D3.1- Bimrocket: Funcionalitats i Desenvolupament

Índex

Objectiu del document 3
Introducció a Bimrocket 4
Funcionalitats de modelatge 8
Funcionalitats generals BIM 10
Regles i informes 15
Gestió Incidències BCF 18
Integració amb GIS 25
Millores i Adaptacions realitzades en el present projecte 26
Dockerització (Comunitat i Pilot) 26
Modificació de sistema Autenticació (Comunitat i Pilot) 26
Passar fitxer de propietats de Back-end (Comunitat i Pilot) 28
Passar fitxer de propietats de front-end com a environment (Comunitat i Pilot) 28
Crear un Mòdul custom que reuneixi eines essencials dels 3 mòduls de la comunitat. (Pilot) 29
Configurar dinàmicament al nou mòdul “custom” de frontend l'URL on es serveixen els serveis de backend. (Pilot) 33
Customitzacions de UI 33
Funcionalitat de canvi de permisos de fitxers i carpetes del repositori models (Comunicat i Pilot) 34
Extensions BCF per defecte des de fitxer de propietats (sempre i quan no hi hagi definit un Project Template) (Pilot) 37
Customització de Filtres BCF (Comunitat i Pilot) 37
Exportació BCF a Excel 39
Bugfix: Error en descàrrega de fitxers ifc de repositori remot 39
Fitxer regles de Autorització Ambiental (Pilot) 40
Funcionalitat Validar i Desar (Pilot) 41
Bugfix: escapament de noms de directori “ifc_project” per tal d’evitar $ 46
Botons d’iniciPeticioInterna , fiPeticioInterna, iniciRequeriment, fiRequeriment, FasePublica, Resolució 46

Inicialització del menú “preferits” incloent per defecte algunes propertySets pre-definides (Pilot) 48
Filtre de vista per Vector/Àrea (Pilot) i Quadre de característiques 49
Eina de Gestió Rols (Comunitat i Pilot) 51
Eina de Gestió Usuaris (Comunitat i Pilot) 52
Adaptació repositori de fitxers per a funcionar en BlobStorage (Comunitat i per a una futura versió productiva Serverless del Pilot) 53
Millores usabilitat Eina BCF i filtre de projectes BCF (comunitat i Pilot) 55
Autenticació via Token bearer 56
Integració oAuth (VALid, Keycloak, GICAR) 57
Eina de Capes GIS 62
Desplegament de la Plataforma per a desenvolupadors 63
Descarregar codi i compilar tant front-end com backend 63
Desplegament Backend i Frontend no dockeritzat utilitzant IntelliJ 65
Solucions a problemes originats per inteliiJ 72
Generació de les classes IFC utilitzades pel Front-end treballant amb Visual Code 73
Desplegament Front-end no-dockeritzat amb Visual Code 73
Desplegament del backend i frontend versió Comunitat (docker) 74
Anexe1: Serveis oferts 76
Scripts de Tasques 76

Objectiu del document

El present document té l'objectiu d'explicar en que consisteix l'eina Bimrocket , peça angular del Pilot del present Projecte.

En la primera secció s'introdueix Bimrocket i s'expliquen les funcionalitats de la versió actual de la comunitat. Dita versió ja inclou moltes de les millores realitzades durant el present projecte Pilot. De totes maneres en la següent secció s'expliciten cadascuna de les funcionalitats més importants incorporades durant el present projecte des que es va partir de la versió de Febrer del 2025 fins a final d'aquest mateix any.

Tot seguit també s'inclouen els passos a realitzar per al desplegament de la plataforma, orientat al desenvolupament, per tal que futurs desenvolupadors de la plataforma puguin continuar la feina feta en aquest projecte.

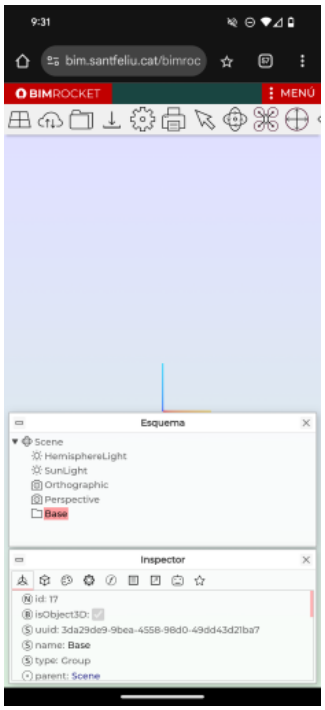
El desplegament en entorns de integració y pre-producció s'explica en el següent entregable T4.1, més enfocat al desplegament Pilot on s'ha de validar la plataforma.

Introducció a Bimrocket

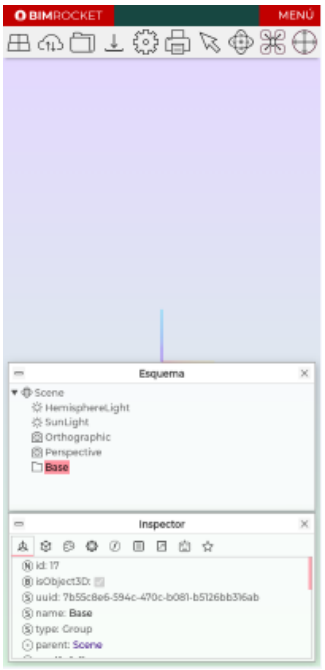
Building Information Modelling. És un paradigma de construcció que es basa en la idea que totes les dades necessàries per a la construcció, autorització, i possiblement futura operació dels Edificis estiguin dintre d'un mateix fitxer (model).
El que és rellevant de la metodologia BIM són les dades que hi ha darrere els plànols tradicionals. L'objectiu principal d'aquesta metodologia és la de facilitar l'edició, compartició i col·laboració per aconseguir definir completament les dades que hi ha darrere l'esquelet del model per a la seva construcció i per la utilització d'aquestes dades durant fases d'operació i manteniment dels edificis (dades guardades en DB, fàcilment analitzables per processos de Business Intelligence).

Bimrocket és una eina Open Source de Modelatge 3D creada per una comunitat fundada per l'Ajuntament de Sant Feliu, que integra les eines principals per poder aplicar la metodologia BIM.
Es recomana visitar la Web informativa (https://bimrocket.github.io/es/)per a entendre millor la plataforma, des de la seva arquitectura, funcionalitats tècniques, manual d'usuari fins a les instruccions per instal·lar-ho i desplegar-ho.
Està desplegada a: bim.santfeliu.cat/bimrocket , per tant, es pot fer servir per analitzar l'eina.
És una eina que s'adapta al format de dispositiu i pantalla, en conseqüència, quan un usuari accedeix a l'eina per mitjà de l'URL d'accés, es detecta automàticament el tipus de dispositiu que està intentant accedir. D'aquesta manera, es pot visualitzar en les següents versions:

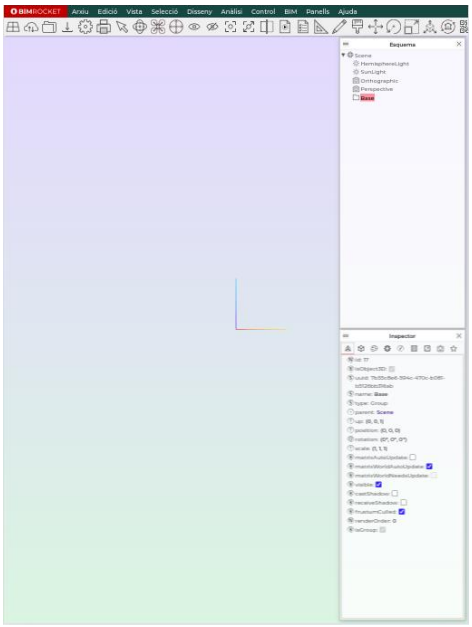
Versió mòbil al navegador Chrome:



Versió responsive del navegador:



Versió tablet del navegador:



La versió de la comunitat BIMROCKET ofereix totes les funcionalitats llistades en la pagina de "features" : https://bimrocket.github.io/es/features/

Per entendre com utilitzar a nivell general l'aplicació cal visualitzar els vídeos de la web: https://bimrocket.github.io/es/user_guides/

Amb els anteriors vídeos podem fer-nos una idea completa de com fer servir bimrocket, de totes maneres a continuació es resumeixen algunes de les funcionalitats més rellevants de la versió comunitat ja existents que ens han estat d'utilitat i d'altres les quals s'ha contribuït al seu desenvolupament dintre del marc del present projecte Pilot.

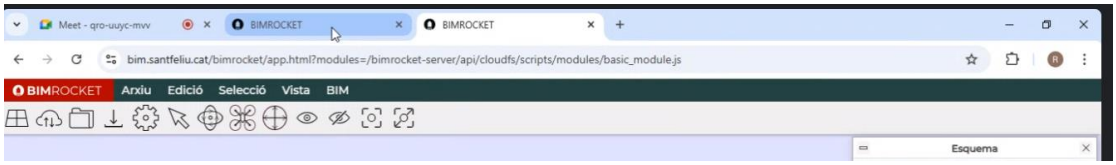
Per defecte en la instal·lació de la versió Comunitat es carreguen els mòduls següents disponibles:

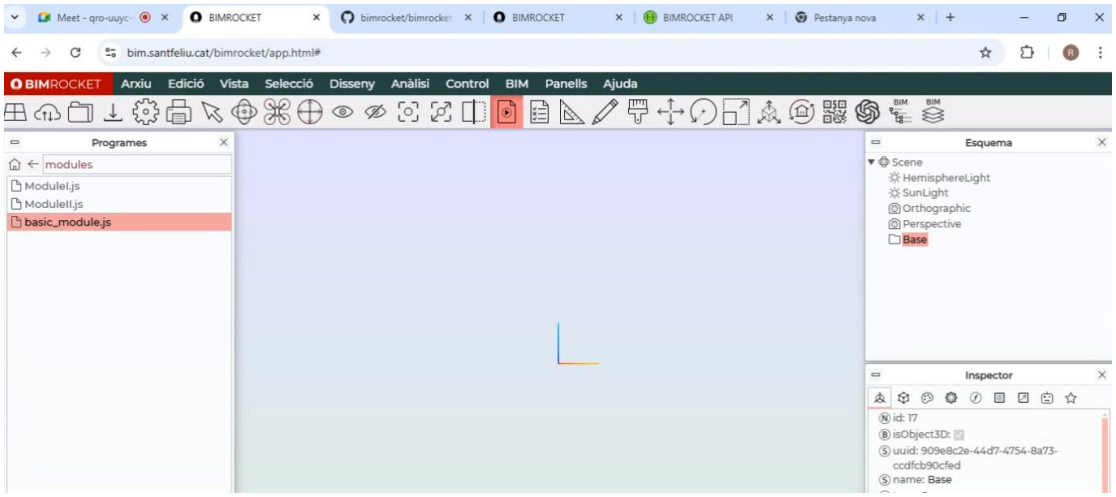
- Base: modelatge genèric
- BIM: funcionalitats BIM
- GIS

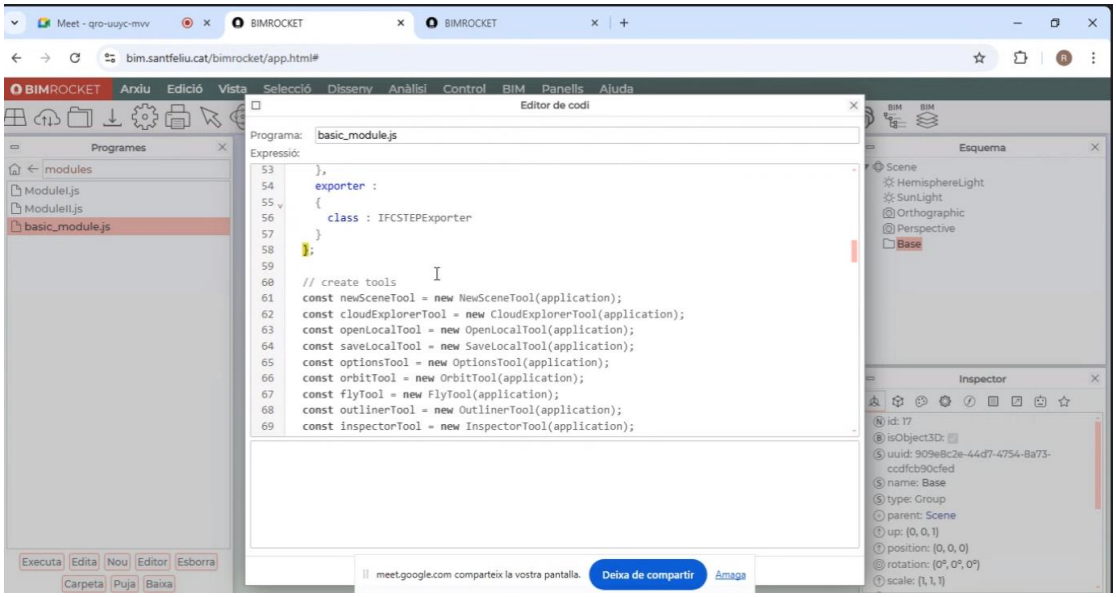
També es poden especificar els mòduls a carregar com argument a la url:

- *?modules=base*
- *?modules=base,gis*
- *=/bimrocket-server/api/cloudfs/scripts/modules/basic_module.js*

A la web de producció de St Feliu:
https://bim.santfeliu.cat/bimrocket/app.html?modules=/bimrocket-server/api/cloudfs/scripts/modules/basic_module.js



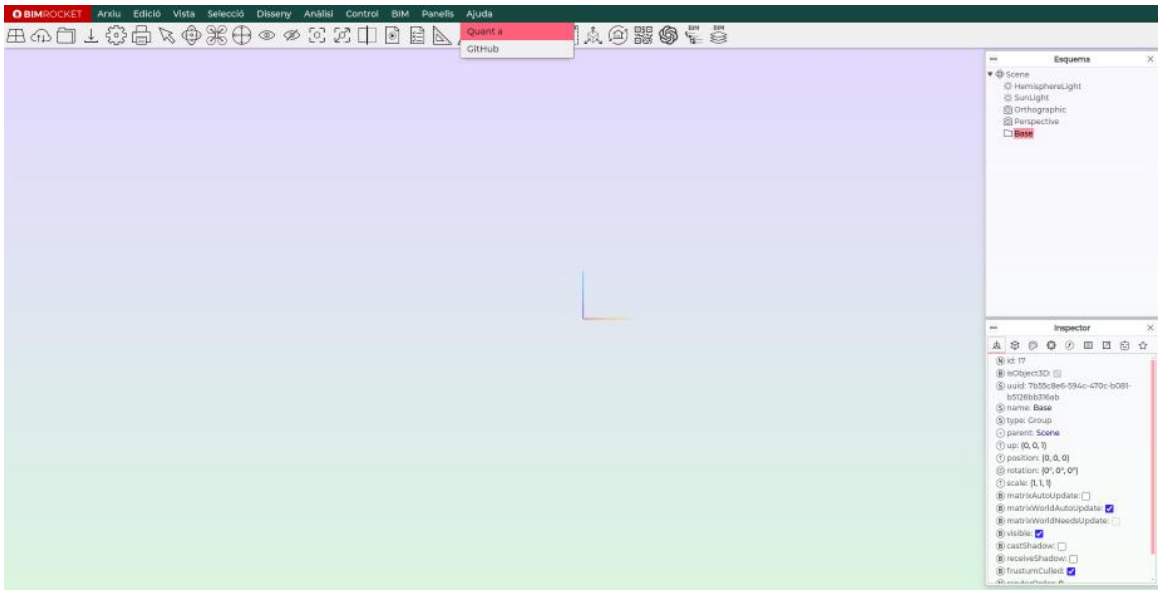




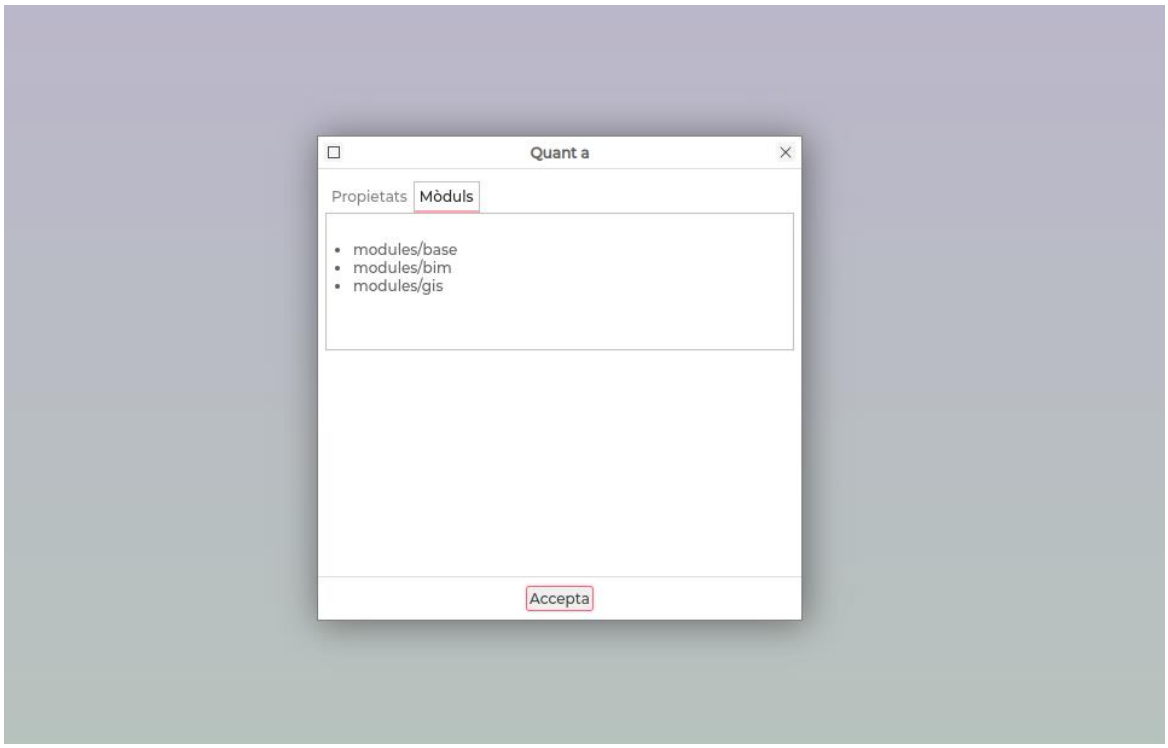
Al desplegament crea les carpetes: scripts, modules i dos més.

Si volem veure els mòduls carregats, cal seguir els passos següents:

- Al Menú superior clicar a ***Ajuda***, després a ***Quant a***:



L'opció ***Quant a*** obre una finestra modal i a la pestanya ***Mòduls*** veiem els mòduls carregats per defecte:



Cadascun d'aquests mòduls es pot desactivar i se'n poden afegir de nous des del codi. Per tant, ho hauria de fer el programador.

Funcionalitats de modelatge

Bimrocket va néixer com a eina de modelatge, per tant, incorpora totes les funcionalitats bàsiques de les eines de modelatge 3D. Les funcionalitats de modelatge són útils sobretot pel sol·licitant, que és la persona que crearà el model i qui haurà de realitzar modificacions.

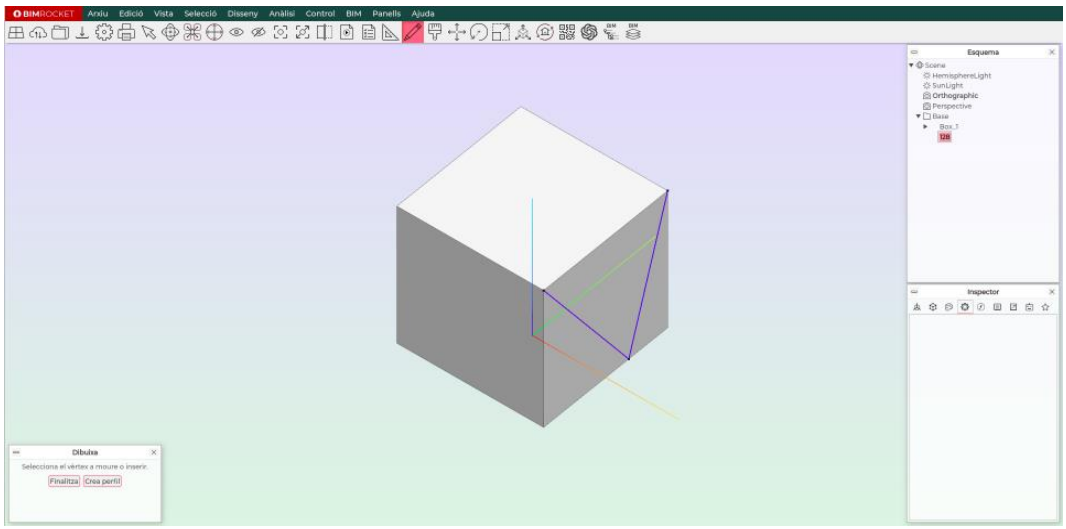
A continuació es llisten algunes funcionalitats relacionades amb el modelatge i com es poden fer servir:

Crear, Obrir Models
L'eina permet la creació d'un nou model buit, així com carregar models desats en el PC local i també en remot (com s'explica més endavant). Es poden carregar varis models al mateix temps.

Creació, Edició i Eliminació d'elements geomètrics

- Perfils: Elements 2D. Poden ser paramètrics o no paramètrics. Quan són paramètrics es poden modificar la seva base i l'alçada si són rectangles, i si són cercles el seu radi, des de l'inspector. N'hi ha de més tipus. Combinen amb altres operacions de disseny com l'extrusió.

- Corda: Són línies 3D que es van dibuixant unint diferents punts sobre els sòlids. També poden combinar-se amb altres operacions com l'extrusió. Per dibuixar una corda pots seleccionar l'acció que té la icona del llapis des del menú d'opcions ràpides:



- Sòlid**:** Element 3D que sol ser el resultant d'aplicar operacions a partir de perfils i cordes.

Selecció d'elements
Cal seleccionar els elements, sigui clicant sobre part d'ells o bé agafant una àrea exterior a l'element amb el ratolí per tal d'aplicar operacions sobre els elements seleccionats.

Creació d'agrupacions d'objectes

Es poden crear grups d'objectes sobre els quals es poden aplicar operacions de manera conjunta.

Operacions

Les operacions que es poden aplicar sobre objectes són:

- **Unió**
- **Intersecció**
- **Resta**

Eines

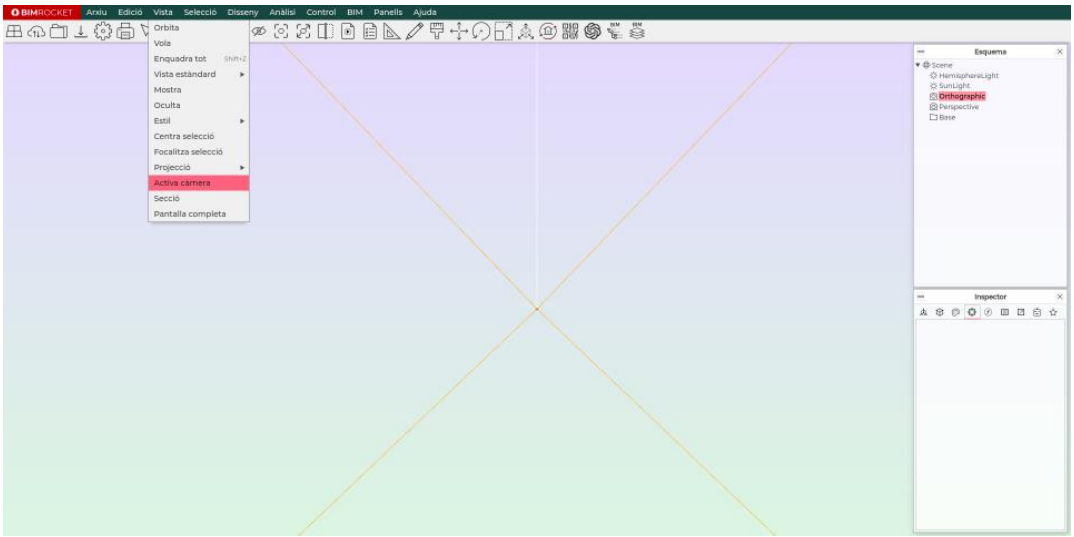
Existeixen dos tipus d'eines:

- Eines d'aplicació immediata que al clicar-les realitzen l'acció directament sobre l'element seleccionat.
- Eines d'aplicació NO immediata que al clicar apareix una finestra emergent que et permet governar l'eina

Múltiples càmeres per anar movent els punts de vista

Hi ha dues càmeres per defecte: *Orthographic* i *Perspective*, ambdues estan llistades a l'Escena de l'Esquema de la dreta.

De vegades pots tenir més de dues càmeres, pots activar cadascuna d'elles seleccionant-la primer des del llistat de càmeres de l'Esquema i clicant al menú superior Vista > Activa càmera.

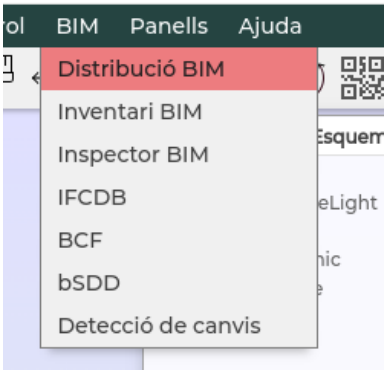


Sempre funciona amb coordenades cartesianes, no utilitza el tipus de coordenades del Google maps. Això no representa cap problema, ja que quan s'integra amb sistemes GIS ja prèviament transforma les coordenades al format cartesià.

Desat

Incorpora diverses maneres de poder desar el model. Quan deses en local es desa a la ubicació del PC local que el navegador tingui configurada. També es pot desar al cloud per mitjà de serveis de backend, tal com s'explica més endavant.

Funcionalitats generals BIM



Les funcionalitats de Bimrocket relacionades amb l'ús de la metodologia BIM permeten al sol·licitant i validadors poder interactuar amb les dades que hi ha darrere els elements geomètrics. Tant el sol·licitant a l'hora d'introduir i autovalidar aquestes dades, com els validadors a l'hora de realitzar comprovacions necessiten utilitzar aquestes funcionalitats.

Format IFC

El format estàndard de BIM és IFC. Bimrocket pot carregar models creats en diferents versions d'aquest format: 2x3, 4, i 4x3 i FUTURES versions.

Cada fitxer pot tenir una mida de l'ordre d'uns 100MB. Òbviament, aquesta mida pot variar segons el contingut, però un projecte estàndard tindria una mida propera a aquest valor.

Càrrega de model IFC

El procés de càrrega d'un model IFC consisteix en les següents fases:

1. Llegir el fitxer en memòria
2. Construir el Graf, construint els objectes IFC.
3. Convertir els objectes del IFC a objectes de l'arbre d'escena.

Exportació a IFC

Es poden exportar els models que estan carregats a Bimrocket a un fitxer IFC, encara que aquesta exportació, igual que passa amb la majoria d'eines comercials, pot portar a certa degeneració de les geometries.

El procés d'exportació consisteix en les següents fases:

1. Convertir l'arbre d'escena a elements objectes IFC.
2. Transformar el graf a nodes IFC.
3. Omplir informació en un fitxer .ifc.

Altres formats compatibles: BRF, OBJ, etc. No els farem servir perquè normalment importar els models en IFC.

Els elements dels edificis normalment tenen els **tipus d'elements** definits en l'estàndard IFC (standards.buildingsmart.org/MVD/RELEASE/IFC4)

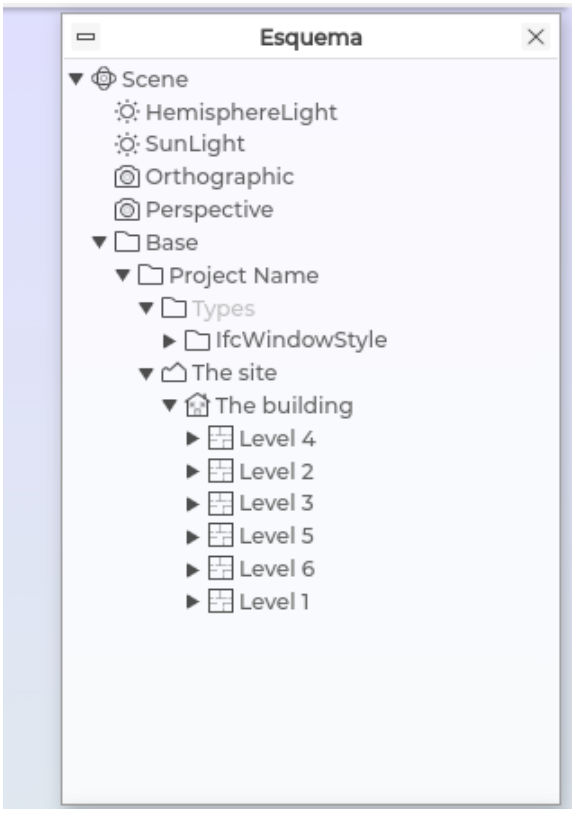
Inspector BIM

Ens mostra la informació de l'IFC.

Esquema.
A la dreta de la pantalla hi ha una finestra que mostra un arbre d'escena, que actua com a inventari de tots els elements del model. Aquesta estructura és una versió simplificada del model IFC.
En l'arbre d'escena (esquema), el model es representa com un arbre jeràrquic, mentre que en el model IFC (inspector BIM) es presenta com un graf.

Organització de l'arbre d'escena:
Al nivell Base hi ha una carpeta principal que conté tots els models carregats: Dins de cada model trobem:

- Una carpeta per als tipus.
- Una carpeta per a l'entorn (*environment*), que inclou:
  - Edificis.
  - Dins de cada edifici, les diferents plantes.
  - Dins de cada planta, els elements específics.



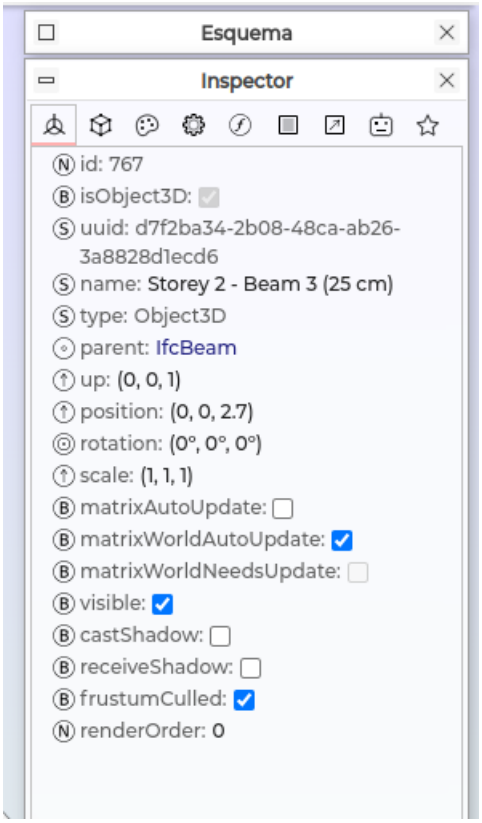
Els elements poden ser d'un tipus determinat (heretant les propietats del tipus) o bé no tenir cap tipus assignat.

Inspector.
A la part inferior de la dreta, hi ha una finestra on es mostren les dades particulars dels elements seleccionats (a l'esquema i a la vista geomètrica).



Trobem diverses pestanyes relatives a tipologies de dades:

- Objecte: modificar la posició i visibilitat, edició d'ombres, etc.
- Geometria: modificar la geometria de l'objecte
- Material: seleccionar materials, colors, opacitat
- Constructor: ofereix detalls de com està construït l'objecte
- Propietats (userData): Són propietats descriptives, com poden ser el pes o el tipus de material. Són editables amb un editor JSON i es poden anar afegint.
- Relacions amb altres objectes. Per exemple, un pany d'una porta podria enllaçar amb una porta.
- Controladors (Robot): Petits programes que actuen sobre un element. p.ex: afegir moviment



Algunes de les anteriors propietats poden ser seleccionades i es poden afegir a “destacades”
Quan fas canvis sobre “l’inspector” has de fer “reconstruir” perquè s’actualitzi la imatge.
Es poden fer visualitzacions de només les tipologies d’objectes que t’interessen (p. ex. objectes de construcció)
Els objectes tenen unes propietats per defecte segons la tipologia.
La utilització de plantilles permeten afegir elements ja dissenyats com “portes”, “finestres”, etc, que internament estan fets d’altres elements IFC. Aquests elements utilitzats en les plantilles també tenen el seu tipus IFC, definit en els DataDictionaris oficials de l'estàndard IFC.
Un **objecte** sempre té una **classe, un tipus, relacions i PropertySets**
Es poden afegir propertySets per tipologies d’objectes o bé associats a objectes concret.
Els PropertySets (Psets) són conjunts de propietats associades a elements d’un model BIM (Building Information Modeling). Aquestes propietats proporcionen informació addicional sobre els elements constructius

Desat remot
Els models es poden desar en un servidor Webdav. Aquesta funcionalitat està implementada mitjançant un servei de backend que s’ha de configurar. Per tant, primer caldria configurar l'URL del servidor. Posteriorment, abans de poder desar, s’han d'introduir les credencials d’accés al servei.

Càrrega des de remot
De la mateixa manera com es pot desar en remot, també es poden carregar models des d'un servidor remot. També s'ha de configurar primer el servidor remot i a l'hora d'utilitzar el server l'usuari s'haurà d'autenticar.

Exportació a DDBB
Bimrocket permet poder exportar un model a una DDBB estructurada (tipus SQL). D'aquesta manera es podria desar un model de manera estructurada en comptes de com a fitxer. Aquesta funcionalitat s'implementa per mitjà d'un servei de Backend.

Importació d'un model o subconjunt d'aquest des de DDBB
De la mateixa manera, podem carregar un model desat en DDBB o inclús un subconjunt d'aquest model que correspon a una consulta a la DDBB que ens retorni un model vàlid. També s'implementa aquesta funcionalitat mitjançant un servei de Backend.

Compatibilitat amb nous elements i propietats IFC.
Els Validadors poden voler validar l'existència i també compliment de valors d'alguns paràmetres que avui dia no estan mapejats com a cap propietat dintre dels elements estàndard actuals IFC que Bimrocket coneix.

D'aquesta manera, nous elements i noves propietats d'aquests elements s'han de poder incloure tant a l'estàndard IFC, com posteriorment poder ser utilitzats dintre de Bimrocket per tal que els creadors del model puguin incloure totes les propietats que els Validadors necessiten conèixer sobre l'edifici, incloent-hi elements que no siguin de caràcter geomètric.

Si volem afegir propietats per a un element determinat, hem d'afegir aquestes propietats dintre de Bimrocket mitjançant la creació de conjunts d'elles (propertySets) i que es poden lligar tant a:

- una o més classes IFC
- a instàncies d'objectes

Aquesta funcionalitat d'afegir nous **PropertySets**, és clau per poder evolucionar l'eina per ser compatible per a realitzar qualsevol mena de validació en diferents tràmits.

Sempre, abans d'afegir un propertySet, la propietat s'hauria d'estandarditzar dintre del CoBIMCAT, que és l'organisme d'estandardització de BIM a Catalunya i BuildingSmart que és una comunitat de transformació digital del món de la construcció a nivell espanyol i una de les més avançades en tecnologia BIM.

Bimrocket permet **carregar Models** que poden incloure propertySets (estàndards i nous que han estat afegits amb altres eines comercials), cosa que permet poder treballar amb aquestes noves propietats .

Regles i informes

Tota la informació de les propietats dels elements del model, ja sigui geomètrica (la informació representada al dibuix), com abstracta o general del projecte, pot ser consultada mitjançant l'arbre d'escena i l'inspector. Però recórrer tots els elements del model pot ser una tasca molt costosa per les entitats validadores. Per aquest motiu, Bimrocket ofereix una solució d'automatitzar aquest procés mitjançant la generació i execució de regles de validació amb la finalitat de validar els valors de les propietats dels elements del model. D'aquesta manera el projectista pot assegurar-se de complir una sèrie de restriccions amb anterioritat a què pugui ser validat per entitats validadores, i evitar així diversos cicles d'anada i vinguda de validació i correcció.

Les regles es troben localitzades a la pestanya ***anàlisi > informes***, o bé fent clic directament a la seva icona de la barra d'eines (marcada amb color vermell en la següent imatge).



El menú d'informes ofereix dues possibles opcions:

- La pestanya "**remote**", on s'accedeix a les regles que ja han estat creades i es troben emmagatzemades al servidor.
- La pestanya "**local**", on s'accedeix a les regles creades i pujades localment a l'ordinador de l'usuari.

Els **fitxers de regles** es poden crear dinàmicament des de la mateixa secció d'informes fent clic al botó "Nou" i seleccionant el tipus de format que ve per defecte, *BIMROCKET Script (BRS)*, aquest format és propi de BIMRocket i és més potent que l'IDS, que és el format estàndard.

Un cop creat l'arxiu, apareix la finestra d'edició d'informe amb la plantilla predeterminada per crear una regla de validació.



Cal tenir en compte que no hi ha un nom d'informe predeterminat, per la qual cosa serà necessari escriure'n un; en cas contrari, el botó "Guardar" no s'activarà.

Un fitxer de regles accepta la declaració o execució de codi fora de la variable ***rules***, fet que és convenient si es volen crear altres funcionalitats que poden ser específiques només per un conjunt de regles. A la següent imatge es crea una variable local ***targetIfcTypes***, que serà utilitzada per cadascuna de les regles creades al mateix fitxer, i la funció ***highlight***, que com es pot veure ha estat afegida dins de la regla.

Reglas:

```
1  const targetIfcTypes = ["IfcSpace", "IfcBuildingElementProxy"];
2
3  function highlight(object) {
4    bimrocket.updateAppearance(($) => $("name") === "IfcRepresentation", {…});
21
22   bimrocket.updateAppearance(null, {…});
34
35   bimrocket.useTool("center_selection");
36 }
37
38 let rules = [
39   {
40     code: "Matèries primeres",
41     severity: "error",
42     selectExpression: ($) => $("type") == "Object3D",
43     checkExpression: ($) => {…},
55     highlight: () => highlight(),
56     message: (object) => {…},
62     summary: (objects) => `S'han trobat ${objects.length} objectess`,
63   },
```

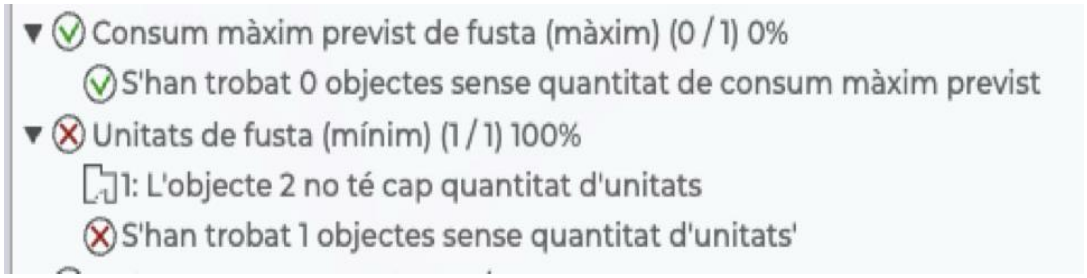
En un mateix fitxer es poden crear tantes regles com es vulgui, però totes han de seguir, com a mínim, la següent estructura:

1. Un codi identificador de la regla que mostra a l'inspector el nom de la regla individual executada.
2. Un grau de severitat que mostra una icona diferent segons la severitat indicada. Les possibles opcions son ***info***, ***warn*** i ***error***, però si tots els elements compleixen la regla apareixerà un tic en verd.
3. Una expressió de selecció que determina sobre quins elements del model s'aplica la regla.
4. Una expressió que contindrà tota la lògica executada que determina si un element passa la regla de validació o no.
5. Un missatge personalitzat per mostrar en cas que la regla es compleixi sobre un o més elements.
6. Un missatge de resum personalitzat que apareixerà tant si els elements compleixen les regles com si no.

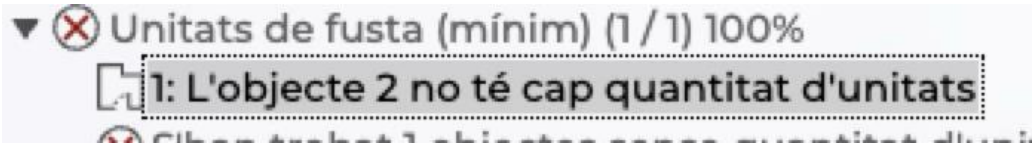
Amb les regles ja definides, l'usuari pot clicar el botó **executar** per fer la validació automàtica amb el fitxer de regles que tingui seleccionat. Això generarà un **report (l'informe) de validació** que resumeix l'estat de compliment de cada regla visible a l'inspector de la part inferior esquerre.

Si agafem com a exemple els elements que siguin de fusta, no seria lògic que hi hagués un element que utilitzi fusta, però no tingués cap unitat, per això, si les unitats es configuren a 0, la regla ens indicarà mitjançant un missatge personalitat que s'han trobat elements de fusta on no hi ha unitats.





En aquest cas com s'ha creat la funció ***highlight*** prèviament i la regla la fa servir, fent clic al missatge ens portarà directament l'objecte que no passa la validació.



També s'ofereix la funcionalitat de programar **scripts** més avançats per validar alguns aspectes concrets o que no es poden crear fàcilment mitjançant regles. Aquests programes es troben localitzats a la pestanya ***diseño > programas***, o bé fent clic directament a la seva icona de la barra d'eines (marcada amb color vermell en la següent imatge).



Gestió Incidències BCF

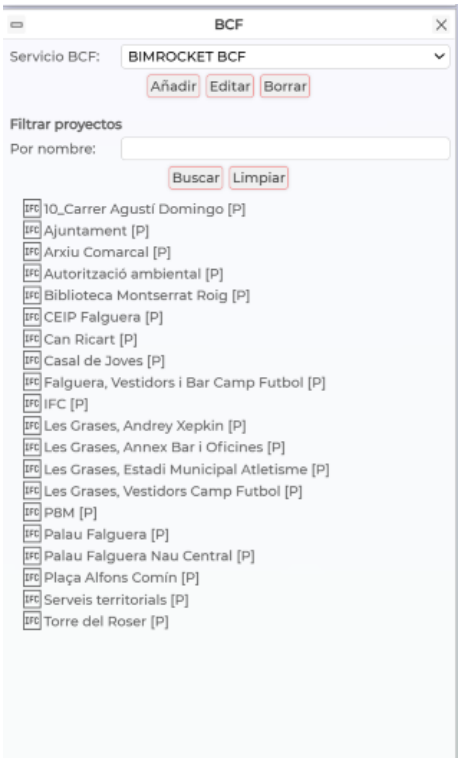
Per tal de permetre l'intercanvi de missatges entre diferents usuaris sobre un mateix model, Bimrocket integra un sistema BCF per a la gestió d'incidències. Aquest sistema s'implementa com a serveis de Backend. la BCF API es descriu aquí:

- bim.santfeliu.cat/bimrocket-server/swagger
- Es poden crear diferents tipus d'element BCF. El més important per al nostre cas és el d'Incidència.
- Les **incidències** es creen **vinculades al posicionament de la càmera** en el moment d'enfocar un element del model. Per tant, indirectament estan l**ligades als elements** del model que es troben en aquella posició de la vista de càmera. Per crear una incidència, s'ha de seleccionar un element en el model (internament es posiciona la càmera en la zona on es troba aquell element).
- Es poden **adjuntar captures del que s'està visualitzant**. Així sempre tens la imatge de com era l'element "congelat en un moment determinat del temps", cosa que és molt útil quan s'utilitza el sistema BCF sobre diferents versions d'un model, en les que potser s'ha substituït elements antics per uns de nous en la mateixa posició del model.
- Es poden **filtrar Incidències per tipus, Estat, Prioritat o per Assignació**.
- Es poden **adjuntar comentaris** quan es crea, edita o visualitza una incidència.
- També es poden **adjuntar documents via URL** (no locals).
- Finalment, es poden **definir nous "tipus"** d'incidència.
    - Per exemple, es pretén crear una incidència tipus "PARÀMETRE FINAL" que permet definir, en format de text lliure, valors màxims, mínims, recomanats, etc d'algunes característiques dels elements que no apareguessin com propietats dels elements, així com algunes consideracions que es vulguin deixar constància per part de les entitats validadores.

Projectes BCF

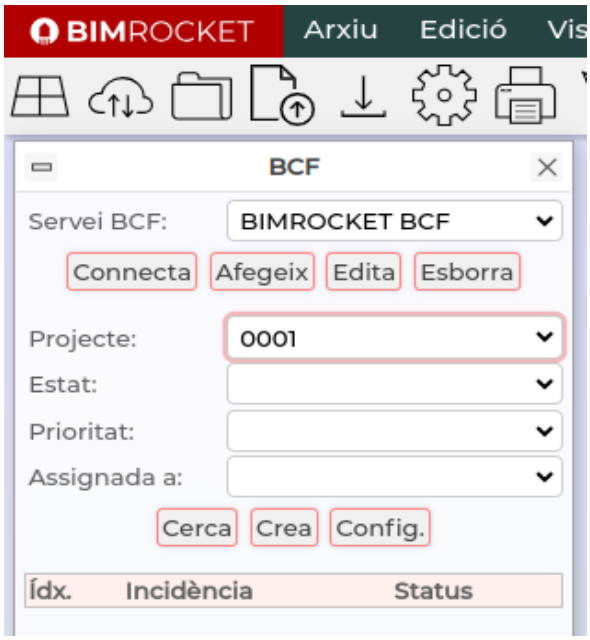
Per crear una incidència en un model, primer cal que tingui un projecte BCF assignat.
Si tenim carregat un model al visor, sigui des del remot o des de l'ordinador, en inicialitzar l'eina BCF, l'usuari veurà els projectes BCF disponibles prèviament creats (es veuran amb una "P" al costat que indica que ja s'han persistit) i a més el nou projecte BCF del model que acaba d'obrir (que en aquest cas apareixerà amb una "V" al costat indicant que està carregat al visor. Si el projecte carregat actualment al visor ja s'havia desat com a projecte BCF, llavors apareixen tant "P" com "V" al costat del nom del projecte.

El llistat de projectes s'emmagatzemen a l'API i, per tant, sempre estan disponibles hi hagi o no un model al visor.



Tenint en compte el que acabem d'explicar, per crear i desar un projecte BCF els passos a dur a terme són els següents:

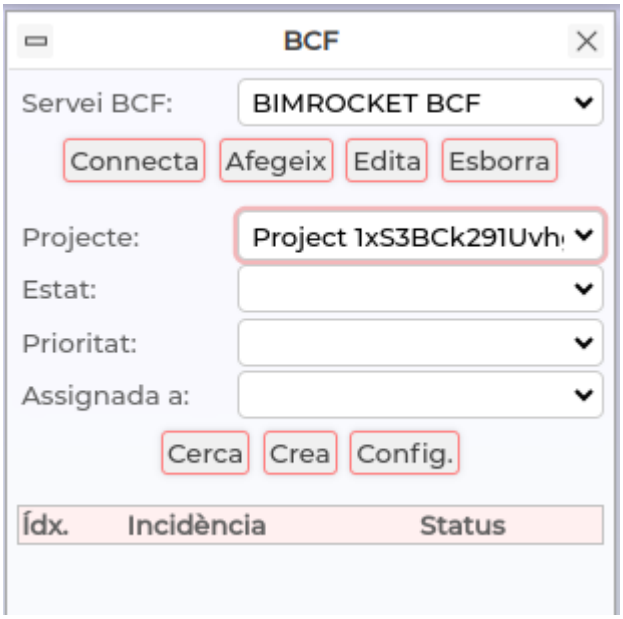
1. carregar un model al visor
2. A partir d'aquí ens dirigim a l'eina BCF, ens connectem i veurem que al desplegar el selector "Projecte" s'ha afegit un nou ítem amb el nom del model i al costat apareix una "V" majúscula, indicant que està carregat actualment al visor. Per defecte no es selecciona automàticament, ni tampoc s'emmagatzema el projecte BCF a la base de dades.



3. Caldrà seleccionar-lo i entrar al menú de “config”. En aquest menú podrem crear el nou projecte clicant a “crear” que surt en la part superior del quadre de text, o bé aprofitar per personalitzar les propietats BCF d’aquest projecte i clicar en el “config” de la part inferior del quadre de text. Amb aquesta segona opció, el projecte ja es crea amb les propietats setejades. A partir d’aquest moment s’emmagatzema el projecte BCF a la base dades amb l’IFCProject del model obert al visor d’aquesta manera:


```
{
"project_id": "1xS3BCk291UvhgP2a6eflL",
"name": "Project 1xS3BCk291UvhgP2a6eflL"
}
```


4. El selector de Projecte del BCF mostra finalment el valor de “name” del projecte, així com també la lletra “P” al costat, indicant que s’ha persistit a DB:



Important: els projectes BCF sempre estan disponibles encara que hi hagi un model obert al visor o no (aquests projectes apareixen sense la “V” al costat, per tant es poden crear incidències sense tenir un model al visor.

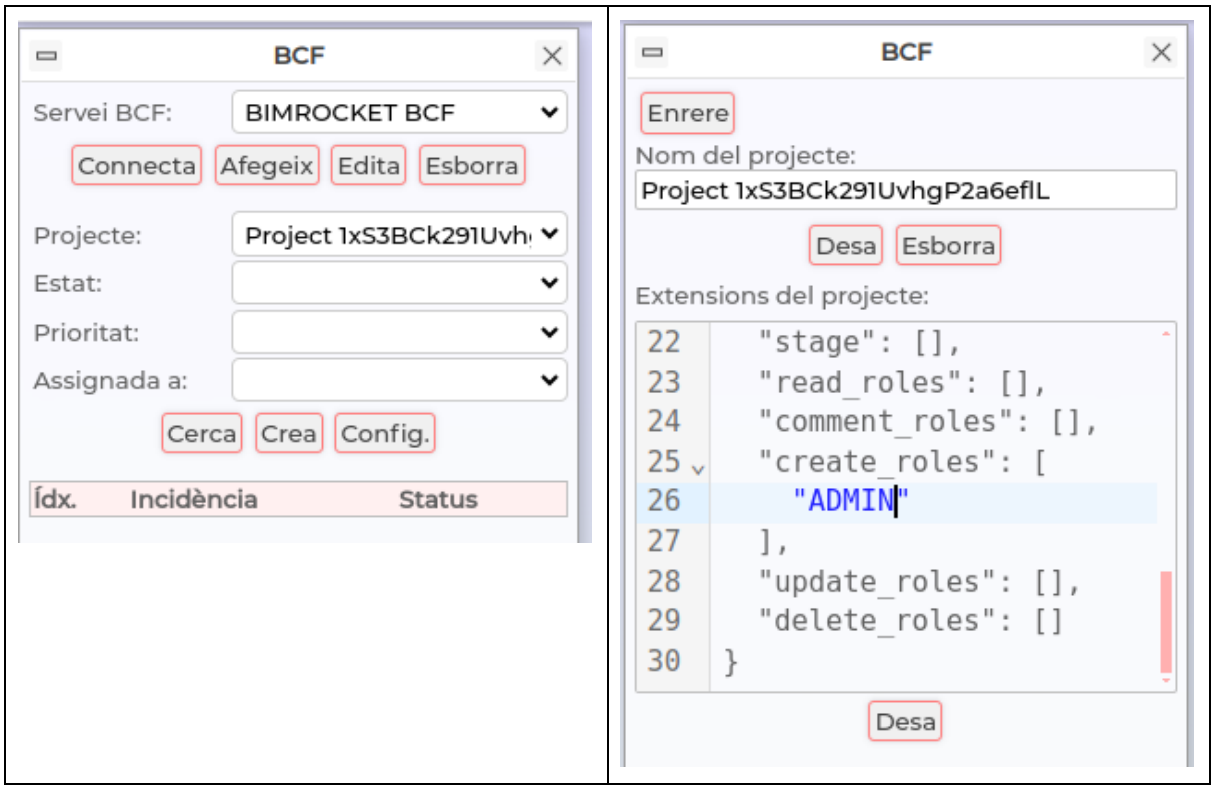
Extensions BCF per configurar permisos de visualització i edició en funció de rols

Es poden assignar permisos a totes les accions que es poden fer per projecte BCF:

- per llegir incidències (read_roles)
- per deixar comentaris (comment_roles)
- per crear incidències (create_roles)
- per actualitzar-les (update_roles)
- per eliminar-les (delete_roles)

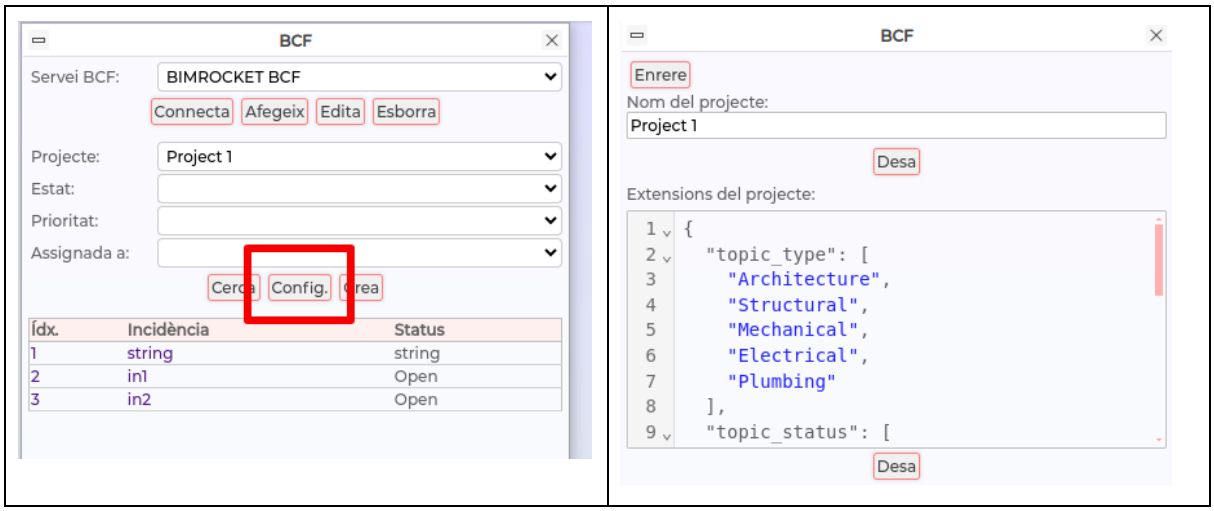
Cada projecte BCF pot tenir una configuració de rols diferent que afecta al seu projecte. Un exemple: en el projecte 1, el rol ADMIN pot crear incidències i el rol i2CAT pot eliminar incidències, i en el projecte 2, es defineix que el rol i2CAT pot actualitzar incidències.
Si hi ha permisos definits a la configuració, Bimrocket demana l'autenticació de l'usuari per permetre fer-li les accions que tenen rols assignats. En l'exemple anterior, l'usuari amb rol i2CAT vol crear una incidència en el projecte 1, en el moment de clicar l'acció "Crea", li demanarà autenticació i no tindrà permisos per continuar l'acció.
Si no hi ha permisos definits amb rols a la configuració el formulari d'autenticació no es mostra i, per tant, no té l'acció restringida.

Per definir rols es pot fer des de l'aplicació, seleccionant el projecte BCF del llistat i el botó "Config." el qual obrirà un element on s'hauran d'especificar els rols que tenen permisos per fer les accions del BCF:



Tipus d'incidència i altres extensions

Els tipus d'incidència es defineixen dintre de la configuració de l'extensió "*:topic_type*".
Per veure o modificar la configuració per defecte, cal accedir a la configuració del propi projecte dins del panell BCF:



Dins del camp de text "editor de codi" hi ha les extensions del projecte que podrem editar, afegir i eliminar, tots els paràmetres que introduïm són cadenes de text "strings". Tots aquests paràmetres es poden recuperar amb l'endpoint:

GET
bimrocket-server/api/bcf/2.1/projects/1/extensions

- topic_type: el tipus d'incidència, aquí podem afegir, editar i eliminar un altre tipus.
- topic_status: l'estat de la incidència, aquí podem afegir, editar i eliminar un altre estat.
- snippet_type: no apareix cap camp al panell BCF, no obstant es podem afegir paràmetres aquí.
- priority: la prioritat, aquí podem afegir, editar i eliminar una altra
- user_id_type: aquí afegim els usuaris a qui podem assignar l'incidència.
- stage: la fase de l'incidència, aquí podem afegir, editar i eliminar una altra.

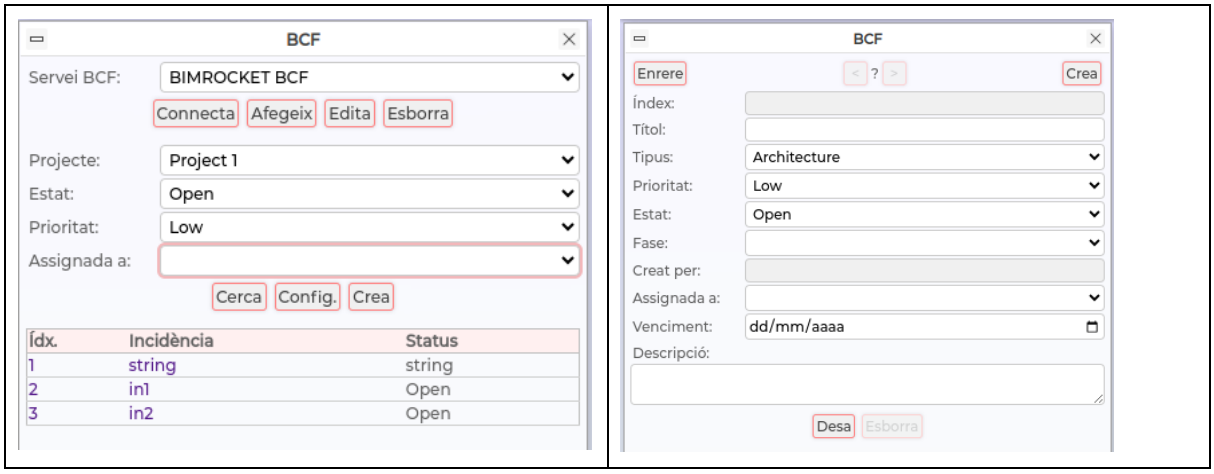

```
{
 "topic_type": [
  "Architecture",
  "Structural",
  "Mechanical",
  "Electrical",
  "Plumbing"
 ],
 "topic_status": [
  "Open",
  "Reopen",
  "Closed"
 ],
 "topic_label": [],
```

```
 "snippet_type": [],
 "priority": [
  "Low",
  "Medium",
  "High"
 ],
 "user_id_type": [],
 "stage": []
}
```


Quan tenim la configuració tal com la volem la podem desar, l'acció "desa" apunta cap a l'endpoint:

PUT
bimrocket-server/api/bcf/2.1/projects/1/extensions

Els paràmetres que posem en la configuració els veurem en la creació d'una incidència i per seleccionar les incidències:



Template BCF

Es pot crear un únic projecte Template amb la possibilitat de poder-lo utilitzar com a plantilla per a la creació dels projectes que es creïn posteriorment. D'aquesta manera, quan es crea un nou projecte BCF, en cas d'haver definit i Creat un projecte Template, el nou projecte que es crea incorpora per defecte les extensions definides en el projecte Template. Això ens permetrà automatitzar molt la creació de nous projectes bcf.
Per a definir un projecte com Template, primer cal definir el nom del template en el fitxer de configuració del backend (bimrocket-server.yaml). La propietat que defineix el nom del projecte Template és la de services.bcf.templateProjectName. Per defecte seria "template" (en el codi comunitat) i "BCFTemplate" en el cas del Pilot. Perquè el sistema detecti el template, el valor de la propietat ha de coincidir amb **NOM** (no id) amb que el projecte bcf ha estat desat en la DB.

Per tant, una vegada s'hagi definit el nom que rep el projecte Template dintre del fitxer de propietats, caldrà editar en un projecte que agafem com a exemple (no és important el contingut del model a excepció de la línia IFCPROJECT.
Per a editar el fitxer podem utilitzar un editor de text, buscar la propietat IFCPROJECT i modificar el primer i el tercer valor de la llista d'aquesta propietat.
En el següent exemple, es pot veure que s'han editat la 1a i la 3a posició dels arguments de la línia IFCPROJECT amb nom que li hem posat al valor de la propietat del services.bcf.templateProjectName del bimrocket-server.yaml.

```
#25=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Box','Model',*,*,*,*,#22,$,.MODEL_VIEW.,$);
#26=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('FootPrint','Model',*,*,*,*,#22,$,.MODEL_VIEW.,$);
#27=IFCPROJECT('BCFTemplate',#18,'BCFTemplate',$,$,'','',(#22),#294522);
#28=IFCPROPERTYSINGLEVALUE('01-01-ACATOGE_Antecedents',$,IFCTEXT('Aquesta planta t\X\E9 una auto
#29=IFCPROPERTYSINGLEVALUE('01-02-ACATOGE_Objecte',$,IFCTEXT('Implantaci\X\F3 d''una activitat d
```

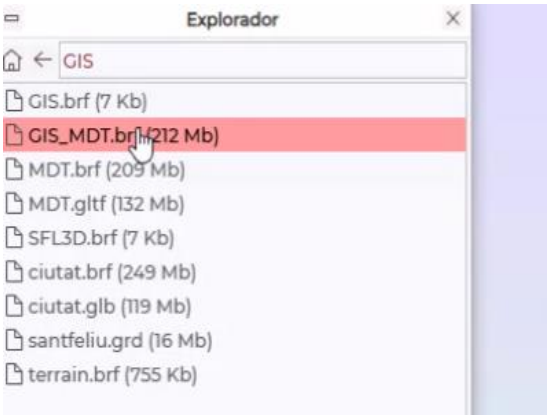
Tot seguit s'haurà de carregar al visor el model editat que volem utilitzar com a Project Template, Obrir la Tool de BCF i entrant com ADMIN, seleccionar el nou projecte al desplegable de projectes del BCF, clic al botó config.
En aquest panell hem de assegurar-nos que el nom es l'escollit en el .yaml, (es probable que haguem de eliminar la paraula "Project " del nom. Seguidament cliquem botó "Save" que apareix just a sota del nom.

Seguidament, podem editar les extensions amb l'altre botó "Save" que apareix sota el Text Area d'Extensions.
A partir d'aquest moment sempre que es creï un nou projecte BCF, l'aplicació automàticament copiarà les extensions del projecte Template (encara que abans de desar el nou projecte sempre es permet a l'usuari modificar-les).

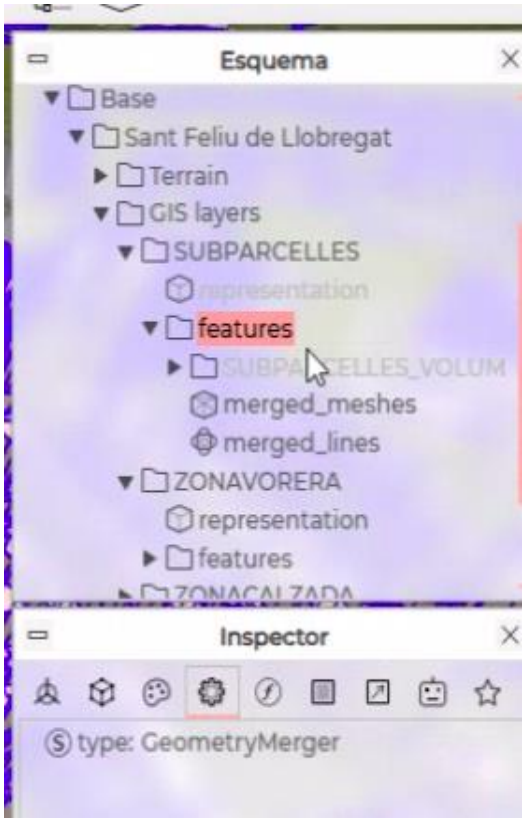


Integració amb GIS

Gràcies a la **compatibilitat WFS** Bimrocket es pot integrar amb capes GIS d'un servidor WFS. Això permet a l'usuari poder carregar capes GIS sobre el model per poder, per exemple, mesurar distàncies dels elements del model, o de les parcel·les respecte elements del terreny que puguin aparèixer dintre de la informació de la capa.



- Mirar model a PUBLIC>GIS_MDT.brf, perquè treu les capes a temps real.
- Mirar amb quines credencials accedir a un mapa de Catalunya.
- S'ha de publicar el teu GeoJson.
- la informació externa de capes s'ajunta amb la informació del model.
- Mirar Esquema:



Millores i Adaptacions realitzades en el present projecte

Dockerització (Comunitat i Pilot)

S'han realitzat les adaptacions necessàries per a desplegar l'eina via contenidors. L'explicació del desplegament a Integració/producció ja té en compte les adaptacions fetes per tal d'aconseguir-ho.

Modificació de sistema Autenticació (Comunitat i Pilot)

El sistema d'autenticació ha estat modificat pel client incorporant l'opció d'utilitzar LDAP o bé la base de dades. En cas de la base de dades, s'han incorporat a l'API una sèrie de endpoints per poder mantenir usuaris i rols. Els usuaris poden tenir el rol d'ADMIN, PROJECTISTA o VECTOR-UT-OGE. Segons quins endpoints necessitaran tenir privilegis d'administrador.

Per tal de poder crear usuaris i rols, en el fitxer de projecte application.yaml dins de bimrocket-server tenim les credencials de l'administrador per autenticar-nos a l'API. Un cop autenticats, podem donar mantenir els usuaris i rols amb els següents endpoints que ofereix l'API:



Com a exemple, per crear un nou usuari:

Important: El password ha de tenir mínim 8 caràcters, mínim una majúscula i mínim un número

Dins de l'apartat Security



Si volem crear un usuari ADMIN:


```
{
  "active": true,
  "creation_date": "string",
  "email": "string",
  "id": "jordi",
  "modify_date": "string",
  "name": "Jordi",
  "password": "Provauser1#",
  "password_hash": "string",
  "roles": [
    "ADMIN"
  ]
}
```


Si volem crear un usuari PROJECTISTA:


```
{
  "active": true,
  "creation_date": "string",
  "email": "string",
  "id": "jordi2",
  "modify_date": "string",
  "name": "Jordi",
  "password": "Provauser1#",
  "password_hash": "string",
  "roles": [
    "PROJECTISTA"
  ]
}
```

Passar fitxer de propietats de Back-end (Comunitat i Pilot)

S'ha implementat un mecanisme per poder customitzar de manera elegant les propietats existents del web.xml en el desplegament a producció, sense tenir necessitat d'accedir al codi font del projecte i sense haver de compilar de nou el .war de backend
En el desplegament del backend, en el directori d'usuari es crea un directori **bimrocket** amb el fitxer **bimrocket-server.yaml.** Aquest fitxer és una còpia del fitxer de projecte application.yaml i conté les propietats.
A més, es defineix una variable d'entorn, BIMROCKET_DATA_PATH que ens indica on queda ubicat el directori bimrocket dins del contenidor. Podem personalitzar aquesta ruta amb un altra ubicació, però caldrà tenir en compte que si canviem la ruta de la variable també hem de canviar la ruta interna del volum corresponent a bimrocket-server.yml que ha de ser la mateixa que el valor de la variable. Com a exemple:

BIMROCKET_DATA_PATH=**/usr/local**

Volum:
application.yaml → **/usr/local**/bimrocket-server.yaml

En el nostre cas, hem creat una imatge de docker amb el war incorporat.
En el moment de desplegar amb docker-compose agafem aquesta imatge i creem un volum:

- Un volum per sincronitzar un fitxer application.yaml extern (es una copia d'application.yaml de l'aplicació) i el fitxer ubicat al directori d'usuari amb les propietats

application-yaml -> /home/user/bimrocket(bimrocket-server.yaml

Un cop desplegat el contenidor qualsevol canvi que fem en el fitxer de host application.yaml es veurà reflectit al fitxer intern de l'usuari després de reiniciar el contenidor.
En els apartats de Tool de Gestió de Rols i Tool de Gestió d'usuaris s'expliquen les eines que s'han implementat en la aplicació Web que fan us d'aquesta API d'usuaris.

Passar fitxer de propietats de front-end com a environment (Comunitat i Pilot)

En el codi original s'ha fet una sèrie de canvis per part del client que mitjançant variables d'entorn es pugui localitzar un fitxer de propietats extern.

El funcionament es basa en una "cascada de cerca" que realitza el servidor Java (EnvironmentServlet). Quan l'aplicació carrega al navegador, sol·licita /js/Environment.js. El servidor decideix quin fitxer lliurar seguint aquest ordre:

1. **Variable d'entorn:** Valor definit a BIMROCKET_ENV.
2. **Propietat JVM:** Paràmetre -DBIMROCKET_ENV.
3. **Configuració Context:** Paràmetre init-param al web.xml.
4. **Home de l'usuari:** Fitxer a ~/Environment.js.
5. **Intern:** El fitxer per defecte dins del .war.

S'ha creat una variable d'entorn en el docker-compose del frontend:

```
environment:
  - BIMROCKET_ENV=/usr/local/tomcat/webapps/bimrocket/js/Environment.js
```

En el docker-compose.yml, la variable d'entorn serveix com a "punter". El codi Java (Servlet) llegeix aquesta variable per saber on ha d'anar a buscar el fitxer de configuració:

- **Nom de la variable:** BIMROCKET_ENV
- **Valor:** La ruta absoluta dins del contenidor (ex: /usr/local/tomcat/webapps/bimrocket/js/Environment.js).

Al recollir la variable d'entorn, estem accedint als fitxers de propietats que són utilitzades pels diferents mòduls.

El fitxer Environment.js inclou:

Connexió al Servidor

- SERVER_URL: Defineix on és l'API (p. ex., http://localhost:9090/bimrocket-server per a pilots).
- MODULES: Llista de connectors. Permet carregar scripts de propietats personalitzades des d'URLs externes.

Gestió d'Autenticació (AUTH_ENVIRONMENT)

Aquesta és la part més crítica. Permet que BIMROCKET sàpiga quins botons de login mostrar segons l'URL que l'usuari té al navegador:

- **Domini Local/Pilot:** Configurat per a proves amb localhost.
- **Domini Producció (Sant Feliu):** Configurat amb els clientId i redirectUri oficials de la AOC (VÀLid) o GICAR.

Crear un Mòdul custom que reuneixi eines essencials dels 3 mòduls de la comunitat. (Pilot)

Per al projecte Pilot s'ha desenvolupat un nou mòdul personalitzat anomenat "custom", amb l'objectiu d'unificar en un sol espai totes les eines essencials que fins ara estaven repartides en tres mòduls independents: base, bim i gis.

L'objectiu principal d'aquest nou mòdul és controlar i adaptar quines eines es mostren i estan disponibles per als usuaris que accedeixen a l'aplicació, simplificant-ne la gestió i configurant un punt centralitzat de funcionalitats.

Cada mòdul ofereix les seves eines a la barra d'eines; per tant, en crear un mòdul cal especificar quines eines es volen incloure, així com configurar els diferents paràmetres corresponents a cadascuna, les connexions amb els serveis de backend, etc.
Per tal que el mòdul personalitzat custom es carregui automàticament al visor, sense haver-lo de passar com a paràmetre a l'URL, cal indicar-ho al fitxer d'entorn Environment.js, dins de

la propietat que especifica quins mòduls s'han de carregar. Els mòduls els guardem a la carpeta modules que es troba a src/main/webapp/js/modules. De manera que en el fitxer d'entorn indiquem els mòduls que volem habilitar a la propietat MODULES:

```
export const Environment =
{
  MODULES : ["/bimrocket-server/api/cloudfs/scripts/default_properties.js", "custom", "plus"]
};
```

Una altra manera de crear i carregar un mòdul custom (també esmentada breument més amunt a l'apartat Funcionalitats destacades i com fer-les servir) és accedint-hi directament mitjançant l'URL, passant el mòdul creat com a paràmetre.
Aquest mòdul es pot crear des del front-end i guardar-lo dins la carpeta scripts del directori cloudfs. Per fer-ho, es pot utilitzar l'eina Programes (Scripts), on l'editor permet implementar el mòdul personalitzat i desar-lo al sistema.
Per carregar aquest mòdul desat al cloudfs des d'una URL, només cal afegir-lo com a paràmetre a l'adreça següent:

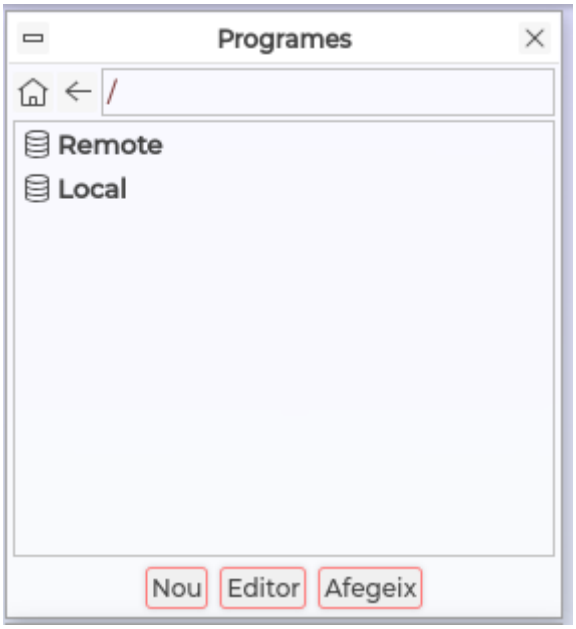
```
http://<HOST>/src/main/webapp/app.html?modules=<HOST>/bimrocket-server/api/cloudfs/scripts/basic_module.js
```

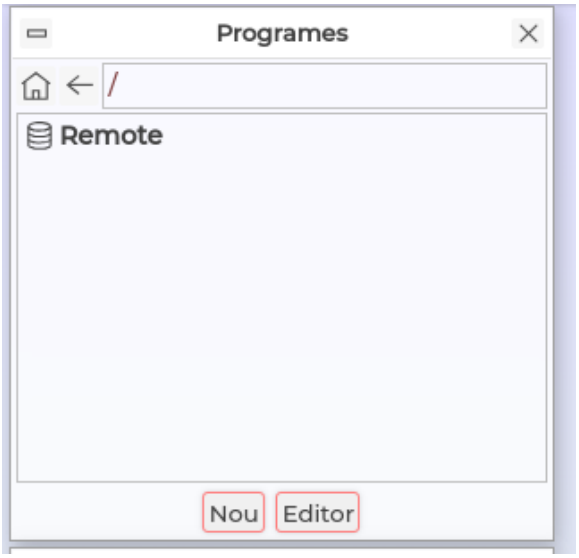
A l'entorn de producció l'URL seria:

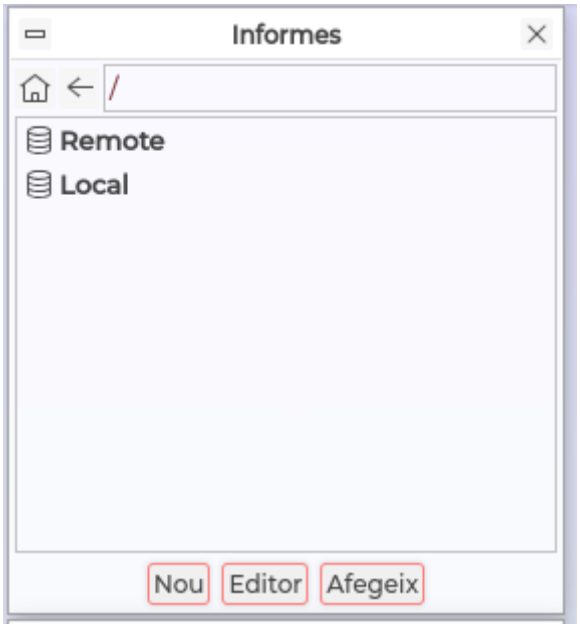
```
https://bim.santfeliu.cat/bimrocket/app.html?modules=/bimrocket-server/api/cloudfs/scripts/modules/basic_module.js
```

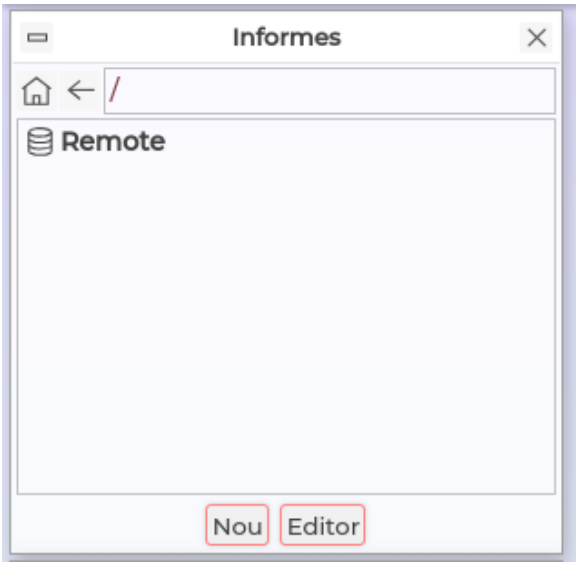
Nota: no es pot utilitzar l'URL relativa perquè només està a Environment.js per models i no per reports.
l'URL on es troba emmagatzemat aquest nou mòdul, que tindríem a SERVER_URL/cloudfs/scripts/public/ +<nom del modul>.js .

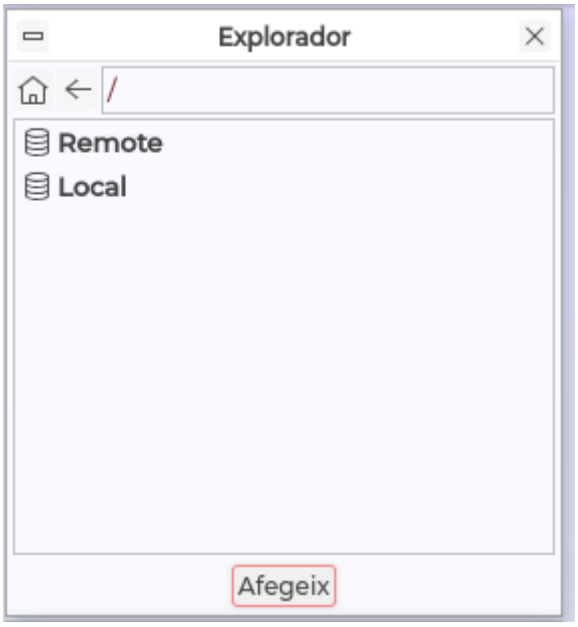
Bimrocket comunitat	Custom (Pilot)


Programes
/
Remote
Local
Nou
Editor
Afegeix


Programes
/
Remote
Nou
Editor

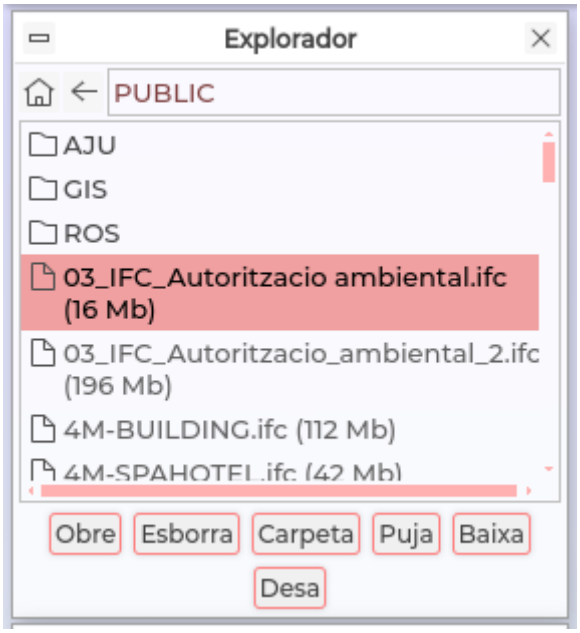

Informes
/
Remote
Local
Nou
Editor
Afegeix


Informes
/
Remote
Nou
Editor





Botons Explorador





Configurar dinàmicament al nou mòdul “custom” de frontend l'URL on es serveixen els serveis de backend. (Pilot)

Gràcies a les funcionalitats anteriors 3 i 4, ja podem carregar dinàmicament propietats definides en un fitxer d’entorn. D’aquesta manera, en el mòdul custom que hem creat, es defineixen les URLs dels serveis backend de manera dinàmica, com a concatenacions de propietats definides en el fitxer d’entorn (backend-url, etc)

Gràcies a les funcionalitats esmentades anteriorment, ara podem carregar dinàmicament les propietats definides en el fitxer d'entorn **Environment.js**. Això permet que, en el mòdul personalitzat que hem creat, es defineixin les URLs dels serveis backend de manera dinàmica, mitjançant la concatenació de les propietats especificades en el fitxer d'entorn (com ara backend-url, entre d'altres).

```
if (application.services.model === undefined)
 {
  const webdav = new WebdavService({
   name: "models",
   description : "Remote",
   url : (Environment.SERVER_URL || "/bimrocket-server") + "/api/cloudfs/models"
  });
 }
```

Customitzacions de UI

Moure funcionalitats de disseny a un mòdul “plus”

Algunes de les funcionalitats avançades de bimrocket, així com les més orientades cap al disseny de models, s’han mogut a un nou modul Plus, que per defecte no s’inicialitza en el Pilot, ja que podria desfocalitzar els usuaris que volen utilitzar bimrocket amb l’objectiu de validar el model i de agilitzar la comunicació entre projectista-validador.
De totes maneres, el mòdul “plus” es pot incloure en la configuració dels mòduls tal com s’ha explicat anteriorment, sempre i quan també s’inclogui el modul “custom”.

Canviar icona cloud per “baixar de cloud” i eliminar icones per a operacions no permeses al Pilot (Pilot)

S’han eliminat de l’explorador de fitxers cloud les opcions (botons) de pujar, desar i descarregar a local ja que per al present Pilot es restringeix que només es puguin pujar al cloud els models via previa-validació i per tant, mitjançant la funcionalitat de Validar i Desar. Tampoc es permet la opció de descarregar per evitar que es puguin obtenir els ifc del Cloud, permetent només la interacció “per lectura” amb aquests.
D’aquesta manera tenia molt més sentit canviar la icona de cloud per un icona representatiu per “carregar fitxers del cloud”.

Funcionalitat de canvi de permisos de fitxers i carpetes del repositori models (Comunicat i Pilot)

La funcionalitat dels permisos en els arxius i les carpetes allotjats en servidors remots és establir quins usuaris hi poden accedir i modificar-los, sense necessitat d'accés directe al servidor. Per defecte, l'usuari creador de l'arxiu en tindrà el control total, així com els usuaris que tenen el rol ADMIN.

L'accés a la funcionalitat és el següent:

- Accedeix a l'eina Explorador d'arxius.
- Seleccioneu el servidor "Remote" al panell lateral.
- Exploreu l'estructura de directoris, cada directori permet la modificació de permisos.
- Feu clic a "Edita ACL"
- Entreu les credencials de l'usuari.



En fer clic a «Edita ACL», apareix una finestra emergent que mostra una llista dels rols d'usuari i el tipus de permisos permesos per a cadascun. Hi ha l'opció de poder-los modificar, afegir o eliminar privilegis a un tipus de rol.
Per tal d'implementar aquesta funcionalitat que hem mostrat en el front-end, ha calgut la implementació de les següents tasques:

1. **Mètode ACL (setejar) (interfície WebDav de Backend)**

El mètode doAcl de la classe WebdavServlet necessita rebre des de la request un path i un Webdav XML amb els permisos definits. Un exemple Acl per a la carpeta autorització ambiental seria:

```
<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:">

 <!-- Access for ADMIN in .acl -->
 <D:ace>
   <D:principal>
     <D:href>ADMIN</D:href>
   </D:principal>
   <D:grant>
     <D:privilege><D:read-acl/></D:privilege>
     <D:privilege><D:write-acl/></D:privilege>
   </D:grant>
 </D:ace>

 <!-- Access for PROJECTISTA in .acl -->
 <D:ace>
   <D:principal>
     <D:href>PROJECTISTA</D:href>
   </D:principal>
   <D:grant>
     <D:privilege><D:read/></D:privilege>
     <D:privilege><D:write/></D:privilege>
   </D:grant>
 </D:ace>

 <!-- Access for VECTOR-UT-OGE in .acl -->
 <D:ace>
   <D:principal>
     <D:href>VECTOR-UT-OGE</D:href>
   </D:principal>
   <D:grant>
     <D:privilege><D:read/></D:privilege>
     <D:privilege><D:write/></D:privilege>
   </D:grant>
 </D:ace>

 <D:ace>
   <D:principal>
     <D:all/>
   </D:principal>
   <D:grant>
     <D:privilege><D:read/></D:privilege>
   </D:grant>
 </D:ace>

</D:acl>
```

El fitxer resultant ha de tenir el format JSON:

```
{
  "*" : {
    "READ_ACL" : [ "ADMIN" ],
    "READ" : [ "PROJECTISTA", “VECTOR-UT-OGE”, "EVERYONE" ],
    "WRITE" : [ "PROJECTISTA" ],
    "WRITE_ACL" : [ "ADMIN" ]
  }
}
```

Per tant, es necessita una conversió prèvia de XML a MutableACL abans de fer la crida al fileService:

```
// i2CAT: Call to deserialize XML
MutableACL acl = MutableACLXMLDeserializer.deserialize(requestXmlData,
user.getId());

fileService.setACL(path, acl);
```

La nova classe MutableACLXMLDeserializer, des del mètode deserialize, s’encarrega de convertir el XML de la request a objecte MutableACL i el fileService crea el fitxer en el format JSON desitjat en el path indicat.
Veiem que quan fem la crida al MutableACLXMLdeserialize li passem com a paràmetre el id d'usuari que està en sessió en aquell moment. Això és útil perquè en el cas que ens trobem la etiqueta <D:owner/> quan llegim el XML, substituirem aquest valor per l’ id user quan anem a informar la propietat corresponent al rol de l’objecte MutableACL.

En el XML, els rols amb nom reservat per WEBDAV tals com **all** estan definits de la següent manera:

```
<D:principal>
  <D:all/>
</D:principal>
```

El Backend quan troba aquests valors els canvia per valors definits a SecurityConstants:

all --> EVERYONE

Pel que fa als rols definits a la BD com **ADMIN**, **PROJECTISTA** o **VECTOR-UT-OGE** o el **userid** es defineixen a l'xml com:

```
<D:principal>
  <D:href>ADMIN</D:href>
</D:principal>
...
<D:principal>
  <D:href>VECTOR-UT-OGE</D:href>
</D:principal>
...
<D:principal>
  <D:href>PROJECTISTA</D:href>
</D:principal>
...
<D:principal>
  <D:href>userid</D:href>
</D:principal>
```

El backend, si troba **href** agafa el valor de l'etiqueta directament. És important que aquests rols estiguin en majúscules

2. **Implementar un Mètode propfind de interfície Webdav per tal d'obtenir informació d'ACL**
3. **Implementar un WebDav Client en el front-end**

Extensions BCF per defecte des de fitxer de propietats (sempre i quan no hi hagi definit un Project Template) (Pilot)

S'ha incorporat un mecanisme que permet definir en el fitxer de propietats de backend (bimrocket-server.yaml) les extensions per defecte que s'injecten en la creació d'un projecte BFC sempre que no hi hagi definit cap Project Template. En cas d'haver-hi definit un Project Template, sempre es prioritza les extensions definides en aquest Template en comptes de les definides en el fitxer de propietats.

Aquesta funcionalitat només està inclosa en el Pilot i a l'apartat d'inicialització del Pilot es pot veure com es pot utilitzar concretament.

Customització de Filtres BCF (Comunitat i Pilot)

Aquesta funcionalitat controla quins camps per filtrar incidències es mostren a l'eina BCF Panel. Els camps que es poden inhabilitar són els següents:

- Tipus d'incidència
- Estat
- Prioritat
- Usuari assignat

La funció **isFilterFieldEnabled(fieldName)**, inclosa al codi de la Comunitat bimrocket, per defecte sempre retorna “True”. Per tant si no es sobreescriu la funcionalitat, sempre se mostren tots els camps de filtre. Però el que és interessant es que permet que els mòduls que inicialitzen un bcfTool puguin sobreescriure la lògica que implementa la funció.
D’aquesta manera al codi del Projecte Pilot s’ha sobreescrit la lògica de la següent manera:

```
bcfTool.panel.isFilterFieldEnabled = (name) => {
  return getCustomFilters("BCFPanel", role).includes(name);
};
```

Amb aquesta redefinició, només es consideren habilitats els camps que estan inclosos en la llista retornada per getCustomFilters(...), en funció del mòdul "BCFPanel" i el role actiu. Aquesta llista defineix els filtres que l’usuari té permís per utilitzar.

La funció **updateFilterControls()** amaga o mostra els camps de filtre (topic_type, topic_status, priority, assigned_to) segons la funció isFilterFieldEnabled(...).

```
const enableField = (fieldElem, fieldName) =>
{
  const enabled = this.isFilterFieldEnabled(fieldName);
  fieldElem.parentElement.style.display = enabled ? "" : "none";
};
```

Customització de botons del CloudExplorer (Pilot)

Aquesta funcionalitat permet controlar dinàmicament la visibilitat dels botons d'acció en el panell d’“Explorador d’arxius”, segons els permisos associats al rol de l'usuari.

La funció **isActionEnabled(name)** determina si una acció concreta està habilitada. Per defecte, en el codi de la Comunitat Bimrocket, la funció retorna sempre true. No obstant això, es redefineix posteriorment en el context de l’eina CloudExplorerTool (Explorador d’arxius).

```
cloudExplorerTool.panel.isActionEnabled = (name) =>
  getAllowedActions("FileExplorer", role).includes(name);
```

Aquesta redefinició (implementada dintre del mòdul Customitzat pel Pilot personalitza la lògica perquè només estiguin habilitades les accions que estan incloses en la llista retornada per getAllowedActions(...), segons el mòdul "FileExplorer" i el “role” de l'usuari.

Es defineix una funció **addContextButton(name, label, action, isVisible)**
que habilita els botons al mòdul "FileExplorer",

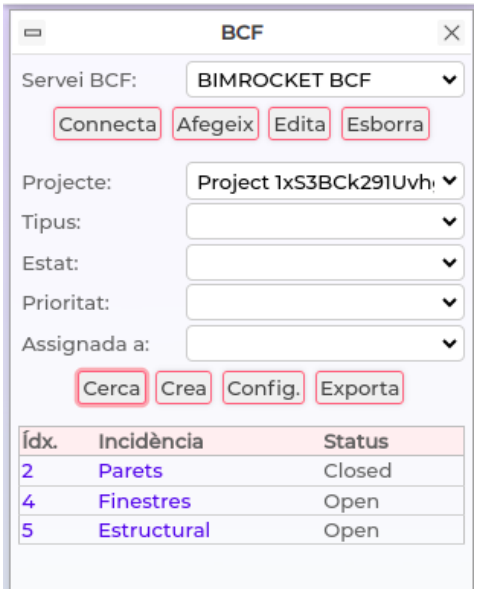
```
const buttonElem = Controls.addButton(this.buttonsPanelElem, name, label, action);
buttonElem._isVisible = () => this.isActionEnabled(name) && isVisible();
```

El botó només serà visible si les següents condicions són certes:

- Si l'acció està habilitada (isActionEnabled(name)), en funció dels permisos del rol de l'usuari.
- Si la condició externa isVisible() també retorna true.

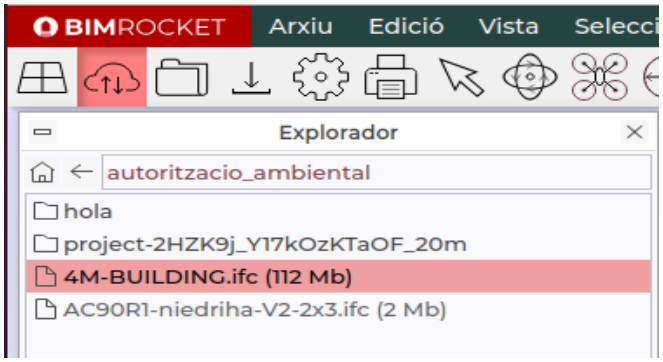
Exportació BCF a Excel

Aquesta funcionalitat permet a l'usuari exportar fàcilment totes les incidències d'un projecte, tant si estan obertes com si no.
Quan l'usuari consulta les incidències BCF d'un projecte, visualitza un llistat que pot tenir actius diversos filtres: per persona assignada, estat, tipus o prioritat.
L'exportació es basa en el llistat d'incidències visible en pantalla i respecta els filtres que l'usuari hagi aplicat, com ara: persona assignada, estat, tipus o prioritat.
En prémer el botó Exporta, es genera un fitxer .csv amb les dades de les incidències visibles al llistat actual. Cada fila representa una incidència i inclou les següents columnes: nom de la incidència, tipus, prioritat, estat, l'autor qui l'ha creat, a qui està assignada, data de finalització i descripció.



Bugfix: Error en descàrrega de fitxers ifc de repositori remot

Per tal d'evitar que es pugui descarregar un nou model mentre s'està descarregant un altre, s'ha optat per bloquejar la llista de models. Quan s'inicia la descàrrega d'un model es bloqueja la llista de models i es torna a activar o desbloquejar en el moment que s'ha completat la descàrrega.



Quan seleccionem un fitxer d'una grandària considerable i cliquem el botó de "Baixa" o download, aquest explorador queda bloquejat i no es pot seleccionar cap altre element de la llista.

Fitxer regles de Autorització Ambiental (Pilot)

El primer pas per a permetre un bon procés de validació es el de crear un fitxer de regles que inclogui diferents regles per a la comprovació dels paràmetres ambientals que s'han acordat incloure en aquest Pilot.
La última versió del fitxer de regles informe_autoritzacio_ambiental-v.0.2.brs està present en la carpeta "informes" del repositori de Data del codi de bimrocket.

Al principi del fitxer es defineix un diccionari ***(PSET_KEYS)*** que mapeja noms interns (de fácil comprensió) amb els noms reals dels **Property Sets** (conjunts de propietats) dins del fitxer IFC. Això permet que el codi sàpiga exactament on anar a buscar la informació de residus, atmosfera o ACA.

2. Funció d'Entorn Visual ()

La funció ***highlight*** realitza les següents accions:

- Fa que la resta del model sigui transparent i de color gris clar.
- Pinta l'objecte amb l'error de **color vermell** amb línies blaves d'un gruix determinat per ressaltar-lo.
- Centra la càmera en l'objecte afectat.

El fitxer conté una llista de comprovacions automàtiques dividides per categories. Cada regla (rule) segueix la mateixa lògica: selecciona uns objectes (selectExpression), mira si compleixen un requisit (checkExpression) i, si no, llança un error o avís.

Principals àrees de control:

- **Gestió del Model:** Comprova que només hi hagi un model carregat alhora.
- **Informació General:** Verifica que existeixin camps obligatoris com els antecedents, l'objecte del projecte, el nom de l'empresa i l'autor, i que cap d'aquests estigui buit.
- **Atmosferes (Emissions):** Valida dades tècniques de focus d'emissió (codis CAPCA, potència instal·lada, hores de funcionament). Controla que siguin valors numèrics positius i que no superin límits màxims (ex: 9999).
- **ACA (Agència Catalana de l'Aigua):** Revisa dades de consum anual d'aigua, destins d'abocament, matèries en suspensió i volums de recirculació.
- **Residus:** Comprova que els objectes tipificats com a residus tinguin la descripció, el codi LER, el procés de generació i la capacitat d'emmagatzematge correctament informats.
- **Instal·lacions (Canyeries i Brides):** Verifica dades físiques com la longitud de la canonada, l'elevació i els diàmetres/gruixos de les brides.

El codi no només mira si el camp existeix, sinó que també valida la **qualitat de la dada**:

1. **Existència:** El camp hi és?
2. **Contingut:** Està buit?
3. **Tipus de dada:** És un nombre natural?

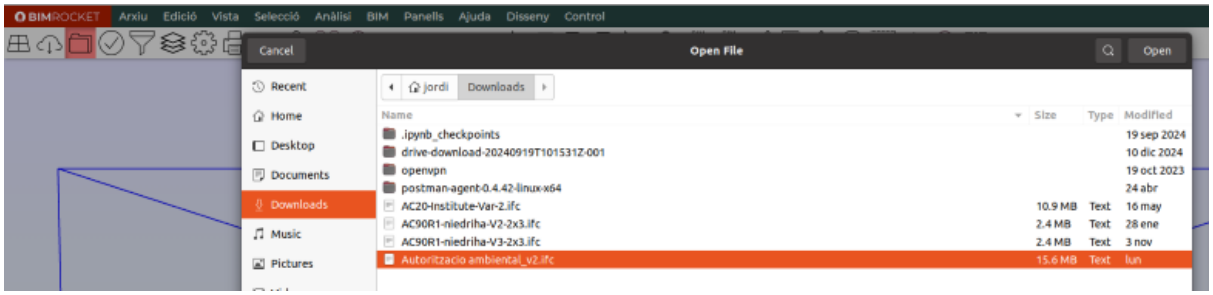
Límits: És un valor incoherentment alt (prop de 9999)? En aquest cas, llança un warn (avís) o un error.

Funcionalitat Validar i Desar (Pilot)

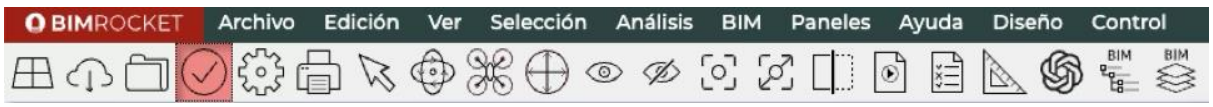
S'ha implementat una eina per tal de executar la validació del model i permetre la posterior pujada d'aquest en cas que la validació s'hagi finalitzat amb èxit.

La funcionalitat està gestionada pel mòdul UploadModelPanel.js. Mitjançant aquesta funcionalitat s'assegura el control de qualitat que assegura la integritat de les dades BIM que es demanen en els models abans del seu desat al servidor. Aquest mòdul actua com un orquestrador entre l'escena 3D del navegador, els motors de regles normatives i els processos de desat de dades del backend (Task Data i File System).

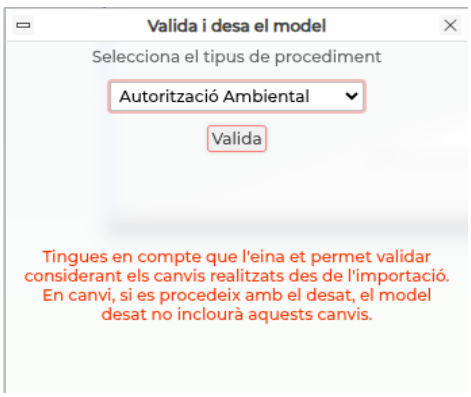
El primer que cal fer per a validar un model amb aquesta eina serà obrir aquest model IFC (normalment des de local, ja que l'enfocament de l'eina està orientat a poder desar aquest model al repositori de models, i per tant no té molt sentit validar un model que ja ha estat desat al repositori. Tot i així també es pot repetir la validació i desar una nova versió, encara que no presentarà cap diferencia respecte la versió que hem obert anteriorment, ja que la versió del Pilot que utilitzem de Bimrocket no permet fer canvis al model.
A la següent imatge es pot veure com es pot carregar un model des de l'ordinador local:



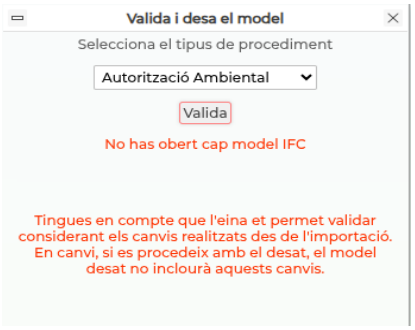
Un cop carregat el model ifc al visor, l'eina per validar el model es troba localitzada fent clic sobre la seva icona a la barra d'eines (marcada amb un color vermell a la següent imatge).



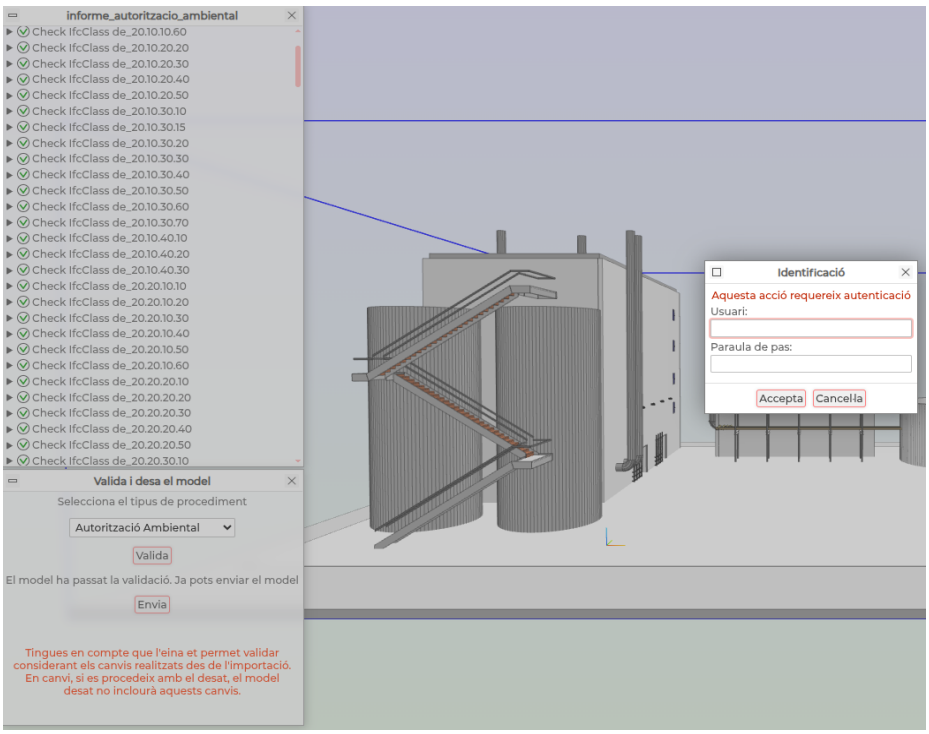
Un cop es fa clic a la seva icona, s'obre una finestra a la cantonada inferior esquerra que inicialment permet escollir un model des d'un desplegable, i just al costat del desplegable es troba el botó "valida".



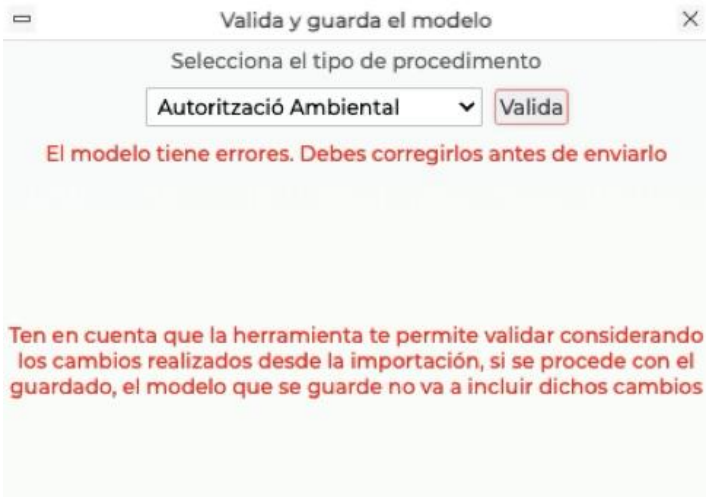
L'eina proporciona informació a l'usuari mitjançant un missatge de color vermell que bé pot ser un missatge informatiu o bé pot indicar si hi ha hagut un error en validar i a què és degut. Per exemple a la imatge anterior es mostra un missatge informatiu (missatge que està disponible en castellà i català segons l'idioma configurat a l'entorn de l'usuari). Si es fes clic sense haver carregat cap model llavors indicaria que hi ha hagut un error i la seva causa, com es pot comprovar a la següent imatge.



Si es fa clic a validar apareix un menú on, si l'usuari ja estava autenticat, apareixerà el seu usuari de manera automàtica i no es podrà editar, però, si no s'havia autenticat prèviament, es permetrà introduir el nom d'usuari i la contrasenya per fer-ho.
Un cop completada l'autenticació, es continua amb la validació del model executant l'avaluació de cadascuna de les regles existents en els fitxers de regles corresponents al tràmit.
En el cas de l'"autorització ambiental", es comprova el fitxer de regles vinculat amb aquest tramit, definit a l'arxiu script anomenat **default_properties.js**.
Un cop executada la validació, el llistat de regles apareix en una finestra a la cantonada superior esquerra, just a sobre de la finestra de l'eina de validació.



Si hi ha algun element que no compleixi les regles, llavors, a la part superior del llistat, apareixeran els elements que incompleixen les regles amb què estan relacionats. En fer clic sobre la regla s'indicarà el motiu pel qual s'incompleix. A més, des de l'eina de validació, no apareixerà el menú d'autenticació, sinó que apareixerà un missatge indicant que hi ha hagut algun error amb el model.



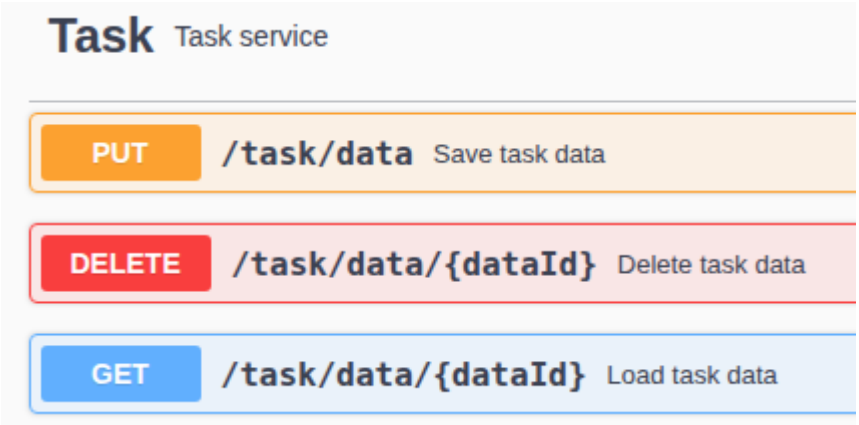
Si tant el model com l'usuari són correctes, aleshores, en fer clic al botó d'"enviar", apareixerà un missatge confirmant que s'ha pogut desar correctament el model, que quedarà guardat dins la carpeta de models dins de la carpeta del tràmit seleccionat.
Cal dir que tota la customització del tràmit es realitza mitjançant l'edició per part d'un usuari Administrador d'un fitxer de propietats default_properties.js que permet definir quins tràmits s'ofereixen, quins fitxers de regles executen, on s'emmagatzemen els models pujats, amb quins permisos es creen les carpetes de projecte dintre de les carpetes del tramit concret, entre d'altres opcions.

En relació a la jerarquia de carpetes, tots els models es troben al repositori cloud de models (carpeta models dintre de cloudfs). Dintre de la carpeta de models es troben les carpetes del tràmit que han de ser creades prèviament (i amb aquest objectiu s'explica en l'apartat de inicialització com es poden crear fàcilment dites carpetes per un Administrador).
Dintre de les diferents carpetes de tràmit, com seria la d'autoritzacio_ambiental, els projectes s'organitzen en carpetes segons l'ID del projecte. Automàticament es considera que l'ID de projecte d'un model que hem carregat es troba al camp globalId de la informació IFC continguda en el fitxer IFC del model.

Una vegada s'inicia el subprocés intern de desat del model, després d'haver estat validat, es comprova si el que s'està pujant correspon a una nova versió d'un projecte ja creat o bé es tracta d'un nou projecte que s'ha de crear.
En aquest sentit, el desat d'un model implica dues accions:

1) Creació o actualització de les metadades del projecte dintre del que considerem la DB de projectes , així com crear el bcfProject vinculat al projecte dintre de la DB de BCF.
2) Creació del directori de projecte si no existeix i desat del fitxer ifc corresponent a la versió de model que s'està pujant. Cada fitxer de versió queda identificada, ja que automàticament es desa el model amb la seva data i hora.

Per tal de dur a terme la creació de un “projecte” com a tal i desar les seves metadades es fa servir internament l’estructura de Task Data oferta per Bimrocket. A continuació es pot veure els endpoints que s’ofereixen a bimrocket per crear/modificar, eliminar o obtenir dades d’un element TaskData.



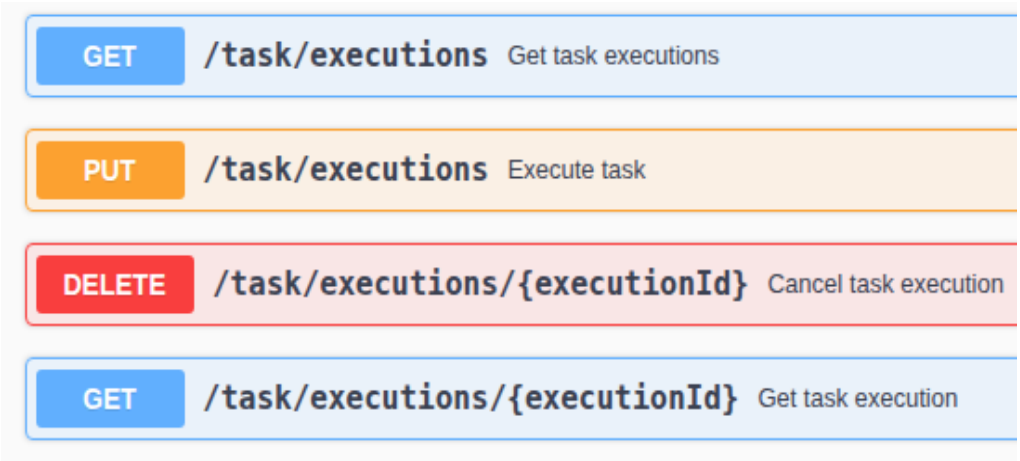
En el present Pilot, s’ha definit que per cada projecte que es crei, es desaran les següents metadades en el TaskData:



Però el front-end no realitza invocacions directament als endpoints del TaskData sinó que ho fa per mitjà d’un altre endpoint Task Executions que permet executar unes rutines que s’han preparat concretament per tal de, entre altres accions:

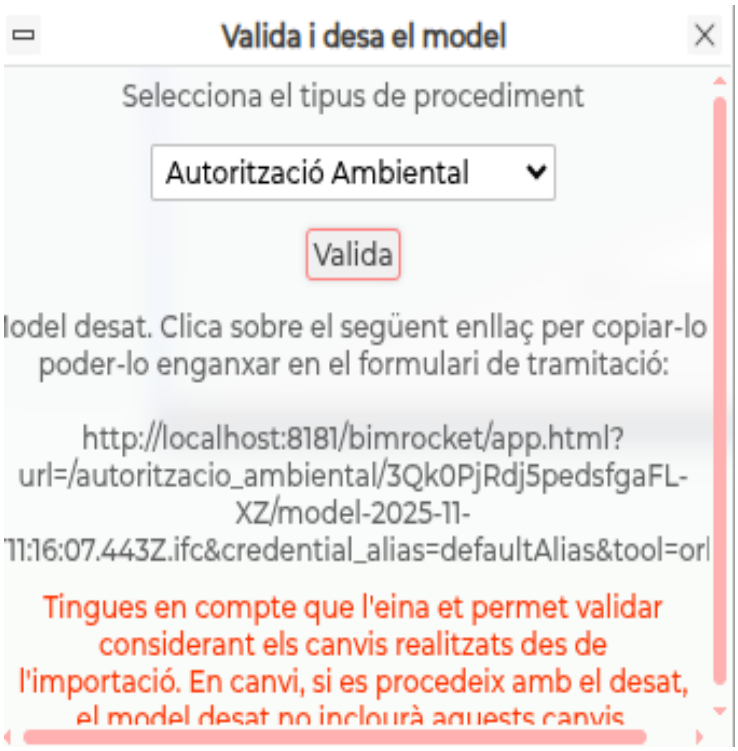
- Crear_projecte (inclou tant la creació de l’element que representa al projecte dintre del Task Data , així com també la creació del bcfProject dintre la BCF DB)
- Obtenir estat i altres metadades d’un projecte
- Actualitzar la versió del model (vinculant una nova versió del model)
- Actualitzar la fase en que es troba el projecte
- Etc.

D’aquesta manera, la manera com es pot arribar a generar aquest element amb el qual representem els Projectes serà executant l’endpoint PUT de la API de Task/Executions enviant els inputs necessaris.



En el cas que globalIFCid del model correspongui a un projecte ja existent, el sistema ho detecta i l'acció a realitzar serà la de pujar una nova versió, pujant el fitxer ifc al directori de projecte corresponent i actualitzant el camp currentVersion d'aquell projecte dintre del Task/Data. Això si, abans es comprova que la persona que realitza la operació tingui permisos (**per defecte, s'asseguren les restriccions que només el mateix projectista pot pujar noves versions d'un model, i que el projecte es trobi en la fase de "Requeriment al Projectista"** (el canvi de fase el gestionen els usuaris ADMIN del pilot).

Una vegada desat el model es mostra un missatge que proporciona un enllaç que, si es fa clic a sobre, es copiarà automàticament i es podrà enganxar al navegador per obrir-lo tal com ha estat desat.



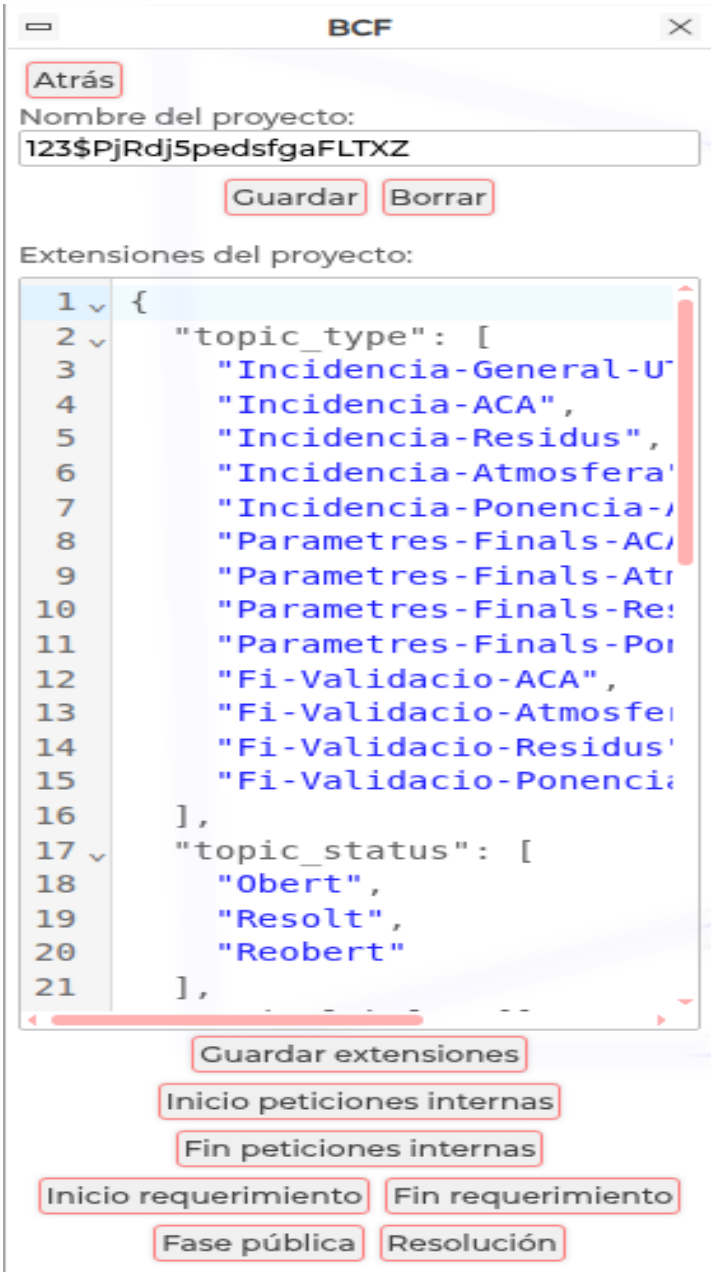
Bugfix: escapament de noms de directori “ifc_project” per tal d’evitar $

Amb la funcionalitat de validar i desar, un cop feta la validació , si tot és correcte i passa el fitxer de regles completament, es procedeix a desar el model.
Primer es comprova que existeixi la carpeta de projecte i de no ser així la crea. És en aquests moments en què es podria produir un error si el project_ifc conté el caràcter “$” (com havia succeït en alguns models d’exemple). Per això s’ha optat per escapar aquest caràcter, substituint-lo per un caràcter vàlid en la crida que fa el front-end cap al backend per a la verificació/creació de la carpeta de projecte en el repositori de models.
D’aquesta manera al backend sempre li arriben directoris amb nom vàlid que no genera cap error.
Aquesta mesura d'escapament actua com una capa de **sanitització de dades** imprescindible per garantir la interoperabilitat entre el front-end i el sistema de fitxers del servidor. En neutralitzar caràcters especials reservats pel sistema operatiu o pel protocol de transferència, s'assegura que el flux de treball no s'interrompi durant la persistència de les dades.
En resum, aquesta implementació no només resol la incidència específica amb el símbol “$”, sinó que estableix un **protocol de seguretat en la generació de rutes**. Això permet que el procés de desat sigui resilient davant de nomenclatures no estàndard en els models IFC, garantint que l'estructura del repositori es mantingui íntegra i accessible en tot moment.

Botons d’iniciPeticioInterna , fiPeticioInterna, iniciRequeriment, fiRequeriment, FasePublica, Resolució

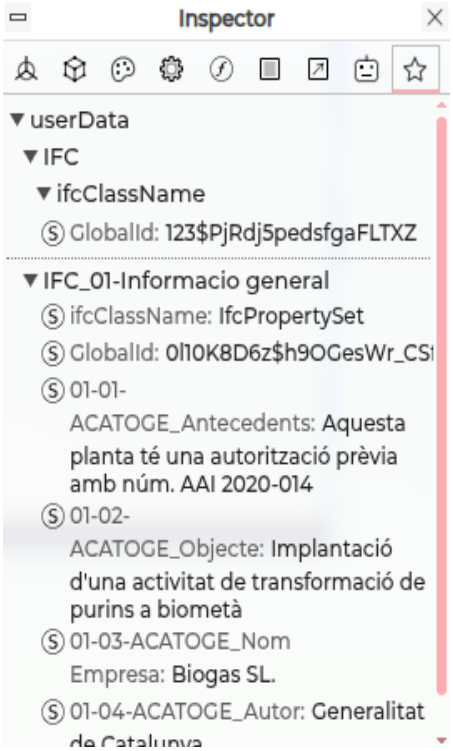
S’ha inclòs una funcionalitat dintre del menú “config” d'un projecte BCF (al visor Bimrocket) per tal que un administrador pugui realitzar canviar la fase en que es troba un projecte (inclosos els respectius permisos del sistema de incidències BCF corresponents a cadascuna de les fases), senzillament clicant el botó corresponent.
D’aquesta manera els ADMINS de la plataforma que farien el paper de Unitat Tramitadora, poden fàcilment canviar l’estat del projecte facilitant el fluxe de tramitació descrit en el document de requeriments.

- **Inici Petició Interna**. Fase (underVectorReview) en que els validadors poden realitzar les corresponents validacions i crear incidències.
- **Fi Petició Interna**. Marca el final de un cicle de validació, de manera que es prohibeix la creació de més incidències en aquesta fase (Utmanagement).
- **Inici de Requeriment**. Fase (faseRequerimentProjectista) en que els projectistes poden pujar noves versions d’un model, així com visualitzar les incidències que lis han creat els validadors
- **Fi de Requeriment**. Marca el final de la fase de Requeriment, tornant a la fase UTmanagement . D’aquesta manera es prohibeix que els projectistes puguin veure les incidències i puguin pujar noves versions d’un projecte.
- **Fase Pública** : El projecte queda restringit a Admin i Vectors per a la seva validació final.
- **Publicació o fi de validació**. En aquest moment el projecte es considera validat i passa a ser definitiu, per la qual cosa es procedeix amb el desat de les dades contingudes en la darrera versió del model cap a la base de dades estructurada IFCDB.


BCF
Atrás
Nombre del proyecto:
123$PjRdj5pedsfgaFLTXZ
Guardar
Borrar
Extensiones del proyecto:
{
"topic_type": [
"Incidencia-General-U
"Incidencia-ACA",
"Incidencia-Residus",
"Incidencia-Atmosfera
"Incidencia-Ponencia-
"Parametres-Finals-AC
"Parametres-Finals-At
"Parametres-Finals-Re
"Parametres-Finals-Po
"Fi-Validacio-ACA",
"Fi-Validacio-Atmosfe
"Fi-Validacio-Residus
"Fi-Validacio-Ponenci
],
"topic_status": [
"Obert",
"Resolt",
"Reobert"
],
Guardar extensiones
Inicio peticiones internas
Fin peticiones internas
Inicio requerimiento
Fin requerimiento
Fase pública
Resolución

Inicialització del menú “preferits” incloent per defecte algunes propertySets pre-definides (Pilot)

En lloc d'haver de navegar per múltiples pestanyes (Geometria, Material, Dades d'usuari, etc.) per trobar les dades que més rellevància tenen pels validadors, la pestanya de Preferits actua com un recull dels rellevants en matèria ambiental.



La manera com funciona és basa en un filtratge intel·ligent: El menú és dinàmic. Encara que s’hagin guardat 20 propietats com a preferides, l'inspector només mostrarà aquelles que realment existeixin en l'objecte que tenim seleccionat en aquell moment. Si canviem a un element que no té una d'aquestes dades, aquesta desapareixerà automàticament de la vista per no embrutar la interfície.
A més s’ha implementat perquè no sigui només una vista de lectura. Podem modificar els valors directament des d'aquí (canviar un número, un color o un text) i els canvis s'aplicaran a l'objecte exactament igual que si ho féssim a la seva pestanya original.

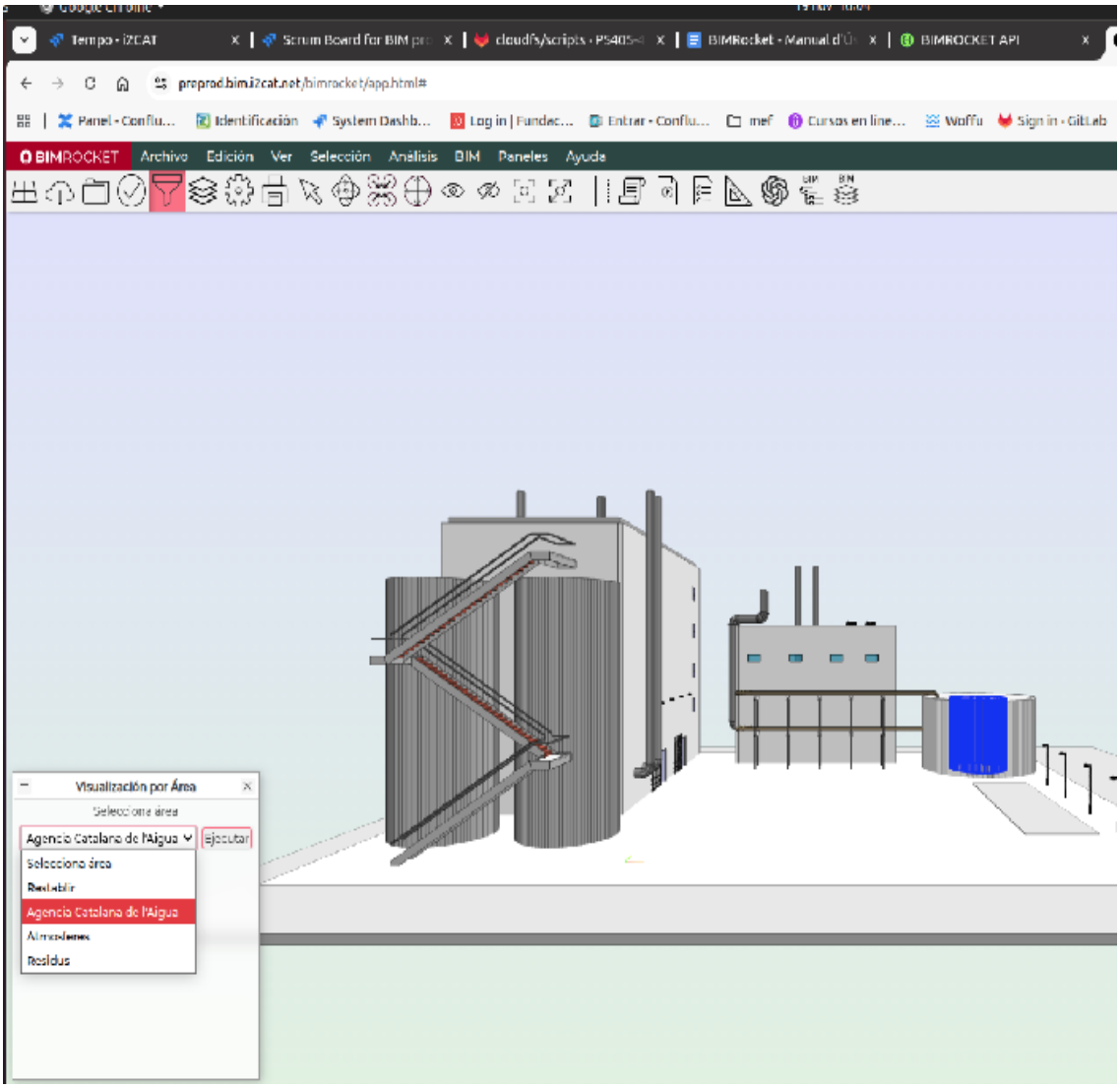
L'usuari té el control absolut sobre què apareix en aquest menú:

- Afegir i treure sobre la marxa: Mitjançant el botó dret sobre qualsevol propietat de l'inspector, es pots "enviar-la" a preferits o eliminar-la si ja no cal.
- Persistència: El sistema recorda les preferències. Si tanquem l'aplicació i tornem l'endemà, el llistat de preferits seguirà configurat tal com es va deixar.
- Neteja ràpida: Existeix una opció per "buidar" tot el tauler de preferits d'un sol cop si vols tornar a començar la teva configuració des de zero.

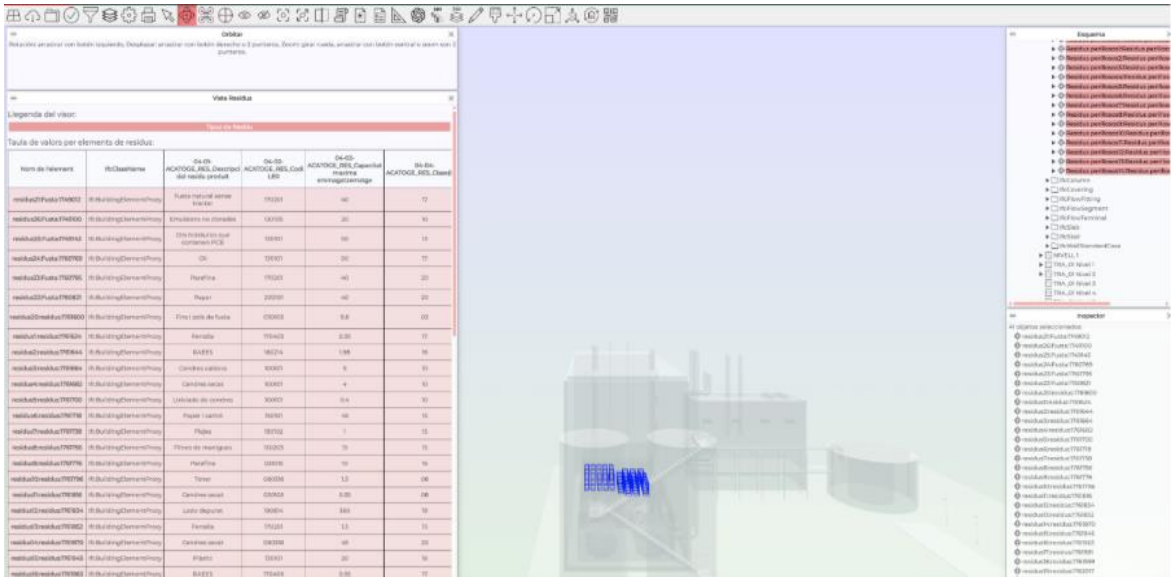
Per defecte, a banda dels possibles preferits que l’usuari hagi pogut afegir manualment, sempre es mostren les propietats que han estat definides en el fitxer default_properties.js, sempre i quan l’element concret que s’està inspeccionant contingui dita propietat.

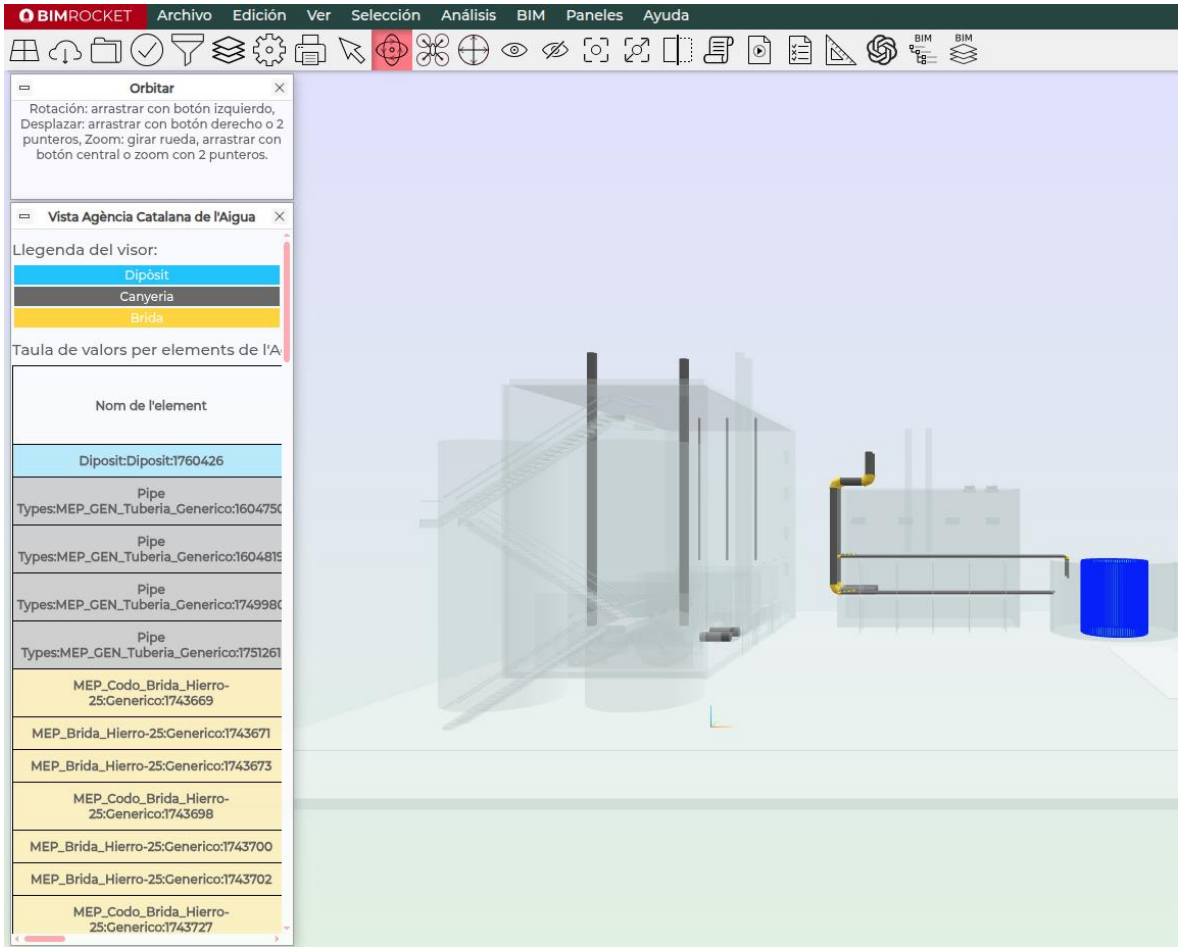
Filtre de vista per Vector/Àrea (Pilot) i Quadre de característiques

Per tal de facilitar la validació per part de les àrees validadores, s'ha creat una eina de Filtre per Àrea que permet destacar els elements que són de rellevància per aquell vector/àrea



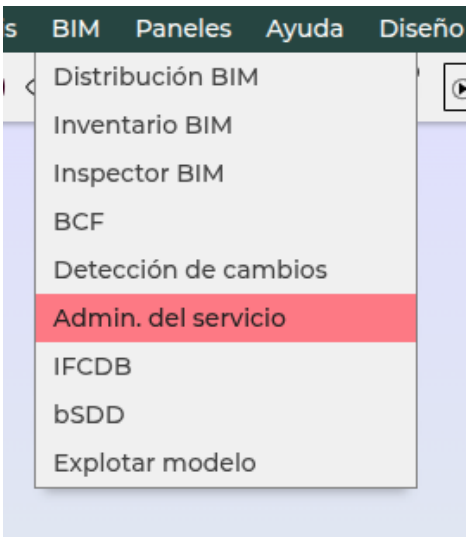
L'execució d'un d'aquests filtres, mostrarà en el visor els elements que s'han considerat rellevants per aquell vector i seleccionarà els elements pertinents.
A més es mostra una taula resum de les propietats (columnes de la taula) rellevants en matèria "ambiental" de cadascun dels elements seleccionats pel filtre (files de la taula).



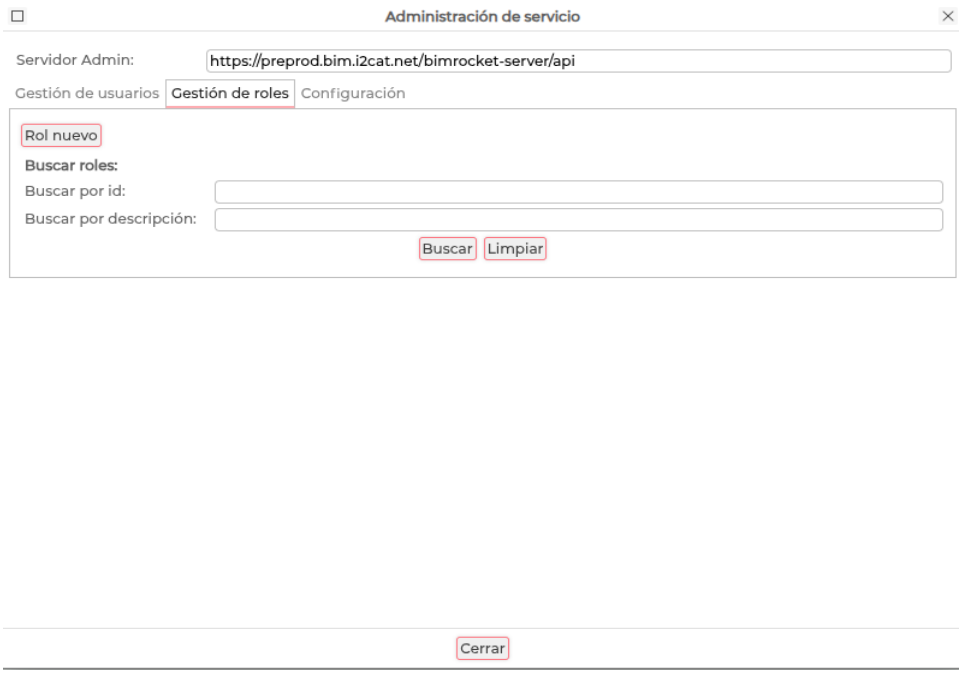

BIMROCKET
Archivo
Edición
Ver
Selección
Análisis
BIM
Paneles
Ayuda
Orbitar
Rotación: arrastrar con botón izquierdo, Desplazar: arrastrar con botón derecho o 2 punteros, Zoom: girar rueda, arrastrar con botón central o zoom con 2 punteros.
Vista Agència Catalana de l'Aigua
Llegenda del visor:
Dipòsit
Canyeria
Brida
Taula de valors per elements de l'A
Nom de l'element
Diposit:Diposit:1760426
Pipe Types:MEP_GEN_Tuberia_Generico:160475
Pipe Types:MEP_GEN_Tuberia_Generico:160481
Pipe Types:MEP_GEN_Tuberia_Generico:174998
Pipe Types:MEP_GEN_Tuberia_Generico:175126
MEP_Codo_Brida_Hierro-25:Generico:1743669
MEP_Brida_Hierro-25:Generico:1743671
MEP_Brida_Hierro-25:Generico:1743673
MEP_Codo_Brida_Hierro-25:Generico:1743698
MEP_Brida_Hierro-25:Generico:1743700
MEP_Brida_Hierro-25:Generico:1743702
MEP_Codo_Brida_Hierro-25:Generico:1743727

Eina de Gestió Rols (Comunitat i Pilot)

S'ha implementat una eina que permet la creació de rols de la plataforma, permetent herència de altres rols.
Des de la barra de menús, escollim BIM i seleccionem Admin. del servei:



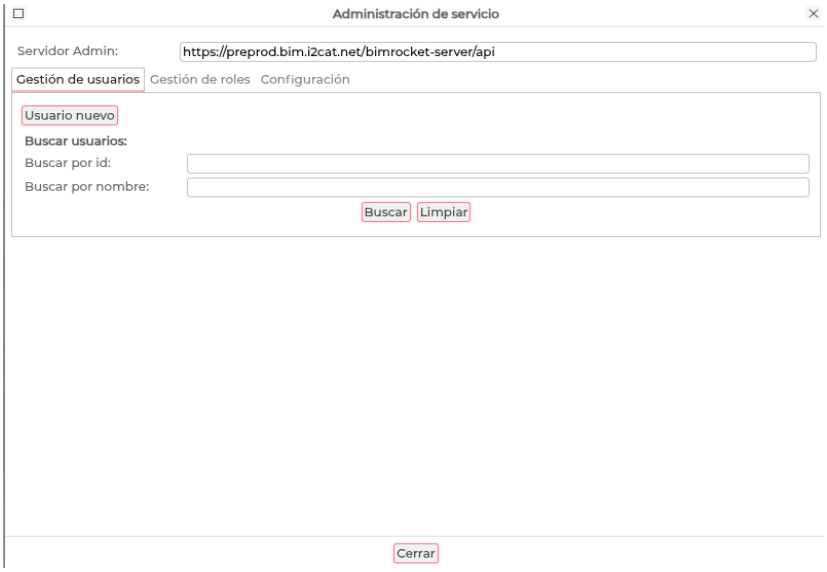
Aquesta eina permet realitzar cerques de rols creats filtrant pel seu id o descripció.

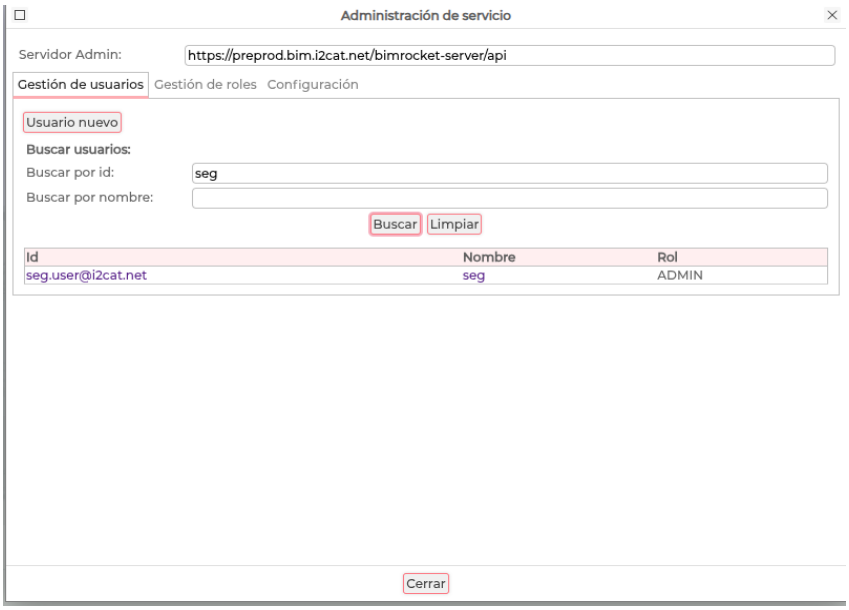


Eina de Gestió Usuaris (Comunitat i Pilot)

També s'ha implementat una eina de gestió de Usuaris, a la qual podem accedir també des de Administració del Servei.
En la cerca d'usuaris es pot filtrar els usuaris existents per id i nom.
Fent clic a **Cerca** es poden buscar els usuaris d'acord amb els camps de filtre establerts. Si no es posa cap text llavors es retornen tots els usuaris.
Clicant **Neteja** es poden buidar els camps.

En la creació i modificació d'un usuari es important explicar que la contrasenya es obligatòria en la creació. D'altra banda, si en la modificació la deixem en blanc, llavors no es modificarà la contrasenya actual. També es poden escollir d'un seleccionable dinàmic quins rols (d'entre els rols que s'han creat en el sistema mitjançant l'eina anterior) es volen seleccionar per aquell usuari.





Adaptació repositori de fitxers per a funcionar en BlobStorage (Comunitat i per a una futura versió productiva Serverless del Pilot)

Com a complement al sistema tradicional de fitxers s'ha implementat una nova classe en el projecte (**`AzureBlobFileStore`**).
A l'application.yaml s'han definit les propietats d'accés al Blob Storage així com la classe utilitzada. No es pot treballar a la vegada amb sistema d'arxius tradicional y Blob Storage, s'ha d'escollir amb quina modalitat volem treballar.

Si volem treballar amb sistema tradicional:

```yaml
# file service
file:
  folders:
    - models
    - scripts
    - reports
    - ifc_snapshots
    - tasks
  store:
    class: org.bimrocket.service.file.store.filesystem.FileSystemFileStore
    filesystem:
      directory: ${BIMROCKET_DATA_PATH}/cloudfs
```

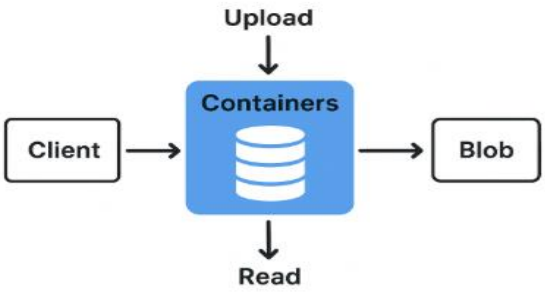
Si volem treballar amb Blob Storage:

```yaml
# file service
file:
  folders:
    - models
    - scripts
    - reports
    - ifc_snapshots
    - tasks
  store:
    class: org.bimrocket.service.file.store.filesystem.AzureBlobFileStore
    azure:
      connection-string: >
        DefaultEndpointsProtocol=http;
        AccountName=devstoreaccount1;
        AccountKey=Eby8vdM02xNOcqFeqCnb3w==;
        BlobEndpoint=http://azurite:10000/devstoreaccount1;
      container-name: azurite
```

Per confidencialitat hem indicat un connection-string fent servir un contenidor docker Azurite per simular un accés a Azure real per evitar mostrar les dades reals de connexió a un servei Azure real.

El flux bàsic del BlobStorage és:

1. El client envia dades (fitxers, imatges).
2. Les dades es desa en **containers**, que són com “carpetes”.
3. Cada fitxer es converteix en un **blob** (block, append o page).
4. Les aplicacions poden **llegir, escriure, llistar o esborrar** blobs segons permisos.



L'estructura dins del BlobStorage seria:

```
Storage Account
    └── Container
            └── Blob
```

A Azure Blob Storage, les carpetes i subcarpetes no existeixen realment com a directoris físics: són pseudo carpetes, creades només per la *ruta del nom del blob*. Això vol dir que tota l'estructura és plana, però es mostra com si hi hagués carpetes.

Estructura conceptual:

```
Storage Account
  └── Container
        ├── carpeta1/fitxerA.txt
        ├── carpeta1/subcarpeta1/fitxerB.jpg
        ├── carpeta1/subcarpeta1/fitxerC.json
        ├── carpeta2/video.mp4
        └── fitxerArrel.pdf
```

Estructura real:

```
Container
   ├── Blob: "carpeta1/fitxerA.txt"
   ├── Blob: "carpeta1/subcarpeta1/fitxerB.jpg"
   ├── Blob: "carpeta2/video.mp4"
   └── Blob: "fitxerArrel.pdf"
```

No podem accedir al Blob Storage de manera directa com si fos un sistema tradicional de fitxers. Haurem d’utilitzar algun client de tipus command line o GUI per consultar els blobs i que siguin mostrats de manera llegible i també poder veure el seu contingut.

Millores usabilitat Eina BCF i filtre de projectes BCF (comunitat i Pilot)

S'ha realitzat una modificació substancial de la UI del panell BCF per millorar la usabilitat de l'eina, separant la lògica en 3 nivells (cercador projectes, informació d'un projecte particular i els seus tòpics, informació d'un tòpic concret).
En la descripció de la funcionalitat de BCF de la secció anterior, es pot veure el funcionalment i aparença del panell en el seu estat actual, però cal remarcar que dita aparença i funcionament és posterior a les millores implementades.



Entre d'altres millores, es permet filtrar els projectes bcf filtrant per nom. De totes maneres, a la versió del pilot només es mostren els projectes BCF corresponents a un model carregat al visor, per la qual cosa el filtre no serà gaire utilitzat en el Pilot.



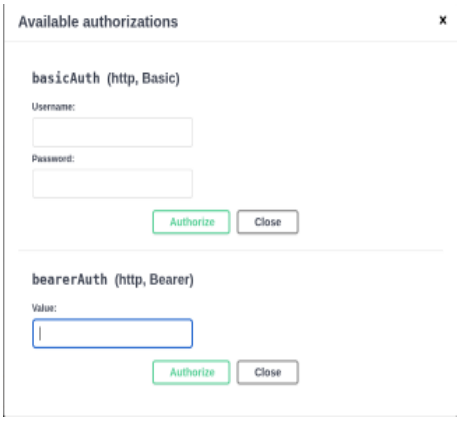
Autenticació via Token bearer

Amb aquesta millora l'aplicació permet accedir als endpoints de la API utilitzant un token Basic a partir de les credencials o utilitzant un token JWT de tipus Bearer. Per tal de poder implementar aquest procés s'ha fet una sèrie de canvis a l'aplicació.

S'ha afegit al fitxer openapi.json un nou schema per incloure el Bearer:

```
{
  "securitySchemes" : {
    "basicAuth" : { "type" : "http", "scheme" : "basic" },
    "bearerAuth" : { "type" : "http", "scheme" : "bearer", "bearerFormat": "JWT" }
  }
},
"security" : [{ "basicAuth" : [] }, { "bearerAuth" : [] }]
```

Per veure-ho de forma visual, si accedim al swagger del backend tenim la possibilitat d'autoritzar-nos amb credencials o amb token.



Per tal de generar els token d'accés i de refresh, cal fer login des del Frontend.
En aquest procés si l'usuari no existeix a la BD de l'aplicació el donem d'alta, sinó l'editem. En ambdós casos, recollim els token obtinguts de keycloak, Vàlid o Gicar i els guardem a la BD juntament amb les altres dades d'usuari . El Backend envia al Frontend l'accessToken si el procés de login ha estat satisfactori.

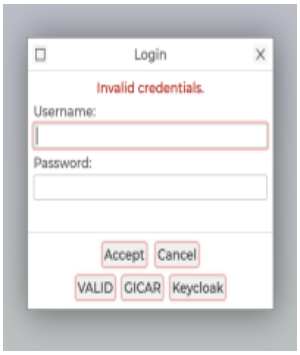
Cada cop que fem servir l'accessToken des del Frontend, el SecurityService del backend comprova si el token existeix per a un usuari a la BD La BD guarda a la taula User l'accessToken i el refreshToken. Si no existeix cap usuari amb el token donem un missatge d'error. Si l'usuari existeix es guarda a la cache de l'aplicació les dades d'usuari incloent els token.

Quan fem una invocació als endpoints de la API des del Frontend verifiquem si l'accessToken guardat a la cache ha expirat o no. El backend comprova si el refresh token ha expirat, si ha expirat donarem un missatge d'error i serà necessari fer login des del Frontend per autenticar-se altre vegada per tal de regenerar els token, si no ha expirat el refresh token, reutilitzem l'accessToken generat prèviament i només augmentem la seva expiració en 5 minuts més.

L'accessToken té una duració de 5 min i el refreshToken de 2 hores.

Integració oAuth (VALid, Keycloak, GICAR)

S'ha afegit un sistema d'autenticació basat en OAuth OIDC (OpenID Connect). Aquest mecanisme permet als usuaris iniciar sessió mitjançant proveïdors d'identitat externs (IdP) com Keycloak, VÀLID o GICAR, mostrant botons dinàmics al diàleg de login.



Configuració de l'entorn

La lògica d'autenticació i la personalització dels botons disponibles al formulari de login es defineixen a l'arxiu Environment.js, en concret és la clau AUTH_ENVIRONMENT la que conté la configuració específica per a cada entorn de desplegament.

L'aplicació detecta automàticament l'entorn comparant l'origen del navegador (la URL des de la qual s'obre l'aplicació) amb les claus definides a l'objecte.

Estructura de AUTH_ENVIRONMENT

Dins de AUTH_ENVIRONMENT cal definir l'URL d'origen, dins d'aquesta clau, s'afegeixen els diferents proveïdors d'identitat.

El nom de la clau de cada proveïdor (per exemple: Vàlid, Gicar o Keycloak entre altres) serà el text que es mostrarà al botó del formulari de login.

```
export const Environment =
{
  // ... altres configuracions
  AUTH_ENVIRONMENT : {
    "https://preprod.bim.i2cat.net": {
      valid: {
        authUrl: "https://valid-pre.aoc.cat/o/oauth2/auth",
        clientId: "...",
        scope: "autenticacio_usuari",
        redirectUri: "https://preprod.bim.i2cat.net/bimrocket-server/api/oauth2/authCode/valid",
      },
      gicar: {
        authUrl: "...",
        clientId: "...",
        scope: "...",
        redirectUri: "https://preprod.bim.i2cat.net/bimrocket-server/api/oauth2/authCode/gicar",
      },
      keycloak: {
        authUrl: "https://iam.i2cat.net/auth/realms/SEG/protocol/openid-connect/auth",
        clientId: "bim",
        scope: "openid",
        redirectUri: "https://preprod.bim.i2cat.net/bimrocket-server/api/oauth2/authCode/keycloak",
      },
    },
    // configuració per a un altre entorn
    "https://bim.santfeliu.cat": {
       // ... configuració específica per a aquest domini
    },
  }
};
```

Paràmetres de Configuració del Proveïdor

Per a cada proveïdor configurat, és obligatori definir els següents paràmetres segons l'estàndard OAuth2:

- authUrl: L'URL del Proveïdor d'Identitat (IdP) on s'inicia el procés d'autenticació; és la pàgina que mostrarà el formulari extern per introduir les credencials.
- clientId: L'identificador únic de l'aplicació client; aquest valor és proporcionat per l'IdP i serveix perquè aquest reconegui l'origen de la petició.
- scope: Defineix l'abast de l'accés sol·licitat, habitualment el valor és "openid", però pot variar segons l'IdP (per exemple: "autenticacio_usuari" en el cas de Vàlid). Cal consultar la documentació de l'IdP.
- redirectUri: L'URL de retorn on l'IdP redirigirà l'usuari un cop autenticat correctament; aquesta URL ha d'apuntar a l'endpoint específic del Backend encarregat de processar el codi d'autorització.

Integració amb el Backend (TokenValidationServlet)

El flux de treball connecta el Frontend amb el Backend mitjançant la redirectUri. Quan l'usuari s'autentica a l'IdP, aquest fa una redirecció al servidor afegint un paràmetre code a l'URL.

Per tal d'obtenir l'accessToken des del Backend durant aquest procés, s'ha implementat un nou TokenValidationServlet que implementa un mètode GET què és invocat automàticament per la redirecció de la plataforma corresponent i per tant cada vegada que es fa una petició de login a la corresponent plataforma (keycloak, Vàlid o Gicar).

El sistema utilitza el path final de la URL per discriminar quina plataforma s'està utilitzant i processar la petició adequadament, aquest són uns exemples d'estructura de la petició de redirecció:

- Keycloak: .../api/oauth2/authCode/keycloak
- VÀLID: .../api/oauth2/authCode/valid
- GICAR: .../api/oauth2/authCode/gicar

La URL completa construïda (que és la redirectUri configurada a l'environment) tindria un aspecte similar a: https://url-del-servei/auth?client_id=bim&response_type=code&scope=openid&redirect_uri=.../api/oauth2/authCode/keycloak

Comportament del flux i persistència de les credencials

En finalitzar el flux OAuth correctament, el Backend retorna l'identificador d'usuari i el token d'accés al Frontend, aquestes credencials s'assignen i s'emmagatzemen automàticament a l'àlies del servei que ha requerit l'autenticació.

L'emmagatzematge en memòria no és persistent, com a conseqüència de recarregar la pàgina les credencials es perden, això implica que la següent crida al servei tornarà a sol·licitar autenticació, si l'usuari torna a fer clic al botó del proveïdor i la sessió del IdP encara és activa i/o el token no ha caducat, la redirecció es realitzarà de manera gairebé transparent a l'usuari sense la necessitat de que torni a autenticar-se manualment.

A l'application.yaml, s'han definit les següents propietats sota **security** amb els accessos a les diferents plataformes:

```yaml
oauth2:
  keycloak:
    urlAuthenticationToken: https://iam.i2cat.net/auth/realms/SE6/protocol/openid-connect/token
    clientId: bim
    secretId: secret_id
  valid:
    urlAuthenticationToken: https://accounts-dev.aoc.cat/o/oauth2/token
    urlGetUserInfo: https://identitats-pre.aoc.cat/serveis-rest/getUserInfo
    clientId: serveis.aoc.cat
    secretId: secret_id
  gicar:
    urlAuthenticationToken: https://preproduccio.endpointma.autenticaciogicar4.extranet.gencat.cat/realms/gicarcpd4/protocol/openid-connect/token
    urlGetUserInfo: https://preproduccio.endpointma.autenticaciogicar4.extranet.gencat.cat/realms/gicarcpd4/protocol/openid-connect/userinfo
    clientId: client_id
    secretId: secret_id
```

El TokenValidation, depenent de l'issuer o plataforma crida a la classe corresponent per tal d'executar el procés de login. Les noves classes s-on:

- KeycloakAuthManager
- ValidAuthManager
- GicarAuthManager

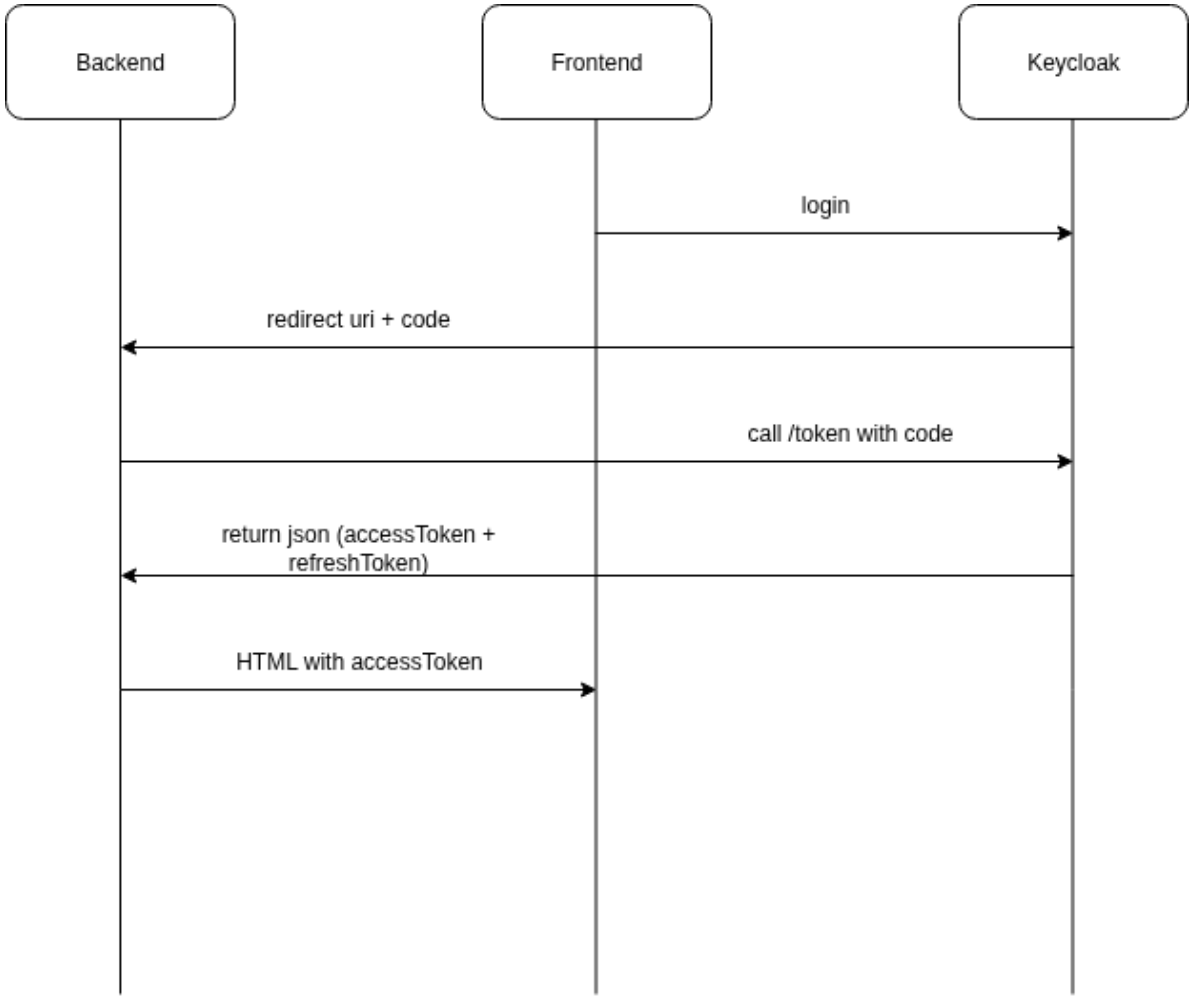
Aquestes classes hereten de la interfície **`AuthenticationManager`** que defineix els mètodes d'autenticació comuns per tots els processos.

```java
public interface AuthenticationManager  3 usages  3 implementations  miguel-diaz-i2cat
{
  public String getAuthenticationToken(String code, String redirectUri) throws Exception;

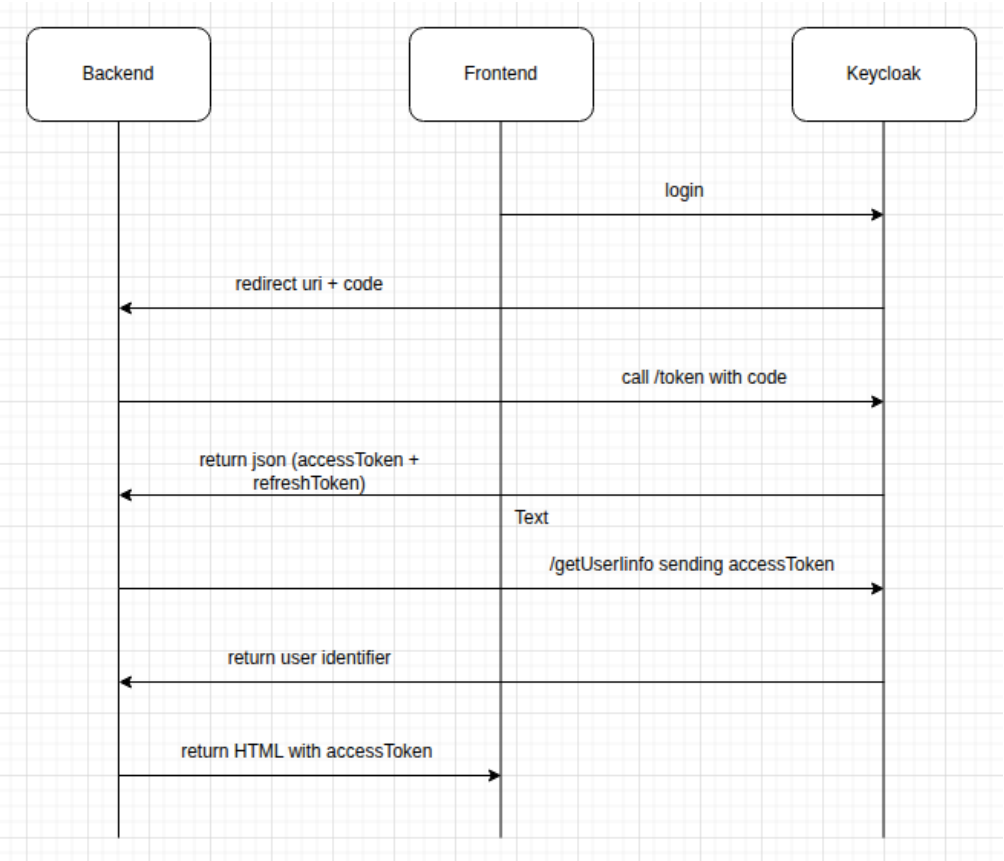
  public UserToken getUseridFromToken(String jsonToken) throws Exception;  3 usages  3 impleme
}
```

En el procés final del login, el Backend genera un HTML enviant l'accessToken al Frontend. Aquest HTML s'obrirà com a finestra filla (que es tanca immediatament) del diàleg del Frontend que ha fet la petició de login.

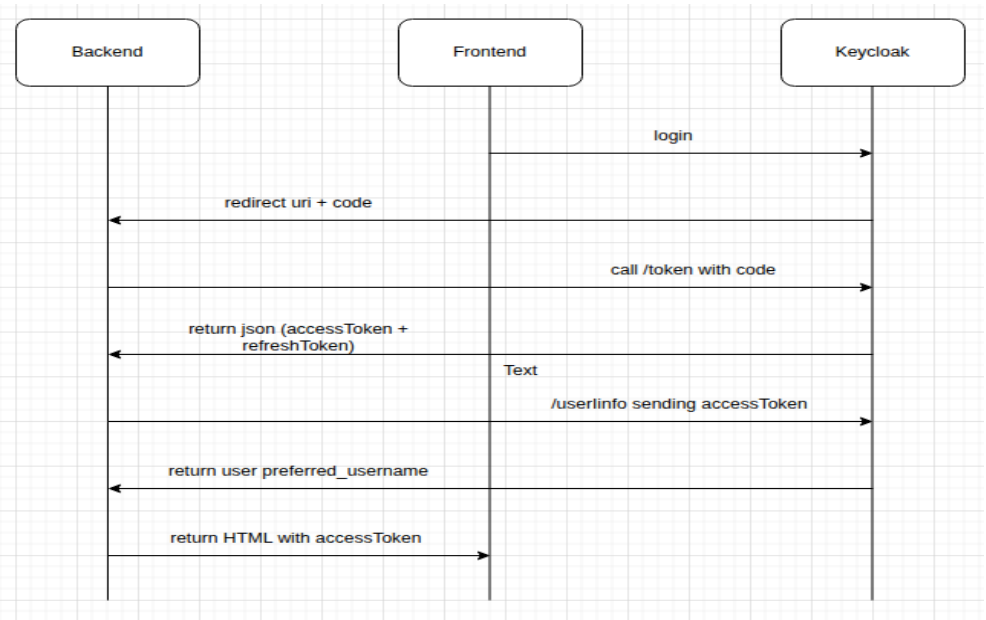
Per il·lustrar el procés de login per cadascun dels issuers o plataformes, els següents diagrames de seqüencia detallen els passos.



Flux keycloak pel login



Flux Vàlid pel login



Flux Gicar pel login

Eina de Capes GIS

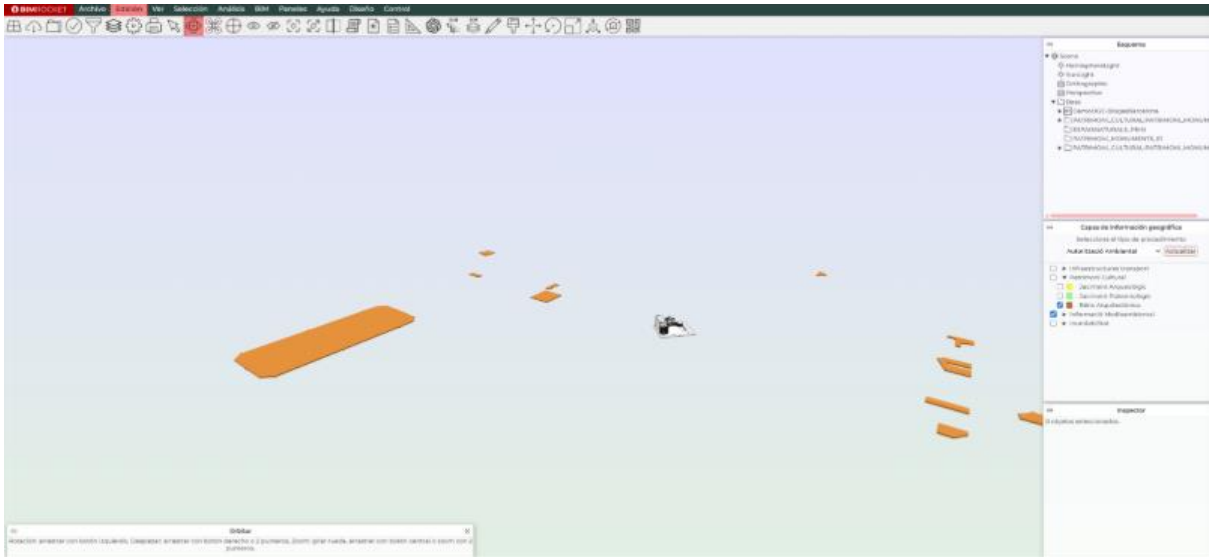
S'ha implementat una funcionalitat per Poder carregar les capes GIS que es considerin necessàries.

Per fer-ho, cal clicar a la icona: 
, apareix un panell a la dreta on es pot seleccionar el tràmit (ex. Autorització Ambiental) pel qual es vol que es mostrin les capes disponibles (clicar a actualitzar per confirmar el tràmit):



Es pot veure una llegenda al costat del nom de la capa que ens indica el color amb el que apareixen els elements d'aquella capa. En cas de que la capa tingui varies categories es mostren varis indicadors al costat del nom i l'usuari pot veure la informació relacionada amb aquell color just posicionant-se sobre el quadrat de color, moment en que apareix un *tooltip* amb el nom de la categoria.

En el panell anterior, l'usuari pot anar seleccionant i desactivant (fent clic en els checkbox) les capes que es volen visualitzar en les proximitats del model.



Quan es carrega una capa, l'element Esquema del visor s'actualitza amb un nou projecte que conté els elements obtinguts del servei de capes consultat. Quan es deshabilita una capa es suprimeix aquest projecte.

Cal indicar que la llista de capes disponibles (i els alguns paràmetres particulars per cridar al servei corresponent) es defineix en el fitxer de propietats default_properties.js i que pot ser modificat dinàmicament per incloure noves capes.

Les capes suportades són GeoJson, GML i en un futur pròxim també capes d'imatge d'un servei WMS.

Desplegament de la Plataforma per a desenvolupadors

Bimrocket es divideix en dos components principals: una UI (front-end) amb la que l'usuari pot interactuar i un backend que ofereix funcionalitats addicionals. Cal dir que el front-end per si sol pot funcionar sense backend i només en cas de necessitar alguna de les funcionalitats de backend, aquestes s'haurien de desplegar.
A continuació es detallen els passos a seguir segons casos diferents, des d'un nivell més orientat a desenvolupadors fins a un nivell orientat a desplegament productiu.

Descarregar codi i compilar tant front-end com backend

- **Java JDK 17**
- **Maven 3.6.3**
  Per generar el .war és necessari tenir instal·lat maven:
  i. Linux: *sudo apt install maven -y*
  ii. Windows: https://maven.apache.org/

- **Clonar el repositori**:
  - Per a la comunitat:
    Clona el repositori del projecte via SSH:
    git clone git@github.com:Fundacio-i2CAT/bimrocket.git
    Si hi ha algun problema d'accés al Github per certificats SSL, seguir aquests passos previs:
    - Verificar que la verificació SSL està activada:
      - *git config --global http.sslVerify true*
    - Actualitzar els certificats CA:
      - Linux:
        - *sudo apt update && sudo apt install --reinstall ca-certificates*
        - *sudo update-ca-certificates*
      - Windows:
        - Instal·lar els últims updates amb Windows Update

  - Per al Pilot:
    En el Pilot el backend i el frontend estan dividits en dos repositoris diferents, per tant, estaran descarregats en directoris diferents.
    Per assegurar que no es produeix cap error posterior de compilació en els dos mòduls, en estar separats en repositoris diferents, tant el backend com el frontend necessiten la llibreria bimrocket-commons. Per tant, cal fer clone del repositori bimrocket-commons i compilar-lo amb maven per tenir-la llibreria descarregada en el repositori de maven. Farem aquestes dues passes:

    git clone git@gitlab.i2cat.net:areas/software-engineering-group/bim/bimrocket-development/bimrocket-common.git

Des de terminal executem:
mvn clean install

Backend
git clone git@gitlab.i2cat.net:areas/software-engineering-group/bim/bimrocket-development/bimrocket-server.git
Frontend
git clone git@gitlab.i2cat.net:areas/software-engineering-group/bim/bimrocket-development/bimrocket-webapp.git

- **Preparar el fitxer de propietats pel backend. Segons el tipus de desplegament les propietats variaran.**
  bimrocket/bimrocket-server/src/main/resources/application.yaml
  (Comunitat)
  bimrocket-server/src/main/resources/application.yaml
  (Pilot)

  Una de les modificacions requerides serà incorporar les URL permeses perquè el proxy d'accés a serveis externs permeti la redirecció.
  De totes maneres en els següents passos expliquem com es poden modificar les propietats, directament en els fitxers de propietats una vegada desplegats al Tomcat.

- **Preparar el fitxer Environment.js per al frontend**

  Editar el fitxer:
  Comunitat –> *bimrocket/bimrocket-webapp/src/main/webapps/js/Environment.js*
  Pilot –> *bimrocket-webapp/src/main/webapps/js/Environment.js*

  *Per defecte, el fitxer d'Environment està setejat per a les versions **NO-dockeritzades***

- **Generar els fitxers .war per detectar que compili bé:**
  - Des de la comunitat:
  - Des del directori bimrocket: *mvn clean install*
  - Els **.war** generats es troben en les següents rutes:
    - bimrocket/bimrocket-server/target
    - bimrocket/bimrocket-webapp/target

  - Des del Pilot:
  - Des del directori bimrocket-server: *mvn clean install*
  - Des del directori bimrocket-webapp: *mvn clean install*
  - Els **.war** generats es troben en les següents rutes:
    - bimrocket-server/target
    - bimrocket-webapp/target

Desplegament Backend i Frontend no dockeritzat utilitzant IntelliJ

El Back-end és necessari per poder fer servir el sistema d'incidències i comentaris BCF, també per desar les classes IFC i altres fitxers (com per exemple models) en el servidor cloud (cloudFS), així com per la funcionalitat d'exportar a PDF.

Els serveis de backend s'ofereixen per mitjà de varis endpoints de l'API. Tota crida a aquests endpoints es du a terme via Autenticació bàsica, passant nom d'usuari i contrasenya.

El mòdul de back-end pot configurar-se per utilitzar un LDAP (que es pot configurar en el web.xml (dintre de WebPages>WEB-INF al paràmetre "userStore.class") però també es disposa d'altres tipus de mecanismes de gestió d'usuaris. Una de les millores que es poden veure en l'última secció del document és la creació d'un mecanisme de gestió d'usuaris híbrid per a ser utilitzat en aquest projecte Pilot.

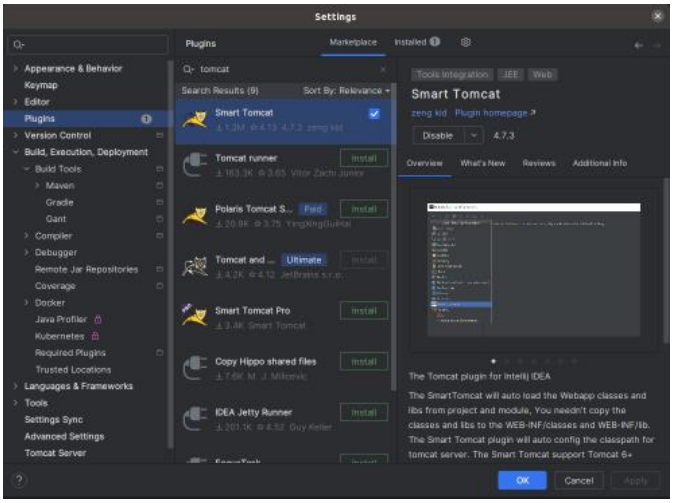
Per a configurar tant backend com frontend en local, fent servir intellij per a la compilació i desplegant a servidor Tomcat, hem de seguir les següents passes:

1) Descarregar Apache Tomcat 10.1:
    - Linux:
      https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.33/bin/apache-tomcat-10.1.33.zip

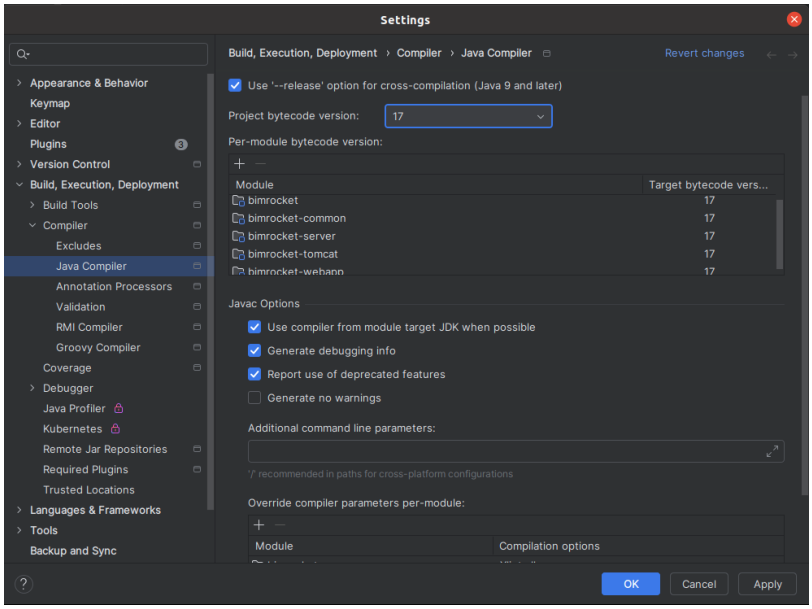
      Descomprimir en el lloc desitjat, anar a la carpeta /bin per tal de donar permisos d'execució als següents scripts:
      chmod +x startup.sh
      chmod +x shutdown.sh
      chmod +x catalina.sh

    - Windows:
      https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.33/bin/apache-tomcat-10.1.33-windows-x64.zip

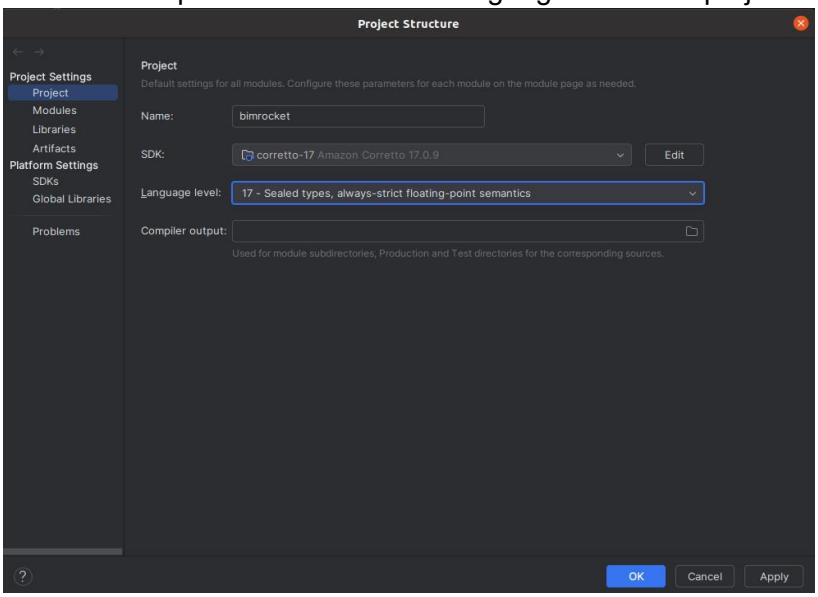
2) Des de intellij obrir el projecte clonat.
   Per a la **comunitat** obrirem el projecte des de la carpeta bimrocket.
   Per al **Pilot** obrirem backend i frontend en finestres de intelliJ diferents, bimrocket-server pel backend i bimrocket-webapp pel frontend
3) Obrir File-> settings
4) Instal·lar el plug in Smart Tomcat



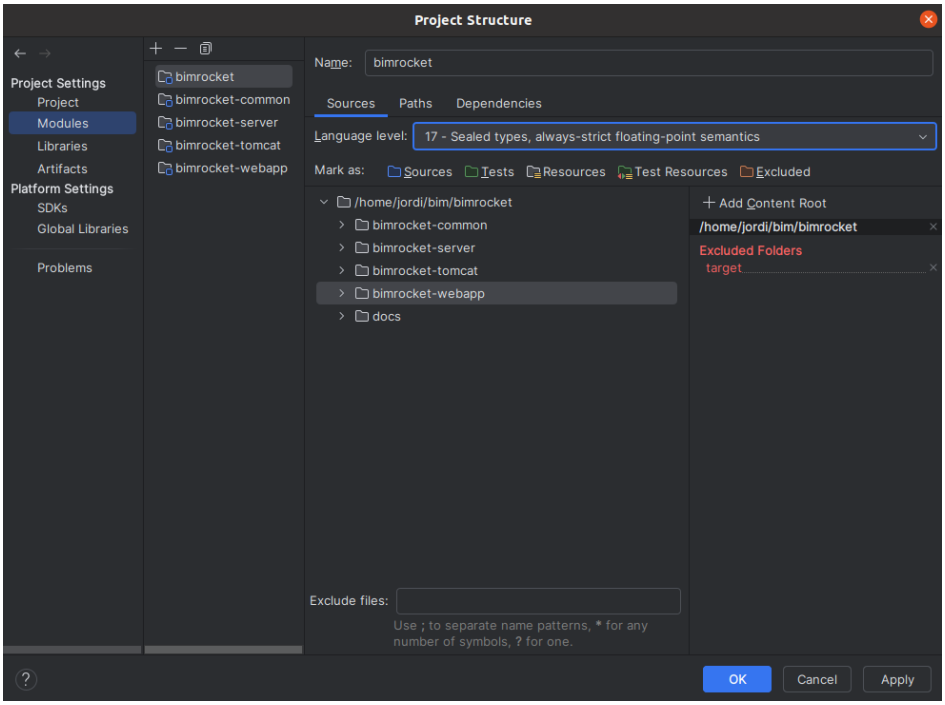
5) Definir el JDK per a Build, Execution i Deployment:



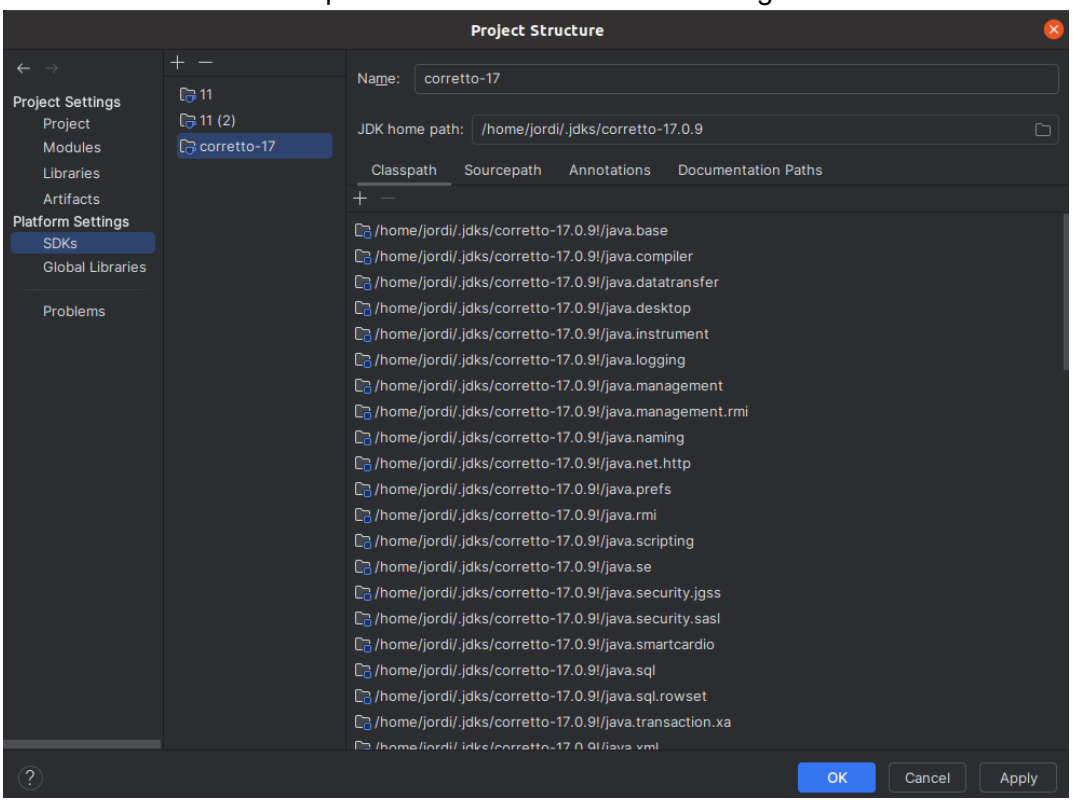
6) Des de File -> Project Structure
   - Especificar el SDK i el Language Level del projecte:



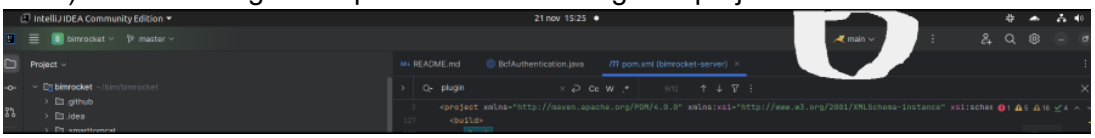
- Per cada mòdul especificar el Language level:



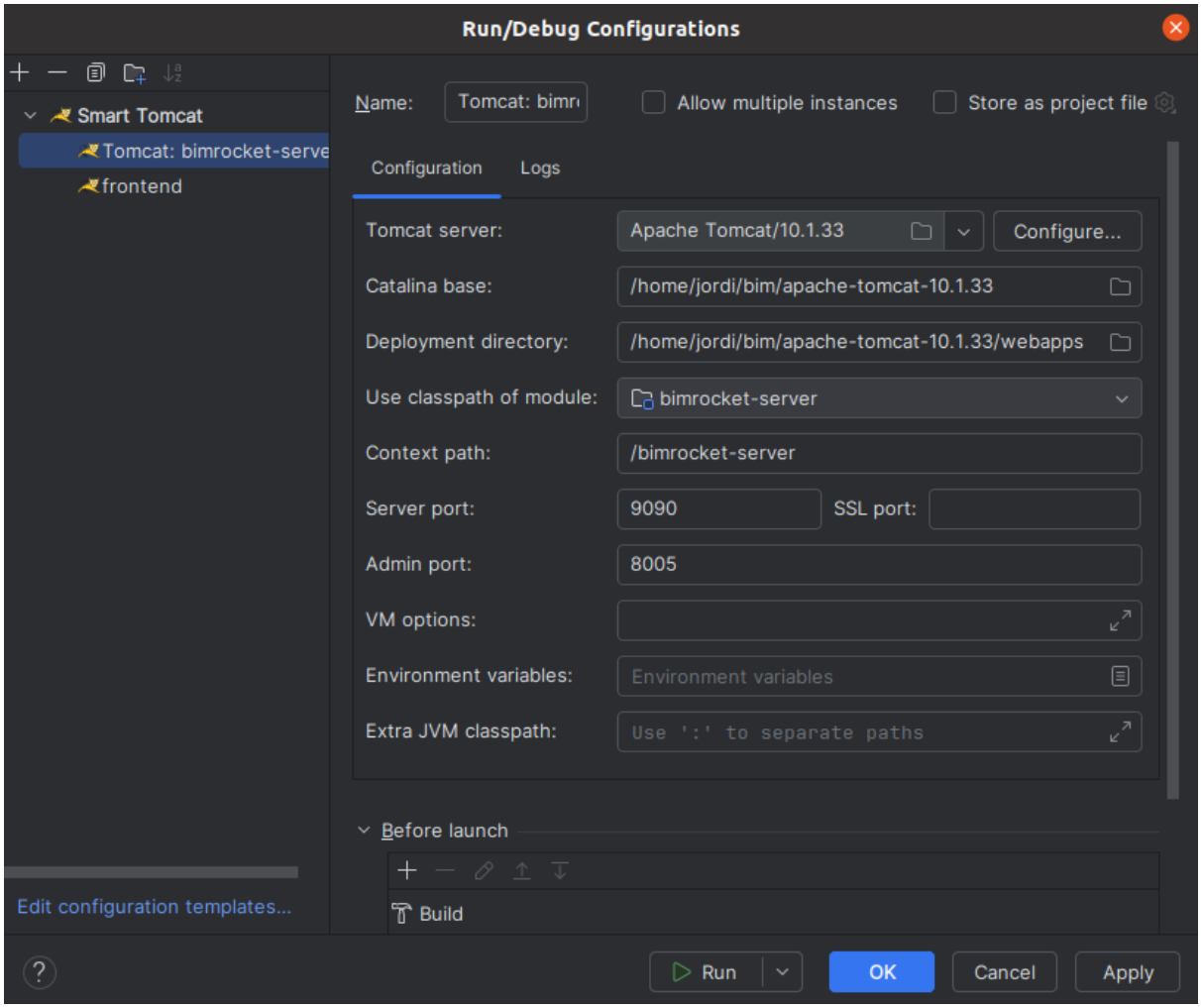
- Especificar el SDK a Platform Settings:



7) Crear Configuració per Executar i Debugar el projecte

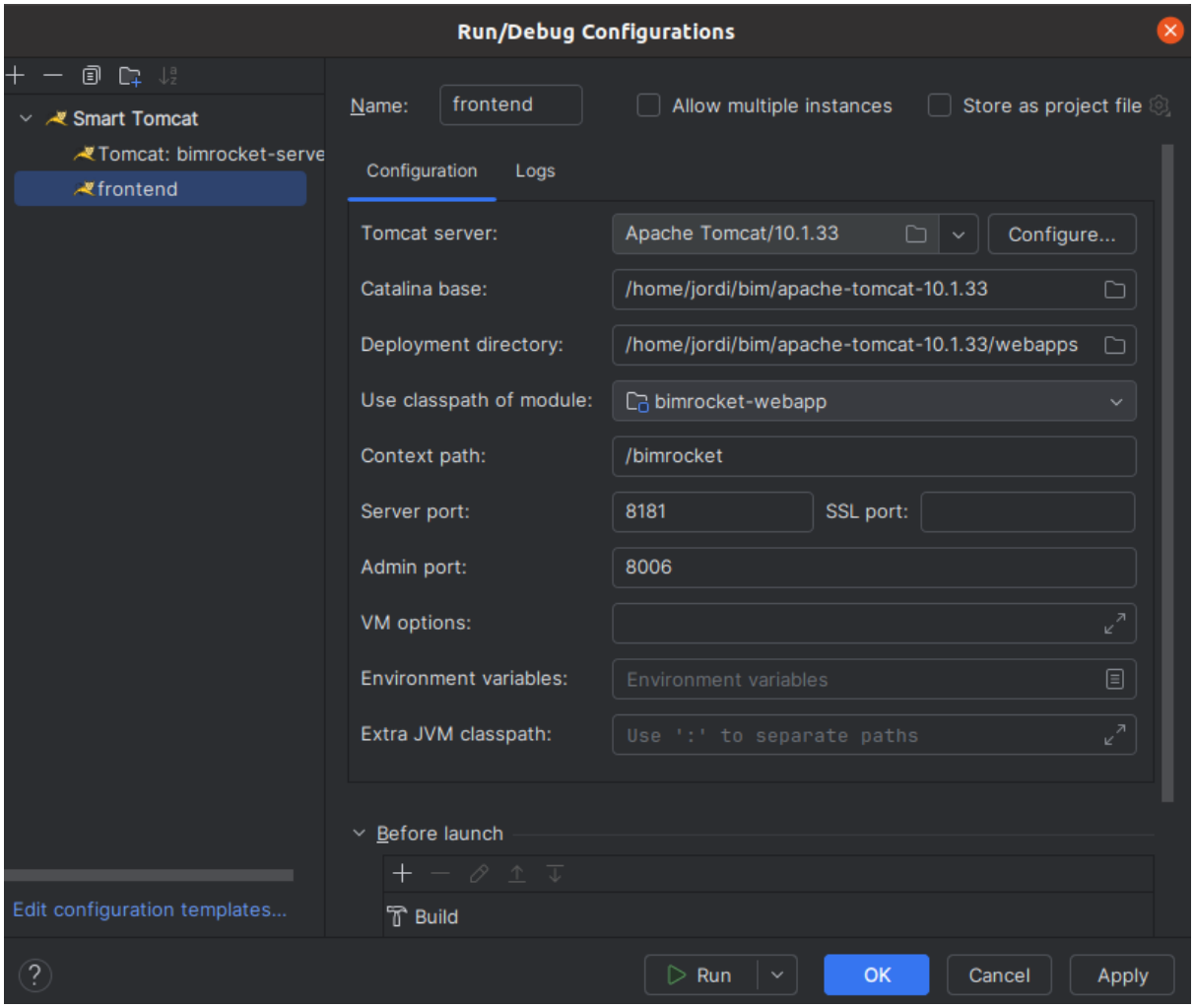


8) Especificar els paràmetres de configuració pel backend



- **Tomcat Server:** Configure… per anar a buscar el lloc on hem descarregat el servidor Apache Tomcat
- **Catalina base:** Path on hem descarregat el servidor Apache Tomcat
- **Deployment directory:** Path on es despleguen les apps en el servidor Apache Tomcat descarregat
- **Use classpath of module:** mòdul del backend en aquest cas
- **Context path:** mòdul del backend en aquest cas
- **Server port:** escollir port personalitzat

9) Especificar els paràmetres de configuració pel frontend



- **Tomcat Server:** Configure… per anar a buscar el lloc on hem descarregat el servidor Apache Tomcat
- **Catalina base:** Path on hem descarregat el servidor Apache Tomcat
- **Deployment directory:** Path on es despleguen les apps en el servidor Apache Tomcat descarregat
- **Use classpath of module:** mòdul del frontend en aquest cas
- **Context path:** mòdul del frontend en aquest cas
- **Server port:** escollir port personalitzat

Abans d'executar el projecte, comprovem que el fitxer d'environment.js del front-end està configurat segons els nostres requeriments. Com hem dit, per defecte aquest fitxer està preparat per a funcionar per a la versió no dockeritzada, per tant, només caldrà comprovar la part dels mòduls que vulguem carregar.

Execució o Debugging del projecte:

- Prèviament, cal generar el .war per recollir els canvis realitzats en el codi.
- *mvn clean install*
- El .war generat es guardarà en la carpeta **target** del projecte.
- Si volem que es copiï automàticament a la carpeta del servidor Apache Tomcat descarregat, afegirem al pom.xml del mòdul bimrocket-server i del bimrocket-webapp un plugin dins de la secció build i dins de la secció plugins al final de l'arxiu. S'ha d'afegir aquest bloc de codi:

```
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-antrun-plugin</artifactId>
 <version>3.1.0</version>
 <executions>
   <execution>
     <phase>package</phase> <!-- Es fa després de generar el WAR -->
     <goals>
       <goal>run</goal>
     </goals>
     <configuration>
       <target>
         <copy
file="${project.build.directory}/${project.build.finalName}.war"
               todir="/home/user/bim/apache-tomcat-10.1.33/webapps"
overwrite="true"/>
       </target>
     </configuration>
   </execution>
 </executions>
</plugin>
```

Nota: Recordar d'especificar a **todir** el path on despleguem les app del servidor Apache Tomcat descarregat.

Després de fer la compilació amb maven, es creen a la ruta **/home/<user>/bimrocket:**

- directori db
- bimrocket-server.yaml.
- directory cloudfs

El fitxer bimrocket-server.yaml es crea després del desplegament del war de backend en engegar el Tomcat. El procés de desplegament fa una còpia del fitxer application.yaml

bimnrocket/bimrocket-server/src/main/resources/application.yaml (Comunitat)
bimrocket-server/src/main/resources/application.yaml (Pilot)

en el nostre /home d'usuari

~/bimrocket/bimrocket-server.yaml

A la secció de databases del fitxer bimrocket-server.yaml es on s'especifica la configuració amb les dades de connexió a base de dades MongoDB allotjada en un contenidor Docker:

```
# database config
  bimdb:
    url: mongodb://root:mongodb@localhost:27017/bimdb?authSource=admin
  IFC2X3:
    url: mongodb://root:mongodb@localhost:27017/IFC2X3?authSource=admin
  IFC4:
    url: mongodb://root:mongodb@localhost:27017/IFC4?authSource=admin
```

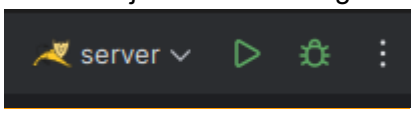
Per poder treballar amb la base de dades MongoDB, hem de desplegar un contenidor docker en local. Per això podem aprofitar el docker compose de la carpeta docker de bimrocket-server del projecte comentant en el fitxer el servei de backend i utilitzar només la part de servei mongodb.

```
mongodb:
  image: mongo:8.0.13
  container_name: mongodb
  restart: always
  ports:
    - "27017:27017"
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: mongodb
  volumes:
    - ../mongo_data/db:/data/db
  networks:
    - bimrocket_network
```

Un cop modificat el fitxer **bimrocket/bimrocket-server/docker/docker-compose.yml** aixequem el contenidor des de la carpeta docker amb:

docker-compose up -d

- Fem Run del server des de Intellij des de la configuració de projectes



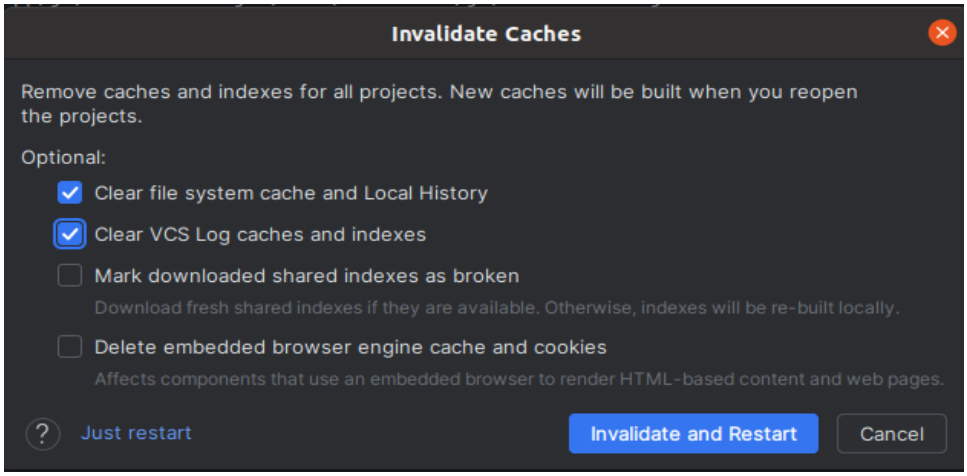
Solucions a problemes originats per inteliiJ

Sovint ens podem trobar amb problemes de sincronització o de caché amb el codi, especialment quan canviem de branch del repositori o quan canviem de projecte.

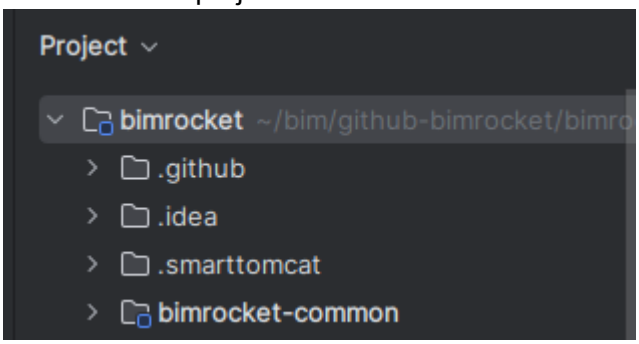
A intelliJ, fent clic sobre el símbol  apareix la barra de menú.

Per tal d'evitar-ho hi ha una sèrie d'accions:

- Invalidar caché
  Des de File → Invalidate caches



- Sincronitzar amb maven el projecte
  Seleccionar el projecte



  i amb botó dret del ratolí anar a buscar l'opció Maven, al final de la llista.
  Seleccionar Maven i Sync Project

- Rebuild del projecte
  Des de barra de menú, seleccionar Build i Rebuild Project

- Verificar que els pom.xml estiguin correctes

- Verificar la configuració de Run i Debugging del frontend i del backend

Generació de les classes IFC utilitzades pel Front-end treballant amb Visual Code

La versió descarregada del repositori de Bimrocket ja inclou les classes dels elements IFC necessàries per al front-end. No obstant això, si es produeix alguna modificació en aquestes classes, caldrà recompilar-les utilitzant una eina disponible al back-end i desar-les a la ubicació corresponent dins del codi del front-end.

El procés és el següent:

- Obtenir les classes en format Express des de la pàgina de standards.buildingsmart.org/IFC.
- Convertir aquestes classes a Javascript utilitzant l'eina IfcJavascriptClassesGenerator, que es troba al directori commons.

  bimrocket-common/src/main/java/org/bimrocket/generators/IfcJavascriptClassesGenerator.java
  bimrocket-common/src/main/java/org/bimrocket/generators/IfcJavascriptClassesGenerator.java

- Un cop generades, copiar les classes al directori del front-end a la ruta: *bimrocket-webapp/src/main/webapp/js/io/ifc/schemas*

Desplegament Front-end no-dockeritzat amb Visual Code

Desplegar amb Visual Studio Code:
Per fer desplegar i córrer el front-end en local, **una vegada compilat:**

1. localitzar els fitxers estàtics dintre de la carpeta “webapp” resultat de la compilació (i que es genera dintre de carpeta “Target”).
2. Seguidament, podem servir el front-end (contingut de la carpeta “webapp”) amb un servidor http:

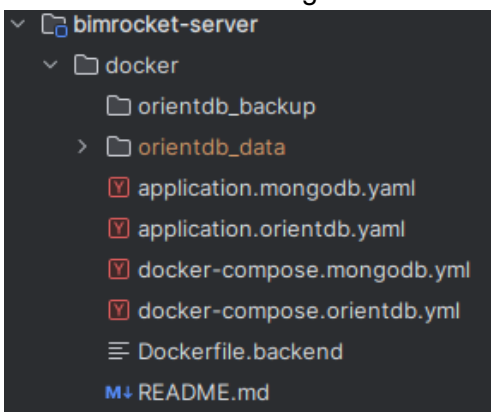
    1. Obre Visual Studio Code.
    2. Instal·la **Live Server** (si no el tens instal·lat). Ves a **Extensions** (menú de l’esquerra) i cerca **Live Server** i instal·la’l.
    3. Fes córrer l’arxiu HTML: app.html
       Cerca l’arxiu **app.html** en el VSCode que està en el següent directori: `bimrocket-webapp/src/main/webapp/app.html`

       Amb el botó dret clica sobre l’arxiu i selecciona **Open with Live Server.** Això iniciarà un servidor local a http://127.0.0.1:5500/bimrocket-webapp/src/main/webapp/app.html que carregarà el fitxer app.html, aquest permetrà al navegador carregar el fitxer Application.js i tots els altres recursos js.
       Una altra manera d’iniciar el **Live Server** és des de l’opció **Go Live** del menú inferior del VScode:



Desplegament del backend i frontend versió Comunitat (docker)

A la carpeta **bimrocket-server/docker** tenim la següent estructura:



- Primer de tot hem de compilar el projecte amb
  mvn clean install

- Un cop hem generat el .war, anem a construir la imatge de Docker. En aquest cas veiem que hi ha dos docker-compose, un per desplegar amb orientdb i l'altre amb mongo. El Dockerfile es comú per aquest dos docker-compose, però cadascun ja té el seu application.yml preparat per cada cas:

- Per desplegar amb mongo, docker-compose utilitzarà application-mongodb.yml. Això genera un contenidor per backend i un per mongo:

  docker-compose -f docker-compose.mongodb.yml up –build -d

- Per desplegar amb orientdb, docker-compose utilitzarà application-orientdb.yml. Això genera un contenidor per backend i un per orientdb:

  docker-compose -f docker-compose.orientdb.yml up –build -d

Els volums utilitzats pel bimrocket-server són :

```
- ./application.mongodb.yml:/root/bimrocket/bimrocket-server.yaml
- bimrocket_data:/root/bimrocket/cloudfs
```

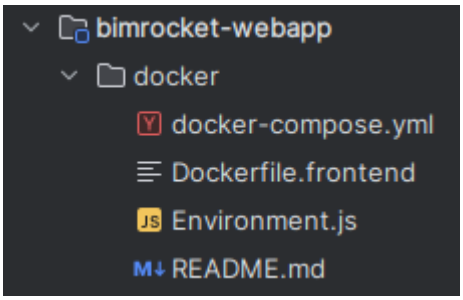
Per mongo:

```
- mongodb_databases:/data/db
- mongodb_config:/data/configdb
```

Per orientdb:

```
- orientdb_databases:/orientdb/databases
- orientdb_config:/orientdb/config
- orientdb_backup:/orientdb/backup
```

A la carpeta **bimrocket-webapp/docker** tenim la següent estructura:



Per desplegar el contenidor de backend:

docker-compose -f docker-composeyml up –build -d

Els volums utilitzats pel bimrocket-webapp són :

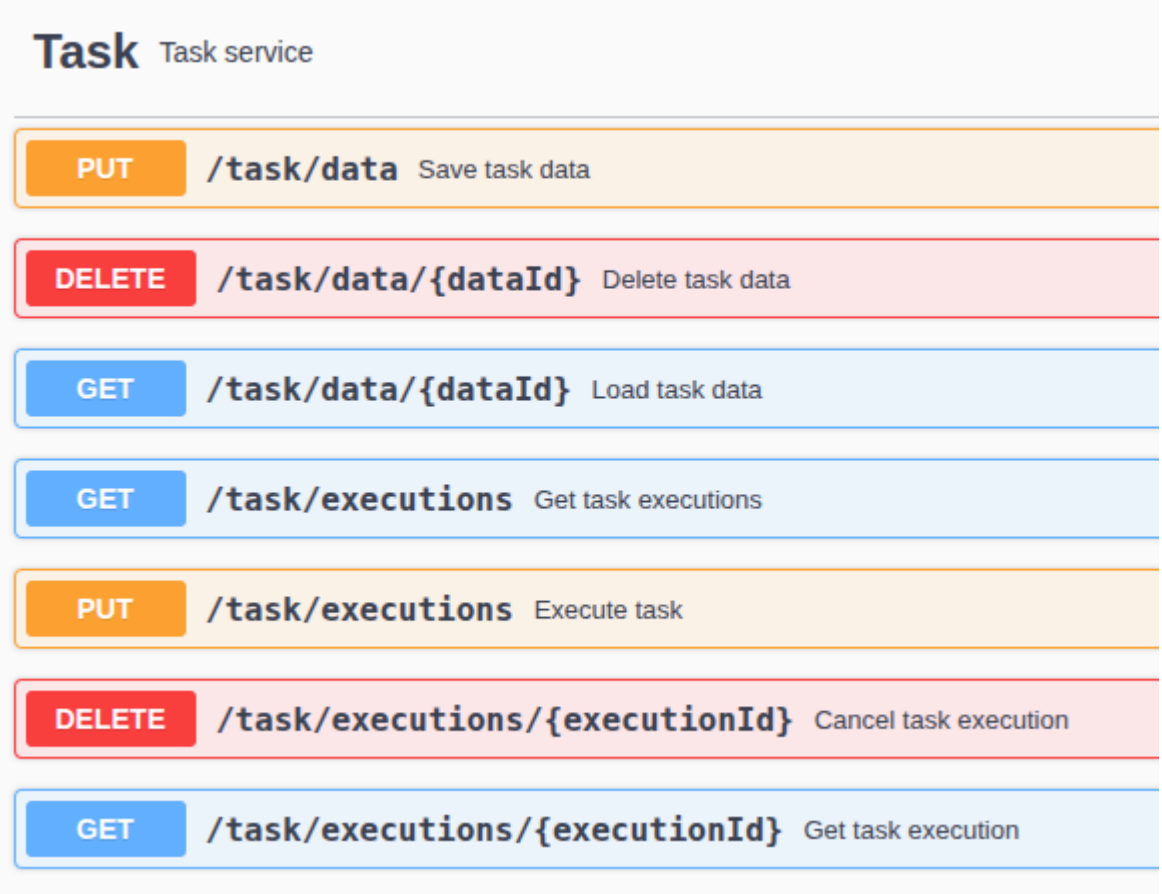
```
- ./Environment.js:/root/bimrocket/js/Environment.js
```

Anexe1: Serveis oferts

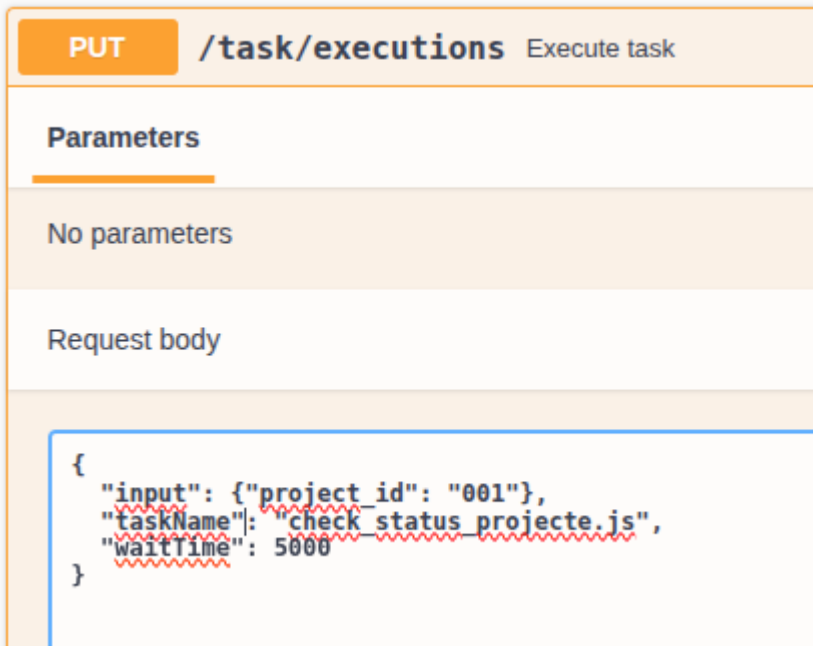
Scripts de Tasques

El backend, a través dels endpoints de Task ofereixen la possibilitat d'executar un tipus de scripts .js que no són exactament com els scripts convencionals, sino que s'executen de manera interna en el backend com si fossin classes Java i que permeten utilitzar els serveis oferts pel backend per tal d'executar accions tals com crear projectes BCF, comprovar el status d'aquest projectes, entre altres.

Els endpoints oferts pel backend són:



Els endpoints encarregats d'executar els scripts son els relacionats amb executions.
Per executar un script utilitzem el mètode PUT d'Executions, on s'envia un payload amb el script a utilitzar. Com a exemple, si volem comprovar el status d'un projecte BCF invocaríem una petició tal:



project_id seria l'id de projecte BCF, taskName el script en qüestió, i també podem especificar un temps d'espera de finalització amb waitTime.

Al repositori
https://gitlab.i2cat.net/areas/software-engineering-group/bim/bimrocket-development/bimrocket-data

A **cloudfs/tasks** podem veure la **llista de scripts** i un **README** amb les opcions i els payloads possibles per cada script.

Aquest scripts sovint utilitzen la funcionalitat de Task Data per guardar informació suplementaria. Internament, els endpoints de task data són cridats pels scripts d'executions i són invisibles de cara l'usuari.