# A qui va dirigit

Aquest how-to va dirigit a tots aquells desenvolupadors/arquitectes que vulguin utilitzar GICAAR en entorns de desenvolupament en aplicacions Canigó 3.1.x.

# Versió de Canigó

Els passos descrits en aquest document apliquen a la versió 3.1.x del Framework Canigó.

# Introducció

S’ha habilitat una imatge Docker per a replicar el funcionament de GICAR en entorns de desenvolupament. En aquest how-to expliquem com utilitzar-la amb l’aplicació REST que genera el plugin de Canigó.

# Configuració GICAR

Afegir el mòdul de seguretat a l’aplicació Canigó:

Al **pom.xml** afegir les següents dependències

*…*

*<properties>*

*…*

*<canigo.security>[1.1.0,1.2.0)</canigo.security>*

*</properties>*

*…*

*<dependencies>*

*…*

*<dependency>*

*<groupId>cat.gencat.ctti</groupId>*

*<artifactId>canigo.security</artifactId>*

*<version>${canigo.security}</version>*

*<exclusions>*

*<exclusion>*

*<artifactId>spring-security-ldap</artifactId>*

*<groupId>org.springframework.security</groupId>*

*</exclusion>*

*</exclusions>*

*</dependency>*

*</dependencies>*

*…*

Al fitxer **web.xml** afegir el filtre de seguretat:

*...*

*<filter>*

*<filter-name>springSecurityFilterChain</filter-name>*

*<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>*

*</filter>*

*<filter-mapping>*

*<filter-name>springSecurityFilterChain</filter-name>*

*<url-pattern>/\*</url-pattern>*

*</filter-mapping>*

Afegir el fitxer de Spring on es configura la seguretat (**src/main/resources/app-custom-security.xml**). En aquest exemple hem protegit tot el bloc de l’api logs i les operacions PUT, POST i DELETE de l’api d’equipaments.

*<?xml version="1.0" encoding="UTF-8"?>*

*<beans xmlns="http://www.springframework.org/schema/beans"*

*xmlns:security="http://www.springframework.org/schema/security"*

*xmlns:context="http://www.springframework.org/schema/context"*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xmlns:jdbc="http://www.springframework.org/schema/jdbc"*

*xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.1.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.2.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.1.xsd">*

*<!-- Secure patterns -->*

*<security:http use-expressions="true">*

*<security:intercept-url pattern="/\*\*" access="permitAll" method="OPTIONS" />*

*<security:intercept-url pattern="/api/equipaments/\*\*" access="hasRole('ROLE\_ADMIN')" method="DELETE"/>*

*<security:intercept-url pattern="/api/equipaments/\*\*" access="hasRole('ROLE\_ADMIN')" method="PUT"/>*

*<security:intercept-url pattern="/api/equipaments/\*\*" access="hasRole('ROLE\_ADMIN')" method="POST"/>*

*<security:intercept-url pattern="/api/logs/\*\*" access="hasRole('ROLE\_ADMIN')" />*

*<security:form-login login-processing-url="/j\_spring\_security\_check" login-page="/j\_spring\_security\_check" />*

*<security:custom-filter ref="proxyUsernamePasswordAuthenticationFilter" before="FORM\_LOGIN\_FILTER" />*

*</security:http>*

*<security:authentication-manager alias="authenticationManager">*

*<security:authentication-provider ref="gicarProvider"/>*

*</security:authentication-manager>*

*<bean id="proxyUsernamePasswordAuthenticationFilter" class="cat.gencat.ctti.canigo.arch.security.authentication.ProxyUsernamePasswordAuthenticationFilter">*

*<property name="siteminderAuthentication" value="true" />*

*<property name="authenticationManager" ref="authenticationManager" />*

*<property name="authenticationFailureHandler" ref="failureHandler" />*

*</bean>*

*<bean id="failureHandler" class="org.springframework.security.web.authentication.SimpleUrlAuthenticationFailureHandler">*

*<property name="defaultFailureUrl" value="/gicar-error.html" />*

*</bean>*

*<bean id="gicarProvider" class="cat.gencat.ctti.canigo.arch.security.provider.siteminder.SiteminderAuthenticationProvider">*

*<description>GICAR Provider</description>*

*<property name="userDetailsService" ref="gicarUserDetailsService"/>*

*</bean>*

*<bean id="gicarUserDetailsService" class="cat.gencat.ctti.canigo.arch.security.provider.gicar.GICARUserDetailsServiceImpl">*

*<description>User Detail service implementation for GICAR provider</description>*

*<property name="httpGicarHeaderUsernameKey" value="${security.gicar.httpGicarHeaderUsernameKey:NIF}"/>*

*<property name="authoritiesDAO" ref="authoritiesDAO"/>*

*</bean>*

*<bean id="authoritiesDAO" class="cat.gencat.ctti.canigo.arch.security.provider.sace.authorities.AuthoritiesDAOImpl">*

*<description>Authorities DAO implementation for SACE. Gets granted authorities for specified user</description>*

*<property name="dataSource" ref="dataSource"/>*

*</bean>*

*</beans>*

Desplegament amb Docker

Primer de tot s’ha de tenir instal·lat Docker (<https://docs.docker.com/>)

Després creem la carpeta howto, i dintre d’aquesta carpeta dues carpetes amb el nom app i gicar.

A la carpeta **app** deixem el war de l’aplicació (app.war per exemple) i es crea el fitxer **Dockerfile** amb el següent contingut:

*FROM gencatcloud/tomcat:7*

*COPY app.war /opt/tomcat/webapps/*

*CMD ["/entrypoint.sh"]*

Amb aquest fitxer estem desplegant el nostre war en un tomcat 7 utilitzant la imatge gencatcloud/tomcat:7

A la carpeta gicar es crea el fitxer de configuració del nostre Apache amb la següent configuració de proxy pass:

*ProxyPass /${context}/ http://${APPSERVER\_PORT\_8080\_TCP\_ADDR}:${APPSERVER\_PORT\_8080\_TCP\_PORT}/${context}/*

*ProxyPassReverse /${context}/ http://${APPSERVER\_PORT\_8080\_TCP\_ADDR}:${APPSERVER\_PORT\_8080\_TCP\_PORT}/${context}/*

I es crea el fitxer **Dockerfile** amb el següent contingut:

*FROM gencatcloud/gicar:1.0*

*COPY httpd.conf /etc/httpd/conf/*

*CMD ["/entrypoint.sh"]*

Amb aquest fitxer estem desplegant la imatge gencatcloud/gicar:1.0 amb el fitxer httpd.conf modificat.

A la carpeta **howto** es crea el fitxer docker-compose.yml, amb el següent contingut

*gicar:*

*build: ./gicar/*

*links:*

*- demo*

*ports:*

*- 80:80*

*environment:*

*PS\_IP: [ip]*

*AgentConfigDocker: [aco]*

*ContainerHostName: [nom del contenidor]*

*AGENTNAME: [agent\_name].[nom de domini]*

*HCOGICAR: [hco]*

*GICARUSER: [user]*

*GICARPWD: [password]*

*APPSERVER\_PORT\_8080\_TCP\_ADDR: demo*

*APPSERVER\_PORT\_8080\_TCP\_PORT: 8080*

*context: app*

*demo:*

*build: ./app/*

*ports:*

*- 8080:8080*

*- 8000:8000*

Nota: Els valors de les variables d’entorn del contenidor de GICAR per aquesta howto s’actualitzaran en breu

Amb aquest fitxer estem posant en marxa dos contenidors, el de gicar que hem definit a la carpeta /gicar i de l’aplicació el construïm amb el fitxer Dockerfile de la carpeta /app que hem creat al punt anterior.

Per a realitzar aquesta operació executem des de la carpeta **howto**:

**docker-compose –f docker-compose.yml up –d**