MASTER CLOUD APPS

TRABAJO FIN DE MASTER

APLICACIÓN DE PLANIFICACIÓN DE REDES SOCIALES CON TBD Y CI/CD

Alumnos:
Andrea Colina
Juan Manuel Guijarro

Tutor: Francisco Gortázar

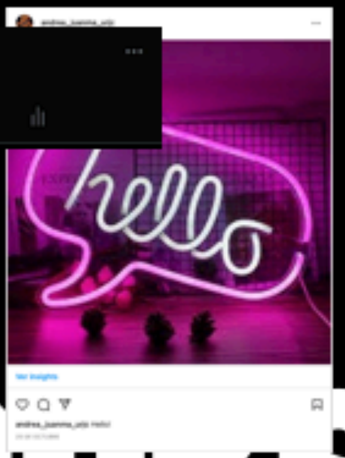
ÍNDICE

QUE ES TBD-SOCIAL-NETWORKS-PLANNER ?

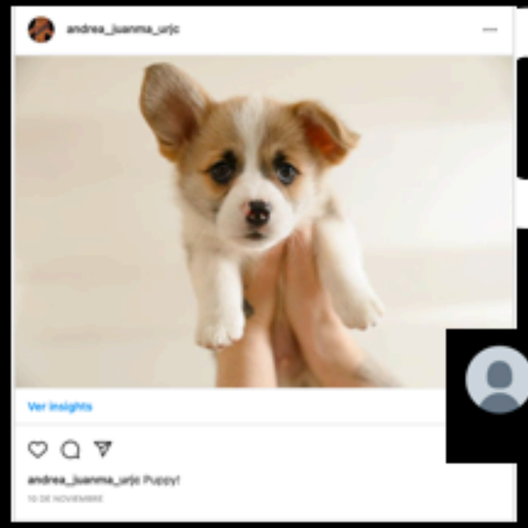
Andrea Juanma @andrea_juanma · Nov 19
Hola desde la The Social Networks Planner app 18 NOV!

Andrea Juanma @andrea_juanma · Oct 16
Replying to @andrea_juanma
Responder a un tweet desde la app y ver todas las respuestas ootra vez mas

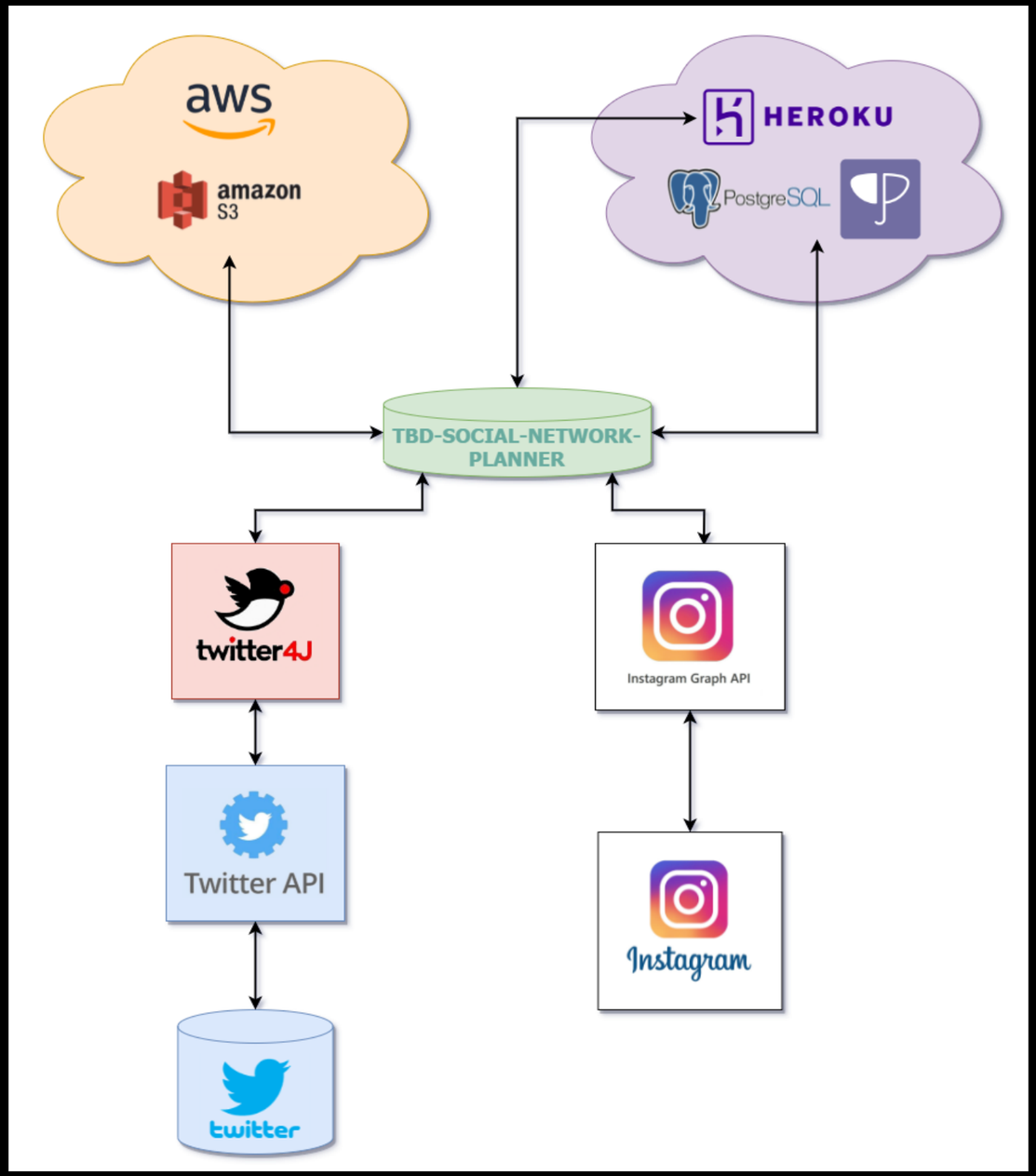
Andrea Juanma @andrea_juanma · Oct 9
Hello from Spring Java!



QUE ES TIBUTNET SOCIAL-PLAN



Andrea Juanma @andrea_juanma · Oct 26
Prueba de carga en BBDD interna

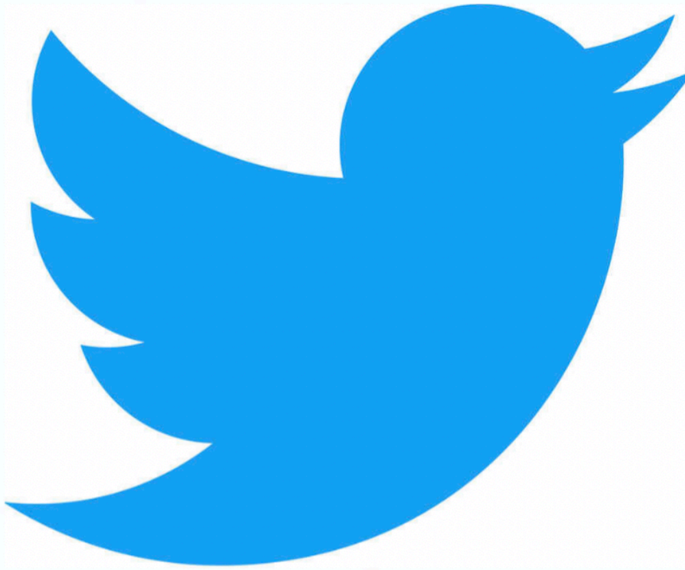

aws
amazon
S3
HEROKU
PostgreSQL
TBD-SOCIAL-NETWORK-PLANNER
twitter4J
Twitter API
twitter
Instagram Graph API
Instagram


spring


spring


HEROKU


spring




HEROKU

twitter4J

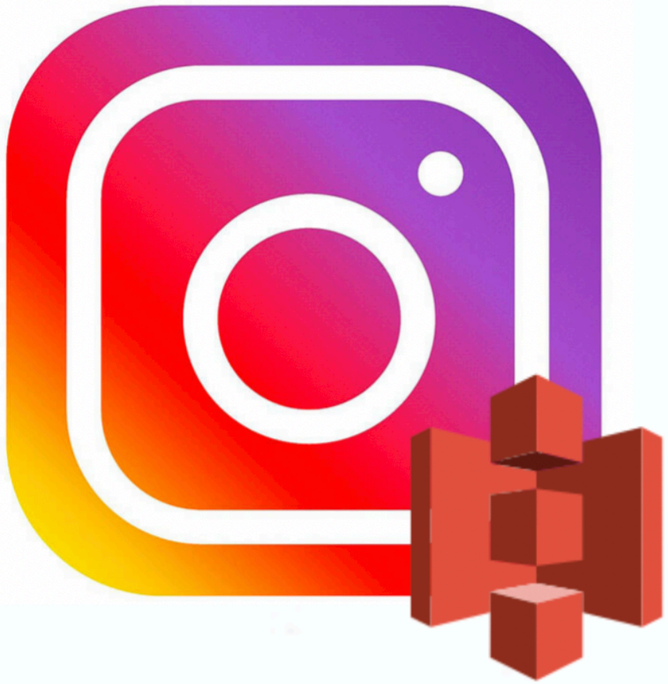
spring

HEROKU

twitter4J

spring

HEROKU



amazon
S3

HEROKU

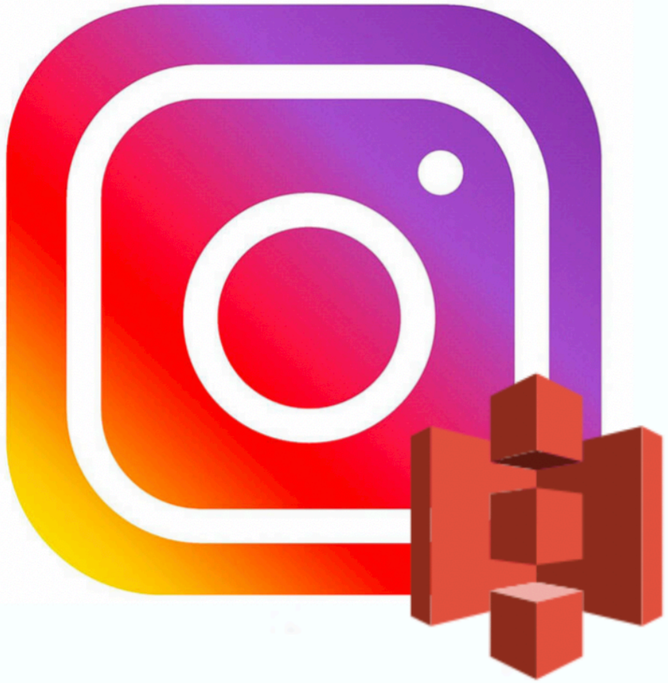
spring

twitter4J

git

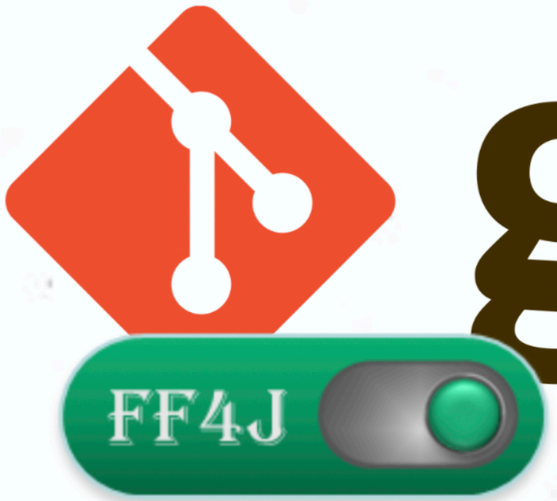
twitter4J

spring



HEROKU

amazon
S3



git

twitter4J

spring

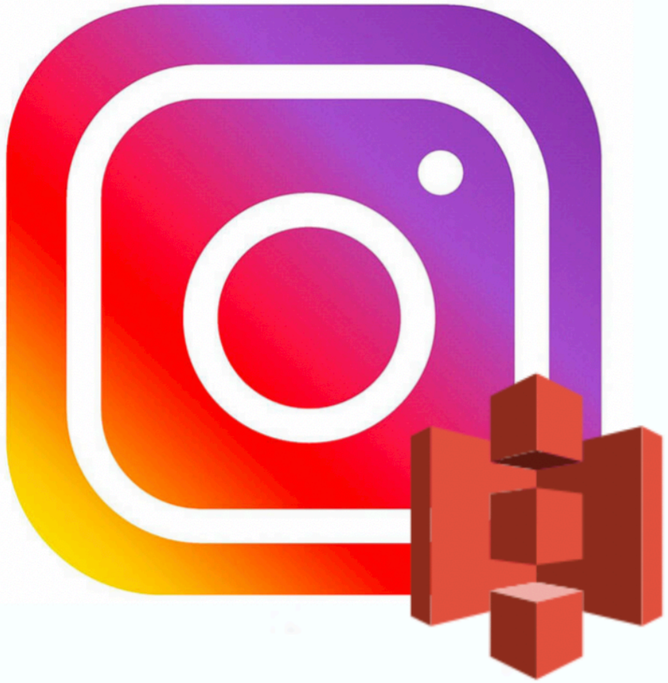
HEROKU

amazon
S3

git

twitter4J

spring



HEROKU

amazon
S3

Swagger™

CARACTERÍSTICAS:
TRUNK BASED DEVELOPMENT

» Ramas de corta duración

» Commits pequeños

» Se trabaja sobre trunk

» Pull requests sobre main (trunk)

» Uso de técnicas de CI/CD

TÉCNICAS APLICADAS:
TRUNK BASED DEVELOPMENT

» Feature Toggles

» Branch by Abstraction

TÉCNICAS APLICADAS: FEATURE TOGGLES
TRUNK BASED DEVELOPMENT

» Importamos la libreria FF4J:

» Creamos un feature toggle:

```
public static String FEATURE_ABSTRACT_IG_CLIENT = "Instagram client";

if (!ff4j.exist(FEATURE_ABSTRACT_IG_CLIENT)) {
            ff4j.createFeature(FEATURE_ABSTRACT_IG_CLIENT, false);
}
```

TÉCNICAS APLICADAS: FEATURE TOGGLES
TRUNK BASED DEVELOPMENT

» Usamos el toggle:

```
if (ff4j.check(FEATURE_ABSTRACT_IG_CLIENT)) {
    // New code (if toggle is active)
    return this.instagramRestClient.login();

} else {
    // Old code
    DeviceLoginRequest deviceLoginRequest = DeviceLoginRequest.builder()
        .accessToken(loginAccessToken)
        .redirectUri(redirectUri)
        .scope(scope)
    ...
```

TÉCNICAS APLICADAS: FEATURE TOGGLES
TRUNK BASED DEVELOPMENT

Activamos y desactivamos el toggle a través de la interfaz grafica:

https://ais-tbd-social-networks.herokuapp.com/ff4j-web-console/features

TÉCNICAS APLICADAS: BRANCH BY ABSTRACTION
TRUNK BASED DEVELOPMENT

Abstraemos la implementación de una funcionalidad a través de una interfaz.

Nos permite intercambiar las implementaciones.

Ambas implementaciones coexisten en el proyecto.

TÉCNICAS APLICADAS: BRANCH BY ABSTRACTION
TRUNK BASED DEVELOPMENT

» Creación de la interfaz:

```
public interface InstagramClient {

    InstagramDeviceLoginResponse login() throws InstagramException;

    String authenticate() throws InstagramException;

    String post(ResourceResponse resource, String caption) throws InstagramException;

    InstagramPostInfoResponse getPostInfo(String id) throws InstagramException;

    InstagramMediaResponse getAllMedia() throws InstagramException;

}
```

TÉCNICAS APLICADAS: BRANCH BY ABSTRACTION
TRUNK BASED DEVELOPMENT

» Implementation de la interfaz:

```
public class InstagramRestClient implements InstagramClient {

    @Override
    public InstagramDeviceLoginResponse login() throws InstagramException {
    ...
```

TÉCNICAS APLICADAS: BRANCH BY ABSTRACTION
TRUNK BASED DEVELOPMENT

» Uso de feature flag para activar desactivar la funcionalidad:

```
if (ff4j.check(FEATURE_ABSTRACT_IG_CLIENT)) {
    // Nueva implementación
    return this.instagramRestClient.login();

} else {
     // Código antiguo
     try {
        log.info("Request to Facebook Business API: " + deviceLoginUrl);

        ResponseEntity<InstagramDeviceLoginResponse> response =
    this.restTemplate.exchange(deviceLoginUrl, HttpMethod.POST,
    getEntity(deviceLoginRequest.toJsonString()), InstagramDeviceLoginResponse.class);
              ...
```

TÉCNICAS APLICADAS: BRANCH BY ABSTRACTION
TRUNK BASED DEVELOPMENT

» Futuros pasos:

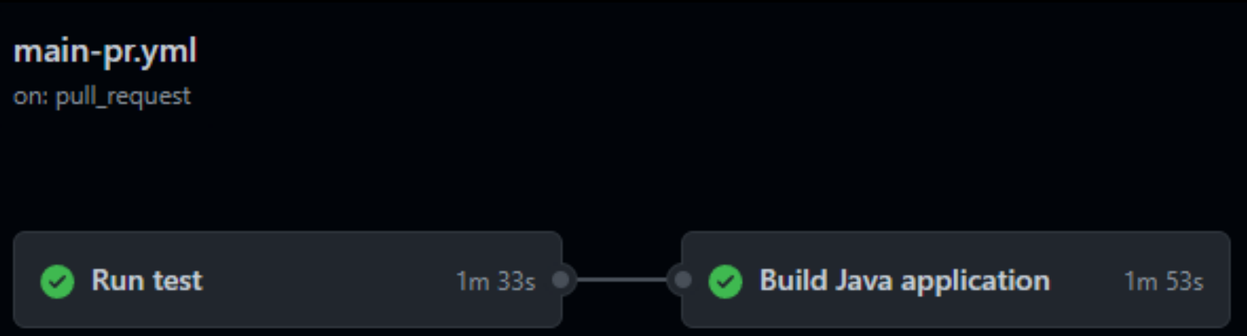
Esta abstracción nos va a facilitar en un futuro sustituir el cliente Rest por una librería, por ejemplo.

CI/CD

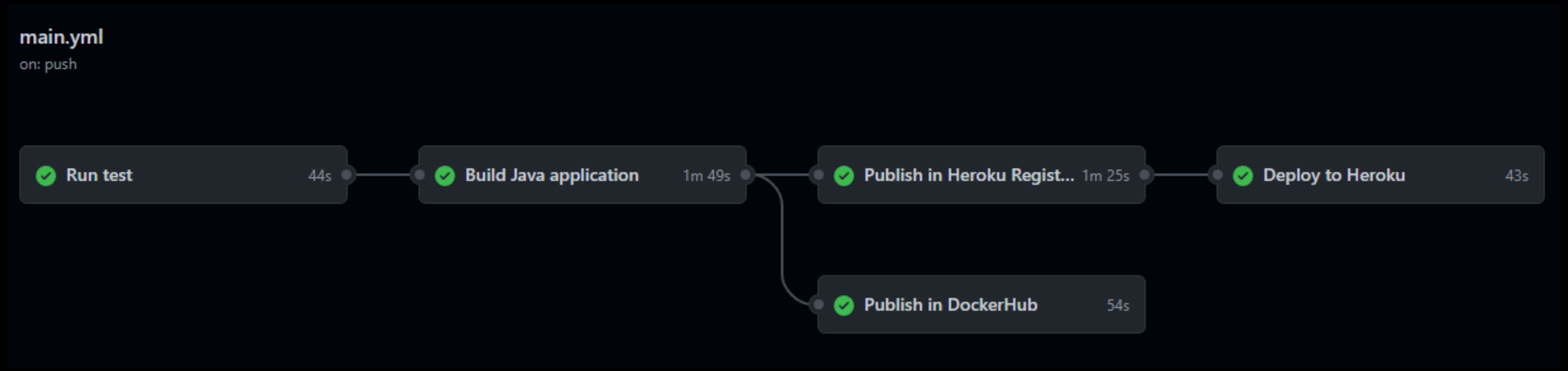
» GitHub Actions

» Testeo Automatizado

GITHUB ACTIONS
CI/CD



GITHUB ACTIONS
CI/CD



TESTEO AUTOMATIZADO

» Tests Unitarios

» Tests de Integración

CONCLUSIONES:

» Periodo de adaptación corto

» Conciencia de equipo

» GitHub Actions

FUTUROS PASOS

» Gestión de versiones

» Gestión de usuarios

» Terminar de configurar callback de Instagram

» Branch by Abstraction en Twitter Client

GRACIAS!