Portale Federato

Documentazione valida per la versione 1.0.1 del portale

Introduzione

Il portale federato è uno strumento che permette all'utente di visualizzare più applicativi web (erogati da *Geoweb* o esterni), e più in generale risorse (documenti), in un unico menu strutturato ad albero. Con questo sistema si potrà accedere più facilmente a ogni contenuto, su cui l'utente è abilitato, semplicemente navigando l'albero di sinistra (vedi struttura). I contenuti dell'albero sono generalmente erogati, da dei provider che vanno esplicitamente dichiarati (vedi il file provider.json), oppure sono risorse esterne esplicitamente dichiarate nella struttura (vedi il file template.json) La modalità con cui questi contenuti vengono strutturati nell'albero può essere:

- Modalità totalmente automatica
- Modalità derivante da configurazione (vedi file template.json/configuration.properties)
- Modalità mista (mix delle precedenti)

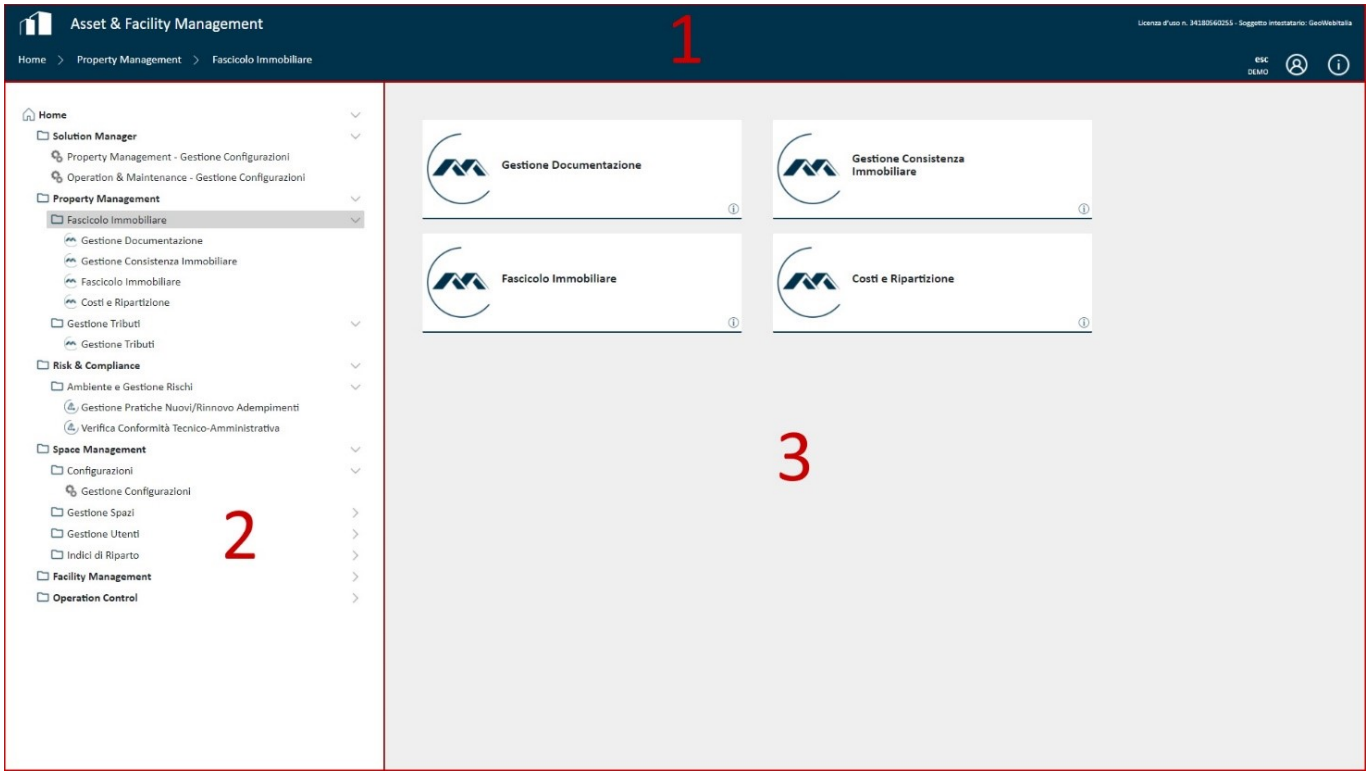
I contenuti dell'albero effettivamente visualizzati, varieranno di volta in volta in base al gruppo con cui si accede al portale, mediante il quale vengono filtrati in base ai permessi. Inoltre grazie ad un sistema di SSO (esterno, tipicamente Keycloak o l'SSO del cliente) l'utente non dovrà effettuare ulteriori login quando vorrà accedere all'applicazione scelta.

I ruoli, in base a cui vengono filtrati i contenuti, hanno dominio nel portale e derivano direttamente dal sistema di SSO utilizzato (tipicamente Keycloak o custom del cliente). Questi ruoli definiti nel portale, sono trasversali a tutti i provider. Ogni provider viene interrogato sulle risorse accessibili dall'utente corrente, che si porta dietro tutti i ruoli associati. I gruppi che tipicamente vengono configurati in *Geoweb* possono essere totalmente svincolati da questi ruoli (ma comunque rimappati tramite opportuni meccanismi)

Struttura

Il portale è suddividibile in tre aree:

1. Toolbar
2. Albero di Navigazione
3. Contenuto



Toolbar

Nella toolbar troviamo, a sinistra, il titolo del portale preceduto, opzionalmente, da una icona riferita al portale (configurabili rispettivamente tramite le proprietà *label* e *labelBanner* nella proprietà del template.json) Sempre a sinistra, ma più in basso, troviamo il percorso attuale. Il percorso mostra il punto in cui ci troviamo nell'albero ed è navigabile, ovvero è possibile tornare ai nodi precedenti cliccando su di essi. A destra troviamo, in alto, un sommario delle info sulla licenza d'uso. In basso troviamo, in ordine:

1. Utente (sopra) e gruppo (sotto)
2. un button che apre un menu contestuale, con la possibilità di cambiare gruppo ed opzionalmente effettuare il *logout* (se non disabilitato nel configuration.properties)

3. un button *Info*, che al click apre un dettaglio sulla licenza

Albero di Navigazione

Permette di navigare tra i vari nodi visualizzandone di volta in volta la sottostruttura nell'area 'Contenuti' sulla destra. I nodi possono essere espansi e collassati a piacimento. L'ultimo nodo selezionato è evidenziato.

Contenuto

Nel contenuto troviamo tutto quello che è presente nel nodo selezionato dall'albero di navigazione.

Nell'esempio precedente troviamo, su sfondo blu, le *sotto-cartelle* e, su sfondo bianco, le *applicazioni* presenti all'interno di un determinato nodo. Sono sempre visualizzate per prima le eventuali sotto-cartelle, e più in basso le eventuali applicazioni del nodo corrente. Al click sulla cartella verrà aperto il nodo dell'albero corrispondente e mostrato tutto il suo contenuto. Al click sull'applicazione questa sarà aperta in una nuova finestra. Sono disponibili tooltip per le applicazioni, utili qualora le stringhe label/description siano particolarmente estese.

Configurazione

Per poter visualizzare correttamente il portale è necessario un primo passaggio di configurazione dei file:

- providers.json
- template.json
- configuration.properties

Tutti i file sono obbligatori. In assenza di configurazioni significative possono tuttalpiù essere lasciati vuoti. In sintesi mentre il *providers.json* si occupa di censire i vari erogatori di contenuti, il *template.json*, unito al *configuration.properties*, descrive come essi debbano eventualmente essere organizzati.

providers.json

Esempio di *providers.json*n vuoto:

providers.json

```
[
]
```

Il *providers.json* viene utilizzato per censire gli erogatori di contenuti. Tipicamente vengono censiti applicativi *Geoweb* (ma nulla vieta di censire applicativi che implementano gli stessi servizi erogati da *Geoweb* per la costruzione dell'albero). Applicazioni, documenti, link esterni a *Geoweb* NON devono essere censiti qui, ma configurati nel template.json. I ruoli, sulla cui base a cui vengono filtrati i contenuti dell'albero, hanno dominio nel portale e, in genere, derivano direttamente dal sistema di SSO utilizzato (tipicamente *Keycloak* o custom del cliente). Questi ruoli definiti nel portale, sono trasversali a tutti i provider. Nel caso semplice, i ruoli associati all'utente, che ha fatto login nel portale, esistono già come Gruppi nei vari provider Geoweb censiti. Spesso accade che i gruppi configurati negli applicativi *Geoweb* siano totalmente svincolati da questi ruoli. Nelle aziende strutturate spesso gli utenti sono già racchiusi in vari ruoli, codificati all'interno dell'azienda, derivanti dal loro modello organizzativo. Tali aziende spesso richiedono espressamente di mantenere la loro strutturazione dei ruoli. In questi casi, tipicamente, nelle installazioni dei vari provider si utilizzano plugin di *Geoweb*, che dichiarano dei servizi custom che implementano l'interfaccia *UserDetailsService* (*org.springframework.security.core.userdetails.UserDetailsService*), opportunamente coordinati con la configurazione dello spring-security.xml. Il compito di tali servizi è recuperare, tipicamente tramite apposite tabelle di relazione, i gruppi di *Geoweb* mappati con i ruoli provenienti dal Portale. C'è assoluta libertà nel mappare i ruoli del cliente con i gruppi di *Geoweb*: si possono creare associazioni 1:1, cosi come configurare associazioni N:1 o 1:N. I gruppi di Geoweb cosi recuperati seguono poi i vari flussi tipici di Geoweb. Sarà compito del portale, di volta in volta, apire gli applicativi Geoweb con il gruppo che Geoweb si aspetta. Dato che l'UserDetailsService e le sottostanti tabelle sono dominio dei vari provider, anche il Portale deve essere a conoscenza delle associazioni fra i *ruoli trasversali del portale* e *gruppi censiti nel singolo applicativo* in Geoweb. Questi vincoli vengono dichiarati tramite la proprietà *groupsMapping* nel *providers.json*.

Esempio, estratto:

estratto_providers.json

```
[
    {
        "id": "1",
        "name": "property_management",
        "url": "https://gwdemo.gwcloud.it/property",
        "groupsMapping": [
            {
                "sourceName": "PMA_manager",
                "targetName": "Uff. Patrimonio",
                "groupLabel": "Uff. Patrimonio"
            },
            {
                "sourceName": "TRAINING",
                "targetName": "DEMO",
                "groupLabel": "DEMO"
            },
            {
                "sourceName": "CA_SolutionManager",
                "targetName": "Solution Manager",
                "groupLabel": "Solution Manager"
            }
        ]
    },
...
]
```

La configurazione prevede una lista di oggetti in formato JSON, i quali hanno le seguenti proprietà:

- **id**: identificativo, obbligatorio. Deve essere univoco nel file. Ammessi numeri e lettere maiuscole e minuscole. No spazi. Unico carattere speciale ammesso il punto; . (regex validazione: “^((?=[A-Za-z0-9.]).)*$”)
- **name**: nome del provider. Questa informazione va nella proprietà *provider* di un object JSON *app* nel *template.json* (Ammessi numeri e lettere maiuscole e minuscole, anche accentate. No spazi. Caratteri speciali ammessi: ,.:'-_() /) (regex validazione: “^((?=[A-Za-z0-9&àèéìòù,.:'-_() /])(?![|!\\\”£$%{}ç°#§\\\\=?^*@+;<>]).)*$“)
- **url**: l’indirizzo base del provider da chiamare, obbligatorio, non può essere vuoto
- **groupsMapping**: opzionale, lista di object JSON, dove vengono mappati i gruppi del portale(***targetName***) con quelli di *Geoweb* (***sourceName***). Questa proprietà, come scritto nella tabella sotto, è obbligatoria, ma può anche essere una lista vuota ([]). Se dovesse essere così il contenuto che verrà restituito da quell’ambiente non sarà visibile con nessun gruppo, a meno che il ruolo del portale non sia già di fatto uguale ad un gruppo censito nell’applicativo *Geoweb*

Proprietà	Obbligatorio	Univoco	Tipo	Caratteri/Valori ammessi
id	SI	SI	String	A-Z a-z 0-9 .
name	SI	NO	String	A-Z a-z 0-9 , . : ' - _ ( ) /

Proprietà	Obbligatorio	Univoco	Tipo	Caratteri/Valori ammessi
url	SI	NO	String	*
groupsMapping	SI	NO	JSON list	JSON List

template.json

Il *template.json* permette di configurare la struttura dell'albero di navigazione. Esempio di *template.json* vuoto:

template.json

```
{
    "banner": "geoweb-logo-hq.jpg",
    "labelBanner": "portal_logo.jpg",
    "label": "Asset & Facility Management",
    "license" : {
        "number": "34180560255",
        "productOwner": "GeoWebItalia",
        "representative": "GeoWebItalia",
        "representativeRole": ""
    },
    "treeData": {
        "id": "home",
        "name": "Home",
        "label": "Home",
        "icon": "",
        "open": true,
        "groups": [
        ],
        "apps":[]
    }
}
```

Per prima cosa è necessario configurare le proprietà:

- **banner**: String, obbligatorio, immagine da usare come logo del cliente che troviamo nel caricamento iniziale del portale. Può avere come valore:
  - il nome del file (es.: name.png"), l'immagine dovrà essere messa nei contenuti statici, sotto il percorso "WEB-INF\classes\META-INF\static-resources\images\"; da questo percorso potranno essere create anche delle sottocartelle a piacimento, da specificare insieme al nome del file (es: "folder_img/name.png")
  - immagine in formato base64 (es.: "...")
- **labelBanner**: String, opzionale, logo del cliente che troviamo nella toolbar (vedi struttura). path dell'immagine rispetto ai contenuti statici, o essere espressa nel formato base 64
- **label**: String, opzionale, etichetta che troviamo della toolbar (vedi struttura)
- **license**: object JSON, obbligatoria, contiene i dettagli licenza d'uso (vedi struttura) Le proprietà *productOwner* e *number* non possono essere lasciate vuote

Nella toolbar del portale possiamo trovare sia *labelBanner* che la *label*, indipendentemente l'una

dall'altra. **Vincolo**: almeno una delle due deve essere valorizzata.

Ulteriori configurazioni possono essere necessarie a seconda della modalità di popolamento dell'albero di navigazione voluta:

- Modalità totalmente automatica
- Modalità derivante da configurazione (vedi file template.json/configuration.properties)
- Modalità mista (mix delle precedenti)

Modalità totalmente automatica

La modalità di popolamento automatica è sempre attiva. Tutti i contenuti (in particolare gli applicativi *Geoweb*) erogati dai vari provider, se ci sono i relativi permessi per il gruppo dell'utente corrente, vengono in una qualche modalità resi disponibili nell'albero di navigazione. L'attivazione perpetua della modalità automatica fa si che non si debba modificare il template.json, ogniqualvolta si aggiunge una qualsivoglia risorsa (es: applicativo *Geoweb*), su cui l'utente abbia i relativi permessi di apertura: essa comparirà automaticamente nell'albero senza dover in alcun modo modificare la configurazione del portale.

I contenuti sono resi disponibili nell'albero di navigazione secondo le seguenti regole.

Modalità automatica ad 1 livello

Se l'applicativo (AKA gwProject Geoweb) non è dichiarato esplicitamente nel template.json (vedi modalità derivante da configurazione, sotto), esso viene aggiunto sotto una folder di primo livello (direttamente sotto il nodo *Home*), chiamata come la *projectCategory* (Categoria di Progetto) configurata nei metadati *Geoweb* dello specifico provider. Se la folder non è per una qualunque ragione già presente (in quanto configurata nel *template.json* o già automaticamente creata), essa viene creata ed aggiunta in modo da rispettare l'ordine del *name* delle cartelle (DESC ORDER). Gli applicativi senza Categoria di Progetto vengono aggiunti su una folder '*Altri*' (scritta localizzata), posta in ultima posizione.

Modalità automatica a 2 livelli

E stata introdotta anche una seconda modalità automatica, che va esplicitamente abilitata dal file configuration.properties tramite il flag:

```
webportal.tree.unboundedProjects.organizeByProvider=true
```

Con questa modalità l'albero viene automaticamente organizzato su due livelli:

1. **provider**
2. **projectCategory** (*Categoria di Progetto*)

Vengono utilizzate le seguenti convenzioni:

- I provider verranno usati per fare cartelle nel primo livello, derivandoli dal *name* secondo la trasformazione

```
    private String computeProviderFolderName(String providerName) {
        String providerFolderName = null;
        if(providerName!=null) {
            providerFolderName =
WordUtils.capitalizeFully(providerName, new char[] {'_'});
            providerFolderName = providerFolderName.replace("_", "
").trim();
        }
        return providerFolderName;
    }
```

La quale sostituisce i '_' con degli spazi ' ', e mette le prime lettere di ogni parola in maiuscolo. Quindi, per esempio, un provider chiamato 'risk_&_compliance' genererà una folder di primo livello denominata 'Risk & Compliance'

- L'ordinamento usato sarà il medesimo con cui sono censiti i provider nel file providers.json
- Nel secondo livello verranno eventualmente create cartelle usando come name i vari *projectCategory*
- I progetti senza *projectCategory* saranno direttamente inseriti sotto le folder di primo livello.
    - L'ordinamento in questo caso sarà per *projectCategory ASC*
- I progetti all'interno di una qualsiasi folder, saranno ordinati per il *name*, anche se ne verrà mostrata la label/description
- L'*immagine* visualizzata sarà quella *configurata nei metadati del provider*.

Questa feature è disponibile dalla versione del webportal **1.0.1**.

Dettagli su questa modalità automatica qui |issue #1

Modalità derivante da configurazione

Si può configurare in maniera molto precisa le risorse, la loro posizione ed il loro aspetto nel *template.json*. Ciò è realizzato modificando coerentemente la proprietà *treeData* . Essa è composta da due tipi di liste:

- Lista object JSON groups
- Lista object JSON apps

Liste groups (di fatto sottocartelle) possono essere innestate dentro elementi group (cartelle) a piacere, senza vincoli di profondità. Ogni elemento group può avere definiti al suo livello un certo numero di elementi app (applicativi/risoprse esterne)

Esempio di object treeData:

```
...
"treeData": {
    "id": "home",
    "name": "Home",
    "label": "Home",
    "icon": "",
    "open": true,
```

```
    "groups": [
        {
            "id": "sm",
            "name": "Solution Manager",
            "label": "Solution Manager",
            "icon": "",
            "open": false,
            "groups": [
            ],
            "apps": [
                {
                "id": "sm.1",
                "name": "Property Management - Gestione Configurazioni",
                "descr": "Modulo applicativo per la gestione degli archivi
di base e delle configurazioni",
                "provider": "property_management",
                "uri": "/project/Gestione_Configurazioni",
                "type": "Applicazione",
                "version": "1.0",
                "icon": "CA_Configuration.jpg",
                "links": []
                },
            ...
            ]
        }
    ],
    "apps":[
    ]
}
```

groups

Corrisponde ad una cartella e può contenere al suo interno altre cartelle o delle risorse (dentro apps)
Un object *group* ha per proprietà:

- **id**: String, obbligatorio. L'identificativo. Deve essere univoco nel file. Ammessi numeri e lettere maiuscole e minuscole. No spazi. Unico carattere speciale ammesso il punto; . (regex validazione: "^((?=[A-Za-z0-9.]).)*$")
- **name**: String, obbligatorio, la label della cartella (Ammessi numeri e lettere maiuscole e minuscole, anche accentate. No spazi. Caratteri speciali ammessi: , . : ' - _ ( ) / (regex validazione: "A-Z a-z 0-9 & à è é ì ò ù , . : ' - _ ( ) /")
- **label**: String, obbligatorio, sarà la label della cartella. Non ammessi spazi in cima o in fondo ne il carattere " (regex validazione: "^((?![\"]).)*$")
- **icon**: String, obbligatorio, tuttalpiù vuoto, icona del raggruppamento. La proprietà può avere come valore:
  - il nome del file (es.: "image.png"), l'immagine dovrà essere messa nei contenuti statici, sotto il percorso *"WEB-INF\classes\META-INF\static-resources\images\"*; da questo percorso potranno essere create anche delle sottocartelle a piacimento, da specificare insieme al nome del file (es: "folder_img/image.png")
  - immagine in formato base64 (es.: "...").

Se viene lasciata vuota verrà utilizzata l'immagine di default per le cartelle

- **open**: boolean, opzionale, default false, indica se all'avvio il nodo sarà aperto o chiuso. Ne può essere dichiarato aperto solo uno per ogni livello. Un'eccezione verrà generata in caso di multipli open a *true* per lo stesso livello
- **groups**: Lista object JSON, obbligatorio, tuttalpiù vuoto [], può contiene una lista di object JSON *group*
- **apps**: Lista object JSON, obbligatorio, tuttalpiù vuoto [], può contiene una lista di object JSON *app*

Proprietà	Obbligatorio	Univoco	Tipo	Caratteri/Valori ammessi
id	SI	SI	String	A-Z a-z 0-9 .
name	SI	NO	String	A-Z a-z 0-9 , . : ' - _ ( ) /
label	NO	NO	String	A-Z a-z 0-9 caratteri speciali (no ")
icon	NO	NO	String	A-Z a-z 0-9 caratteri speciali
open	SI	NO	Boolean	true false
groups	SI	NO	JSON list	folder object
apps	SI	NO	JSON list	app object

Esempio minimale di object JSON group:

```
{
    "id": "sm",
    "name": "Solution Manager",
    "label": "Solution Manager",
    "icon": "",
    "open": false,
    "groups": [],
    "apps": []
}
```

apps

Corrisponde ad un contenuto (applicativo Geoweb | link esterno | documento | etc..) ospitato sotto una qualche folder (*group*) Un object *app* ha per proprietà:

- **id**: String, obbligatorio. L'identificativo. Deve essere univoco nel file. Ammessi numeri e lettere maiuscole e minuscole. No spazi. Unico carattere speciale ammesso il punto; . (regex validazione: "^((?=[A-Za-z0-9.]).)*$");
- **name**: String, obbligatorio, la label del contenuto app (Ammessi numeri e lettere maiuscole e minuscole, anche accentate. No spazi. Caratteri speciali ammessi: , . : ' - _ ( ) / (regex validazione: "A-Z a-z 0-9 & à è é ì ò ù , . : ' - _ ( ) /")
- **descr**: String, obbligatorio, la descrizione del contenuto *app*. Può essere lasciato stringa vuota.
- **provider**: String, obbligatorio, in un'applicazione di *Geoweb* corrisponde al nome del provider configurato nel *providers.json* e serve ad identificare l'ambiente che ospita l'applicativo; questo perché potremmo trovare alcuni progetti con lo stesso nome in ambienti diversi. Se si sta configurando un'applicazione esterna a *Geoweb*, un documento, un link, ecc. allora la sua configurazione prevede la stringa: "external-app";
- **uri**: String, obbligatorio per gli applicativi Geoweb,

    - in un'applicazione di Geoweb, serve ad identificare il progetto e sarà formato dalla stringa ***/project*** e il nome del progetto, ad esempio *"/project/area_bim_manager"*; serve ad identificare il progetto.
    - nel caso di un'applicazione esterna a Geoweb, un documento, un link, ecc. l'uri non sarà necessario;
- **type**: String, obbligatorio, tipo di contenuto (Applicazione, Documento, ...), viene visualizzato al passaggio del mouse sopra all'icona (vedi contenuto);
- **version**: String, obbligatorio, versione del contenuto (Applicazione, Documento, ...), viene visualizzato al passaggio del mouse sopra all'icona (vedi contenuto);
- **icon**: : String, obbligatorio, tuttalpiù vuoto, icona del raggruppamento. La proprietà può avere come valore;
    - il nome del file (es.: "image.png"), l'immagine dovrà essere messa nei contenuti statici, sotto il percorso *"WEB-INF\classes\META-INF\static-resources\images\"*; da questo percorso potranno essere create anche delle sottocartelle a piacimento, da specificare insieme al nome del file (es: "folder_img/image.png")
    - immagine in formato base64 (es.: "...").

Se viene lasciata vuota verrà utilizzata l'immagine di default per le app

- **links**: lista object, obbligatori, tuttalpù vuoto []
    - in un'applicazione di *Geoweb* verrà popolato in automatico: ma deve comunque essere dichiarato come lista vuota [] (vedi tabella sotto)
    - Nel caso di un'applicazione esterna a *Geoweb*, un documento, un link, ecc. esso sarò una lista di object JSON con le seguenti proprietà:
        - **label**: etichetta del link, opzionale;
        - **link**: l'url del contenuto, obbligatorio;
        - **groupName**: il nome del gruppo del portale, obbligatorio;
        - **groupLabel**: etichetta del gruppo, obbligatorio.

Proprietà	Obbligatorio	Univoco	Tipo	Caratteri/Valori ammessi
id	SI	SI	String	A-Z a-z 0-9 .
name	SI	NO	String	A-Z a-z 0-9 , . : ' - _ ( ) /
descr	SI	NO	String	
provider	SI	NO	String	Vedi name providers.json
uri	SI	NO	String	A-Z a-z 0-9 _ /
type	SI	NO	String	A-Z a-z 0-9
version	SI	NO	String	A-Z a-z 0-9 .
icon	NO	NO	String	A-Z a-z 0-9 caratteri speciali
links	SI	NO	JSON list	[]

Esempio minimale di object JSON app:

```
{
    "id": "sm.1",
    "name": "Property Management - Gestione Configurazioni",
    "descr": "Modulo applicativo per la gestione degli archivi di base e
delle configurazioni",
    "provider": "property_management",
    "uri": "/project/Gestione_Configurazioni",
    "type": "Applicazione",
    "version": "1.0",
```

```
    "icon": "CA_Configuration.jpg",
    "links": []
}
```

Modalità mista (mix delle precedenti)

Prevede l'utilizzo simultaneo di entrambe le modalità viste precedentemente. In particolare ha senso configurare il *template.json* per:

- aggiungere, posizionandoli con assoluta precisione, tutti gli oggetti app di type *external-app*, (link esterni, documenti) i quali non sono applicativi erogati dai provider *Geoweb* configurati.
- cambiare il posizionamento automatico degli applicativi *Geoweb*, derivante dalla modalità automatica (su 1 livello o 2 livelli), in quanto se l'applicativo è già dichiarato nel *template.json*, esso mantiene il suo posizionamento. Inoltre anche se non fosse già presente in una qualche folder (AKA elemento JSON group), se la folder viene già trovata al livello prestabilito non viene ricreata. In particolare le cartelle definite nel *template.json*, non create automaticamente, sono aggiunte sotto quelle derivanti dalla modalità automatica. Esse mantengono l'ordine definito nel *template.json* (mentre quelle generate in maniera automatica seguono altri criteri di ordinamento).

configuration.properties

Questo file è obbligatorio e serve ad ospitare tutte le eventuali configurazioni che non trovano posto altrove.

Proprietà

- portale
  - **webportal.providersFileName**: String, opzionale, default providers.json. Serve per poter specificare un file alternativo. Questo è utili soprattutto in contesti dove, previo rimappaggio della proprità sul remapConfiguration.properties ad una variabile di ambiente, l'applicativo passa in maniera più o meno automatica da ambiente di svil a test e prod, e in ogni macchina virtuale *dockerizzata* viene esposta una variabile d'ambiente che punta al giusto file, che a sua volta contiene i providers con i giusti puntamenti (i quali probabilmente cambieranno anchessi nei vari ambienti)
  - **webportal.tree.unboundedProjects.organizeByProvider**: Boolean, opzionale, default false. Quando a true forza la modalità automatica di popolamento dell'albero di navigazione generare un albero strutturato su due livelli: provider/projectCategory
- logout
  - **logout.enabled**: Boolean, opzionale, default false. Quando a false è disabilitata la possibilità di effettuare il logout da UI
  - **logout_url**: St opzionale, valutato solo con logout.enabled a true. E' l'url invocato al click da UI sul menu di logout.
- generale
  - **basePath**: String, required. Utilizzato per dichiarare la folder dei contenuti statici
  - **lang**: String, required. Il locale di default dell'applicativo
- database (da usare come in *configuration.properties* di *Geoweb*)

- JNDI (da usare coordinatamente alle configurazione nello *spring-security.xml*)
  - **jndiName.metadata**
  - **jndiName.data**
- JDBC (da usare coordinatamente alle configurazione nello *spring-security.xml*)
  - **jdbc.driverClassName**
  - **jdbc.url**
  - **jdbc.username**
  - **jdbc.password**
  - **jdbc.maxActive**
  - **jdbc.minIdle**
  - **jdbc.maxIdle**
  - **jdbc.validationQuery**
  - **jdbcmetadata.driverClassName**
  - **jdbcmetadata.url**
  - **jdbcmetadata.username**
  - **jdbcmetadata.password**
  - **jdbcmetadata.maxActive**
  - **jdbcmetadata.minIdle**
  - **jdbcmetadata.maxIdle**
  - **jdbcmetadata.validationQuery**

Esempio configuration.properties:

configuration.properties

```
##########################################################################
# PORTAL BEHAVIOR
##########################################################################
webportal.providersFileName=providers.json
webportal.tree.unboundedProjects.organizeByProvider=true


##########################################################################
# L O G O U T
##########################################################################
logout.enabled=false
#logout_url
#absolute
# url (ex: https://localhost/webapp/logout)
#relative
# url (sso/logout)
#when used with keycloack use the relative
# sso/logout
logout_url=https://gwdemo.gwcloud.it/asfportal/sso/logout


##########################################################################
# B A S E P A T H
##########################################################################
basePath=file:////datastor/deploy.geoweb4/projects/ModuloAEC/WEB/
```

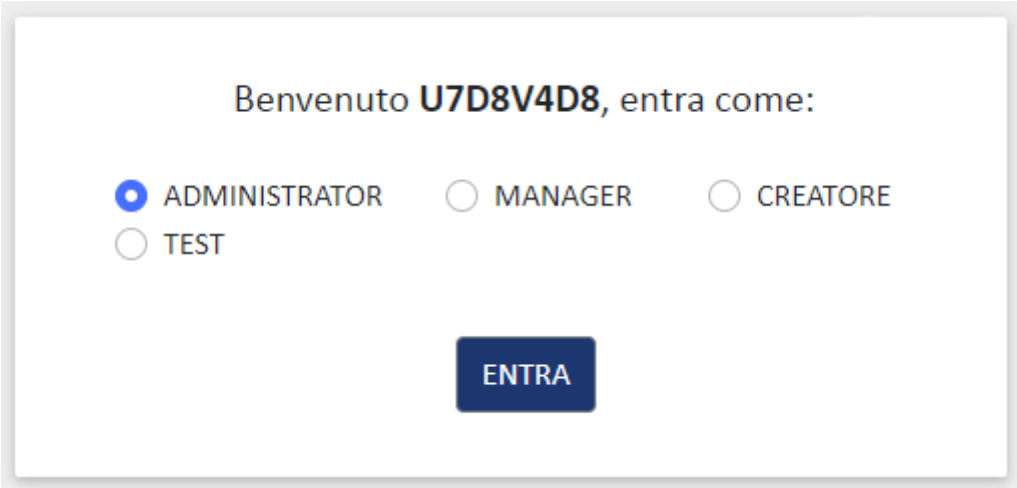
```
#########################################################################
# L A N G U A G E S
#########################################################################
lang=en


#########################################################################
# D A T A B A S E S
#########################################################################

#JNDI DRIVEN
#jndiName.metadata=java:jboss/drei0/gwMetadata
#jndiName.data=java:jboss/drei0/gwData

#########################################################################
# database ORACLE
#########################################################################

#jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbc.url=jdbc:oracle:thin:@ora11dev.gruppoesc.it:1521:ORA11DEV
#jdbc.username=AEC_DBDATI_GW
#jdbc.password=AEC_DBDATI_GW
#jdbc.maxActive=6
#jdbc.minIdle=2
#jdbc.maxIdle=6
#jdbc.validationQuery=select 1 from dual
#
#jdbcmetadata.driverClassName=oracle.jdbc.driver.OracleDriver
#jdbcmetadata.url=jdbc:oracle:thin:@ora11dev.gruppoesc.it:1521:ORA11DEV
#jdbcmetadata.username=AEC_METADATA_GW
#jdbcmetadata.password=AEC_METADATA_GW
#jdbcmetadata.maxActive=6
#jdbcmetadata.minIdle=2
#jdbcmetadata.maxIdle=6
#jdbcmetadata.validationQuery=select 1  from dual


#########################################################################
# database POSTGRES
#########################################################################

jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/AFGW
jdbc.username=space_dati_gw
jdbc.password=space_dati_gw
jdbc.maxActive=6
jdbc.minIdle=2
jdbc.maxIdle=6
jdbc.validationQuery=select 1
```

```
jdbcmetadata.driverClassName=org.postgresql.Driver
jdbcmetadata.url=jdbc:postgresql://127.0.0.1:5432/AFGW
jdbcmetadata.username=space_metadata_gw
jdbcmetadata.password=space_metadata_gw
jdbcmetadata.maxActive=6
jdbcmetadata.minIdle=2
jdbcmetadata.maxIdle=6
jdbcmetadata.validationQuery=select 1


#####################################################################
# database SQLSERVER
#####################################################################

#jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
#jdbc.url=jdbc:sqlserver://192.168.0.99:1433;databaseName=geoweb
#jdbc.username=FMI_MAGLIE_GW
#jdbc.password=FMI_MAGLIE_GW
#jdbc.maxActive=6
#jdbc.minIdle=2
#jdbc.maxIdle=6
#jdbc.validationQuery=select 1
#
#jdbcmetadata.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDri
ver
#jdbcmetadata.url=jdbc:sqlserver://192.168.0.99:1433;databaseName=geowe
b
#jdbcmetadata.username=FMI_MAGLIE_GW_METADATA
#jdbcmetadata.password=FMI_MAGLIE_GW_METADATA
#jdbcmetadata.maxActive=6
#jdbcmetadata.minIdle=2
#jdbcmetadata.maxIdle=6
#jdbcmetadata.validationQuery=select 1
```

Uso del Portale

All’avvio del portale viene mostrata una grafica con il logo del cliente, mentre si procede al recupero dei dati.

A caricamento avvenuto, si potrà scegliere il gruppo con cui entrare nel portale.



Una volta selezionato il gruppo verrà aperta la home page del portale, che presenterà la struttura ad albero filtrata in base ai permessi dell'utente/gruppo correnti.

Si potra quindi navigare il menu ad albero ed aprire gli applicativi e le risorse.

Installazione

Integrazione con Keycloak