



VERT.X SIMPLICIDADE

Que nos traz **FELICIDADE**

AGENDA

VERT.X CONCEPTS

REDUCING COMPLEXITY

OPEN API -> OPENID

JDBC ASYNC

METRICS

DISTRIBUTED TRACING

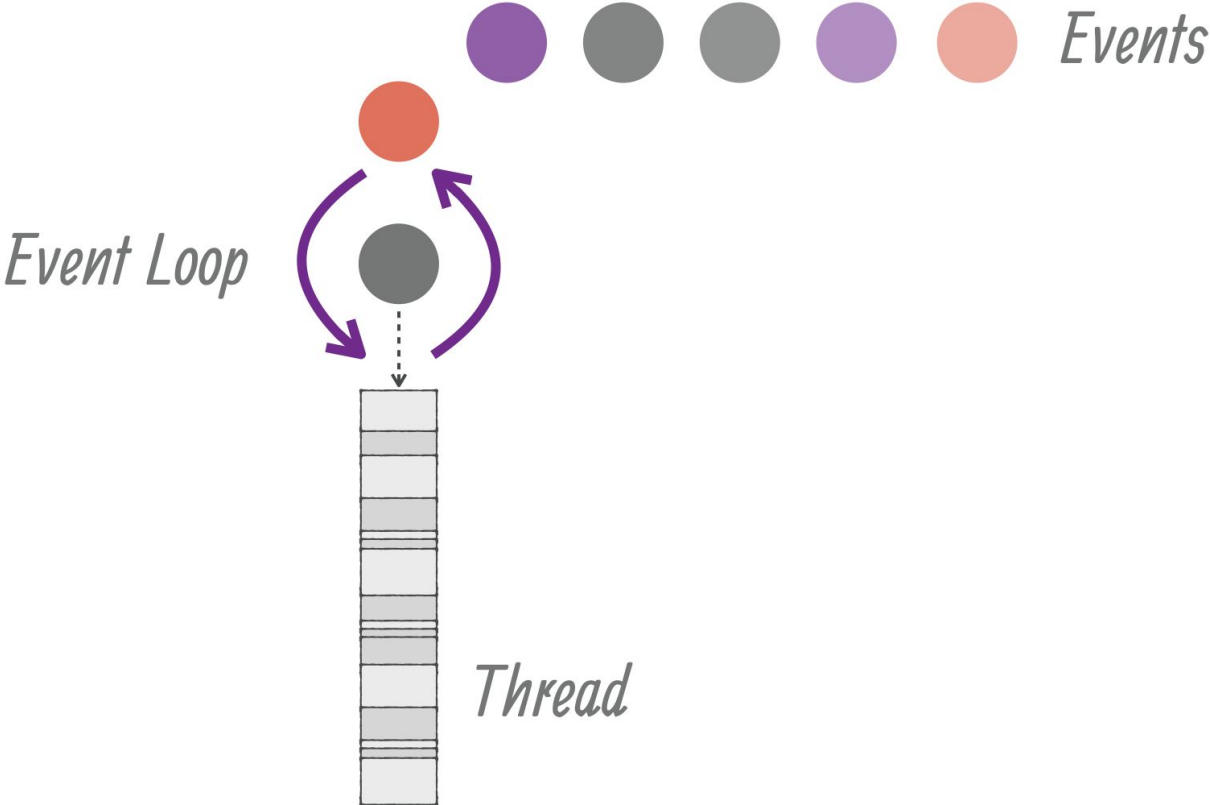
VERT.X VERTICLES

Vertices are chunks of code that get deployed and run by Vert.x. A Vert.x instance maintains N event loop threads (where N by default is $\text{core} * 2$) by default.

Worker - Thread Pool in general for blocking code

Standard - Event Loop

VERT.X EVENT LOOP



```
package io.vertx.example.web.helloworld;

import io.vertx.core.AbstractVerticle;
import io.vertx.example.util.Runner;
import io.vertx.ext.web.Router;

/*
 * @author <a href="http://tfox.org">Tim Fox</a>
 */
public class Server extends AbstractVerticle {

    // Convenience method so you can run it in your IDE
    public static void main(String[] args) {
        Runner.runExample(Server.class);
    }

    @Override
    public void start() throws Exception {

        Router router = Router.router.vertx();

        router.route().handler(routingContext -> {
            routingContext.response().putHeader("content-type", "text/html").end("Hello World!");
        });

        vertx.createHttpServer().requestHandler(router).listen(8080);
    }
}
```

VERT.X EVENT BUS

Is the nervous system of Vert.x.

The event bus allows different parts of your application to communicate with each other, and whether they're in the same Vert.x instance, or in a different Vert.x instance.

```

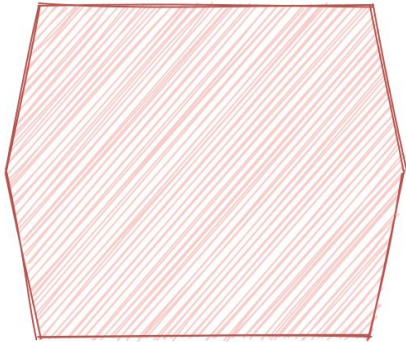
29
30 @Override
31 public void start() throws Exception {
32
33     Router router = Router.router(vertex);
34
35     // Allow events for the designated addresses in/out of the event bus bridge
36     SockJSBridgeOptions opts = new SockJSBridgeOptions()
37         .addOutboundPermitted(new PermittedOptions().setAddress("feed"));
38
39     // Create the event bus bridge and add it to the router.
40     SockJSHandler ebHandler = SockJSHandler.create(vertex);
41     router.mountSubRouter("/eventbus", ebHandler.bridge(opts));
42
43     // Create a router endpoint for the static content.
44     router.route().handler(StaticHandler.create());
45
46     // Start the web server and tell it to use the router to handle requests.
47     vertex.createHttpServer().requestHandler(router).listen(8080);
48
49     EventBus eb = vertex.eventBus();
50
51     vertex.setPeriodic(1000, t -> {
52         // Create a timestamp string
53         String timestamp = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.MEDIUM).format(Date.from(Instant.now()));
54
55         eb.send("feed", new JsonObject().put("now", timestamp));
56     });
57 }
58 }

```


[illegible]

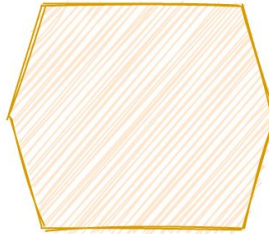
1990 - 2000

MVC - Data-Centric -
etc....



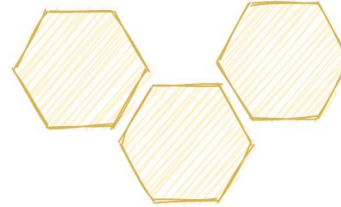
2000 - 2013

SOA - Event-Driven



2013 - 2015

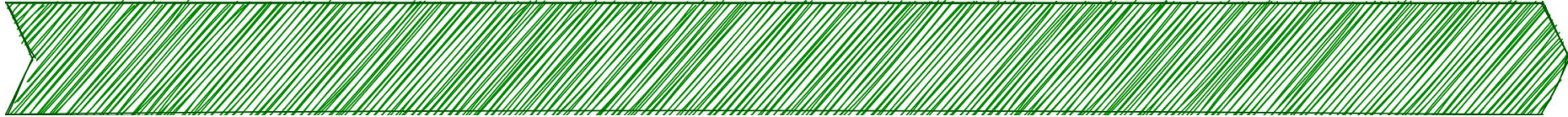
Microservices



-code and a lot of
integrations

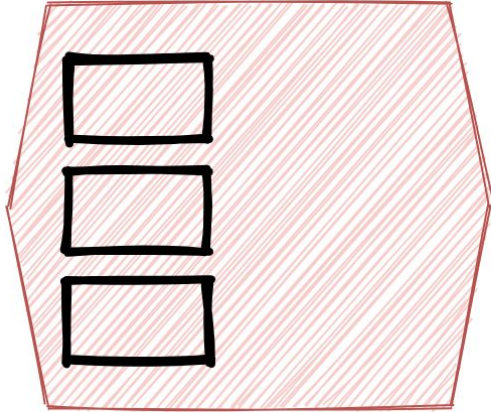
2015

Serverless



1990 - 2000

MVC - Data-Centric -
etc....



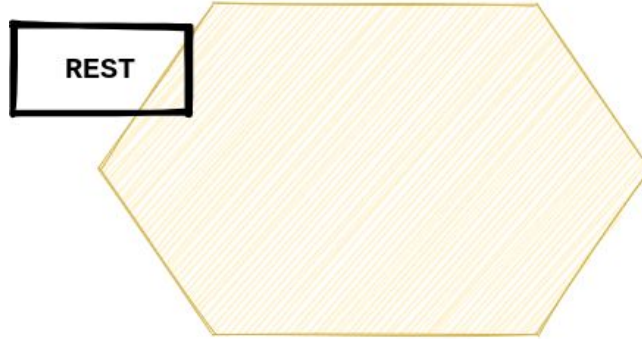
million lines of code....

LOTS OF MODULES
DIFFERENT INTEGRATIONS MODELS
DESIGN PATTERNS IS VERY IMPORTANT

DDD
HEXAGONAL ARCHITECTURE

2013 - 2015

Microservices

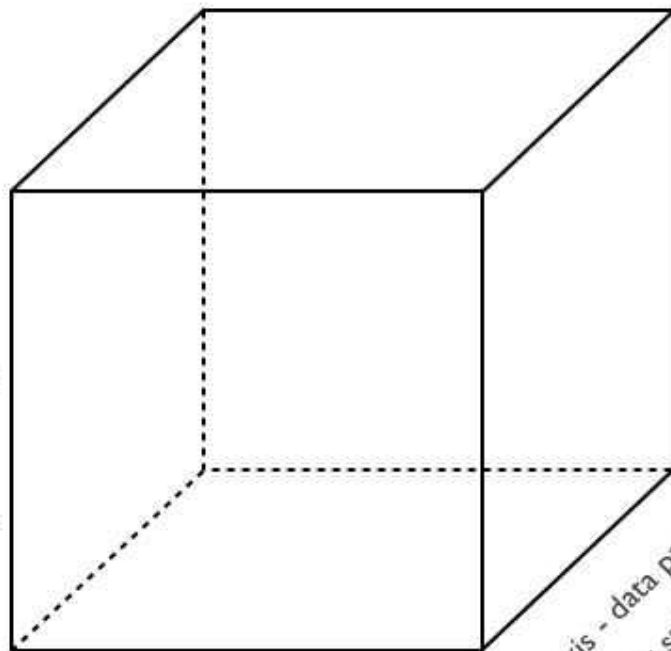


APPLICATIONS SCOPES ARE REDUCED
MANY INTEGRATIONS
CONTRACTS WILL BE VERY HELPFUL

3 dimensions to scaling

Y axis -
functional
decomposition

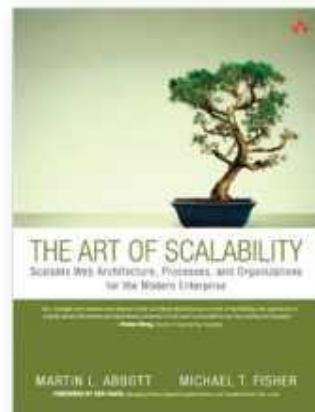
Scale by
splitting
different things



X axis - horizontal duplication

Scale by cloning

Z axis - data partitioning
Scale by splitting similar things

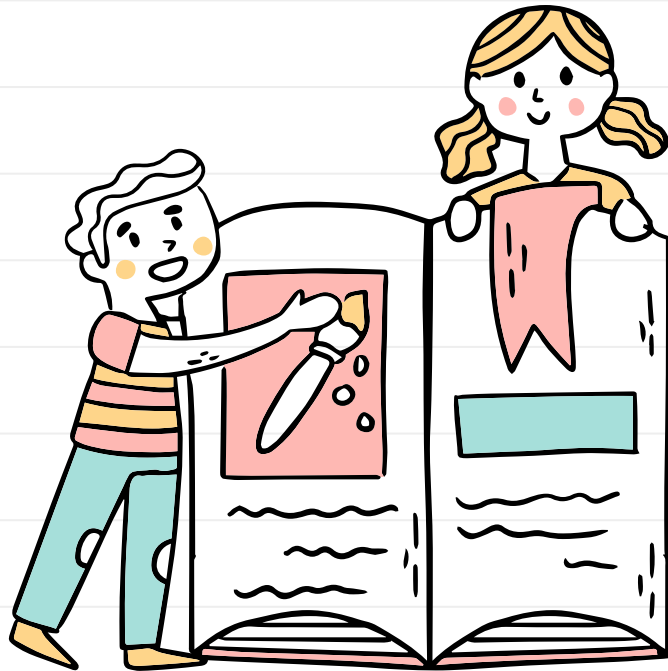




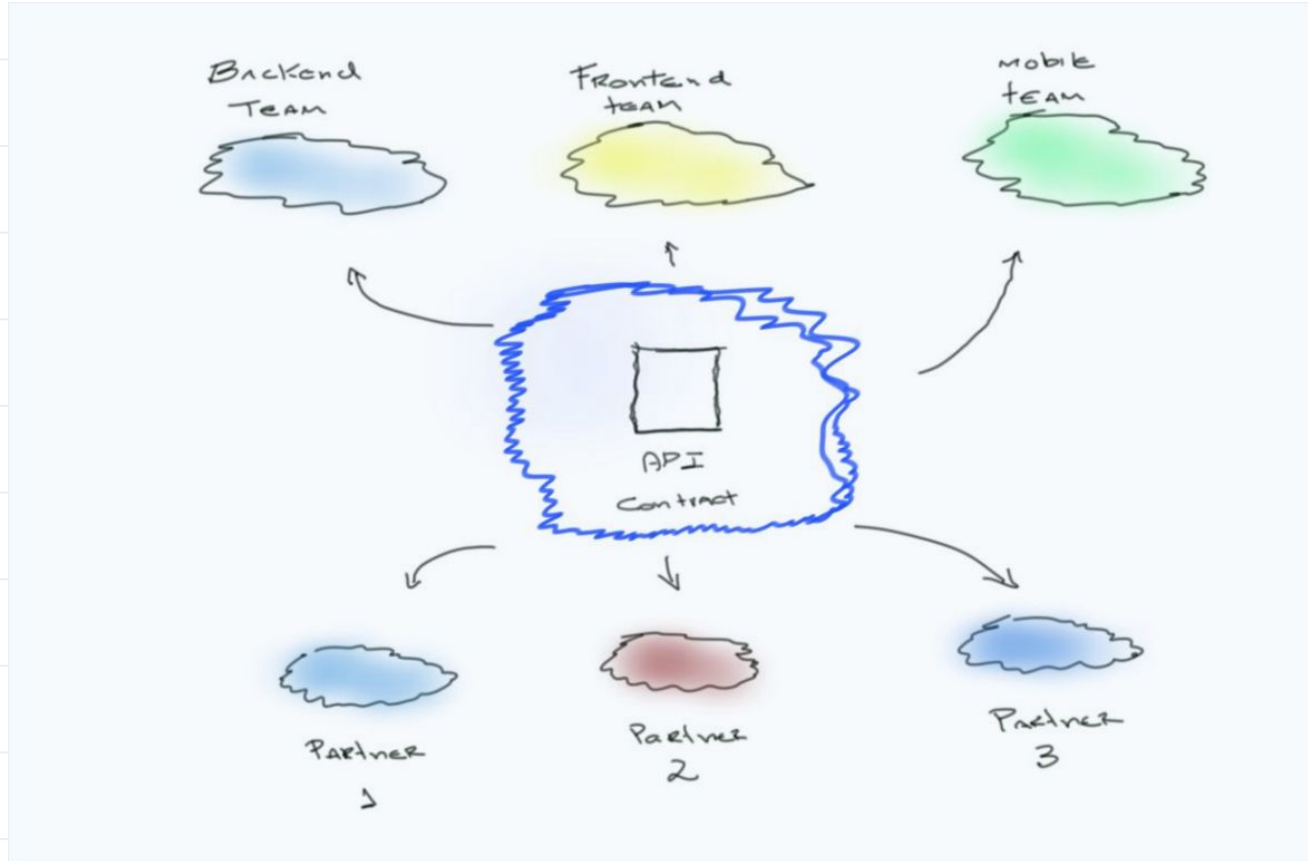
THE CONTRACT IS
OUR "NEW"
ABSTRACTION

API FIRST

An *API-First* approach means that for any given development project, your APIs are treated as *first-class citizens*.



OVERVIEW DIAGRAM



users 1.0 OAS3

API to create users on chat systems

[Claudio E. de Oliveira - Website](#)
[Send email to Claudio E. de Oliveira](#)
MIT

Servers

▾

users Users API's



GET `/users/{userId}` Get User Info by User ID

PATCH `/users/{userId}` Update user data to blocked or non-blocked

POST `/users` Create New User

Schemas



User >



NewUser >



How

VERT.X

SUPPORTS IT????

```
69 ▾ /users:
70 ▾   post:
71     summary: Create New User
72     operationId: create-user
73 ▾   tags:
74     - users
75 ▾   responses:
76 ▾     '200':
```

```
49 LOG.info("OAS: " + cfg.getString( key: "openAPI"));
50 Future<Void> builderPromise = RouterBuilder.create(vertex, cfg.getString( key: "openAPI"))
51   .compose(builder -> {
52     builder.operation( operationId: "get-users-userId").handler(new FindUserHandler(client));
53     builder.operation( operationId: "create-user").handler(new CreateUserHandler(client, vertex));
54     builder.operation( operationId: "update-user").handler(new UpdateUserHandler(client));
55     var router : Router = builder.createRouter();
56     router.errorHandler( statusCode: 400, ctx -> {
57       LOG.error( message: "Bad Request", ctx.failure());
58     });
59     // Infra endpoints
```

```

public class FindUserHandler implements Handler<RoutingContext> {

    private final Logger LOG = LoggerFactory.getLogger(this.getClass());

    private final PgPool client;

    public FindUserHandler(PgPool client) {
        this.client = client;
    }

    @Override
    public void handle(RoutingContext rc) {
        var userId :String = rc.pathParam( name: "userId");
        SqlTemplate.forQuery(this.client, template: "SELECT * FROM users WHERE id=#{id}").mapTo(UserRowMapper.INSTANCE).execute(
            Collections.singletonMap("id",userId)).onSuccess(rows ->{
                if(rows.iterator().hasNext()){
                    LOG.info("Loading user ID: " + userId);
                    var user :User = rows.iterator().next();
                    rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8").end(user.toJson().encode());
                }else{
                    LOG.error("User was not found ID: " + userId);
                    rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8").setStatusCode(404).end();
                }
            }).onFailure(err ->{
                LOG.error( message: "Error to execute instruction in database ", err);
                rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8").setStatusCode(400).end();
            });
    }
}

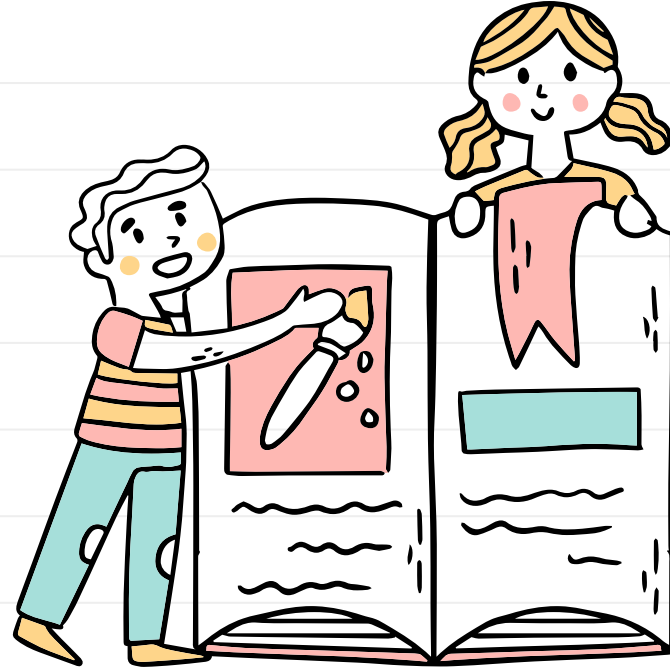
```

HOW ABOUT VERT.X AND CODE COMPLEXITY???

1. Easy way to implement OpenAPI endpoints
2. Validate input data using OpenAPI definition
3. Handlers is totally Single Response Principle (SRP)
4. SQL Client in Vert.x is asynchronous
5. Avoid callback Vert.x 4 encourages *Futures*, sometimes called "Futurisation"

API SECURITY

APIs is very important to achieve business goals. Security should be **first-class citizen** in APIs.





OpenID

How

VERT.X

SUPPORTS IT????

HOW ABOUT VERT.X SUPPORTS OPENID

1. Auth JWT
2. Configure issuer (OP) in configuration file
3. Download keys from /certs obtained from .well-know endpoints
4. Easy way to achieve token verifier

```

private Future<JWTAuth> jwtAuth() {
    LOG.info("Starting configuring IDP servers...");
    LOG.info("===== IDP ===== " + this.jwtConfig.issuer());
    var promise : Promise<JWTAuth> = Promise.<JWTAuth>promise();
    LOG.info("IDP server: " + jwtConfig.issuer());
    var jwksUri : String = jwtConfig.jwks();
    var webClient : WebClient = WebClient.create(this.vertx);
    // fetch JWKS from `/certs` endpoint
    webClient.getAbs(jwksUri)
        .as(BodyCodec.jsonObject())
        .send(ar -> {
            if (!ar.succeeded()) {
                LOG.error( message: "Fail on IDP configuration", ar.cause());
                promise.fail(String.format("Could not fetch JWKS from URI: %s", jwksUri));
                return;
            }
            var response : HttpResponse<JsonObject> = ar.result();
            var jwksResponse : JsonObject = response.body();
            var keys : JsonArray = jwksResponse.getJsonArray( key: "keys");
            // Configure JWT validation options
            var jwtOptions = new JWTOptions();
            jwtOptions.setIssuer(jwtConfig.issuer());
            // extract JWKS from keys array
            var jwks : List<JsonObject> = ((List<Object>) keys.getList()).stream()
                .map(o -> new JsonObject((Map<String, Object>) o))
                .collect(Collectors.toList());
            // configure JWTAuth
            var jwtAuthOptions = new JWTAuthOptions();

            jwtAuthOptions.setJwks(jwks);
            jwtAuthOptions.setJWTOptions(jwtOptions);
            JWTAuth jwtAuth = JWTAuth.create(vertx, jwtAuthOptions);
            LOG.info("IDP configured successfully!!");
            promise.complete(jwtAuth);
        });
    LOG.info("Still configuring IDP servers...");
    return promise.future().compose(Future::succeededFuture);
}

```

```

securitySchemes:
  Authorization:
    type: openIdConnect
    openIdConnectUrl: 'http://35.188.212.146/auth/realms/tdc-innovation/.well-known/openid-configuration'

security:
  - openId:
    - openid

```

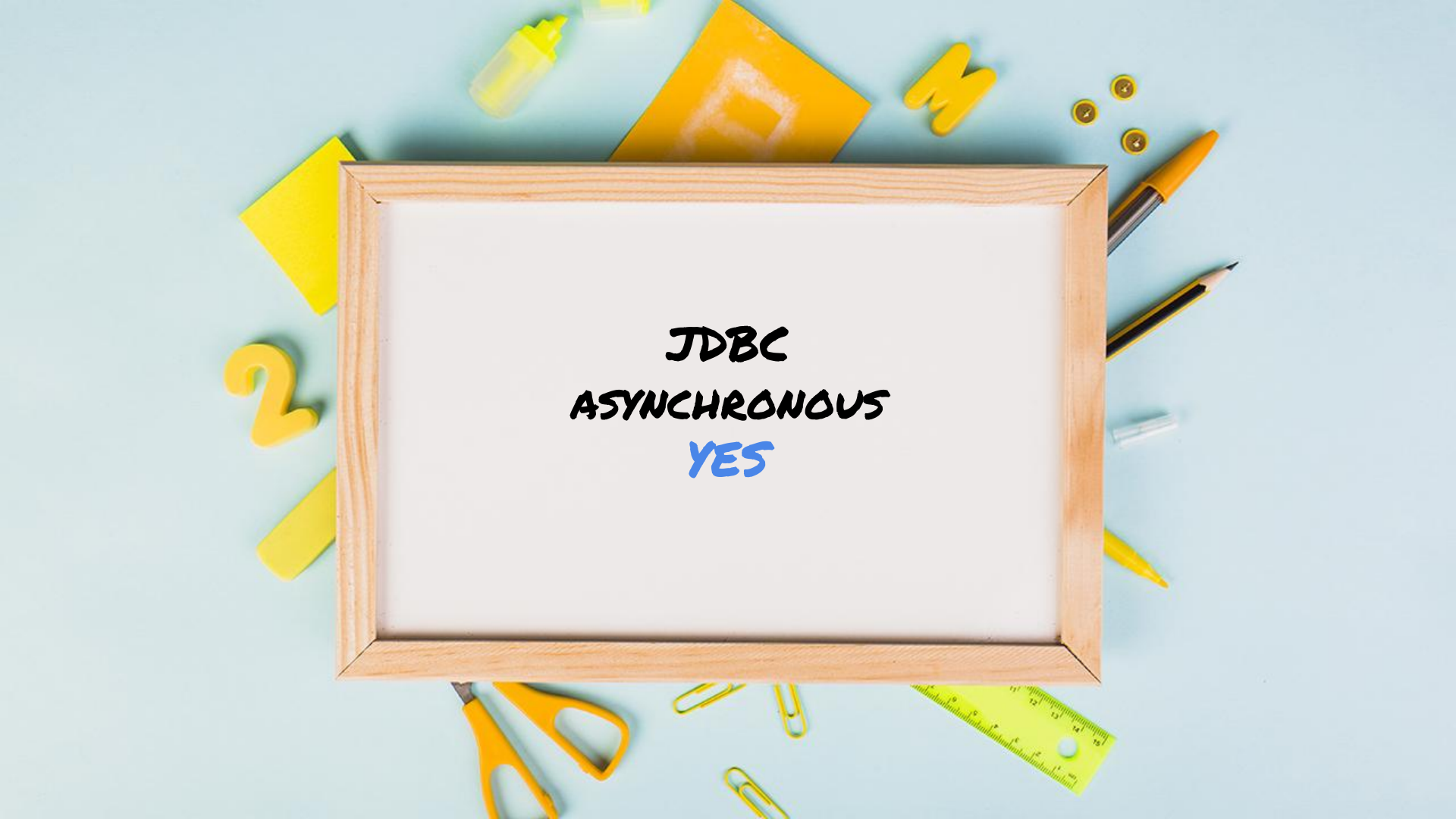
```

private Future<Router> createRouter(JWTAuth jwtAuth) {
  LOG.info("OAS: " + this.apiConfig.getPath());
  return RouterBuilder.create(vertex, apiConfig.getPath()).compose(builder -> {

    builder.securityHandler( securitySchemaName: "openId",JWTAuthHandler.create(jwtAuth));

    builder.operation( operationId: "create-thread").handler(new CreateThreadHandler(this.client,this.vertx));
    builder.operation( operationId: "create-messages").handler(new CreateMessageHandler(this.client,this.vertx,this.policeOfficer));
    var router :Router = builder.createRouter();
    router.errorHandler( statusCode: 400, ctx -> {
      LOG.error( message: "Bad Request", ctx.failure());
    });
  });
}

```



JDBC
ASYNCHRONOUS
YES

```
7
8 insertTemplate.execute(user).onSuccess(result -> {
9     if (result.rowCount() > 0) {
10         rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8")
11         .setStatusCode(201).end(user.toJson().encode());
12     } else {
13         LOG.error("User was not created ID:" + user.getId());
14         rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8")
15         .setStatusCode(500).end(new JsonObject().encode());
16     }
17 }).onFailure(err -> {
18     LOG.error( message: "Error to execute instruction in database ", err);
19     rc.response().putHeader( name: "content-type", value: "application/json; charset=utf-8").setStatusCode(400)
20     .end();
21 });
22
```

```
9
10 @DataObject(generateConverter = true)
11 @RowMapped(formatter = SnakeCase.class)
12 @ParametersMapped(formatter = SnakeCase.class)
13 public class User {
14
15     private String id;
16     @Column(name = "first_name")
17     private String firstName;
18     @Column(name = "last_name")
19     private String lastName;
20     private String email;
21     @Column(name = "email_verified")
22     private Boolean emailVerified;
23
24     private Boolean blocked;
25
26     public User() {
27     }
28
29     public User(JsonObject json) {
30         UserConverter.fromJson(json, obj: this);
31     }
```

FEATURES

1. Supports sql-templates
2. Use annotations
3. Add dependency to process annotations
4. It is asynchronous

[illegible]

HOW ABOUT VERT.X AND METRICS

1. You can use micrometer metrics
2. Add prometheus a metric collector
3. Metrics from HTTP Endpoints are out-of-box
4. Easy way to expose these metrics

How

VERT.X

SUPPORTS IT????



```
Vertx vertx = Vertx.vertx(new VertxOptions().setMetricsOptions(  
    new MicrometerMetricsOptions()  
        .setPrometheusOptions(new VertxPrometheusOptions().setEnabled(true))  
        .setEnabled(true))
```

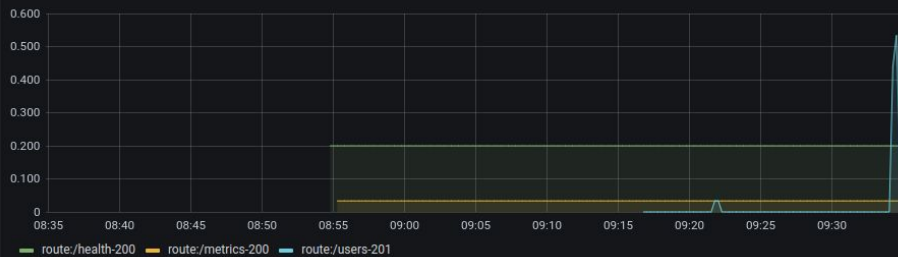
```
// Infra endpoints
```

```
router.route( path: "/metrics").handler(PrometheusScrapingHandler.create());
```

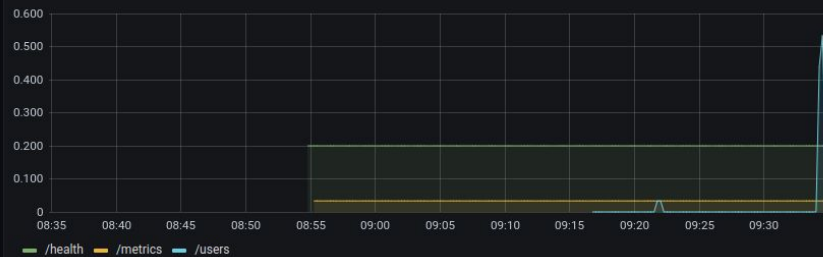
```
this.meterRegistry.counter(NEW_THREAD_METRIC, ...tags: "owner_id", threadCreated.getOwner()).increment();
```

Users Infra Metrics

RPS per Route/Service By Status Code



Users Route TPS



Users Business Metrics

New Users



Users By Domain



DISTRIBUTED TRACING



HOW ABOUT VERT.X AND DISTRIBUTED TRACING

1. It Supports OpenTracing and Zipkin
2. Configuration are done using environment
3. You can choose the tracing policy PROPAGATE, ALWAYS, and IGNORE
4. You can enable trace in vert.x Event Bus

How

VERT.X

SUPPORTS IT????

```
public static void main(String[] args) {  
    Vertx vertx = Vertx.vertx(new VertxOptions().setMetricsOptions(  
        new MicrometerMetricsOptions()  
            .setPrometheusOptions(new VertxPrometheusOptions().setEnabled(true))  
            .setEnabled(true))  
        .setTracingOptions(new OpenTracingOptions()));  
    vertx.deployVerticle(new MainVerticle());  
    vertx.deployVerticle(new UserDataVerticle());  
}
```

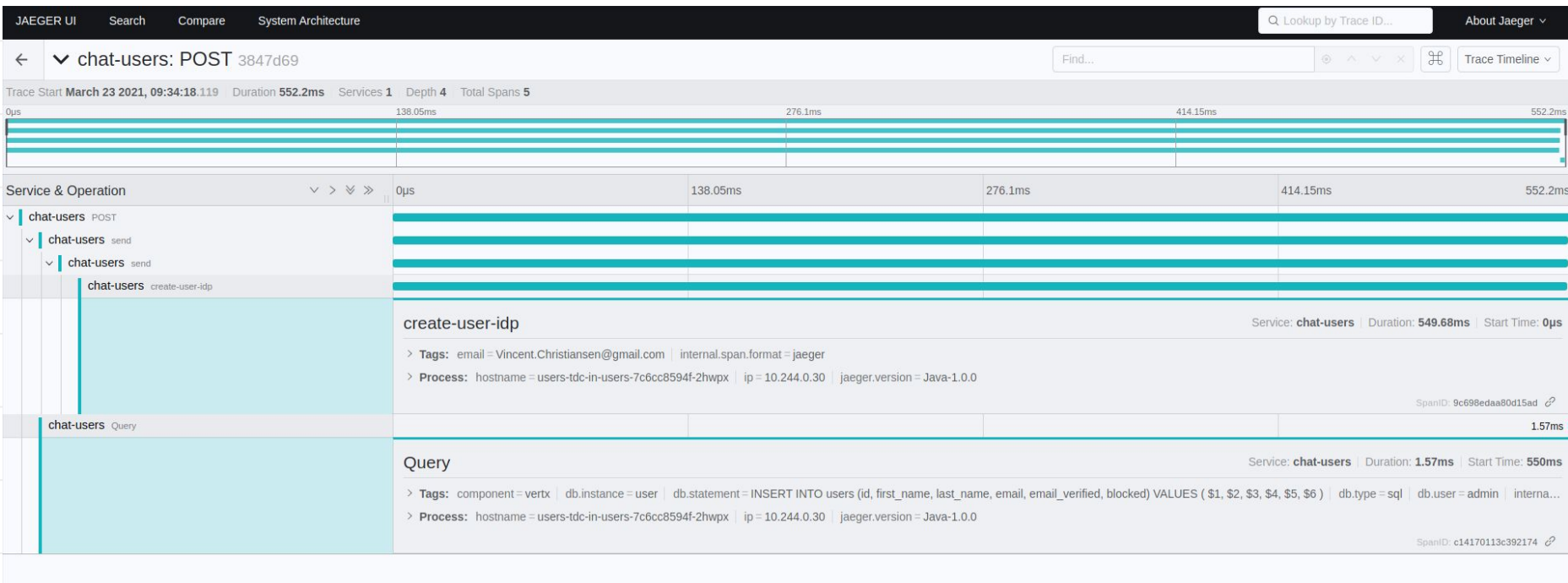
```
private final Vertx vertx;

private final DeliveryOptions deliveryOptions = new DeliveryOptions().setTracingPolicy(TracingPolicy.ALWAYS);

private final MeterRegistry meterRegistry = BackendRegistries.getDefaultNow();

public CreateThreadHandler(PgPool dbClient, Vertx vertx) {
    this.dbClient = dbClient;
    this.vertx = vertx;
}

@Override
public void handle(RoutingContext rc) {
    var userId :String = rc.user().principal().getString( key: "sub");
    | this.vertx.eventBus().request( address: "request.user.data", Json.encode(new UserId(userId)),this.deliveryOptions)
        .onSuccess(data -> {
            var user :User = Json.decodeValue(data.body().toString(), User.class);
            SqlTemplate<Thread, SqlResult<Void>> insertTemplate = SqlTemplate
                .forUpdate(this.dbClient,
```



OTHER FEATURES

CONFIG

Read Vert.x configuration from json, env and others.

REDIS

Supports Redis Clients in Asynchronous Fashion

GRAPHQL

Support graphQL endpoints in your Vert.x Services

CIRCUIT BREAKER

Implementation of the circuit-breaker pattern to mitigate failures.

KAFKA

This component provides a Kafka client for reading and sending messages from/to an Apache Kafka cluster.

MONGODB

A Vert.x client allowing applications to interact with a MongoDB instance, whether that's saving, retrieving, searching, or deleting documents

CONCLUSIONS

01



Think about
Software
Architecture
and choose
carefully

02



Vert.x support
microservices
"standards"
metrics,
tracing....

03

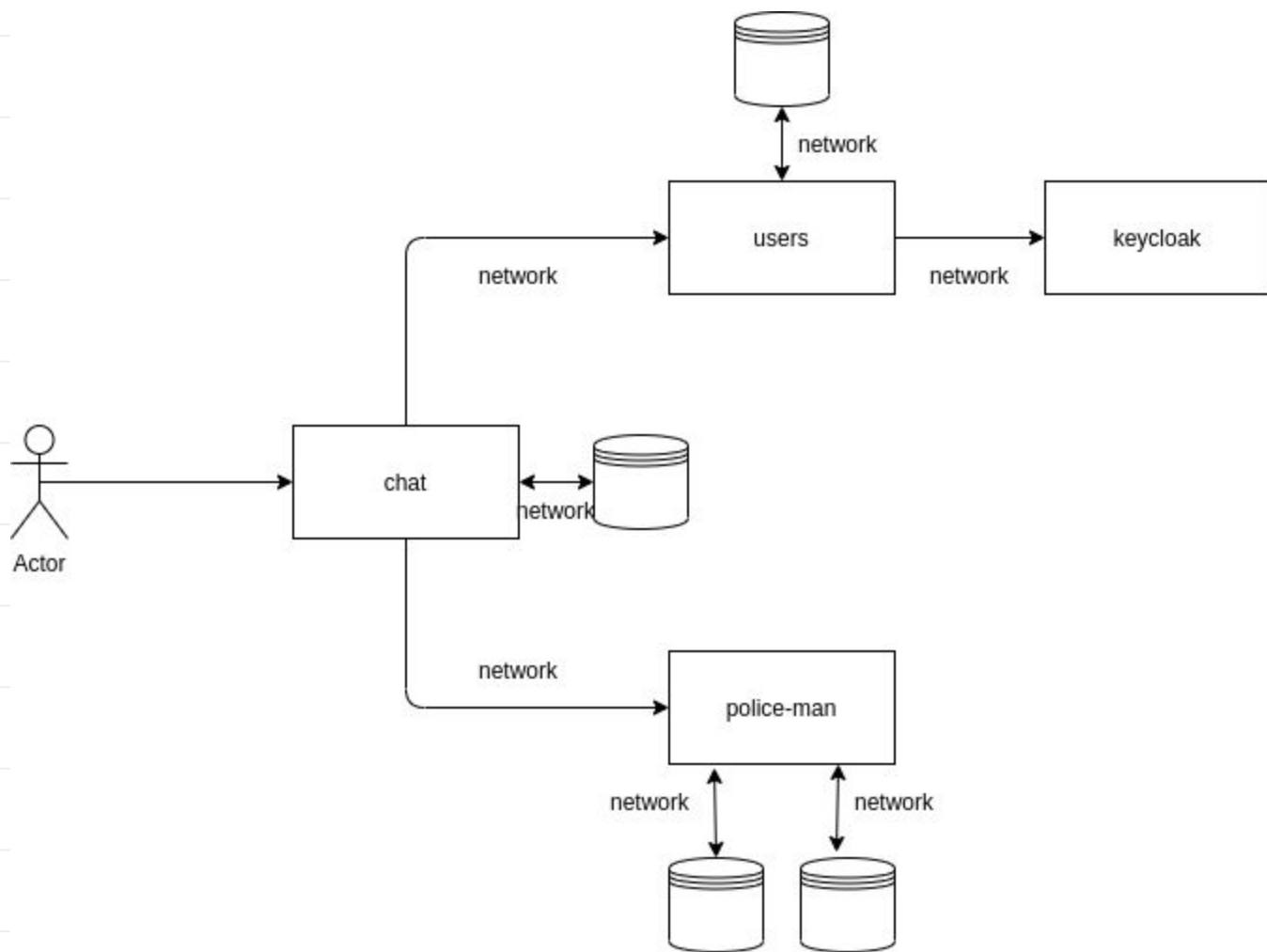


Use contracts
to improve
communication
between
teams

04



It is not a
silver bullet,
think in
another
option in your
toolbox



THANKS!

Do you have any questions?
claudioed.oliveira@gmail.com
@claudioed
<https://claudioed.tech>



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

GITHUB

<https://github.com/claudioed/tdc-inn-users>

<https://github.com/claudioed/tdc-inn-chat>

<https://github.com/claudioed/tdc-inn-police-man>