# Guide pas à pas utilisateur filière IHM Light

# Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/WebHome?section=1)

Ce guide à pour but d'aider les projets dans l’écriture de leurs application IHM hors développement de la filière portail collaborateur.

Il détaillera toutes les étapes et implémentation des patterns d'architecture pouvant être nécessaires aux équipes en charge de déployer une application **IHM Light**.

Pour cela il s'appuie sur:

* Des patterns d'architectures
* Une application blanche

Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/WebHome?section=1)

Ce guide à pour but d'aider les projets dans l’écriture de leurs application IHM hors développement de la filière portail collaborateur.

Il détaillera toutes les étapes et implémentation des patterns d'architecture pouvant être nécessaires aux équipes en charge de déployer une application **IHM Light**.

Pour cela il s'appuie sur:

* Des patterns d'architectures
* Une application blanche

Intervenants de la filière

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/WebHome?section=2)

|  |  |
| --- | --- |
| **Intervenants** | **Rôle** |
| **Architectes** | Définition des patterns d'architecture |
| **Squad Socle Applicatifs** | Développement de l'application blanche et de la rédaction de ce guide utilisateur |
| **Chapitre Digital Front** | Développement des applications **Front**et **Back For Front**pour les squad |
| **Chapitre UX** | Définition des interfacesutilisateur |

Documents d'architecture

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/WebHome?section=3)

L'ensemble des documents d'architecture de la filière se trouve [ici](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Transverse/Architecture/Patterns%20d%E2%80%99architecture/)

# Pré-requis

Lé développement et le deploiement d'une application nécessite certains pré-requis indispensables

* **Droits Mouv** pour bénificier des outils de développement sur son poste développeur
* L'installation de ces outils et d'un environnement de développement adéquat

# Droits

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=1)

Pour bénéficier des outils de développement adéquat pour la filière, il est nécessaire de faire les demandes de droits suivants.

|  |  |  |
| --- | --- | --- |
| **Outils** | **Détails** | **Droits** |
| **GIT-S** | Gestionnaire de source pour les applications de type **Front End** | Droit mouv ? |
| **RTC** | Gestionnaire de source pour les applications de type **Back For Front / Image Docker / Stack** | Droit mouv ? |
| **Visual Studio Code** | Editeur de code pour les applications de type **Front End** | Droit mouv ? |
| **Eclipse Neon / Eclipse Kepler** | Editeur de code pour les application des type **Back For Front** | Droit mouv ? |
| **NPM** | Gestionnaire de module pour les application de type **Front End** | Droit mouv ? |
| **Jenkins CI** | Build des application de type **Front End** | Droit mouv ? |

# Environnement de développement & Framework

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=2)

Pour démarrer le projet devra posséder un code application, il faut vérifier l’existence de ce code dans [CATS application](https://ttp10-appl.ca-technologies.fr/), si le code est inexistant il faut faire une demande de création via ce même site.

La commande des **DOCKER Engine** et des**VIP/DNS** se fera suivant le schéma d’architecture technique définie par l‘architecte technique du projet et suivant le mode d’exposition de l’application : client et/ou collaborateur

## **Poste développeur**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=3)

|  |  |  |  |
| --- | --- | --- | --- |
| **Quoi** | **But** | **Qui** | **Comment** |
| **Visual Studio Code** | Permet le développement d’application type **Front End**  Raccourci Visual Studio code dans l'espace PFAB / Devops | PO/SM | Droits Mouv |
| **Eclipse SI Client** | Permet la création des images **Docker**  Raccourci Visual Studio code dans l'espace PFAB/ Branche A ou B / SI\_Client | PO/SM | Droits Mouv |
| **Eclipse Neon** | Permet le développement d’application type **Back For Front** (Java 8)  Raccourci Visual Studio code dans l'espace **PFAB/ Branche A ou B / Eclipse Neon** | PO/SM | Droits Mouv |
| **Raccourci PFAB** | Permet l’accès aux plateformes  de développement | PO/SM | Droits Mouv |
| **Nodejs** | Permet l’utilisation de **NPM**. Nécessaire pour les applications **Front End** | PO/SM | Viens avec le profil dev ? |
| **Git Bash** | Terminal de commande **GIT** | PO/SM | Viens avec le profil dev ? |

## **Gestionnaire de source**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=4)

|  |  |  |  |
| --- | --- | --- | --- |
| **Quoi** | **But** | **Qui** | **Comment** |
| **Création d’un projet dans Git** | Permet le stockage des sources du projet **Front end** | PO/SM | Demande PFAB |
| **Création d’un projet dans RTC** | Permet le stockage des sources du projet **Back For Front**  Permet le stockage des sources des **images / stacks Docker** | PO/SM | Demande PFAB |

## **Docker Engine / VIP / DNS**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quoi** | **But** | **Qui** | **Comment** |  |
| **Commande Docker Engine** | Permet de déployer les applications | SM/CPT | A la main du CPT |  |
| **Création VIP** | Permet d’exposer l’application | CPT | A la main du CPT |  |
| **Supervision** | Superviser les VIP de l’application | PO/SM | <http://mup10-posu.ca-technologies.fr/index.php?module=accueil> |  |
| **Création DNS** | Résolution depuis le client ou collaborateur | PO/SM | Effectuer une demande de service REMEDY D0032 |  |

## **Codes Application**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Pr%C3%A9-requis/WebHome?section=6)

|  |  |  |  |
| --- | --- | --- | --- |
| **Quoi** | **But** | **Qui** | **Comment** |
| **Création du Code application** | Permet la commande de **Docker Engine** | PO/SM/CPT | [https://ttp10-appl.ca-technologies.fr/](https://ttp10-appl.ca-technologies.fr/compte/identification) |
| **Client Id / Secret Id** | Connexion et accès **Api Manager / X Connect** | PO/SM | Déclarer l’application et avoir les droits Responsable Application sur le portail **API Manager** |

# Applications exemples

# Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/WebHome?section=1)

Dans le cadre de la filière, deux applications exemples ont été développées pour mettre en œuvre les différents Patterns d’architecture.

Elles se composent de la manière suivante

|  |  |
| --- | --- |
| **Angular** | **React** |
| **Front End**(**Angular 8**) | **Front End**(**React 16**) |
| **Back For Front**(Web Application **Java**déployée sous **Tomcat**) | **Back For Front**(Web Application **Java**déployée sous **Tomcat**) |
| **Redis**(partagée entre les 2 applications) | **Redis**(partagée entre les 2 applications) |

Dans la suite de ce guide, ces applications exemples serviront à démontrer l’implémentation des patterns. Il est donc recommandé des les installer localement.

Néanmoins, l'installation de l'application exemple **React**ne sert qu'à démontrer le pattern de transfert de contexte, son installation n'est donc pas obligatoire.

# Accès

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/WebHome?section=2)

Les applications exemples déployées sur **Docker CE** sont accessibles depuis les URL :

* [**Angular**](https://ihmlang.ca-devtu-zudb0.credit-agricole.fr/ihml-demo/)
* [**React**](https://ihmlang.ca-devtu-zudb0.credit-agricole.fr/ihml-demo-react)

error**Erreur DNS**

​

Selon les postes et l’accès réseau, la résolution DNS peut ne pas se faire.

Afin de résoudre ce problème, il faut ajouter à son fichier hosts (ou configurer Fiddler) avec les lignes suivantes :

10.166.142.138    zuda0-xconihmauthcol.ca-technologies.fr   
10.77.13.96        ihmlang.ca-devtu-zudb0.credit-agricole.fr  
10.77.13.97        ihmlbff.ca-devtu-zudb0.credit-agricole.fr   
​

# ​Installation locale

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/WebHome?section=3)

## **Composants**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/WebHome?section=4)

Afin de mieux comprendre l'implémentation des patterns d'architecture dans les deux applications exemples (**Angular**et **React**), nous conseillons d'installer sur son poste locale les deux applications.

Cette section documente:

* Les pré-requis du poste développeur
* L'installation et la configuration de l'application **Angular**
  + [**Front End**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/)
  + [**Back For Front**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/)
* L'installation et la configuration de l'application **React**
  + [**Front End**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/)
  + [**Back For Front**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/)
* L'installation et configuration de la base [**Redis**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Base%20Redis/)

## **Pré-Requis**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/WebHome?section=5)

L’installation, le développement des applications nécessite les outils suivants sur le poste de travail du développeur.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Front** | **Back For Front** | **REDIS** |
| **IDE** | Visual Studio Code | Eclipse NEON |  |
| **Framework** | Angular 8 / React 16 | Java 8 |  |
| **Gestion source** | GIT | RTC |  |
| **Gestion binaires** | NPM | Maven |  |
| **Redis** |  |  | Serveur Redis Local |

# Application Angular

# Framework

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=1)

L’application **Front End** utilise les Frameworks suivants

* **Angular 8** : <https://angular.io/>
* **NgRx** (implémentation de Redux) : <https://ngrx.io/>

Le but de ce guide n’étant pas détailler leur utilisation, il est recommandé de se documenter sur ces Frameworks avant de poursuivre le guide.

Le choix de **NgRx**est un choix de la **squad Socle Applicatif**, gérant l’application. Il n’est pas obligatoire et chaque équipe est libre d’utiliser les Frameworks souhaités.

Cependant, elle devra adaptée le code lié à chaque pattern selon ses choix techniques.

# Récupération des sources

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=2)

Les sources du projet **Front End** de l’application sont hébergées sur **GIT-S** :

* <http://ttp10-gits.ca-technologies.fr/tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-angular>

Pour récupérer les sources du projet il faut :

* Demander le droit d’accès à la **squad Socle Applicatifs** par mail [GSI.SOCLE.EXECUTION@ca-ts.fr](mailto:GSI.SOCLE.EXECUTION@ca-ts.fr)
* Cloner le projet grâce à l’utilitaire **Git Bash** installé sur son poste

## **SSH**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=3)

Nécessite d’avoir configure une **clé SSH** dans son profil **GitLab**(voir documentation **GitLab**)

git clone git@ttp10-gitsssh.ca-technologies.fr:tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-angular.git

## **HTTPS**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=4)

Il faut utiliser son compte ETXXXXX

git clone https://ttp10-gits.ca-technologies.fr/tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-angular.git

## **Branche**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=5)

error**Branche Git**

Attention la branche est très importante. Il existe d'autre branche qui ne correspondent à pas l'application déployée.

Se placer sur la branche **docker-ce**

git checkout docker-ce

# Installation

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Application%20Blanche%20Angular/WebHome?section=6)

Vous pouvez lancer **Visual Studio Code** (VSC) et ouvrir le dossier où se situent les sources de l’application.

Depuis le terminal de **VSC**se positionner à la racine du répertoire **ihml-front-angular** :

* Lancer la commande pour récupérer les modules **NPM**nécessaires à l’application.
  + npm install
* Lancer la commande pour démarrer le serveur web local
  + npm run **start**

Vous devriez accéder à l’application à via l'URL:  <http://localhost:4200/ihml-demo>

# Application React

# Framework

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=1)

L’application **Front End** utilise les Frameworks suivants

* **React** : [https://fr.reactjs.org/](https://angular.io/)
* **React scripts**: <https://create-react-app.dev/>
* **React-Redux**(implémentation de Redux) : [https://react-redux.js.org/](https://ngrx.io/)
* Redux-Thunk: <https://github.com/reduxjs/redux-thunk>

Le but de ce guide n’étant pas détailler leur utilisation, il est recommandé de se documenter sur ces Frameworks avant de poursuivre le guide.

Le choix de **React-Redux**est un choix de la **squad Socle Applicatif**, gérant l’application. Il n’est pas obligatoire et chaque équipe est libre d’utiliser les Frameworks souhaités.

Cependant, elle devra adaptée le code lié à chaque pattern selon ses choix techniques.

# Récupération des sources

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=2)

Les sources du projet **Front End** de l’application sont hébergées sur **GIT-S** :

* [https://ttp10-gits.ca-technologies.fr/tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-react](http://ttp10-gits.ca-technologies.fr/tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-angular)

Pour récupérer les sources du projet il faut :

* Demander le droit d’accès à la **squad Socle Applicatifs** par mail [GSI.SOCLE.EXECUTION@ca-ts.fr](mailto:GSI.SOCLE.EXECUTION@ca-ts.fr)
* Cloner le projet grâce à l’utilitaire **Git Bash** installé sur son poste

## **SSH**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=3)

Nécessite d’avoir configure une **clé SSH** dans son profil **GitLab**(voir documentation **GitLab**)

git clone git@ttp10-gitsssh.ca-technologies.fr:tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-react.git

## **HTTPS**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=4)

Utiliser son compte **ETXXXXX**

git clone https://ttp10-gits.ca-technologies.fr/tribu-socles-architecture/socle-applicatifs/ihm-light/ihml-front-react.git

## **Branche**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=5)

error**Branche Git**

Attention la branche est très importante. Il existe d'autre branche qui ne correspondent à pas l'application déployée.

Se placer sur la branche **docker-ce**

git checkout docker-ce

# Installation

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/React/WebHome?section=6)

Une fois les sources récupérées en local, vous pouvez lancer **Visual Studio Code** (VSC) et ouvrir le dossier où se situent les sources de l’application.

Depuis le terminal de **VSC**se positionner à la racine du répertoire **ihml-front-react**:

* Lancer la commande pour récupérer les modules **NPM**nécessaires à l’application.
  + npm install
* Lancer la commande pour démarrer le serveur web l
  + npm run **start**

Vous devriez pouvoir accéder à l’application à l’adresse:  <http://localhost:3000/ihml-demo-react>

# Back For Front

information Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

# Framework

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=1)

Le **Back For Front** (BFF) est basée sur les frameworks

* **Java 8**
* **Spring 5**
* **Spring Session** (Corn-RC1)
* Apache http client pour les appels d’API

# Récupération des sources

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=2)

Les projets **Back For Front** des applications exemples sont hébergés sur **RTC**:

* Flux: <https://ttp10-jazz.ca-technologies.credit-agricole.fr/ccm/resource/itemOid/com.ibm.team.scm.Workspace/__yy-0EgfEeq93dOmK4R5NQ>
* Composant : **WebApp\_ihml\_bff\_java** pour le **Back For Front** **Angular**
* Composant: **WebApp\_ihml\_bjj\_java\_react**pour le **Back For Front** **React**

Pour récupérer les sources il faut :

* Lancer **Eclipse Neon DAF** depuis les raccourcis de la Plateforme de fabrication
* Se connecter à **RTC** et récupérer les sources via le lien **RTC**
* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Matiere**
* Composants : **WebApp\_ihml\_bff\_java**ou **WebApp\_ihml\_bjj\_java\_react**

# Installation

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=3)

## **Serveur Tomcat**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=4)

Afin de pouvoir lancer le **Back For Front** en local, il est nécessaire d’avoir sur son poste: **Apache Tomcat 8.5.23**

Il faut ensuite créer un serveur **Apache Tomcat 8.5** dans **Eclipse Neon** :

* Fenêtre ⇒ Préférences ⇒ Server ⇒ **Runtime Environments**
* Cliquer sur **Add**et sélectionner **Apache Tomcat v8.5**
* Cliquer sur **Suivant**
  + Rechercher le dossier d’installation d’**Apache Tomcat 8.5.23** sur votre poste
  + Sélectionner le JRE : **jdk 1.8**
* Cliquer sur **Terminer**
* Le serveur doit être affiché dans la vue **Servers**
* Double-cliquer sur le serveur et sélectionner les options:
  + Use Tomcat installation (takes control of Tomcat installation)
  + Server path: chemin d’installation du **Tomcat**
  + Deploy path: répertoire webapps du dossier d’installation du **Tomcat**
* Double-cliquer sur le fichier **tomcat-users.xml** pour spécifier un User de connexion au manager **Tomcat**
  + Ici le user / mot de passe : tom / tom
* A ce stade vous pouvez lancer le serveur **Tomcat**et accéder au manager à l’URL: <http://localhost:8080/manager/html>

## **Déploiement application**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=5)

Pour pouvoir déployer les applications **Back For Front** **Angular** ou **React** il faut au préalable compiler et spécialiser.

Dans **Eclipse Neon**, les raccourcis PFAB ne sont pas encore intégrés.

Il faut donc créer deux exécutions pour lancer les jobs **Maven** qui permettent l’assemblage et la spécialisation:

* Exécuter ⇒ **Exécuter les configurations**
* Créer un nouveau **Maven Buid** : **Assemblage**
  + **Base directory** : sélectionner le projet en cliquant sur le bouton Workspace (**WebApp\_ihml\_bff\_java** ou **WebApp\_ihml\_bff\_java\_react**)
  + **Goals** : clean pic:assemblage
  + **Profiles** : gen,local,assemblage
  + **Maven Runtime** : sélectionner l’installation **Maven**située dans **C:/LOGICIELS/Apache**
* Créer un autre **Maven Build** : **Assemblage Specialisation**
  + **Base directory** : sélectionner le projet en cliquant sur le bouton Workspace (**WebApp\_ihml\_bff\_java** ou **WebApp\_ihml\_bff\_java\_react**)
  + **Goals**: pic:specialisation
  + **Profiles**: specialisation,local
  + **Maven Runtime** : sélectionner l’installation **Maven**située dans **C:/LOGICIELS/Apache**
* Lancer le **Maven Build** : **Assemblage**
* Lancer le **Maven Build** : **Assemblage Specialisation**
* A ce stade un fichier **/target/WebApp\_ihml\_bff\_java-${version}.war** ou **/target/WebApp\_ihml\_bff\_java\_react-${version}.war**best créé dans le répertoire du projet

Pour déployer l’application, il faut le faire via le manager **Tomcat**

* Rentrer un chemin de contexte : **/WebApp\_ihml\_bff\_java** ou **/WebApp\_ihml\_bff\_java\_react**
* Rentrer le chemin vers le fichier : {chemin vers le repertoire du projet}**/target/WebApp\_ihml\_bff\_java-${version}.war**ou{chemin vers le repertoire du projet}**/target/WebApp\_ihml\_bff\_java-\_react${version}.war**
* Cliquer sur Déployer
* Vous devriez avoir vos applications déployées

Pour s’assurer du bon déploiement vous pouvez effectuer la requête suivante via **POSTMAN**

* GET : <http://localhost:8080/WebApp_ihml_bff_java/api/security/login> ou [http://localhost:8080/WebApp\_ihml\_bff\_java\_react/api/security/login](http://localhost:8080/WebApp_ihml_bff_java/api/security/login)
* Le résultat doit être une erreur **HTTP 400**.
  + C’est normal cela prouve que le serveur répond (mais que des paramètres sont manquant pour la requête)

Pour son bon fonctionnement, le **Back For Front** a besoin de paramètres. Ces paramètres sont renseignés dans le fichier

* **src/main/resources/application.properties**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Détails** | **Utilisation** |
| **security.clientID** | Id permettant d’identifier l’application dans l’**API Manager** | Appel à l’**API AUC9** |
| **security.secretID** | Secret permettant d’authentifier l’application dans l’**API Manager**  **Ne doit pas être divulgué au Front End** | Appel à l’**API AUC9** |
| **security.session.cookie.domain** | Domain sur lequel l’application est déployée | Dans les cookies de session et state renvoyés après la connexion |
| **security.session.cookie.path** | Path à spécifier dans les cookies de session et state | Dans les cookies de session et state renvoyés après la connexion |
| **security.session.cookie.secure** | Spécifie si les cookies du **Back For Front** doivent être sécurisés ou non | Dans les cookies de session et state renvoyés après la connexion |
| **service.auc9** | URL de l’**API AUC9** | Appel à l’**API AUC9** |
| **service.ctx9** | URL de l'**API CTX9** | Appel à l'**API CTX9** |
| **service.places** | URL de l’**API Places** | Appel à l’**API Places** |
| **redis.host** | Host de la base de données **Redis** | Connexion à la base de données **Redis** |
| **redis.ports** | Liste de ports de connexion à la base de données **Redis**(plusieurs ports car utilisation d’un cluster **Redis**) | Connexion à la base de données **Redis** |

On peut voir que ces paramètres sont variabilisées. En effet selon l’environnement de déploiement de l’application, leurs valeurs seront différentes. Leur spécialisation selon les environnements sera abordée dans les chapitres sur la [CI / CD](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/).

# Configuration locale

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=6)

Pour le **Back For Front**, il est nécessaire de spécifier que nous souhaitons travailler avec notre serveur **Redis** local et non le cluster sur le **Docker Engine**.

Pour cela il faut modifier le fichier **src/main/local/local.properties** avec les bonnes valeurs pour la connexion au **Redis** local

local.properties

1  
2  
3

# Connexion au REDIS  
IHML\_redisHost=127.0.0.1  
IHML\_redisPorts=6379

Le code gérera automatiquement la connexion à une simple instance de **Redis** et non à un cluster.

information**Relancer les jobs Maven**

Penser à bien relancer les jobs Maven d’assemblage et de spécialisation avant de redéployer l’application.

information Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

# Framework

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=1)

Le **Back For Front** (BFF) est basée sur les frameworks

* **Java 8**
* **Spring 5**
* **Spring Session** (Corn-RC1)
* Apache http client pour les appels d’API

# Récupération des sources

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=2)

Les projets **Back For Front** des applications exemples sont hébergés sur **RTC**:

* Flux: <https://ttp10-jazz.ca-technologies.credit-agricole.fr/ccm/resource/itemOid/com.ibm.team.scm.Workspace/__yy-0EgfEeq93dOmK4R5NQ>
* Composant : **WebApp\_ihml\_bff\_java** pour le **Back For Front** **Angular**
* Composant: **WebApp\_ihml\_bjj\_java\_react**pour le **Back For Front** **React**

Pour récupérer les sources il faut :

* Lancer **Eclipse Neon DAF** depuis les raccourcis de la Plateforme de fabrication
* Se connecter à **RTC** et récupérer les sources via le lien **RTC**
* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Matiere**
* Composants : **WebApp\_ihml\_bff\_java**ou **WebApp\_ihml\_bjj\_java\_react**

# Installation

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=3)

## **Serveur Tomcat**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=4)

Afin de pouvoir lancer le **Back For Front** en local, il est nécessaire d’avoir sur son poste: **Apache Tomcat 8.5.23**

Il faut ensuite créer un serveur **Apache Tomcat 8.5** dans **Eclipse Neon** :

* Fenêtre ⇒ Préférences ⇒ Server ⇒ **Runtime Environments**
* Cliquer sur **Add**et sélectionner **Apache Tomcat v8.5**
* Cliquer sur **Suivant**
  + Rechercher le dossier d’installation d’**Apache Tomcat 8.5.23** sur votre poste
  + Sélectionner le JRE : **jdk 1.8**
* Cliquer sur **Terminer**
* Le serveur doit être affiché dans la vue **Servers**
* Double-cliquer sur le serveur et sélectionner les options:
  + Use Tomcat installation (takes control of Tomcat installation)
  + Server path: chemin d’installation du **Tomcat**
  + Deploy path: répertoire webapps du dossier d’installation du **Tomcat**
* Double-cliquer sur le fichier **tomcat-users.xml** pour spécifier un User de connexion au manager **Tomcat**
  + Ici le user / mot de passe : tom / tom
* A ce stade vous pouvez lancer le serveur **Tomcat**et accéder au manager à l’URL: <http://localhost:8080/manager/html>

## **Déploiement application**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=5)

Pour pouvoir déployer les applications **Back For Front** **Angular** ou **React** il faut au préalable compiler et spécialiser.

Dans **Eclipse Neon**, les raccourcis PFAB ne sont pas encore intégrés.

Il faut donc créer deux exécutions pour lancer les jobs **Maven** qui permettent l’assemblage et la spécialisation:

* Exécuter ⇒ **Exécuter les configurations**
* Créer un nouveau **Maven Buid** : **Assemblage**
  + **Base directory** : sélectionner le projet en cliquant sur le bouton Workspace (**WebApp\_ihml\_bff\_java** ou **WebApp\_ihml\_bff\_java\_react**)
  + **Goals** : clean pic:assemblage
  + **Profiles** : gen,local,assemblage
  + **Maven Runtime** : sélectionner l’installation **Maven**située dans **C:/LOGICIELS/Apache**
* Créer un autre **Maven Build** : **Assemblage Specialisation**
  + **Base directory** : sélectionner le projet en cliquant sur le bouton Workspace (**WebApp\_ihml\_bff\_java** ou **WebApp\_ihml\_bff\_java\_react**)
  + **Goals**: pic:specialisation
  + **Profiles**: specialisation,local
  + **Maven Runtime** : sélectionner l’installation **Maven**située dans **C:/LOGICIELS/Apache**
* Lancer le **Maven Build** : **Assemblage**
* Lancer le **Maven Build** : **Assemblage Specialisation**
* A ce stade un fichier **/target/WebApp\_ihml\_bff\_java-${version}.war** ou **/target/WebApp\_ihml\_bff\_java\_react-${version}.war**best créé dans le répertoire du projet

Pour déployer l’application, il faut le faire via le manager **Tomcat**

* Rentrer un chemin de contexte : **/WebApp\_ihml\_bff\_java** ou **/WebApp\_ihml\_bff\_java\_react**
* Rentrer le chemin vers le fichier : {chemin vers le repertoire du projet}**/target/WebApp\_ihml\_bff\_java-${version}.war**ou{chemin vers le repertoire du projet}**/target/WebApp\_ihml\_bff\_java-\_react${version}.war**
* Cliquer sur Déployer
* Vous devriez avoir vos applications déployées

Pour s’assurer du bon déploiement vous pouvez effectuer la requête suivante via **POSTMAN**

* GET : <http://localhost:8080/WebApp_ihml_bff_java/api/security/login> ou [http://localhost:8080/WebApp\_ihml\_bff\_java\_react/api/security/login](http://localhost:8080/WebApp_ihml_bff_java/api/security/login)
* Le résultat doit être une erreur **HTTP 400**.
  + C’est normal cela prouve que le serveur répond (mais que des paramètres sont manquant pour la requête)

Pour son bon fonctionnement, le **Back For Front** a besoin de paramètres. Ces paramètres sont renseignés dans le fichier

* **src/main/resources/application.properties**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Détails** | **Utilisation** |
| **security.clientID** | Id permettant d’identifier l’application dans l’**API Manager** | Appel à l’**API AUC9** |
| **security.secretID** | Secret permettant d’authentifier l’application dans l’**API Manager**  **Ne doit pas être divulgué au Front End** | Appel à l’**API AUC9** |
| **security.session.cookie.domain** | Domain sur lequel l’application est déployée | Dans les cookies de session et state renvoyés après la connexion |
| **security.session.cookie.path** | Path à spécifier dans les cookies de session et state | Dans les cookies de session et state renvoyés après la connexion |
| **security.session.cookie.secure** | Spécifie si les cookies du **Back For Front** doivent être sécurisés ou non | Dans les cookies de session et state renvoyés après la connexion |
| **service.auc9** | URL de l’**API AUC9** | Appel à l’**API AUC9** |
| **service.ctx9** | URL de l'**API CTX9** | Appel à l'**API CTX9** |
| **service.places** | URL de l’**API Places** | Appel à l’**API Places** |
| **redis.host** | Host de la base de données **Redis** | Connexion à la base de données **Redis** |
| **redis.ports** | Liste de ports de connexion à la base de données **Redis**(plusieurs ports car utilisation d’un cluster **Redis**) | Connexion à la base de données **Redis** |

On peut voir que ces paramètres sont variabilisées. En effet selon l’environnement de déploiement de l’application, leurs valeurs seront différentes. Leur spécialisation selon les environnements sera abordée dans les chapitres sur la [CI / CD](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/).

# Configuration locale

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Back%20For%20Front/WebHome?section=6)

Pour le **Back For Front**, il est nécessaire de spécifier que nous souhaitons travailler avec notre serveur **Redis** local et non le cluster sur le **Docker Engine**.

Pour cela il faut modifier le fichier **src/main/local/local.properties** avec les bonnes valeurs pour la connexion au **Redis** local

local.properties

1  
2  
3

# Connexion au REDIS  
IHML\_redisHost=127.0.0.1  
IHML\_redisPorts=6379

Le code gérera automatiquement la connexion à une simple instance de **Redis** et non à un cluster.

information**Relancer les jobs Maven**

Penser à bien relancer les jobs Maven d’assemblage et de spécialisation avant de redéployer l’application.

# Principe

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Base%20Redis/WebHome?section=1)

Une base de données **Redis**est nécessaire afin de pouvoir sauvegarder :

* Les informations sur les sessions utilisateurs
* Les informations à mettre en cache

Comme pour le **Back For Front** et le **Front End**, la base de données Redis sera installée sur **Docker CE**.

Cela sera détaillé dans les chapitres sur la [CI/CD](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/).

# Installation locale

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Base%20Redis/WebHome?section=2)

Il faut télécharger la dernière version de redis pour Windows sur le site suivant <https://github.com/microsoftarchive/redis/releases>

* Choiser la dernière release et la dézipper dans**C:\LOGICIELS\redis**
* lancer l'exécutable **redis-serve.exe**via une invit de command DOS ou PowerShell

L'instance redis local servira aussi bien à l'application **Angular**qu'à l'application **React**

Une fois installer, le serveur Redis est accessible sur

* **Host**: 127.0.0.1
* **Port**: 6379 (par défaut)

# Connexion Redis

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/angular/Base%20Redis/WebHome?section=3)

Il est possible de se connecter au **Redis** local pour vérifier si les clés sont bien créées.

* Exécuter l'exécutable **redis-cli.exe** installer avec le serveur **Redis**, via une invit de command DOS ou PowerShell

redis-cli.exe –h 127.0.0.1

Puis en utilisant les commandes **Redis**vous pouvez consulter les clés : <https://redis.io/commands/command>

Le client peut également être utilisée pour interroger le cluster **Redis**situé sur le **Docker CE**

redis-cli.exe –h ${ip redis docker ce} -p ${port redis docker ce}

L'installation sur Docker CE étant en cluster, celle-ci est donc répartie entre plusieurs noeuds sur le même Docker Engine.  
La sauvegarde des clés se fait donc aléatoirement sur les noeuds.  
Il sera donc nécessaire de trouver le bon noeuds ( ports [7001-7006] ) pour vérifier si les clés sont bien présentes

# Implémentation des patterns

Listes des patterns

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/WebHome?section=1)

La filière IHM Light définit plusieurs patterns d'architecture selon les fonctionnalités à réaliser par l'application.

Ci dessous un tableau récapitulatif des patterns et leur état d'avancement d'implémentation dans les applications exemples

|  |  |  |
| --- | --- | --- |
| **Patterns** | **Détails** | **Statut Implémentation** |
| **Versioning technique** |  |  |
| **Versioning applicatif** |  |  |
| [**Connexion à une application client ou collaborateur**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/) | Définit la procédure de connexion de l'application via la mire **X-Connect** et les **API AUC9 ou AUI9** | accept |
| [**Déconnexion à une application client ou collaborateur**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/) | Définit la procédure de déconnexion de l'application via la mire **X-Connect** et les **API AUC9 ou AUI9** | accept |
| [**Persistance de session**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/) | Définit la procédure de stockage des informations de session et leur utilisation dans l'application | accept |
| [**Appel d'API**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/) | Définit la procédure pour l'appel d'API (enregistrée dans l'**API Manager**) | accept |
| **Lancement d’une application depuis le NPC** | Définit la procédure de lancement d'une application depuis le **NPC** |  |
| **Lancement d’une application depuis le portail collaborateur** | Définit la procédure de lancement d'une application depuis le **Portail Collaborateur** |  |
| **Transfert de session** | Définit la procédure de partage de session entre deux applications IHM Light |  |
| [**Transfert de contexte**](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/) | Définit la procédure de partage de contexte entre deux applications IHM Light | accept |
| **Traçabilité et log** | Définit la procédure de gestion des logs dans l'application |  |

# Connexion à une application client ou collaborateur

Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/WebHome?section=1)

Dans ce chapitre nous allons démontrer comment le pattern de connexion (authorization code) a été implémenté dans les applications exemples.

Pour rappel, le but de ce pattern est de pouvoir générer un access\_token via la mire **X-Connect**, permettant l’utilisation future des API.

Cinématiques

# Mire X connect et API d'authentification

# Mires X-Connect

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Mire%20X%20connect%20et%20API%20d%27authentification/WebHome?section=1)

Il existe 4 types de mires **X-Connect** :

* **MAUO**: authentification des clients à partir d’application du monde open (i.e. non interne CA)
* **MAUC**: authentification des collaborateurs (avec **kerberos**) depuis une application non exposée sur internet
* **MACE**: authentification des collaborateurs (sans kerberos) depuis une application exposée sur internet
* **MAUI:** authentification des clients  partir d’une application

Les applications exemples utilisent la mire **MAUC**en environnement de développement.

Afin de connaître les URL de connexion aux mires selon les environnements, il faut contacter les équipes **X-Connect**.

# API AUC9 / AUI9

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Mire%20X%20connect%20et%20API%20d%27authentification/WebHome?section=2)

Afin de générer un **acces\_token** il faut utiliser l’API :

* **AUC9**: pour une identification collaborateurs
* **AUI9**: pour une identification clientes

L’application utilise l’API **AUC9**.

Comme pour les mires, afin de connaître les URL de connexion selon les environnements, vous pouvez contacter les équipes **X-Connect** et/ou récupérer les informations sur le [Portail des API](https://rct-private.developer.credit-agricole.fr/store/#top-page)

information Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

Dans ce pattern, le **Back For Front** doit :

* Exposer une ressource pour permettre de se connecter
* Appeler l’**API AUC9** pour récupérer un **access\_token**
* Stocker l’**access\_token** via le pattern de persistance de session ([voir pattern persistance de session](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/))
* Retourner deux cookies pour la gestion de persistance session ([voir pattern persistance de session](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/))

# Ressource Login

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Back%20For%20Front/WebHome?section=1)

Pour commencer, le **Back For Front** doit exposer une ressource permettant de faire la cinématique de connexion.

Dans notre application exemple, cela est fait dans la classe: **SecurityController.java**

SecurityController.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

/\*\*  
  \* Ressource d'identification à l'application  
  \*   
  \* @param code Authorization code fourni par la mire X Connect  
  \* @param redirectUrl Url de redirection de l'application founie à X Connect  
  \* @param state Id de "session" fourni lors de la connexion à la mire X Connect  
  \* @param httpSession Spring Session  
  \* @return   
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 @GetMapping("/login")  
 **public** ResponseEntity<Void> login(@RequestHeader (value = HttpHeaders.AUTHORIZATION) String code,  
   @RequestHeader (value = Constants.REDIRECT\_URL\_HEADER) String redirectUrl,  
   @RequestHeader (value = Constants.STATE\_HEADER) String state, HttpSession httpSession) **throws** ApiException, IOException {  
    
  **this**.securityService.login(code, redirectUrl, state, httpSession);  
    
  // Création du cookie STATE permettant de récupérer le refreh token dans le cache REDIS  
  HttpHeaders responseHeaders = **new** HttpHeaders();  
  String cookieString = AppUtils.buildCookie(Constants.STATE\_COOKIE, state, domain, path, cookieSecure, **null**, **false**);  
  responseHeaders.set("Set-Cookie", String.format(cookieString.toString(), Constants.STATE\_COOKIE, state, domain, path));  
     **return** ResponseEntity.ok().headers(responseHeaders).build();  
 }

On peut voir que la ressource attend les paramètres suivants :

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **code** | Header **Authorization** | Authorization code envoyée par le **Front End** |
| **redirectUrl** | Header **redirectUrl** | Url de redirection envoyée par le **Front End** |
| **state** | Header **state** | Id de state généré par le **Front End** et envoyé à la mire **X-Connect** |
| **httpSession** | La session HTTP est injectée par **Spring Session** | Elle contient les informations sur la session en cours. Son Id principalement |

Cette ressource va faire appel au **SecurityService**pour demander un **access\_token** à l’**API AUC9**.

En retour de cet appel, une session est créé et la ressource renvoie le cookie **STATE**avec l’Id précédemment généré par le **Front End**. En background un cookie **SESSION\_ID** est aussi renvoyé. Celui-ci est géré automatiquement par **Spring Session** ([voir pattern persistance de session](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/))

# Génération de l' access\_token (AT)

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Back%20For%20Front/WebHome?section=2)

La récupération de l’**access\_token** est effectuée dans la méthode **login**de la classe: **SecurityService.java**

SecurityService.java (login)

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46

/\*\*  
  \* Permet de récupérer un access token auprès de l'API AUC9  
  \* @param code Autorization Code renvoyée par la mire X Connect  
  \* @param redirectUrl Url de redirection de l'application founie à X Connect  
  \* @param state Id de "session" fourni lors de la connexion à la mire X Connec  
  \* @param httpSession Spring Session  
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 **public** void login(String code, String redirectUrl, String state, HttpSession httpSession)  
   **throws** ApiException, IOException {  
  mostLogger.debugInfo(String.format("Demande de jeton OAuth pour la session %s", httpSession.getId()));  
  
  // Construction de la requête  
  HttpPost postTokenRequest = **new** HttpPost(String.format(Constants.OPENID\_TOKEN\_URL, auc9Url));  
  
  // Set Headers  
  postTokenRequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_FORM\_URLENCODED\_VALUE);  
  postTokenRequest.addHeader(HttpHeaders.AUTHORIZATION, String.format("Basic %s", **this**.getAuthorizationToken()));  
  postTokenRequest.addHeader(Constants.CORRELATION\_ID\_HEADER, UUID.randomUUID().toString());  
  postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_HEADER, catsConsommateur);  
  postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_ORIGINE\_HEADER, catsConsommateurorigine);  
  postTokenRequest.addHeader(Constants.CATS\_CANAL\_HEADER, catsCanal);  
  
  // Set Body  
  List<NameValuePair> params = **new** ArrayList<>();  
  params.add(**new** BasicNameValuePair(Constants.GRANT\_TYPE\_PARAM, Constants.AUTORIZATION\_CODE\_VALUE));  
  params.add(**new** BasicNameValuePair(Constants.SCOPE\_PARAM, Constants.SCOPE\_VALUE));  
  params.add(**new** BasicNameValuePair(Constants.CODE\_PARAM, code));  
  params.add(**new** BasicNameValuePair(Constants.REDIRECT\_URI\_PARAM, redirectUrl));  
  postTokenRequest.setEntity(**new** UrlEncodedFormEntity(params));  
  
  // Execute request  
  OAuthToken oAuthToken = **this**.httpService.execute(postTokenRequest, OAuthToken.class);  
  oAuthToken.setUser(**this**.getUserFromTokenId(oAuthToken.getIdToken()));    
  
  // Sauvegarde de l'access token, du user et de la date d'expiration dans la session (et donc REDIS)  
  httpSession.setAttribute(Constants.ACCESS\_TOKEN\_KEY, oAuthToken.getAccessToken());  
  httpSession.setAttribute(Constants.USER\_ID\_KEY, oAuthToken.getUser().getId());  
  httpSession.setAttribute(Constants.EXPIRES\_AT\_KEY, AppUtils.delayFromNowToMilliseconds(oAuthToken.getExpiresIn()));  
  
  // Sauvegarde de refresh token dans le cache REDIS  
  **this**.refreshTokenCacheService.storeRefreshToken(state, oAuthToken.getRefreshToken());  
  
  mostLogger.debugInfo(String.format("Demande de jeton OAuth pour la session %s --> OK", httpSession.getId()));  
 }

On peut voir que cette méthode va construire une requête **REST**de type **POST**pour appeler la ressource ***/openid/token*** de l’**API AUC9**.

En retour de l’appel à la ressource, elle construit un objet **OAuthToken** contenant

|  |  |
| --- | --- |
| **Session** | **Détails** |
| **OAuthToken** | Information sur le token de connexion   * Id * Type * Access\_Token * Refresh\_token * Date d’expiration * User |

On peut noter que l’**access\_token** et certaines autres informations sont sauvegardées dans la session (puis dans un **Redis**). Cette partie sera expliquée dans le [pattern de Persistance de session](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/).

# Front End Angular

Dans ce pattern le **Front End** doit :

* Tester si la page de l’application à accéder est sécurisée
* Rediriger l’utilisateur vers la mire **X-Connect**
* Exposer une page de retour pour la mire **X-Connect**
* Appeler le **Back For Front** avec les informations renvoyées par la mire **X-Connect** (via la page de retour)
* Rediriger l’utilisateur vers la page demandée.

# Sécurisation d’une page

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=1)

La sécurisation d’une page via **Angular**, se fait grâce à la notion de **Guard**.

## **Routing**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=2)

Dans le fichier de **routing**de l’application **Angular**, il est possible de spécifier si une page doit être protégée par un **Guard**ou non.

Dans l’application exemple cela est fait dans le fichier: **src/app/app-routing.module.ts**

app-routing.modules.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

/\*\*  
 \* Tableau de routage de l'application  
 \*/  
**const** routes: Routes = [  
  { path: '', children: [  
    { path: '', redirectTo: '/home', pathMatch: 'full' },  
    { path: 'authorize', component: AuthorizeComponent },  
    { path: 'error', component: ErrorComponent },  
    { path: 'home', component: HomeComponent, canActivate: [AuthGuard] , data: {isSecure: **false**} },  
    { path: 'postMessage', component: ComPostMessageComponent, canActivate: [AuthGuard], data: {isSecure: **false**} },  
    { path: 'agences', component: SearchAgenceComponent, canActivate: [AuthGuard], data: {isSecure: **true**} },  
  ],  
    component: ContainerComponent  
  }  
];

On peut voir que la route **/agences** est protégée par le **AuthGuard**et qu'elle comporte une data**isSecure: True**. La data **isSecure**sera utilisé dans le **AuthGuard**pour tester l'accès sécurisé à la page.

## **Guard**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=3)

**AuthGuard**est déclaré dans le fichier: **src/auth/auth.guard.ts**

Ce fichier implémente l’interface **CanActivate**qui demande l’implémentation d’une méthode **canActivate**.

Cette méthode doit renvoyer

* **true**: si l’accès à la page est autorisée
* **false**: si l’accès à la page est non autorisé

auth.guard.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20

   /\*\*  
     \* Méthode pour détermine l'accès à une page ou non  
     \* @param route La route à accèder  
     \* @param state Information sur la route  
     \*/  
    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): **boolean** {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = **this**.getPathInfos(location.search).searchParams;  
  
        // Test si on n'a pas de context Id et si la route demandée n'est pas sécurisée  
        **const** noContextNoSecure = !route.data.isSecure && !\_.has(queryParamsMap, Constants.ROUTES.PARAMS.ID\_CONTEXT);  
         **if** (**this**.authService.authenticated || noContextNoSecure) {  
            // si on est authentifié ou si on n'a pas besoin de sécurité. On donne accès à la page  
            **return** **true**;  
        } **else** {  
            // Si ce n'est pas le cas on demande à effectué la cinématique de login  
            **this**.authService.login(state.url);  
            **return** **false**;  
        }  
    }

Dans notre cas la sécurisation d'une page est lié à plusieurs facteurs

* La présence du paramètre de requête **idCtx**
  + Ce paramètre est utilisé dans le pattern de partage de contexte
  + La partage de contexte nécessite d'être authentifier pour pouvoir être utilisé
  + Donc s'il est présent, on ne donne pas accès quelque soit la page
  + On redirige l'utilisateur vers la mire **X-Connect**
    - Appel de la méthode **login()** du **AuthService**
* Le paramètre **isSecure**de la page demandée
  + Si ce paramètre est à **False**, l'utilisateur peut accéder à la page demandée sans être authentifié
  + Si ce paramètre est à **True**et que l'utilisateur n'est pas authentifié
  + On redirige l'utilisateur vers la mire **X-Connect**
    - Appel de la méthode **login()** du **AuthService**
* Si l'utilisateur est déjà authentifié
  + Ceci est testé par la méthode **authenticated()**du **AuthService**
  + Alors l'utilisateur peut accéder à la page demandée

## **Service d'authentification**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=4)

Le service d'authentification à la charge de gérer la sécurisation de l'application: **src/services/auth.service.ts**

auth.service.ts

1  
2  
3  
4  
5  
6

   /\*\*  
     \* Permet de savoir si un utilisateur est authentifié  
     \*/  
    get authenticated(): **boolean** {  
        **return** !\_.isEmpty(**this**.user);  
    }

On peut voir que le service va vérifier si un utilisateur a été identifié. L’identification d’un utilisateur est faite au chargement de l’application dans la méthode **initialiseApp()** du **appInitService**

app-init-service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47

/\*\*  
     \* Permet d'initialisation l'application.  
     \* Appeler au lancement de l'application  
     \* Important: It should return a Promise  
     \*/  
    **public** initialiseApp() {  
  
        // Récupération de la configuration de l'application  
        **return** **this**.http.get('configuration/app-config.json').pipe(  
            map((config) => {  
                \_.assign(environment, config);  
                **return**;  
            }),  
            mergeMap(() => {  
                // Récupération du manifest du bff  
                **return** **this**.http.get(`**${**environment.apiBasePath**}**/manifest`);  
            }),  
            map((manifest: BffState) => {  
                // Mise à jour du manifes bff  
                **this**.store.dispatch(ManifestActions.setBffManifest({manifest}));  
                **return** manifest;  
            }),  
            map(() => {  
                // Mise à jour du manifest front  
                **const** manifest: FrontState = { version };  
  
                **this**.store.dispatch(ManifestActions.setFrontManifest({manifest}));  
                **return** manifest;  
            }),  
            mergeMap(() => {  
                // Récupération de l'utilisateur lié à la session  
                **return** **this**.http.get(`**${**environment.apiBasePath**}**/security/user`);  
            }),  
            map((user: User) => {  
                // S'il y a un utilisateur connecté on le sauvergarde dans le store  
                **this**.store.dispatch(SessionActions.setUser({ user: user }));  
                **return**;  
            }),  
            catchError(err => {  
                **if** (err.status === 404) {  
                    // S'il n'y a pas d'utilisateur on le reset  
                    **this**.store.dispatch(SessionActions.setUser({ user: **null** }));  
                }  
  
                **return** **of**(**null**);  
            })).toPromise();  
    }

* accept Si l’appel à **/security/user** renvoi une réponse **HTTP 200**
  + On sauvegarde l’utilisateur dans le store **Redux**.
  + Comme le **AuthService** écoute la modification du store **Redux**il sera mis à jour avec l’utilisateur provenant de la requête
* exclamation Si l’appel à **/security/user** renvoi une réponse **HTTP 404**
  + Aucun utilisateur trouvé. On met à jour le store **Redux**avec un utilisateur **null**
  + Comme le **AuthService** écoute la modification du store **Redux**il sera mis à jour avec l’utilisateur **null**indiquant que personne n’est authentifié
  + l’utilisateur sera alors renvoyé vers la méthode **login()**du **AuthService** si l’utilisateur du store **Redux**est **null**

# Redirection vers la mire X-Connect

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=5)

La redirection vers la mire **X-Connect** est gérée par la méthode **login()**du **AuthService**

auth.service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29

/\*\*  
     \* Gère la cinématique de connexion à la mire X Connect  
     \* @param redirectTo  Page de redirection une fois l'authentification faite  
     \*/  
    login(redirectTo: string) {  
        // Sauvegarde dans le localstorage la page demandée par l'utilisateur  
        **this**.storeRedirectTo(redirectTo);  
  
        // Génération d'un state pour X Connect et la récupértion du refresh token  
        **const** state: string = uuidv4();  
  
        // Construction URL pour la redirection vers le mire X Connect  
        **const** queryParams = {  
            client\_Id: environment.clientId,  
            redirect\_uri: environment.redirectUrl,  
            response\_type: 'code',  
            scope: 'openid functional\_posts',  
            state: state  
        };  
  
        **let** queryParamsString = `client\_id=**${**queryParams.client\_Id**}**`;  
        queryParamsString = `**${**queryParamsString**}**&redirect\_uri=**${**queryParams.redirect\_uri**}**`;  
        queryParamsString = `**${**queryParamsString**}**&response\_type=**${**queryParams.response\_type**}**`;  
        queryParamsString = `**${**queryParamsString**}**&scope=**${**queryParams.scope**}**`;  
        queryParamsString = `**${**queryParamsString**}**&state=**${**queryParams.state**}**`;  
  
        // Redirection vers la mire X Connect  
        location.assign(`**${**environment.xConnectUrl**}${**Constants.ROUTES.X\_CONNECT.AUTHORIZE**}**?**${**queryParamsString**}**`);  
    }

Cette méthode construit une requête de redirection en utilisant les paramètres nécessaires à la mire de connexion

|  |  |
| --- | --- |
| **Paramètres** | **Détails** |
| **client\_id** | **ClientId** de l’application (récupéré par la configuration) |
| **redirect\_uri** | Url de redirection vers l’application une fois le login de la mire effectué (récupéré par la configuration) |
| **response\_type** | ‘**code**’ (pour authorization code) |
| **scope** | ‘**openid functional\_posts**’ (pour la gestion **openid**avec les postes fonctionnels) |
| **state** | UUID (générée par le **Front End**) |

# Exposition de l’url de redirection

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=6)

La mire **X-Connect** à besoin d’une URL de redirection une fois l’authentification effectuée. Pour cela l’application **Front End** doit exposer une page permettant la gestion du retour de la mire.

Dans notre application, cette page est :

* Déclarée par le chemin **/authorize** dans le fichier **app-routing.module.ts**
* Gérée par le composant : **src/app/components/authorize/authorize.component.ts**

Le composant n’est qu’une page avec un spinner qui délègue la gestion de l’authentification au **AuthService**

authorize.component.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

@Component({  
    selector: 'app-authorize',  
    templateUrl: './authorize.component.html',  
    styleUrls: ['./authorize.component.scss']  
})  
**export** **class** AuthorizeComponent **implements** OnInit {  
  
    constructor(  
        **private** authService: AuthService  
    ) { }  
  
    ngOnInit() {  
        // Demande juste au service d'authentication de gérer le retour de la Mire  
        **this**.authService.handleAuth();  
    }  
}

# Appel du Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/WebHome?section=7)

Une fois la connexion faite, la mire **X-Connect** redirige l’utilisateur vers la page de redirection de l’application avec deux paramètres

|  |  |
| --- | --- |
| **Paramètres** | **Détails** |
| **code** | Authorization code |
| **state** | UUID (générée par le **Front End**) |

Le service **AuthService**va récupérer ces paramètres pour faire une demande d’**access\_token** au **Back For Front** via la méthode **handleAuth()**

auth.service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

   /\*\*  
     \* Gère le retour de la mire X Connect après authentification  
     \*/  
    handleAuth() {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = **this**.getPathInfos(location.search).searchParams;  
        **let** code = **null**;  
        **let** state = **null**;  
        **if** (!\_.isEmpty(queryParamsMap)) {  
            // Sauvegarde de l'authorization code et du state  
            code = queryParamsMap['code'];  
            state = queryParamsMap['state'];  
        }  
  
        **if** (!\_.isEmpty(code) && !\_.isEmpty(state)) {  
            // Si on un code et un state on demande la création d'un OAuthToken au BFF  
            **this**.securityService.login(code, environment.redirectUrl, state).subscribe(() => {  
                // Une fois connecté on demande les informations sur l'utilisateur  
                **this**.store.dispatch(SessionActions.getUser());  
            });  
        } **else** {  
            console.log('Authentification impossible');  
            **this**.router.navigate(['/error']);  
        }  
    }

La méthode **handleAuth()** utilise le service **SecurityService**pour

* Faire l’appel **/securiy/login** au **Back For Front**.
  + Qui va setter le **cookie SESSION\_ID** avec l’id de session
  + Qui va setter le **cookie STATE** avec l’UUID généré par le **Front End**
* Une fois le login effectué faire l’appel **/security/user** (via les effects **Redux**)
  + Qui va setter le **user**dans le store **Redux**
* Rediriger l’utilisateur vers la page initialement demandée.
  + Une fois l’utilisateur mis à jour dans le store **Redux**
  + Cette partie est gérée dans le constructeur du **AuthService**

auth.service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20

    constructor(  
        **private** securityService: SecurityService,  
        **private** router: Router,  
        **private** store: Store<AppState>  
    ) {  
        // Ecoute du store pour savoir si un utilisateur a été sauvegardé  
        **this**.store.pipe(select(UserSelectors.selectUser)).subscribe((user: User) => {  
            **this**.user = user;  
  
            **if** (!\_.isEmpty(user)) {  
                // Si un utilisateur existe on le redirige vers la page initialement demandée.  
                **const** redirectTo = **this**.getRedirectTo();  
                **if** (!\_.isEmpty(redirectTo)) {  
                    **const** pathInfos = **this**.getPathInfos(redirectTo);  
                    **this**.removeRedirectTo();  
                    **this**.router.navigate([pathInfos.root], {queryParams: pathInfos.searchParams});  
                }  
            }  
        });  
    }

# Front End React

Dans ce pattern le **Front End** doit :

* Tester si la page de l’application à accéder est sécurisée
* Rediriger l’utilisateur vers la mire **X-Connect**
* Exposer une page de retour pour la mire **X-Connect**
* Appeler le **Back For Front** avec les informations renvoyées par la mire **X-Connect** (via la page de retour)
* Rediriger l’utilisateur vers la page demandée.

# Sécurisation d’une page

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=1)

Par défaut, le framework **React**n'offre pas de solution de sécurisation de page (tel que peux le faire **Angular**). C'est donc à la charge des développeurs de l'application d'implémenter cette solution.

Dans le cas de l'application exemple **React**, cela est fait via:

* Le routage grâce au module **react-router-dom**
* Un composant **React PrivateRoute**développé dans l'application

## **Routing**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=2)

Afin de définir un mapping entre une URL et un composant **React**(page de l'application), il est possible d'utiliser un module **react-router\_dom**.

Ce module contient divers composant permettant la navigation dans une application **React**.

Dans l'application, ce module est utilisé dans le composant principal: **src/app/components/container/Container.js**

***...  
                    <div className="app-content">  
                        <Switch>  
                            <Route path="/authorize" component={Authorize} />  
                            <Route path="/error" component={ErrorPage} />  
                            <PrivateRoute path="/home" component={Home} isSecure={false} />  
                            <PrivateRoute path="/agences" component={SearchAgenceRoot} isSecure={true} />  
                            <Redirect from="\*" to="/home" />  
                        </Switch>  
                    </div>  
...***

On peut notre l'utilisation des composants:

* **Switch:**en charge du routage vers le bon composant selon le **path**
* **Route:**en charge du mapping entre un **path**et un **component**
* **PrivateRoute:**Le composant spécifique développé dans l'application pour gérer la sécurisation des pages

## **PrivateRoute**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=3)

 Le composant PrivateRoute est spécifique à l'application. Il n'est pas importé d'un module NPM externe.

Celui ci est  implémenté dans le fichier: src/app/components/routes/PrivateRoute.js

PrivateRoute.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33

/\*\*  
     \* Rendu du composant  
     \*/  
    render() {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = getPathInfos(**this**.props.history.location.search);  
  
        // Test si on n'a pas de context Id et si la route demandée n'est pas sécurisée  
        **const** noContextNoSecure = !**this**.props.isSecure && !\_.has(queryParamsMap, Constants.ROUTES.PARAMS.ID\_CONTEXT);  
  
        // Récupération de la page demandée  
        **if** (!\_.isEmpty(**this**.props.user) || noContextNoSecure) {  
            // Si un utilisateur est connecté on affiche le composant principal de la page demandée  
            **return** <Route path={**this**.props.path} component={**this**.props.component}/>  
        } **else** {  
            **let** redirectTo = **this**.props.history.location.pathname;  
  
            // Si aucun utilisateur n'est connecté  
            **if** (!\_.isEmpty(**this**.props.history.location.search)) {  
                redirectTo = `**${**redirectTo**}${this**.props.history.location.search**}**`;  
            }  
  
            // Gestion de la connexion avec X Connect  
            **this**.authentificationService.login(redirectTo);  
  
            // On affiche un spinner  
            **return** <div className="loader">  
                <div className="preloader">  
                    <CircularProgress size={100}/>  
                </div>  
            </div>  
        }  
    }

Dans notre cas la sécurisation d'une page est lié à plusieurs facteurs

* La présence du paramètre de requête **idCtx**
  + Ce paramètre est utilisé dans le pattern de partage de contexte
  + La partage de contexte nécessite d'être authentifier pour pouvoir être utilisé
  + Donc s'il est présent, on ne donne pas accès quelque soit la page
  + On redirige l'utilisateur vers la mire **X-Connect**
    - Appel de la méthode **login()** du **AuthentificationService**
* Le paramètre **isSecure**de la page demandée
  + Si ce paramètre est à **False**, l'utilisateur peut accéder à la page demandée sans être authentifié
  + Si ce paramètre est à **True**et que l'utilisateur n'est pas authentifié
  + On redirige l'utilisateur vers la mire X-Connect
    - Appel de la méthode **login()** du **AuthentificationService**
* Si l'utilisateur est déjà authentifié
  + Ceci est testé par le fait que **l'utilisateur**n'est pas **null**dans le store **Redux**
  + Alors l'utilisateur peut accéder à la page demandée

## **Utilisateur connecté**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=4)

On a vu que le composant **PrivateRoute**, s'appuie sur le store **Redux**pour savoir si un utilisateur est déjà connecté.

### **Composant principal**

L'application contient un composant principal qui englobe tous les autres composants: **src/app/components/container/Container.js**.

Container.js

1  
2  
3  
4  
5  
6  
7

   /\*\*  
     \* Appeler quand le composant est affiché dans le DOM  
     \*/  
    componentDidMount() {  
        // On charge la configuration à l'initialisation de l'application  
        **this**.props.setEnv();  
    }

Une fois que ce composant est **"monté"** dans le DOM, on demande le chargement de l'environnement via la méthode **setEnv** (Injectée par **React-Redux**)

### **Thunk**

error**Thunk redux**

Un Thunk middleware est une notion lié à l'utilisation de Redux (thunk-middleware Redux).

Cette implémentation du chargement de l'environnement est donc spécifique à notre application.

D'autres façons existent ne nécessitant pas l'utilisation de Redux.

Le thunk est définit dans le fichier **src/app/redux/thunks/EnvThunk.js**

EnvThunk.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35

/\*\*  
 \* Thunk pour setter l'environnement de l'application  
 \*/  
**export** **const** setEnv = () => {  
    **return** (dispatch) => {  
        **const** configurationService = iocContainer.get(TYPES.ConfigurationService);  
        **const** manifestService = iocContainer.get(TYPES.ManifestService);  
        **const** securityService = iocContainer.get(TYPES.SecurityService);  
        // On charge la configuration à l'initialisation de l'application  
        configurationService.getConfiguration().then(configuration => {  
            // Mise à jour du store avec le titre de l'application  
            dispatch(setTitle(configuration[Constants.APPLICATION.CONFIGURATION.TITLE]));  
            **return**;  
        }).then(() => {  
            // Récupération du manifest du Front End  
            **const** packJson = require('../../../../package.json');  
            dispatch(setFrontManifest({version: packJson.version}));  
            // Récupération du manifest du bff  
            **return** manifestService.getManifest().then((manifest) => {  
                dispatch(setBffManifest(manifest));  
            });  
        }).then(() => {  
            // On demande des informations sur l'utilisateur connecté  
            **return** securityService.getUserConnected().then((user) => {  
                dispatch(setUser(user));  
            })  
        }).**catch**((error) => {  
            **if** (error.status === 404) {  
                dispatch(setUser(**null**));  
            }  
        }).**finally**(() => {  
            dispatch(setEnvloaded(**true**));  
        });  
    }  
}

Cette méthode

* Charge le fichier de configuration **app-config-react.json** via le **service de configuratio**n et sa méthode **getConfiguration**
* Demande des informations aux **Back For Front** (manifest)
* Demande appelle la méthode **getUserConnected()** du **SecurityService**
  + accept Si l’appel à **/security/user** renvoi une réponse **HTTP 200**
    - On sauvegarde l’utilisateur dans le store **Redux**.
    - Comme le composant **PrivateRoute**écoute la modification du store **Redux**il sera mis à jour avec l’utilisateur provenant de la requête
  + exclamation Si l’appel à **/security/user** renvoi une réponse **HTTP 404**
    - Aucun utilisateur trouvé. On met à jour le store **Redux**avec un utilisateur **null**
    - Comme le composant **PrivateRoute** écoute la modification du store **Redux**il sera mis à jour avec l’utilisateur **null**indiquant que personne n’est authentifié
    - l’utilisateur sera alors renvoyé vers la méthode **login()**du **AuthentificationService**si l’utilisateur du store **Redux**est **null**

# Redirection vers la mire X-Connect

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=5)

La redirection vers la mire **X-Connect** est gérée par la méthode **login()** de l'**AuthentificationService**

AuthentificationService

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30

/\*\*  
     \* Gestion de la connexion  
     \* @param {} redirectTo Url de la page demandée par l'utilisateur  
     \*/  
    login(redirectTo) {  
        // Sauvegarde dans le localstorage de la page demandée par l'utilisateur  
        storeRedirectTo(redirectTo);  
  
        // Génération d'un state pour X Connect et la récupértion du refresh token  
        **const** state = uuidv4();  
  
        // Construction URL pour la redirection vers le mire X Connect  
        **const** queryParams = {  
            client\_Id: **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.CLIENT\_ID),  
            redirect\_uri: **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.REDIRECT\_URL),  
            response\_type: 'code',  
            scope: 'openid functional\_posts',  
            state: state  
        };  
  
        **let** queryParamsString = `client\_id=**${**queryParams.client\_Id**}**`;  
        queryParamsString = `**${**queryParamsString**}**&redirect\_uri=**${**queryParams.redirect\_uri**}**`;  
        queryParamsString = `**${**queryParamsString**}**&response\_type=**${**queryParams.response\_type**}**`;  
        queryParamsString = `**${**queryParamsString**}**&scope=**${**queryParams.scope**}**`;  
        queryParamsString = `**${**queryParamsString**}**&state=**${**queryParams.state**}**`;  
  
        // Redirection vers la mire X Connect  
        **const** xConnectRootUrl = **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.X\_CONNECT\_URL);  
        window.location.assign(`**${**xConnectRootUrl**}**/**${**Constants.ROUTES.X\_CONNECT.AUTHORIZE**}**?**${**queryParamsString**}**`);  
    }

 Cette méthode construit une requête de redirection en utilisant les paramètres nécessaires à la mire de connexion

|  |  |
| --- | --- |
| **Paramètres** | **Détails** |
| **client\_id** | **ClientId** de l’application (récupéré par la configuration) |
| **redirect\_uri** | Url de redirection vers l’application une fois le login de la mire effectué (récupéré par la configuration) |
| **response\_type** | ‘**code**’ (pour authorization code) |
| **scope** | ‘**openid functional\_posts**’ (pour la gestion **openid**avec les postes fonctionnels) |
| **state** | UUID (générée par le **Front End**) |

# Exposition de l'URL de retour

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=6)

La mire **X-Connect** à besoin d’une URL de redirection une fois l’authentification effectuée. Pour cela l’application **Front End** doit exposer une page permettant la gestion du retour de la mire.

Dans notre application, cette page est :

* Déclarée par la Route **/authorize** dans le fichier **Container.js**
* Gérée par le composant : **src/app/components/authorize/Authorize.js**

Le composant n’est qu’une page avec un spinner qui délègue la gestion de l’authentification au **AuthtificationService**

AuthentificationService

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

/\*\*  
     \* Appelé une fois le composant affiché dans le DOM  
     \*/  
    componentDidMount() {  
        **this**.authentificationService.handleAuth().then(() => {  
            // Une fois connecté on demande les informations sur l'utilisateur  
            **return** **this**.props.getUser();  
        }).then((user) => {  
            **if** (!\_.isEmpty(user)) {  
                // Si un utilisateur existe on le redirige vers la page initialement demandée.  
                // Sinon vers la page d'acceuil  
                **const** redirectTo = getRedirectTo();  
                **if** (!\_.isEmpty(redirectTo)) {  
                    removeRedirectTo();  
                    history.push(redirectTo);  
                } **else** {  
                    // Redirection vers la page d'accueil par défaut  
                    history.push('/home');  
                }  
            }  
        }).**catch**(error => {  
            // Rédirection vers la page d'erreur  
            history.push('/error');  
        });;  
    }

Si l'appel à la méthode **handleAuth()** de **l'AuthentificationService**s'est bien passé, on demande  au **Back For Front** l'utilisateur connecté via la méthode **getUser()** du Thunk **SessionThunk.js** **(src/app/redus/thunks/SessionThunk.js)**

ServiceThunk.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

/\*\*  
 \* Thunk pour récupérer l'utilisateur connecté  
 \*/  
**export** **const** getUser = () => {  
    **return** (dispatch) => {  
        **const** securityService = iocContainer.get(TYPES.SecurityService);  
        **return** securityService.getUserConnected().then(  
            (user) => {  
                dispatch(setUser(user));  
                **return** user;  
            }  
        )  
    }      
}

Si l'utilisateur est bien connecté on le met à jour dans le store **Redux**.

Puis on redirige l'utilisateur vers la page demandée.

# Appel du Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/WebHome?section=7)

Une fois la connexion faite, la mire **X-Connect** redirige l’utilisateur vers la page de redirection de l’application avec deux paramètres

|  |  |
| --- | --- |
| **Paramètres** | **Détails** |
| **code** | Authorization code |
| **state** | UUID (générée par le **Front End**) |

Le service **AuthtificationService**va récupérer ces paramètres pour faire une demande d’**access\_token** au **Back For Front** via la méthode **handleAuth()**

AuthentificationService.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

   /\*\*  
     \* Gestion du retour de l'auhthentification de la mire X Connect  
     \*/  
    handleAuth() {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = getPathInfos(window.location.search);  
        **let** code = **null**;  
        **let** state = **null**;  
        **if** (!\_.isEmpty(queryParamsMap)) {  
            // Sauvegarde de l'authorization code et du state  
            code = queryParamsMap['code'];  
            state = queryParamsMap['state'];  
        }  
  
        **if** (!\_.isEmpty(code) && !\_.isEmpty(state)) {  
            // Si on un code et un state on demande la création d'un OAuthToken au BFF  
            **const** redirectUrl = **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.REDIRECT\_URL)  
            **return** **this**.securityService.login(code, redirectUrl, state);  
        } **else** {  
            // Erreur lors de la connexion à la mire X Connect  
            console.log('Authentification impossible');  
            history.push('/error');  
            **return** Promise.resolve();  
        }  
    }

La méthode **handleAuth()** utilise le service **SecurityService**pour

* Faire l’appel **/securiy/login** au **Back For Front**.
  + Qui va setter le **cookie SESSION\_ID** avec l’id de session
  + Qui va setter le **cookie STATE** avec l’UUID généré par le **Front End**

# Déconnexion à une application client ou collaborateur

# Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/WebHome?section=1)

Dans ce chapitre nous allons démontrer comment le pattern de déconnexion a été implémenté dans l’application exemple.

Pour rappel, le but de ce pattern est de pouvoir se déconnecter complètement de l’application.

A savoir :

* Supprimer la session en cours
* Se déconnecter de la mire **X-Connect**

# Cinematiques

# Back For Front

Dans ce pattern, le **Back For Front** doit :

* Exposer une ressource pour la déconnexion
* Supprimer la session en cours
* Révoquer le **refresh\_token** de la session en cours auprès de l’**API AUC9**
* Invalider la session liée au **cookie SESSION\_ID**
* Révoquer le **cookie STATE**

# Logout ressource

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/Back%20For%20Front/WebHome?section=1)

Comme pour la connexion, le **Back For Front** expose une ressource pour se déconnecter via le fichier: **SecurityController.java**

SecurityService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19

/\*\*  
  \* Ressource de déconnexion  
  \* @param state d de "session" fourni lors de la connexion à la mire X Connect  
  \* @param httpSession Spring Session  
  \* @return  
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 @GetMapping("/logout")  
 **public** ResponseEntity<Void> logout(@CookieValue(Constants.STATE\_COOKIE) String state, HttpSession httpSession) **throws** ApiException, IOException {  
  **this**.securityService.logout(httpSession, state);  
    
  // Révocation du cookie STATE  
  HttpHeaders responseHeaders = **new** HttpHeaders();  
  String cookieString = AppUtils.buildCookie(Constants.STATE\_COOKIE, "deleted", domain, path, cookieSecure, **null**, **true**);  
  responseHeaders.set("Set-Cookie", String.format(cookieString.toString(), Constants.STATE\_COOKIE, "deleted", domain, path));  
  
     **return** ResponseEntity.ok().headers(responseHeaders).build();  
 }

Cette méthode attend le paramètre suivant

|  |  |  |
| --- | --- | --- |
| **Paramètre** | **Type** | **Détails** |
| **state** | Cookie **STATE** | UUID généré par le **Front End** lors de la connexion à la mire **X-Connect** |
| **httpSession** | La session HTTP est injectée par **Spring Session** | Elle contient les informations sur la session en cours. Son Id principalement |

Elle délègue la déconnexion au **SecurityService.java**. On notera également qu’un **header Set-Cookie** est envoyé pour invalider le **cookie STATE**.

# Révocation du refresh\_token

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/Back%20For%20Front/WebHome?section=2)

La méthode **logout()**va tout d’abord révoquer le **refresh\_token** de la session en cours en appelant la ressources **/openid/revoke** de l’**API AUC9** en passant les paramètres

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **Content-Type** | Header | application/x-www-form-urlencoded |
| **Authorization** | Header | Concatenation clientId:SecretId encodé en Base 64 |
| **correlationId** | Header | Id de la session en cours |
| **token** | Body | Valeur du **refresh\_token** de la session en cours |

SecurityService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

/\*\*  
  \* Gestion de la déconnexion  
  \* @param httpSession Spring Session  
  \* @param state Id de "session" fourni lors de la connexion à la mire X Connec  
  \* @throws IOException  
  \* @throws ApiException  
  \*/  
 **public** void logout(HttpSession httpSession, String state) **throws** IOException, ApiException {  
  mostLogger.debugInfo(String.format("Demande de déconnexion pour la session %s" ,httpSession.getId()));  
    
  // Révocation du refresh token lié à la session    
  **this**.revokeSession(httpSession, state);  
    
  mostLogger.debugInfo(String.format("Demande de déconnexion pour la session %s --> OK" ,httpSession.getId()));  
 }

 On peut noter qu’à la fin de la méthode de révocation, la session est invalidée. On détaillera plus loin dans le [pattern de persistance de session](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/) son impact.

# Front End React

Dans ce pattern le **Front End** doit :

* Appeler l’api **/security/logout** du **Back For Front**
* Rediriger vers la page de déconnexion de la mire **X-Connect**

# Appel du Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/Front%20End%20React/WebHome?section=1)

Sur le clique du bouton de déconnexion, l’application **Front End** va appeler la méthode **logout()** du **AuthentificationService (src/app/services/AuthentificationService.js)**

AuthentificationService.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11

   /\*\*  
     \* Gestion de la déconnexion  
     \*/  
    logout() {  
        // Appel de la déconnexion coté BFF pour revoké le refresh token  
        **this**.securityService.logout().**finally**(  
            () => {  
                // Redirection vers la mire X Connect pour la déconnexion  
                window.location.assign(`**${this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.X\_CONNECT\_URL)**}**/logout`);  
            });  
    }

L’application **Front End** appelle la méthode **logout()** du **SecurityService (src/app/services/SecrityService.js)** qui fait un appel a la ressource ***/security/logout*** du **Back For Front** pour gérer la suppression de la session.

# Redirection vers la page de déconnexion de X-Connect

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Gestion%20de%20la%20d%C3%A9connexion%20%C3%A0%20une%20application%20client%20ou%20collaborateur/Front%20End%20React/WebHome?section=2)

Une fois la déconnexion cotée **Back For Front** effectuée, l’application **Front End** redirige l’utilisateur vers la page de déconnexion de **X-Connect**, pour pouvoir gérer la déconnexion complète au niveau de la mire.

L’utilisateur devra cliquer sur un bouton de déconnexion de la mire (ce qui supprimera le cookie de session au niveau de la mire).

# Persistance de session

# Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/WebHome?section=1)

Dans ce chapitre nous allons démontrer comment le pattern de persistance de session a été implémenté dans l’application exemple.

Pour rappel, le but de ce pattern est de pouvoir :

* Sauvegarder les informations de toutes les sessions en cours sur l’application
* Sécuriser l’appel aux API du **Back For Front** via les informations de la session

# Cinématiques

# Back For Front

Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

Dans ce pattern, le **Back For Front** doit :

* Configurer **Spring Session** pour sauvegarder les informations de session dans **Redis**
* Utiliser **Spring session** pour sauvegarder les informations liées à la session
* Utiliser **Spring session** pour révoquer la session
* Sécuriser les ressources avec la validité de session
* Gérer le renouvellement du token

# Spring Session

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=1)

Le framework **Spring Session** est utilisé dans l’application blanche pour la gestion de la session.

**Spring Session** va :

* Se placer dans la chaîne de filtre de requête **Spring** pour utiliser son implémentation de l’interface **HTTPRequest**
* Permettre de sauvegarder des informations sur la session en cours
* Générer automatiquement un cookie de session qui sera renvoyé au **Front End** au moment de la première requête au **Back For Front** (dans le cas de notre application **/security/user**)

## **Configuration de Spring Session**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=2)

Pour que le framework **Spring Session** puisse sauvegarder les informations dans Redis il est nécessaire de le configurer.

Cela se fait via la classe **SessionConfig.java**.

Cette classe permet de :

* Activer la sauvegarde des informations de session dans **Redis**via l’annotation : **@EnableRedisHttpSession**
  + La propriété **cleanupCron**permet de laisser la main au Redis pour le nettoyage des clés.
  + La propriété **redisNamespace**permet de définir un préfix pour toutes les clés Redis gérées par **Spring Session**

SessionConfig.java

1

@EnableRedisHttpSession(cleanupCron = "0 0 0 29 02 ?", redisNamespace = Constants.NAMESPACE\_KEY)

* Déclarer une connexion a **Redis**

SessionConfig.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

/\*\*  
  \* Création du bean pour la connexion au REDIS  
  \*   
  \* @return Une instance pour la gesion des connexions au REDIS  
  \* @see {@link RedisConnectionFactory}  
  \*/  
 @Bean  
 RedisConnectionFactory jedisConnectionFactory() {  
  boolean localRedis = redisPorts.size() == 1 && redisPorts.contains(6379) && redisHost.equals("127.0.0.1");  
    
  **if** (localRedis){  
   mostLogger.erreurWarn("Aucun port ou host redis spécifié. Utilisation d'un REDIS local. Développement local seulement");  
   // Cas aucun port spécifié ou aucun host. Développement en local  
   **return** **new** JedisConnectionFactory();  
  } **else** **if** (redisPorts.size() > 1 && !redisHost.isEmpty()) {  
   mostLogger.debugInfo("Utilisation d'un cluster REDIS");  
   // Cas plusieurs port spécifiés. Utilisation cluster REDIS  
   Collection<String> clusterNodes = redisPorts.stream().map(port -> String.format("%s:%d", redisHost, port))  
     .collect(Collectors.toList());  
   **return** **new** JedisConnectionFactory(**new** RedisClusterConfiguration(clusterNodes));  
  } **else** {  
   **throw** **new** IllegalStateException("Le cache REDIS est mal configuré. Vérifié votre configuration");  
  }  
 }

* Déclarer une instance pour pouvoir interagir avec le cache **Redis**

SessionConfig.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18

/\*\*  
  \* Création du bean pour l'utilisation du cache REDIS  
  \*   
  \* @param factory  
  \*            Une factory de connexion au REDIS  
  \* @return Un Redis template permettant la manipulation des objets dans le  
  \*         cache REDIS  
  \* @see {@link RedisTemplate}  
  \*/  
 @Bean  
 RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory) {  
  SilentConnectionFailureRedisTemplate<String, Object> redisTemplate = **new** SilentConnectionFailureRedisTemplate<String, Object>();  
  redisTemplate.setConnectionFactory(jedisConnectionFactory());  
  redisTemplate.setKeySerializer(**new** StringRedisSerializer());  
  redisTemplate.setHashValueSerializer(**new** GenericToStringSerializer<Object>(Object.class));  
  redisTemplate.setValueSerializer(**new** GenericToStringSerializer<Object>(Object.class));  
  **return** redisTemplate;  
 }

* Redéfinir les paramètres du cookie de session **Spring**

SessionConfig.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26

/\*\*  
  \* Création du bean permettant la personnalisation du cookie de session  
  \* Spring  
  \*   
  \* @return  
  \*/  
 @Bean  
 **public** CookieSerializer cookieSerializer() {  
  DefaultCookieSerializer serializer = **new** DefaultCookieSerializer();  
  serializer.setCookieName(Constants.SESSION\_ID\_COOKIE);  
    
  **if** (Objects.nonNull(path)) {  
   serializer.setCookiePath(path);     
  }    
    
  **if** (Objects.nonNull(domain) && !domain.isEmpty()) {  
   serializer.setDomainName(domain);     
  }  
  
  serializer.setUseHttpOnlyCookie(**true**);  
  serializer.setUseSecureCookie(cookieSecure);  
  **if** (!cookieSecure) {  
   serializer.setSameSite("Lax");  
  }  
  **return** serializer;  
 }

* Spécifier que **Spring Session** ne doit pas modifier la configuration du Redis

SessionConfig.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11

/\*\*  
  \* Création du bean permettant de spécifier que spring n'effectue aucune  
  \* action de modification de configuration du REDIS  
  \*   
  \* @return Un objet de configuration du REDIS  
  \* @see {@link ConfigureRedisAction}  
  \*/  
 @Bean  
 **public** **static** ConfigureRedisAction configureRedisAction() {  
  **return** ConfigureRedisAction.NO\_OP;  
 }

* Déclarer un cache manager pour spécifier la date d’expiration des clés **Redis**(12 Heures. Choix arbitraire)

SessionConfig.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22

/\*\*  
  \* Création du bean permettant la création du cache manager Redis  
  \*   
  \* @param redisConnectionFactory  
  \*            Une factory de connexion au REDIS  
  \* @return Un cache manager  
  \* @see {@link CacheManager}  
  \*/  
 @Bean  
 **public** CacheManager cacheManager(RedisConnectionFactory redisConnectionFactory) {  
  // Creation liste des caches  
  Map<String, RedisCacheConfiguration> cacheConfigurations = **new** HashMap<String, RedisCacheConfiguration>();  
    
  // Création configuration par défaut des caches  
  RedisCacheConfiguration defaultCacheConfiguration = RedisCacheConfiguration.defaultCacheConfig();  
    
  // Ajout du cache pour la gestion des Refresh Token  
  cacheConfigurations.put(Constants.TOKEN\_CACHE, defaultCacheConfiguration.entryTtl(Duration.ofHours(12)).prefixKeysWith(Constants.TOKEN\_KEY));  
    
  **return** RedisCacheManager.builder(redisConnectionFactory)  
    .cacheDefaults(defaultCacheConfiguration).withInitialCacheConfigurations(cacheConfigurations).build();  
 }

Le cache manager nous permet de gérer plusieurs caches, dans notre cas nous avons créé

* **TOKEN\_CACHE**: le cache qui permettra de gérer les **refresh\_token**.
  + Nous avons paramétré une date d’expiration de 12 heures

Il faut noter «également » que les clés de session sauvegardées dans **Redis**possèdent

* Une période de validité sans interaction de **30 minutes**
  + Si aucun accès au **Back For Front** n’est fait pendant **30 minutes** la session expire
  + Il faudra demander un nouvel **access\_token** via le **refresh\_token** (géré par le filtre d’**ApiAuthenticationFilter**. Voir section [Sécurisation des Ressources](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/#S%C3%A9curisation%20des%20ressources))

La gestion de session peut très bien se faire sans ce framework. Il sera alors à la charge de l’application de réaliser les actions qui suivront.

## **Sauvegarde de la session**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=3)

On a vu précédemment que **Spring Session** va se charger de créer une session et de lui attribuer un **Id**. Il est possible par la suite d’ajouter des attributs à cette session afin de pouvoir les sauvegarder dans **Redis**.

Lors de la demande de jeton à l’**API AUC9**, il va être important de sauvegarder :

* L’**access\_token** pour pouvoir le récupérer lors d’un appel futur aux **API Places**
* La **date d’expiration** pour pouvoir tester si l’**access\_token** est toujours valide
* L’utilisateur de la session pour pouvoir renvoyer l’information au **Front End**

error**Sauvegarde refresh\_token**

De plus il est nécessaire de sauvegarder dans le cache Redis le refresh\_token. Ce dernier ne doit pas être sauvegardé avec la session mais à part dans une autre clé. En effet la session à une durée de validité dans Redis, si celle-ci vient à expirée, il faut que l’on soit toujours capable de demander un nouvel access\_token via le refresh\_token. C’est pour cela que l’on utilise le bean redisTemplate pour sauvegarder le refresh\_token dans le Redis. On utilise comme clé, le state généré par le Front End lors de la connexion.

Toutes ces actions sont faites lors de la phase de connexion dans la méthode login du **SecurityService**.

SecurityService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26

/\*\*  
  \* Permet de récupérer un access token auprès de l'API AUC9  
  \* @param code Autorization Code renvoyée par la mire X Connect  
  \* @param redirectUrl Url de redirection de l'application founie à X Connect  
  \* @param state Id de "session" fourni lors de la connexion à la mire X Connec  
  \* @param httpSession Spring Session  
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 **public** void login(String code, String redirectUrl, String state, HttpSession httpSession)  
   **throws** ApiException, IOException {  
...  
  // Execute request  
  OAuthToken oAuthToken = **this**.httpService.execute(postTokenRequest, OAuthToken.class);  
  oAuthToken.setUser(**this**.getUserFromTokenId(oAuthToken.getIdToken()));    
  
  // Sauvegarde de l'access token, du user et de la date d'expiration dans la session (et donc REDIS)  
  httpSession.setAttribute(Constants.ACCESS\_TOKEN\_KEY, oAuthToken.getAccessToken());  
  httpSession.setAttribute(Constants.USER\_ID\_KEY, oAuthToken.getUser().getId());  
  httpSession.setAttribute(Constants.EXPIRES\_AT\_KEY, AppUtils.delayFromNowToMilliseconds(oAuthToken.getExpiresIn()));  
  
  // Sauvegarde de refresh token dans le cache REDIS  
  **this**.refreshTokenCacheService.storeRefreshToken(state, oAuthToken.getRefreshToken());  
  
  mostLogger.debugInfo(String.format("Demande de jeton OAuth pour la session %s --> OK", httpSession.getId()));  
}

A noter qu’une classe **RefreshTokenCacheService**a été créé pour centraliser les méthodes d’accès aux clés du cache pour les **refresh\_token**.

## **Révocation de la session**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=4)

Lorsque l’utilisateur souhaite se déconnecter de l’application, il faut alors :

* Révoquer le **refresh\_token**.
  + Cela consiste en un appel de la ressource **/openId/revoke** de l’**Api AUC9**.
* Supprimer la clé **STATE** contenant le **refresh\_token** dans **Redis**
* Invalider la session **Spring**

Ces actions sont faites dans la méthode **revokeSession()** du **SecurityService**.

SecurityService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43

/\*\*  
  \* Permet la revocation du refresh token lié à la session en cours  
  \* @param httpSession Spring session  
  \* @param state Clé pour retrouver le refresh token dans le cache REDIS  
  \* @throws IOException  
  \* @throws ApiException  
  \*/  
 **public** void revokeSession(HttpSession httpSession, String state) **throws** IOException, ApiException {  
  mostLogger.debugInfo(String.format("Revocation du refresh token pour la session %s et la clé %s", httpSession.getId(), state));  
  
  // Récupération du refresh token  
  String refreshToken = **this**.getRefreshToken(state);  
    
  **if** (Objects.nonNull(refreshToken)) {  
   // Construction de la requête  
   HttpPost postTokenRequest = **new** HttpPost(String.format(Constants.OPENID\_REVOKE\_URL, auc9Url));  
  
   // Set Headers  
   postTokenRequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_FORM\_URLENCODED\_VALUE);  
   postTokenRequest.addHeader(HttpHeaders.AUTHORIZATION,  String.format("Basic %s", **this**.getAuthorizationToken()));  
   postTokenRequest.addHeader(Constants.CORRELATION\_ID\_HEADER, UUID.randomUUID().toString());  
   postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_HEADER, catsConsommateur);  
   postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_ORIGINE\_HEADER, catsConsommateurorigine);  
   postTokenRequest.addHeader(Constants.CATS\_CANAL\_HEADER, catsCanal);  
  
   // Set Body  
   List<NameValuePair> params = **new** ArrayList<>();  
   params.add(**new** BasicNameValuePair(Constants.TOKEN\_PARAM, **this**.refreshTokenCacheService.getRefreshToken(state)));  
   postTokenRequest.setEntity(**new** UrlEncodedFormEntity(params));  
  
   // Execute request  
   **this**.httpService.execute(postTokenRequest, **null**);  
     
   // Suppression de la clé dans le cache REDIS  
   **this**.refreshTokenCacheService.deleteRefreshToken(state);  
   mostLogger.debugInfo(String.format("Revocation du refresh token pour la session %s et la clé %s --> OK", httpSession.getId(), state));     
  } **else** {  
   mostLogger.erreurWarn(String.format("Aucune refresh token trouvé pour la session %s et la clé %s", httpSession.getId(), state));  
  }  
  
  // Invalidation de la session spring  
  httpSession.invalidate();  
 }

 Enfin il faut notifier au browser que le ***cookie STATE*** n’est plus valable.

Ceci est fait dans la méthode **logout()** du **SecurityController**.

SecurityController.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19

/\*\*  
  \* Ressource de déconnexion  
  \* @param state d de "session" fourni lors de la connexion à la mire X Connect  
  \* @param httpSession Spring Session  
  \* @return  
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 @GetMapping("/logout")  
 **public** ResponseEntity<Void> logout(@CookieValue(Constants.STATE\_COOKIE) String state, HttpSession httpSession) **throws** ApiException, IOException {  
  **this**.securityService.logout(httpSession, state);  
    
  // Révocation du cookie STATE  
  HttpHeaders responseHeaders = **new** HttpHeaders();  
  String cookieString = AppUtils.buildCookie(Constants.STATE\_COOKIE, "deleted", domain, path, cookieSecure, **null**, **true**);  
  responseHeaders.set("Set-Cookie", String.format(cookieString.toString(), Constants.STATE\_COOKIE, "deleted", domain, path));  
  
     **return** ResponseEntity.ok().headers(responseHeaders).build();  
 }

## **Sécurisation des ressources**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=5)

Le Framework **Spring**permet de définir des filtres qui peuvent être exécutés avant que la requête ne soit transmise réellement à la ressource. Le **Back For Front** définit plusieurs filtres :

* **CorsFilter.java**
  + Permettant de rajouter les directives Cors
* **ApiAuthenticationFilter.java**
  + Pour sécuriser les ressources du **Back For Front**
* **ErrorHandlerFilter.java**
  + Permettant de gérer les requêtes en erreur

La déclaration de ces filtres se fait dans le fichier **ApplicationInitializer.java**

ApplicationInitializer.java

1  
2  
3  
4  
5

@Override  
 **protected** Filter[] getServletFilters() {  
  // Ajout des différents filtre à la chaine de filtrage. L'ordre est important  
  **return** **new** Filter[]{**new** CorsFilter(), **new** ApiAuthenticationFilter(), **new** ErrorHandlerFilter()};  
 }

A noter que l’ordre des filtres est important. La requête va en effet passer par tous ces filtres successivement.

Dans le cadre du pattern de persistance de session et la sécurisation des ressources, le filtre qui nous intéresse est l’**ApiAuthenticationFilter.java**

ApiAuthentificationFilter.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69

@Override  
 **public** void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  
   **throws** IOException, ServletException {  
  HttpServletRequest httpServletRequest = (HttpServletRequest) request;   
  
  // Par défaut on laisse passer la requête  
  boolean refreshTokenNeeded = **false**;  
  
  // Application du filtre seulement si la requête n'est pas de type  
  // OPTIONS.  
  **if** (matchPatternUrls(httpServletRequest)  
    && !httpServletRequest.getMethod().equals(HttpMethod.OPTIONS.toString())) {  
   mostLogger.debugInfo(  
     String.format("L'accès à l'URL %s est gérée par le filtre: API", httpServletRequest.getPathInfo()));  
  
   // Récupération de la clé d'expiration du token de la session  
   Object expiresKey = httpServletRequest.getSession().getAttribute(Constants.EXPIRES\_AT\_KEY);  
   long sessionExpirationDateInMilli = Objects.nonNull(expiresKey) ? (long) expiresKey : 0;  
  
   **if** (sessionExpirationDateInMilli > 0) {  
    // Clé valide. On calcul la date d'expiration  
    LocalDateTime sessionExpirationDate = AppUtils.dateFromMilliseconds(sessionExpirationDateInMilli);  
  
    // on se laisse 30 secondes de marge pour que l' access token ne  
    // devienne pas invalide au milieu d'une séquence d'appels  
    LocalDateTime threshold = LocalDateTime.now().plusSeconds(30);  
  
    **if** (threshold.isAfter(sessionExpirationDate)) {  
     // Si la session va expiré dans 30 seconds on demande le  
     // refresh du token  
     refreshTokenNeeded = **true**;  
     mostLogger.debugInfo(String.format(  
       "L'access token de la session %s va expiré dans 30 secondes. On demande le refresh du token",  
       httpServletRequest.getSession().getId()));  
    }  
    mostLogger.debugInfo(String.format("L'access token de la session %s est autorisé à accéder à l'URL %s",  
      httpServletRequest.getSession().getId(), httpServletRequest.getPathInfo()));  
   } **else** {  
    // Si la session a expiré. Pas d'action depuis le TTL, on  
    // demande le refresh du token car l'access\_token n'est plus  
    // disponible  
    refreshTokenNeeded = **true**;  
    mostLogger.debugInfo(  
      String.format("L'access token de la session %s a expiré. On demande le refresh du token",  
        httpServletRequest.getSession().getId()));  
   }  
  }  
  
  **if** (refreshTokenNeeded) {  
   **try** {  
    // Récupération du cookie STATE pour avoir le refreshToken  
    Map<String, Cookie> requestCookies = **this**.getRequestCookies(httpServletRequest);  
    String state = requestCookies.get(Constants.STATE\_COOKIE).getValue();  
  
    // Demande de refresh token  
    **this**.securityService.refreshToken(httpServletRequest.getSession(), state);  
   } **catch** (Exception e) {  
    // Une erreur est intervenue dans la demande du refresh\_token.  
    // On rejete la requête  
    mostLogger.erreurWarn(e.getMessage());  
    writeResponse(HttpStatus.SC\_UNAUTHORIZED, Constants.ACCESS\_UNAUTHORIZED\_RESPONSES, (HttpServletResponse) response);  
      
    // On return pour casser la chaine de filtrage et renvoyer la réponse directement  
    **return**;  
   }  
  }  
    
  chain.doFilter(request, response);  
 }

 Le filtre va

1. Vérifier que la ressource demandée est à sécurisée
2. Récupérer la date d’expiration de l’**access\_token** dans le Redis
   1. Si la clé existe (**EXPIRES\_AT\_KEY**)
      1. On teste si la session expire dans **30 secondes**
         1. Si oui on demande le refresh du token
      2. Si la session est valide on donne l’accès à la ressource
   2. Si la clé n’existe pas (elle a expiré)
      1. On demande le refresh du token
3. Si le refresh du token est demandé
   1. On teste si les **cookies STATE** et **SESSION\_ID**existent
      1. Si l’un des cookies n’existe pas on refuse l’accès à la ressource. Code **HTTP 401** exclamation
   2. On récupère la clé **Redis**associé au **STATE**
      1. Si la clé n’existe pas (expiration ou aucune session créé pour ce **STATE**)
         1. On refuse l’accès à la ressource. Code **HTTP 401** exclamation
      2. Si la clé existe
         1. On effectue le renouvellement du token
            1. Si une erreur intervient pendant le renouvellement on refuse l’accès à la requête. Code **HTTP 401** exclamation
            2. Si le renouvellement est bon on donne l’accès à la ressource

## **Refresh du token**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Back%20For%20Front/WebHome?section=6)

Lorsque la session expire, nous devons demander un nouvel **access\_token** pour continuer à accéder aux API.

* Cela est fait via un appel à la ressource **/openid/token** de l’**API AUC9**.
* Cette action est gérée par la méthode **refreshToken()** du **SecurityService**.

SecurityService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47

/\*\*  
  \* Permet le renouvellement de l'access token à partir du refresh token  
  \* @param httpSession Spring session  
  \* @param refreshToken Le refresh toke  
  \* @throws ApiException  
  \* @throws IOException  
  \*/  
 **public** void refreshToken(HttpSession httpSession, String state) **throws** ApiException, IOException {  
  mostLogger.debugInfo(String.format("Renouvellement de l'access token pour la session %s et la clé %s", httpSession.getId(), state));  
  
  mostLogger.debugInfo(String.format("Récupération du refresh token pour la session %s et la clé %s", httpSession.getId(), state));  
  String refreshToken = **this**.getRefreshToken(state);  
  **if** (Objects.nonNull(refreshToken)) {  
     
   // Construction de la requête  
   HttpPost postTokenRequest = **new** HttpPost(String.format(Constants.OPENID\_TOKEN\_URL, auc9Url));  
  
   // Set Headers  
   postTokenRequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_FORM\_URLENCODED\_VALUE);  
   postTokenRequest.addHeader(HttpHeaders.AUTHORIZATION,  String.format("Basic %s", **this**.getAuthorizationToken()));  
   postTokenRequest.addHeader(Constants.CORRELATION\_ID\_HEADER, UUID.randomUUID().toString());  
   postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_HEADER, catsConsommateur);  
   postTokenRequest.addHeader(Constants.CATS\_CONSOMMATEUR\_ORIGINE\_HEADER, catsConsommateurorigine);  
   postTokenRequest.addHeader(Constants.CATS\_CANAL\_HEADER, catsCanal);  
  
   // Set Body  
   List<NameValuePair> params = **new** ArrayList<>();  
   params.add(**new** BasicNameValuePair(Constants.GRANT\_TYPE\_PARAM, Constants.REFRESH\_TOKEN\_VALUE));  
   params.add(**new** BasicNameValuePair(Constants.SCOPE\_PARAM, Constants.SCOPE\_VALUE));  
   params.add(**new** BasicNameValuePair(Constants.REFRESH\_TOKEN\_PARAM, refreshToken));  
   postTokenRequest.setEntity(**new** UrlEncodedFormEntity(params));  
  
   // Execute request  
   OAuthToken oAuthToken = **this**.httpService.execute(postTokenRequest, OAuthToken.class);  
   oAuthToken.setUser(**this**.getUserFromTokenId(oAuthToken.getIdToken()));  
     
   // Sauvegarde du nouvel access token, du user et de la date d'expiration dans la session (et donc REDIS)  
   httpSession.setAttribute(Constants.ACCESS\_TOKEN\_KEY, oAuthToken.getAccessToken());  
   httpSession.setAttribute(Constants.USER\_ID\_KEY, oAuthToken.getUser().getId());  
   httpSession.setAttribute(Constants.EXPIRES\_AT\_KEY, AppUtils.delayFromNowToMilliseconds(oAuthToken.getExpiresIn()));  
  
   mostLogger.debugInfo(String.format("Renouvellement de l'access token pour la session %s et la clé %s --> OK", httpSession.getId(), state));   
  } **else** {  
   **throw** **new** IllegalArgumentException(String.format("Le refresh token pour la session %s et le state %s n'existe pas", httpSession.getId(), state));  
  }  
    
 }

On peut noter que lorsque le nouvel **access\_token** est reçu, on le sauvegarde ainsi que sa date d’expiration dans la session (et donc dans **Redis**). En ce qui concerne le **refresh\_token** on ne fait rien car celui-ci est toujours valide **(pour une durée de plusieurs jours)**.

# Front End Angular

Dans ce pattern le **Front End** doit :

* Transmettre le cookie de session lors de chaque requête
* Gérer le retour de code **HTTP 401** exclamation

# Cookies

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Front%20End%20Angular/WebHome?section=1)

Lors de la connexion à l’application via la mire **X-Connect**, le **Back For Front** a transmis deux cookies à l’application **Front End**.

Ces cookies seront donc transmis automatiquement à chaque requête faite au **Back For Front** et vérifier par celui-ci.

# Gestion code HTTP 401

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Front%20End%20Angular/WebHome?section=2)

L’application **Front End** doit être capable de gérer les codes retour en erreur des requêtes envoyées au **Back For Front**. Dans le pattern de persistance de session, le code retour à gérer et le **HTTP 401**exclamation.

Celui-ci est renvoyé par le **Back For Front** lorsque :

* La session n’existe plus dans **Redis**
* Le **cookie STATE** a expiré (ou la clé associée n’existe plus dans Redis)
* Aucun des **cookies STATE ou SESSION\_ID** n’est présent pour accéder à une ressource sécurisée

Dans ce cas le **Front End** doit intercepter l’erreur et afficher la mire de connexion **X-Connect** pour que l’utilisateur s’authentifie.

Pour cela une classe a été créée: **src/app/services/http.service.ts**

Elle centralise la mécanique d’envoi des requêtes REST vers le **Back For Front**. Chaque service souhaitant envoyer une requête au **Back For Front** doit hériter de cette classe.

Elle contient une méthode **handleError()** qui a la charge de gérer les codes retours

http.service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

   /\*\*  
     \* Gestion des erreurs lors d'une requête  
     \* @param url Chemin de la requête  
     \* @param result Resultat de la requête  
     \*/  
    **private** handleError<T>(url = 'operation', options?: any) {  
        **return** (error: any): Observable<T> => {  
            console.warn(error.message);  
            **switch** (error.status) {  
                **case** 401:  
                    // Si on a un 401 on lance la cinématique de connexion  
                    **this**.authService.logout();  
                    **break**;  
                **case** 404:  
                    **if** (!options.accept404) {  
                        **this**.router.navigate(['/error']);  
                    }  
                    **break**;  
                **default**:  
                    **this**.router.navigate(['/error']);  
            }  
            **throw** error;  
        };  
    }

On peut voir que sur un code retour **HTTP 401**exclamation, on demande à l’**AuthService** de lancer la déconnexion :

* Demande de révocation du **refresh\_token** au **Back For Front**
* Redirection vers la mire **X-Connect**

# Front End React

Dans ce pattern le **Front End** doit :

* Transmettre le cookie de session lors de chaque requête
* Gérer le retour de code **HTTP 401** exclamation

# Cookies

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Front%20End%20React/WebHome?section=1)

Lors de la connexion à l’application via la mire **X-Connect**, le **Back For Front** a transmis deux cookies à l’application **Front End**.

Ces cookies seront donc transmis automatiquement à chaque requête faite au **Back For Front** et vérifier par celui-ci.

# Gestion code HTTP 401

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Persistance%20de%20session/Front%20End%20React/WebHome?section=2)

L’application **Front End** doit être capable de gérer les codes retour en erreur des requêtes envoyées au **Back For Front**. Dans le pattern de persistance de session, le code retour à gérer et le **HTTP 401**exclamation.

Celui-ci est renvoyé par le **Back For Front** lorsque :

* La session n’existe plus dans **Redis**
* Le **cookie STATE** a expiré (ou la clé associée n’existe plus dans Redis)
* Aucun des **cookies STATE ou SESSION\_ID** n’est présent pour accéder à une ressource sécurisée

Dans ce cas le **Front End** doit intercepter l’erreur et afficher la mire de connexion **X-Connect** pour que l’utilisateur s’authentifie.

Pour cela une classe a été créée: **src/app/services/HttpService.js**

Elle centralise la mécanique d’envoi des requêtes REST vers le **Back For Front**. Chaque service souhaitant envoyer une requête au **Back For Front** doit hériter de cette classe.

Dans la méthode **executeRequest()**, les erreurs sont **catcher**est traitées:

HttpService.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27

   /\*\*  
     \* Execution des requêtes  
     \* @param {} options Options de la requête (voir librairie axios)  
     \* @param {\*} errorOptions Options pour la gestion des erreurs  
     \*/  
    executeRequest(options, errorOptions) {  
        **return** axios(options).then(response => {  
            // Retour de la réponse  
            **return** response.data;  
        }).**catch**((error => {  
            // Gestion des cas d'erreurs  
            **switch** (error.response.status) {  
                **case** 401:  
                    **this**.authentificationService.logout();  
                    **break**;  
                **case** 404:  
                    **if** (!\_.isNull(errorOptions) && !errorOptions.accept404) {  
                        history.push('/error');  
                    }  
                    **break**;  
                **default**:  
                    console.warn(error.response.data);  
                    history.push('/error');  
            }  
            **throw** error.response.data;  
        }))  
    }

On peut voir que sur un code retour **HTTP 401**exclamation, on demande à l’**AuthentificationService** de lancer la déconnexion :

* Demande de révocation du **refresh\_token** au **Back For Front**
* Redirection vers la mire **X-Connect**

# Appel d’API

Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/WebHome?section=1)

Dans ce chapitre nous allons démontrer comment le pattern d’appel d’API a été implémenté dans l’application exemple.

Pour rappel, le but de ce pattern est de pouvoir faire appel à des API déclarées dans l’**API Manager**.

Ce pattern concerne principalement le**Back For Front** car ce sera lui qui fera l’appel aux API du SI CATS (même si les requêtes seront initialement demandée par le **Front End**).

Cinématiques

# API Places

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/API%20Places/WebHome?section=1)

L’application exemple, permet d’afficher la liste des agences du Crédit Agricole selon les caisses régionales et une ville.

Pour cela, elle s’appuie sur l’**API Places** déclarée dans l’**API Manager**.

Afin de pouvoir consommer cette API, l’application doit

* Souscrire à l’API via le portail des API.
* Appeler l’API avec l’**access\_token** via le **Back For Front**.

# Souscription à l’API

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/API%20Places/WebHome?section=2)

La souscription à une API se faire sur le portail des API :

* [Hors-Prod](https://rct-private.developer.credit-agricole.fr/store/)
* [NEHOM](https://hom-private.developer.credit-agricole.fr/store/)
* [Production](https://private.developer.credit-agricole.fr/store/)

Pour souscrire à une API il faut :

* Il faut se connecter avec le compte utilisateur qui a permis la création de l’application dans le portail.
* Rechercher l’API souhaité
  + Dans le cas de places : **Places v1.8**
* Aller dans l’onglet Souscription et sélectionner votre application
  + **CATS\_APP\_IHML-DCK** ⇔ Application **Angular**
  + **CATS\_APP\_IHML-REACT** ⇔ Application **React**
* L’API doit apparaître dans les souscriptions de votre application
* A noter que votre application doit déjà avoir souscris à l’**API Authentification Collaborateur** pour pouvoir générer les **access\_token**

# Back For Front

Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

Dans ce pattern, le **Back For Front** doit:

* Exposer une ressource pour que le **Front End** puisse appeler l’**API Places**
* Appeler l’End point de l’**API Places** en fournissant l’**access\_token** de la session

# Exposition de la ressource

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/Back%20For%20Front/WebHome?section=1)

Le **Back For Front** expose des ressources pour appeler l’API Places via le fichier: **PlacesController.java**

PlacesController.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46

@RestController  
@RequestMapping("/places")  
**public** **class** **PlacesController** {  
  
 @Autowired  
 **private** PlacesService placesService;  
   
 /\*\*  
  \* Ressource pour récupérer la liste des agences  
  \* @return Liste de {@link CR}  
  \* @throws IOException  
  \* @throws ApiException  
  \*/  
  @GetMapping("/regional\_banks")  
  **public** ResponseEntity<CR[]> getPlaces() **throws** IOException, ApiException {  
   CR[] crs = **this**.placesService.getCRList();  
   **return** ResponseEntity.ok().body(crs);  
  }  
    
  /\*\*  
   \* Ressource pour récupérer la liste des villes d'une CR  
   \* @param crId Id de la CR  
   \* @return Liste d' {@link Entity}  
   \* @throws IOException  
   \* @throws ApiException  
   \*/  
  @GetMapping("/regional\_banks/{crId}/cities\_with\_distribution\_entities")  
  **public** ResponseEntity<Entity[]> getCREntities(@PathVariable("crId") String crId) **throws** IOException, ApiException {  
   Entity[] entities = **this**.placesService.getCREntities(crId);  
   **return** ResponseEntity.ok().body(entities);  
  }  
    
  /\*\*  
   \* Ressource pour récupérer les agences d'une ville d'une CR  
   \* @param crId Id de la CR  
   \* @param zipCode Zipcode de l'Entity (ville)  
   \* @return Liste d' {@link DistributionEntity}  
   \* @throws IOException  
   \* @throws ApiException  
   \*/  
  @GetMapping("/distribution\_entities/search\_by\_city/{crId}/{zipCode}")  
  **public** ResponseEntity<DistributionEntity[]> getCRAgencesList(@PathVariable("crId") String crId, @PathVariable ("zipCode") String zipCode) **throws** IOException, ApiException {  
   DistributionEntity[] distributionEntities = **this**.placesService.getCRAgencesList(crId, zipCode);  
   **return** ResponseEntity.ok().body(distributionEntities);  
  }  
}

On peut voir que pour chaque ressource des paramètres sont nécessaires pour l’appel de l’**API Places**.

# Appel de l’API Places

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/Back%20For%20Front/WebHome?section=2)

Le contrôleur délègue l’appel de l’API Places au service **PlacesServices.java**

Exemple pour la récupération de la liste de CR:

PlacesServices.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26

/\*\*  
  \* Retourne la liste des CR  
  \* @return La liste des CR  
  \* @throws IOException  
  \* @throws ApiException  
  \* @see {@link CR}  
  \*/  
 **public** CR[] getCRList() **throws** IOException, ApiException {  
  mostLogger.debugInfo("Récupération de la list des CR");  
    
  // Construction de la requête  
  HttpGet getCRRequest = **new** HttpGet(String.format(Constants.REGINONAL\_BANKS\_URL, placesUrl));  
    
  // Set security Headers  
  **this**.addSecurityRequestHeader(getCRRequest);  
    
  // Set other headers  
  getCRRequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_UTF8\_VALUE);  
    
     // Execution de la requete  
  CR[] crs = **this**.httpService.execute(getCRRequest, CR[].class);  
       
     // Mapping de la réponse de la requête  
  mostLogger.debugInfo("Récupération de la list des CR --> OK");  
  **return** crs;  
 }

Pour chaque appel à l’**API Places** le mode opératoire est le même

* Création de la requête
  + Construction de l’URL
  + Ajout des **Headers**de sécurité ( Géré dans la classe parente **ApiService.java**)
    - **Authorization**:
      * Récupération de l’**access\_token** via la session.
    - **correlationId**
    - **cats\_consommateur**
    - **cats\_consommateur\_origine**
    - **cats\_canal**
  + Ajout du Header: Content-type
* Exécution de la requête (via la librairie **Apache http Client)**
  + Géré par le service **HttpService.java**
    - Envoi de la requête
    - Mapping de la réponse sur un Objet Java de l'application
* Envoi de la réponse au contrôleur qui la transmet au **Front End**

# Front End Angular

Dans ce pattern, le **Front End** doit :

* Envoyer une requête au **Back For Front**

# Envoi de requête au Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Appel%20d%E2%80%99API/Front%20End%20Angular/WebHome?section=1)

Afin de centraliser la façon d’envoyer des requêtes au **Back For Front**, un service **HttpService (src/app/services/http.service.ts)** a été créé

* Il expose les différents types de requêtes possibles : **GET, POST, PUT, PATCH, DELETE**
* Il gère les codes retour d’erreur

Les services ayant besoin d’envoyer des requêtes au **Back For Front** vont ainsi utiliser les méthodes ce service.

Pour exemple voici le **PlacesServices (src/app/services/places.service.ts)** qui envoi des demandes pour requêter l’**API Places**

places.service.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34

/\*\*  
 \* Service pour la gestion des appels aux API places  
 \*/  
@Injectable()  
export **class** **PlacesService** {  
    constructor(  
        **private** httpService: HttpService  
    ) { }  
  
    /\*\*  
     \* Retourne la list des CR  
     \*/  
    getCRList() {  
        **return** **this**.httpService.get(`/places/regional\_banks`);  
    }  
  
    /\*\*  
     \* Retourne la liste des villes d'une CR  
     \* @param crId L'id de la CR  
     \*/  
    getCREntities(crId: string) {  
        **return** **this**.httpService.get(`/places/regional\_banks/${crId}/cities\_with\_distribution\_entities`);  
    }  
  
    /\*\*  
     \* Retourne la liste des agences d'une ville d'une CR  
     \* @param crId L'Id de la CR  
     \* @param zipCode La code postal de la ville  
     \*/  
    getCRAgencesList(crId: string, zipCode: string) {  
        **return** **this**.httpService.get(`/places/distribution\_entities/search\_by\_city/${crId}/${zipCode}`);  
    }  
}

# Front End React

Dans ce pattern le **Front End** doit :

* Gérer la récupération d'un contexte
* Sauvegarder un contexte pour l'envoyer à une autre application

Dans le cas de notre application, la partage de contexte permet d'afficher la liste des agences d'un Caisse Régionale, qui aura précédemment choisie dans l'application **Angular**

# Récupération de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=1)

Si un contexte doit être récupéré par l'application, cela sera indiqué par un paramètre dans l'url d'appel de l'application.

* Exemple: https://ihmlang.ca-devtu-zudb0.credit-agricole.fr/ihml-demo-react/agences?**idCtx=0f87eec2-4a91-49e0-9f02-aef37d0741fa**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **idCtx** | Query Param | Id du contexte à récupérer |

## **Sécurité**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=2)

Afin de récupérer un contexte, l'utilisateur doit obligatoire être authentifié. Cette vérification est fait dans le composant**PrivateRoute(src/app/components/route/PrivateRoute.js)**

PrivateRoute.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33

/\*\*  
     \* Rendu du composant  
     \*/  
    render() {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = getPathInfos(**this**.props.history.location.search);  
  
        // Test si on n'a pas de context Id et si la route demandée n'est pas sécurisée  
        **const** noContextNoSecure = !**this**.props.isSecure && !\_.has(queryParamsMap, Constants.ROUTES.PARAMS.ID\_CONTEXT);  
  
        // Récupération de la page demandée  
        **if** (!\_.isEmpty(**this**.props.user) || noContextNoSecure) {  
            // Si un utilisateur est connecté on affiche le composant principal de la page demandée  
            **return** <Route path={**this**.props.path} component={**this**.props.component}/>  
        } **else** {  
            **let** redirectTo = **this**.props.history.location.pathname;  
  
            // Si aucun utilisateur n'est connecté  
            **if** (!\_.isEmpty(**this**.props.history.location.search)) {  
                redirectTo = `**${**redirectTo**}${this**.props.history.location.search**}**`;  
            }  
  
            // Gestion de la connexion avec X Connect  
            **this**.authentificationService.login(redirectTo);  
  
            // On affiche un spinner  
            **return** <div className="loader">  
                <div className="preloader">  
                    <CircularProgress size={100}/>  
                </div>  
            </div>  
        }  
    }

On peut voir que si un contexte est demandé et que l'utilisateur n'est pas authentifié, alors on redirige l'utilisateur vers la mire de connexion **X-Connect** ([pattern de connexion](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/)).

## **Récupération du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=3)

Une fois authentifié, il est possible de récupérer le contexte souhaité.

Cette partie est gérée par le composant principal de l'application: **src/app/components/container/Container.js**

Container.js

1  
2  
3  
4  
5  
6  
7  
8

    componentDidUpdate() {  
        // Chargement du contexte si on est connecté  
        **const** queryParamMaps = getPathInfos(**this**.props.history.location.search);  
        **if** (!**this**.state.contextLoaded && !\_.isNull(**this**.props.user) && \_.has(queryParamMaps, Constants.ROUTES.PARAMS.ID\_CONTEXT)) {  
            **this**.props.getContext(queryParamMaps[Constants.ROUTES.PARAMS.ID\_CONTEXT]);  
            **this**.setState({ contextLoaded: **true** })  
        }  
    }

Lors de son initialisation:

* On teste que le contexte n'a pas encore été chargé
* On teste que l'utilisateur est bien authentifié (user existe)
* On teste si le paramètre de requête **idCtx**existe
  + S'il existe on demande à récupérer le contexte
  + Cette action est dans le Thunk: **src/app/redux/thunks/ContextThunk.js**

ContexteThunk.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20

/\*\*  
 \* Thunk pour setter l'environnement de l'application  
 \*/  
**export** **const** getContext = (idCtx) => {  
    **return** (dispatch) => {  
        **const** contextService = iocContainer.get(TYPES.ContextService);  
        // Chargement du context  
        **return** contextService.getContext(idCtx).then((context) => {  
            dispatch(setContext(context))  
        }).**catch**((error) => {  
            **if** (error.status === 404) {  
                dispatch(setContext({  
                    regional\_bank\_id: **null**,  
                    zip\_code: **null**  
                }));  
            }  
            **return** Promise.resolve(**null**);  
        })  
    }  
}

La méthode:

* Appelle la méthode **getContext()** du **ContextService (src/app/services/ContextService.js)**
* Sauvegarde le contexte dans le store **Redux**
* Dans le cas où le contexte n'existe pas on sauvegarde un contexte par défaut (spécifique à l'application)

## **Utilisation du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=4)

L'utilisation du contexte est faite dans le composant **src/app/components/search-agence/root/SearchAgenceRoot.js**

SearchAgenceRoot.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31

    componentDidUpdate(prevProps) {  
        // Selection de la cr par contexte  
        **if** (!\_.isEmpty(**this**.props.crs) && !\_.isNull(**this**.props.context.regional\_bank\_id) && !**this**.state.context.crSelected) {  
            **this**.props.getCREntities(**this**.props.context.regional\_bank\_id)  
            **this**.props.setSelectedCr(**this**.props.context.regional\_bank\_id);  
              
            // Mise à jour du state - la cr a été séléctionnée par le contexte  
            **const** context = {  
                crSelected: **true**,  
                entitySelected: **false**  
            }              
            **this**.setState({context});  
  
        }  
  
        // Selection de la ville par contexte  
        **if** (!\_.isEmpty(**this**.props.entities) && !\_.isNull(**this**.props.context.zip\_code) && !**this**.state.context.entitySelected) {  
            **this**.props.getCRAgencesList(**this**.props.context.regional\_bank\_id, **this**.props.context.zip\_code);  
            **this**.props.setSelectedEntityByZipCode(**this**.props.context.zip\_code);  
              
            // Mise à jour du state - la ville a été séléctionnée par le contexte  
            **const** context = {  
                crSelected: **true**,  
                entitySelected: **true**  
            }              
            **this**.setState({context});  
  
            // Reset du context une fois celui-ci chargé  
            **this**.props.setContext({regional\_bank\_id: **null**, zip\_code: **null**});  
        }  
    }

La récupération se fait en écoutant le store **Redux**.

L'exploitation du contexte ainsi récupéré est spécifique à l'application et ne sera donc pas détaillé ici.

# Sauvegarde de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=5)

Pour transmettre, un contexte à une autre application, il faut:

* Sauvegarder le contexte via le **Back For Front**
* Transmettre l'id de contexte récupéré à la nouvelle application

## **Sauvegarde du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=6)

La sauvegarde du contexte se fait:

* En construisant une chaîne représentant le contexte à sauvegarder (chaîne JSON par exemple)
* En appelant la ressource **Back For Front** en charge de la sauvegarde du contexte

Dans notre application cela est démontré dans le fichier **src/app/components/search-agence/root/SearchAgenceRoot.js**

SearchAgenceRoot.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

   /\*\*  
     \* Listener pour ouvrir l'application Angular avec un context  
     \*/  
    onOpenAngularApplication() {  
        **const** context = {  
            client\_id: **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.ANGULAR\_CLIENT\_ID),  
            regional\_bank\_id: **this**.props.selectedCR.regional\_bank\_id,  
            zip\_code: **this**.props.selectedEntity.zip\_code  
        }  
  
        // sauvegarde du context puis on ouvre l'application angular  
        **this**.contextService.setContext(context).then((context) => {  
            **const** angularUrl = `**${this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.ANGULAR\_URL)**}**/**${**Constants.ROUTES.ANUGLAR.AGENCES**}**`  
            window.open(`**${**angularUrl**}**?idCtx=**${**context.context\_id**}**`);  
        })  
    }

On peut voir que lorsque l'utilisateur clique sur le lien **Ouvrir application React**:

* Un contexte est créé avec la CR et la ville sélectionnée
* Un appel à la méthode **setContext()**du **ContexteService** est fait
  + Appel a la ressource **/context (POST)**du **Back For Front**
* En retour de l'appel de l'API, on demande à ouvrir l'URL de l'application **Angular**en rajoutant le paramètre **idCtx**reçu du **Back For Front**

# Transfert de contexte

# Introduction

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/WebHome?section=1)

## **But**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/WebHome?section=2)

Le but de ce pattern est de pouvoir transférer du contexte applicatif entre:

* 2 applications IHM Light

Pour cela nous allons utiliser l'**API CTX9** (partage de contexte) mis à disposition par les équipes **X-Connect**.

## **Pré-requis**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/WebHome?section=3)

Afin d'utiliser l'**API CTX9**, il est nécessaire de:

* Souscrire à l'**API CTX9** - Partage de contexte sur le portail des API pour chaque application
* Etre authentifier en tant qu'utilisateur au près d'une des mires **X-Connect**, afin d'avoir un **access\_token** permettant l'accès aux API.

# Cinématiques

# API Partage de contexte (CTX9)

# API Partage de Contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/API%20CTX9/WebHome?section=1)

L’application exemple, permet de démontrer le partage de contexte entre applications.

Pour cela, elle s’appuie sur l’**API CTX9** déclarée dans l’**API Manager**.

Afin de pouvoir consommer cette API, l’application doit

* Souscrire à l’API via le portail des API.
* Appeler l’API avec l’**access\_token** via le **Back For Front**.

# Souscription à l’API

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/API%20CTX9/WebHome?section=2)

La souscription à une API se faire sur le portail des API :

* [Hors-Prod](https://rct-private.developer.credit-agricole.fr/store/)
* [NEHOM](https://hom-private.developer.credit-agricole.fr/store/)
* [Production](https://private.developer.credit-agricole.fr/store/)

Pour souscrire à une API il faut :

* Il faut se connecter avec le compte utilisateur qui a permis la création de l’application dans le portail.
* Rechercher l’API souhaité
  + Dans le cas de CTX9 : **PartageContexte v1.0**
* Aller dans l’onglet Souscription et sélectionner votre application
  + **CATS\_APP\_IHML-DCK** ⇔ Application **Angular**
  + **CATS\_APP\_IHML-REACT** ⇔ Application **React**
* L’API doit apparaître dans les souscriptions de votre application
* A noter que votre application doit déjà avoir souscris à l’**API Authentification Collaborateur** pour pouvoir générer les **access\_token**

# Fonctionnement

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/API%20CTX9/WebHome?section=3)

## **Ressources**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/API%20CTX9/WebHome?section=4)

L'**API CTX9** expose deux ressources

* Sauvegarde de contexte (**POST**)
  + **CTX9**sauvegarde le contexte dans une base Redis
  + **CTX9**renvoi un id de contexte
* Récupération de contexte (**GET**)
  + **CTX9**recherche dans sa base Redis l'id de contexte demandé

## **Persistance**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/API%20CTX9/WebHome?section=5)

* **Le temps de persistance d'un contexte dans le cache Redis est court. 1 minute max**
* **Une fois que le contexte a été récupéré, il est supprimé de la base Redis et il n'est plus possible de le récupérer à nouveau**

# Back For Front

Cette implémentation du **Back For Front** est valide pour les deux applications

* Angular
* React

 Dans ce pattern le **Back For Front** doit:

* Exposer deux ressources pour **sauvegarder**et **récupérer**le contexte
* Appeler l'**API CTX9** pour sauvegarder ou récupérer le contexte

Dans notre application exemple cela est fait dans le fichier **ContextController.java**

# Sauvegarde du contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Back%20For%20Front/WebHome?section=1)

ContextController.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

/\*\*  
  \* Ressource pour récupérer le context de l'application  
  \*   
  \* @param ctxId  
  \*            Id du contexte à récupérer  
  \*   
  \* @return {@link ApplicationContext}  
  \* @throws IOException  
  \* @throws ApiException  
  \*/  
 @PostMapping()  
 **public** ResponseEntity<Ctx9WriteContext> setContext(@RequestBody ApplicationContext applicationContext) **throws** IOException, ApiException {  
  Ctx9WriteContext ctx9WriteContext = **this**.contextService.setContext(applicationContext);  
  **return** ResponseEntity.ok().body(ctx9WriteContext);  
 }

On peut voir que la sauvegarde du contexte attend le paramètre suivant:

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **applicationContexte** | Body | Cet objet représente le contexte à sauvegarder. C'est un objet propre à notre application. Il sera différend selon les applications. |

Le controller délègue ensuite la gestion de la sauvegarde au **ContexteService.java**

ContexteService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41

/\*\*  
  \* Retourne le contexte de l'application  
  \*   
  \* @param ctxId  
  \*            Id du contexte  
  \* @return Le contexte de l'application  
  \* @throws IOException  
  \* @throws ApiException  
  \* @see {@link ApplicationContext}  
  \*/  
 public Ctx9WriteContext setContext(ApplicationContext applicationContext)  
   throws IOException, ApiException {  
  mostLogger.debugInfo("Sauvegarde du contexte");  
  
  // Construction de la requête  
  HttpPost postContextTequest = new HttpPost(String.format(Constants.POST\_CONTEXT\_URL, ctx9Url));  
    
    
  // Add security headers  
  this.addSecurityRequestHeader(postContextTequest);    
    
  // Set Headers  
  postContextTequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_FORM\_URLENCODED\_VALUE);  
  
  // Set Body  
  List<NameValuePair> params = new ArrayList<>();  
  params.add(new BasicNameValuePair(Constants.USER\_CONTEXT\_PARAM, applicationContext.toJsonString()));  
  
  **if** (Objects.nonNull(applicationContext.getClientId())) {  
   params.add(new BasicNameValuePair(Constants.CLIENT\_ID\_PARAM, applicationContext.getClientId()));  
  }  
  postContextTequest.setEntity(new UrlEncodedFormEntity(params));  
  
  // Execution de la requete  
  Ctx9WriteContext writeContext = this.httpService.execute(postContextTequest, Ctx9WriteContext.class);  
  
  // Mapping de la réponse de la requête  
  mostLogger.debugInfo("Sauvegarde du contexte --> OK");  
  **return** writeContext;  
 }

La méthode:

* Créer la requête
  + Construction de l’URL
  + Ajout des **Headers**de sécurité ( Géré dans la classe parente **ApiService.java**)
    - **Authorization**:
      * Récupération de l’**access\_token** via la session.
    - **correlationId**
    - **cats\_consommateur**
    - **cats\_consommateur\_origine**
    - **cats\_canal**
  + Ajout du **Header**: **Content-type**
  + Ajout des paramètres de la requête CTX9

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **user\_context** | Body (form) | Cet objet représente le contexte à sauvegarder. C'est une string. Dans notre application nous créons, une chaine JSON. |
| **client\_id** | Body (form) | Le clientId de l'application destinée à récupérer le contexte |

* Exécution de la requête (via la librairie **Apache http Client**)
  + Géré par le service **HttpService.java**
    - Envoi de la requête
    - Mapping de la réponse sur un Objet Java de l'application
* Envoi de la réponse au contrôleur qui la transmet au **Front End**
  + Le **Front End** reçoit un Id de contexte généré par l'**API CTX9**

# Récupération du contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Back%20For%20Front/WebHome?section=2)

ContexteController.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

/\*\*  
  \* Ressource pour récupérer le context de l'application  
  \*   
  \* @param ctxId  
  \*            Id du contexte à récupérer  
  \*   
  \* @return {@link ApplicationContext}  
  \* @throws IOException  
  \* @throws ApiException  
  \*/  
 @GetMapping("/{ctxId}")  
 **public** ResponseEntity<ApplicationContext> getContext(@PathVariable("ctxId") String ctxId) **throws** IOException, ApiException {  
  ApplicationContext applicationContext = **this**.contextService.getContext(ctxId);  
  **return** ResponseEntity.ok().body(applicationContext);  
 }

On peut voir que la sauvegarde du contexte attend le paramètre suivant:

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **ctxId** | Path Param | L'id du contexte a récupérer. |

Le controller délègue ensuite la gestion de la récupération au **ContexteService.java**

ContexteService.java

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29

/\*\*  
  \* Retourne le contexte de l'application  
  \*   
  \* @param ctxId  
  \*            Id du contexte  
  \* @return Le contexte de l'application  
  \* @throws IOException  
  \* @throws ApiException  
  \* @see {@link ApplicationContext}  
  \*/  
 **public** ApplicationContext getContext(String ctxId) **throws** IOException, ApiException {  
  mostLogger.debugInfo(String.format("Récupération du context: %s", ctxId));  
  
  // Construction de la requête  
  HttpGet getContextTequest = **new** HttpGet(String.format(Constants.GET\_CONTEXT\_URL, ctx9Url, ctxId));  
    
  // Add security headers  
  **this**.addSecurityRequestHeader(getContextTequest);  
  
  // Set Headers  
  getContextTequest.addHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE);  
  
  // Execution de la requete  
  Ctx9ReadContext ctx9Context = **this**.httpService.execute(getContextTequest, Ctx9ReadContext.class);  
  
  // Mapping de la réponse de la requête  
  mostLogger.debugInfo(String.format("Récupération du context: %s --> OK", ctxId));  
  **return** ctx9Context.toApplicationContext();  
 }

La méthode:

* Créer la requête
  + Construction de l’URL
  + Ajout des **Headers**de sécurité ( Géré dans la classe parente **ApiService.java**)
    - **Authorization**:
      * Récupération de l’**access\_token** via la session.
    - **correlationId**
    - **cats\_consommateur**
    - **cats\_consommateur\_origine**
    - **cats\_canal**
  + Ajout du **Header**: **Content-typeamètres**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **id\_contexte** | Path Param | L'id du contexte a récupérer. |

* Exécution de la requête (via la librairie **Apache http Client**)
  + Géré par le service **HttpService.java**
    - Envoi de la requête
    - Mapping de la réponse sur un Objet Java de l'application
* Envoi de la réponse au contrôleur qui la transmet au **Front End**
  + Le **Front End** reçoit le contexte correspondant à l'ID

# Front End React

Dans ce pattern le **Front End** doit :

* Gérer la récupération d'un contexte
* Sauvegarder un contexte pour l'envoyer à une autre application

Dans le cas de notre application, la partage de contexte permet d'afficher la liste des agences d'un Caisse Régionale, qui aura précédemment choisie dans l'application **Angular**

# Récupération de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=1)

Si un contexte doit être récupéré par l'application, cela sera indiqué par un paramètre dans l'url d'appel de l'application.

* Exemple: https://ihmlang.ca-devtu-zudb0.credit-agricole.fr/ihml-demo-react/agences?**idCtx=0f87eec2-4a91-49e0-9f02-aef37d0741fa**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **idCtx** | Query Param | Id du contexte à récupérer |

## **Sécurité**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=2)

Afin de récupérer un contexte, l'utilisateur doit obligatoire être authentifié. Cette vérification est fait dans le composant**PrivateRoute(src/app/components/route/PrivateRoute.js)**

PrivateRoute.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33

/\*\*  
     \* Rendu du composant  
     \*/  
    render() {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = getPathInfos(**this**.props.history.location.search);  
  
        // Test si on n'a pas de context Id et si la route demandée n'est pas sécurisée  
        **const** noContextNoSecure = !**this**.props.isSecure && !\_.has(queryParamsMap, Constants.ROUTES.PARAMS.ID\_CONTEXT);  
  
        // Récupération de la page demandée  
        **if** (!\_.isEmpty(**this**.props.user) || noContextNoSecure) {  
            // Si un utilisateur est connecté on affiche le composant principal de la page demandée  
            **return** <Route path={**this**.props.path} component={**this**.props.component}/>  
        } **else** {  
            **let** redirectTo = **this**.props.history.location.pathname;  
  
            // Si aucun utilisateur n'est connecté  
            **if** (!\_.isEmpty(**this**.props.history.location.search)) {  
                redirectTo = `**${**redirectTo**}${this**.props.history.location.search**}**`;  
            }  
  
            // Gestion de la connexion avec X Connect  
            **this**.authentificationService.login(redirectTo);  
  
            // On affiche un spinner  
            **return** <div className="loader">  
                <div className="preloader">  
                    <CircularProgress size={100}/>  
                </div>  
            </div>  
        }  
    }

On peut voir que si un contexte est demandé et que l'utilisateur n'est pas authentifié, alors on redirige l'utilisateur vers la mire de connexion **X-Connect** ([pattern de connexion](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End%20React/)).

## **Récupération du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=3)

Une fois authentifié, il est possible de récupérer le contexte souhaité.

Cette partie est gérée par le composant principal de l'application: **src/app/components/container/Container.js**

Container.js

1  
2  
3  
4  
5  
6  
7  
8

    componentDidUpdate() {  
        // Chargement du contexte si on est connecté  
        **const** queryParamMaps = getPathInfos(**this**.props.history.location.search);  
        **if** (!**this**.state.contextLoaded && !\_.isNull(**this**.props.user) && \_.has(queryParamMaps, Constants.ROUTES.PARAMS.ID\_CONTEXT)) {  
            **this**.props.getContext(queryParamMaps[Constants.ROUTES.PARAMS.ID\_CONTEXT]);  
            **this**.setState({ contextLoaded: **true** })  
        }  
    }

Lors de son initialisation:

* On teste que le contexte n'a pas encore été chargé
* On teste que l'utilisateur est bien authentifié (user existe)
* On teste si le paramètre de requête **idCtx**existe
  + S'il existe on demande à récupérer le contexte
  + Cette action est dans le Thunk: **src/app/redux/thunks/ContextThunk.js**

ContexteThunk.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20

/\*\*  
 \* Thunk pour setter l'environnement de l'application  
 \*/  
**export** **const** getContext = (idCtx) => {  
    **return** (dispatch) => {  
        **const** contextService = iocContainer.get(TYPES.ContextService);  
        // Chargement du context  
        **return** contextService.getContext(idCtx).then((context) => {  
            dispatch(setContext(context))  
        }).**catch**((error) => {  
            **if** (error.status === 404) {  
                dispatch(setContext({  
                    regional\_bank\_id: **null**,  
                    zip\_code: **null**  
                }));  
            }  
            **return** Promise.resolve(**null**);  
        })  
    }  
}

La méthode:

* Appelle la méthode **getContext()** du **ContextService (src/app/services/ContextService.js)**
* Sauvegarde le contexte dans le store **Redux**
* Dans le cas où le contexte n'existe pas on sauvegarde un contexte par défaut (spécifique à l'application)

## **Utilisation du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=4)

L'utilisation du contexte est faite dans le composant **src/app/components/search-agence/root/SearchAgenceRoot.js**

SearchAgenceRoot.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31

    componentDidUpdate(prevProps) {  
        // Selection de la cr par contexte  
        **if** (!\_.isEmpty(**this**.props.crs) && !\_.isNull(**this**.props.context.regional\_bank\_id) && !**this**.state.context.crSelected) {  
            **this**.props.getCREntities(**this**.props.context.regional\_bank\_id)  
            **this**.props.setSelectedCr(**this**.props.context.regional\_bank\_id);  
              
            // Mise à jour du state - la cr a été séléctionnée par le contexte  
            **const** context = {  
                crSelected: **true**,  
                entitySelected: **false**  
            }              
            **this**.setState({context});  
  
        }  
  
        // Selection de la ville par contexte  
        **if** (!\_.isEmpty(**this**.props.entities) && !\_.isNull(**this**.props.context.zip\_code) && !**this**.state.context.entitySelected) {  
            **this**.props.getCRAgencesList(**this**.props.context.regional\_bank\_id, **this**.props.context.zip\_code);  
            **this**.props.setSelectedEntityByZipCode(**this**.props.context.zip\_code);  
              
            // Mise à jour du state - la ville a été séléctionnée par le contexte  
            **const** context = {  
                crSelected: **true**,  
                entitySelected: **true**  
            }              
            **this**.setState({context});  
  
            // Reset du context une fois celui-ci chargé  
            **this**.props.setContext({regional\_bank\_id: **null**, zip\_code: **null**});  
        }  
    }

La récupération se fait en écoutant le store **Redux**.

L'exploitation du contexte ainsi récupéré est spécifique à l'application et ne sera donc pas détaillé ici.

# Sauvegarde de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=5)

Pour transmettre, un contexte à une autre application, il faut:

* Sauvegarder le contexte via le **Back For Front**
* Transmettre l'id de contexte récupéré à la nouvelle application

## **Sauvegarde du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20React/WebHome?section=6)

La sauvegarde du contexte se fait:

* En construisant une chaîne représentant le contexte à sauvegarder (chaîne JSON par exemple)
* En appelant la ressource **Back For Front** en charge de la sauvegarde du contexte

Dans notre application cela est démontré dans le fichier **src/app/components/search-agence/root/SearchAgenceRoot.js**

SearchAgenceRoot.js

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

   /\*\*  
     \* Listener pour ouvrir l'application Angular avec un context  
     \*/  
    onOpenAngularApplication() {  
        **const** context = {  
            client\_id: **this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.ANGULAR\_CLIENT\_ID),  
            regional\_bank\_id: **this**.props.selectedCR.regional\_bank\_id,  
            zip\_code: **this**.props.selectedEntity.zip\_code  
        }  
  
        // sauvegarde du context puis on ouvre l'application angular  
        **this**.contextService.setContext(context).then((context) => {  
            **const** angularUrl = `**${this**.configurationService.get(Constants.APPLICATION.CONFIGURATION.ANGULAR\_URL)**}**/**${**Constants.ROUTES.ANUGLAR.AGENCES**}**`  
            window.open(`**${**angularUrl**}**?idCtx=**${**context.context\_id**}**`);  
        })  
    }

On peut voir que lorsque l'utilisateur clique sur le lien **Ouvrir application React**:

* Un contexte est créé avec la CR et la ville sélectionnée
* Un appel à la méthode **setContext()**du **ContexteService** est fait
  + Appel a la ressource **/context (POST)**du **Back For Front**
* En retour de l'appel de l'API, on demande à ouvrir l'URL de l'application **Angular**en rajoutant le paramètre **idCtx**reçu du **Back For Front**

# Front End Angular

Dans ce pattern le **Front End** doit :

* Gérer la récupération d'un contexte
* Sauvegarder un contexte pour l'envoyer à une autre application

Dans le cas de notre application, la partage de contexte permet d'afficher la liste des agences d'un Caisse Régionale, qui aura précédemment choisie dans l'application **React**

# Récupération de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=1)

Si un contexte doit être récupéré par l'application, cela sera indiqué par un paramètre dans l'url d'appel de l'application.

* Exemple: https://ihmlang.ca-devtu-zudb0.credit-agricole.fr/ihml-demo/agences?**idCtx=0f87eec2-4a91-49e0-9f02-aef37d0741fa**

|  |  |  |
| --- | --- | --- |
| **Paramètres** | **Type** | **Détails** |
| **idCtx** | Query Param | Id du contexte à récupérer |

## **Sécurité**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=2)

Afin de récupérer un contexte, l'utilisateur doit obligatoire être authentifié. Cette vérification est fait dans **Guard Angular**: **src/auth/auth.guard.ts**

auth.guard.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20

   /\*\*  
     \* Méthode pour détermine l'accès à une page ou non  
     \* @param route La route à accèder  
     \* @param state Information sur la route  
     \*/  
    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): **boolean** {  
        // Récupération des paramètres de la requête  
        **const** queryParamsMap = **this**.getPathInfos(location.search).searchParams;  
  
        // Test si on n'a pas de context Id et si la route demandée n'est pas sécurisée  
        **const** noContextNoSecure = !route.data.isSecure && !\_.has(queryParamsMap, Constants.ROUTES.PARAMS.ID\_CONTEXT);  
         **if** (**this**.authService.authenticated || noContextNoSecure) {  
            // si on est authentifié ou si on n'a pas besoin de sécurité. On donne accès à la page  
            **return** **true**;  
        } **else** {  
            // Si ce n'est pas le cas on demande à effectué la cinématique de login  
            **this**.authService.login(state.url);  
            **return** **false**;  
        }  
    }

On peut voir que si un contexte est demandé et que l'utilisateur n'est pas authentifié, alors on redirige l'utilisateur vers la mire de connexion **X-Connect** ([pattern de connexion](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/introduction/Front%20End/)).

## **Récupération du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=3)

Une fois authentifié, il est possible de récupérer le contexte souhaité.

Cette partie est gérée par le composant principal de l'application: **src/app/components/container/container.component.ts**

container.component.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

    ngOnInit() {  
       ...  
        // Récupération des query params  
        **this**.route.queryParamMap.subscribe(params => {  
            **this**.idCtx = params.get(Constants.ROUTES.PARAMS.ID\_CONTEXT) || **null**;  
  
            // Récupération du context si besoin  
            **if** (!\_.isEmpty(**this**.idCtx)) {  
                **this**.store.dispatch(ContextActions.getContext({ idCtx: **this**.idCtx }));  
            }  
        });  
  
        ...  
    }

Lors de son initialisation:

* On teste si le paramètre de requête **idCtx**existe
  + S'il existe on demande à récupérer le contexte
  + Cette action est dans le fichier: **src/app/redux/effects/context.effects.ts**

context.effects.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

   /\*\*  
     \* Récupération du context de l'application  
     \*/  
    getApplicationContext$ = createEffect(() => **this**.actions$.pipe(  
        ofType(ContextActions.getContext),  
        exhaustMap(action => {  
            **return** **this**.contextService.getContext(action.idCtx).pipe(  
                switchMap((context: Context) => {  
                    **return** [  
                        ContextActions.setContext({context}),  
                    ];  
                }),  
                catchError(err => {  
                    **const** context: Context = {  
                        regional\_bank\_id: **null**,  
                        zip\_code: **null**  
                    }  
                    **return** [  
                        ContextActions.setContext({context})  
                    ];  
                })  
            );  
        })  
    ));

La méthode:

* Appelle la méthode **getContext()** du **ContextService (src/app/services/context.service.ts)**
* Sauvegarde le contexte dans le store **Redux**

## **Utilisation du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=4)

L'utilisation du contexte est faite dans le composant **src/app/components/search-agence/root/search-agence.component.ts**

search-agence.component.ts

1  
2  
3  
4  
5  
6  
7  
8  
9

ngOnInit() {  
...  
  
        // Initialisation redux  
        **this**.subs.push(**this**.store.pipe(select(ContextSelectors.selectContextState)).subscribe((context: Context) => {  
            **this**.context = context;  
        }));  
        ...  
    }

La récupération se fait en écoutant le store **Redux**.

L'exploitation du contexte ainsi récupéré est spécifique à l'application et ne sera donc pas détaillé ici.

# Sauvegarde de contexte

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=5)

Pour transmettre, un contexte à une autre application, il faut:

* Sauvegarder le contexte via le **Back For Front**
* Transmettre l'id de contexte récupéré à la nouvelle application

## **Sauvegarde du contexte**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Impl%C3%A9mentation%20des%20patterns/Transfert%20de%20contexte/Front%20End%20Angular/WebHome?section=6)

La sauvegarde du contexte se fait:

* En construisant une chaîne représentant le contexte à sauvegarder (chaîne JSON par exemple)
* En appelant la ressource **Back For Front** en charge de la sauvegarde du contexte

Dans notre application cela est démontré dans le fichier **src/app/components/search-agence/root/search-agence.component.ts**

search-agence.component.ts

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

   /\*\*  
     \* Listener sur le lien pour ouvrir l'application React  
     \*/  
    onOpenReactApplication() {  
  
        // Sauvegarde du contexte  
        **const** context: Context = {  
            client\_id: environment.reactClientId,  
            regional\_bank\_id: **this**.selectedCR.regional\_bank\_id,  
            zip\_code: `**${this**.selectedEntity.zip\_code**}**`  
        };  
  
        **this**.contextService.setContext(context).subscribe((savedContext: SavedContext) => {  
            window.open(`**${**environment.reactUrl**}**/**${**Constants.ROUTES.REACT.AGENCES**}**?idCtx=**${**savedContext.context\_id**}**`);  
        });  
    }

On peut voir que lorsque l'utilisateur clique sur le lien **Ouvrir application React**:

* Un contexte est créé avec la CR et la ville sélectionnée
* Un appel à la méthode **setContext()**du **ContexteService** est fait
  + Appel a la ressource **/context (POST)**du **Back For Front**
* En retour de l'appel de l'API, on demande à ouvrir l'URL de l'application **React**en rajoutant le paramètre **idCtx**reçu du **Back For Front**

# CI - Intégration continue

Le process d'intégration continu est valide pour les deux applications

* Angular
* React

La CI va nous permettre de :

1. Compiler les binaires de nos applications
2. Créer les images **Docker**de nos applications

# Construction du binaire Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=1)

La construction du binaire **Back For Front** se fait via la **PIC**dans **Eclipse Kepler (ou Neon)**. Comme pour un projet war classique, il faut

* Distribuer les modifications sur le flux
* Exécuter une génération d’assemblage
  + Pour cela un Wilot est nécessaire (voir documentation JAZZ)

Une fois la génération réalisée, le binaire est déployé dans Nexus. Il pourra être récupéré dans l’image **Docker**.

# Construction du binaire Front End

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=2)

La construction du binaire **Front End** se fait via la **PIC**dans **Jenkins**.

Pour pouvoir construire un binaire via **Jenkins**il faut :

* Faire une demande de création de job **Jenkins**aux équipes **PFAB via une demande de support JAZZ / Outils Jenkins**
* Spécifier le projet **Git (projet front end)**sur lequel le job **Jenkins**devra se reposer

Un Guide Utilisateur PIC Javascript est disponible sur le portail documentation : [ici](https://spe1.collab.ca-technologies.credit-agricole.fr/produits/d2p12/P0100/Guides/Guide%20utilisateur%20PIC%20JavaScript.docx?Web=1)

Une fois le job **Jenkins**exécuté avec succès deux binaires sont déployés

* Un binaire **NPM**dans la **registry NPM CATS**
* Une **archive tgz (Maven)** dans le **repository Nexus**
  + L’archive sera utilisée dans l’image **Docker**.

# Génération image docker

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=3)

Maintenant que les binaires du **Back For Front** et du **Front End** sont déployés, nous allons pouvoir les embarqués dans des images **Docker**.

La gestion des projets de type image **Docker**se fait dans **Eclipse Si Client** (dans les raccourcis de la plateforme de fabrication)

## **Back For Front**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=4)

Le paramétrage de la PIC oblige à créer deux images pour pouvoir exposer le war du **Back For Front**

* Image **Tomcat 8** qui embarque le war
* Image **Apache**qui va exposer le **Tomcat**sur le réseau

### **Image Tomcat 8**

L’application **Back For Front** étant codée en **Java 8**, il a été fait le choix de s’appuyer sur un **Tomcat 8**. Le projet de l’image **Back For Front** **Tomcat 8** est hébergé sur RTC

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Docker**
* Composant : **im\_app\_ihml\_tom\_tom8** **(à noter que le suffixe tom\_tom8 est obligatoire pour la CD)**

La création du projet s’est faite en utilisant le Wizard pour Projet Image Docker (INDUS)

Pour récupérer le war du **Back For Front** il faut déclarer la dépendance **Maven**dans le fichier **pom.xml**

pom.xml

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16

**<dependencies>**  
  **<dependency>**  
   **<groupId>**fr.ca.cat.webapp**</groupId>**  
   **<artifactId>**WebApp\_ihml\_bff\_java**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**war**</type>**  
   **<classifier>**gen**</classifier>**  
  **</dependency>**  
  **<dependency>**  
   **<groupId>**fr.ca.cat.webapp**</groupId>**  
   **<artifactId>**WebApp\_ihml\_bff\_java\_react**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**war**</type>**  
   **<classifier>**gen**</classifier>**  
  **</dependency>**  
 **</dependencies>**

Puis pour injecter le war dans l’image Docker, il faut modifier le fichier DockerFile

DockerFile

1  
2  
3  
4  
5  
6

**FROM** ttp10-dckreg.ca-technologies.fr/ind/im\_ind\_tom\_tom8:1.0.9  
  
MAINTAINER "ET02720 <ET02720@ca-ts.fr>"  
  
RUN mv /opt/appl/WebApp\_ihml\_bff\_java-\*.war /opt/appl/WebApp\_ihml\_bff\_java.war  
RUN mv /opt/appl/WebApp\_ihml\_bff\_java\_react-\*.war /opt/appl/WebApp\_ihml\_bff\_java\_react.war

On peut voir que l’on renomme les war **WebApp\_ihml\_java.war** et **WebApp\_ihml\_java\_react.war**. Ceci permet de spécifier le chemin de contexte de notre **Back For Front** dans **Tomcat**

* Chemin de contexte: **/WebApp\_ihml\_java** ⇔ **Angular**
* Chemin de contexte: **/WebApp\_ihml\_bff\_java\_react** ⇔ **React**

### **Image Apache mjk0**

Pour pouvoir exposer le **Tomcat**à l’extérieur du **Docker Engine**, il est nécessaire d’avoir également une image **Apache**avec le module **ModJk**installé. Le module **ModJk** permet d’établie une communication avec le protocole **AJP**entre **l’Apache**et le **Tomcat**.

Le déploiement combiné de ces deux images via une **stack docker** ([voir CD](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/D%C3%A9ploiement%20continue%20CD/)) permettra l’exposition du **Tomcat**et par conséquent de notre **Back For Front**

Le projet de l’image Apache mjk0  est hébergé sur **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Docker**
* Composant : **im\_app\_ihml\_apa\_mjk0** **(à noter que le suffixe apa\_mjk0 est obligatoire pour la CD)**

La création du projet s’est faite en utilisant le Wizard pour Projet Image Docker (INDUS)

Le projet contient un simple fichier **DockerFile**qu’il n’est pas nécessaire de modifier.

### **Construction des images**

Une fois les deux projets images créés, il faut:

* Exécuter une génération d’assemblage

Ceci permettra de mettre à disposition les images dans **Nexus**pour la CD.

## **Front End**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=5)

A l’inverse du **Back For Front**, une seule image de type **Apache**est nécessaire pour exposer la matière statique du **Front End** des applications..

### **Image Apache**

La matière **Front End** sera exposée via une image **Apache**classique qui nécessitera néanmoins quelques paramétrages.

Le projet de l’image **Apache Front End** est hébergé sur **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Docker**
* Composant : **im\_app\_ihml\_apa\_apa4** **(à noter que le suffixe apa\_apa4 est obligatoire pour la CD)**

Pour récupérer l’**archive tgz** du **Front End** il faut déclarer la dépendance **Maven**dans le fichier **pom.xml**

pom.xml

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

**<dependencies>**  
  **<dependency>**  
   **<groupId>**fr.ca.cats.js**</groupId>**  
   **<artifactId>**ihml-front-angular**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**tgz**</type>**  
  **</dependency>**  
  **<dependency>**  
   **<groupId>**fr.ca.cats.js**</groupId>**  
   **<artifactId>**ihml-front-react**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**tgz**</type>**  
  **</dependency>**  
 **</dependencies>**

Une fois les dépendance déclarées, il faut les injecter dans l’image via le fichier **DockerFile**

DockerFile

FROM ttp10-dckreg.ca-technologies.fr/ind/im\_ind\_apa\_apa4:1.0.5  
  
USER root  
  
#Clean du dossier  
RUN rm -rf ${PROJET\_PATH}/\*  
  
#copie du fichier .htaccess inutile avec nouvelle image  
#COPY files/.htaccess ${PROJET\_PATH}/.htaccess  
  
# Copie du projet angular  
RUN mkdir /archive-projet  
RUN mkdir ${PROJET\_PATH}/ihml-demo  
COPY /dependencies/ihml-front-angular-\*.tgz /archive-projet/ihml-front-angular.tgz  
RUN tar -xzf /archive-projet/ihml-front-angular.tgz -C /archive-projet/  
RUN mv /archive-projet/package/\* ${PROJET\_PATH}/ihml-demo  
RUN rm -rf /archive-projet  
  
# Copie du projet react  
RUN mkdir /archive-projet  
RUN mkdir ${PROJET\_PATH}/ihml-demo-react  
COPY /dependencies/ihml-front-react-\*.tgz /archive-projet/ihml-front-react.tgz  
RUN tar -xzf /archive-projet/ihml-front-react.tgz -C /archive-projet/  
RUN mv /archive-projet/package/build/\* ${PROJET\_PATH}/ihml-demo-react  
RUN rm -rf /archive-projet  
  
RUN chown -R cats ${PROJET\_PATH}  
  
USER cats

On peut voir que :

* L’archive **ihml-front-angular-\*.tgz** est copié dans un répertoire temporaire **/archive-projet**
* Décompressé dans un répertoire **${PROJET\_PATH}/ihml-demo (/var/www/ihml-demo)**
  + L’application **Front End** sera ainsi exposée sur le contexte **/ihml-demo**
* On supprimer le dossier temporaire **/archive-projet**

Le process est le même pour l'archive React

Faire bien attention au nom du répertoire dans lequel sont copié et décompressé les archives.  
Ce nom de répertoire correspond au chemin racine du de votre application  
/ihml-demo = https://${host}/ihml-demo  
/ihml-demo-react = https://${host}/ihml-demo-react

Pour Spécifier les règles de routage Apache vers notre application Front End il faut

* Créer un fichier projet.conf

projet.conf

RewriteEngine On  
# **If** an existing asset or directory is requested go to it as it is  
RewriteCond %{DOCUMENT\_ROOT}%{REQUEST\_URI} -f [OR]  
RewriteCond %{DOCUMENT\_ROOT}%{REQUEST\_URI} -d  
RewriteRule ^ - [L]  
# **If** the requested resource doesn't exist, use index.html  
RewriteRule ^ /ihml-demo/index.html  
  
Header always set Content-Security-Policy "frame-ancestors https://\*.credit-agricole.fr/"  
  
# No cache **for** html  
<FilesMatch "\.(html|htm)$">  
  FileETag None  
  <IfModule mod\_headers.c>  
    Header unset ETag  
    Header set Cache-Control "max-age=0, no-cache, no-store, must-revalidate"  
    Header set Pragma "no-cache"  
    Header set Expires "Wed, 12 Jan 1980 05:00:00 GMT"  
  </IfModule>  
</FilesMatch>

* Demander via une **Demande de Support dans RTC** à injecter ce fichier dans le container (process recup file)

### **Construction des images**

Une fois le projet image créé, il faut :

* Exécuter une génération d’assemblage

Ceci permettra de mettre à disposition l’image dans **Nexus**pour la CD

## **Redis**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/WebHome?section=6)

Pour la création des images Redis, il est conseillé de suivre le guide : [GUI AC Création Stack Redis](https://ca-ts.ca-mocca.com/espaces/01160-Fabrication-SI/_layouts/15/WopiFrame.aspx?sourcedoc=/espaces/01160-Fabrication-SI/Lists/DocumentsPublics/Socle_API/StackRedis/GUI_AC_Creation_Stack_Redis.doc&action=default&DefaultItemOpen=1)

Pour notre application blanche les composants des images sont hébergés dans **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Docker**
* Composants :
  + **im\_app\_ihml\_rdi\_rdi0**
  + **im\_app\_ihml\_rdi\_rdic**
  + **im\_app\_ihml\_rwd\_rdi0**

# Déploiement continue CD

La **CD** va nous permettre de :

* Créer des **stacks Docker** avec les images de nos applications
* Déployer nos applications sur nos **Docker Engine**.

# Stacks Docker

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/D%C3%A9ploiement%20continue%20CD/WebHome?section=1)

Le mode opératoire de la **PIC**oblige à créer des **stacks Docker** pour pouvoir ensuite déployer nos applications (**images Docker**) sur les **Docker Engine**.

## **Back For Front**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/D%C3%A9ploiement%20continue%20CD/WebHome?section=2)

Le projet de la stack **Back For Front** est hébergé sur **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Image\_Docker**
* Composant : **stack\_ihml-bff-java**

Il s’agit d’un simple fichier **pom.xml** qui va déclarer

* Le nom de la stack : **IHML**

pom.xml

1

**<stack.name>**IHML**</stack.name>**

* Les dépendances vers les deux images Docker du Back For Front

pom.xml

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12

**<dependency>**  
   **<groupId>**fr.ca.cat.docker.im.app**</groupId>**  
   **<artifactId>**im\_app\_ihml\_apa\_mjk0**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**pom**</type>**  
  **</dependency>**   
  **<dependency>**  
   **<groupId>**fr.ca.cat.docker.im.app**</groupId>**  
   **<artifactId>**im\_app\_ihml\_tom\_tom8**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**pom**</type>**  
  **</dependency>**

## **Front End**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/D%C3%A9ploiement%20continue%20CD/WebHome?section=3)

Le projet de la stack **Front End** est hébergée sur **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Image\_Docker**
* Composant : **stack\_ihml-ang**

Il s’agit d’un simple fichier ***pom.xml*** qui va déclarer

* Le nom de la stack : **IHML-ANG**

pom.xml

1

**<stack.name>**IHML-ANG**</stack.name>**

* Les dépendances vers les deux images Docker du Back For Front

pom.xml

1  
2  
3  
4  
5  
6

**<dependency>**  
   **<groupId>**fr.ca.cat.docker.im.app**</groupId>**  
   **<artifactId>**im\_app\_ihml\_apa\_apa4**</artifactId>**  
   **<version>**1.1.1-SNAPSHOT**</version>**  
   **<type>**pom**</type>**  
  **</dependency>**

## **Redis**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/D%C3%A9ploiement%20continue%20CD/WebHome?section=4)

Pour la création des **stacks Redis**, il est conseillé de suivre le guide : [GUI AC Création Stack Redis](https://ca-ts.ca-mocca.com/espaces/01160-Fabrication-SI/_layouts/15/WopiFrame.aspx?sourcedoc=/espaces/01160-Fabrication-SI/Lists/DocumentsPublics/Socle_API/StackRedis/GUI_AC_Creation_Stack_Redis.doc&action=default&DefaultItemOpen=1)

Pour notre application blanche les composants des stacks sont hébergés dans **RTC**:

* Projet : **PRJF\_T200052**
* Flux : **Flux\_T200052\_Lot1\_IHML\_Image\_Docker**
* Composants :
  + **stack\_ihml-rdi0**
  + **stack\_ihml\_rwd0**

# Spécialisation via NGP

Comme nous l’avons vu lors de l’implémentation des patterns, les projets **Back For Front** et **Front End** contiennent des paramètres qui seront différents selon les environnements :

* Fichier **src/main/resources/application.properties** pour le **Back For Front**
* Fichier **src/configuration/app-config-prod.json** pour le **Front End Angular**
* Fichier **public/configuration/app-config-react.json** pour le **Front End React**

Pour pouvoir spécialiser ces fichiers selon l’environnement, il faut déclarer ces variables dans **NGP**.

Les équipes de l’industrialisation ont mis à disposition un ServiceDEVOPS.VariablesNGP pour pouvoir être autonome sur la déclaration de ces variables.

* <https://ttp10-xidv.ca-technologies.fr/ServicesApiIndus/65>

Nous ne détaillerons pas son utilisation ici mais pour exemple voici la définition des variables pour notre application.

# Back For Front

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/WebHome?section=1)

A noter :

* Le niveau logique : **NL\_TOM\_DCK**
* Le composant : **/app/im\_app\_ihml\_tom\_tom8**

L'utilisation d'un serveur Apache en frontal du Tomcat, nous oblige a a déclare un alias de serveur pouvant être accepté pour accéder à l'application.  
Ceci est fait en déclarant une variable HTTPD\_SERVERALIAS dans NGP pour notre application

A noter :

Le niveau logique : **NL\_APAMJK\_DCK**  
La composant : **/app/im\_app\_ihml\_apa\_mjk0**  
La variable : **HTTPD\_SERVERALIAS**  
permet de définir les alias acceptés par l’Apache Modjk0  
Dans notre cas \*.credit-agricole.fr car notre application est exposée sur ce domaine

# Front End

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/WebHome?section=2)

## **Variables**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/WebHome?section=3)

A noter :

* Le niveau logique : **NL\_APA\_DCK**
* Le composant : **/app/im\_app\_ihml\_apa\_apa4**

## **Fichiers**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/WebHome?section=4)

### **Configuration Front End**

Pour l’application **Front End**en plus de déclarer les variables dans **NGP**, il faut aussi déclarer le contenu des fichiers **app-config-prod.json** (**Angular**) et **app-config-react.json** (**React**) auprès des équipes de l’industrialisation.

Le processus de déclaration d'un fichier dans NGP n'est pas encore automatisé.  
Il faut

* Faire une demande de support via RTC.
* Utiliser le même fichier template que pour les variables
  + Onglet ReglesPourTypeFichier

Pour plus d'information se rapprocher des équipes d'industrialisation

Lors du lancement de l’application dans le container, les scripts d’industrialisation vont

* Récupérer le contenu du fichier
* Spécialiser le fichier selon l’environnement de déploiement (avec les variables préalablement déclarées dans **NGP**)
* Copier le fichier dans le bon répertoire.

Voici un extrait des deux variables

### **Routage Apache**

Comme le routage dans nos application **Angular**et **React**est pris en charge par ces frameworks, il est nécessaire de définir un routage au niveau **Apache**pour rediriger l'utilisateur vers les fichier **index.html** des applications.

Cela est fait via des règles de routage **Apache**à déclarer dans un fichier **projet.conf**. Le contenu du fichier est détaillé dans cette [page](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Int%C3%A9gration%20continue%20CI%20/)

Ce fichier doit ensuite être charger dans **NGP**

# Redis

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/WebHome?section=5)

Pour la spécialisation des variables Redis, il est conseillé de suivre le guide : [GUI AC Création Stack Redis](https://ca-ts.ca-mocca.com/espaces/01160-Fabrication-SI/_layouts/15/WopiFrame.aspx?sourcedoc=/espaces/01160-Fabrication-SI/Lists/DocumentsPublics/Socle_API/StackRedis/GUI_AC_Creation_Stack_Redis.doc&action=default&DefaultItemOpen=1).

Pour information voici une extraction des variables pour l’application blanche

A noter :

* **RDI0\_clusterNodes** : pour déclarer les nœuds du cluster Redis

# Installation sur Docker CE

error**Sépcialisation NGP**

[La spécialisation des variagles NGP](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Sp%C3%A9cialisation/) doit être faite au préalable

Pour déployer nos applications sur **Docker CE**, il faut :

* Déclarer les **Docker Engine** où seront installé nos stacks
* Déployer les stacks

# Déclaration Docker Engine

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Installation%20sur%20Docker%20CE/WebHome?section=1)

Lors de l’initialisation du projet, une demande a été faite par le **BO/CPT** pour la mise à disposition de **Docker Engine**.

Normalement une adresse réseau a été attribuée à chaque Docker Engine.

Dans le cas de notre application :

* Docker Engine **Back For Front** : **ELZUDEGGSCB1.yres.ytech**
* Docker Engine **Front End**: **ELZUDEGGSCB2.yres.ytech**
* Docker Engine **Redis**: **ELZUDEGGSCB4.yres.ytech**

Par l’intermédiaire de **NGP**nous allons déclarer quelle stack doit être déployée sur quel **Docker Engine**.

Toujours en utilisant le **ServiceDEVOPS.VariablesNGP** nous avons déclaré les variables suivantes

|  |  |
| --- | --- |
| **Variables** | **Détails** |
| **listeCiblesDocker1** | Liste des **Docker Engine** où déployer la stack |
| **quantiemeAPI** | Pour pouvoir déployer différentes version d’une stack sur un même **Docker Engine** |

A noter

* La valeur du niveau logique
  + **NL\_IHML#001DCK** pour le **Back For Front** (valeur de la propriété **stack.name** dans le **pom.xml** de la stack)
  + **NL\_IHML-ANG#001DCK** pour le **Front End** (valeur de la propriété **stack.name** dans le **pom.xml** de la stack)
* La valeur du composant : **DCK\_CIBLE**

## **Déploiement**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Installation%20sur%20Docker%20CE/WebHome?section=2)

Maintenant que les **Docker Engine** sont spécifiés pour chaque stack, on peut :

* Lancer la génération d’assemblage projet des deux stacks (ci-dessous un exemple pour la stack-ihml-ang)
* Lancer la génération de spécialisation et d’installation sur **ZUDB0**pour chaque stack
  + Pour cela il faut récupérer le **numéro de snapshot** qui a été créé par la génération d’assemblage précédente

Si l’installation se termine avec succès, l’application devrait être disponible.

error**Redéployer les containers**

Si l’on souhaite redéployer son application suite à des modifications, il sera nécessaire d’arrêter et de supprimer les containers déjà en cours d’exécution sur le Docker Engine ([voir cette page](http://elzudeggscan.yres.ytech:8080/xwiki/bin/view/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/G%C3%A9rer%20les%20containeurs%20sur%20Docker%20CE/))

# VIP / DNS

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/Installation%20sur%20Docker%20CE/WebHome?section=3)

En plus des **Docker Engine**, le **BO/CPT**aura demandé des **VIP**pour pouvoir accéder aux **Docker Engine**.

Il existe une interface permettant de vérifier le bon fonctionnement des **VIP**.

* <http://mup10-posu.ca-technologies.fr/index.php?module=consoles&sousModule=urbanisation>

Ci-dessous les **VIP**pour l’application exemple

# Gérer les containeurs sur Docker CE

Pour des raisons d'analyse de problème ou de redéploiement de nos applications, il est nécessaire de pouvoir interagir avec nos containeurs installés sur **Docker CE**.

# Logs / Debug

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/G%C3%A9rer%20les%20containeurs%20sur%20Docker%20CE/WebHome?section=1)

Une fois notre application déployée sur les **Docker Engine**, il peut être nécessaire de :

* Consulter les logs d’un container
* Gérer un container
  + Se connecter
  + Arrêter,
  + Supprimer
  + consulter les logs d'un container en erreur

Pour information ces opérations sont détaillés dans la documentation disponible dans les raccourcis de la Plateforme de fabrication

* Branche A ou B ⇒ SI Client ⇒ Aide  la Dockerisation Doc

## **Consultation des logs**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/G%C3%A9rer%20les%20containeurs%20sur%20Docker%20CE/WebHome?section=2)

Il peut être nécessaire de pouvoir consulter les logs de nos containers pour des raisons de debug ou d’analyse de problème.

En utilisant un **client SSH** (**MRemoteNG**par exemple), il est possible de se connecter au **Docker Engine** et de consulter les logs de ses containers.

Pour cela il faut :

* Lancer **MRemoteNG**et se connecter avec l’adresse de votre **Docker Engine** et votre compte **ETAXXXX**
* Se connecter en **root**
* Les logs des containers du **Docker Engine** se trouvent dans le répertoire :
  + **/var/log/DCK0**

Ici on peut voir deux répertoire pointant sur les logs des deux containers de l’image **Back For Front**

* **cn\_zudb0\_ihml-1\_apa\_mjk0\_p0001: Apache ModJk**
* **cn\_zudb0\_ihml-1\_tom\_tom8\_p0001: Tomcat 8**

Ci-dessous le contenu du répertoire **cn\_zudb0\_ihml-1\_tom\_tom8\_p0001**

On a ainsi accès au fichier **catalina.log** du **Tomcat 8**.

On peut aussi noter l’existence du répertoire indus. Il s’agit du répertoire contenant les logs des scripts d’indus lancés au démarrage du container (spécialisation de la matière)

## **Gestion des containers**

[Modifier](http://elzudeggscan.yres.ytech:8080/xwiki/bin/edit/Front/IHM%20Light/Guide%20pas%20%C3%A0%20pas%20utilisateur%20fili%C3%A8re%20IHM%20Light/Int%C3%A9gration%20et%20d%C3%A9ploiement%20continue%20%28%20CI%20%26%20CD%29/G%C3%A9rer%20les%20containeurs%20sur%20Docker%20CE/WebHome?section=3)

En plus de vouloir consulter les logs on peut souhaiter

* Se connecter directement au container
* Arrêter le container
* Supprimer le container
* Consulter les logs d'un container en erreur

### **Connexion au container**

Pour se connecter au container il faut :

* Récupérer l’id du container en utilisant la commande pour lister les containers du **Docker Engine**

docker ps –a

* Exécuter la commande pour lancer un bash dans le container

docker **exec** –it ${container id} bash

Ici nous sommes connectés au container **Tomcat 8** et nous avons visualisé le contenu du répertoire **conf**de **Tomcat**.

* Pour se déconnecter, exécuter la commande

**exit**

### **Arrêter le container**

Pour arrêter le container il faut :

* Récupérer l’id du container avec la commande

docker ps –a

* Exécuter la commande

docker stop ${container id}

### **Supprimer le container**

Pour supprimer le container il faut :

* Récupérer l’id du container avec la commande

docker ps –a

* Exécuter la commande

docker rm ${container id}

### **Consulter les logs d'un container en erreur**

Pour des raisons diverses, il se peut qu'un container soit en erreur. Par conséquent il n'est pas possible de s'y connecter. Cependant, on peut toujours consulter les logs émise par le container.

* Récupérer l’id du container avec la commande

docker ps –a

* Exécuter la commande

docker logs ${container id}