



# BUSTRACKER

APLICACIÓN PARA VER LOS TIEMPOS DEL TRANSPORTE PUBLICO

Roberto Blázquez Martín

# TECNOLOGIAS

- Ktor
- React
- Mongo
- Docker
- Nginx

# KTOR



Ktor es un marco de desarrollo de aplicaciones web en Kotlin. Proporciona un conjunto de herramientas y características que permiten crear rápidamente aplicaciones web y API.

# ¿POR QUÉ KTOR?

- Ktor permite desarrollar apis rápido y además tiene mucho mejor rendimiento que alternativas sus alternativas como Spring.

# REACT



- React es una biblioteca de JavaScript utilizada para construir interfaces de usuario interactivas y reactivas.

# ¿POR QUÉ REACT?

- Eficiencia y rendimiento
- Componentes reutilizables
- Gran ecosistema y comunidad
- Multiplataforma

# MONGO



- MongoDB es una base de datos NoSQL (No Relacional) que se caracteriza por su enfoque en la escalabilidad, la flexibilidad y el almacenamiento.

# ¿POR QUÉ MONGO?

- Flexibilidad en el esquema de datos
- Escalabilidad horizontal
- Consultas flexibles



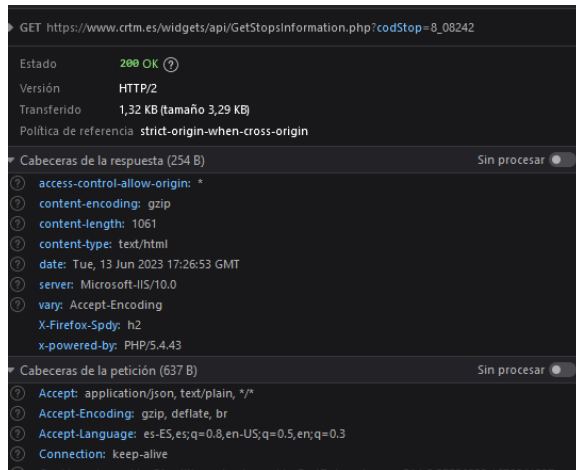
# BACKEND

- Tiempos de las paradas de bus
- Localización de los autobuses
- Tiempos de las estaciones de metro
- Usuarios
- Autenticación
- Paradas Favoritas

# TIEMPOS BUS

- ¿Dónde obtengo los datos?
- Normalizar los datos

# OBTENCIÓN DATOS



```
3 public class WebServices
4 {
5     public static String key = null;
6     private static Date lastKey;
7     public static String privateKey = "pruebapruebapruebapruebaprueba12";
8     public static String server = "http://www.citram.es:8080/WSMultimodalInformation/MultimodalInformation.svc?wsdl";
9     static final String serverV2_viejo = "http://sbit1.crtm.es:50080/spai-crtm/srv/prepago/venta/";
10    static final String serverV2pre = "http://www.citram.es:50081/VENTAPREPAGOTITULO/VentaPrepagoTitulo.svc?wsdl";
11    public static final String server_dev = "http://www.citram.es:8080/WSMultimodalInformation_DEV/MultimodalInformation.svc?wsdl";
12    public static final String server_pre = "http://www.citram.es:8080/WSMultimodalInformation_TEST/MultimodalInformation.svc?wsdl";
13    public static final String server_pro = "http://www.citram.es:8080/WSMultimodalInformation/MultimodalInformation.svc?wsdl";
14 }
```

# GENERAR SOAP

```
implementation("com.sun.xml.ws:jaxws-tools:4.0.1")
```

```
task( name: "wsimport-myservice") {  
    group = BasePlugin.BUILD_GROUP  
    val destDir = file( path: "$projectDir/src/main/java")  
    destDir.mkdirs()  
    val sourceDestDir = file( path: "$projectDir/src/main/java")  
    sourceDestDir.mkdirs()  
    doLast {  
        ant.withGroovyBuilder {  
            "taskdef"( ...keywordArguments:  
                "name" to "wsimport",  
                "classname" to "com.sun.tools.ws.ant.WsImport",  
                "classpath" to jaxws.asPath  
            )  
  
            "wsimport"( ...keywordArguments:  
                "keep" to true,  
                "sourcedestdir" to sourceDestDir,  
                //"destDir" to destDir, already compiled java classes, not needed  
                "package" to "crtm.soap",  
                "wsdl" to "http://www.citram.es:8080/WSMultimodalInformation/MultimodalInformation.svc?wsdl",  
            ) {  
                "xjcarg"( ...keywordArguments: "value" to "-XautoNameResolution")  
            }  
        }  
    }  
}
```

# CONSUMIR SOAP

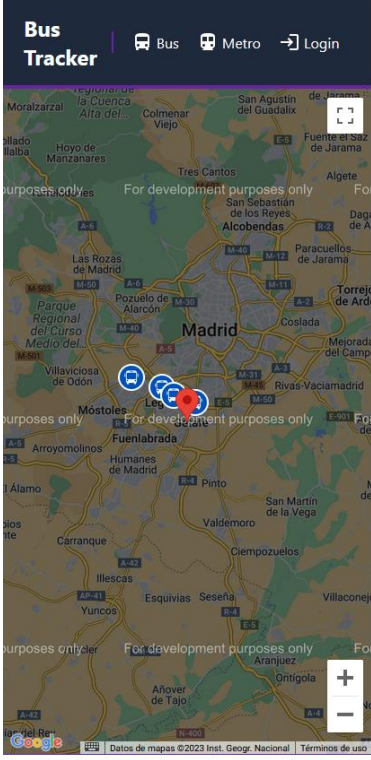
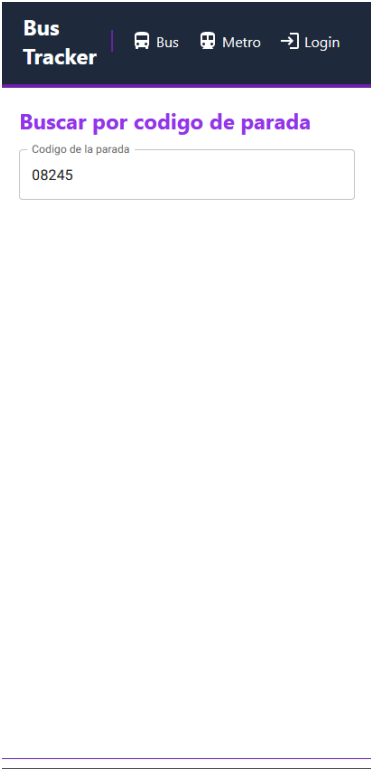
```
💡  
val defaultClient = MultimodalInformation_Service().basicHttp  
  
val privateKey = "pruebapruebapruebapruebaprueba12".toByteArray()  
👤 Roberto Blázquez  
fun MultimodalInformation.auth(): AuthHeader {  
    val key = getPublicKey(PublicKeyRequest())  
    return authHeader(key.key.toByteArray(), privateKey)  
}
```

```
fun getStopTimes(stopCode: String, codMode: String?): ShortStopTimesResponse? {  
    val request = ShortStopTimesRequest().apply {  
        codStop = stopCode  
        type = 1  
        orderBy = 2  
        stopTimesBuIti = 3  
        authentication = defaultClient.auth()  
    }  
    💡 if (codMode != null) request.codMode = codMode  
    return defaultClient.getShortStopTimes(request)  
}
```

# EXPONER LOS ENDPOINTS

```
fun Route.stopsRouting() = route( path: "/stops") {  
    get( path:("/{stopCode}/times") {  
        val stopCode = createStopCode( codMode: "8", call.parameters["stopCode"]!!)  
        val codMode = call.request.queryParameters["codMode"]  
        val timedVCached = try {  
            withTimeout(20.seconds) {  
                val stopTimes = CoroutineScope(Dispatchers.IO).async { getStopTimes(stopCode, codMode) }.await()  
                stopTimes?.stopTimes?.times?.shortTime?.map(::buildStopTimesJson)?.asJson()?.timed()  
            } ?: stopTimesCache.get(stopCode)  
        } catch (e: Exception) {  
            if (e is TimeoutCancellationException) stopTimesCache.get(stopCode)  
            else null  
        } ?: return@get call.respond(HttpStatusCode.BadRequest)  
  
        stopTimesCache.put(stopCode, timedVCached)  
  
        val json = jObject {  
            "data" += timedVCached.value  
            "lastTime" += timedVCached.createdAt.toEpochMilli()  
        }  
        call.respondText(json.serialized(), ContentType.Application.Json)  
    }  
}
```

# FRONT

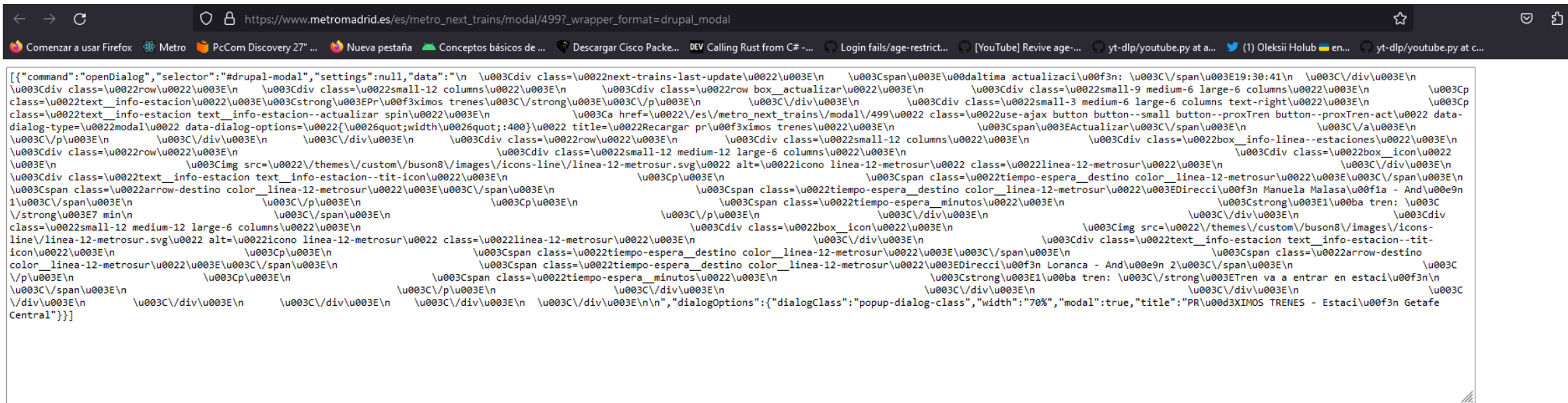


# TIEMPO METRO

- ¿Dónde obtengo los datos?
- Normalizar los datos



## OBTENCIÓN DATOS



# OBTENCIÓN DATOS

The screenshot shows a GitHub repository for 'Caul58 / ScriptableElements'. The file 'libs/TransportTiming.js' is open, showing a JavaScript script that fetches data from a REST API. The script includes comments about variables and a function 'metroTiming' that makes an async request and displays the results in a UI table. The interface includes a sidebar with a file explorer, a top navigation bar with repository stats, and a right sidebar with a 'Symbols' panel.

**Repository:** Caul58 / ScriptableElements (Public)

**Navigation:** Code, Issues, Pull requests, Actions, Projects, Security, Insights

**File Explorer (libs):**

- TransportTiming.js
- BarrioPilarMetroTiming.js
- LICENSE
- PuertaArgandaMetroTiming.js
- README.md

**File:** ScriptableElements / libs / TransportTiming.js

**Commit:** Caul58 Update libs/TransportTiming.js (9f64be4 · 5 years ago) History

**Code View:** 46 lines (36 loc) · 1.61 KB

```
1 // Variables used by Scriptable.
2 // These must be at the very top of the file. Do not edit.
3 // icon-color: red; icon-glyph: bus-alt; share-sheet-inputs: plain-text;
4 var metroTiming = async function (station, presentUI, callback) {
5
6   let req = new Request("https://serviciosapp.metromadrid.es/servicios/rest/teleindicadores/" + station + "?")
7   req.headers = {"Accept": "application/json"}
8   let json = await req.loadJSON()
9
10  let array = json["Vtelindicadores"]
11  let table = new UITable()
```

**Symbols Panel:**

- func metroTiming

# CONSUMIR API

```
fun urlBuilder() = HttpUrl.Builder()
    .scheme("https")
    .host("serviciosapp.metromadrid.es")
    .addPathSegment(pathSegment: "servicios")
    .addPathSegment(pathSegment: "rest")
    .addPathSegment(pathSegment: "teleindicadores")

Roberto <unknown> +1

fun getTimes(id: String? = null): JsonNode? {
    val url = urlBuilder()
        .also { if (id != null) it.addPathSegment(id) }
        .build()




    val request = Request.Builder()
        .url(url)
        .get()
        .addHeader(name: "Accept", value: "application/json")
        .build()
}
```

# FRONT

**Bus Tracker** |  Bus  Metro  Login

**Buscar por estacion**

Nombre de la estacion  
Leganes

**Bus Tracker** |  Bus  Metro  Login

**Leganés Central**

**Linea 12**  
**Anden 1**  
- 3 minutos  
- 11 minutos

**Linea 12**  
**Anden 2**  
- 2 minutos  
- 10 minutos

# USUARIOS

- Registro
- Inicio de sesión
- Verificar
- Restaurar contraseña


# REGISTRO


- Introducen los datos
- Validar y enviar correo
- Usuario verifica


The image shows the front cover of a book. The cover is a solid dark blue color. On the left side, the word "FRONT" is printed in a large, white, sans-serif, all-caps font. The top edge of the image shows a sliver of the book's spine and the edges of the pages, which are white and blue.

Bus

Tracker

 Bus

 Metro


 Login

Register

Email


Username

Password



Register

Account Verification



BusTracker <robertob1022@gmail.com>




martes, 13 de junio de 2023 16:04:20



Deliverability



Responder



Trasladar



Imprimir



Eliminar

Click here to verify your account: <https://159.223.249.18:7777/v1/users/verify?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdwQiOiJ0dXN0dHJ2Y2tlciIsImIzcyI6Im01c198cmFja2YyIiwiaWZ1aWw1OiJhbG9ubnAtMTFhMmZlMz08B538tyY2V1LmNvb3J9.bq1byT90CpqxfTjgPN4i11zW54Sx5J6YFw8Rr4o2w&redirectUrl=https://159.223.249.18/login>

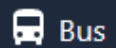
# LOGIN

- Introducen los datos
- Devuelve jwt con la información del usuario



# FRONT

**Bus  
Tracker**



**Buscar por codigo de parada**

Codigo de la parada

**Paradas favoritas**

08242



# RESETEAR CONTRASEÑA

- Introducen los datos
- Se envía un correo con el link para resetear la contraseña
- Se resetea la contraseña

# FRONT

Bus  
Tracker



Bus



Metro



Login

## Reset password

Email

Send email

We will send you an email with a link to reset your password

Bus  
Tracker



Bus



Metro




Logout


## New password

Password

Set new password

## Reset Password

 BusTracker <robertobl022@gmail.com>

 martes, 13 de junio de 2023 20:02:32

 Deliverability

 Responder

 Trasladar

 Imprimir

 Eliminare



Click here to reset your password: <https://159.223.249.18/new-password?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJidXNfdHJhY2t1ciIsImIzcyI6ImJ1c190cmFja2VyIiwiaWZwIjhaWwiOiJhbHQubnAtMTFhMmJmZ0B5b3BtYWlsLmNvbSJ9.bq1byT9DCpqxfTJgPN4i11zW545xSJ6HyPw8Rr4ozJw>



# FAVORITOS

- Guardar
- Leer
- Borrar

# FRONT

**Bus Tracker** |  Bus  Metro  Logout




**Ultima actualizacion de CRTM**  
20:04:06  
Añadir a favoritos  



**1**  
- 20:04:12  
- 20:29:00  
- 20:42:57

**441**  
- 20:12:06  
- 20:28:00  
- 20:40:00

**462**  
- 20:07:00  
- 20:38:53  
- 21:29:04




**N801**  
- 0:46:00  
- 23:26:00

**Bus Tracker** |  Bus  Metro  Logout



**Leganés Central**  
Añadir a favoritos  


**Linea 12**  
**Anden 1**  
- 3 minutos  
- 10 minutos

**Linea 12**  
**Anden 2**  
- 1 minutos  
- 9 minutos

**Bus Tracker** |  Bus  Metro  Logout

**Buscar por codigo de parada**

**Paradas favoritas**  
08242   
08243 

# DESPLIEGE

Primero se compila y se sube la imagen a Docker hub

```
push_to_registry:
  name: Push Docker image to Docker Hub
  runs-on: ubuntu-latest
  needs: build
  steps:
    - name: Check out the repo
      uses: actions/checkout@v3

    - name: Log in to Docker Hub
      uses: docker/login-action@65b78e6e13532edd9afa3aa52ac7964289d1a9c1
      with:
        username: ${ secrets.DOCKER_USERNAME }
        password: ${ secrets.DOCKER_PASSWORD }

    - name: Extract metadata (tags, labels) for Docker
      id: meta
      uses: docker/metadata-action@9ec57ed1fcd9b14dcef7dfbe97b2010124a938b7
      with:
        images: xbank/bus_tracker_api

    - name: Build and push Docker image
      uses: docker/build-push-action@f2ald5e99d037542a71f64918e516c093c6f3fc4
      with:
        context: .
        push: true
        tags: ${ steps.meta.outputs.tags }
        labels: ${ steps.meta.outputs.labels }
        name: remote ssh command

deploy:
```

# DESPLIEGE

Después se despliega en la maquina ejecutando el Docker compose

```
deploy:
  name: Build
  runs-on: ubuntu-latest
  needs: push_to_registry
  env:
    JWT_SECRET: ${ secrets.JWT_SECRET }
    JWT_AUDIENCE: ${ secrets.JWT_AUDIENCE }
    JWT_ISSUER: ${ secrets.JWT_ISSUER }
    STMP_HOST: ${ secrets.STMP_HOST }
    STMP_PORT: ${ secrets.STMP_PORT }
    STMP_USERNAME: ${ secrets.STMP_USERNAME }
    STMP_PASSWORD: ${ secrets.STMP_PASSWORD }
    MONGO_CONNECTION_STRING: ${ secrets.MONGO_CONNECTION_STRING }
    MONGO_DATABASE_NAME: ${ secrets.MONGO_DATABASE_NAME }
    MONGO_INITDB_ROOT_USERNAME: ${ secrets.MONGO_INITDB_ROOT_USERNAME }
    MONGO_INITDB_ROOT_PASSWORD: ${ secrets.MONGO_INITDB_ROOT_PASSWORD }
  steps:
    - name: executing remote ssh commands using password
      uses: appleboy/ssh-action@v0.1.10
      with:
        envs: JWT_SECRET,JWT_AUDIENCE,JWT_ISSUER,STMP_HOST,STMP_PORT,STMP_USERNAME,STMP_PASSWORD,MONGO_CONNECTION_STRING,MONGO_DATABASE_NAME,MONGO_INITDB_ROOT_USERNAME,MONGO_INITDB_ROOT_PASSWORD
        host: ${ secrets.HOST }
        username: ${ secrets.USERNAME }
        password: ${ secrets.PASSWORD }
        port: ${ secrets.PORT }
        script: |
          export JWT_SECRET=$JWT_SECRET
          export JWT_AUDIENCE=$JWT_AUDIENCE
          export JWT_ISSUER=$JWT_ISSUER
          export STMP_HOST=$STMP_HOST
          export STMP_PORT=$STMP_PORT
          export STMP_USERNAME=$STMP_USERNAME
          export STMP_PASSWORD=$STMP_PASSWORD
          export MONGO_CONNECTION_STRING=$MONGO_CONNECTION_STRING
          export MONGO_DATABASE_NAME=$MONGO_DATABASE_NAME
          export MONGO_INITDB_ROOT_USERNAME=$MONGO_INITDB_ROOT_USERNAME
          export MONGO_INITDB_ROOT_PASSWORD=$MONGO_INITDB_ROOT_PASSWORD
          docker compose -f ../bus-tracker-back/docker-compose.yml stop
          docker compose -f ../bus-tracker-back/docker-compose.yml rm -f
          docker compose -f ../bus-tracker-back/docker-compose.yml pull
```



GRACIAS