Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Table des matières

[Introduction 1](#_Toc167455784)

[Chapitre 1 : Keycloak 2](#_Toc167455785)

[1. Introduction 2](#_Toc167455786)

[2. Installation et mise en place de Keycloak 3](#_Toc167455787)

[Docker compose 5](#_Toc167455788)

[Postgres 5](#_Toc167455789)

[Keycloak 6](#_Toc167455790)

[Dockerfile 7](#_Toc167455791)

[Création de l’image personnalisée 8](#_Toc167455792)

[3. Configuration de Keycloak 10](#_Toc167455793)

[Realm 10](#_Toc167455794)

[User federation 15](#_Toc167455795)

[Clients 22](#_Toc167455796)

[OAuth/OIDC 23](#_Toc167455797)

[General settings 23](#_Toc167455798)

[Capability config 24](#_Toc167455799)

[Login settings 26](#_Toc167455800)

[SAML 27](#_Toc167455801)

[Modifier l’image – Thèmes 28](#_Toc167455802)

# Introduction

Ce document est destiné au service informatique de HELMo.

Celui-ci va aborder différents sujets que j’ai dû étudier dans le cadre de mon TFE.

Ce document n’est pas exhaustif, je ne traite pas les sujets que je ne maitrise pas un minimum et dont je n’ai pas pu travailler durant mon stage.

Ce document est libre et des parties de celui-ci peuvent être utilisée dans d’autres documents au besoin.

Cependant le document comporte des informations confidentielles qui ne doivent pas être partagées avec quiconque ne faisant pas partie de l’équipe informatique.  
Il est donc demandé de ne pas partager ce document hors de l’équipe informatique.  
Une version avec les données anonymisées sera effectuée dans le cadre de mon mémoire.

CETTE DOCUMENTATION A ÉTÉ ADAPTEE POUR NE PLUS ETRE CONFIDENTIELLE DANS LE BUT DETRE FOURNIE AVEC LE MÉMOIRE DE SASHA BROUHON.

# Chapitre 1 : Keycloak

## 1. Introduction

Keycloak est le nouvel outil SSO de HELMo. Il a été choisi par le jury de l’équipe informatique à la suite de l’analyse en détails de plusieurs solutions SSO.  
Keycloak permet pour le moment d’effectuer l’authentification des utilisateurs des applications d’HELMo et ce via l’Active Directory.  
Dans ce chapitre je vais traiter le sujet de la mise en place de Keycloak, de la création et modification d’une image de Keycloak, de la synchronisation d’un Active Directory avec Keycloak, de la création d’un client Oauth2/OIDC avec une application communiquant avec ces protocoles, d’un client SAML avec une application communiquant avec ce protocole, de la gestion des realms et des paramètres de realms.

## 2. Installation et mise en place de Keycloak

Pour l’installation nous avons décidé d’uniquement travailler avec docker. C’est-à-dire que nous utiliserons la dernière version de Keycloak comme image et créerons des containers à partir de celle-ci. Cependant nous souhaitons créer nos propres thèmes et pour cela nous devons créer notre propre image de Keycloak.

Pour l’installation de Keycloak j’ai utilisé Visual Studio Code avec Docker Desktop.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Figure 1-Visual Studio Code et Docker Deskstop

Pour commencer il est nécessaire de créer deux premiers fichiers :

* Docker-compose.yml
* .env



Ces deux fichiers vont permettre de créer un premier container avec une image de Keycloak non personnalisée.

Une image contenant texte, capture d’écran

Description générée automatiquement

Figure 2-docker-compose.yml

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 3-.env

Dans le fichier docker compose ici présent l’on utilise comme image « helmokeycloak », cette image correspond à l’image personnalisée de keycloak que j’ai créée. Si l’on souhaite utiliser une image non personnalisée l’on peut remplacer l’image par « quay.io/keycloak/keycloak:latest » où latest signifie que chaque fois que l’on va initialiser le container avec le fichier docker compose l’image utilisée sera la dernière release de Keycloak. C’est préférable d’éviter cette pratique lorsque l’on souhaite utiliser en production un outil car si la version vient à changer et que dans la nouvelle version certains paramètres ne sont plus configurés de la même façon pour une raison x ou y cela pourrait causer des erreurs non désirées.

Il est donc préférable d’utiliser une formulation comme celle-ci : « quay.io/keycloak/keycloak:24.0» . Au moment où je rédige ceci la version 24 est la dernière release majeure de Keycloak et donc celle à utiliser de préférence.

Pour la commande entrypoint, l’on peut voir que j’ai ajouté des paramètres.  
Il est préférable de ne pas mettre de paramètre si possible si l’on travaille avec des variables ENV et donc voici la version préférable :

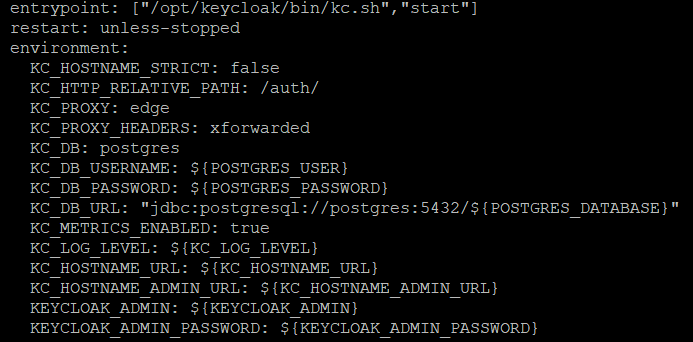


Figure 4-modification préférable du fichier docker compose

Pour trouver les nomenclatures des variables ENV elles sont disponibles sur la doc de Keycloak. Une simple recherche sur un navigateur de recherche du paramètre souhaité permet de trouver la variable ENV correspondante.

Exemple pour hostname-strict : <https://www.keycloak.org/server/hostname>

Exemple pour « http-relative-path », « proxy » et « proxy-headers » : <https://www.keycloak.org/server/reverseproxy>

Pour parler du contenu du fichier docker compose, dans la première partie nous définissions l’utilisation de postgres comme DB de Keycloak.

### Docker compose

#### Postgres

* Image : L’on définit l’image de postgres souhaitée définie env tant que variable dont la valeur sera définie dans le fichier .env (visualisation du fichier plus loin).
* Restart : Le paramètre restart défini sur « unless-stopped » permet de conserver le statut UP du container tant que l’on ne le stop pas explicitement.
* Healthcheck : Le healthcheck quant à lui correspond à un ou des tests effectués dans le but de vérifier certaines conditions avant de passer à la suite. Le but de cette opération est de stopper le lancement si un problème est survenu.
* Environment : C’est ici que sont définies les variables d’environnement liées à postgres. Si l’on souhaite définir les valeurs dans le fichier docker compose la syntaxe est la suivante : « VARIABLE: valeur »il est important qu’entre le « : » et la valeur il y ait un espace. Ou alors l’on peut définir les valeurs dans un fichier « .env » et si l’on souhaite définir une variable de cette façon la syntaxe est la suivante : « VARIABLE : ${nom de la variable correspondante dans le fichier .env} » . Il est préférable lorsque les valeurs à définir sont confidentielles ou importantes de les définir avec le fichier .env que l’on ajoutera alors dans .gitignore *(permet de ne pas push le fichier .env qui contient des informations à ne pas divulguer)* dans le cas d’un déploiement avec git.
* Volumes : Ici l’on crée notre ou nos volumes. Puisqu’une image ne peut être modifiée, il est nécessaire de créer des volumes pour conserver des données. Et donc pour postgres puisque l’on souhaite que la DB conserve des données l’on doit lui définir un volume, le chemin du volume à créer est disponible sur la doc officielle.

#### Keycloak

* Image : Pour l’image il y a deux possibilités. Soit utiliser une version de Keycloak que l’on souhaite (ou la latest) ou alors utiliser une image personnalisée.

Au début nous avons utilisé une image officielle, mais pour des besoins de modifications de l’image nous avons ensuite travaillé avec une image personnalisée (utilisation d’un thème personnalisé dans notre image).

Et donc si l’on souhaite utiliser une image personnalisée il suffit de mettre le chemin vers celle-ci. Actuellement nous utilisons « git-registry.helmo.be/si/infrastructure/helmo-keycloak:latest », j’y reviendrai plus tard car cette image fait partie du déploiement continu qui est en train d’être mis en place.

* Entrypoint : Ce paramètre définit la commande à exécuter lors du démarrage du service. Ici l’on définit « start » en paramètre de « kc.sh » . Start comme paramètre signifie que l’on souhaite lancer le service en « production mode » et si l’on souhaite le lancer en « development mode » il suffit de mettre « start-dev ».
* Restart : Même chose que pour postgres.
* Environment : Même chose que pour postgres.
* Ports : Paramètre définissant le port source et destination du service.

### Dockerfile

Donc maintenant que l’on a créé le docker compose nous devons maintenant créer l’image de keycloak. Pour commencer je vais présenter la création de l’image manuelle dans Visual Studio Code et dans la suite je présenterai comment implémenter cela avec l’intégration continue.  
Tout d’abord il faut créer un fichier Dockerfile.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

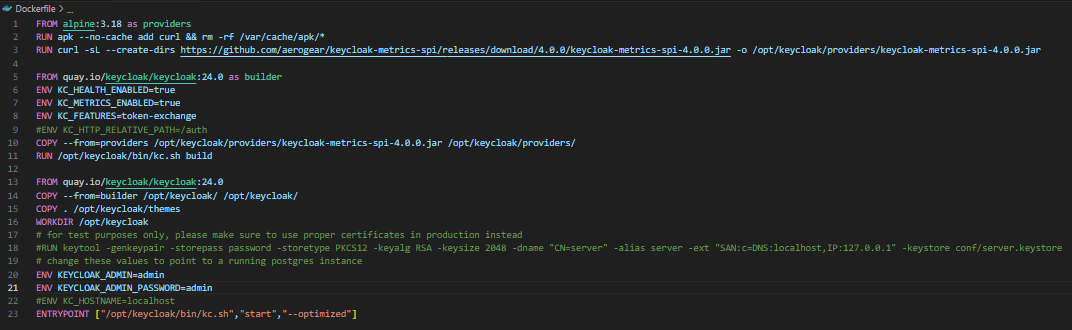


Figure 5-Dockerfile

Ce Dockerfile se base sur un Dockerfile qui a été fourni.  
La première partie de celui-ci implémente les keycloak-metrics.  
Ensuite l’on défini notre image keycloak et on la crée.  
CE qui est important de remarquer est la ligne « COPY ./opt/keycloak/themes »,  
cette ligne copie en fait dans le répertoire courant les thèmes personnalisés dans la nouvelle image de keycloak personnalisée. Et donc c’est de cette façon que l’on peut créer notre propre image (j’y reviendrai juste après).  
Il faut aussi savoir que ce qui est défini dans l’image est overwritten par ce qui est écrit dans fichier docker compose, comme l’entrypoint.

Une fois que l’on a ces fichiers de fait pour le moment sans thème personnalisé donc la ligne dans le fichier Dockerfile devrait être supprimée, l’on peut maintenant créer notre image et l’utiliser. Pour ce faire en ligne de commande il faut tout d’abord effectuer « cd /chemin/vers/le/fichier/dockerfile » ensuite « docker build -t monimage » (-t/--tag permet de nommer son image utile pour ensuite la spécifier dans le fichier docker compose et la retrouver facilement). Maintenant que l’image est créée il faut dans le fichier docker compose spécifier le chemin vers celle-ci dans le paramètre image.

Maintenant que nous avons notre image pour créer des containers avec celle-ci et notre DB postgres, il faut se rendre dans le répertoire du fichier .yml et faire docker compose -d (-d/--detach permet de lancer en daemon les services) et si l’on souhaite voir en direct les logs ne pas spécifier le -d, ce qui peut être intéressant pour un premier lancement.

Et voilà maintenant Keycloak est accessible via l’url spécifiée dans le fichier de configuration. Dans notre cas sso2.helmo.be, mais pour cela il a fallu auparavant créer cette url dans la config réseau d’HELMo. Mais il est tout à fait possible de spécifier une url localhost pour des tests.

### Création de l’image personnalisée

Dans cette partie est décrit les étapes pour créer sa propre image de keycloak.  
Puisque dans mon cas j’ai du le faire pour ajouter de nouveaux thèmes je vais être explicite sur ce cas, mais la méthode est la même pour tout autre modification.

Tout d’abord il est nécessaire de posséder le code source de keycloak pour connaitre les chemins à définir concernant les modifications voulues. Par exemple pour mon cas j’avais besoin de trouver où sont posés les thèmes et en cherchant voici le chemin : « /themes/src/main/resources/theme » <https://github.com/keycloak/keycloak/tree/main/themes/src/main/resources/theme>  
Dans ce repository « /theme », il y a « base », « keycloak.v2/login » et « keycloak ». Ces repositories sont les 3 thèmes par défauts de keycloak. Et ce qu’il faut comprendre c’est que le thème base est le thème source. C’est-à-dire que les autres thèmes découlent des propriétés de celui-ci. Ce qui signifie que si dans le thème « base » l’on dit que les pages web sont toutes en background bleu, si l’on choisit le thème « base » nos pages web auront toutes un background bleu. Mais si dans le thème « keycloak » qui a comme parent « base », l’on ne spécifie rien concernant le background, alors le back ground sera aussi bleu. En fait un enfant hérite de toutes les propriétés de son parent mais peut écrire par-dessus toutes celles dont il le souhaite.

Et c’est exactement ce que l’on souhaite faire. Car l’on ne veut pas recréer de 0 toutes les pages web de Keycloak mais uniquement modifier les existantes pour donner un style HELMo (surtout sur la page de login). Donc l’on va créer notre thème dans le repository des thèmes définir dans celui-ci que le parent est le thème « keycloak ou keycloak.v2 » et ensuite modifier ce que l’on souhaite modifier.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Pour commencer il faut donc créer dans notre repository notre repository de thème, ici nous l’avons nommé helmo. Et maintenant il faut savoir de quel thème existant l’on veut se baser. Pour ma part j’ai choisi le thème « keycloak » car il correspondait à ce que je cherchais. Une fois choisi, il faut définir le parent dans le fichier « theme.properties » mais aussi « import » qui donne le chemin depuis lequel l’on va importer les ressources. :



Maintenant que l’on a définit notre parent, l’on doit modifier ce que l’on souhaite.  
Personnellement je veux modifier la page de login, je veux changer le background, les textes et aussi le logo. Pour cela il faut recréer le chemin vers le fichier login.css qui est le fichier css de la page de login.

Donc je souhaite commencer par modifier le logo de keyloak par le logo de HELMo. Pour cela je trouve dans le fichier css l’endroit de définition du logo et je le modifie avec mes besoins. J’ai alors le choix entre modifier le chemin source du logo ou garder le même mais dans le repository source déposer mon logo HELMo avec le nom défini dans le fichier css (ce que j’ai fait dans l’exemple).

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

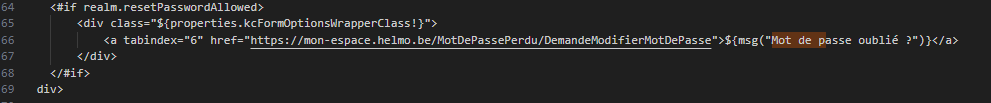
Figure 6-définition du logo HELMo

Maintenant que j’ai modifié la partie du code que je souhaitais modifier, il me faut ajouter mon image dans le dossier source que voici :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Pour le background j’ai juste modifier l’image par mon background que j’ai personnalisé au préalable. Mais maintenant je veux aussi modifier le texte qui est en anglais par du français. Pour cela il faut créer le fichier login.ftl et modifier toutes les occurrences de « msg » par du français.



Comme l’on peut le voir ci-dessus j’ai remplacé le message anglais par du français avec « Mot de passe oublié ? » . Mais ce n’est pas la seule modification faite ici, car je voulais aussi que le lien ne renvoit plus vers la page de changement de mot de passe keycloak mais celle de HELMo. Pour cela j’ai créé le bouton que l’on voit sur la capture qui redirige vers lien demandé.

Et voilà après avoir modifié l’image, l’avoir rebuild et lancé les containers vous devrieze arriver sur cette page. Si c’est le cas c’est que tout a fonctionné.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

## 3. Configuration de Keycloak

### Realm

Un realm qu’est-ce que c’est ? Un realm c’est instance de keycloak séparée des autres realms. C’est-à-dire que si j’ai un realm A et un realm B aucune configuration faite dans le A n’a d’impact sur le B. Par exemple si je lie un AD au realm A et donc que dans users je peux consulter les users de l’AD, si je me connecte au realm B je n’aurai pas accès aux users de l’AD. Ce que ça permet c’est que l’on puisse faire de la séparation sans devoir créer plusieurs keycloak.

Dans keycloak par défaut il y a un realm de créé c’est le realm keycloak aka le realm master. Ce realm c’est le realm admin, il ne doit donc pas être utilisé pour des fins autre que l’administration.  
Donc première chose à faire lorsque l’on peut utiliser keycloak c’est de créer son realm. Pour notre cas j’ai créé le realm helmo  
Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Figure 7-Créer un nouveau realm étape 1

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Figure 8-Créer un nouveau realm étape 2

C’est tout pour la partie création du realm.  
Maintenant pour les « settings » du realm il existe des paramètres assez intéressant pour nous.   
 Donc rendez-vous dans l’onglet **Configure>Realm setting>login** , dans celui-ci il est disponible plusieurs paramètres utiles voici la liste (en rouge ceux que je n’ai pas activés et en vert ceux que j’ai activés ) :

* Login screen customization
  + User regisration : Enable/disable the registration page. A link for registration will show on login page too.
  + Forgot password : Show a link on login page for user to click when they have forgotten their credentials.
  + Remember me : Show checkbox on login page to allow user to remain logged in between browser restarts until session expires.
* Email settings
  + Email as username : Allow users to set email as username.
  + Login with email : Allow users to log in with their email address.
  + Duplicate emails : Allow multiple users to have the same email address. Changing this setting will also clear the user's cache. It is recommended to manually update email constraints of existing users in the database after switching off support for duplicate email addresses.
  + Verifiy email : Require user to verify their email address after initial login or after address changes are submitted.
* User info setting
  + Edit username : If enabled, the username field is editable, readonly otherwise.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Pour le bouton forgot password, je l’ai modifié comme expliqué précédemment avec mon propre bouton. Mais en activant l’option je valide la condition « if » de ce paramètre pour afficher mon bouton.  
Login with email est utile pour helmo du fait que pour un étudiant il est plus simple de retenir son email que son matricule et que de toute façon en BD l’email HELMo est associé au matricule donc ça ne change rien pour nous qu’ils utilisent l’un ou l’autre.

Ensuite dans **Configure>Realm settings>Themes**, l’on peut ici mettre notre thème que l’on a créé pour la page de login. Puisque je l’avais nommé helmo il est ici sélectionné sous ce nom.  
Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 9-Choix des thèmes

Maintenant dans **Configure>Realm settings>Security defenses**, il y a possibilité de mettre des protections sur les Headers et de contre les attaques par Brute force. Pour les Headers cela se situe dans **Configure>Realm settings>Security >defenses>Headers**. Les paramètres par défaut sont déjà bien configurés mais il est possible si souhaiter de modifier ceux-ci.  
Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure 10-Protection des Headers

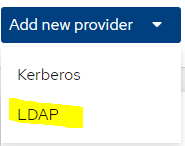
Et dans **Configure>Realm settings>Security >defenses>Brute force detection**, l’on peut activer le mode de protection qui par défaut est désactivé et personnellement je conseille de l’activer.  
Il existe 3 modes de protection (le 4 -ème étant désactivé) le premier **Lockout permanently** je ne le conseille par car ce mode bloque de façon permanente un compte après un nombre de « login failure » défini. Il est ensuite possible de débloquer le compte mais c’est un admin qui devra le faire manuellement, à voir le taux de faux positif avec cette alternative.  
L’option **Lockout temporarily**, quant à elle peut être plus intéressante du fait qu’elle ne demandera pas d’intervention manuelle ou du moins beaucoup moins souvent.  
Ici le temps entre chaque blocage dû à un nombre de « login failure » dans un temps déterminé augmente. L’on peut bloquer en secondes,minutes,heures ou même jours.  
L’option **Lockout permanently after temporary lockout** quant à elle peut être plus intéressante que les deux autres car elle les combine. Ce qui fait que l’on a la protection du « **lockout temporarily** » qui permet d’éviter de bloquer définitivement trop rapidement mais en même temps si un utilisateur est trop bloqué temporairement à partir d’un moment il est bloqué définitivement sauf si action manuelle de déblocage.

Il existe beaucoup d’autres options disponibles pour le realm mais j’ai cité ici les principales qui nous intéressent. Toutes ces options sont décrites soit dans l’outil même à l’aide de ce symbole  qui décrit l’option. Ou sinon sur la doc officielle tout est décrit.

### User federation

Dans cette partie je vais décrire comment créer sa « user federation ». C’est via cette federation que l’on va pouvoir lier notre Active Directory où sont enregistrés les utilisateurs HELMo avec Keycloak. Pour commencer il faut se rendre dans **Configure>User federation**, une fois dedans l’on peut y retrouver le bouton **Add new provider**.  
Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

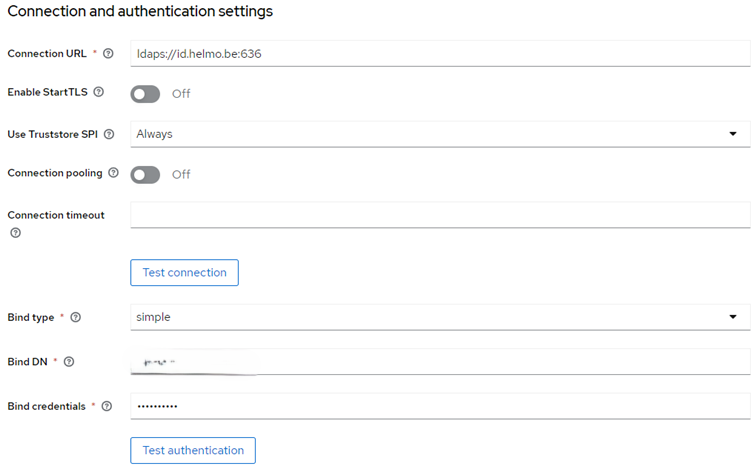
Après avoir cliqué dessus l’on peut voir que deux options s’offrent à nous **Kerberos** et **LDAP**. Pour notre part notre Active Directory utilise **LDAP** donc l’on sélectionne **LDAP**.  
  
Nous arrivons alors sur la page **Add LDAP provider**, tout d’abord je vais décrire chaque option puis expliciter comment je les ai complétées.

* **General options**

Une image contenant texte, capture d’écran, ligne

Description générée automatiquement

* + UI display name
    - Display name of provider when linked in the Admin UI
    - idp.helmo.be (ou tout autre nom peu importe)
  + Vendor : LDAP vendor (provider)
    - Active Directory
* **Connection and authentication settings**



* + Connection URL
    - Connection URL to your LDAP server
    - ldaps://id.helmo.be:636 (l’URL de notre AD avec le port secure de ldaps, possible d’utiliser aussi ldap pour le mode non sécurisé)
  + Enable StartTLS
    - Encrypts the connection to LDAP using STARTTLS, which will disable connection pooling
    - Off
  + Use Trustore SPI
    - Specifies whether LDAP connection will use the Truststore SPI with the truststore configured in command-line options. 'Always' means that it will always use it. 'Never' means that it will not use it. Note that even if Keycloak truststore is not configured, the default java cacerts or certificate specified by 'javax.net.ssl.trustStore' property will be used.
    - Always
  + Connection pooling
    - Determines if Keycloak should use connection pooling for accessing LDAP server.
    - Off
  + Connection timeout
    - LDAP connection timeout in milliseconds
    - NULL (vide)
  + Test connection
    - Bouton permettant de tester la connexion avec l’URL fournie si réussie le message suivant apparaitra :  
      
  + Bind type
    - Type of the authentication method used during LDAP bind operation. It is used in most of the requests sent to the LDAP server. Currently only 'none' (anonymous LDAP authentication) or 'simple' (bind credential + bind password authentication) mechanisms are available.
    - simple
  + Bind DN
    - DN of the LDAP admin, which will be used by Keycloak to access LDAP server
    - XXX (l’utilisateur dans l’Active Directory qui est utilisé pour récupérer les utilisateurs et les groupes dans l’Active Directory, celui-ci doit avoir été créé et paramétré au préalable dans l’Active Directory)
  + Bind credentials
    - Password of LDAP admin. This field is able to obtain its value from vault, use ${vault.ID} format.
    - HIDDEN (mot de passe lié au compte guestrc@id.helmo.be)
  + Test authentication
    - Bouton permettant de tester l’authentification du compte fourni dans Bind DN, si réussite le message suivant apparait : 
* **LDAP searching and updating**

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

* + Edit mode
    - READ\_ONLY is a read-only LDAP store. WRITABLE means data will be synced back to LDAP on demand. UNSYNCED means user data will be imported, but not synced back to LDAP.
    - READ\_ONLY
  + Users DN
    - Full DN of LDAP tree where your users are. This DN is the parent of LDAP users. It could be for example 'ou=users,dc=example,dc=com' assuming that your typical user will have DN like 'uid='john',ou=users,dc=example,dc=com'.
    - OU=People,DC=id,DC=helmo,DC=be  
      C’est le chemin DN depuis lequel l’on souhaite que keycloak récupère les utilisateurs. Ce chemin est hiérarchique. C’est-à-dire que si l’on spécifie le chemin jusque People tous les utilisateurs qui sont dans des OU ou des OU dans des OU dans People seront récupérés.
  + Username LDAP attribute
    - Name of the LDAP attribute, which is mapped as Keycloak username. For many LDAP server vendors it can be 'uid'. For Active directory it can be 'sAMAccountName' or 'cn'. The attribute should be filled for all LDAP user records you want to import from LDAP to Keycloak.
    - sAMAccountName
  + RDN LDAP attribute
    - Name of the LDAP attribute, which is used as RDN (top attribute) of typical user DN. Usually it's the same as the Username LDAP attribute, however it is not required. For example for Active directory, it is common to use 'cn' as RDN attribute when username attribute might be 'sAMAccountName'.
    - cn
  + UUID LDAP attribute
    - Name of the LDAP attribute, which is used as a unique object identifier (UUID) for objects in LDAP. For many LDAP server vendors, it is 'entryUUID'; however some are different. For example, for Active directory it should be 'objectGUID'. If your LDAP server does not support the notion of UUID, you can use any other attribute that is supposed to be unique among LDAP users in tree. For example 'uid' or 'entryDN'.
    - objectGUID
  + User object classes
    - All values of LDAP objectClass attribute for users in LDAP, divided by commas. For example: 'inetOrgPerson, organizationalPerson'. Newly created Keycloak users will be written to LDAP with all those object classes and existing LDAP user records are found just if they contain all those object classes.
    - person, organizationalPerson, user
  + User LDAP filter
    - Additional LDAP filter for filtering searched users. Leave this empty if you don't need an additional filter. Make sure that it starts with '(' and ends with ')'.
    - NULL (vide)
  + Search scope
    - For one level, the search applies only for users in the DNs specified by User DNs. For subtree, the search applies to the whole subtree. See LDAP documentation for more details.
    - Subtree  
      Permet comme expliqué précédemment de prendre tous les users sous People.
  + Read timeout
    - LDAP read timeout in milliseconds. This timeout applies for LDAP read operations.
    - NULL (vide)
  + Pagination
    - Whether the LDAP server supports pagination
    - On  
      Permet d’afficher sous forme de pages les utilisateurs.
  + Referral
    - Specifies if LDAP referrals should be followed or ignored. Please note that enabling referrals can slow down authentication as it allows the LDAP server to decide which other LDAP servers to use. This could potentially include untrusted servers.
    - ignore
* Synchronization settings

Une image contenant texte, capture d’écran, ligne, Police

Description générée automatiquement

* + Import users
    - If true, LDAP users will be imported into the Keycloak DB and synced by the configured sync policies.
    - Import users
  + Sync Registrations
    - Sync Registrations
    - Off
  + Batch size
    - Count of LDAP users to be imported from LDAP to Keycloak within a single transaction
    - NULL
  + Periodic full sync
    - Whether periodic full synchronization of LDAP users to Keycloak should be enabled or not
    - Off
  + Periodic changed users sync
    - Whether periodic synchronization of changed or newly created LDAP users to Keycloak should be enabled or not
    - Off
* Kerberos integration

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

* + Allow Kerberos authentication
    - Enable/disable HTTP authentication of users with SPNEGO/Kerberos tokens. The data about authenticated users will be provisioned from this LDAP server.
    - Off
  + Use Kerberos for password authentication
    - User Kerberos login module for authenticating username/password against Kerberos server instead of authenticating against LDAP server with Directory Service API
    - Off
* Cache settings
  + Cache policy
    - Cache Policy for this storage provider. 'DEFAULT' is whatever the default settings are for the global cache. 'EVICT\_DAILY' is a time of day every day that the cache will be invalidated. 'EVICT\_WEEKLY' is a day of the week and time the cache will be invalidated. 'MAX\_LIFESPAN' is the time in milliseconds that will be the lifespan of a cache entry.
    - DEFAULT
* Advanced settings
  + Enable the LDAPv3 password modify extended operation
    - Use the LDAPv3 Password Modify Extended Operation (RFC-3062). The password modify extended operation usually requires that LDAP user already has password in the LDAP server. So when this is used with 'Sync Registrations', it can be good to add also 'Hardcoded LDAP attribute mapper' with randomly generated initial password.
    - Off
  + Validate password policy
    - Determines if Keycloak should validate the password with the realm password policy before updating it. For the case when user's password is saved in LDAP, some Keycloak password policies will not work (Not Recently Used, Expire Password, Hashing Iterations, Hashing Algorithm) due the fact that Keycloak does not have direct control over the password storage. It is needed to enable password policies at the LDAP server layer if you want to leverage those password policies.
    - Off
  + Trust Email
    - If enabled, email provided by this provider is not verified even if verification is enabled for the realm.
    - On

### Clients

Dans cette partie sera expliqué comment créer un nouveau client et le configurer.  
Cette partie est séparées en deux, d’une part la partie SAML et d’autre part la partie OAuth/OIDC.

Pour commencer il faut se rendre dans Manage>Clients.  
Ensuite nous allons soit créer un client soit en importer un.  
Importer un client est très utile pour SAML mais est utile de façon plus générale par exemple pour le déploiement continu. Car l’on peut alors exporter automatiquement nos clients et les importer automatiquement lors du déploiement.  
Il est à noter qu’il est possible d’importer deux formats de fichiers : JSON et XML.

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

#### OAuth/OIDC

Après avoir cliqué sur « Create client », l’on se retrouve sur le premier des 3 étapes de configurations, « **General settings** ».

##### General settings

* Client type
  + 'OpenID Connect' allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server.'SAML' enables web-based authentication and authorization scenarios including cross-domain single sign-on (SSO) and uses security tokens containing assertions to pass information.
  + OpenID Connect  
    Car nous sommes dans une config OIDC.
* Client ID
  + The client identifier registered with the identity provider.
  + Monclientid  
    Le Client ID qui doit correspondre à celui présent dans le client.
* Name
  + Specifies display name of the client. For example 'My Client'. Supports keys for localized values as well. For example: ${my\_client}
  + Lenomdemonclient  
    Un nom défini dans keycloak, il n’est que visuel mais il est recommandé de le rendre pertinent.
* Description
  + Specifies description of the client. For example 'My Client for TimeSheets'. Supports keys for localized values as well. For example: ${my\_client\_description}
  + Une description de mon client  
    Il est recommandé de rendre cette description pertinente car elle sert de documentation sur le client.
* Always display in UI
  + Always list this client in the Account UI, even if the user does not have an active session.
  + On  
    Intéressant de l’activer pour les clients dont on se sert le plus.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Maintenant que l’on a fini de configurer les « General settings », nous allons passer à la configuration de « **Capability config** ». Cette partie de la configuration concerne la partie AA (authentication, authorization)

##### Capability config

* Client authentication
  + This defines the type of the OIDC client. When it's ON, the OIDC type is set to confidential access type. When it's OFF, it is set to public access type
  + Off  
    Non exhaustif.
* Authorization
  + This defines the type of the OIDC client. When it's ON, the OIDC type is set to confidential access type. When it's OFF, it is set to public access type
  + Off  
    Non exhaustif.
* Authentication flow
  + Standard flow
    - This enables standard OpenID Connect redirect based authentication with authorization code. In terms of OpenID Connect or OAuth2 specifications, this enables support of 'Authorization Code Flow' for this client.
    - On  
      Pour une configuration basique fonctionnelle avec une authentification login, password, activer uniquement « **standard flow** » est suffisant.
  + Direct access grants
    - This enables support for Direct Access Grants, which means that client has access to username/password of user and exchange it directly with Keycloak server for access token. In terms of OAuth2 specification, this enables support of 'Resource Owner Password Credentials Grant' for this client.
    - Off  
      Non exhaustif
  + Implicit flow
    - This enables support for OpenID Connect redirect based authentication without authorization code. In terms of OpenID Connect or OAuth2 specifications, this enables support of 'Implicit Flow' for this client.
    - Off  
      Non exhaustif
  + Service accounts roles
    - Allows you to authenticate this client to Keycloak and retrieve access token dedicated to this client. In terms of OAuth2 specification, this enables support of 'Client Credentials Grant' for this client.
    - Off  
      Non exhaustif
  + OAuth 2.0 Device Authorization Grant
    - This enables support for OAuth 2.0 Device Authorization Grant, which means that client is an application on device that has limited input capabilities or lack a suitable browser.
    - Off  
      Non exhaustif
  + OIDC CIBA Grant
    - This enables support for OIDC CIBA Grant, which means that the user is authenticated via some external authentication device instead of the user's browser.
    - Off  
      Non exhaustif

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

##### Login settings

* Root URL
  + Root URL appended to relative URLs
  + <http://localhost:5000>  
    A adapter à votre configuration
* Home URL
  + Default URL to use when the auth server needs to redirect or link back to the client.
  + http://localhost:5000/  
    A adapter à votre configuration
* Valid redirect URIs
  + Valid URI pattern a browser can redirect to after a successful login. Simple wildcards are allowed such as 'http://example.com/\*'. Relative path can be specified too such as /my/relative/path/\*. Relative paths are relative to the client root URL, or if none is specified the auth server root URL is used. For SAML, you must set valid URI patterns if you are relying on the consumer service URL embedded with the login request.
  + \*  
    Mettre étoile permet de définir que tout est autorisé.  
    A adapter à votre configuration.
* Valide post logout redirect URIs
  + Valid URI pattern a browser can redirect to after a successful logout. A value of '+' or an empty field will use the list of valid redirect uris. A value of '-' will not allow any post logout redirect uris. Simple wildcards are allowed such as 'http://example.com/\*'. Relative path can be specified too such as /my/relative/path/\*. Relative paths are relative to the client root URL, or if none is specified the auth server root URL is used.
  + \*  
    Mettre étoile permet de définir que tout est autorisé.  
    A adapter à votre configuration.
* Web origins
  + Allowed CORS origins. To permit all origins of Valid Redirect URIs, add '+'. This does not include the '\*' wildcard though. To permit all origins, explicitly add '\*'.
  + \*  
    Mettre étoile permet de définir que tout est autorisé.  
    A adapter à votre configuration.

Une image contenant texte, ligne, nombre, Police

Description générée automatiquement

Une fois votre client, vous aurez normalement accès à ces onglets.  
Celui qui nous intéresse maintenant est l’onglet « credentials ».  
C’est dans celui-ci que l’on va pouvoir générer et récupérer notre « Client Secret » nécessaire à notre configuration du client.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

#### SAML

Pour SAML la méthode de création est un différente. Si vous lisez-ceci je me base sur le fait que vous connaissez SAML et donc que vous savez que dans SAML il y la connaissance du fichier metadata de chacun des communicants.  
Et donc je vais prendre comme exemple un service provider Keycloak et un Identity Provider Keycloak. Alors une fois que l’on a récupéré notre fichier metadata du client, l’on peut alors l’importer et alors Keycloak va autocompléter les champs définis dans le fichier .xml importé.  
Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Je ne vais pas détailler l’énorme quantité de paramètres configurables car ce n’est plus du ressort de Keycloak mais plus de la configuration du client impactée dans Keycloak ce qui est donc à vous de voir en fonction de votre client.

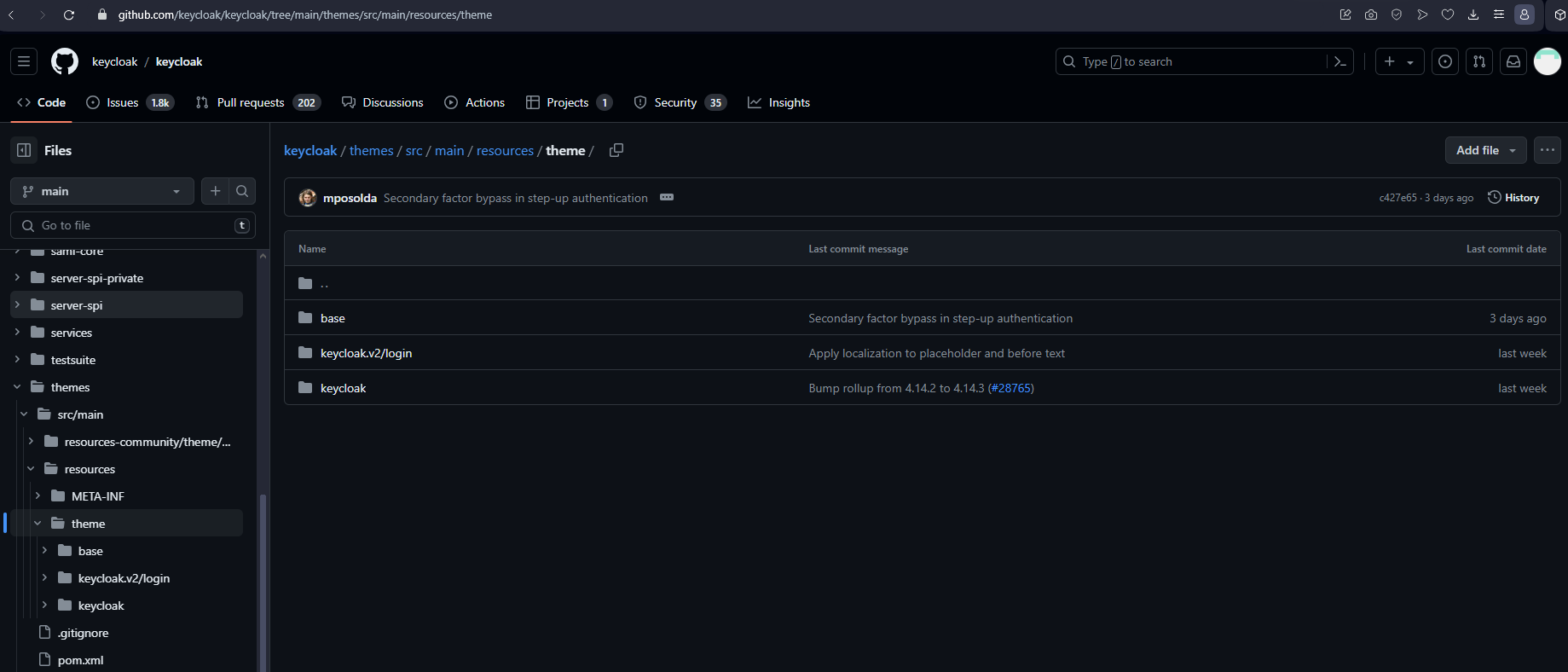
### Modifier l’image – Thèmes

Dans cette partie je vais présenter comme créer son image de Keycloak mais surtout comme créer et personnaliser son thème Keycloak.  
Il faut tout d’abord savoir que dans Keycloak il existe 4 zones où l’on peut appliquer un thème.

* Le thème de connexion (page de login)
* Le thème du compte (page du account)
* Le thème de la UI d’administration (page de gestion de Keycloak)
* Le thème pour le courriel (thème utilisé pour les emails envoyés via Keycloak)

Et pour chacune de ces zones l’on peut choisir de mettre un thème ou non.  
Par défaut dans Keycloak il existe 4 thèmes :

* base
* keycloak
* keycloak.v2
* keycloak.v3

Tous ces thèmes ne sont pas utilisables dans toutes les zones certains sont spécifiques à une seule zone. Et nous la zone qui nous intéresse c’est la zone « login ».  
Car l’on veut que la page de connexion que les étudiants rencontrerons soit personnalisée dans le style HELMo.  
  
Pour commencer il est nécessaire de se rendre dans le github de keycloak.  
Il faut ensuite naviguer jusque : « themes/src/main/resources/theme »  
C’est dans ce répertoire que je l’on retrouve les thèmes de keycloak.  
  


Maintenant il faut savoir ce que l’on veut faire, soit l’on modifie un des thèmes qui existe avec nos besoins, soit l’on crée notre propre thème avec comme base l’un des thèmes existant (ce que l’on va préférer).  
Il faut connaitre l’utilité du fichier theme.properties pour comprendre comment fonctionne l’utilisation d’un thème existant. Dans le fichier theme.properties l’on peut définir le parent de notre thème et tous les thèmes se base sur le thème « base ».  
Par exemple le thème keycloak a comme parent le thème « base » ce qui signifie qu’il a comme caractéristiques toutes celles de « base ». Mais c’est récursif, c’est-à-dire que par exemple le thème « keycloak.v2 » a comme parent le thème « keycloak » qui lui-même comme parent le thème « base » et donc le thème « keycloak.v2 » a les caractéristiques de « base » et « keycloak ». Et nous pour personnaliser notre page de login l’on va utiliser le thème « keycloak ».  
Créons maintenant notre thème que l’on nommer « helmo », je l’ai créé au même niveau que mon fichier Dockerfile car dans celui-ci j’ai spécifié que le chemin était « . ».

Une image contenant texte, capture d’écran, logiciel, multimédia

Description générée automatiquement

Première chose à faire c’est de créer dans notre thème le fichier « theme.properties ».  
Dans celui-ci l’on va définir deux choses : que le parent est le thème « keycloak » et que le chemin d’importation est « common/keycloak ».

Une image contenant logiciel, Logiciel multimédia, Logiciel de graphisme, Icône d’ordinateur

Description générée automatiquement

Une fois créé il faut savoir ce que l’on veut modifier, pour notre part l’on veut modifier le background, le logo et les écritures. Pour cela on aura besoin de modifier le fichier « login.css » et les images dans « img ».  
Donc l’on va créer les chemins de dossiers qui contiennent les fichiers que l’on veut modifier.

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

Maintenant que l’on a créé nos dossiers et nos fichiers (login.css et les deux images), l’on doit modifier dans le fichier « login.css » premièrement les chemins vers les images keycloak pour les orienter vers nos images HELMo.

Une image contenant capture d’écran, texte

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Voici les deux endroits à modifier, pour le logo je conseille remplacer la configuration de base par la mienne car j’ai fait en sorte que le logo reste toujours à un bon endroit dans de bonnes proportions.

Une fois que cela est fait vous avez déjà une version de votre thème avec des images personnalisées. Maintenant l’on veut modifier le texte, pour cela l’on va devoir utiliser un fichier qui se nomme « messages\_en.properties », en fait ce fichier contient toutes les valeurs des variables utilisées dans tout ce qui est affichage html. Et nous l’on souhaite traduire en français et ajouter les termes spécifiques à HELMo dans la page de login.  
Pour cela l’on va devoir créer le répertoire « messages » dans le répertoire « login » et dans ce nouveau répertoire créer un fichier « messages\_fr.properties » qui est la version française des variables.

Une image contenant texte, Police, logiciel, capture d’écran

Description générée automatiquement

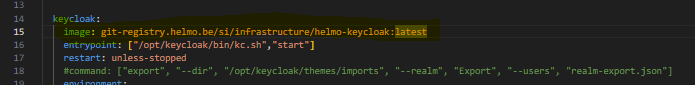
Ci-joint dans le fichier « messages\_fr.properties » , vous retrouverez le fichier déjà configuré avec les besoins d’HELMo. Maintenant que l’on a le fichier français configuré il est aussi important de configurer celui en anglais avec les caractéristiques d’HELMo comme les liens ou les mots comme matricule.

Maintenant l’on va créer l’image car il faut accéder au panneau de configuration de Keycloak.  
Pour cela il faut avoir créer le fichier Dockerfile et maintenant cliquer sur   
Une fois effectué, l’on va se rendre dans le panneau de configuration de Keycloak après avoir créé l’image keycloak-helmo.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Maintenant que l’image est créée et définie et que l’on a défini dans notre fichier docker-compose.yml l’utilisation de notre image personnalisée l’on peut lancer le « docker-compose.yml ».



Maintenant que le docker est up, l’on accède à notre realm helmo. Dedans dans la configuration du domaine, l’on va se rendre dans « localization » et activer l’internationalization. Cela permet d’avoir plusieurs langues définies dans keycloak, puisque l’on souhaite avoir l’anglais et le français, l’on va sélectionner l’anglais et le français dans les locales supportées. Et en locale par défaut l’on va mettre français parce que HELMo est une école francophone.

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Nous allons maintenant personnaliser notre réinitialisation de mot de passe avec un redirect, personnaliser les boutons disponibles sur la page de login et modifier l’affichage d’onglet de la page web.

Une image contenant texte, capture d’écran, Système d’exploitation

Description générée automatiquement

Comme l’on peut le voir le fond est modifié, le logo aussi, le texte en français et personnalisé pour HELMo, il y a le menu déroulant de la langue.  
Maintenant il faut se rendre de nouveau dans keycloak UI.  
Dedans comme expliqué précédemment l’on doit avoir activé « Forgot password », « remember me » et « login with email ».

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Ce qui est important c’est que le bouton « forgot password » soit un redirect vers la page mot de passe oublié d’HELMo. Pour cela nous devons créer le fichier login.ftl dans /login/ .

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Dans « login.ftl », l’on doit modifier la ligne de reset password pour y modifier le bouton de reset password pour qu’il redirige vers la page de réinitialisation officielle d’HELMo.

Une image contenant texte, capture d’écran

Description générée automatiquement