Configuración de Market

Requerimientos Mínimos Versiones

Node:v12.4.0
Npm: 6.14.5
Mysql: 8.0.20
Angular CLI: 9.1.7
Ionic: 6.4.0

API: Enode-Api

Repositorio: https://bitbucket.org/galvintec/enode-api/src/master/
Rama: release-prod
Bd market_db: subir el sql market_db (Esta Bd esta sincronizada y lista para usar) -> esta es la mima bd que se usa Krack-Services
Bd marketplace_login: subir el sql marketplace_login (Esta Bd esta sincronizada y lista para usar)

.env necesitado

```
#
# APPLICATION
#
APP_NAME=spurtCommerce
APP_SCHEMA=http
APP_HOST=127.0.0.1
APP_PORT=8500
APP_ROUTE_PREFIX=/api
APP_BANNER=true
#
# LOGGING
#
LOG_LEVEL=debug
LOG_OUTPUT=dev
#
# MySQL DATABASE
#
TYPEORM_CONNECTION=mysql
TYPEORM_HOST=localhost
TYPEORM_PORT=3306
```

```
TYPEORM_USERNAME=xxxx
TYPEORM_PASSWORD=xxxx
TYPEORM_DATABASE=market_db
TYPEORM_SYNCHRONIZE=false
TYPEORM_LOGGING=["query", "error"]
TYPEORM_LOGGER=advanced-console
#
# GRPC CONFIGURATION
#
GRPC_PORT=14000
#
# MARKETPLACES CONFIGURATION
#
## ZALANDO
ZALANDO_KRACK_MERCHANTS_ID=6ec6dc57-fd95-48c9-b7d4-d682ff582f7f
ZALANDO_CHANNEL_ID=bf48ba35-149d-4b76-8ac9-d08d126b517f
### ZALANDO SANDBOX
ZALANDO_SB_CLIENT_ID=37db6a5895a8332b172095f17ac24ee4
ZALANDO_SB_CLIENT_SECRET=bde26bf9-b0a0-4c1f-ae75-fb88e63004a5
ZALANDO_SB_BASE_URL=https://api-sandbox.merchants.zalando.com
### ZALANDO PROD
ZALANDO_CLIENT_ID=73dbe1647e1890cf0009c361bdfbaf84
ZALANDO_CLIENT_SECRET=bbcec26c-fc84-40dd-a3c3-7ba5e02e7bed
ZALANDO_BASE_URL=https://api.merchants.zalando.com
#ZALANDO_CLIENT_ID=37db6a5895a8332b172095f17ac24ee4
#ZALANDO_CLIENT_SECRET=bde26bf9-b0a0-4c1f-ae75-fb88e63004a5
#ZALANDO_BASE_URL=https://api-sandbox.merchants.zalando.com
## CDISCOUNT
CDISCOUNT_CLIENT_ID=krackonline-api
CDISCOUNT_CLIENT_SECRET=jfr89ej93H(U
CDISCOUNT_PREPROD_AUTH_BASE_URL=https://sts.cdiscount.com
CDISCOUNT_PREPROD_API_BASE_URL=https://wsvc.cdiscount.com
##Authentication service in production : https://sts.cdiscount.com
##API in production : https://wsvc.cdiscount.com
##Sellershop in production : https://seller.cdiscount.com
##Authentication service in preproduction : https://sts.preprod-cdiscount.com
##API in preproduction : https://wsvc.preprod-cdiscount.com
##Sellershop in preproduction : https://seller.preprod-cdiscount.com
GRPC_SSL_INSECURE=true
#
# STORAGE PROVIDERS CONFIGURATION
#
## PRESTASHOP
#PRESTASHOP_BASE_URL=https://preproduccion.krackonline.com/module/enodesync
PRESTASHOP_BASE_URL=https://www.krackonline.com/module/enodesync
## AVELON (example)
```

```
AMAZON_FIELD_A=123
AMAZON_FIELD_B=abc
#
# PATH STRUCTRUE
#
TYPEORM_MIGRATIONS=src/database/migrations/**/*.ts
TYPEORM_MIGRATIONS_DIR=src/database/migrations
TYPEORM_ENTITIES=src/api/models/**/*.ts
TYPEORM_ENTITIES_DIR=src/api/models
CONTROLLERS=src/api/controllers/**/*Controller.ts
MIDDLEWARES=src/api/middlewares/**/*Middleware.ts
INTERCEPTORS=src/api/interceptors/**/*Interceptor.ts
SUBSCRIBERS=src/api/subscribers/**/*Subscriber.ts
RESOLVERS=src/api/resolvers/**/*Resolver.ts
#
# Apidoc
#
APIDOC_ENABLED=true
APIDOC_ROUTE=/apidoc
#
# Status Monitor
#
MONITOR_ENABLED=true
MONITOR_ROUTE=/monitor
MONITOR_USERNAME=admin
MONITOR_PASSWORD=1234
#
# Mail
#
MAIL_DRIVER=smtp
MAIL_HOST=SMTPHOST
MAIL_PORT=465
MAIL_USERNAME=SMTPHOSTUSERNAME
MAIL_PASSWORD=SMTPHOSTPASSWORD
MAIL_ENCRYPTION=tls
MAIL_SECURE=true
MAIL_FROM=no-reply@spurtcommerce.com
#
# File Upload System (local or s3)
#
IMAGE_SERVER = local
#
# AWS S3 Bucket
#
AWS_ACCESS_KEY_ID=AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY=AWS_SECRET_ACCESS_KEY+oaddtUNFNy
```

AWS_DEFAULT_REGION=AWS_DEFAULT_REGION
AWS_BUCKET=AWS_BUCKET

SYNC_TIME_RUNNING_AWAIT=30
SYNC_TIME_SAVING_ON_DB=30
SYNC_RECEIVEGRAPH_TIME=10000
URL_MARKETS_AVELON=http://localhost:3009/api

Datos importantes .env

URL_MARKETS_AVELON -> conecta con el otro micro, se debe tener activo el router, igual que los micros de krack2020, si el router esta activo el puerto seria 8080 pero si no esta activo colocar el puerto del micro

SYNC_TIME_RUNNING_AWAIT -> tiempo de espera en minutos si el micro se daño y para que se borren los registros automáticamente para volver a ejecutar otra sincronización

SYNC_TIME_SAVING_ON_DB -> tiempo en segundos donde se están guardando un registro par saber que se esta ejecutando alguna sincronzacion, esta variable es necesaria para que la anterior pueda funcionar correctamente

SYNC_RECEIVEGRAPH_TIME -> tiempo de espera en segundo para que se ejecuta la sicronzacion de respuesta de zalando, cuando se envía un zapato a zalando el no responde de inmediato esto puede tardar hast 1 minuto solo para decir que recibió, aunque la respuesta real puede tardar desde 5 minutos hast 2 o 3 horas (no existe un tiempo especifico)

Sincronizaciones para el envio de zapatos a zalando

Comando:yarn sync-zalando onboarding2 5 7
Uso: este comando envia los zapatos a zalando siempre y cuando se haya hecho un onboarding exitoso desde el front, las 2 variables (5,7) tienen un significado, la cantidad de veces que se ejecutara ese comando y como un ciclo y cada cuantos segundos lo hara. Este comando debe colocarse en un cronjob del servidor que se ejecute cada 5 minutos para que pueda dar chance a que zalando este preparado para recibir los zapatos

Comando:yarn sync-zalando receive-graph 20 3
Uso: este comando recibe la confirmación de zalando que los zapatos con el comando anterior entraron y trae el estatus o el error de las variaciones, las 2 variables (10,2) tienen un significado,

la cantidad de veces que se ejecutara ese comando como un ciclo y cada cuantos segundos lo hara. Este comando debe colocarse en un cronjob del servidor que se ejecute cada 9 minutos para que pueda dar chance a que zalando pueda procesar los zapatos y mostrarlos en su plataforma Este comando es necesario ya que con eso podemos saber si el zapato ya esta en zalando (solamente si entro el zapato no tiene que ver con los errores posteriores)

Ejecución del codigo

Comando: npm start serve
Notas: este comando debe permanecer prendido en segundo plano, caso contrario no funcionara correctamente con comando como nohup, bg o pm2

Front:**Enode-Admin**

Repositorio: https://bitbucket.org/galvintec/enode-admin/src/master/
Rama: release-prod2

Información de environment

Entorno: Produccion
Ruta: src/environments/environment.prod.ts

Entorno: Desarrollo Local
Ruta: src/environments/environment.ts

Nota: Cambiar las variables interna como ip y puerto según el servidor que van a usar

Ejecución del codigo

Comando: ng serve
Notas: este comando debe permanecer prendido en segundo plano, caso contrario no funcionara correctamente con comando como nohup, bg o pm2, o hacer un ng build —prod y colocar todo el codigo en un servidor apache (esto es lo mas recomendable)

Api:Krack-Services

Repositorio: https://bitbucket.org/galvintec/krack-avelon-services/src/dev/
Rama: market_dev
Bd market_db: subir el sql market_db (Esta Bd esta sincronizada y lista para usar) -> esta es la mima bd que se usa enode-api
Bd marketplace_login: subir el sql marketplace_login (Esta Bd esta sincronizada y lista para usar)

.env necesitado

```
APP_NAME=krack-micro-market-oboarding
NODE_ENV=local
DEBUG=true

#Hydra
SERVICE_NAME=krack-micro-market-oboarding
SERVICE_DESCRIPTION=API Micro Avelon
APP_PORT=3009


TYPEORM_DATABASE=market_db
TYPEORM_USERNAME=xxxx
TYPEORM_PASSWORD=xxxx
TYPEORM_PORT= 3306
TYPEORM_HOST= localhost
TYPEORM_CONNECTION = mysql
TYPEORM_ENTITIES = lib/API/Application/Domain/Entities/*.js
TYPEORM_SYNCHRONIZE = true
TYPEORM_ENTITY_PREFIX=krack_
MIGRATION_FORCE = true

TYPEORM_ENTITIES_SYNC = src/API/Application/Domain/Entities/
TYPEORM_SYNC_OUTPUT = tmp_sync/

SGA_CLIENT_NAME=krack-client-sga
SGA_CLIENT_SECRET=fGx4=yU-j4^jAAjZtV+YTDsm-@R$HAK3
SGA_CLIENT_ENABLED=true
AL_CLIENT_NAME=krack-client-al
AL_CLIENT_SECRET=k4a4yBrqW54L@uX_^p8EMGDFb?qj*TKe
AL_CLIENT_ENABLED=true
```

# Reject duplicated request if occurs in this time window (NanoSeconds)
TIME_BETWEEN_REQUESTS_NS = 100000

Login service
AUTH_SERVICE = krack-auth

TYPEORM_DATABASE_LOGIN= marketplace_login

ZALANDO_SERVER_HOSTNAME = api.merchants.zalando.com
ZALANDO_GALVINTEC_API_PORT= 443
ZALANDO_GALVINTEC_API_TOKEN=
ZALANDO_CLIENTE_ID=73dbe1647e1890cf0009c361bdfbaf84
ZALANDO_SECRET_ID=bbcec26c-fc84-40dd-a3c3-7ba5e02e7bed

PRESTASHOP_SERVER_HOSTNAME = https://www.krackonline.com
PRESTASHOP_GALVINTEC_API_PORT= 443
PRESTASHOP_GALVINTEC_API_TOKEN=

Ejecución del codigo

Comando: yarn start
Notas: este comando debe permanecer prendido en segundo plano, caso contrario no funcionara correctamente con comando como nohup, bg o pm2

`Front:Sga`

Repositorio: https://bitbucket.org/galvintec/sga-app-logistica-krack/src/master/
Rama: market_dev

Información de environment

Entorno: Produccion
Ruta: sga-app-logistica-krack/libs/services/src/environments/environment.prod.ts

Entorno: Desarrollo Local
Ruta: sga-app-logistica-krack/libs/services/src/environments/environment.ts

Nota: Cambiar las variables interna como ip y puerto según el servidor que van a usar

Ejecución del codigo

Comando: ionic serve
Notas: este comando debe permanecer prendido en segundo plano, caso contrario no funcionara correctamente con comando como nohup, bg o pm2, o hacer un ionic build —prod y colocar todo el codigo en un servidor apache (esto es lo mas recomendable)