

1 Knowledge Transfer

1.1 Libraries

Para poder utilizar alguna librería en un proyecto, buscar primero si ya existe en los [artefactos](#) de Azure.

Si existe, simplemente con un **yarn “nombre del paquete”** sería suficiente (consultar en los siguientes puntos si la librería necesita algún tipo de configuración). Si la librería no aparece en los artefactos seguir los pasos de la sección [Artifacts](#).

1.1.1 Authentication

Esta [biblioteca](#) contiene toda la lógica relacionada con la autenticación y autorización. Incluye la configuración de MSAL (Azure) y los componentes visuales que implementan la lógica de negocio de login, redirects, etc.

Para más información se puede consultar la [documentación](#). Concretamente el documento de [usage](#).

1.1.1.1 Extras

En la implementación de Login en el proyecto Landing, se ha proporcionado un parámetro en la URL para volver a una dirección inicial. Es decir, una aplicación externa puede detectar que no existan credenciales válidos y redirigir a /login indicando cuál es la URL origen a la que volver una vez se haya completado el inicio de sesión. Es importante que el parámetro sea codificado.

Por ejemplo:

https://dev-gnla.zurich.com/login?redirect_uri=https%3A%2F%2Fdev-gnla.zurich.com%2Freporting

1.1.2 Common (Shared Components)

Esta biblioteca contiene componentes comunes para las diferentes aplicaciones de GNLA. Para más información se puede consultar la [documentación](#).


La idea es desarrollar el componente con Storybook de forma atómica para después poder aplicarlo a los diferentes aplicativos.

1.2 Azure DevOps



1.2.1 Artifacts

Los artefactos son el resultado de la compilación de las librerías que han sido publicados en Azure. Están disponibles en el siguiente [enlace](#).

Para administrar los permisos:

gnla-applications
Connect to Feed
+ Create Feed
Recycle Bin


Filter by keywords
View
Source

<input type="checkbox"/>	Type	Package	Views	Source
<input type="checkbox"/>		@gnla-applications/auth Version 1.1.6		This feed
<input type="checkbox"/>		@gnla-applications/common Version 1.0.4		This feed

← Feed Settings
Delete feed

Feed Details
Permissions
Views
Upstream Sources

PermissionsAdd users/groups

User/Group	Role	Inherited
David Segovia Tomas	Owner	
GERARD ALBIOL	Owner	
VICTOR ENRIQUE ALCAZAR LOPEZ	Owner	
[zurichinsurance]\Project Collection Administrators	Owner	✓
[EDAA Azure DevOps GNLA]\Project Administrators	Owner	
Project Collection Build Service (zurichinsurance)	Contributor	
EDAA Azure DevOps GNLA Build Service (zurichinsurance)	Contributor	
[EDAA Azure DevOps GNLA]\Contributors	Contributor	

1.2.1.1 Configuración

Cada proyecto, tanto librería como consumidores, deben configurar el fichero `.npmrc` (y `.yarnrc` si se usa yarn).

.npmrc

```
@gnla-
applications:registry=https://pkgs.dev.azure.com/zurichinsurance/45f1f759-86f2-424e-a157-e158971c78a8/_packaging/gnla-
applications/npm/registry/
```

```
always-auth=true
```

.yarnrc

```
"@gnla-applications:registry"
"https://pkgs.dev.azure.com/zurichinsurance/45f1f759-86f2-424e-a157-e158971c78a8/_packaging/gnla-applications/npm/registry/"
```

Es importante borrar la carpeta **node_modules** y el fichero **package-lock.json** (o **yarn.lock**) si ya tenías las dependencias instaladas anteriormente. Una vez borrados, se deben reinstalar las dependencias con **npm install** (o **yarn**).

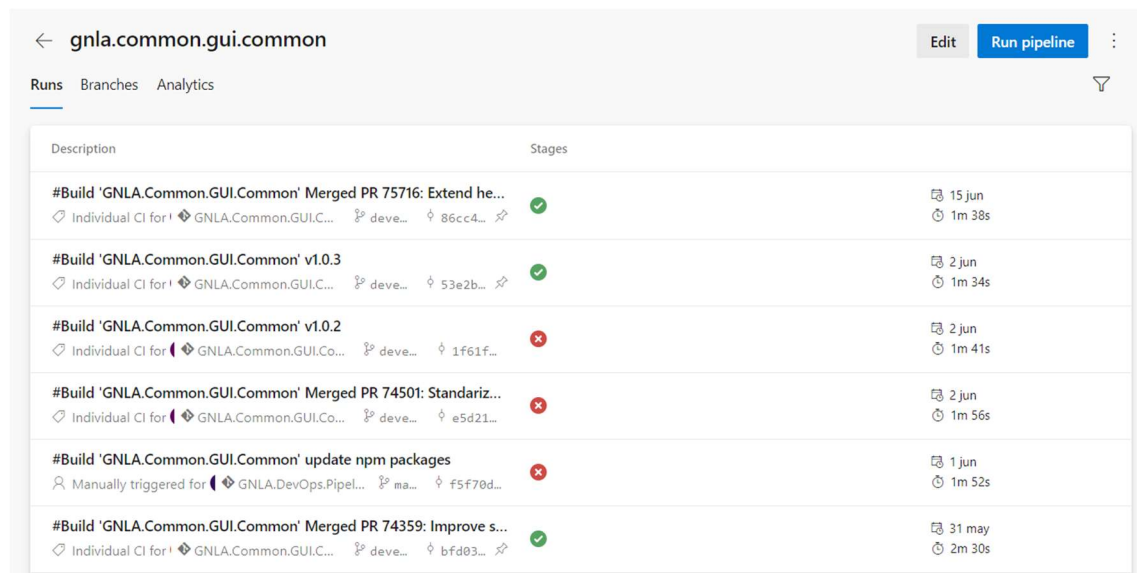
IMPORTANTE: para instalar las dependencias localizadas en Azure es importante iniciar sesión con tu usuario (npm) y tener permisos en el feed **gnla-applications** (ver sección anterior). Para iniciar sesión se puede usar el siguiente comando:

```
npx vsts-npm-auth -config .npmrc
```

Este comando añadirá tus credenciales automáticamente en **\$HOME/.npmrc**.

1.2.1.2 Publicar desde Azure Pipelines

Las pipelines han sido configuradas por el equipo de DevOps. En el siguiente [enlace](#) podemos acceder a todas las pipelines de los diferentes proyectos. Si accedemos a uno de los proyectos veremos el resultado de las ejecuciones.



Description	Stages	
#Build 'GNLA.Common.GUI.Common' Merged PR 75716: Extend he... Individual CI for GNLA.Common.GUI.C... deve... 86cc4...	✓	15 jun 1m 38s
#Build 'GNLA.Common.GUI.Common' v1.0.3 Individual CI for GNLA.Common.GUI.C... deve... 53e2b...	✓	2 jun 1m 34s
#Build 'GNLA.Common.GUI.Common' v1.0.2 Individual CI for GNLA.Common.GUI.Co... deve... 1f61f...	✗	2 jun 1m 41s
#Build 'GNLA.Common.GUI.Common' Merged PR 74501: Standariz... Individual CI for GNLA.Common.GUI.Co... deve... e5d21...	✗	2 jun 1m 56s
#Build 'GNLA.Common.GUI.Common' update npm packages Manually triggered for GNLA.DevOps.Pipel... ma... f5f70d...	✗	1 jun 1m 52s
#Build 'GNLA.Common.GUI.Common' Merged PR 74359: Improve s... Individual CI for GNLA.Common.GUI.C... deve... bfd03...	✓	31 may 2m 30s

Actualmente, se disparan las builds al pushear a la rama **develop**.

#Build 'GNLA.Common.GUI.Common' Merged PR 75716: Extend header with process flow icon ...

gnla.common.gui.common

Run new

1

This run is being retained as one of 3 recent runs by main (Branch).

View retention leases

Summary

Artifactory

Mend Bolt

WhiteSource Bolt Build Report

Triggered by

DT

 David Segovia Tomas

View 72 changes

Repositories 2

GNLA.DevOps.Pipelines , +1

See Sources card for details

Time started and elapsed

15 jun at 10:17

1m 38s

Related

0 work items

1 consumed

Tests and coverage

Get started

Jobs

Name	Status	Duration
<div><div></div>Build & Publish</div>	Success	1m 29s

Jobs in run #Build 'GNL...

gnla.common.gui.common

Build Stage

Build & Publish

1m 29s

Initialize job

4s

Checkout pipeline repo...

1s

Checkout code reposi...

<1s

Install Node.js 16.x

<1s

npmAuthenticate

<1s

yarn install

1m 1s

yarn build

12s

Npm

6s

Post-job: npmAuthent...

<1s

Post-job: Checkout co...

<1s

Post-job: Checkout pi...

<1s

Finalize Job

<1s

Report build status

<1s

Build & Publish

View raw log

1 Pool: Azure Pipelines

2 Image: ubuntu-latest

3 Agent: Hosted Agent

4 Started: 15 jun at 10:18

5 Duration: 1m 29s

6

7 Job preparation parameters

INTERNAL USE ONLY

The screenshot displays the Azure DevOps interface for a build job. On the left, a sidebar titled 'Jobs in run #Build 'GNL...' shows a list of build stages. The 'Build & Publish' stage is selected, and its sub-steps are listed below it. The 'Npm' step is highlighted with a red rectangle. The main panel on the right shows the details of the 'Build & Publish' stage, including a green checkmark, a search icon, and a 'View raw log' button. The details list the pool (Azure Pipelines), image (ubuntu-latest), agent (Hosted Agent), start time (15 jun at 10:18), duration (1m 29s), and job preparation parameters.

Build Stage	Duration
Build & Publish	1m 29s
Initialize job	4s
Checkout pipeline repo...	1s
Checkout code reposi...	<1s
Install Node.js 16.x	<1s
npmAuthenticate	<1s
yarn install	1m 1s
yarn build	12s
Npm	6s
Post-job: npmAuthent...	<1s
Post-job: Checkout co...	<1s
Post-job: Checkout pi...	<1s
Finalize Job	<1s
Report build status	<1s

En este step se publica el artefacto en Azure. Es importante controlar las versiones del **package.json** para que no colisione con los artefactos existentes. Para aumentar la versión de un paquete se debe usar **yarn version** y pushear los cambios antes de mergear en develop.

1.2.1.3 Publicar desde tu máquina local

Como método alternativo, es posible publicar un artefacto desde la propia máquina del desarrollador. Para esto, es importante haber compilado el proyecto antes de publicarlo. Cada comando de compilación es diferente así que recomendamos leer sus respectivos README.md para obtener dicha información.

Una vez el proyecto ha sido compilado, se puede usar **yarn publish** para publicar el artefacto en Azure.

1.3 Build

1.3.1 Local

Para ejecutar los aplicativos en local, es importante consultar el README de cada proyecto. Es posible que cada aplicativo use una versión de node distinta. Para obtener la versión correcta se debe consultar el fichero **.nvmrc** de cada proyecto.

1.3.2 CI

Las pipelines compilan los proyectos en cada push a develop. Por este motivo, se ha unificado el método de compilación al comando **yarn build:ci** que creará una carpeta **/build** localizada en la raíz del proyecto.

1.3.3 Entornos

En el fichero package.json, podremos ver las diferentes configuraciones para poder hacer un build en diferentes entornos, DEV, SIT, ...

Simplemente ejecutamos el script: **yarn build:<env>**

1.4 Deployment

Para poder hacer el deploy de un aplicativo, nos tendremos que ir al fichero README.md o tambien podremos encontrar la carpeta **docs/CONTRIBUTING.md**

En estos ficheros se explicará paso a paso todo lo necesario para poder hacer un deploy en el entorno de **DEV**.

NOTA: *Este siguiente paso es temporal hasta que se haga de forma automática con las pipelines*

Para poder hacer un deploy en **SIT**, tendremos que simplemente hacer un Build con el entorno de SIT, consultar [sección anterior](#). Una vez, tenemos el build.zip preparado, tendremos que enviarlo al equipo de DevOps para que ellos hagan el deploy.