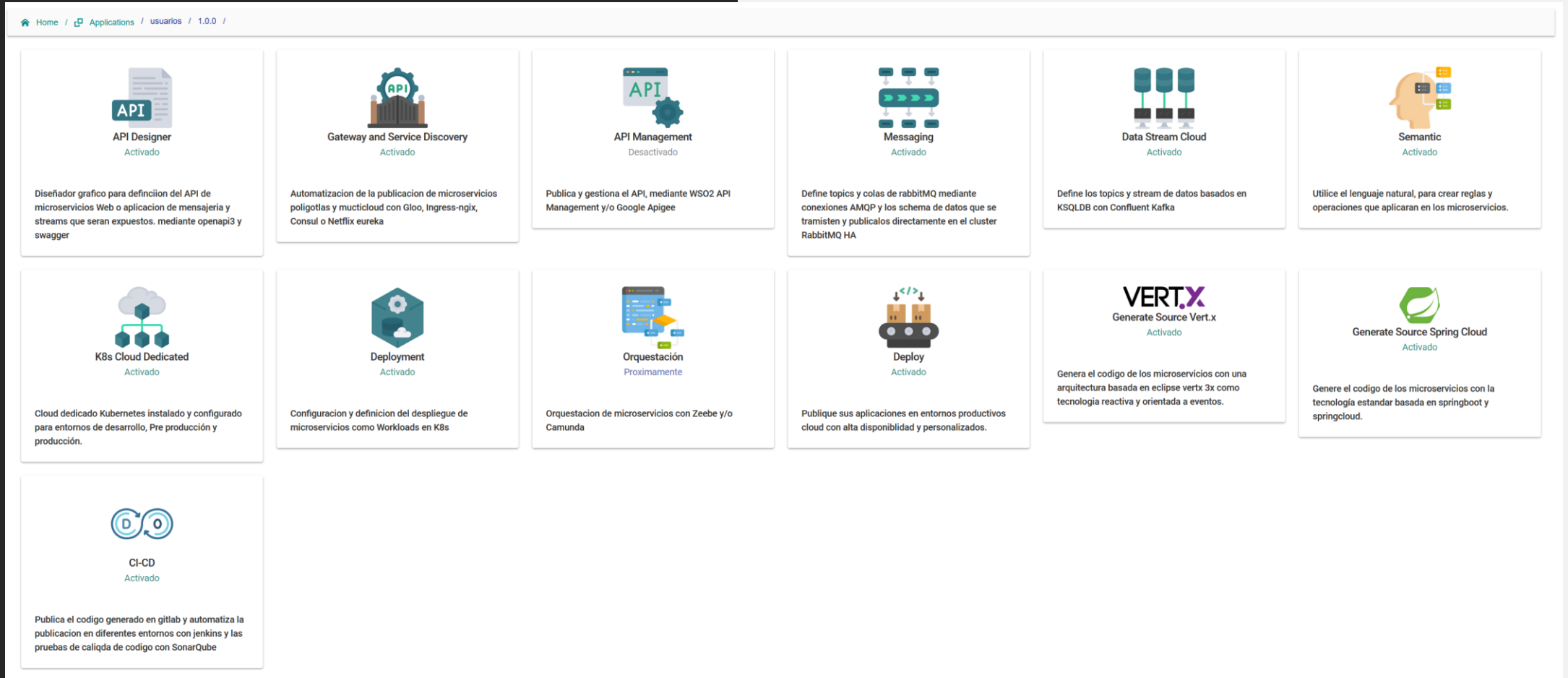




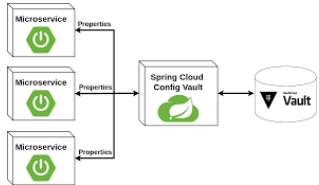
Open Low code Platform  
for Microservices and  
Business Applications

# Arquitectura Poliglota de Microservicios

## Roadmap Base



# Stack Tecnológico Micro Services



Register and Service  
Discovery



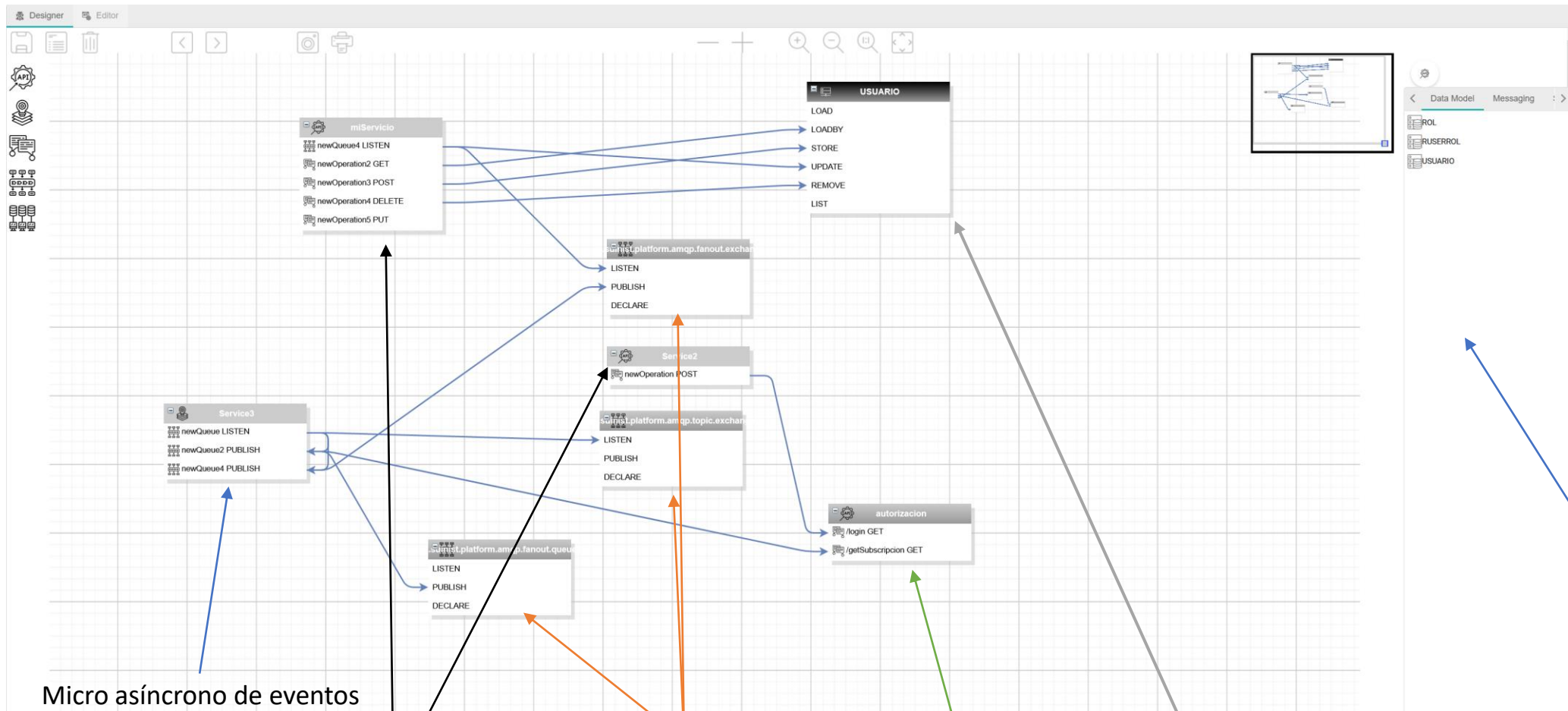
BPMN / BPEL



Orchestration



# Diseñador de Micro Servicios y API



Micro asíncrono de eventos y streams

Micro síncrono admite llamadas REST, mensajes y Streams

Mensajes o Streams

Llamada a servicio externo

Tabla, Vista o Stored procedure Base de datos

Mediante el diseñador Drag and Drop , se puede definir el API de cada micro servicio asi como las llamadas a eventos, datos, streams u otros servicios


Paleta con tablas, servicios, mensajes ....

# Editor de Micro Servicios y API

Al editar un servicio, podremos personalizar sus operaciones, seguridad, schemas, documentación ... asi como incluir validadores y el mapeo de valores del schema hacia el modelo de datos, eventos, llamadas a servicios externos ... El proceso de generación generara el codigo completamente operativo según la definición del servicio y sus operaciones.

Designer

Editor



Service

Paths

Schema

Operation

Schema Json

Documentation

Schema Security

Enviroments

Tags

Service Suinsit 3.0.2

Service	miServicio	Url	<input type="text" value="/miServicio"/>
Version	1.0.0	Title	<input type="text" value="miServicio"/>
Terminos URL	<input type="text" value="terminos"/>	Contact	<input type="text" value="info@suinsit.io"/>
Licencia	<input type="text" value="EUPL"/>	Url	<input type="text" value="https://licence.suinsit.io"/>
Open API Version	3.0.2	Solo Autorizados	<input checked="" type="checkbox"/>

Summary

Operations

Path/Operation	
<input type="button" value="GET"/> newOperation2	<input checked="" type="checkbox"/>
<input type="button" value="PUT"/> newOperation5	<input checked="" type="checkbox"/>
<input type="button" value="DELETE"/> newOperation4	<input checked="" type="checkbox"/>
<input type="button" value="MESSAGE"/> newQueue4	<input checked="" type="checkbox"/>
<input type="button" value="POST"/> newOperation3	<input checked="" type="checkbox"/>

# Diseño modelo datos, mensajes y streams

El diseño de tablas y vistas del modelo de datos se realiza de forma visual , incluyendo las relaciones ascendientes y descendientes , asi como validaciones en tipos de campos, comportamientos .... Todo el diseño se realiza en tiempo real conectado contra una base de datos que permite crear interfaces dinámicas o importación desde ficheros csv.

Configuration

Play Application

Screens

microservices.message

microservices.streams

microservices.api

Data Base

bbdd.data

Model

Table

Views

bbdd.export

Deploy

1

1

3

2

2

2

3

2

3

1

Home / Applications / usuarios / 1.0.0 / model

Design Table

Fields

Text

RichText

Integer

Number

Currency

Yes/No

Date

Time

Timestamp

File

Image

Encrypt

Password

List-String

List-Number

Radio-Text

Table name: USUARIO

is System ☐

Auditar ☐

Crear Historico ☐

Fields

Relations

Sub tables

Documentation

Add fields, drag and drop field here

Type	Name	Unique	Primary key	Size	Required	Criteria	Increment	
Text	ALTA	<input type="checkbox"/>	x		<input type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	EMAIL	<input checked="" type="checkbox"/>	x	100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	FIRSTNAME	<input type="checkbox"/>	x	100	<input type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	IDXUSUARIO	<input type="checkbox"/>	✓		<input type="checkbox"/>	<input type="checkbox"/>		
Text	LASTLOGIN	<input type="checkbox"/>	x		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	LASTNAME	<input type="checkbox"/>	x	100	<input type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	LOCKED	<input type="checkbox"/>	x		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	PASSWORD	<input type="checkbox"/>	x		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	x	
Text	USERNAME	<input checked="" type="checkbox"/>	x	25	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	x	

Properties/Entities

Properties

Entities

Property	Value
Label	email
Description	
Type	Text
Roles Edit	
Roles View	





# Diseño modelo datos, mensajes y streams


La interface permite definir las colas de mensajes y su tipología (Mensajes y Streams) conectados directamente contra RabbitMQ y Confluent Kafka.


Además de definir las colas, topics... también permite definir el schema del mensaje o streams (atributos, valores por defecto, tipos de campo...) similar a como se diseña una tabla de datos.


Pendiente añadir editor AsyncAPI 2.0


 Configuration


 Play Application


 Screens


 microservices.message


 microservices.message.topics

 microservices.message.schema

 microservices.streams

 microservices.api

 Data Base

 Deploy

1

1

3

2

2



2













3

1

Home / Applications / usuarios / 1.0.0 / model

Events - Topics



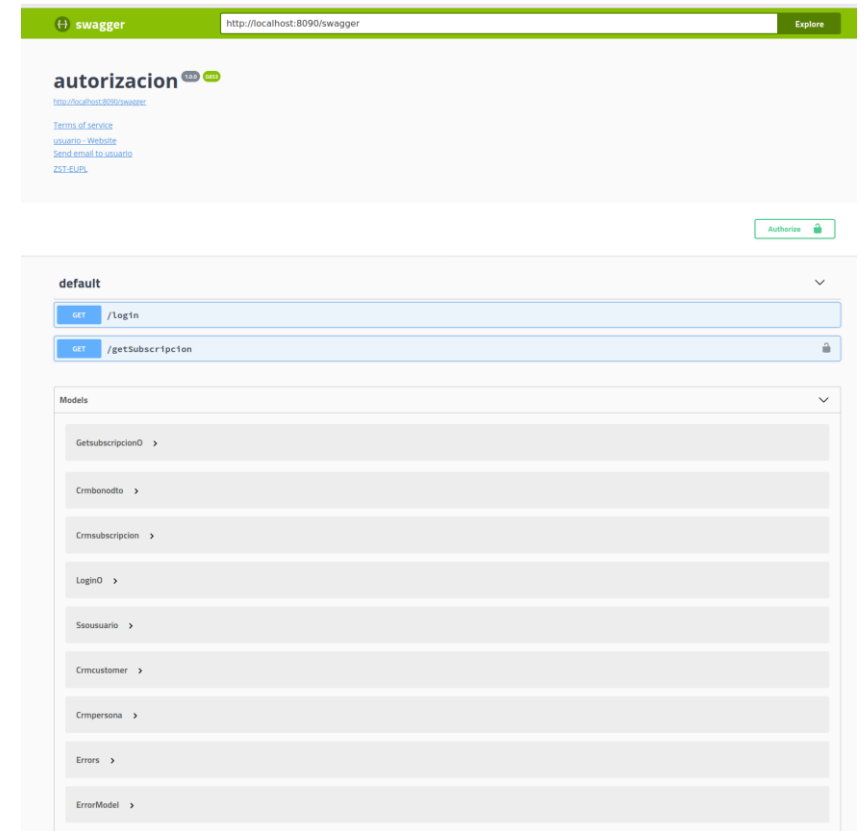
Name	Type	application owner	Date	
io.suinist.platform.amqp.fanout.exchange	FANOUTEXCHANGE	usuarios	Aug 13, 2020, 8:59:34 PM	 
io.suinist.platform.amqp.fanout.queue1	FANOUT	usuarios	Aug 13, 2020, 8:59:34 PM	 
io.suinist.platform.amqp.fanout.queue2	FANOUT	usuarios	Aug 13, 2020, 8:59:34 PM	 
io.suinist.platform.amqp.topic.exchange	TOPICEXCHANGE	usuarios	Aug 13, 2020, 8:49:09 PM	 
io.suinist.platform.amqp.topic.queue1	TOPIC	usuarios	Aug 13, 2020, 8:49:08 PM	 
io.suinist.platform.amqp.topic.queue2	TOPIC	usuarios	Aug 13, 2020, 8:49:09 PM	 

```
MainApp.java | suinsit.vertx.oa3/pom.xml | autorizacion.yml | WebSecure.java | HttpServerVerticle.java | OpenApiServer.java
6 import org.enartframework.vertx.annotation.OpenAPI;
7 import org.enartframework.vertx.annotation.WebSecure;
8 import org.enartframework.vertx.annotation.WebServer;
9 import org.enartframework.vertx.web.HttpServerVerticle;
10 import io.suinsit.vertx.oa3.service.IAutorizacionJS;
11 import io.vertx.core.Future;
12 import io.vertx.core.Vertx;
13 import io.vertx.core.json.JsonObject;
14
15 /**
16  * @author manuel
17  */
18
19 @OpenAPI(showAPI = true, urlYml = "autorizacion.yml", proxyService = IAutorizacionJS.class)
20 @Configuration(patternGit = "miapp")
21 @WebServer(host = "localhost", port = 8090)
22 @WebSecure(auth = WebSecure.AUTHPROVIDER.JWT, publicPaths = {"/login"})
23 public class MainApp extends HttpServerVerticle {
24     JsonObject config;
25
26     @Override
27     public void iniciar(Future<Void> startFuture) throws Exception {
28         vertx.eventBus().consumer(CONFIG_QUEUE, handler->{
29             config = (JsonObject) handler.body();
30         });
31     }
32
33 }
```

Markers | Properties | Data Source Explorer | Snippets | Console | Progress | Search | Git Staging | Terminal | Console | Boot Dashboard

suinsit.vertx.oa3 [Maven Build]

```
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 13 source files to /home/manuel/discovery/suinsit.vertx.oa3/target/classes
[INFO]
[INFO] --- exec-maven-plugin:1.5.0:java (default-cli) @ suinsit.vertx.oa3 ---
[ERROR] StatusLogger No Log4j 2 configuration file found. Using default configuration (logging only errors to the console), or user programmatically
RouteImpl@942926300{state=RouteState{path='null', order=1, enabled=true, methods=null, consumes=null, emptyBodyPermittedWithConsumes=false,
RouteImpl@218763693{state=RouteState{path='/login', order=2, enabled=true, methods=[GET], consumes=null, emptyBodyPermittedWithConsumes=false,
RouteImpl@1315738783{state=RouteState{path='/getSubscription', order=3, enabled=true, methods=[GET], consumes=null, emptyBodyPermittedWithConsumes=false,
RouteImpl@2003219475{state=RouteState{path='/swagger', order=4, enabled=true, methods=[GET], consumes=null, emptyBodyPermittedWithConsumes=false,
RouteImpl@1756170481{state=RouteState{path='/api/', order=5, enabled=true, methods=null, consumes=null, emptyBodyPermittedWithConsumes=false}
```



Mediante anotaciones java simplifica el desarrollo de microservicios , mostrando el api , tokens con jwt, configuración de diferentes entornos en repositorios git ...



```
AutorizacionImpl.java  IAutorizacionJS.java
8 import io.swagger.annotations.ApiResponse;
9 import io.swagger.annotations.ApiResponses;
10 import io.vertx.core.AsyncResult;
11 import io.vertx.core.Handler;
12 import io.vertx.ext.web.api.OperationRequest;
13 import io.vertx.ext.web.api.OperationResponse;
14 import io.vertx.ext.web.api.generator.WebApiServiceGen;
15
16 @WebApiServiceGen
17 @ProxyCall(nameService = "transactions_manager.myapp", implement = AutorizacionImpl.class)
18 public interface IAutorizacionJS {
19
20
21 @ApiResponses(value = { @ApiResponse(code = 200, message = "OK", response = Login0.class),
22 @ApiResponse(code = 201, message = "Created", @ApiResponse(code = 202, message = "Accepted"),
23 @ApiResponse(code = 400, message = "Application Error"),
24 @ApiResponse(code = 401, message = "Unauthorized", response = Errors.class) })
25 public void login(String erpassword, String txusername, OperationRequest context,
26 Handler<AsyncResult<OperationResponse>> resultHandler);
27
28 @ApiResponses(value = { @ApiResponse(code = 200, message = "OK", response = Getsubscripcion0.class),
29 @ApiResponse(code = 201, message = "Created", @ApiResponse(code = 202, message = "Accepted"),
30 @ApiResponse(code = 400, message = "Application Error"),
31 @ApiResponse(code = 401, message = "Unauthorized", response = Errors.class) })
32 public void getSubscripcion(Integer idusuario, OperationRequest context,
33 Handler<AsyncResult<OperationResponse>> resultHandler);
34 }
35
```

```
try {
    JsonObject message = new JsonObject(Mapper.toJSON(login));
    resultHandler.handle(Future.succeededFuture(OperationResponse.completedWithJson(message)));
} catch (JsonProcessingException e) {
    Errors errors = new Errors();
    errors.addErrorsItem(getErrorModel(new DaoException(e)));
    try {
        resultHandler.handle(Future.succeededFuture(
            OperationResponse.completedWithJson(new JsonObject(Mapper.toJSON(errors))).setStatusCode(400).setStatusMessage("A
    } catch (JsonProcessingException e1) {
        resultHandler.handle(Future.succeededFuture(new OperationResponse().setStatusCode(400).setStatusMessage("Application Erro
    }
}
```

Schema de errores personalizado

Definición de la interface del microservicio  
para su generación como servicio de vertex

Anotación de arquitectura

Implementación del servicio

```
*AutorizacionImpl.java  IAutorizacionJS.java
18 import io.vertx.core.Handler;
19 import io.vertx.core.Vertx;
20 import io.vertx.core.json.JsonObject;
21 import io.vertx.ext.auth.JWTOptions;
22 import io.vertx.ext.auth.jwt.JWTAuth;
23 import io.vertx.ext.web.api.OperationRequest;
24 import io.vertx.ext.web.api.OperationResponse;
25
26 public class AutorizacionImpl extends ServiceREST implements IAutorizacionJS{
27     Vertx vertx;
28     String address;
29     public AutorizacionImpl(Vertx vertx) {
30         super();
31         this.vertx = vertx;
32     }
33     public AutorizacionImpl(Vertx vertx, String address) {
34         super();
35         this.vertx = vertx;
36         this.address = address;
37     }
38     @Override
39     public void login(String erpassword, String txusername, OperationRequest context,
40 Handler<AsyncResult<OperationResponse>> resultHandler) {
41         Login0 login = new Login0();
42         login.setSsousuario(new Ssousuario());
43         login.getSsousuario().setErpassword(erpassword);
44         login.getSsousuario().setTxusername(txusername);
45         login.getSsousuario().setIdxsuario(1);
46         login.getSsousuario().setTxemail("soporte@suinsit.io");
47         JWTAuth provider = vertx.getOrCreateContext().get(CONTEXT_KEY.JWT_PROVIDER.name());
48         JsonObject claims = new JsonObject();
49         claims.put("preferred_username", login.getSsousuario().getTxusername());
50         claims.put("email", login.getSsousuario().getTxemail());
51         claims.put("sid", login.getSsousuario().getIdxsuario());
52         JWTOptions options = new JWTOptions();
53         options.addPermission("ADMIN");
54     }
55
```

## Java Bean representación del schema, incluye validación de tipos, fechas, valores, colecciones y enumeraciones



## Definición YAML del micro servicio según open api 3.0.2

```

82 components:
83   securitySchemes:
84     bearerAuth:           # arbitrary name for the security scheme
85       type: http
86       scheme: bearer
87       bearerFormat: JWT
88   schemas:
89     Getsubscripcion0:
90       properties:
91         crmsubscripcion:
92           $ref: '#/components/schemas/Crmsubscripcion'
93     Crmbonodto:
94       required:
95       - idxbonodto
96       properties:
97         rtperiodicidad:
98           type: string
99           description: use this list when... MES=MENSUAL,ANUAL=ANUAL,
100           enum:
101             - MES
102             - ANUAL
103         nuimpdto:
104           format: double
105           type: number
106         idxbonodto:
107           format: int32
108           type: integer
109         txpartner:
110           type: string
111         fxinicio:
112           format: date
113           description: --Undocumented-- validate from VALUES MAX-MIN (TODAY+/-nD)
114
115         type: string
116         txbono:
117         type: string

```

# Arquitectura vertex 3.9.2

Framework específico basado en anotaciones para vertex 3 X

Mensajería basada en RabbitMQ

- > enartframework.web.vertx [enart master]
- > src/main/java
  - > org.enartframework.vertx
    - Configuration.java
    - KafkaListener.java
    - OpenAPI.java
    - package-info.java
    - ProxyCall.java
    - RabbitAMQP.java**
    - RabbitDeclare.java
    - RabbitListener.java
    - WebSecure.java
    - WebServer.java
  - > org.enartframework.vertx.auth
    - JWTAuthHandler.java
    - JWTAuthHandlerExt.java
    - TypeAuth.java
  - > org.enartframework.vertx.messaging
    - package-info.java
    - RabbitClientVerticle.java
  - > org.enartframework.vertx.web
    - HttpServerVerticle.java
    - OpenApiServer.java
    - package-info.java
  - > org.enartframework.web.vertx
  - > org.enartframework.web.vertx.util

Project Explorer

- enartframework
- > enart-context [enart master]
- > enartframework.web.vertx [enart master]
- > enart-message [enart master]
  - > src/main/java
    - > org.suinsitframework.amqp
      - > org.suinsitframework.amqp.model
        - BuilderQueue.java
        - FORMAT.java
        - FormatJSON.java
        - Message.java
        - Queue.java
        - test.java
        - TYPEQUEUE.java
        - TypeQueueJSON.java
      - > org.suinsitframework.amqp.vertx
        - ClientRabbit.java**
        - MessageBuilder.java
        - Testing.java
      - > org.suinsitframework.model
        - Entity.java
        - EnumValues.java
        - Field.java
        - PropertiesBean.java
        - PropertiesBeanJSON.java
        - TYPEFIELD.java
        - 043-documentation.svg

Crmsubscripcion

```
72 client = RabbitMQClient.create(vertex, config);
73 }
74 if(!client.isConnected()) {
75     client.start(resultHandler ->{
76         if(!resultHandler.succeeded()) {
77             resultHandler.cause().printStackTrace();
78         }
79     });
80 }
81 }
82 private void bindQueueExchange(TYPEQUEUE type, Queue queue, final IAmqListener listener) throws AmqpException {
83     for (Queue q : queue.getVinculados()) {
84         if (q.getType().equals(type)) {
85             client.queueDeclare(q.getName(), q.isDurable(), q.isExclusive(), q.isAutodelete(), result->{
86                 if (result.succeeded()) {
87                     if (listener!=null) {
88                         listener.onDeclare(q.getName(), true);
89                     }
90                 } else {
91                     if (listener!=null) {
92                         listener.onError(new Event("exchangeDeclare:"+q.getName(), new AmqpException(result.cause())));
93                     }
94                     log.error(result.cause());
95                 }
96             });
97         }
98     }
99     for(String s:q.getPatterns()) {
100         client.queueBind(q.getName(), queue.getName(), s, result->{
101             if (result.succeeded()) {
102                 if (listener!=null) {
103                     listener.onDeclare(s, true);
104                 }
105                 log.info(queue.getName() + "Exchange successfully with keyPatterns "+s);
106             } else {
107                 if (listener!=null) {
108                     listener.onError(new Event("exchangeDeclare:"+s, new AmqpException(result.cause())));
109                 }
110             }
111         });
112     }
113 }
```

Outline

- org.suinsitframework.amqp.vertx
  - ClientRabbit
    - log: IEnartLogger
    - serialVersionUID: long
    - client: RabbitMQClient
    - config: RabbitMQOptions
    - configuration: Configuration
    - modelMapper: ModelMapper
    - vertex: Vertex
    - ClientRabbit(Vertex)
    - ClientRabbit(Vertex, Configuration)
    - bindQueueExchange(TYPEQUEUE, Queue, IAmqListener)
    - connect(): void
    - consume(Queue, IAmqListener): void
    - declareExchange(Queue, IAmqListener): void
    - declareQueue(Queue, IAmqListener): void
    - deleteExchange(Queue, boolean, IAmqListener): void
    - deleteQueue(Queue, IAmqListener): void
    - disconnect(): void
    - mappingConfiguration(): void
    - publish(Queue, IAmqListener): void
    - sendDirect(String, String, Message, IAmqListener)

Pendiente autogeneración con AsyncAPI 2.0