Documentación

Api Xaxis Disco

Jorge Araya – Ing. Software

Documentación técnica sobre el proyecto Api Xaxis Disco

Tabla de contenido

[1 Descripción 4](#_Toc115742499)

[2 Instalación 4](#_Toc115742500)

[3 Estructura del proyecto 4](#_Toc115742501)

[3.1 src.main.java 4](#_Toc115742502)

[3.1.1 com.xaxis.disco.api 4](#_Toc115742503)

[3.1.1.1 DiscoApplication 4](#_Toc115742504)

[3.1.1.2 adapters 4](#_Toc115742505)

[3.1.1.2.1 inbound.api 4](#_Toc115742506)

[3.1.1.2.1.1 controller 4](#_Toc115742507)

[3.1.1.2.1.1.1 AccountController 4](#_Toc115742508)

[3.1.1.2.1.1.1.1 get 4](#_Toc115742509)

[3.1.1.2.1.1.1.2 sendEmail 5](#_Toc115742510)

[3.1.1.2.1.1.2 AccountControllerDoc 5](#_Toc115742511)

[3.1.1.2.1.1.3 CampaignController 5](#_Toc115742512)

[3.1.1.2.1.1.3.1 get 5](#_Toc115742513)

[3.1.1.2.1.1.4 CampaignControllerDoc 5](#_Toc115742514)

[3.1.1.2.1.1.5 CreativeMatrixController 5](#_Toc115742515)

[3.1.1.2.1.1.5.1 getBy 5](#_Toc115742516)

[3.1.1.2.1.1.6 CreativeMatrixControllerDoc 5](#_Toc115742517)

[3.1.1.2.1.1.7 TestController 5](#_Toc115742518)

[3.1.1.2.1.1.7.1 health 5](#_Toc115742519)

[3.1.1.2.1.2 response 5](#_Toc115742520)

[3.1.1.2.2 outbound 5](#_Toc115742521)

[3.1.1.2.2.1 acl 6](#_Toc115742522)

[3.1.1.2.2.1.1 MetaApi 6](#_Toc115742523)

[3.1.1.2.2.1.1.1 MetaRepositoryImpl 6](#_Toc115742524)

[3.1.1.2.2.1.1.1.1 getPageBy 6](#_Toc115742525)

[3.1.1.2.2.1.1.1.2 getBusinessManagerBy 6](#_Toc115742526)

[3.1.1.2.3 repository 6](#_Toc115742527)

[3.1.1.2.3.1 handlers 6](#_Toc115742528)

[3.1.1.2.3.1.1 UUIDTypeHandler 6](#_Toc115742529)

[3.1.1.2.3.1.2 AccountRepositoryImpl 6](#_Toc115742530)

[3.1.1.2.3.1.2.1 getBy 6](#_Toc115742531)

[3.1.1.2.3.1.2.2 create(Account) 6](#_Toc115742532)

[*3.1.1.2.3.1.2.3* create(AccountPlatform) 7](#_Toc115742533)

[3.1.1.2.3.1.2.4 create(AccountPage) 7](#_Toc115742534)

[*3.1.1.2.3.1.2.5* getAllPlatform 7](#_Toc115742535)

[3.1.1.2.3.1.3 AccountRepositoryMyBatis 7](#_Toc115742536)

[3.1.1.2.3.1.4 CampaignRepositoryImpl 7](#_Toc115742537)

[3.1.1.2.3.1.4.1 get 7](#_Toc115742538)

[3.1.1.2.3.1.5 CampaignRepositoryMyBatis 7](#_Toc115742539)

[3.1.1.2.3.1.6 ClientRepositoryImpl 7](#_Toc115742540)

[3.1.1.2.3.1.6.1 getBy 7](#_Toc115742541)

[3.1.1.2.3.1.7 ClientRepositoryMyBatis 7](#_Toc115742542)

[3.1.1.2.3.1.8 CreativeMatrixRepositoryImpl 7](#_Toc115742543)

[3.1.1.2.3.1.8.1 get: 7](#_Toc115742544)

[3.1.1.2.3.1.9 CreativeMatrixRepositoryMyBatis 8](#_Toc115742545)

[3.1.1.3 domain 8](#_Toc115742546)

[3.1.1.3.1 config 8](#_Toc115742547)

[3.1.1.3.1.1 AmazonCorsBeanConfig 8](#_Toc115742548)

[3.1.1.3.1.2 AmazonSESBeanConfig 8](#_Toc115742549)

[3.1.1.3.2 model 8](#_Toc115742550)

[3.1.1.3.3 repository 8](#_Toc115742551)

[3.1.1.3.4 service 8](#_Toc115742552)

[3.1.1.3.4.1 platform 9](#_Toc115742553)

[3.1.1.3.4.1.1 MetaService 9](#_Toc115742554)

[3.1.1.3.4.1.1.1 getPages 9](#_Toc115742555)

[3.1.1.3.4.1.1.2 getBusiness 9](#_Toc115742556)

[3.1.1.3.4.2 AccountService 9](#_Toc115742557)

[3.1.1.3.4.2.1 sendMail 9](#_Toc115742558)

[3.1.1.3.4.2.2 get 9](#_Toc115742559)

[3.1.1.3.4.3 CampaignService 9](#_Toc115742560)

[3.1.1.3.4.3.1 get 10](#_Toc115742561)

[3.1.1.3.4.4 CreativeMatrixService 10](#_Toc115742562)

[3.1.1.3.4.4.1 get 10](#_Toc115742563)

[3.2 src.main.resources 10](#_Toc115742564)

[3.2.1 Mybatis 10](#_Toc115742565)

[3.2.1.1 AccountMapper.xml 10](#_Toc115742566)

[3.2.1.2 CampaignMapper.xml 10](#_Toc115742567)

[3.2.1.3 ClientMapper.xml 10](#_Toc115742568)

[3.2.1.4 CreativeMatrixMapper.xml 10](#_Toc115742569)

[3.2.2 application.yml 10](#_Toc115742570)

[3.2.3 application-dev.yml 11](#_Toc115742571)

[3.3 src.test.java 11](#_Toc115742572)

[3.3.1 adapters.inbound.api.controller 11](#_Toc115742573)

[3.3.2 adapters.outbound.acl 11](#_Toc115742574)

[3.3.3 adapters.outbound.repository 11](#_Toc115742575)

[3.3.4 domain.config 11](#_Toc115742576)

[3.3.5 domain.service 11](#_Toc115742577)

[3.3.6 util 11](#_Toc115742578)

[3.4 src.test.resources 11](#_Toc115742579)

[4 Base de datos 12](#_Toc115742580)

[4.1.1 Motor de base de datos 12](#_Toc115742581)

[4.1.2 Configuración Base de datos 12](#_Toc115742582)

[4.1.3 Diagrama MER (modelo entidad relación) 12](#_Toc115742583)

[4.1.4 Flyway – Scripts para creación de tablas en base de datos 12](#_Toc115742584)

[5 Test 13](#_Toc115742585)

[6 Build & Deploy 13](#_Toc115742586)

# Descripción

Proyecto Api backend, que se conecta con api de *Meta*. Está construido bajo una arquitectura Hexagonal, desarrollado con el framework *Spring Boot* en *Java 11*, con base de datos *Postgresql 13*, *Mybatis*, *Flyway, AWS SES*, *Openfeign* y gestor de dependencias *Gradle*.

# Instalación

Al ser un proyecto desarrollado en Java, podrá ser abierto solo con un IDE compatible con *Java*, como por ejemplo *Eclipse*, *Netbeans* o ***IntelliJ*** (recomendado). Deberán instalar la versión 11 de J*ava*; esta puede ser descargada desde [aquí](https://www.oracle.com/cl/java/technologies/javase/jdk11-archive-downloads.html).

# Estructura del proyecto

## src.main.java

Carpeta contenedora del código fuente de la aplicación.

### com.xaxis.disco.api

#### DiscoApplication

Clase que contiene el método main con el cual se da inicio a la Api.

#### adapters

Carpeta que contiene las clases con las cuales la Api se comunica, ya sea hacia afuera como hacia dentro.

##### inbound.api

Carpeta que contiene los controladores(controller) y las entidades de respuestas(response); también podría contener las entidades de entrada (request).

###### controller

Carpeta que contiene los controladores por donde la Api expone los endpoints

AccountController

Clase controller que implementa la interfaz *AccountControllerDoc* con dos métodos:

get

Método get que retorna un Account, con todos los valores necesario para que el frontend los muestre, internamente se conecta con api Meta para obtener los datos del business manager, page id y page name.

sendEmail

Método post que envía un email al correo especificado en su parámetro de entrada to (de tipo String).

AccountControllerDoc

Interface que contiene las especificaciones de Swagger para la documentación de los endpoint de AccountController.

CampaignController

Clase controller que implementa la interfaz *CampaignControllerDoc* con un método:

get

Método get que obtiene una lista de campañas almacenadas en la base de datos.

CampaignControllerDoc

Interface que contiene las especificaciones de Swagger para la documentación de los endpoint de CampaignController.

CreativeMatrixController

Clase controller que implementa la interfaz *CreativeMatrixControllerDoc* con un método:

getBy

Método get que obtiene entidades del tipo creativeMatrix, que se muestran en el módulo de *Creative Matrix* (en el frontend). Son obtenidas desde la base de datos.

CreativeMatrixControllerDoc

Interface que contiene las especificaciones de Swagger para la documentación de los endpoint de CreativeMatrixController.

TestController

Clase controller que implementa la interfaz Test*ControllerDoc* con un método:

health

Método get que retorna un texto, para saber si la Api está ok.

###### response

Carpeta contenedora de clases DTO (data transfer object) utilizadas para las respuestas de los endpoint.

##### outbound

Carpeta contenedora de las clases que se utilizan para comunicar a la Api con otras Apis (acl) o con un repositorio de base de datos (repository).

###### acl

Carpeta contenedora de las clases que se utilizan para comunicar a la Api con otras Apis externas como Meta o TikTok,etc.

MetaApi

Interface que cuenta con dos métodos utilizados para obtener información desde la Api de Meta, donde por medio de FeignClient se realizan las peticiones.

MetaRepositoryImpl

Clase que implementa los métodos de la inteface *MetaApi*:

getPageBy

Método para obtener las páginas asociadas a una cuenta, recibe por parámetros el accountId (ide la cuenta de Facebook) y el token generado del usuario (este puede obtenido a través del [Explorador de la API Graph - Facebook for Developers](https://developers.facebook.com/tools/explorer)).

getBusinessManagerBy

Método para obtener el business manager asociado la cuenta, recibe por parámetros accountId (ide la cuenta de Facebook) y el token generado del usuario (este puede obtenido a través del [Explorador de la API Graph - Facebook for Developers](https://developers.facebook.com/tools/explorer)).

##### repository

Carpeta contenedora de las clases relacionadas con la base de datos

###### handlers

Carpeta contenedora de tipos no nativos de MyBatis y que deben se especificados

UUIDTypeHandler

Clase con la especificación de MyBatis para el tipo de dato UUID, de manera que pueda ser utilizado en los mappers de las entidades especificadas en los archivos ubicados en “*/resources/mybatis/\*.xml”.*

AccountRepositoryImpl

Clase que implementa la interface *AccountRepository*, la cual cuenta con cinco métodos:

getBy

Método que obtiene una lista de cuentas por un identificador de cliente de tipo UUID de la tabla *account*.

create(Account)

Método para crear un nuevo registro en la tabla *account*.

create(AccountPlatform)

Método para crear un nuevo registro en la tabla *account\_platform.*

create(AccountPage)

Método para crear un nuevo registro en la tabla account\_page.

getAllPlatform

Método que obtiene todas las plataformas registradas en la tabla *platform.*

AccountRepositoryMyBatis

Interface que se relaciona con el archivo “*/resources/mybatis/AccountMapper.xml”* donde se especifican los métodos que contiene la interface, por medio de los cuales se realizan las consultas relacionadas a la entidad y tabla *account* en la base de datos.

CampaignRepositoryImpl

Clase que implementa la interface *CampaignRepository*, la cual cuenta con un método:

get

método que retorna una lista de campaign registradas en la tabla campaign desde la base de datos.

CampaignRepositoryMyBatis

Interface que se relaciona con el archivo “*/resources/mybatis/CampaignMapper.xml”* donde se especifican los métodos que contiene la interface, por medio de los cuales se realizan las consultas relacionadas a la entidad y tabla *campaign* en la base de datos.

ClientRepositoryImpl

Clase que implementa la interface C*lientRepository*, la cual cuenta con un método:

getBy

método que obtiene un cliente desde la tabla *client* y recibe por parámetro el identificador del cliente como UUID. Retorna un cliente encapsulado en una clase *Optional* para el caso en el que el cliente no exista.

ClientRepositoryMyBatis

Interface que se relaciona con el archivo “*/resources/mybatis/ClientMapper.xml”* donde se especifican los métodos que contiene la interface, por medio de los cuales se realizan las consultas relacionadas a la entidad y tabla *client* en la base de datos.

CreativeMatrixRepositoryImpl

Clase que implementa la interface *CreativeMatrixRepository*, la cual cuenta con un método:

get:

Método que obtiene una lista de creative matrix desde la tabla *creativeMatrix* almacenadas en la base de datos.

CreativeMatrixRepositoryMyBatis

Interface que se relaciona con el archivo “*/resources/mybatis/CreativeMatrixMapper.xml”* donde se especifican los métodos que contiene la interface, por medio de los cuales se realizan las consultas relacionadas a la entidad y tabla *creative\_matrix* en la base de datos.

#### domain

Carpeta que contiene las clases que componen el dominio de la Api, aquí podemos encontrar los modelos, los servicios, la lógica de negocio, la configuración de los *bean*, etc.

##### config

Carpeta que contiene las clases con la configuración de los *bean*.

###### AmazonCorsBeanConfig

Clase que contiene la configuración del *cors*, actualmente la Api está respondiendo a cualquier origen, mientras se define el origen definitivo (la url desde donde el frontend emite los request a la Api), se debe especificar aquí en el único método que contiene la clase *addCorsMappings.*

###### AmazonSESBeanConfig

Clase que contiene la configuración para el envio de correo a través de AWS SES (Amazon simple email send). Esta clase tiene dos atributos: *accessKey* y *secretKey*; la configuración de estos valores están registrados en el archivo de configuración de la Api ”/*resources/application.yml”.* El Tanto el *accessKey* como el *secretKey*, deben ser generados desde la consola web de la cuenta de AWS. Para esto pueden consultar la documentación de AWS, como obtener un *accessKey* y un *secretKey*.

##### model

Carpeta que contiene las clases con la definición de los modelos correspondientes a las entidades de la base de datos y apis externas (son solo clases DTO sin lógica).

##### repository

Carpeta que contiene las interfaces con la definición de los métodos que son utilizados por los servicios y que deben ser implementados ya sea por un acl o por una implementación de MyBatis. Al ser interfaces carecen de lógica ya que solo son definiciones(contratos) de métodos.

##### service

Carpeta que contiene las clases con los servicios. En los servicios es donde se encuentra y debe encontrarse toda la lógica de negocio.

###### platform

Carpeta que contiene las clases con los servicios de las plataformas, como por ejemplo Meta.

MetaService

Clase de tipo de servicio, que contiene la lógica de negocio para realizar las consultas a la Api de Meta. Esta clase tiene tres atributos como parámetros de configuración;appSecret, accessToken y accountId, cuya asignación la podremos encontrar en el archivo de configuración de la Api *”/resources/application.yml*” respectivamente. La definición de los valores de estos atributos deben ser obtenidos desde Meta, tanto el *appSecret* como el *accountId*, mientras que el *accesToken* puede ser generado a través del [Explorador de la API Graph - Facebook for Developers](https://developers.facebook.com/tools/explorer). Este servicio cuenta con dos métodos

getPages

Método para obtener las páginas asociadas a una cuenta.

getBusiness

Método para obtener business manager asociados a una cuenta.

###### AccountService

Clase de tipo servicio, que contiene la lógica de negocio para la entidad *account*. En esta clase podemos encontrar dos métodos *sendMail* y *get*:

sendMail

Método que envía un email a la dirección de correo especificada en su parámetro de entrada. Para enviar email se utiliza el SDK que proporciona AWS SES para el envío de correo electrónico, donde se especifica el asunto (*SUBJECT*), emisor (.*withSource*), destinatario (.*withToAddresses*) y contenido del mensaje(*.withData*).

get

Método que va a Meta a obtener las páginas y el business manager asociadas a una cuenta y se mapean junto con datos *mock* para complementar la entidad *Account*,con el resto de los datos que necesita el frontend y que son desplegados en el módulo *Account Setup*.

###### CampaignService

Clase de tipo servicio, que contiene la lógica de negocio para la entidad *campaign*. En esta clase podemos encontrar un método *get.*

get

Método que va a la base de datos a buscar las campañas registras en la tabla *campaign*. Para esto utiliza el método get del repositorio de *compaignRepository.*

###### CreativeMatrixService

Clase de tipo servicio, que contiene la lógica de negocio para la entidad *creativeMatrix*. En esta clase podemos encontrar un método *get.*

get

Método que va a la base de datos a buscar creative matrix registras en la tabla *creative\_matrix*. Para esto utiliza el método get del repositorio de *CreativeMatrixRepository.*

## src.main.resources

Capeta de recursos, donde podemos encontrar archivos configuración de la Api y mappers de Mybatis.

### Mybatis

Carpeta que contiene los mappers de Mybatis con las querys para la base de datos.

#### AccountMapper.xml

Archivo que contiene las querys necesarias para persistir un obtener una entidad Account desde la base de datos.

#### CampaignMapper.xml

Archivo que contiene las querys necesarias para persistir un obtener una entidad Campaign desde la base de datos.

#### ClientMapper.xml

Archivo que contiene las querys necesarias para persistir un obtener una entidad Client desde la base de datos.

#### CreativeMatrixMapper.xml

Archivo que contiene las querys necesarias para persistir un obtener una entidad CreativeMatrix desde la base de datos.

### application.yml

Archivo que contiene las configuraciones base la Api, en modo productivo. En este lugar es donde se configura la conexión a la base de datos, el puerto (5000) por donde será expuesta la Api, el path (/disco/api), credenciales de AWS SES (para el envío de correos a través de AWS), credenciales de Meta; como accessToken, appSecret, accountId y url de api de Meta.

### application-dev.yml

archivo que contiene solo las configuraciones que se sobreescriben cuando ejecutan la Api en modo debug.

## src.test.java

En esta carpeta se encuentran todos los test para los distintos componentes de la Api, como los controladores, los servicios, los *bean* deconfiguración, etc.

### adapters.inbound.api.controller

En esta carpeta se encuentran los test relacionados a los controladores, donde para cada controlador existe una clase con tests, para probar los endpoint que expone el controlador.

### adapters.outbound.acl

En esta carpeta se encuentran los test relacionados a los acl de la Api, para testear la implementación de sus métodos.

### adapters.outbound.repository

En esta carpeta se encuentran los test relacionados a los repositorios de la Api, para testear la implementación de myBatis y de los tipos de datos especiales(*handlers*).

### domain.config

En esta carpeta se encuentran los test relacionados a la configuración e inicialización de los *bean* de la Api.

### domain.service

En esta carpeta se encuentran los test relacionados a los servicios, para testear la lógica de negocio.

### util

En esta carpeta se encuentran las clases con los *mocks,* que son utilizados los test para probar los métodos, generando objetos personalizados.

## src.test.resources

En esta carpeta se encuentran los archivos con configuraciones especiales para la ejecución de los test. En esta carpeta se pueden incluir otros tipos de recursos, siempre relacionados a los test.

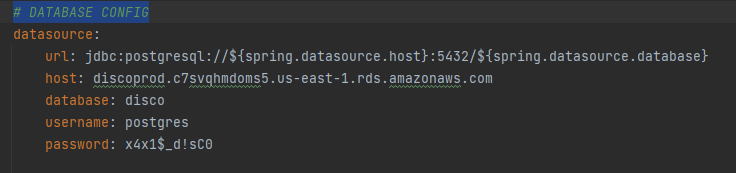
# Base de datos

### Motor de base de datos

El motor de datos que usa la Api es Postgresql version 13.

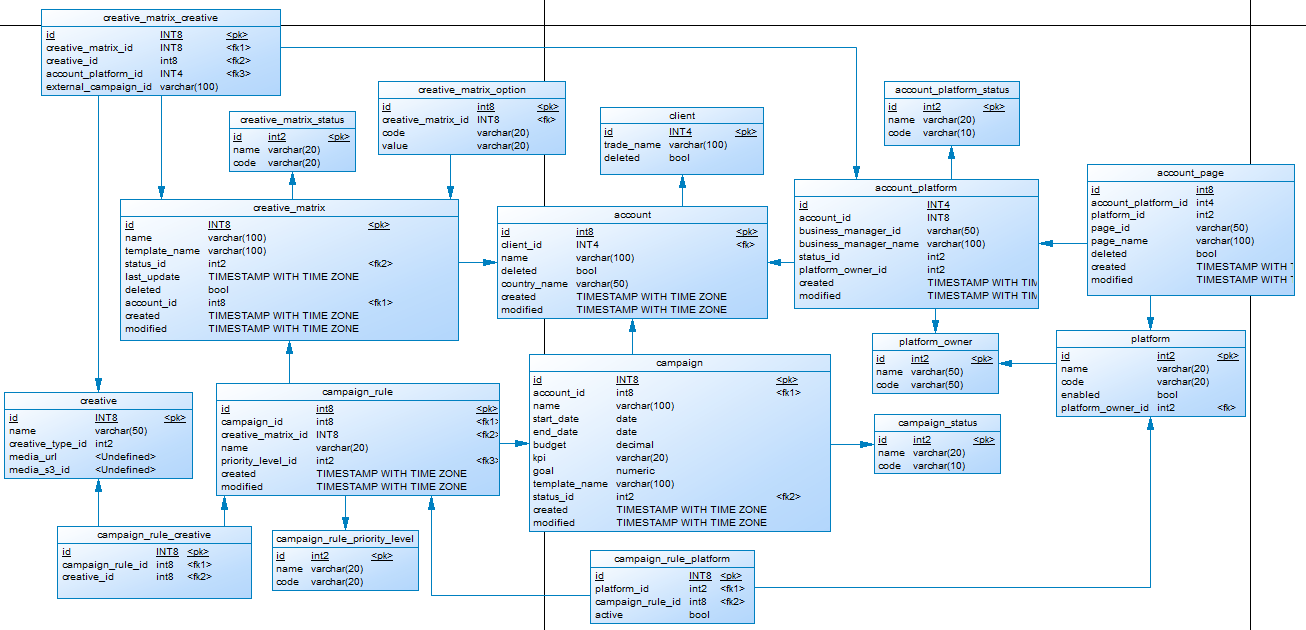
### Configuración Base de datos

La configuración de la conexión a la base de datos, la pueden encontrar en el archivo *“src/main/resources/application.yml”,* en la sección *# DATABASE CONFIG* , donde se debe especificar el *host,database(*nombre de la base de datos*), username, password.*



### Diagrama MER (modelo entidad relación)

Para graficar como se relacionan las entidades en la base de datos. El archivo original fue generado con el Software *Sybase PowerDesigner version 16.* Este archivo lo pueden encontrar en la carpeta *“/resources/diagram/xaxis\_disco-pdb”.*



### Flyway – Scripts para creación de tablas en base de datos

Para el versionamiento de los scripts de sql, para la creación o modificación de las tablas que componen la base de datos, la Api implementa Flyway. Los scripts con la definición de las tablas, sus relaciones y datos iniciales, se pueden encontrar en “/resources/database/”. Para la ejecución de estos scripts se utiliza Flyway, con los comandos a través de gradle *migrate.* La conexión de Flyway a la base de datos está configurada al final del archivo “*build.gradle”* y debiese tener los mismos parámetros de configuración encontrados en el archivo *“src/main/resources/application.yml”.*

# Test

Para la ejecución de los test y comprobar la consistencia de la Api, se debe ejecutar el comando de gradle “*check”*, el cual ejecutará todos los test de la Api informando los Test fallidos.

# Build & Deploy

Para compilar la Api y generar un archivo *.jar*, se debe ejecutar el comando de gradle “*build*”, el cual compilará la Api y dejará el archivo .jar en la siguiente ruta */build/libs/.* Cabe mencionar que la ejecución de este comando, antes de generar la compilación **ejecuta todos los test,** en caso de que uno falle, lanzará un error de compilación; por lo tanto, antes de generar el build, debemos asesorarnos de que todos los test resultaron exitosos. Una vez generado el archivo *.jar*, este puede ser desplegado en un servidor de aplicaciones o eventualmente en *AWS Elastic Beanstalk.*