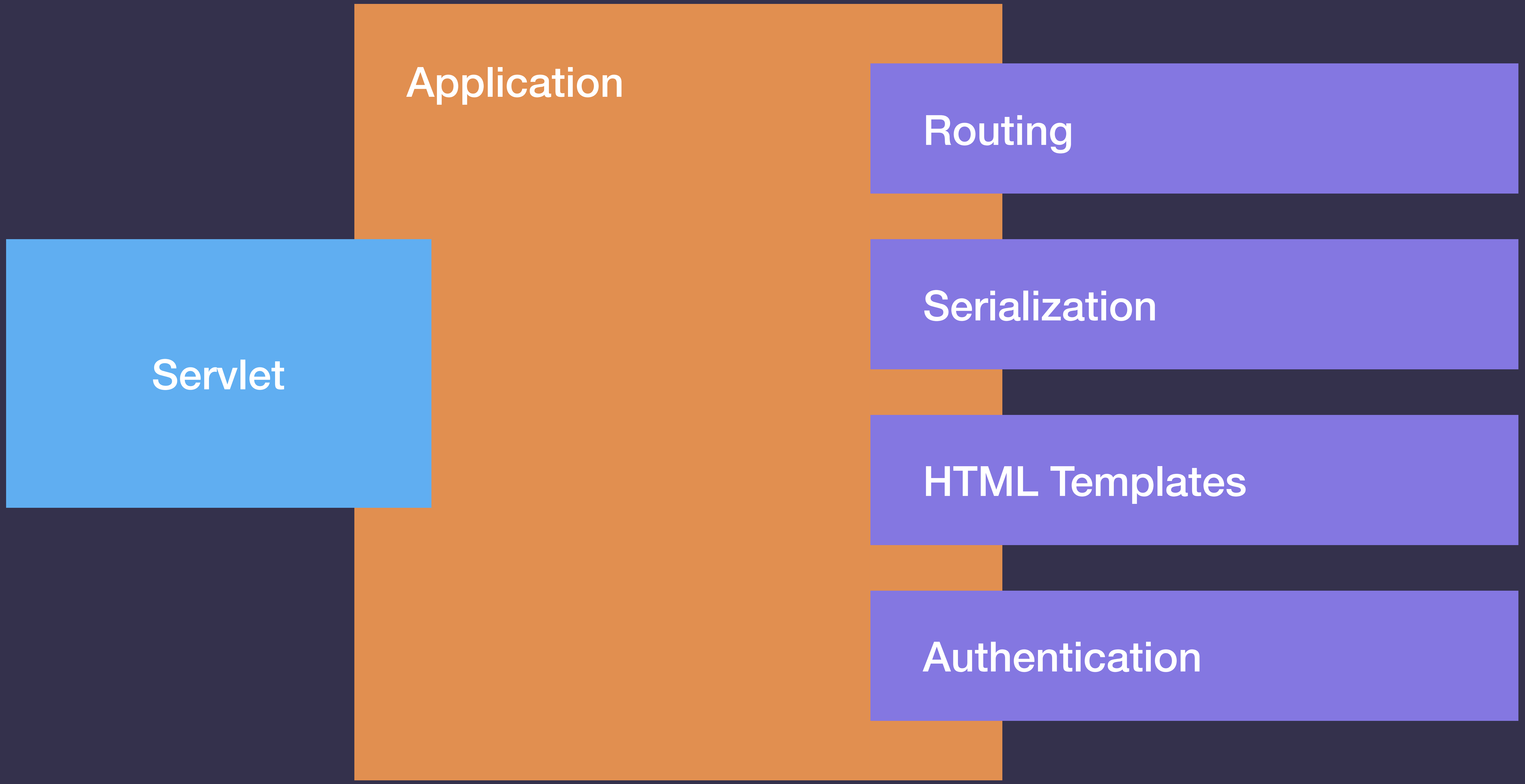
Ktor



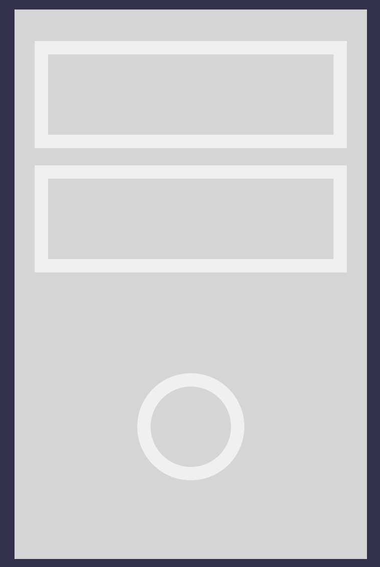
Ktor



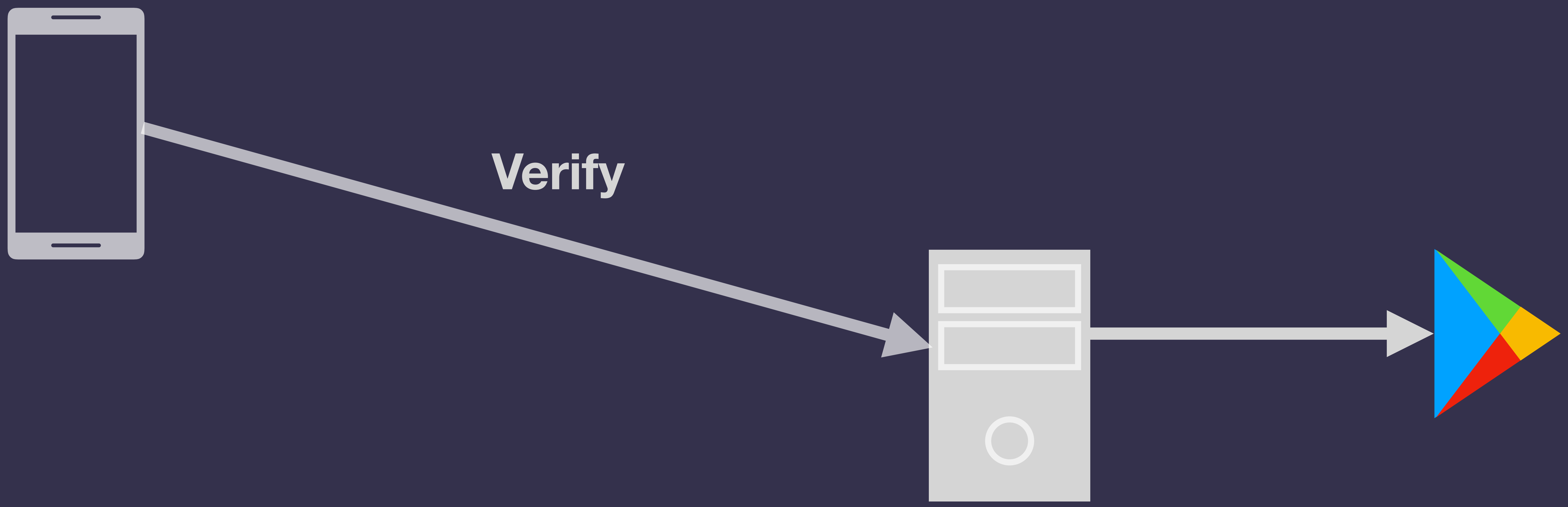
Ktor


Application
Routing
Serialization
HTML Templates
Authentication
Servlet

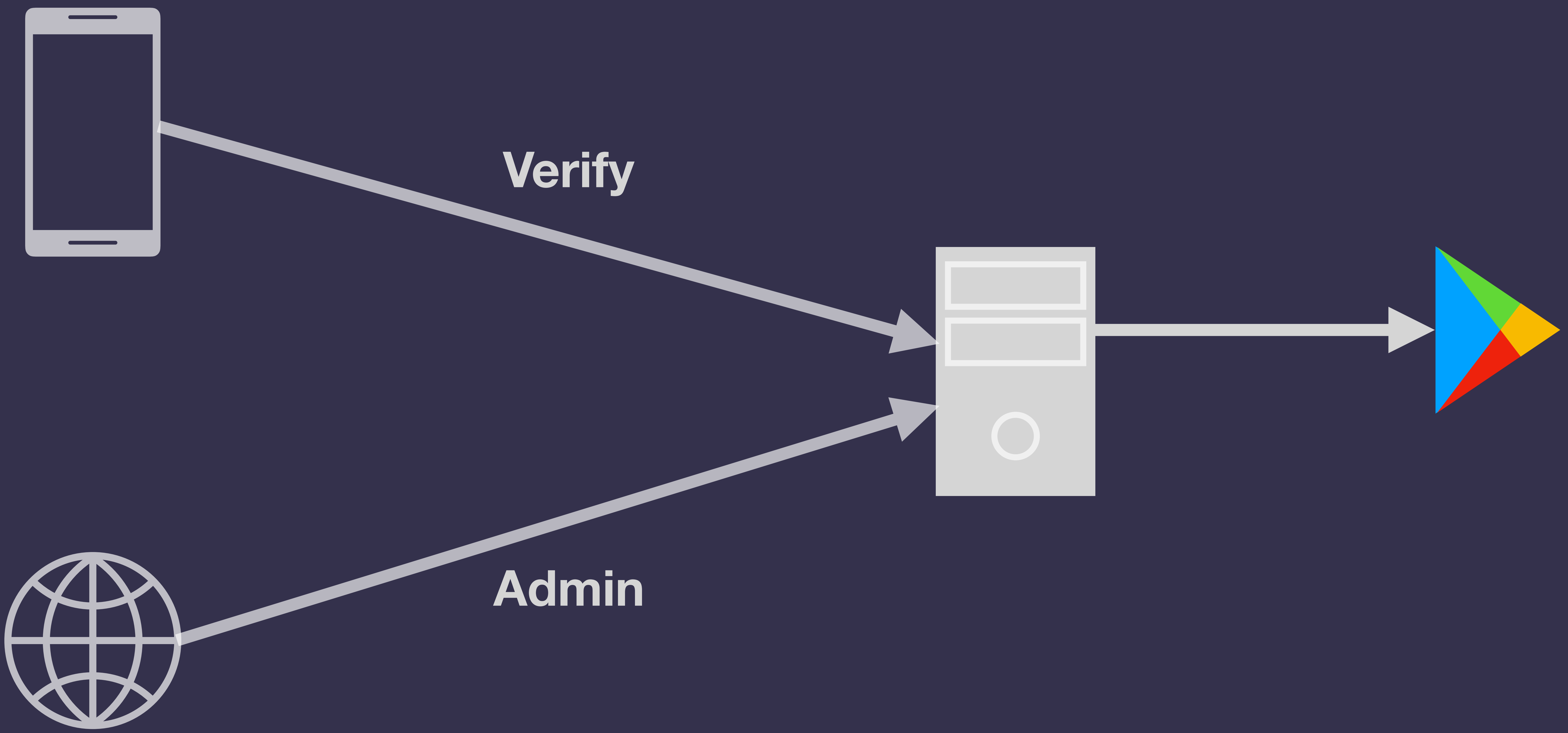
Ktor



Ktor



Ktor



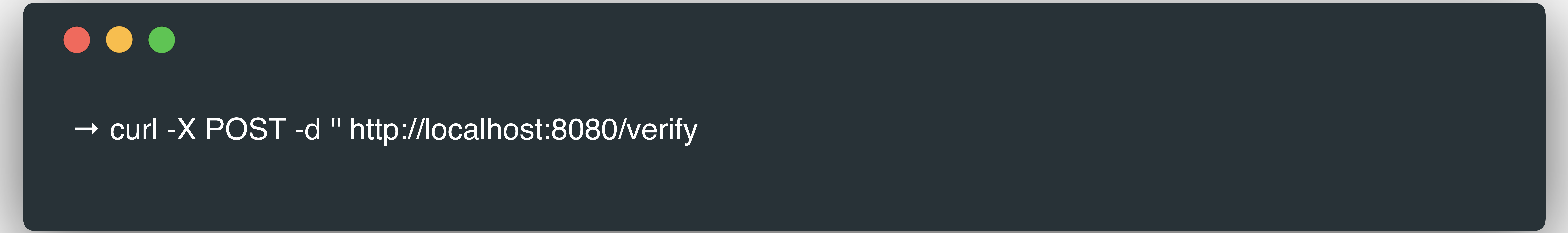
```
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```

```
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```

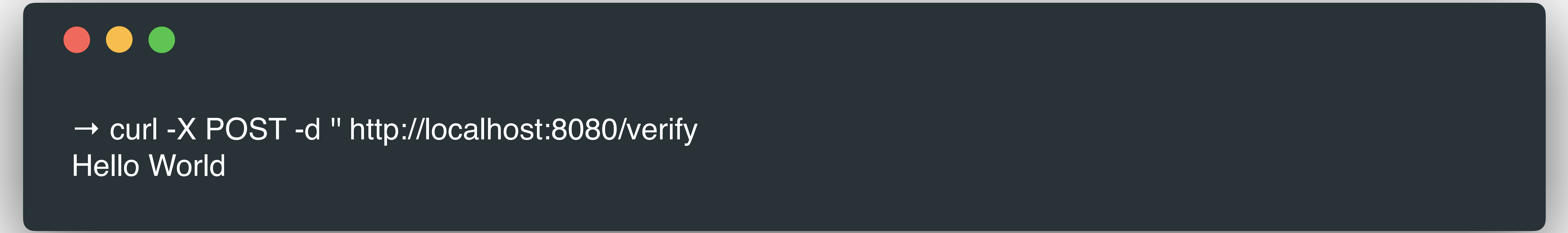
```
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```

```
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```

```kotlin
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```



```
→ curl -X POST -d '' http://localhost:8080/verify
```

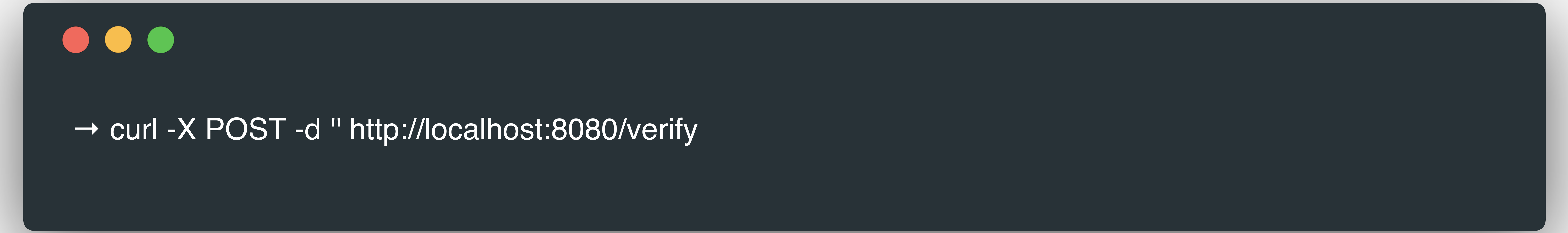


```
→ curl -X POST -d '' http://localhost:8080/verify
Hello World
```

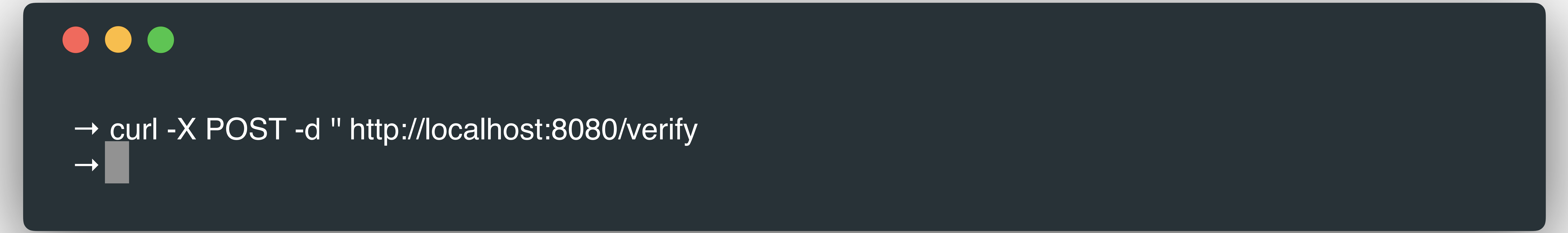
Typed Responses

```
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond("Hello World")
        }
    }
}
```

```kotlin
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

data class Response(val status: String)
```



```
→ curl -X POST -d '' http://localhost:8080/verify
```



```
→ curl -X POST -d '' http://localhost:8080/verify
→ 
```

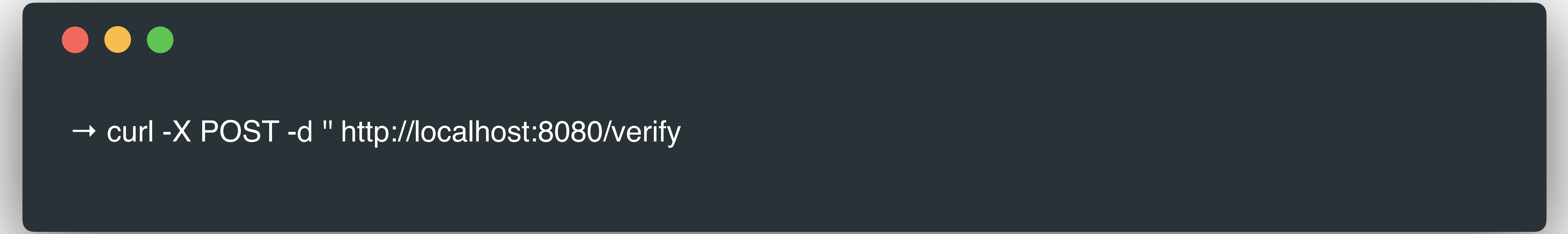
```kotlin
fun Application.verify() {
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages)
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

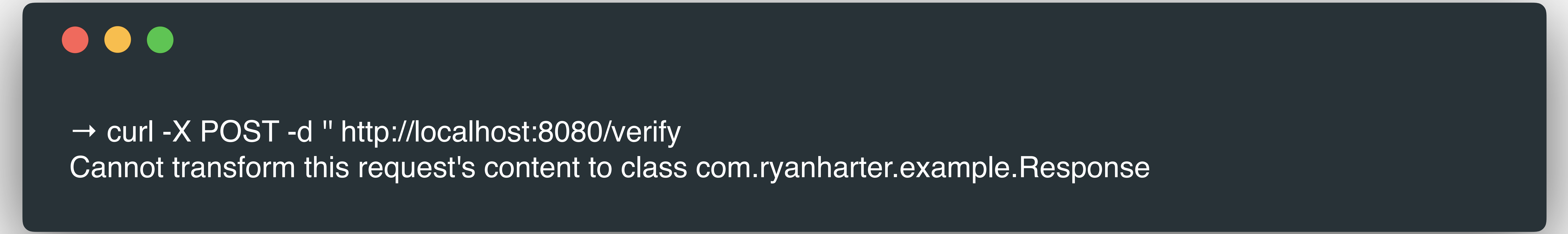
```
fun Application.verify() {
    install(StatusPages)         {
        exception<Throwable> { e ->


        }
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages)        {
        exception<Throwable> { e ->
         call.respondText(e.localizedMessage,
            ContentType.Text.Plain, HttpStatusCode.InternalServerError)
        }
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)            {
        exception<Throwable> { e ->
          call.respondText(e.localizedMessage,
              ContentType.Text.Plain, HttpStatusCode.InternalServerError)
        }
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```



```
→ curl -X POST -d '' http://localhost:8080/verify
```



```
→ curl -X POST -d '' http://localhost:8080/verify
Cannot transform this request's content to class com.ryanharter.example.Response
```

```
fun Application.verify() {
    install(StatusPages)           {
        exception<Throwable> { e ->
          call.respondText(e.localizedMessage,
              ContentType.Text.Plain, HttpStatusCode.InternalServerError)
        }
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)            { ...   }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

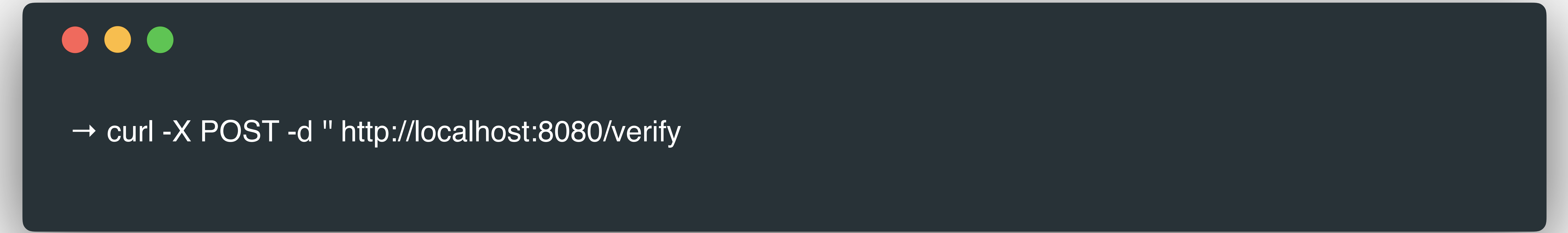
```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)          {

    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

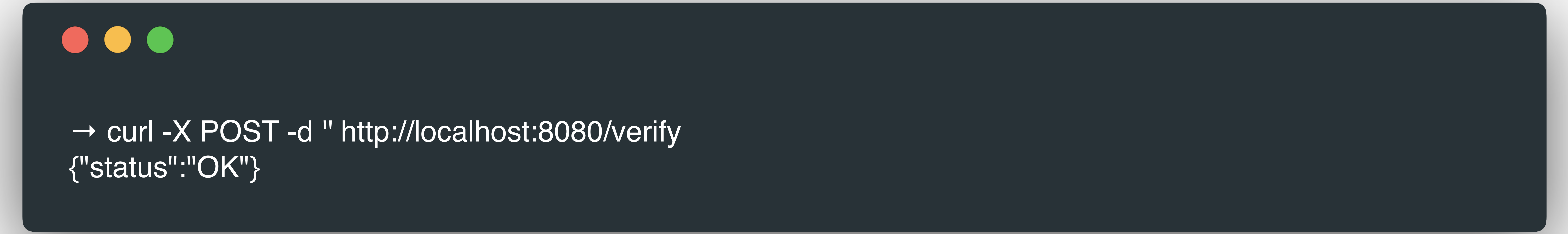
```kotlin
fun Application.verify() {
    install(StatusPages)              { ...    }
    install(ContentNegotiation)                  {
        moshi()
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages) { ... }
    install(ContentNegotiation) {
        moshi()
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages)            { ...   }
    install(ContentNegotiation)                {
        moshi()
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

@JsonClass(generateAdapter = true)
data class Response(val status: String)
```



```
→ curl -X POST -d '' http://localhost:8080/verify
```

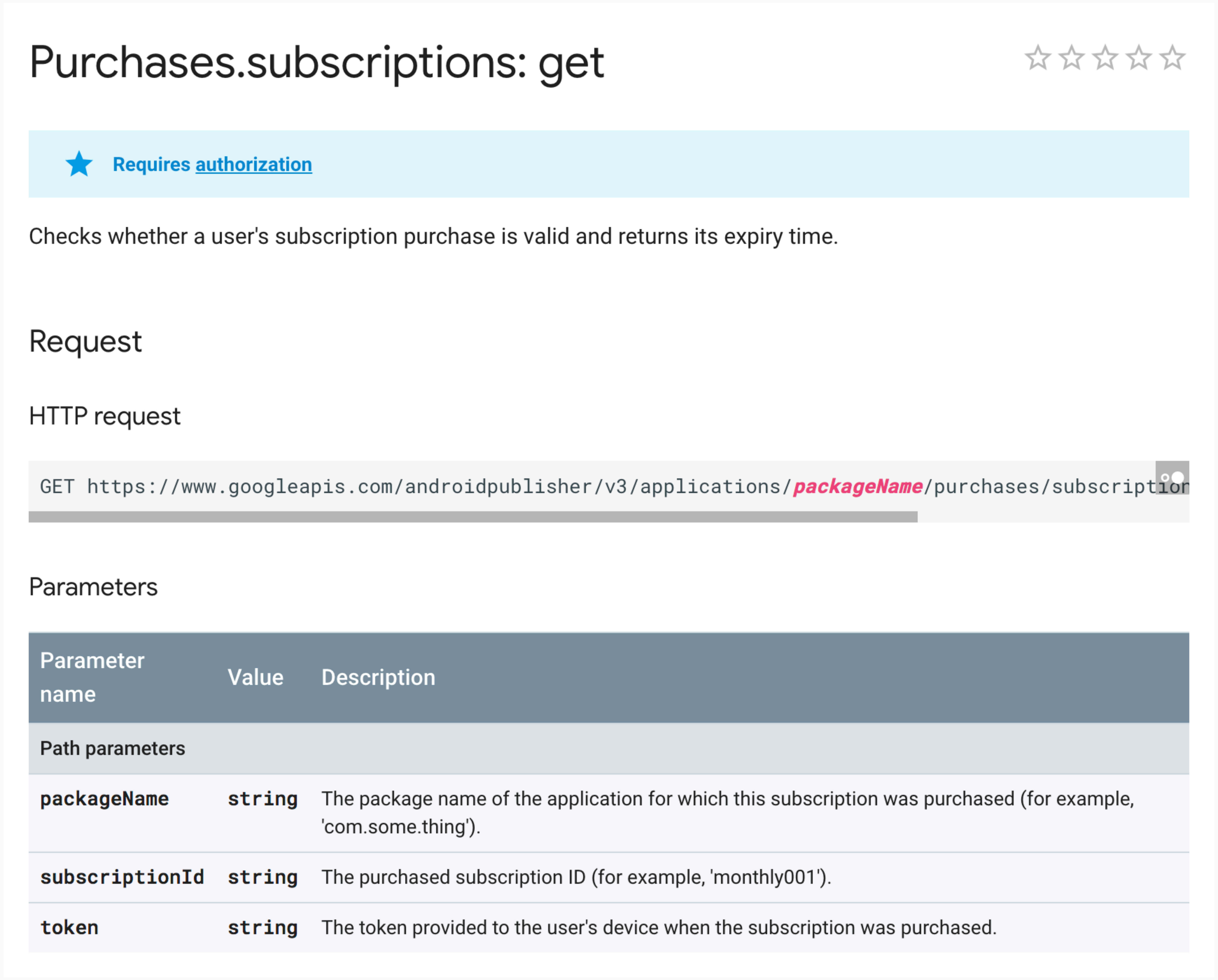


```
→ curl -X POST -d '' http://localhost:8080/verify
{"status":"OK"}
```

```
fun Application.verify() {
    install(StatusPages)            { ...   }
    install(ContentNegotiation)                {
        moshi()
    }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

Typed Requests



Purchases.subscriptions: get

Requires authorization

Checks whether a user's subscription purchase is valid and returns its expiry time.

Request

HTTP request

```
GET https://www.googleapis.com/androidpublisher/v3/applications/packageName/purchases/subscription
```

Parameters

Parameter name	Value	Description
Path parameters		
`packageName`	`string`	The package name of the application for which this subscription was purchased (for example, 'com.some.thing').
`subscriptionId`	`string`	The purchased subscription ID (for example, 'monthly001').
`token`	`string`	The token provided to the user's device when the subscription was purchased.

Request

HTTP request

```
GET https://www.googleapis.com/androidpublisher/v3/applications/packageName/purchases/subscr
```

Parameters

Parameter name	Value	Description
Path parameters		
`packageName`	`string`	The package name of the application for which this subscription was purchased (for examp 'com.some.thing').
`subscriptionId`	`string`	The purchased subscription ID (for example, 'monthly001').
`token`	`string`	The token provided to the user's device when the subscription was purchased.

```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}

@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages) { ... }
    install(ContentNegotiation) { ... }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)              { ...   }
    routing {
        post("/verify") {
            call.respond(Response(status = "OK"))
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

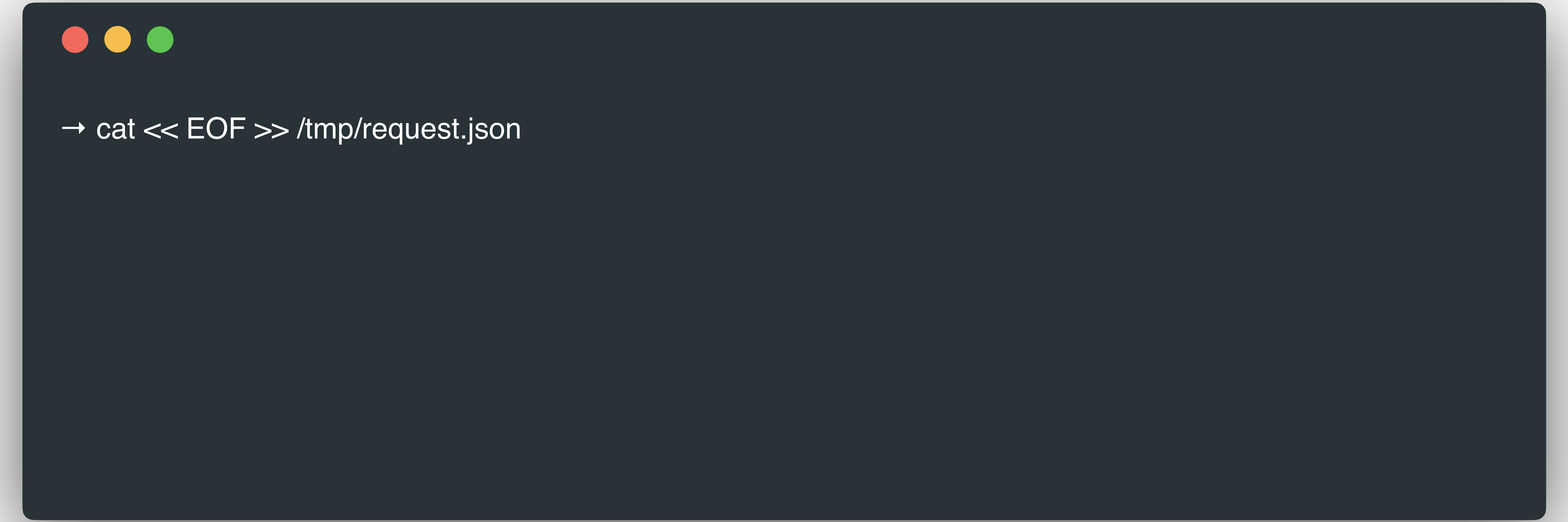
```kotlin
fun Application.verify() {
    install(StatusPages) { ... }
    install(ContentNegotiation) { ... }
    routing {
        post("/verify") {

        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)           { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

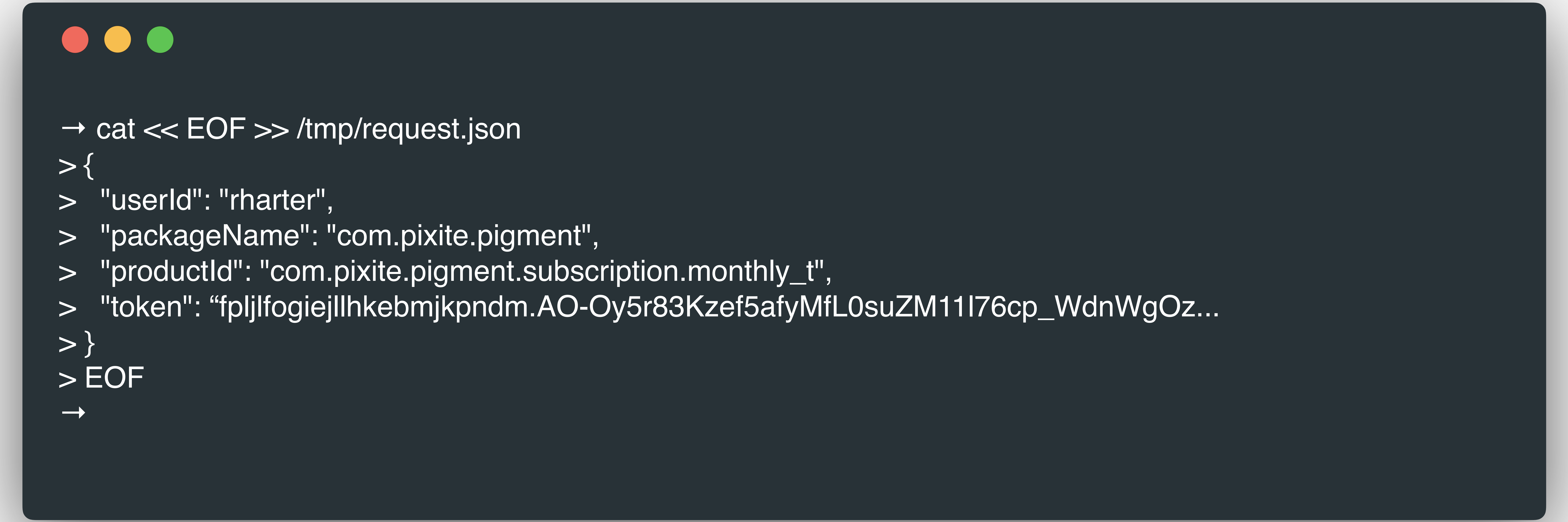
```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)            { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            call.respond(request)
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```kotlin
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            call.respond(request)
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```
→
```

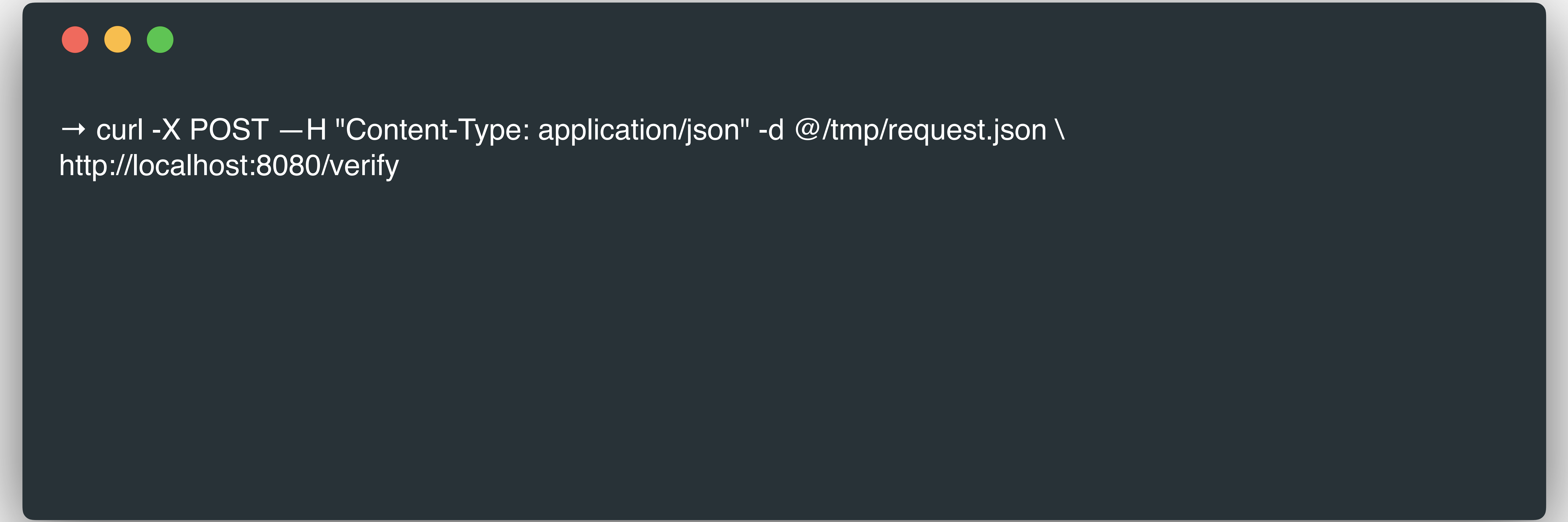


```
→ cat << EOF >> /tmp/request.json
```

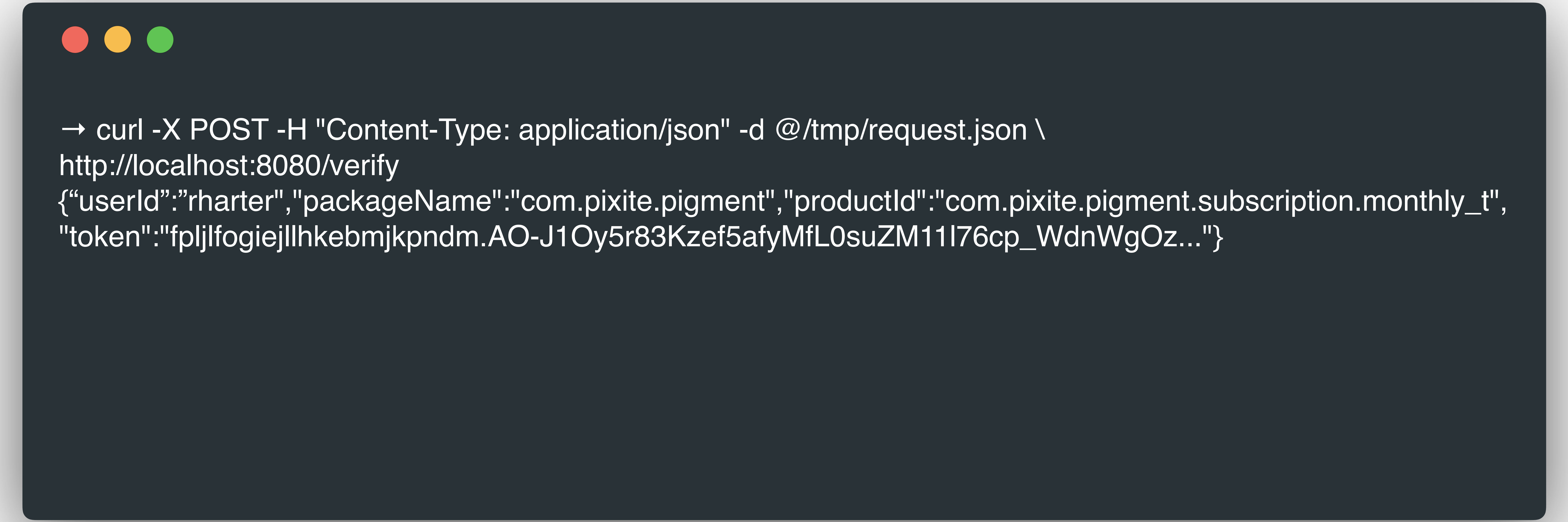


```
→ cat << EOF >> /tmp/request.json
> {
>   "userId": "rharter",
>   "packageName": "com.pixite.pigment",
>   "productId": "com.pixite.pigment.subscription.monthly_t",
>   "token": “fpljlfogiejllhkebmjkpndm.AO-Oy5r83Kzef5afyMfL0suZM11l76cp_WdnWgOz...
> }
> EOF
→
```

```
→
```



```
→ curl -X POST —H "Content-Type: application/json" -d @/tmp/request.json \
http://localhost:8080/verify
```



```
→ curl -X POST -H "Content-Type: application/json" -d @/tmp/request.json \
http://localhost:8080/verify
{“userId”:”rharter","packageName":"com.pixite.pigment","productId":"com.pixite.pigment.subscription.monthly_t",
"token":"fpljlfogiejllhkebmjkpndm.AO-J1Oy5r83Kzef5afyMfL0suZM11l76cp_WdnWgOz..."}
```

```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            call.respond(request)
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
data class Response(val status: String)
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            ???
            call.respond(request)
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
```

External Components

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)            { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            ???
            call.respond(request)
        }
    }
}

@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)

@JsonClass(generateAdapter = true)
```

```
fun Application.verify() {
    install(StatusPages)            { ...   }
    install(ContentNegotiation)                 { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            call.respond(request)
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
```

```
fun Application.verify() {
    install(StatusPages)            { ...   }
    install(ContentNegotiation)            { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            //   Save to database
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)


@JsonClass(generateAdapter = true)
```

```
fun Application.verify() {
    install(StatusPages)            { ...   }
    install(ContentNegotiation)                 { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            //   Save to database
            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)
```

```kotlin
interface Api {
  suspend fun findSubscription(userId: String,
                               packageName: String,
                               productId: String,
                               token: String): Subscription?
}
```

```
class PlayStore(serviceAccountFile: InputStream) : Store {

  private val publisher: AndroidPublisher by lazy {...}

  suspend fun findSubscription(userId: String,
                               packageName: String,
                               productId: String,
                               token: String): Subscription? = coroutineScope {
        val response = async {
          publisher.purchases()
              .subscriptions()
              .get(packageName, productId, token)
              .execute()
        }
        response.await().asSubscription(ownerId, token)
  }
}
```

```
class PlayStore(serviceAccountFile: InputStream) : Store {

  private val publisher: AndroidPublisher by lazy {...}

  suspend fun findSubscription(userId: String,
                               packageName: String,
                               productId: String,
                               token: String): Subscription? = coroutineScope {
        val response = async {
          publisher.purchases()
              .subscriptions()
              .get(packageName, productId, token)
              .execute()
        }
        response.await().asSubscription(ownerId, token)
  }
}
```

```
class PlayStore(serviceAccountFile: InputStream) : Store {

 private val publisher: AndroidPublisher by lazy {...}

 suspend fun findSubscription(userId: String,
                              packageName: String,
                              productId: String,
                              token: String): Subscription? = coroutineScope {
    val response = async {
      publisher.purchases()
        .subscriptions()
        .get(packageName, productId, token)
        .execute()
    }
    response.await().asSubscription(ownerId, token)
 }
}
```

```
interface Database {

  suspend fun subscription(subscriptionId: String): Subscription?

  suspend fun subscriptionByToken(token: String): Subscription?

  suspend fun subscriptionByUserId(userId: String): Subscription?

  suspend fun createSubscription(subscription: Subscription): Subscription

}
```

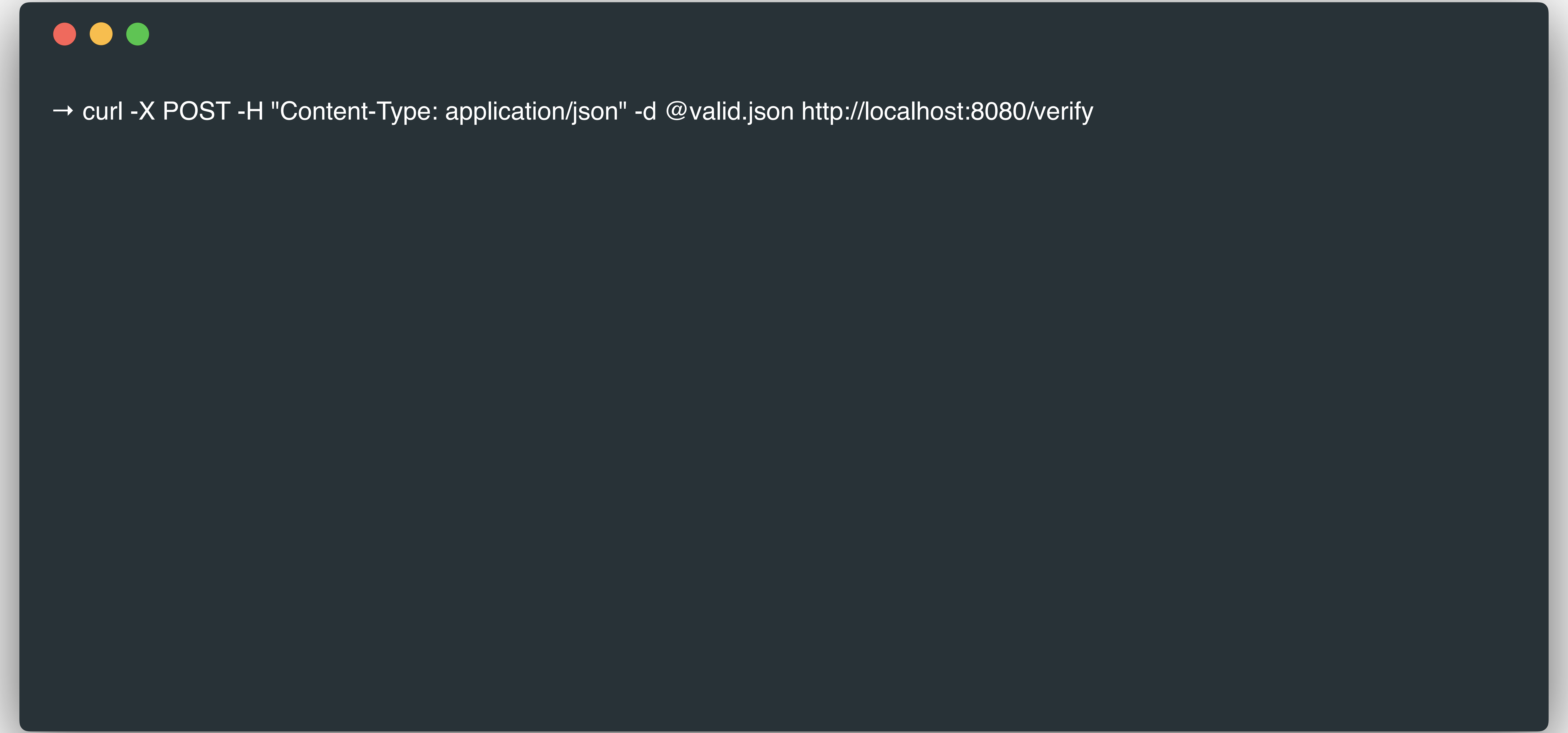
```kotlin
fun Application.verify() {
    install(StatusPages)             { ...   }
    install(ContentNegotiation)                { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            //   Save to database
            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String,
    val token: String
)
```

```
fun Application.verify() {

    val api = TotallyRealApi()

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            //   Save to database
            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
    val productId: String
```

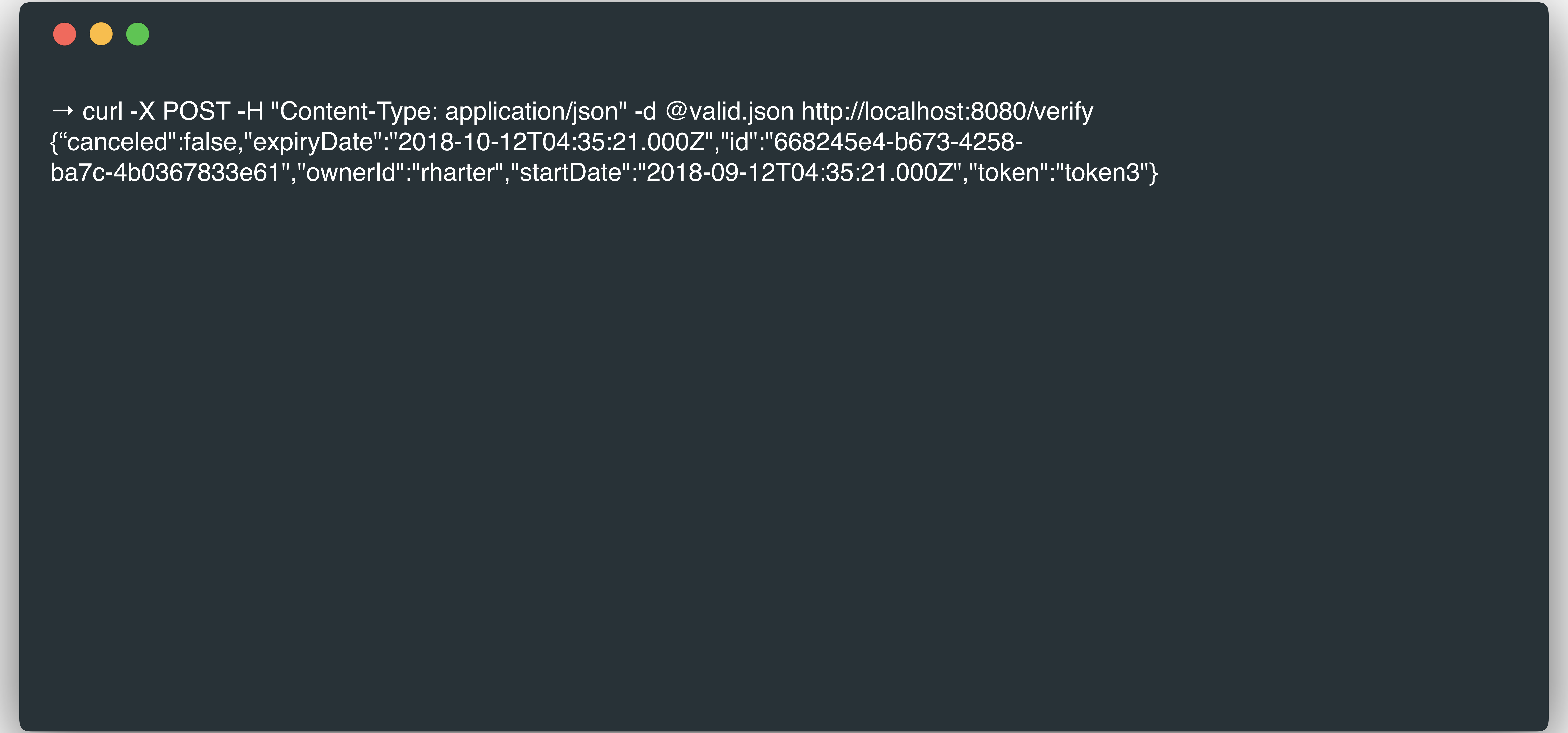
```
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            // Find valid subscription in db or remotely
            //   Save to database
            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
```

```kotlin
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages) { ... }
    install(ContentNegotiation) { ... }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            val subscription = db.subscriptionByUserId(request.userId)
            //  Save to database
            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String,
    val packageName: String,
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            val subscription = db.subscriptionByUserId(request.userId)
                ?: api.findSubscription(request.userId, request.packageName
                                     request.productId, request.token)

            // Return subscription or 404
        }
    }
}


@JsonClass(generateAdapter = true)
data class Request(
    val userId: String
```

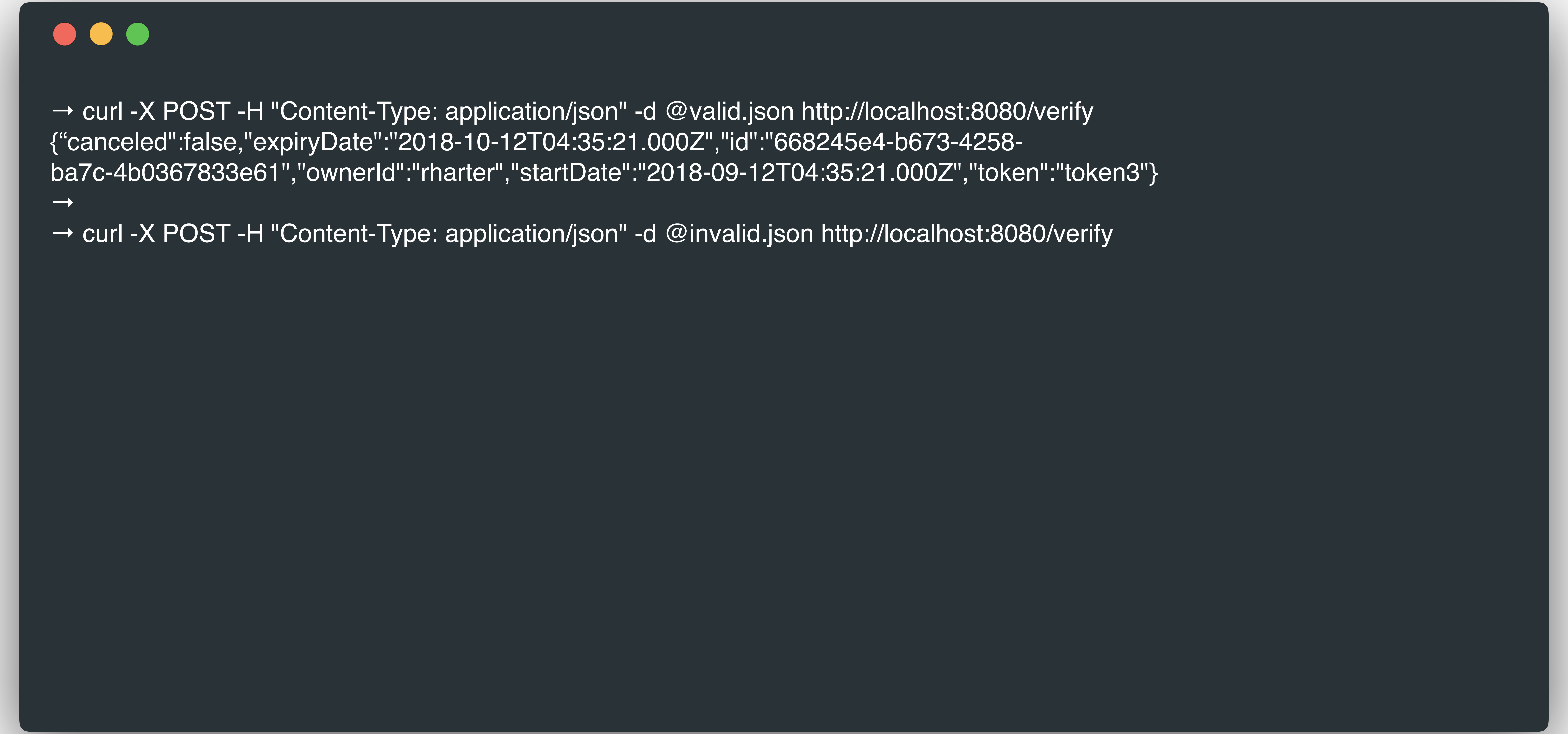
```kotlin
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages) { ... }
    install(ContentNegotiation) { ... }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            val subscription = db.subscriptionByUserId(request.userId)
                ?: api.findSubscription(request.userId, request.packageName
                                        request.productId, request.token)
                    ?.also { db.createSubscription(it) }
            // Return subscription or 404
        }
    }
}

@JsonClass(generateAdapter = true)
data class Request(
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            val subscription = db.subscriptionByUserId(request.userId)
                ?: api.findSubscription(request.userId, request.packageName
                                        request.productId, request.token)
                        ?.also { db.createSubscription(it) }
            if (subscription == null) {
              call.respond(HttpStatusCode.NotFound, "Subscription invalid.")
            } else {
              call.respond(subscription)
            }

        }
    }
```



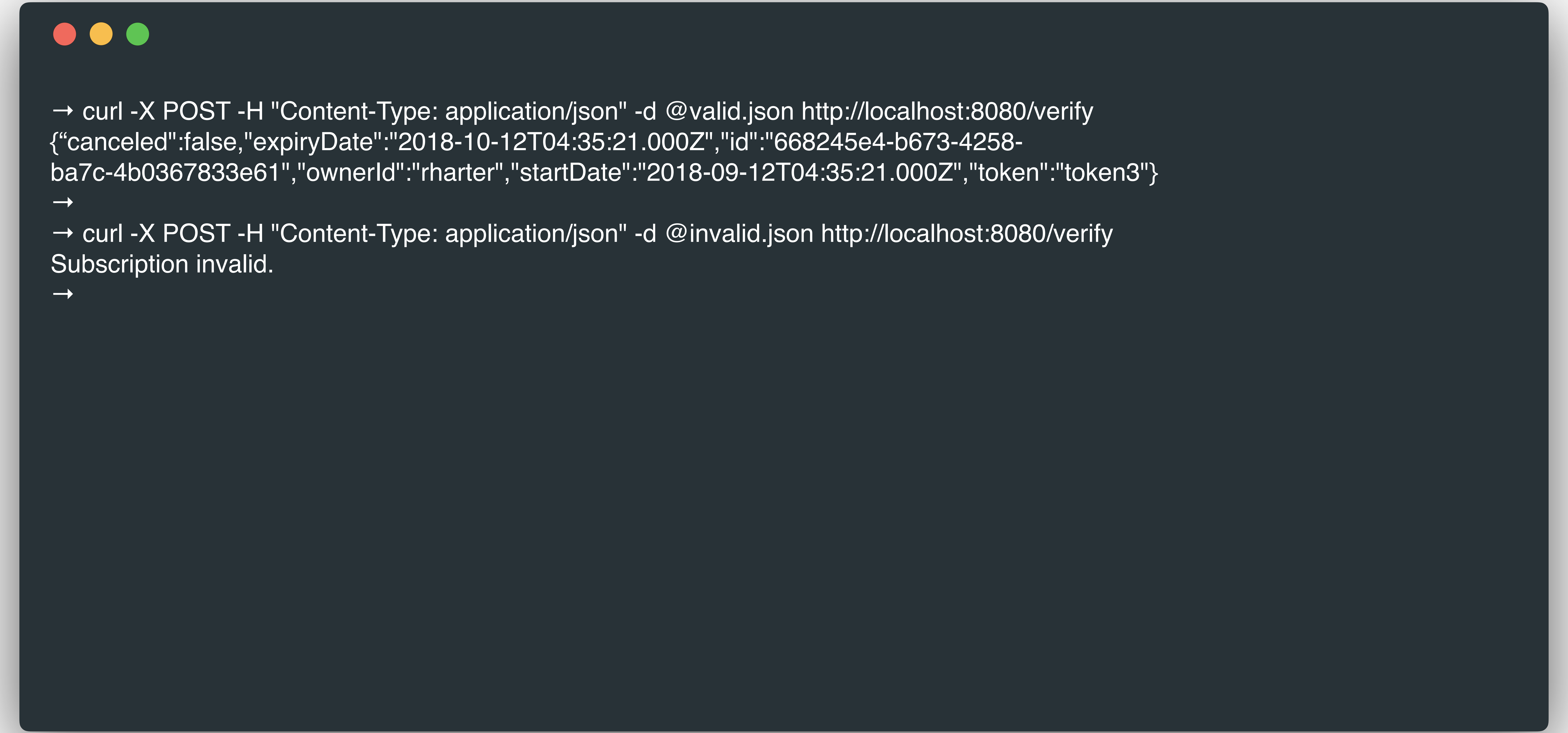
```
→ curl -X POST -H "Content-Type: application/json" -d @valid.json http://localhost:8080/verify
```



```
→ curl -X POST -H "Content-Type: application/json" -d @valid.json http://localhost:8080/verify
{“canceled":false,"expiryDate":"2018-10-12T04:35:21.000Z","id":"668245e4-b673-4258-
ba7c-4b0367833e61","ownerId":"rharter","startDate":"2018-09-12T04:35:21.000Z","token":"token3"}
```



```
→ curl -X POST -H "Content-Type: application/json" -d @valid.json http://localhost:8080/verify
{"canceled":false,"expiryDate":"2018-10-12T04:35:21.000Z","id":"668245e4-b673-4258-
ba7c-4b0367833e61","ownerId":"rharter","startDate":"2018-09-12T04:35:21.000Z","token":"token3"}
→
→ curl -X POST -H "Content-Type: application/json" -d @invalid.json http://localhost:8080/verify
```



```
→ curl -X POST -H "Content-Type: application/json" -d @valid.json http://localhost:8080/verify
{"canceled":false,"expiryDate":"2018-10-12T04:35:21.000Z","id":"668245e4-b673-4258-
ba7c-4b0367833e61","ownerId":"rharter","startDate":"2018-09-12T04:35:21.000Z","token":"token3"}
→
→ curl -X POST -H "Content-Type: application/json" -d @invalid.json http://localhost:8080/verify
Subscription invalid.
→
```

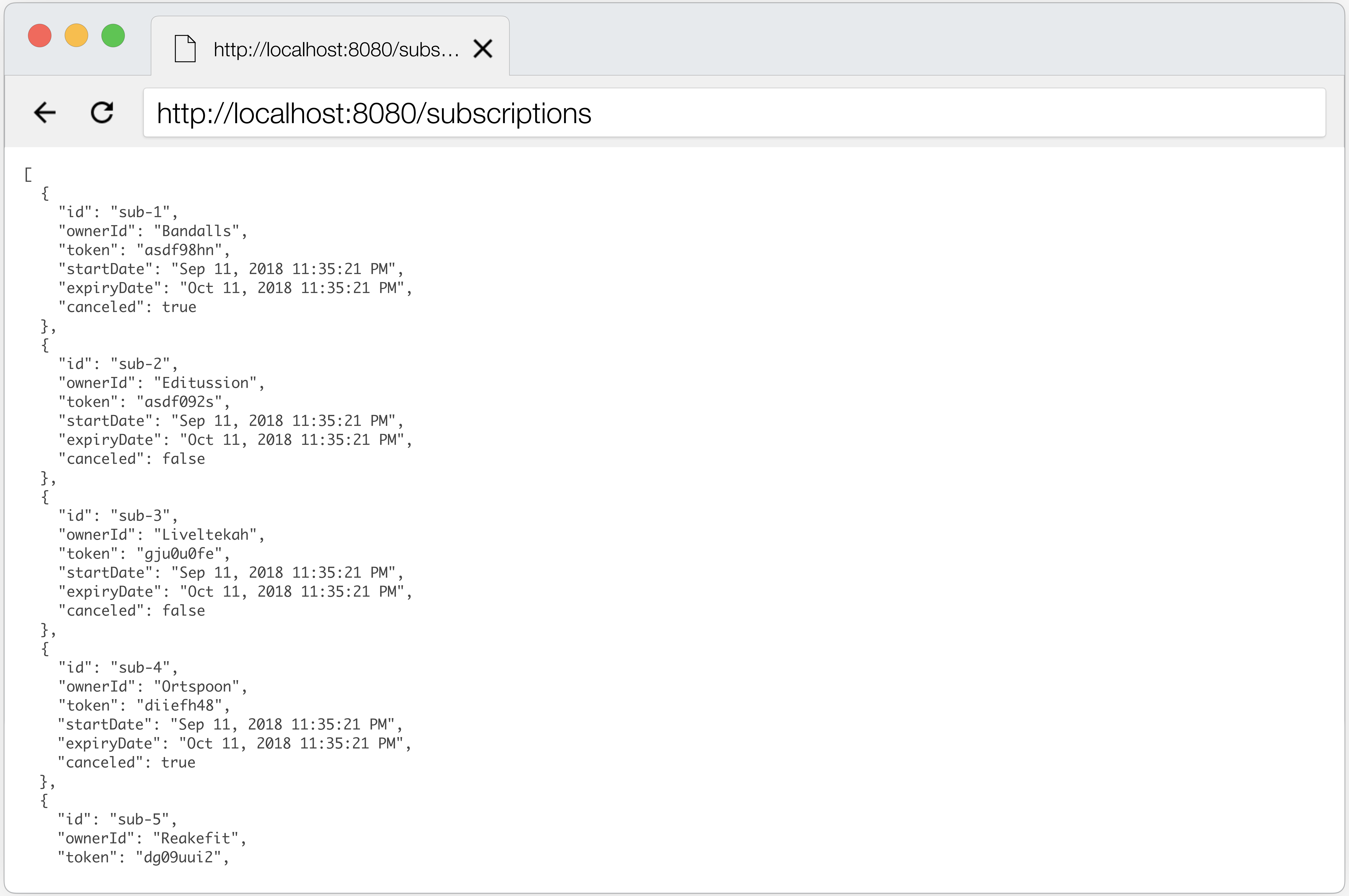
Templates

```
fun Application.verify() {

    val api = TotallyRealApi()
    val db = InMemoryDatabase()

    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    routing {
        post("/verify") {
            val request = call.receive<Request>()
            val subscription = db.subscriptionByUserId(request.userId)
                ?: api.findSubscription(request.userId, request.packageName
                                    request.productId, request.token)
                        ?.also { db.createSubscription(it) }
            if (subscription == null) {
              call.respond(HttpStatusCode.NotFound, "Subscription invalid.")
            } else {
              call.respond(subscription)
            }

        }
    }
```

```
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)                 { ...   }
    routing {
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        get("/subscriptions") {



        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)              { ...   }
    routing {
        get(“/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(subscriptions)
        }
        post("/verify") {          ...    }
    }
}
```


http://localhost:8080/subs…
http://localhost:8080/subscriptions

```
[
  {
    "id": "sub-1",
    "ownerId": "Bandalls",
    "token": "asdf98hn",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": true
  },
  {
    "id": "sub-2",
    "ownerId": "Editussion",
    "token": "asdf092s",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": false
  },
  {
    "id": "sub-3",
    "ownerId": "Liveltekah",
    "token": "gju0u0fe",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": false
  },
  {
    "id": "sub-4",
    "ownerId": "Ortspoon",
    "token": "diiefh48",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": true
  },
  {
    "id": "sub-5",
    "ownerId": "Reakefit",
    "token": "dg09uui2",
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    routing {
        get(“/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(subscriptions)
        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)            { ...   }
    install(ContentNegotiation)               { ...   }
    install(FreeMarker) {               }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(subscriptions)
        }
        post("/verify") {          ...    }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)            { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {
        templateLoader = ClassTemplateLoader(
            this@module.javaClass.classLoader,
            "templates"
        )
    }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(subscriptions)
        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)           { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {               ...   }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(subscriptions)
        }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)         { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(       subscriptions       )
        }
        post("/verify") {          ...   }
    }
}
```

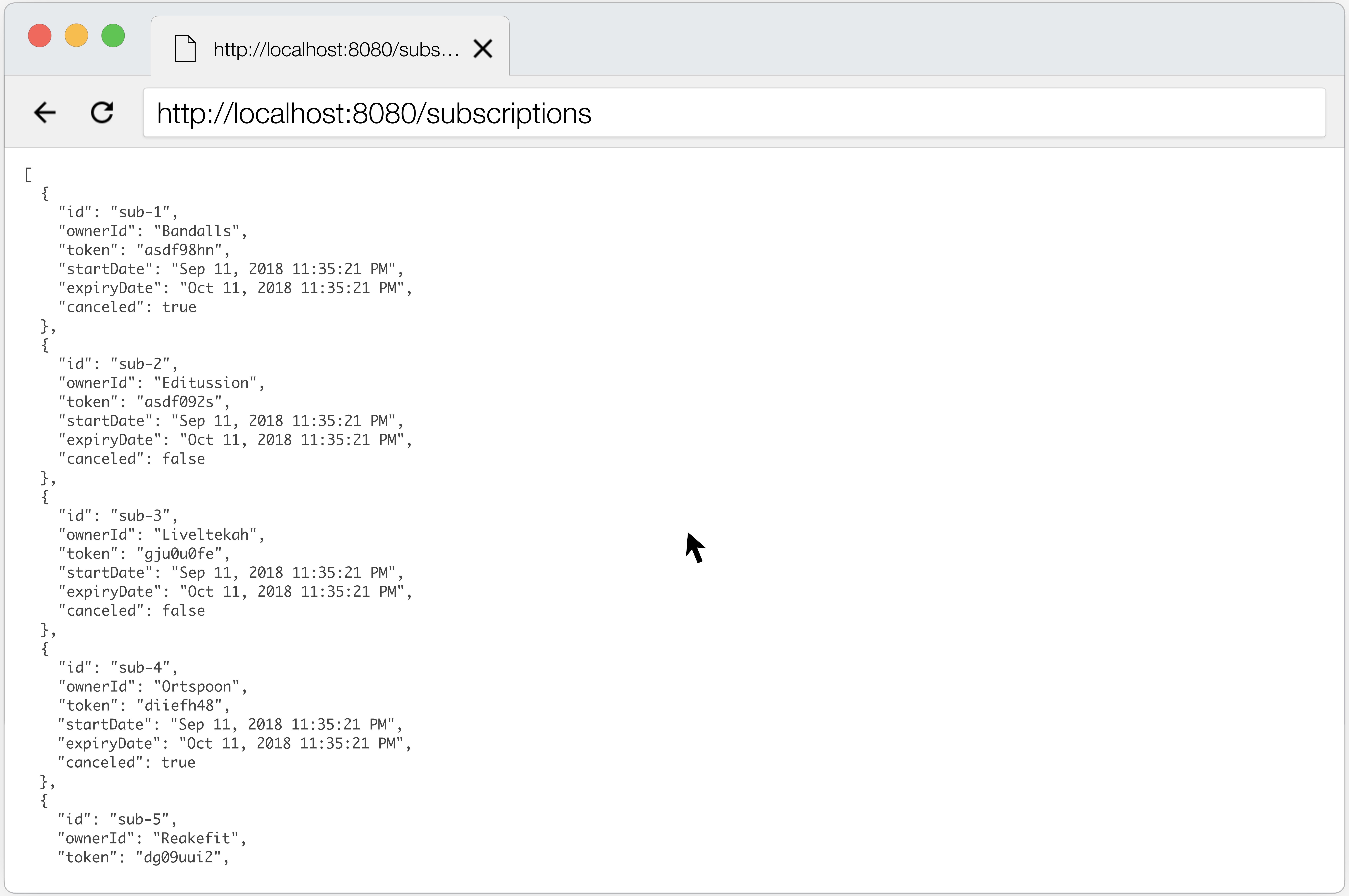
```
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(
                FreeMarkerContent("subscriptions.ftl",
                    mapOf("subscriptions" to              subscriptions       ))
            )
        }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {

  ...

  install(StatusPages)      { ...  }
  install(ContentNegotiation)      { ...  }
  install(FreeMarker) {      ...  }
  routing {
    get("/subscriptions") {
      val subscriptions = db.subscriptions()
      call.respond(
        FreeMarkerContent("subscriptions.ftl",
          mapOf("subscriptions" to      subscriptions    ))
      )
    }
    post("/verify") {      ...  }
  }
}
```

```
fun Application.verify() {

    ...

    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(
                FreeMarkerContent("subscriptions.ftl",
                    mapOf("subscriptions" to              subscriptions      ))
            )
        }
        post("/verify") {           ...   }
    }
}
```

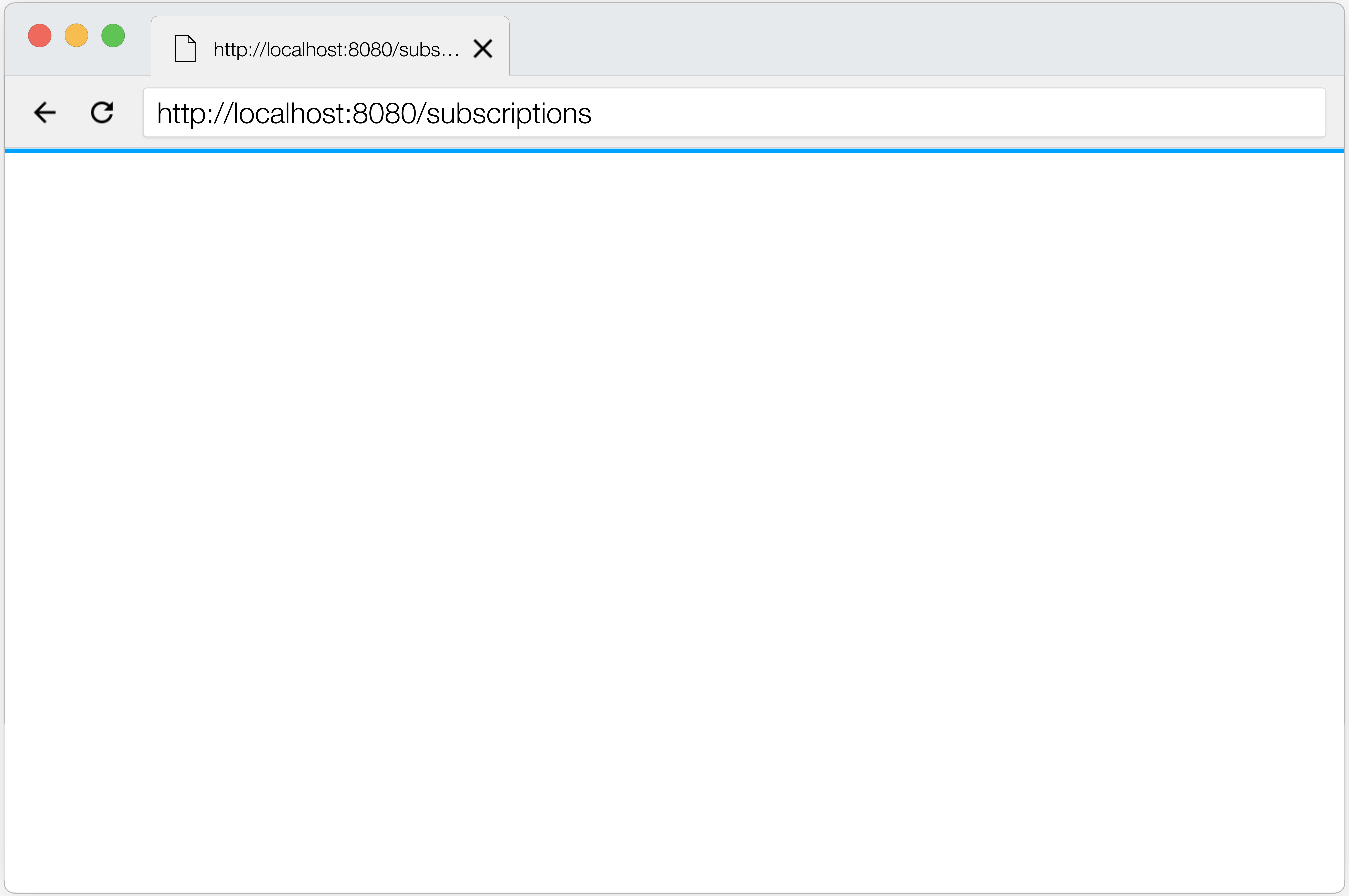
```html
<html>
<head>
  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
  <link rel="stylesheet" href="https://code.getmdl.io/1.3.0/material.indigo-pink.min.css">
  <script defer src="https://code.getmdl.io/1.3.0/material.min.js"></script>
</head>
<body>
  <div class="demo-layout mdl-layout mdl-js-layout mdl-layout--fixed-header">
    <header class="demo-header mdl-layout__header mdl-color--grey-100 mdl-color-text--grey-600">
      <div class="mdl-layout__header-row">
        <span class="mdl-layout-title">Home</span>
        <div class="mdl-layout-spacer"></div>
        <div class="mdl-textfield mdl-js-textfield mdl-textfield--expandable">
          <label class="mdl-button mdl-js-button mdl-button--icon" for="search">
            <i class="material-icons">search</i>
          </label>
          <div class="mdl-textfield__expandable-holder">
            <input class="mdl-textfield__input" type="text" id="search">
            <label class="mdl-textfield__label" for="search">Enter your query...</label>
          </div>
        </div>
        <button class="mdl-button mdl-js-button mdl-js-ripple-effect mdl-button--icon" id="hdrbtn">
          <i class="material-icons">more_vert</i>
        </button>
        <ul class="mdl-menu mdl-js-menu mdl-js-ripple-effect mdl-menu--bottom-right" for="hdrbtn">
          <li class="mdl-menu__item">About</li>
          <li class="mdl-menu__item">Contact</li>
          <li class="mdl-menu__item">Legal information</li>
        </ul>
      </div>
    </header>
    <main class="mdl-layout__content mdl-color--grey-100">
      <div class="mdl-grid">
        <table class="mdl-data-table mdl-js-data-table mdl-data-table--selectable mdl-color--white mdl-shadow--2dp mdl-cell mdl-cell--12-col">
          <thead>
          <tr>
            <th class="mdl-data-table__cell--non-numeric">Owner ID</th>
            <th class="mdl-data-table__cell--non-numeric">Start Date</th>
            <th class="mdl-data-table__cell--non-numeric">Expiry Date</th>
            <th class="mdl-data-table__cell--non-numeric">Cancelled</th>
          </tr>
          </thead>
          <tbody>
          <#list subscriptions as subscription>
            <tr>
              <td class="mdl-data-table__cell--non-numeric">${subscription.ownerId}</td>
              <td class="mdl-data-table__cell--non-numeric">${subscription.startDate?date}</td>
              <td class="mdl-data-table__cell--non-numeric">${subscription.expiryDate?date}</td>
              <td class="mdl-data-table__cell--non-numeric">
                ${subscription.canceled?string('yes', 'no')}
              </td>
            </tr>
          </#list>
          </tbody>
        </table>
      </div>
    </main>
  </div>
</body>
</html>
```

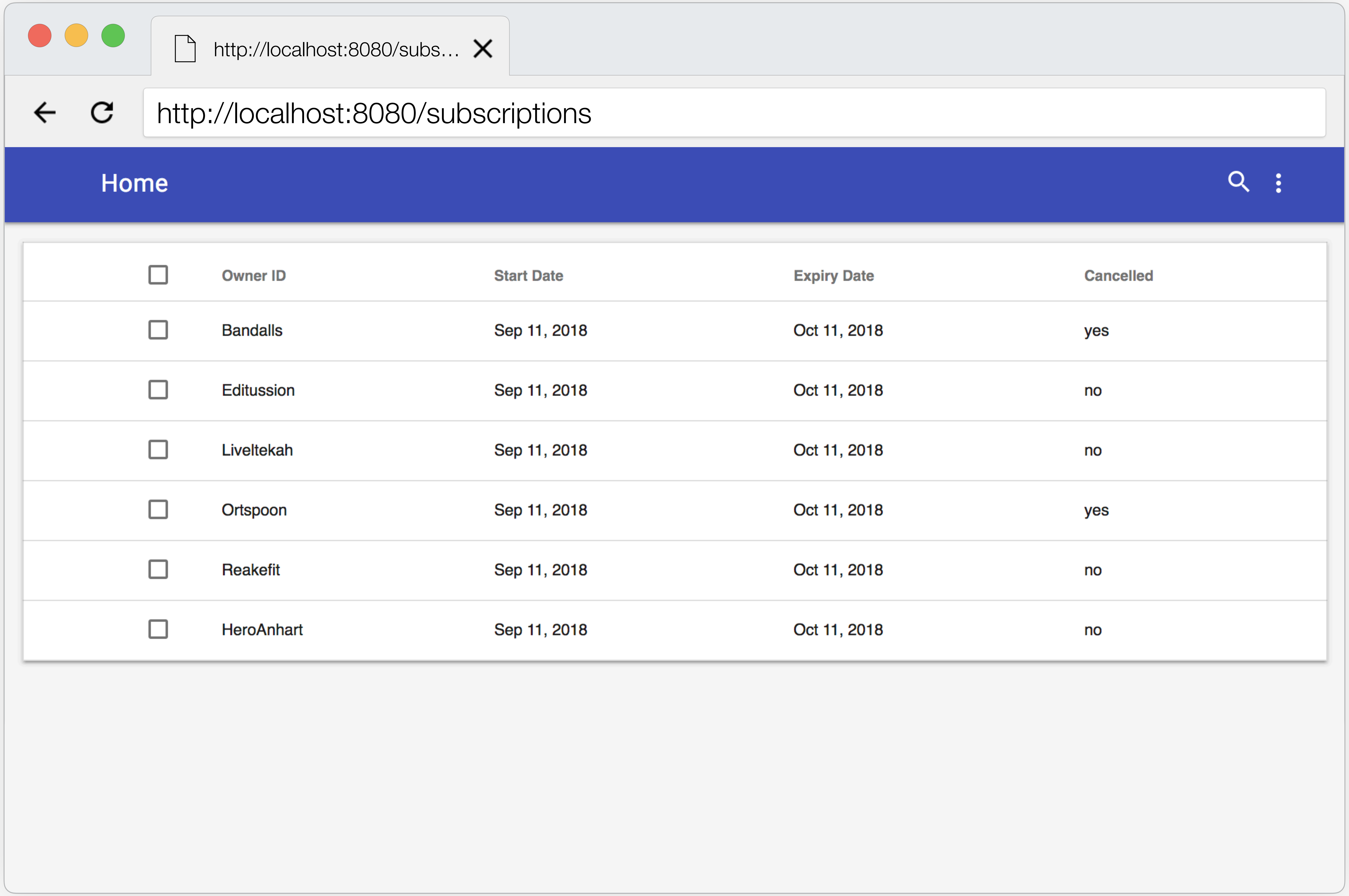
```
s="mdl-data-table__cell--non-numeric">Expiry Date</th>
s="mdl-data-table__cell--non-numeric">Cancelled</th>



criptions as subscription>

ass="mdl-data-table__cell--non-numeric">${subscription.ownerId}</td>
ass="mdl-data-table__cell--non-numeric">${subscription.startDate?date}</td>
ass="mdl-data-table__cell--non-numeric">${subscription.expiryDate?date}</td>
ass="mdl-data-table__cell--non-numeric">
ubscription.canceled?string('yes', 'no')}
```


http://localhost:8080/subs…

http://localhost:8080/subscriptions

```
[
  {
    "id": "sub-1",
    "ownerId": "Bandalls",
    "token": "asdf98hn",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": true
  },
  {
    "id": "sub-2",
    "ownerId": "Editussion",
    "token": "asdf092s",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": false
  },
  {
    "id": "sub-3",
    "ownerId": "Liveltekah",
    "token": "gju0u0fe",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": false
  },
  {
    "id": "sub-4",
    "ownerId": "Ortspoon",
    "token": "diiefh48",
    "startDate": "Sep 11, 2018 11:35:21 PM",
    "expiryDate": "Oct 11, 2018 11:35:21 PM",
    "canceled": true
  },
  {
    "id": "sub-5",
    "ownerId": "Reakefit",
    "token": "dg09uui2",
```


http://localhost:8080/subs…
http://localhost:8080/subscriptions


http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home

	Owner ID	Start Date	Expiry Date	Cancelled
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no

Authentication

```
fun Application.verify() {

    ...

    install(StatusPages)          { ...   }
    install(ContentNegotiation)            { ...   }
    install(FreeMarker) {            ...   }
    routing {
        get("/subscriptions") {
            val subscriptions = db.subscriptions()
            call.respond(
                FreeMarkerContent("subscriptions.ftl",
                    mapOf("subscriptions" to            subscriptions      ))
            )
        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {              ...   }
    routing {
        get("/subscriptions") {            ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)               { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)
    routing {
        get("/subscriptions") {            ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)              {
        basic(name = "userAuth")        {




        }
    }
    routing {
        get("/subscriptions") {           ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)                {
        basic(name = "userAuth")          {
            realm = "Verifier"




        }
    }
    routing {
        get("/subscriptions") {            ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {             ...   }
    install(Authentication)                  {
        basic(name = "userAuth")          {
            realm = "Verifier"
            validate { credentials ->

            }
        }
    }
    routing {
        get("/subscriptions") {           ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)         { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {             ...   }
    install(Authentication)                 {
        basic(name = "userAuth")        {
            realm = "Verifier"
            validate { credentials ->
                authService.authenticate(credentials.name, credentials.password)
            }
        }
    }
    routing {
        get("/subscriptions") {           ...   }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)                   { ...   }
    routing {
        get("/subscriptions") {            ...   }
        post("/verify") {           ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)         { ...   }
    install(ContentNegotiation)          { ...   }
    install(FreeMarker) {             ...   }
    install(Authentication)              { ...   }
    routing {
        get("/subscriptions") {           ...   }
        post("/verify") {          ...   }
    }
}
```

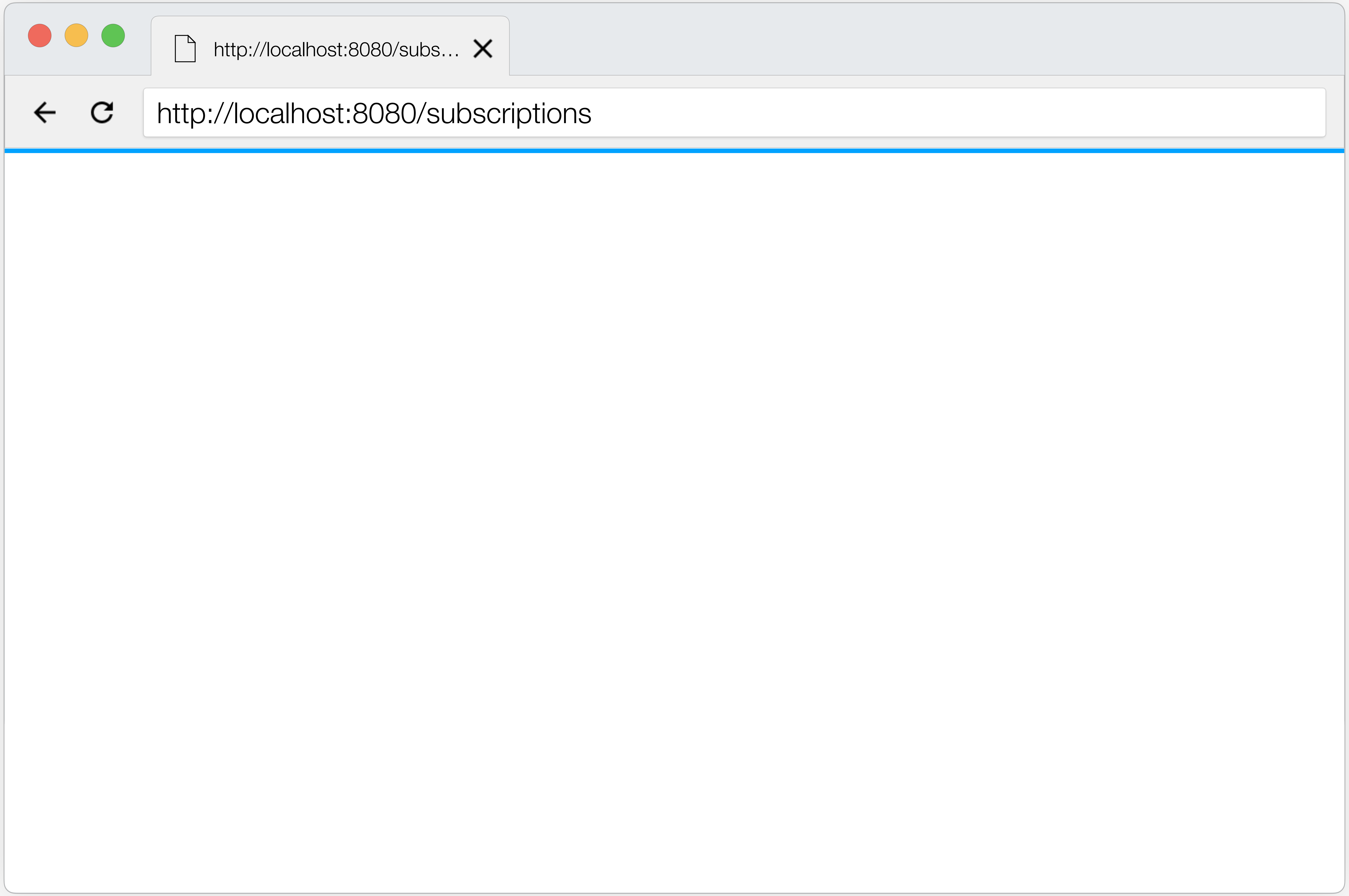
```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)                { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)                { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {            ...   }
        }
        post("/verify") {            ...   }
    }
}
```

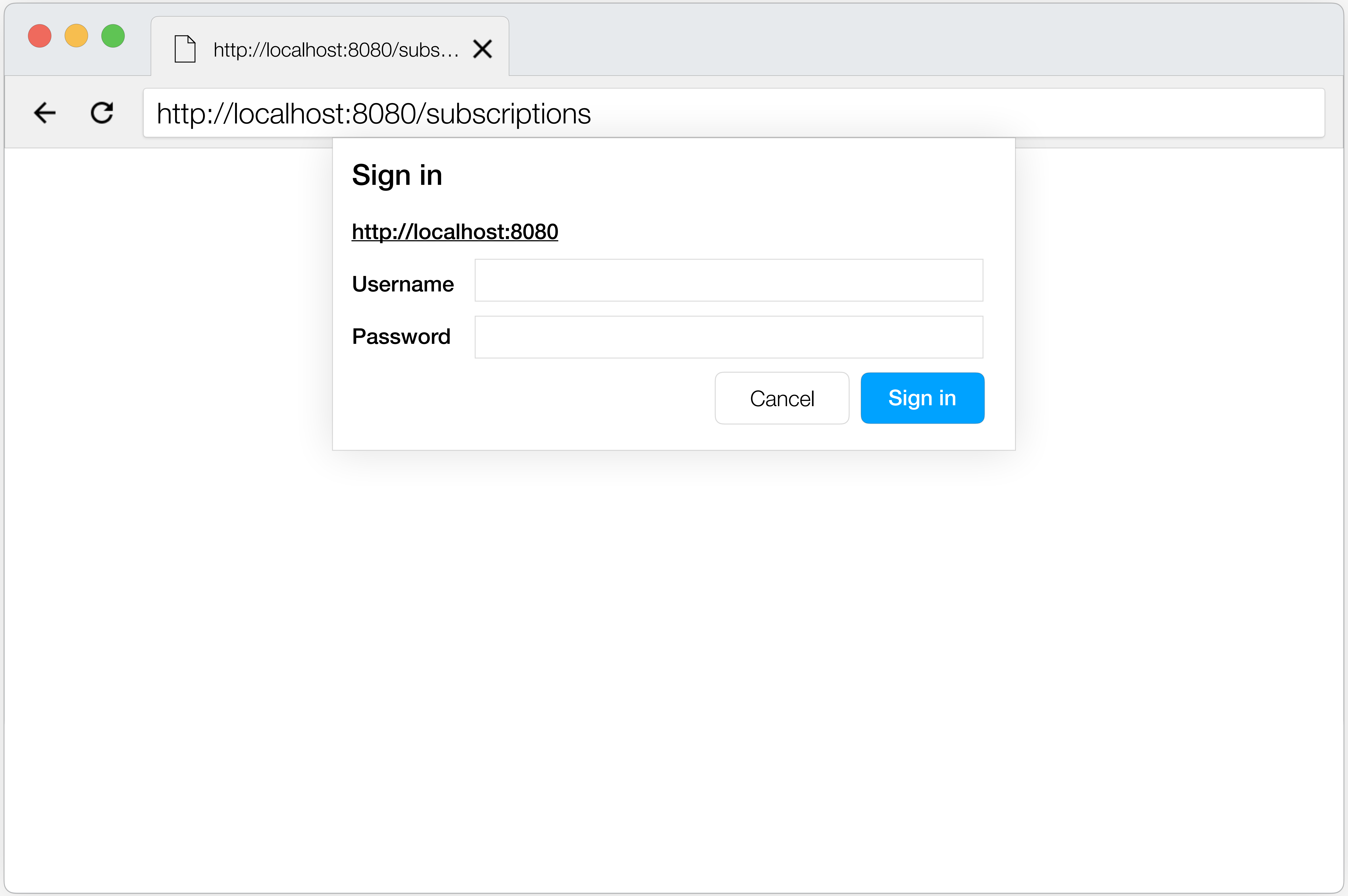
```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
   install(Authentication)                { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {          ...   }
        }
        post("/verify") {          ...   }
    }
}
```

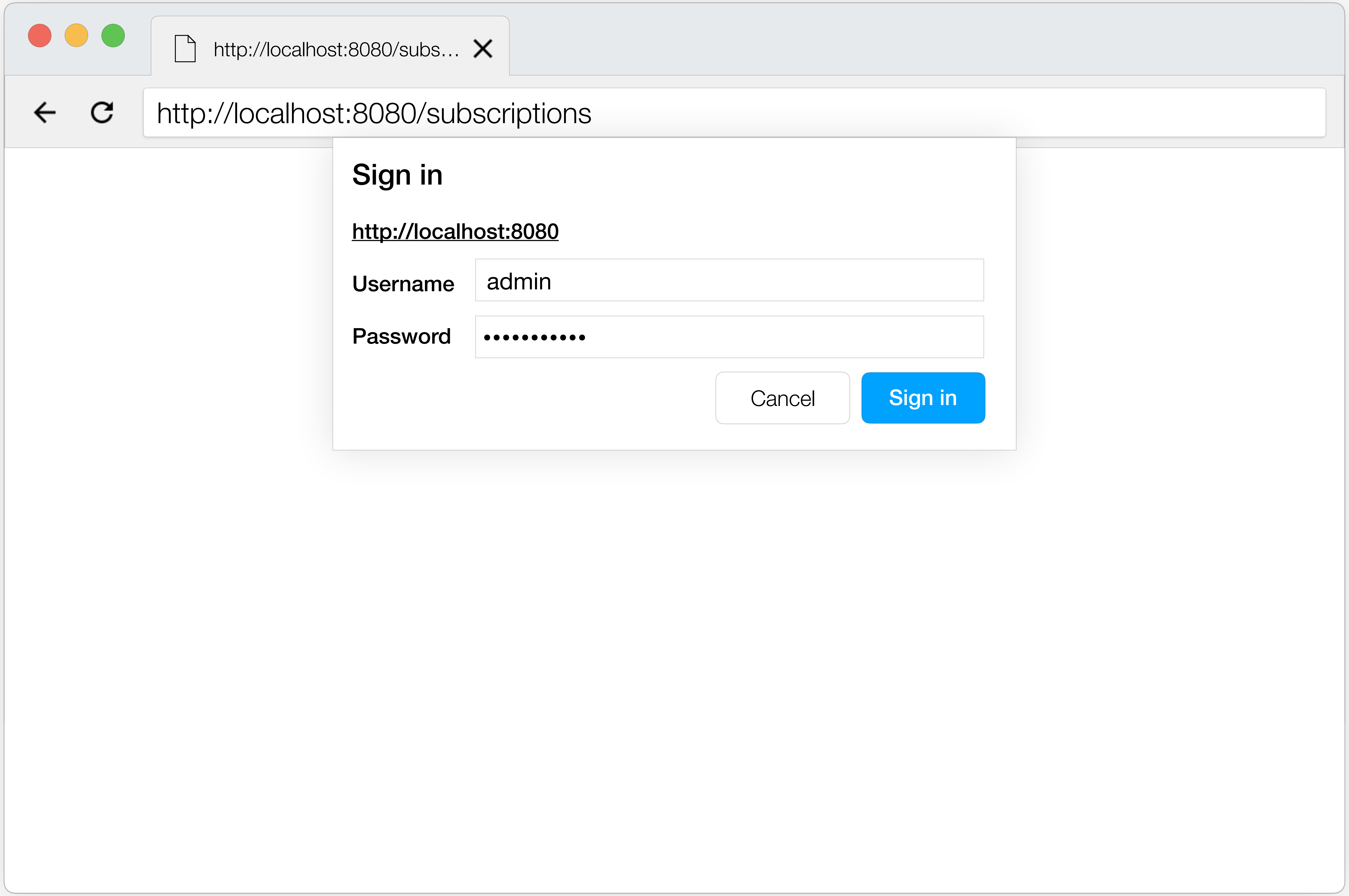
```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)             { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)                 { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {
                val subscriptions = db.subscriptions()
                call.respond(
                    FreeMarkerContent("subscriptions.ftl",
                        mapOf("subscriptions" to              subscriptions       ) )
                )
            }

        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {                ...   }
    install(Authentication)                    { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {
                val user = call.authentication.principal
                val subscriptions = db.subscriptions()
                call.respond(
                    FreeMarkerContent("subscriptions.ftl",
                        mapOf(     "subscriptions" to          subscriptions       ) )
                )
            }
        }
        post("/verify") {          ...   }
    }
}
```

```
fun Application.verify() {
    install(StatusPages)          { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {              ...   }
    install(Authentication)                { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {
                val user = call.authentication.principal
                val subscriptions = db.subscriptions()
                call.respond(
                    FreeMarkerContent("subscriptions.ftl",
                        mapOf(
                            "subscriptions" to        subscriptions      ,
                            "user" to user
                        )
                    )
                )
            }
        }
        post("/verify") {                 }
```


http://localhost:8080/subs…
http://localhost:8080/subscriptions


http://localhost:8080/subs…
http://localhost:8080/subscriptions
Sign in
http://localhost:8080
Username
Password
Cancel
Sign in

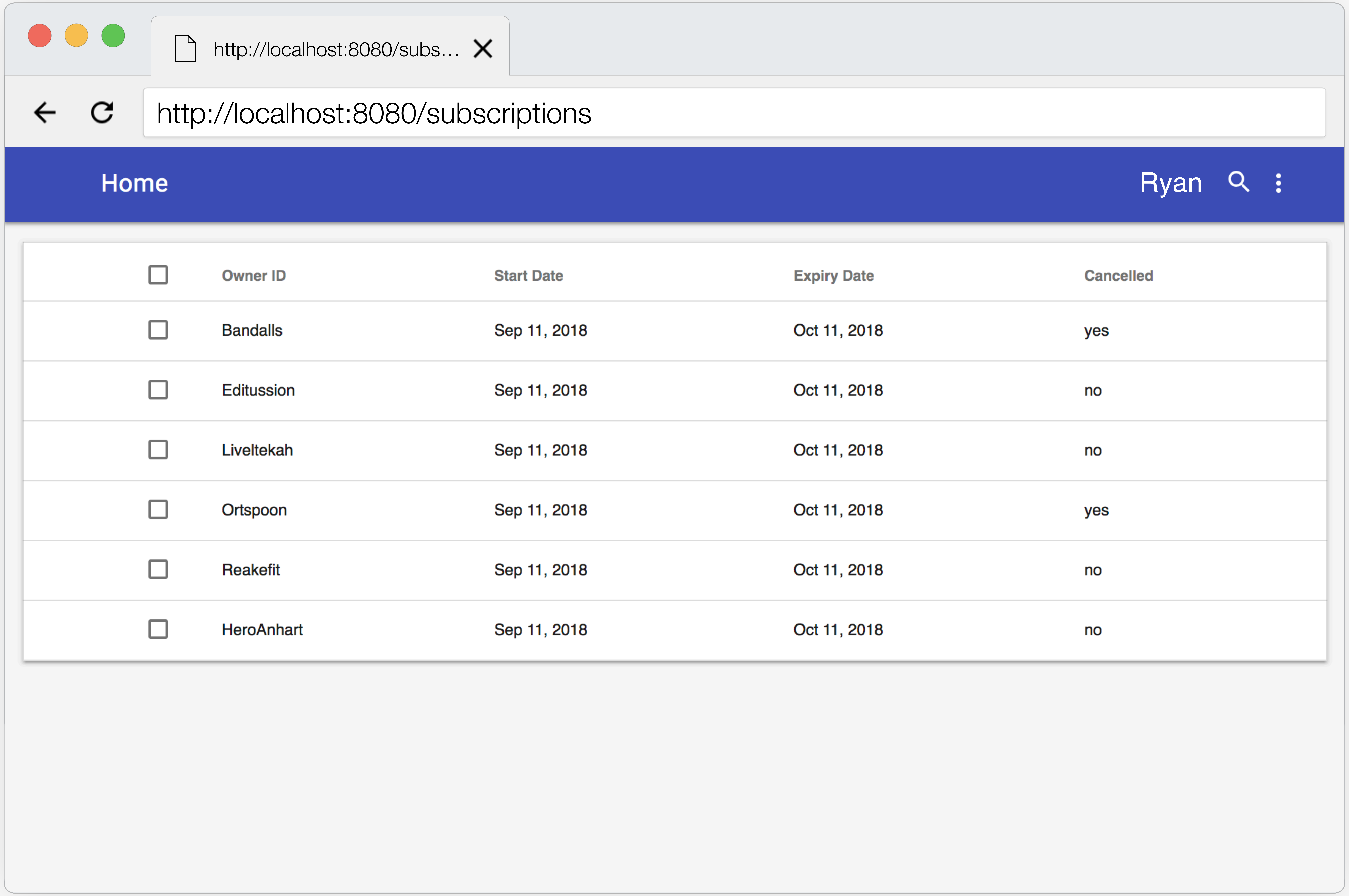

http://localhost:8080/subscriptions

Sign in

http://localhost:8080

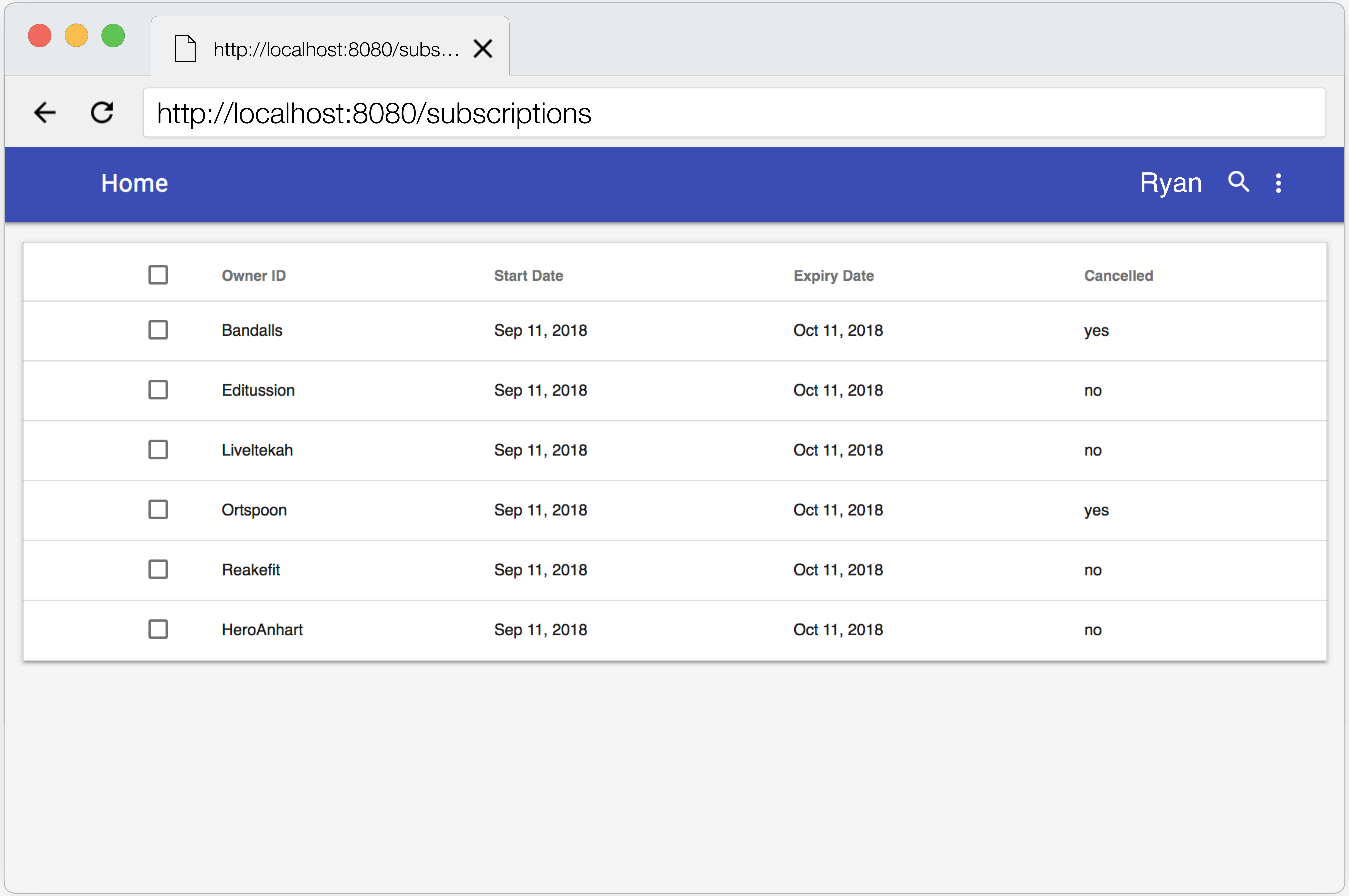
Username admin

Password ••••••••••••

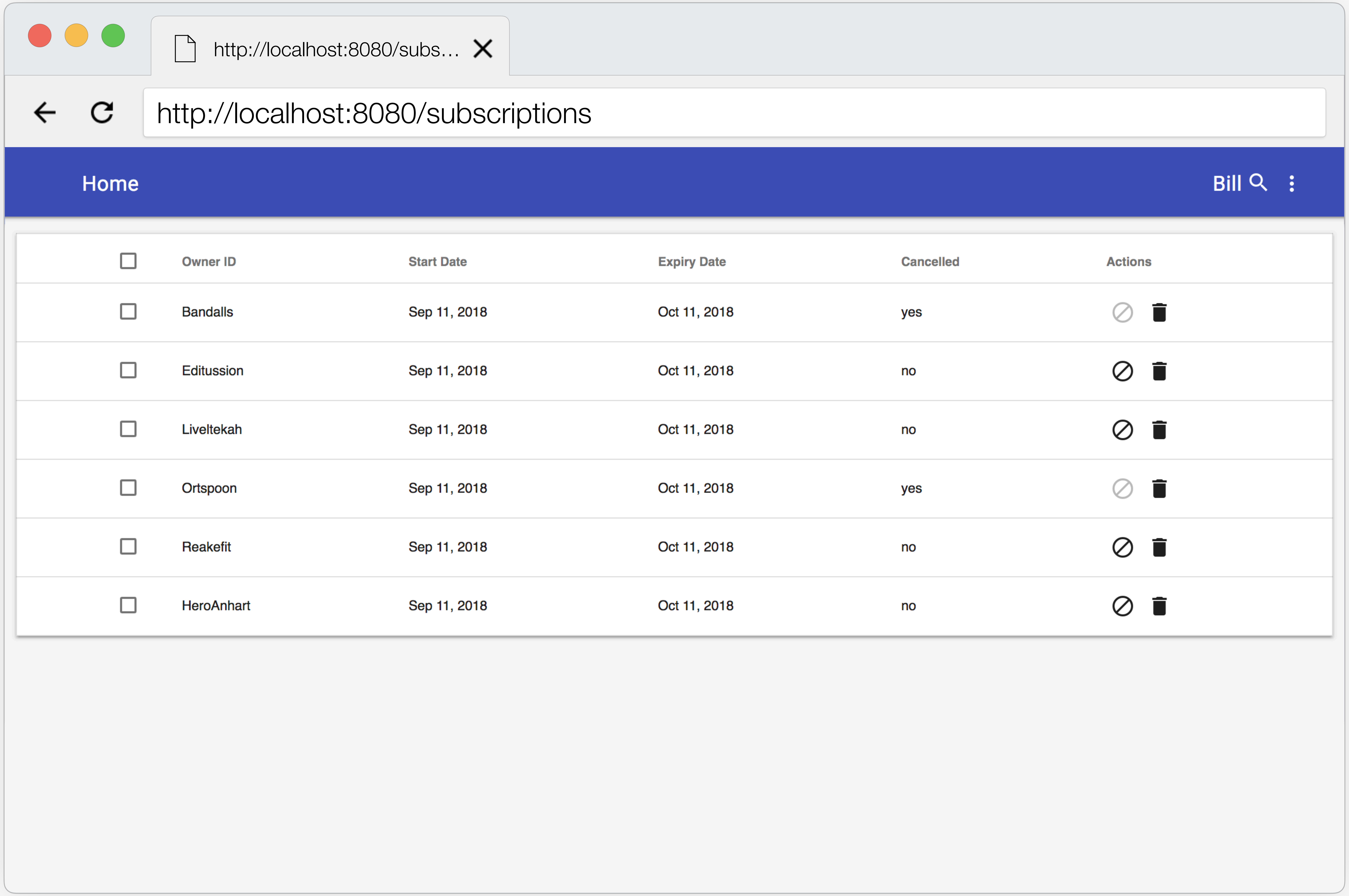

http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home
Ryan

☐	Owner ID	Start Date	Expiry Date	Cancelled
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no

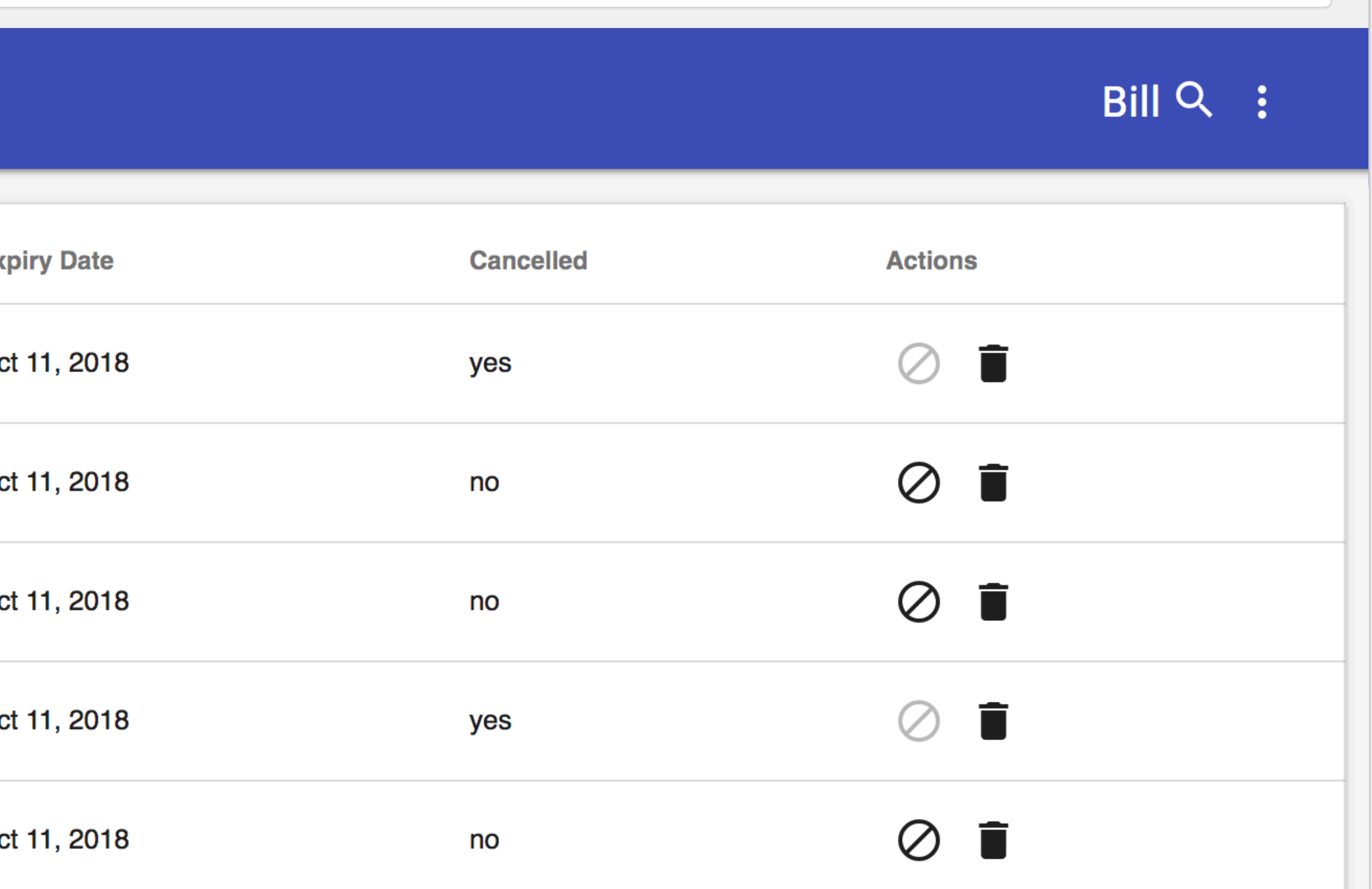
Forms


http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home
Ryan

	Owner ID	Start Date	Expiry Date	Cancelled
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no


http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home
Bill

	Owner ID	Start Date	Expiry Date	Cancelled	Actions
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑



xpiry Date	Cancelled	Actions
ct 11, 2018	yes	⊘ 🗑
ct 11, 2018	no	⊘ 🗑
ct 11, 2018	no	⊘ 🗑
ct 11, 2018	yes	⊘ 🗑
ct 11, 2018	no	⊘ 🗑

```
fun Application.verify() {
    install(StatusPages)           { ...   }
    install(ContentNegotiation)              { ...   }
    install(FreeMarker) {               ...   }
   install(Authentication)                    { ...   }
    routing {
        authenticate("userAuth") {
            get("/subscriptions") {            ...   }
        }
        post("/verify") {           ...   }
    }
}
```

```
install(FreeMarker) {          ...    }
install(Authentication)          { ...    }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...    }
        post("/subscriptions") {













        }
    }
    post("/verify") {          ...    }
}
```

```
install(FreeMarker) {          ...    }
install(Authentication)             { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...    }
        post("/subscriptions") {
            val parameters = call.receiveParameters()














        }
    }
    post("/verify") {          ...    }
```

```
install(FreeMarker) {          ...    }
install(Authentication)              { ...    }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {            ...    }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]













        }
    }
    post("/verify") {            ...    }
```

```
install(FreeMarker) {          ...   }
install(Authentication)              { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {            ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")











        }
    }
    post("/verify") {          ...   }
```

```
install(FreeMarker) {           ...    }
install(Authentication)             { ...    }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {             ...    }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters["id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters["action"]
                ?: throw IllegalArgumentException("Missing parameter: action")









        }
    }
    post("/verify") {           ...    }
```

```
install(FreeMarker) {        ...   }
install(Authentication)           { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters[“action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {







            }

        }
    }
    post("/verify") {          ...   }
}
```

```
install(FreeMarker) {         ...   }
install(Authentication)          { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters[“action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {
                "delete" ->      db.deleteSubscription(id)





            }

        }
    }
    post("/verify") {         ...   }
}
```

```
install(FreeMarker) {        ...   }
install(Authentication)          { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters["id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters["action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {
                "delete" ->       db.deleteSubscription(id)
                "cancel" -> {




                }
            }

        }
    }
    post("/verify") {          ...   }
```

```
install(FreeMarker) {          ...   }
install(Authentication)              { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters[“action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {
                "delete" ->       db.deleteSubscription(id)
                "cancel" -> {
                    db.subscription(id)



                }
            }


        }
    }
    post("/verify") {          ...   }
```

```
install(FreeMarker) {           ...    }
install(Authentication)                 { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {            ...    }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters[“action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {
                "delete" ->       db.deleteSubscription(id)
                "cancel" -> {
                    db.subscription(id)          ?.also {
                        db.putSubscription(it.copy(canceled = true))
                    }
                }
            }

        }
    }
    post("/verify") {          ...    }
```

```
install(FreeMarker) {        ...   }
install(Authentication)          { ...   }
routing {
    authenticate("userAuth") {
        get("/subscriptions") {          ...   }
        post("/subscriptions") {
            val parameters = call.receiveParameters()
            val id = parameters[“id"]
                ?: throw IllegalArgumentException("Missing parameter: id")
            val action = parameters[“action"]
                ?: throw IllegalArgumentException("Missing parameter: action")
            when (action) {
                "delete" -> db.deleteSubscription(id)
                "cancel" -> {
                    db.subscription(id)?.also {
                        db.putSubscription(it.copy(canceled = true))
                    }
                }
            }
            call.respondRedirect("/subscriptions")
        }
    }
    post("/verify") {          ...   }
}
```

```
criptions as subscription>

ss="mdl-data-table__cell--non-numeric">${subscription.ownerId}</td>
ss="mdl-data-table__cell--non-numeric">${subscription.startDate?date}</td>
ss="mdl-data-table__cell--non-numeric">${subscription.expiryDate?date}</td>
ss="mdl-data-table__cell--non-numeric">
bscription.canceled?string('yes', 'no')}
```

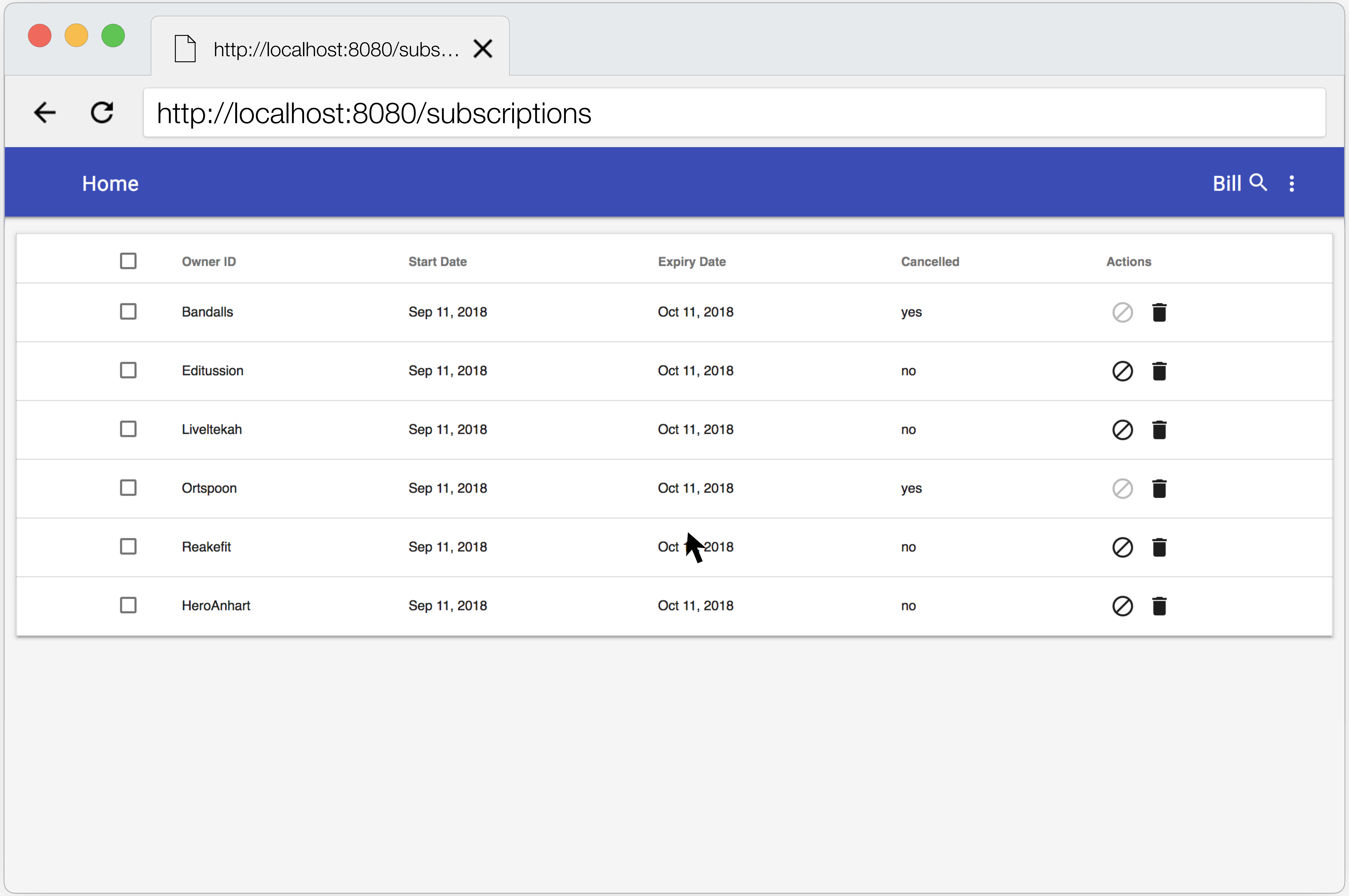
```
<td class="mdl-data-table__cell--non-numeric">
    <form method="post" action="subscriptions">
        <input type="hidden" name="id" value="${subscription.id}"/>
        <input type="hidden" name="action" value="cancel"/>
        <button type="submit" title="Cancel" class="…"
            <#if subscription.canceled>disabled</#if>>
            <i class="material-icons">block</i>
        </button>
    </form>
    <form method="post" action="subscriptions">
        <input type="hidden" name="id" value="${subscription.id}"/>
        <input type="hidden" name="action" value="delete"/>
        <button type="submit" title="Delete" class="…">
            <i class="material-icons">delete</i>
        </button>
    </form>
</td>
```

```
<td class="mdl-data-table__cell--non-numeric">
    <form method="post" action="subscriptions">
        <input type="hidden" name="id" value="${subscription.id}"/>
        <input type="hidden" name="action" value="cancel"/>
        <button type="submit" title="Cancel" class="…"
            <#if subscription.canceled>disabled</#if>>
            <i class="material-icons">block</i>
        </button>
    </form>
    <form method="post" action="subscriptions">
        <input type="hidden" name="id" value="${subscription.id}"/>
        <input type="hidden" name="action" value="delete"/>
        <button type="submit" title="Delete" class="…">
            <i class="material-icons">delete</i>
        </button>
    </form>
</td>
```

```
<td class="mdl-data-table__cell--non-numeric">
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="cancel"/>
    <button type="submit" title="Cancel" class="…"
      <#if subscription.canceled>disabled</#if>>
      <i class="material-icons">block</i>
    </button>
  </form>
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="delete"/>
    <button type="submit" title="Delete" class="…">
      <i class="material-icons">delete</i>
    </button>
  </form>
</td>
```

```
<td class="mdl-data-table__cell--non-numeric">
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="cancel"/>
    <button type="submit" title="Cancel" class="…"
      <#if subscription.canceled>disabled</#if>>
      <i class="material-icons">block</i>
    </button>
  </form>
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="delete"/>
    <button type="submit" title="Delete" class="…">
      <i class="material-icons">delete</i>
    </button>
  </form>
</td>
```

```
<td class="mdl-data-table__cell--non-numeric">
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="cancel"/>
    <button type="submit" title="Cancel" class="…"
      <#if subscription.canceled>disabled</#if>>
      <i class="material-icons">block</i>
    </button>
  </form>
  <form method="post" action="subscriptions">
    <input type="hidden" name="id" value="${subscription.id}"/>
    <input type="hidden" name="action" value="delete"/>
    <button type="submit" title="Delete" class="…">
      <i class="material-icons">delete</i>
    </button>
  </form>
</td>
```



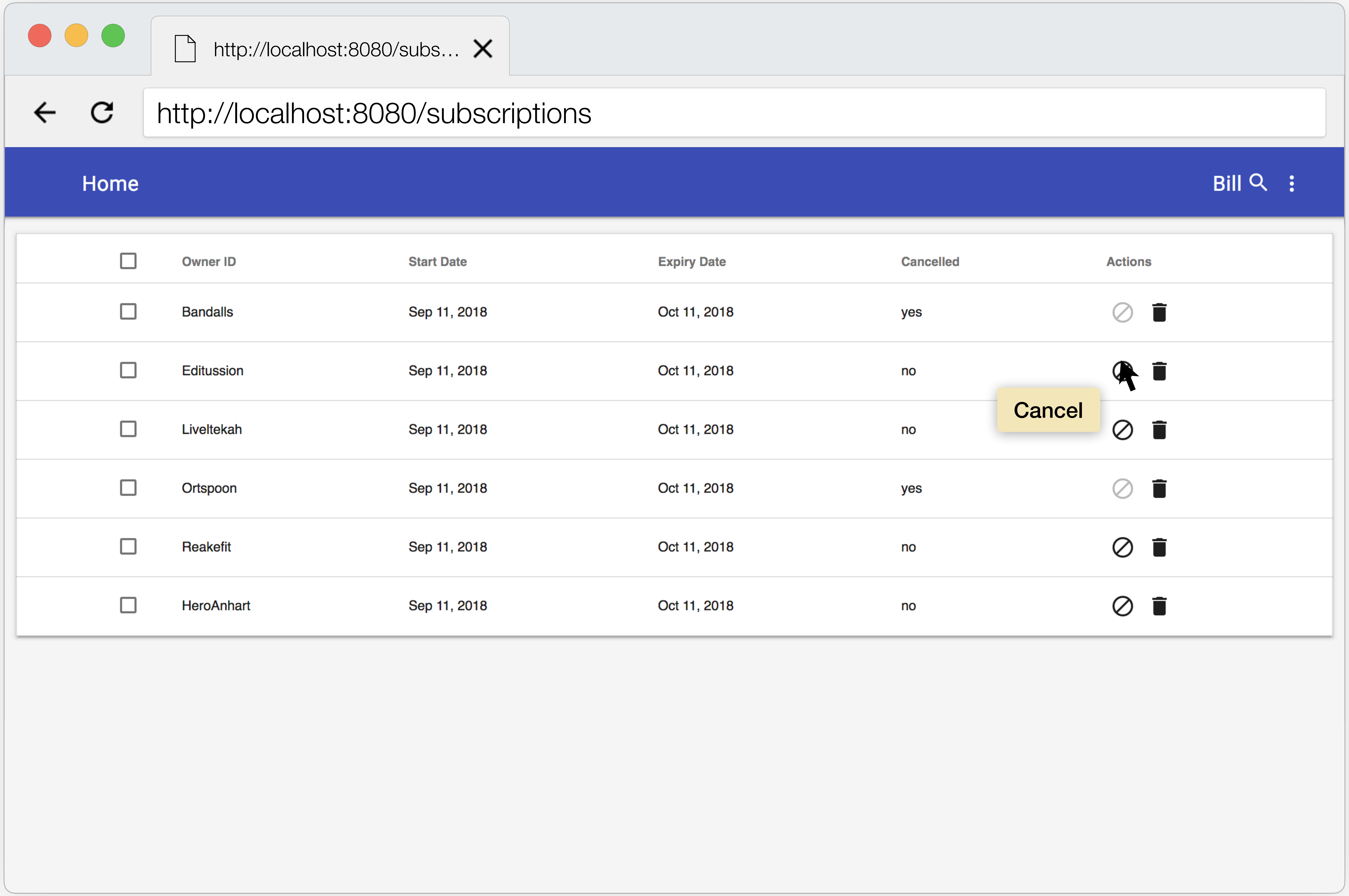
http://localhost:8080/subs…

http://localhost:8080/subscriptions

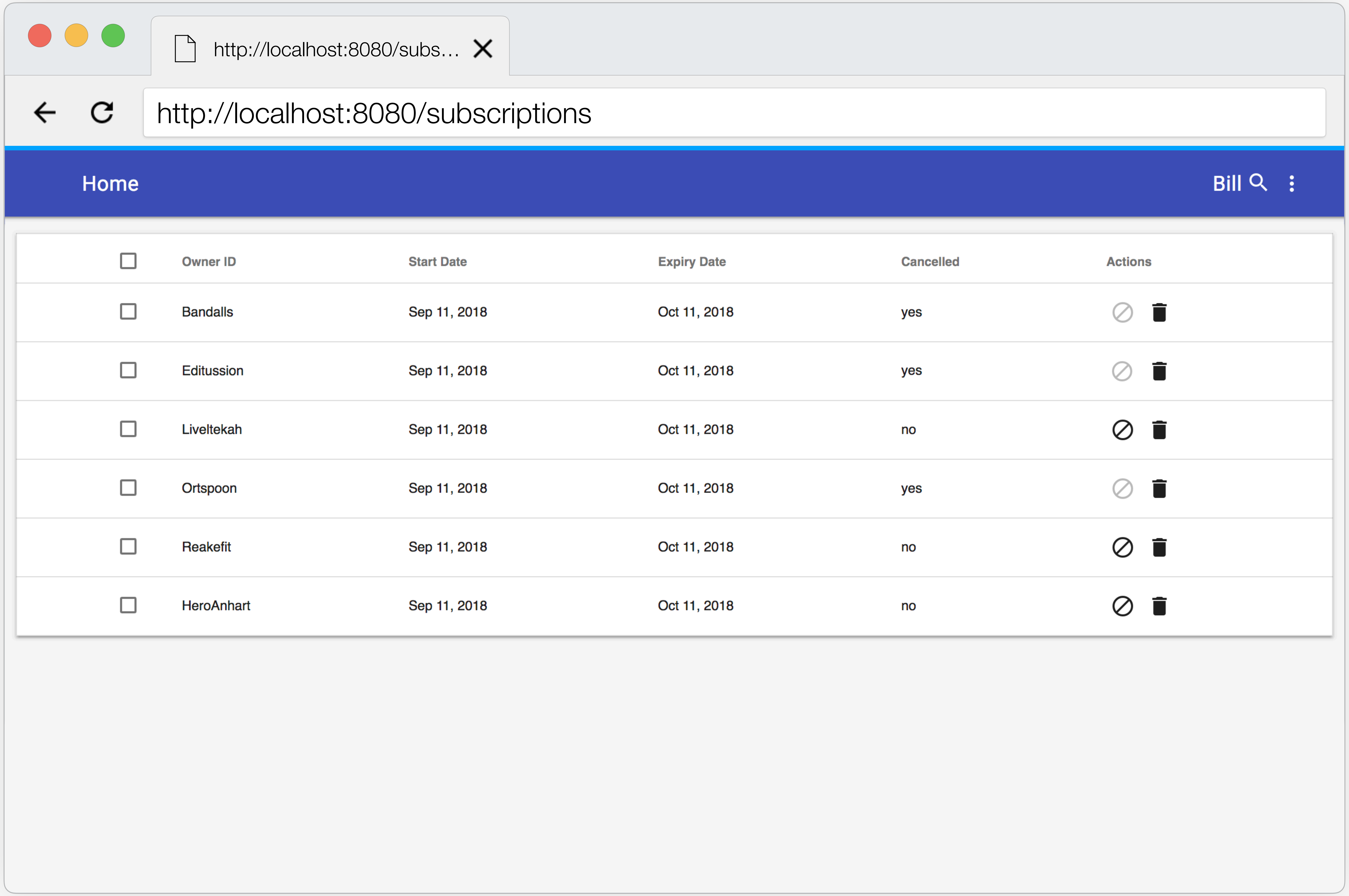
Home

Bill

☐	Owner ID	Start Date	Expiry Date	Cancelled	Actions
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑


http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home
Bill
Cancel

	Owner ID	Start Date	Expiry Date	Cancelled	Actions
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes	
☐	Editussion	Sep 11, 2018	Oct 11, 2018	no	
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no	
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes	
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no	
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no	



http://localhost:8080/subs…
http://localhost:8080/subscriptions
Home
Bill

	Owner ID	Start Date	Expiry Date	Cancelled	Actions
☐	Bandalls	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Editussion	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Liveltekah	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	Ortspoon	Sep 11, 2018	Oct 11, 2018	yes	⊘ 🗑
☐	Reakefit	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑
☐	HeroAnhart	Sep 11, 2018	Oct 11, 2018	no	⊘ 🗑

Bill

Expiry Date	Cancelled	Actions
Oct 11, 2018	yes	
Oct 11, 2018	yes	
Oct 11, 2018	no	
Oct 11, 2018	yes	
Oct 11, 2018	no	
Oct 11, 2018	no	

Expiry Date	Cancelled	Actions
Oct 11, 2018	yes	⊘ 🗑
Oct 11, 2018	yes	⊘ 🗑
Oct 11, 2018	no	⊘ 🗑
Oct 11, 2018	yes	⊘ 🗑
Oct 11, 2018	no	⊘ 🗑
Oct 11, 2018	no	⊘ 🗑

Ktor

- Web http://ktor.io

Ktor

• Web	http://ktor.io
• Github	ktorio / ktor

Ktor

Web	http://ktor.io
Github	ktorio / ktor
Slack	kotlinlang #ktor

Ktor

Web	http://ktor.io
Github	ktorio / ktor
Slack	kotlinlang #ktor
Me	@rharter

Ktor

• Web	http://ktor.io
• Github	ktorio / ktor
• Slack	kotlinlang #ktor
• Me	@rharter
• Slides	https://ryans.link/ktor

Servers ❤️ Kotlin

Ryan Harter
@rharter