UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Simulador de l'eficiència energètica de la BGRF

Treball de Fi de Grau (TFG)

Autor: Carles Cidraque Sala

Especialització: Enginyeria del Software

Director: Pau Fonseca i Casas

Juny 2021

Resum

L'eficiència energètica en els edificis és i serà un dels principals reptes als que ens haurem d'enfrontar en els pròxims anys a nivell mundial. El problema energètic dels edificis afecta tots els països de diferents maneres, no només a termes de contaminació o emissions atmosfèriques sinó també pel que fa a la preservació de fonts d'energia i l'ús racional d'aquesta mateixa.

De la necessitat de millora en la indústria de la construcció neix l'anomenada Construcció 4.0 concepte que engloba tots els canvis tecnològics relacionats amb la implementació de nous mètodes de treball relacionats amb processos, materials i mercats vinculats amb la construcció.

Aquí es on entra en joc l'eina NECADA, eina que s'utilitza conjuntament en aquest projecte. NECADA és un simulador que permet optimitzar tots els processos relacionats amb la construcció, amb l'objectiu de minimitzar els seus impactes ambientals, econòmics i socials durant totes les etapes del cicle de vida d'un edifici o zona urbana (disseny, construcció, ús, rehabilitació i deconstrucció).

Aquest projecte té l'objectiu de posar les bases per a obtenir dissenys òptims de consum energètic de l'edifici de la Biblioteca Rector Gabriel Ferraté (BRGF) de la UPC tot plantant les bases del model que representarà digitalment la biblioteca i de l'exploració i l'inici d'actualització de part del sistema NECADA tenint en compte les actualitzacions que pateix el motor de simulació que fa servir anomenat EnergyPlus. Aquestes dues tasques estan vinculades amb la preparació de l'anomenat escenari de simulació. Aquesta preparació de l'escenari, en aquest projecte, es la preparació prèvia necessària a l'execució de simulacions sobre un edifici o zona urbana en concret al sistema NECADA.

NECADA al formar part del sector de la Construcció 4.0, sector en naixement, constant millora i actualització, i al fet de també emprar sistemes que van actualitzant-se també ha d'anar adaptant-se a les modificacions que fan aquests sistemes utilitzats.

La primera tasca va vinculada amb l'obtenció de l'anomenat *Building Information Model* (BIM) de la BRGF. NECADA en l'actualitat suporta únicament fitxers BIM en format IDF, però EnergyPlus en la actualitat ja accepta un altre tipus de format anomenat epJSON el qual acabarà sent el dominant i l'únic que acabarà sent suportat per aquest motor de simulació d'eficiència energètica. Per tant, en aquest projecte per tal d'estar al dia amb EnergyPlus, es planten les bases del BIM de la biblioteca en format de fitxer IDF com en epJSON (JSON), els dos formats que accepta Energy Plus en l'actualitat tot explorant diferents camins per aconseguir-ho, camins que en el cas de l'obtenció del BIM en epJSON pot servir per actualitzar projectes legacy en format IDF a NECADA perquè aquest sistema es pugui actualitzar a la última versió d'EnergyPlus sense perdre suport als antics projectes.

La segona gran tasca del projecte detecta, explora i dona solució a la necessitat d'adaptar el sistema NECADA per tal de permetre l'entrada i la vinculada importació i validació d'aquest nou tipus de fitxer en format epJSON representant el BIM d'un edifici o zona urbana tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou format de fitxer. Per donar solució a NECADA a aquesta modificació de canvi de format de fitxer d'entrada en aquest projecte es realitza una aplicació de consola de tipus .NET Framework amb Microsoft Visual Studio. L'aplicació de consola que s'ha desenvolupat en aquest projecte te la funció de poder importar la informació que interessa a NECADA a la seva base de dades en dos formats diferents de fitxer d'entrada: IDF i epJSON. L'objectiu final d'aquesta aplicació de consola es que el seu codi es pugui incorporar a NECADA fent els menors canvis possibles. Aquesta manera de procedir on primer es crea una aplicació de consola per després poder adaptar el codi a NECADA ja s'ha efectuat en altres projectes vinculats a aquest sistema per tal d'eliminar complexitat en les primeres fases de desenvolupament.

Abstract

Energy efficiency in buildings is and will be one of the main challenges we will face in the coming years worldwide. The energy problem of buildings affects all countries in different ways, not only in terms of pollution but also in terms of preserving energy sources and the rational use of it.

From the need for improvement in the construction industry arises the so-called Construction 4.0 concept that encompasses all technological changes related to the implementation of new working methods related to processes, materials and markets related to construction.

This is where NECADA tool comes into play, a tool that is used together in this project. NECADA is a simulator that allows to optimize all the processes related to the construction, with the aim of minimizing its environmental, economic and social impacts during all the stages of the life cycle of a building or urban area (design, construction, use, rehabilitation and deconstruction).

This project aims to lay the foundations for obtaining optimal energy consumption designs for the UPC Biblioteca Rector Gabriel Ferraté (BRGF) building by laying the bases for the model that will digitally represent the library and for exploration and the start of upgrading part of the NECADA system taking into account the upgrades undergone by the simulation engine used called EnergyPlus. These two tasks are linked to the preparation of the so-called simulation scenario. This preparation of the scenario, in this project, is the necessary previous preparation for the execution of simulations on a specific building or urban area in the NECADA system.

NECADA, being part of the Construction 4.0 sector, a sector in its infancy, constantly improving and updating, and the fact that it also uses systems that are being updated, must also adapt to the modifications made by these systems used.

The first task is related to obtaining the so-called Building Information Model (BIM) of the BRGF. NECADA currently only supports BIM files in IDF format, but EnergyPlus currently already accepts another type of format called epJSON which will end up being the dominant and the only one that will end up being supported by this energy efficiency simulation engine. Therefore, in this project, in order to keep up to date with EnergyPlus, the bases of the library's BIM are laid out in IDF file format as well as in epJSON (JSON), the two formats that Energy Plus currently accepts while exploring different ways to achieve this. Paths that in the case of obtaining BIM in epJSON can be used to update legacy projects in IDF format of NECADA so that this system can be updated to the latest version of EnergyPlus without losing support for old projects that are in IDF files.

The second major task of the project detects, explores and solves the need to adapt the NECADA system in order to allow the entry and the related import and validation of this new file type in epJSON format representing the BIM of a building or urban area, considering that you we will also need to manage the export of changes to this new file format. A .NET Framework console application with Microsoft Visual Studio is implemented in this project to address this change in the input file format. The console application that has been developed in this project has the function of being able to import the information that NECADA is interested in into its database in two different input file formats: IDF and epJSON. The ultimate goal of this console application is that its code can be incorporated into NECADA by making as few changes as possible. This way of proceeding where a console application is first created and then the code can be adapted to NECADA has already been carried out in other projects related to this system in order to eliminate complexity in the early stages of development.

Índex

1. Introducció

Aquest es un Treball de Fi de Grau (TFG) titulat amb el nom de "Simulador de l'eficiència energètica de la BRGF". Pertany a l'especialitat d'Enginyeria del Software del Grau d'Enginyeria Informàtica (GEI) de la Facultat d'Informàtica de Barcelona (FIB) de la Universitat Politècnica de Catalunya (UPC).

L'autor d'aquest TFG es en Carles Cidraque Sala i el director de projecte es en Pau Fonseca i Casas.

1.1. Motivació

L'eficiència energètica en els edificis és i serà un dels principals reptes als que ens haurem d'enfrontar en els pròxims anys a nivell mundial (1). El problema energètic dels edificis afecta tots els països de diferents maneres, no només a termes de contaminació o emissions atmosfèriques sinó també pel que fa a la preservació de fonts d 'energia i l'ús racional d'aquesta mateixa.

Un article de El Periódico De La Energía (2) recollia, segons un estudi de l'any 2018 realitzat per l'EuroACE (3) (Aliança europea de companyies per l'eficiència energètica en els edificis), que el 84% dels edificis a Espanya son energèticament ineficients, és a dir, consumeixen més energia de la necessària. Segons aquest mateix estudi, es constata que a nivell europeu, son ineficients entre el 70% i el 90% dels edificis i també apunta que en tres anys seguirien existint el 90% d'aquests edificis. De no prendre mesures i no millorar la seva eficiència, l'Agència Internacional de l'Energia (4) estima que la demanda global d'energia augmentarà un 50% abans de 2050.

A Espanya, els edificis són actualment responsables del 31% del consum de l'energia. Una realitat que repercuteix directament en a l'economia del ciutadà, ja que l'eficiència energètica va estretament lligada al seu estalvi en el consum.

Els edificis sempre han estat fonamentals en les nostres vides, al cap i a la fi passem gran part d'aquestes en interiors i encara més amb la situació viscuda del confinament per la COVID-19 que ha posat en relleu més que mai el paper vital que té la renovació d'aquest edificis.

Com a resultat, diverses iniciatives, regulacions i diversos programes nacionals i supranacionals han començat a florir en els últims anys en el sector privat com es el cas de LEED (5), CASBEE (6) i d'altres; definint estàndards i paràmetres per avaluar el nivell de sostenibilitat dels edificis i reduir el seu ús d'energia.

Accions concretes i ambicioses per combatre el consum innecessari d'energia en els edificis han de ser la prioritat absoluta de la política energètica europea i mundial com ho és el compliment i seguiment de la Directiva (UE) 2018/844 (7) relativa a l'eficiència energètica dels edificis.

En els darrers anys, l'edificació *zero carbon building* (ZCB) ha sorgit com un enfocament innovador per reduir els efectes negatius de les emissions de CO2 relacionades amb el sector de la construcció. Tot i que la definició de ZCB a vegades pot deixar espai per a la interpretació, es sol referir un ZCB com un edifici on les emissions de carboni generades a partir de l'ús de combustibles fòssils in situ o fora del lloc s'equilibren amb la quantitat de producció d'energia renovable in situ; aquests també son anomenats *Zero Energy Buildings* (ZEB) o *Net Zero Energy Buildings* (NZEB).

La necessitat d'optimitzar les prestacions de l'edifici condueix a un canvi en la manera de dissenyar-los, canviant un disseny tradicional de diversos passos cap a un disseny integrat (8). Per tant, experts en diferents camps ara participen en el disseny de l'edifici i tant la intuïció com les eines tradicionals ja no són suficients per assolir les prestacions requerides (9).

A la base d'un procés de disseny exitós hi ha la necessitat de que l'equip de disseny tingui tota la informació necessària disponible, i això esdevé cada vegada més important a l'hora d'aconseguir un alt nivell d'objectius en els edificis, com es el cas de ZCB o nZEB. A l'hora d'avaluar el rendiment

energètic dels edificis d'alta eficiència energètica ja no és possible avaluar els components individuals de forma aïllada i les eines tradicionals es tornen inadequades i en molts casos obsoletes, cosa que comporta la necessitat de noves eines més avançades com ho son els softwares per aplicar l'anomenat *Building Performance Simulation* (BPS).

Les BPS poden ajudar a reduir l'emissió de gasos d'efecte hivernacle i a proporcionar millores substancials en els nivells de confort i consum de combustible, tractant els edificis i els seus sistemes tèrmics com a entitats completament optimitzades i no com la suma d'un nombre de subsistemes o components dissenyats i optimitzats per separat (10).

Aquí es on entra en joc l'eina NECADA (11), eina que s'utilitza conjuntament en aquest projecte. NECADA és un simulador que permet optimitzar tots els processos relacionats amb la construcció, amb l'objectiu de minimitzar els seus impactes ambientals, econòmics i socials durant totes les etapes del cicle de vida d'un edifici o zona urbana (disseny, construcció, ús, rehabilitació i deconstrucció).

A mode de resum aquest projecte te l'objectiu de posar les bases per a obtenir dissenys òptims de consum energètic de l'edifici de la Biblioteca Rector Gabriel Ferraté (BRGF) de la UPC tot plantant les bases del model que representarà digitalment la biblioteca i de l'exploració i l'inici d'actualització de part del sistema NECADA tenint en compte les actualitzacions que pateix el motor de simulació que fa servir.

Aquestes dues tasques descrites detalladament al llarg del document estan vinculades amb la preparació de l'anomenat escenari de simulació. Aquesta preparació de l'escenari, en aquest projecte, es la preparació prèvia necessària a l'execució de simulacions sobre un edifici o zona urbana en concret al sistema NECADA.

La primera tasca va vinculada amb l'obtenció de l'anomenat *Building Information Model* (BIM) de la BRGF tant en format de fitxer IDF com en epJSON (JSON).

Com ja s'ha mencionat es un sector on les tecnologies emprades estan en ple naixement i desenvolupament, per tant en constant millora i plena de modificacions i actualitzacions. NECADA al formar part d'aquest sector i al fet de també emprar sistemes que van actualitzant-se també ha d'anar adaptant-se a les modificacions que fan aquests sistemes utilitzats.

Aquí es on entra la segona gran tasca del projecte on s'adapta el sistema NECADA per permetre l'entrada i la vinculada importació i validació d'un nou tipus de fitxer en format epJSON (JSON) representant el BIM d'un edifici o zona urbana tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou fitxer a NECADA.

1.2. Context

Tal i com s'ha mencionat en l'apartat 1.1 aquest TFG està vinculat amb el projecte NECADA. NECADA es un projecte de recerca de la UPC amb el producte en producció. La startup Polyhedra Tech (12) és qui manté el producte.

Polyhedra Tech es una startup tecnològica sorgida de la UPC BarcelonaTech, que té la gestió i disseny, les noves tecnologies de la Industria 4.0 i Construcció 4.0, la sostenibilitat i l'arquitectura com a àrees d'expertesa.

En els següents subapartats es contextualitza l'entorn en el que està situat aquest projecte tot definint els seus termes clau. Aquesta contextualització i ubicació del projecte dins l'entorn en el que es desenvolupa es necessària per tal de que el mateix projecte i els seus objectius cobrin sentit.

1.2.1. Industria 4.0 i Construcció 4.0

NECADA es un sistema que forma part del domini de l'anomenada Construcció 4.0. La Construcció 4.0 no es va originar de forma aïllada, aquest concepte va adoptar el marc teòric d'un concepte més ampli: la Indústria 4.0.

Coneguda com la "Quarta Revolució Industrial", la Industria 4.0 ha guanyat fama en els darrers anys, ja que pot influir en grans indústries a nivell de disseny i fabricació, definint el futur de la productivitat i el creixement de les indústries. Aquest terme es va originar a Alemanya quan va presentar el concepte d'Indústria 4.0 el 2011 com a part de la seva estratègia d'alta tecnologia. Atès que el concepte és bastant nou, encara no hi ha acord sobre una definició comuna, alguns professionals han assenyalat la necessitat d'una definició dinàmica i completa, que descriu aquesta quarta revolució industrial com un "canvi en la lògica de fabricació cap a un enfocament cada vegada més descentralitzat i autoregulador de la creació de valor, habilitat per conceptes i tecnologies com els *cyber-physical systems* (CPS), *Internet of Things* (IoT), *Internet of Services* (IoS), *cloud computing* o les *smart factories* " (13).

La indústria 4.0 facilita la connexió d'informació, objectes i persones, ja que crea escenaris de fabricació física i virtual que permeten a les fàbriques transformar el seu entorn en un de fabricació intel·ligent. La indústria 4.0 també busca una automatització integrada (14) caracteritzada per la seva dependència de l'ús de sistemes informàtics (15).

La indústria de la construcció també s'ha beneficiat d'aquest progrés, donant lloc al terme de Construcció 4.0, que ha guanyat popularitat durant els darrers anys. Esmentat per primera vegada el 2016 per Roland Berger (16), aquest concepte es basa principalment en la consciència de les empreses constructores sobre la digitalització de la indústria de la construcció englobant quatre conceptes clau: dades digitals, automatització, connectivitat i accés digital.

La definició de Construcció 4.0 deriva de la fundació de la Indústria 4.0, però es centra i es relaciona amb el sector de la construcció. Per tant, Construcció 4.0 són tots els canvis tecnològics relacionats amb la implementació de nous mètodes de treball relacionats amb processos, materials i mercats vinculats amb la construcció. Una de les possibles classificacions que permetrien caracteritzar el concepte de construcció 4.0 es basant-la principalment en dos pilars: la digitalització de la indústria de la construcció i la industrialització dels processos de construcció (*Figura 1).*

FIGURA 1. DIAGRAMA AMB ELS DOS PILARS PRINCIPALS QUE CONSTITUEIXEN LA CONSTRUCCIÓ 4.0 **(17)**

El primer pilar referent a la digitalització de la indústria de la construcció abasta el camp de la gestió de dades en forma digital mitjançant l'ús d'Internet i software (18). Invertir en noves tecnologies produeix millores en la productivitat, que és el que més busquen les indústries, com en aquest cas la de la construcció. En aquesta classificació es on NECADA centra més l'atenció i es en la temàtica de la Construcció 4.0 en que es centra aquest TFG.

D'altra banda, la industrialització dels processos de construcció es centra en la construcció automatitzada, que consisteix en un nou conjunt de tecnologies i processos que canvien la manera com es concep la indústria de la construcció (19), on la fabricació digital pren un paper important en la productivitat dels processos de construcció (20).

La Figura 2 mostra els principis de disseny de la Construcció 4.0. El concepte d'Indústria 4.0 es pot combinar bé en la perspectiva de les indústries de la construcció. Tal com exploren Oesterreich i Teuteberg (21), els principals components de la indústria 4.0 com ara IoT, IoS, CPS, Big Data, Cloud

Computing, Robòtica, etc. les seves aplicacions pràctiques de la Construcció 4.0 encara estan en fase inicial.

Les subseccions següents descriuen els principis de disseny per entendre l'estat actual de l'art de la construcció 4.0:

- La idea de la digitalització tracta de tecnologies de comunicació que permeten que els objectes interconnectats i les persones comparteixin informació per assolir els objectius comuns de manera col·laborativa (22). S'ha trobat que el *Building Information Modelling* (BIM) és el mètode de planificació digital destacat en la construcció (21). Juntament amb IoT i IoP (Internet of People), BIM es pot utilitzar per establir els estàndards de comunicació comuns entre els diferents participants del projecte. Tot i que hi ha una manca d'esquema de dades estandarditzat a BIM i una manca de protocol que defineixi la responsabilitat de l'ús d'informació (23), buildingSMART treballa per aconseguir estàndards interoperables de BIM.

- Les dades que es recullen per mitja de sensors i els models digitalitzats s'uneixen entre si per crear una còpia virtual del món físic que ajuda a prendre decisions informades (22). En el context de la construcció, la RV (Realitat Virtual) i l'AR (Realitat Augmentada) són cada vegada més populars (24) i poden minimitzar la bretxa entre el digital (per exemple, el model 3D, el model de simulació, etc.) i el món real (posició o estat dels materials, equips).

- La presa de decisions descentralitzada implica la interconnexió d'objectes i persones utilitzant informació per prendre una millor decisió i millorar la productivitat (22). Els projectes de construcció sovint requereixen la col·laboració entre diferents parts ubicades geogràficament en llocs diversos i de diferents perfils professionals. A més, pot existir la possibilitat de que diferents parts utilitzin diferents plataformes BIM per als seus models que creen problemes d'interoperabilitat (25). Hi ha un interès creixent per utilitzar les tecnologies del núvol com a mitjà de col·laboració en equip durant tot el cicle de vida d'un edifici en un entorn BIM. I els investigadors han estat investigant per desenvolupar especificacions per a una plataforma BIM basada en el núvol (26).

- En un entorn d'indústria 4.0, el paper dels humans està canviant cap a un decisor estratègic en lloc de l'operador de màquines (22). Els treballs insegurs, desagradables i exhaustius són ajudats per robots per als quals cal formar adequadament els humans per a una col·laboració eficaç. Com que és una indústria que requereix molta mà d'obra, la indústria de la construcció té un gran potencial per millorar la seva productivitat mitjançant l'assistència tècnica (per exemple, l'ús de robots), especialment per a treballs perillosos i no segurs per a un ésser humà. A la plataforma de construcció digital, hi ha usos limitats de robots com ara impressió 3D, fabricació de parets, col·locació de barres, soldadura, drons, etc. i tots ells són robots d'una sola aplicació. S'hauria d'investigar l'ús de robots multifuncionals i múltiples tenint en compte les condicions dinàmiques del lloc en els projectes de construcció.

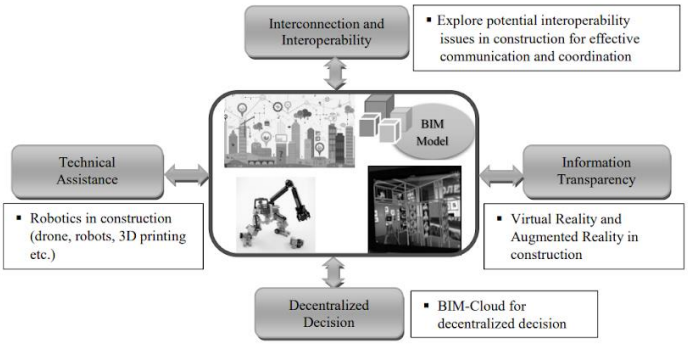


FIGURA 2. PRINCIPIS DE DISSENY DE LA CONSTRUCCIÓ 4.0 (17)

1.2.1.1. Modelatge energètic basat en BIM

Una de les maneres efectives d'aconseguir l'eficiència energètica dels edificis és utilitzar la tecnologia de modelització energètica dels edificis (BEM), que té com a objectiu avaluar dissenys alternatius, la comparació i selecció de sistemes i subsistemes, l'assignació de pressupostos energètics anuals, l'assoliment del compliment dels estàndards energètics i l'optimització econòmica durant el procés de disseny de l'edifici. Tanmateix, BEM no està prou integrat i sincronitzat amb el procés de disseny i planificació digital, com a conseqüència les estratègies de disseny energèticament eficients no s'implementen bé en la fase inicial de disseny. A més, BEM encara no es beneficia del flux continu d'informació en la modelització digital. Per exemple, la informació relacionada amb BEM s'ha de tornar a introduir manualment a les eines BEM (eines de modelatge energètic), que es considera que consumeix temps, és costosa i requereix molts professionals, tot i que aquesta informació ja està disponible en models de disseny digital (per exemple, BIM).

Els Associated General Contractors of America (AGC) van proporcionar una definició del ja presentat Building Information Modeling (BIM) on defineixen el terme com "el desenvolupament i ús d'un model software per simular la construcció i el funcionament d'una instal·lació". El model resultant, un BIM (Building Information Model), és una representació digital rica en dades, orientada a objectes, intel·ligent i paramètrica de la instal·lació, que pot proporcionar dades adequades a les necessitats de diversos usuaris. BIM també es pot utilitzar per extreure i analitzar informació per ajudar els dissenyadors a prendre decisions i millorar el procés d'anàlisi i millora de la instal·lació (27). Basat en la tecnologia BIM, en els darrers anys ha sorgit un nou enfocament, anomenat Modelatge d'Energia d'Edificis basat en el Modelatge d'Informació d'Edificis (BEM, basat en BIM). El BEM basat en BIM utilitza el model BIM predissenyat (que inclou informació sobre el disseny arquitectònic i les càrregues mecàniques, les propietats dels materials i el sistema HVAC (Calefacció, Ventilació i Aire Condicionat)) per crear l'entrada per a les eines BEM i proporciona l'oportunitat de fer de BEM un estalvi de temps, procés econòmic, senzill, més pràctic, coherent i precís. Els dissenyadors poden utilitzar l'enfocament del BEM basat en BIM per avaluar les opcions de disseny i prendre decisions de disseny de manera eficient durant el procés de disseny de l'edifici, així fent que l'objectiu d'eficiència energètica de l'edifici sigui més fàcil d'aconseguir.

El concepte de BIM es va presentar pel projecte de Building Description System per Chuck Eastman el 1974, i va donar la idea que una descripció basada en ordinador d'un edifici ajudava a realitzar la visualització i anàlisi quantitativa d'un projecte d'edifici (28) . L'acrònim "BIM" s'utilitza per representar tant el model d'informació de l'edifici (el model) com el modelatge de la informació de l'edifici (el procés), i són utilitzats alternativament per la comunitat investigadora i els desenvolupadors de software. La diferència entre el model d'informació d'edificis i el modelatge d'informació d'edificis és que el primer representa una representació digital rica en dades, orientada a objectes, intel·ligent i paramètrica d'una instal·lació; en canvi, el modelatge d'informació d'edificis és un procés que implica la generació i la gestió (desenvolupament i ús) de la representació digital d'una característica física i funcional d'una instal·lació durant el seu cicle de vida. La tecnologia BIM prové de la tècnica de modelatge paramètric orientat a objectes. El terme "paramètric" descriu un

procés mitjançant el qual es modifica un element i s'ajusta automàticament un element o conjunt adjacent o que depèn (com ara una porta fixada a una paret) per mantenir una relació prèviament establerta (27).

Pel que fa al terme BEM ja mencionat, aquest és un potent mètode informatitzat per investigar el rendiment dels edificis i avaluar el disseny arquitectònic i mecànic. Permet a l'equip de disseny avaluar els impactes ambientals de diverses opcions de disseny i, per tant, ofereix un disseny d'edifici optimitzat. També ajuda a explorar problemes de disseny complicats. Pot calcular les càrregues de l'edifici i el consum d'energia per determinar les característiques energètiques de l'edifici i dels sistemes, i les càrregues màximes de disseny per a l'equipament i la mida de la planta, també pot avaluar el rendiment de la llum del dia de l'edifici.

1.2.1.2. Canvi en el procés de disseny d'un edifici o zona urbana

En general, les fases del procés de disseny de l'edifici es poden dividir en: disseny conceptual, disseny preliminar, disseny desenvolupat i disseny detallat. Abans de l'etapa de disseny conceptual, es dur a terme una reunió d'inici del projecte per identificar els requisits de l'edifici mitjançant la consulta amb les parts interessades. Generalment, els objectius del projecte i els requisits de disseny (desenvolupament del lloc, orientació de l'edifici, massificació i forma, serveis inicials de construcció i sistema d'estructures) es determinen en la fase de concepte. El disseny preliminar es centra en el desenvolupament dels detalls del disseny conceptual inicial, inclosa la planificació inicial del disseny, factors externs de l'edifici (amb consideració de la il·luminació natural, el rendiment tèrmic i energètic), els sistemes d'estructures, la il·luminació, el disseny d'acústica i confort tèrmic, les opcions de climatització, aigua i sistemes d'aigües residuals, una estratègia de seguretat contra incendis i la selecció de materials. Els esforços preliminars de modelatge tèrmic, de llum natural i energètica comencen a participar en el disseny preliminar. L'etapa de disseny desenvolupada desenvolupa encara més el disseny conceptual dissenyat a l'etapa anterior. Els detalls dels factors externs de l'edifici, la disposició final de l'espai, la integració de sistemes de climatització i ventilació, la integració del disseny estructural, la integració de sistemes elèctrics, la integració de sistemes hidràulics, la integració de seguretat contra incendis i la selecció final de materials es determinen durant el disseny desenvolupat. El disseny i els plans de construcció actualitzats i els requisits i protocols per a la documentació de la construcció han de preparar-se en l'etapa de disseny desenvolupada. Els documents generats en aquesta etapa inclouen plànols i especificacions de consentiment i licitació per a cada disciplina i l'informe de l'expedient. Els documents s'han de coordinar entre totes les disciplines, en cas contrari, el procés de licitació i construcció es veuria influït negativament.

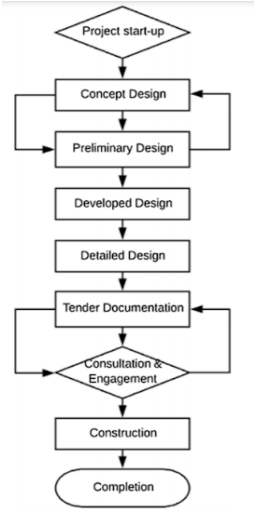


FIGURA 3. PROCÉS DE DISSENY CONVENCIONAL D'EDIFICIS I ZONES URBANES (29)

La majoria de les empreses de disseny han estat seguint en el passat el procés anterior, anomenat procés de disseny convencional, tal com es mostra a la Figura 3. Aquest té una estructura lineal a

causa de les successives contribucions dels dissenyadors a l'equip del projecte. No obstant això, aquest procés de disseny va provocar que els membres de l'equip de disseny tinguessin poca interacció entre ells i treballessin de manera segregada, i els seus pics de treball es produïssin durant l'etapa de disseny detallada i es reduïssin durant les etapes posteriors. Aquest procés de disseny sol començar amb un concepte de disseny acordat per l'arquitecte i el client. Basant-se en aquest concepte de disseny, es demana als enginyers (principalment enginyers civils, d'estructura, mecànics i elèctrics) que implementin el disseny i suggereixin els sistemes adequats (30). D'aquesta manera, els enginyers solen proporcionar aportacions no òptimes per a l'eficiència energètica, perquè estan estretament lligats pel concepte de disseny acordat per l'arquitecte i el client. A més, la simulació d'edificis, com ara BEM, no sol participar en processos de disseny anteriors a causa de les limitacions pressupostàries i de temps, i de les limitacions d'entrada disponible per al modelatge energètic. Com a resultat, el mal rendiment i els alts costos operatius es produeixen amb més freqüència. Tot i que els enginyers poden suggerir sistemes avançats i d'alt rendiment amb l'ajut de la simulació d'edificis en una etapa de disseny posterior, la introducció de sistemes d'alt rendiment en aquesta etapa pot no superar els defectes imposats per les males decisions inicials de disseny.

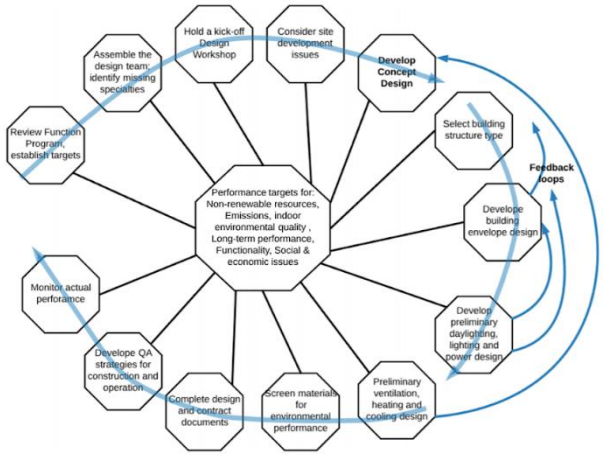


FIGURA 4. PROCÉS DE DISSENY INTEGRAT (IDP) D'EDIFICIS I ZONES URBANES (29)

Per superar els problemes del procés de disseny convencional, es va introduir el procés de disseny integrat (IDP) tal com es mostra a la Figura 4. Es constata que IDP no conté cap element nou radical, dues millores importants que proporciona el IDP respecte el disseny convencional són el desplaçament del volum de treball cap a les primeres fases de disseny i una sèrie de bucles d'activitat iteratius en el procés de disseny de l'edifici. A més de les tasques habituals, es reuneix l'equip de disseny multidisciplinari, s'identifiquen les especialitats que falten i es prepara i es realitza la reunió de disseny inicial abans del disseny conceptual. Durant el disseny de l'edifici, les decisions de disseny de l'estructura de l'edifici, els factors externs de l'edifici, el sistema de climatització, il·luminació, il·luminació natural, etc. no es prenen linealment. Aquestes decisions de disseny es determinen mitjançant una sèrie de bucles de retroalimentació (els treballs col·laboratius de l'equip de disseny multidisciplinari). A més, l'equip de disseny considera els objectius de rendiment durant tot el procés de disseny. Tots els membres de l'equip de disseny multidisciplinari, que solen incloure arquitectes, enginyers estructurals, civils, mecànics i elèctrics especialistes en energia, treballen de manera cooperativa en cada bucle d'activitats per crear decisions òptimes. En el procés de disseny integrat, l'arquitecte es converteix en un líder d'equip per incorporar i conciliar les aportacions d'altres dissenyadors i s'espera que els enginyers tinguin la iniciativa de proporcionar idees sobre el concepte de disseny en la fase inicial del disseny, l'especialista en els costos de construcció i contractes ja no només fa càlculs de costos de construcció, sinó que també realitza anàlisis i avaluació del cicle de vida de materials i incorpora altres sistemes tecnològics al disseny. A més, les simulacions computacionals no només s'utilitzen a la fase final del disseny amb finalitats de verificació i presentació, sinó que també funcionen com a eina assistida pel disseny per ajudar l'equip a comparar diverses opcions per obtenir decisions de disseny optimitzades (31). En resum, IDP es pot caracteritzar com una estructura iterativa oposada a una estructura lineal, un mètode flexible, un enfocament no predeterminat i un procés iteratiu amb aprenentatge continu, no una seqüència

d'esdeveniments preordenada. Per al procés de disseny integrat, BIM es converteix, per tant, en una tecnologia essencial per donar suport als IDP. El que permet BIM es integrar les disciplines fragmentades d'arquitectura, enginyeria i construcció i optimitzar el rendiment del cicle de vida dels edificis (32).

Aquest nou enfocament ja s'ha implementat amb molt èxit en molts altres sectors com el del propi desenvolupament de software on es va passar d'una metodologia de desenvolupament en cascada on calia revisar i verificar les tasques especialitzades completades en una fase abans de passar a la següent fase, per tant es un enfocament lineal i seqüencial cap a un desenvolupament amb la metodologia *agile* on aquest és un tipus de model incremental de desenvolupament de software basat en principis que es centra més en les persones, els resultats, la col·laboració i les respostes flexibles als canvis, en lloc de planificar tot el projecte a l'inici sense tenir gran marge de maniobra, es desglossa el treball de desenvolupament en petits increments completats en iteracions o en períodes de temps curts. Els beneficis del IDP es poden concloure en dos aspectes, un és la millora del rendiment ambiental, l'altre és la millora del programa funcional, en la secció de sistemes estructurals i en l'expressió arquitectònica a causa d'un debat interdisciplinari obert i un enfocament sinèrgic, elements que defineixen la Industria 4.0 i la Construcció 4.0 on en el sector de desenvolupament del software es pot fer l'analogia del mètode de desenvolupament DevOps on s'accentua la comunicació, la col·laboració, la integració, l'automatització i la mesura del nivell de cooperació entre desenvolupadors de software i altres professionals de tecnologies de la informació.

La majoria dels investigadors van acceptar que les decisions de disseny preses en la primera fase del disseny tenen una forta influència en el cost final, l'eficiència energètica i el rendiment general dels edificis (33). El 20% de les decisions de disseny que es prenen en un primer moment influeixen en el 80% següent de totes les decisions de disseny. La raó és que les primeres decisions de disseny estableixen condicions generals per al procés de disseny posterior. Per exemple, la mida de la relació finestra-paret determinada en la fase inicial del disseny tindria un impacte en el disseny de la il·luminació en la fase posterior. A més, el major potencial d'optimització del disseny d'edificis es troba en la fase inicial del disseny, perquè l'impacte del disseny inicial és més elevat i el cost dels canvis de disseny és més baix en la fase inicial, tal i com es mostra a la Figura 5 (29).

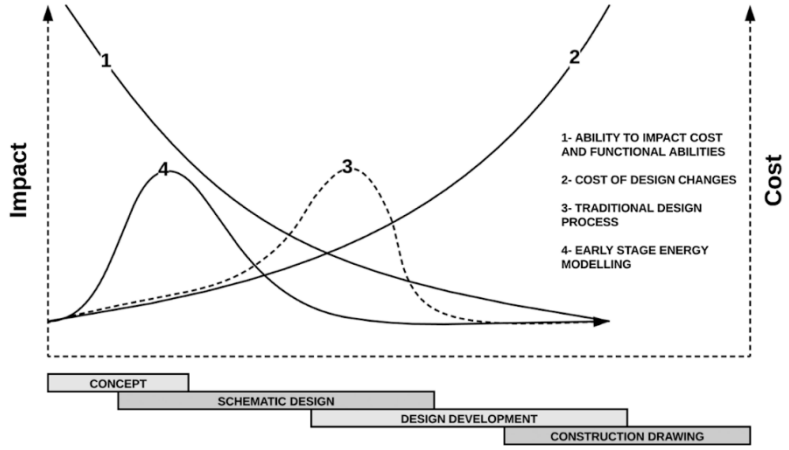


FIGURA 5. GRÀFICA REPRESENTANT EL COST I L'IMPACTE EN LES DIFERENTS ETAPES DEL PROCÉS DE DISSENY D'EDIFICIS SEGONS : 1. CAPACITAT D'INCIDIR EN EL COST I LES CAPACITATS FUNCIONALS, 2. COST DELS CANVIS DE DISSENY, 3. PROCÉS DE DISSENY TRADICIONAL I 4. MODELATGE ENERGÈTIC EN FASE INICIAL (29)

1.2.2. NECADA

NECADA (No Emissions CAD for Architecture), un software de disseny assistit per computador (CAD), tal i com ja s'ha presentat anteriorment en el document és un simulador que permet optimitzar tots els processos relacionats amb la construcció, amb l'objectiu de minimitzar els seus impactes ambientals, econòmics i socials durant totes les etapes el cicle de vida d'un edifici o zona urbana (disseny, construcció, ús, rehabilitació i deconstrucció).

NECADA admet l'execució d'un model de simulació detallat que representa tots els processos de construcció en un supercomputador, un núvol, un clúster d'ordinadors o entorns d'escriptori.

El paràmetre principal per a dur a terme una simulació a NECADA es el model digital representant l'edifici o zona urbana el qual es vol analitzar. Aquest model es el ja presentat en l'apartat 1.2, *Building Information Model* (BIM).

La definició d'aquest model pot ser més o menys complexa i es genera a partir de softwares CAD que permetin el *Building Information Modeling* (BIM) com Autodesk Revit.

El model digital d'un edifici no només pot contenir les seves dades geomètriques, sinó també totes les dades i informació sobre energia, com ara sistemes d'aïllament, estructures vidrades, subministraments d'energia, dades climàtiques, calefacció, refrigeració i ventilació. El model obtingut a partir de fer aquest pas de transformar un model BIM inicial més simple a un amb les característiques addicionals mencionades es el ja presentat *Building Energy Model* (BEM). Aquesta transformació es pot efectuar mitjançant softwares que suportin l'anomenat *Building Energy Modeling* (BEM).

El modelatge energètic dels edificis és un procés iteratiu, es pot dur a terme durant tot el procés de disseny d'un edifici nou o quan es consideren actualitzacions d'edificis existents. També es pot utilitzar més endavant en el cicle de vida d'un edifici relacionat amb la comparació del funcionament real de l'edifici amb l'operació prevista. El ASHRAE Standard 209 titulat "Energy Simulation Aided Design for Buildings Except Low-Rise Residential Buildings" (34) és un document per aplicar el modelatge energètic dels edificis. Descriu una metodologia per aplicar el BEM en el disseny i es va crear per definir procediments fiables i coherents que avancin en l'ús de models energètics que permetin quantificar l'impacte de les decisions de disseny en el moment en què es prenen. L'estàndard inclou diferents "cicles de modelització" per a diferents etapes d'utilització del BEM durant el cicle de vida de l'edifici (35). El primer dels quals es el que es presta atenció i es presenta en aquest projecte tal i com es menciona a l'apartat 1.3.1 d'aquest document.

L'objectiu de la infraestructura NECADA és trobar valors òptims per a diversos paràmetres de construcció i els impactes associats per reduir la demanda o consum energètic de l'edifici o de la zona urbana. La infraestructura combina llenguatges formals i co-simulació per tal d'obtenir una definició completa del model sense ambigüitats. Diem co-simulació ja que NECADA executa les seves simulacions a partir de diferents subsistemes on la comunicació entre ells i la seva execució es invisible per a l'usuari.

A continuació es descriu de forma resumida tot el sistema per tal de posteriorment en l'apartat de problemàtica a resoldre poder ubicar correctament on i perquè sorgeixen possibles millores tractades en aquest TFG i les seves solucions vinculades.

El software NECADA està desenvolupat mitjançant la definició de quatre mòduls principals, mòdul d'entorn, mòdul d'edificis, mòdul de reciclatge i mòdul de compensació.

Mòdul d'entorn. Aquest mòdul permet analitzar les dades climàtiques necessàries per realitzar el càlcul posterior. Això permet tenir cura dels efectes de les modificacions climàtiques en els altres mòduls. En concret, el sistema s'encarrega de tots els paràmetres ambientals que necessitem, com la radiació solar, els vents, les temperatures, etc., sempre per a l'àrea específica de l'anàlisi.

Mòdul d'edificis. Aquest és el mòdul principal del simulador. Es compon de diverses fases que engloben les diferents etapes d'un edifici o zona urbana.

- Fase 1: optimització del disseny del model.
- Fase 2: optimització del procés constructiu, avaluació dels materials i de les solucions constructives realitzades.
- Fase 3: optimització del manteniment i ús.
- Fase 4: optimització del procés de desconstrucció.

Mòdul de reciclatge. S'han de tenir en compte tots els processos necessaris relacionats amb la reutilització i el reciclatge dels residus generats durant el cicle de vida de l'edifici. Aquest procés de reciclatge pot tenir un gran impacte en la nostra manera d'utilitzar l'edifici o en els materials que fem servir.

Mòdul de compensació. Aquest mòdul representa els processos de compensació necessaris per al disseny, construcció, ús i deconstrucció de l'edifici o zona urbana.

Per definir els diferents mòduls i els processos relacionats s'utilitza l'especificació i el llenguatge de descripció (SDL) (36). SDL és un llenguatge gràfic que permet definir els diferents models mitjançant un conjunt de diagrames.

NECADA treballa amb motors de càlcul de renom internacional per a l'optimització tèrmica com EnergyPlus (37), Trnsys o Doe-2.

EnergyPlus pren especial importància en aquest projecte. És un programa d'anàlisi d'energia i simulació de càrrega tèrmica. Basant-se en la descripció d'un edifici que fa l'usuari des de la perspectiva de la composició física de l'edifici, els sistemes mecànics associats, etc., EnergyPlus calcula tots els detalls de simulació que són necessaris per verificar que la simulació funciona tal com ho faria l'edifici real. EnergyPlus permet l'entrada de components d'equips de calefacció i refrigeració configurables per l'usuari que fan la simulació més flexible per descriure les condicions reals.

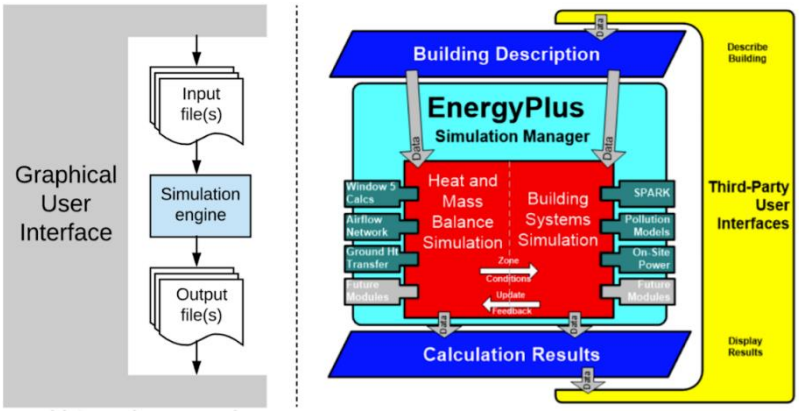


FIGURA 6. A L'ESQUERRA L'ESTRUCTURA GENERAL DE LES EINES BEM. A LA DRETA L'ESTRUCTURA D'ENERGYPLUS (29)

La Figura 6 es mostra l'estructura general de les eines BEM típiques. El motor de simulació (per exemple, EnergyPlus i DOE-2) utilitza fitxers d'entrada d'un format definit per realitzar una simulació (aspecte en el que aquest TFG hi centra l'atenció) i escriu la seva sortida en un o més fitxers de sortida. La interfície gràfica d'usuari sol envoltar aquest procés i permet a l'usuari generar més fàcilment fitxers d'entrada, iniciar la simulació amb el motor i processar els fitxers de sortida per il·lustrar els resultats d'una manera més gràfica. Els fitxers d'entrada inclouen principalment geometria de l'edifici, zones tèrmiques, càrregues internes, sistemes i components HVAC, dades meteorològiques, estratègies i horaris operatius i paràmetres específics de simulació. Un model BIM té la capacitat de proporcionar algunes de les dades d'entrada per a la simulació d'energia, cosa que

significa que els usuaris normalment no necessiten recrear aquesta informació per al model energètic. Aquest és el concepte de BEM basat en BIM que fa que el BEM sigui més eficient.

La Figura 6 també mostra una estructura detallada d'EnergyPlus. A més de la interfície d'usuari de tercers, la descripció de l'edifici (fitxers d'entrada) i els resultats del càlcul (fitxers de sortida), el motor de simulació EnergyPlus té tres components bàsics, incloent un gestor de simulació, un mòdul de simulació de balanç de calor i massa i un mòdul de simulació de sistemes de construcció . El gestor de simulacions controla tot el procés de simulació. El mòdul de simulació de balanç de calor i massa realitza el càlcul del balanç de calor. El mòdul de simulació de sistemes de construcció gestiona la comunicació entre el motor d'equilibri tèrmic i diversos mòduls i bucles HVAC, controla la interacció i l'intercanvi de dades entre EnergyPlus i SPARK i TRNSYS, i gestiona la comunicació de dades entre els mòduls HVAC, les dades d'entrada i l'estructura de dades de sortida (38).

NECADA utilitza el motor de càlcul Radiance per al càlcul de la il·luminació i OpenFoam per calcular la dinàmica de fluids Figura 7. NECADA utilitza algoritmes d'optimització que s'utilitzen àmpliament en altres programes com GenOpt (39) o Dakota (40). Aquests algoritmes d'optimització s'apliquen automàticament al disseny experimental.

El procés de càlcul es realitza de forma local o mitjançant servidors en el núvol, de manera que ens permet realitzar múltiples simulacions i agilitzar el procés de càlcul.

La base de dades NECADA conté les principals característiques dels materials i dels sistemes constructius utilitzats en els estudis: propietats físiques; característiques i impactes ambientals associats (CO2, NOx ...), cost de producció, transport, manteniment i tractament de residus; característiques i impactes econòmics associats i característiques i impactes socials associats.

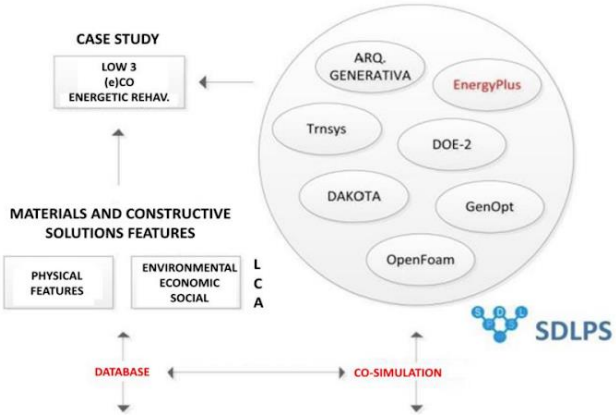


FIGURA 7. RELACIÓ DE LA BASE DE DADES AMB EL SIMULADOR **(41)**

Abans d'entrar en l'arquitectura del sistema cal entendre què és, des del punt de vista de NECADA, un projecte, un experiment i una permutació.

- **Projecte:** un projecte és principalment un conjunt de dades que d'una manera o altra volem analitzar. Com a exemple, un projecte pot estar compost per un conjunt de fitxers climàtics, models de construcció (que representen la geometria dels edificis), fitxers de materials i solucions de construcció. L'objectiu d'un projecte és analitzar, mitjançant l'execució de diversos experiments, quina podria ser la millor alternativa per a un objectiu concret.
- **Experiment:** En un projecte, l'usuari pot definir un conjunt d'experiments. Cada experiment es basa en un subconjunt de models de construcció, fitxers climàtics i solucions constructives que existeixen en un projecte. Principalment estem definint aquí un escenari específic. Volem provar (normalment) milers d'alternatives per entendre quina és la solució òptima.
- **Permutació:** Cadascuna d'aquestes alternatives mencionades és una permutació. Principalment, el motor de simulació parametritza el model segons cadascuna d'aquestes

permutacions i les executa contínuament fins que es fan totes les permutacions d'un experiment o es troba la solució òptima, finalment NECADA informa dels resultats.

L'estructura conceptual de l'arquitectura del sistema es representa a la Figura 8.

Quan un usuari defineix un projecte nou, totes les dades s'emmagatzemen a la base de dades i als directoris del servidor.

Dos *plugins* permeten interactuar amb les dades generades pels diferents motors de càlcul que es poden utilitzar a NECADA. IDFPlugin prepara el projecte per ser entès pel motor de càlcul que s'utilitzarà (Trnsys® o EnergyPlus). Després, les dades, amb l'estructura correcta, s'envien a SDLPS que gestiona el flux d'execució de l'experiment seguint el model de simulació. La connexió amb SDLPS es realitza a través de la seva API. Un cop finalitzada l'execució, SDLSP utilitza el connector ResultsParser per escriure els resultats obtinguts dels diferents motors de càlcul en el format correcte i mostrar-los a l'usuari de manera llegible. Com es mostra a la Figura 7, el model està connectat a una base de dades que conté la informació del material necessària per realitzar el càlcul (Figura 7).

La responsabilitat de la gestió de la simulació i la simulació correspon a SDLPS. El model NECADA està compost per un conjunt de diagrames SDL que defineixen l'estructura i el comportament d'un edifici teòric tal com es descriu a (42). A més, per a cada edifici específic cal detallar les solucions constructives que s'han d'analitzar, l'àrea climàtica està situat l'edifici, la seva orientació i, per descomptat, la seva estructura (basada principalment en fitxers .idf o fitxers Trnsys), entre moltes altres factors.

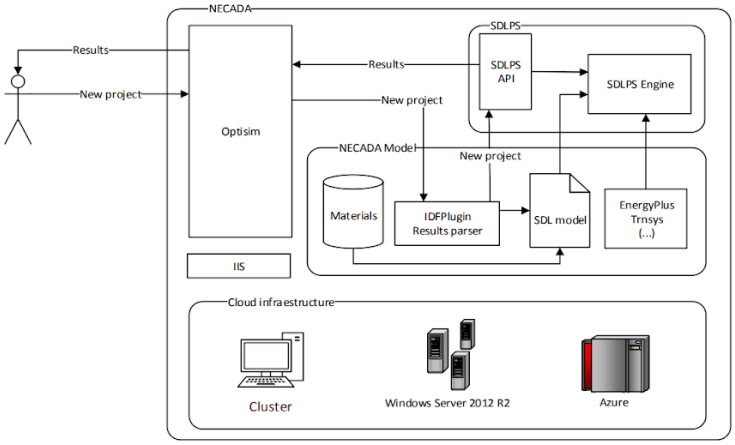


FIGURA 8. ARQUITECTURA DEL SISTEMA. L'USUARI INTERACTUA AMB EL MODEL NECADA A TRAVÉS DE LA INTERFÍCIE OPTISIM. SDLPS ACTUA COM UNA PLATAFORMA DE SIMULADOR QUE PERMET EXECUTAR EL MODEL SDL QUE CONTÉ ELS DETALLS DELS PROCESSOS PRINCIPALS DE L'EDIFICI I ELS MOTORS DE CÀLCUL QUE S'UTILITZEN PER DETALLAR EL CONSUM D'ENERGIA (COM ENERGYPLUS O TRNSYS). **(41)**

La problemàtica d'aquest projecte ve relacionada amb la tasca que realitza Optisim**.** Optisim es la web, basada en el paradigma model-vista-controlador, permet a un no especialista parametritzar el model, crear nous projectes, definir les permutacions, executar l'experiment i analitzar els resultats. Per definir les permutacions, l'usuari ha d'afegir diferents dades al sistema, com ara fitxers climàtics o fitxers BIM. El fitxer BIM de l'edifici ha d'estar en el format que accepta EnergyPlus i NECADA ha d'interpretar la informació que aquest conté i guardar les dades que interessen com ho poden ser els materials que formen l'edifici en la seva base de dades.

1.3. Problemàtica a resoldre

En aquesta secció es plantegen els principals problemes que es volen tractar en aquest projecte. S'ubiquen en el context de tot el sistema NECADA per tal de tenir una perspectiva més precisa d'ells

i, finalment, també continuem introduint i descrivint les idees que ens ajudaran a abordar-los en la part del desenvolupament d'aquest projecte.

Els principals temes a tractar són:

- Investigar i dur a terme el procés d'obtenció d'un BIM (BEM) simplificat de la BRGF que serveixi com a paràmetre d'entrada per la simulació a NECADA i com a base per anar-li introduint més complexitat.
- Investigar la necessitat d'adaptar part del sistema NECADA per que aquest pugui acceptar BIMs amb un nou format anomenat epJSON conjuntament amb el ja suportat format IDF i començar a dur a terme l'adaptació mitjançant una aplicació de consola.

Tots dos temes posen especial èmfasi en la preparació de l'anomenat escenari de simulació.

Tal i com ja s'ha mencionat, un model és una representació i abstracció d'alguna cosa com una entitat o un sistema real o un disseny proposat d'aquest sistema real. La simulació és el fet d'experimentar o exercitar un model o diversos models sota diversos objectius en aquest cas el d'obtenir en un futur escenaris òptims de consum energètic per a la BRGF. Atès que el modelatge és una part integral d'una simulació, anomenem a tot el desenvolupament dels processos com a modelització i simulació (M&S).

Abans d'executar cadascuna de les simulacions desitjades amb el model generat cal que el model compleixi unes condicions mínimes de qualitat perquè el motor de simulació pugui interpretar-lo correctament i poder fer així la seva feina.

Els termes verificació i validació (V&V) es comproven constantment per a qualsevol model generat. El terme de verificació avalua la precisió transformacional del model i aborda la pregunta de si estem creant correctament aquell edifici o zona urbana. El terme de validació va vinculat a l'avaluació de la precisió conductual o representativa del model i aborda la pregunta de si aquest model que estem generant es realment el correcte o desitjat segons els nostres objectius.

La preparació de l'escenari per tant, comprèn tant la "M" de M&S com les dues "V" de V&V. En aquest projecte es du a terme un modelat (M) simplificat inicial de l'edifici de la BRGF (vinculat al primer tema a tractar mencionat a l'inici d'aquest apartat) juntament amb la verificació d'aquest model per tal de que l'escenari quedi ben definit i es puguin llençar les simulacions desitjades.

1.3.1. Obtenció del BIM de la BRGF en format IDF i epJSON

Per executar simulacions al sistema NECADA primer cal obtenir el paràmetre principal de la simulació, el BIM de la BRGF. Aquest paràmetre principal permetrà començar a investigar escenaris òptims de consum energètic per a la biblioteca a través de l'execució de simulacions sobre aquest.

Primerament s'avaluen les diferents possibilitats i camins disponibles per posteriorment desenvolupar un model simplificat.

Diem simplificat ja que el procés de disseny d'un BEM pot arribar a estar format per moltes etapes de perfeccionament fins a arribar a ser un procés molt complex que va dirigit a altres perfils professionals com el d'un arquitecte o enginyer d'estructures.

Per aquest motiu en aquest projecte el que es desenvolupa es un model que es situa en el primer cicle de modelat anomenat *Simple Box Modeling* (43). Amb aquest BEM obtingut no es te l'objectiu d'obtenir ja unes simulacions a NECADA que permetin generar escenaris òptims d'eficiència energètica, sinó de plantar la base del model de la BRGF, tot tenint en compte la geometria de la seva estructura exterior i la seva ubicació obtenint així el que s'anomena una maqueta simplificada de l'edifici. Aquest model servirà com a base en el procés iteratiu de detallar millor el BEM de la BRGF fins a obtenir un *digital twin* (44) que permeti aplicar les modificacions desitjades a l'edifici.

Tal i com s'ha detallat en l'apartat de 1.2.2, EnergyPlus es el motor de simulació utilitzat per SDLPS per a simulacions d'eficiència energètica en edificis, aquest necessita un model d'edifici per a simular. El fitxer d'entrada utilitzat des del 1995 per Energy Plus i des d'un inici per tant per NECADA es un fitxer de dades d'entrada amb l'extensió * .idf (IDF).

Recentment EnergyPlus ha incorporat un nou format, basat en JSON, anomenat epJSON. Té els mateixos elements que l'IDF, però el format que segueix en aquest cas és JSON. Tal i com es detallarà en el següent apartat 1.3.2 en la versió EnergyPlus 10.0 el format IDF deixarà d'estar suportat i per tant passarà a ser un sistema que només acceptarà el nou format epJSON.

Per aquesta raó de l'evolució cap a un sistema que només acceptarà el format de fitxer d'entrada epJSON en aquest projecte s'explora i s'executen els camins necessaris per obtenir un BIM de l'edifici tant en el format IDF com en epJSON.

Tot i que NECADA només accepta fitxers que puguin ser correctament interpretats per el motor de simulació utilitzat EnergyPlus, existeix un format anomenat XML Green Building (gbXML) que també es interessant generar-lo per les seves propietats. El gbXML és un esquema obert desenvolupat per facilitar la transferència de dades d'edificis emmagatzemades en BIM a diferents eines d'anàlisi. Permet la interoperabilitat entre BIM i la simulació de rendiment de l'edifici, que és rellevant per al disseny sostenible d'aquest. Aquest format de fitxer per tant permet la interoperabilitat entre diferents softwares i per tant també entre diferents perfils professionals, aspecte essencial per obtenir un BEM de qualitat. Com en aquest projecte un dels objectius es plantar les bases d'un BIM de la BRGF perquè aquest pugui seguir evolucionant es interessant obtenir aquest BIM simplificat de la BRGF també en format gbXML.

La integració perfecte entre BIM i BEM encara és un domini d'investigació i encara no s'ha adoptat de forma ubiqua a la indústria. Això sovint condueix a la frustració dels diferents professionals per la manca d'una adequada interoperabilitat entre programes. Si s'aconseguís bona interoperabilitat entre BIM i BEM s'obtindrien substancials estalvis en temps i esforç i reducció d 'errors en els models (45). De moment però, molts models BIM provenen de socis de disseny que no estan familiaritzats amb els matisos necessaris pel BEM. Sovint són inconsistents i propensos a errors en la seva creació, i es creen amb eines BIM on els objectes i diferents paràmetres d'aquestes difereixen significativament de la representació necessària en eines de simulació de rendiment (46). Aquests models es tradueixen cap a programes de modelatge d'energia de forma pobre ja que aquests models han estat generats i dirigits per a la documentació de la construcció en lloc de d'anar dirigits cap a la seva simulació de rendiment, per tant, això sovint porta a que determinats detalls arquitectònics siguin irrellevants per a un model energètic a més de no disposar dels detalls necessaris per aquest com que les arestes siguin coincidents o la bona i exacte delimitació dels espais.

Per a realitzar en aquest projecte el model tridimensional de la BRGF en format IDF i epJSON (a més del gbXML mencionat) s'han explorat i utilitzat dos camins diferents que en part comparteixen passos i eines i que han permès copsar millor la gran quantitat de camins que existeixen per aconseguir l'objectiu i les complexitats que pot arribar a tenir aquest procés.

Tots dos camins comencen amb l'obtenció d'un fitxer en format KML des de la Sede Electrònica del Catastro del Govern Estatal on hi és representat l'edifici de la BRGF. CADMapper s'utilitza per a obtenir de forma més ràpida i fàcil un fitxer DXF representant l'edifici de la biblioteca en 3D. Posteriorment s'utilitza Autodesk Civil 3D per fer les neteges necessàries del fitxer DXF obtingut tot comparant les seves dimensions amb les proporcionades en el KML.

Un cop obtenim el DXF netejat i comparat amb el KML es on diferenciem els dos camins. En el cas del primer camí explicat en aquest document (camí blau, Figura 9) s'utilitza Autodesk Revit per a fer les transformacions necessàries al model per tal de que pugui ser exportat a un format gbXML on posteriorment utilitzant OpenStudio podrem fer la conversió d'aquest gbXML que ja conté característiques de modelat energètic al format IDF.

Pel que fa al segon camí (camí vermell, Figura 9) s'utilitza SketchUp amb el plugin d'OpenStudio per generar a partir de la geometria del fitxer DXF el model de la biblioteca tant en format gbXML com directament al format IDF.

L'obtenció del fitxer IDF mitjançant el camí SketchUp i OpenStudio plugin simplifiquen força el camí per a realitzar un model senzill com el que es vol fer, tot i que el camí de Revit i el pas obligatori previ pel format gbXML es molt més adient per afegir complexitat al model tant en estructura i geometria com dels elements per realitzar el modelat energètic.

Finalment tant pel camí blau com pel vermell, per obtenir el format epJSON del model s'utilitza EnergyPlus (Figura 9).

A la següent Taula 1 es pot veure les diferents prestacions que ofereixen dos formats que tenen l'objectiu de permetre la interoperabilitat entre diferents softwares, el primer es l'anomenat Drawing Exchange Format (DXF) que es amb el qual es comença la definició del model BIM, i el segon es el gbXML on ja conté les característiques que es necessiten en un fitxer IDF o epJSON per poder executats en el motor de simulació EnergyPlus i per tant a NECADA.

Building data/data format	*DXF*	*gbXML*
Drawing Units	√	√
Thermal Zone		√
Geometry	√	√
Building components		√
Locations		√
Building Type		√
Building services		
Building materials		
Material thicknesses	√	√

TAULA 1. THE SAN CARLO BUILDING TEST OF INTEROPERABILITY (29)

A continuació s'enumeren els diferents passos representats en la Figura 9 per tal d'identificar-los millor en l'apartat 4.2. Obtenció BIM de la BRGF, on es descriuen més detalladament.

1) Obtenció KML. Cadastre.
2) Obtenció DXF. CADMapper
3) Neteja DXF i comparació KML. Autodesk Civil 3D.
4) DXF a gbXML. Autodesk Revit.
5) gbXML a IDF. OpenStudio.
6) DXF a IDF i gbXML. SketchUp.
7) IDF a epJSON. Energy Plus.

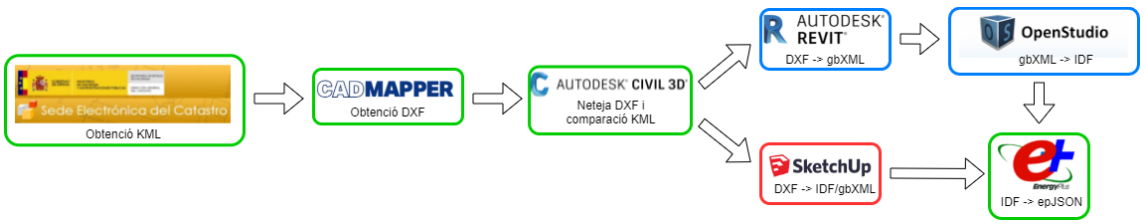


FIGURA 9. DIAGRAMA DELS DOS CAMINS REALITZATS PER A L'OBTENCIÓ DEL MODEL EN FORMAT IDF I EPJSON. EN VERD ELS PASSOS I EINES COMUNES UTILITZADES. EN BLAU EL PRIMER CAMÍ I EN VERMELL EL SEGON.

1.3.2. Adaptació de NECADA al nou format BIM epJSON d'Energy Plus

NECADA conté diversos elements clau que permeten la simplificació de la definició del model per a un usuari no expert de tal forma que s'assegura que no hi ha elements del model que es puguin introduir o modificar que puguin comprometre la integritat del model.

L'estructura conceptual de l'arquitectura del sistema NECADA es la que es veu representada en la Figura 10 on es veu la mateixa estructura que en la Figura 8 però amb la interfície Optisim encerclada en vermell. L'usuari interactua principalment amb aquesta aplicació web anomenada Optisim, es en aquesta aplicació on en aquest apartat hi centrem l'atenció. Optisim permet definir els principals paràmetres d'un experiment de forma fàcil d'utilitzar i intuïtiva. Optisim està basat en l'estructura model-view-controller (MVC) i implementa la interfície principal que utilitzen els usuaris finals del sistema NECADA.

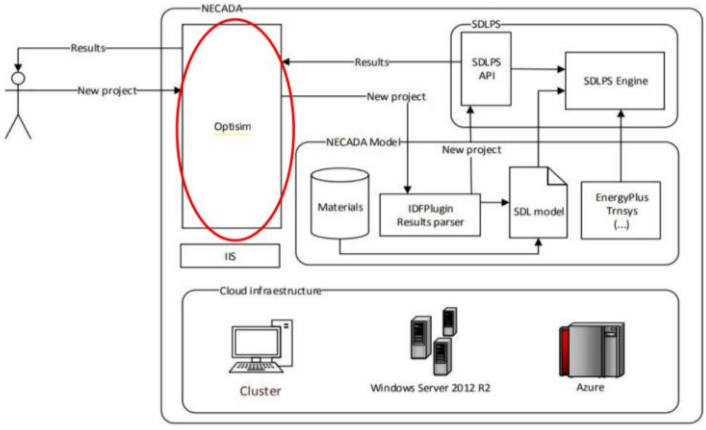


FIGURA 10. ARQUITECTURA DEL SISTEMA NECADA. L'USUARI INTERACTUA AMB EL MODEL NECADA A TRAVÉS DE LA INTERFÍCIE OPTISIM

L'aplicació web Optisim gestiona l'execució de simulacions. Per tant, necessita incorporar fitxers IDF que contenen la definició del model per a realitzar les simulacions. Optisim no permet editar el contingut d'un fitxer IDF, només afegir-lo com fitxer. Per afegir un fitxer IDF es disposa d'un formulari per permet especificar el fitxer a importar, així com el nom que se li donarà dins Optisim i una descripció

(Figura 11). Un cop importat, hi ha un altre formulari per modificar les dades de l'IDF (Figura 12) (però no el seu contingut, que ja no canviarà mai).

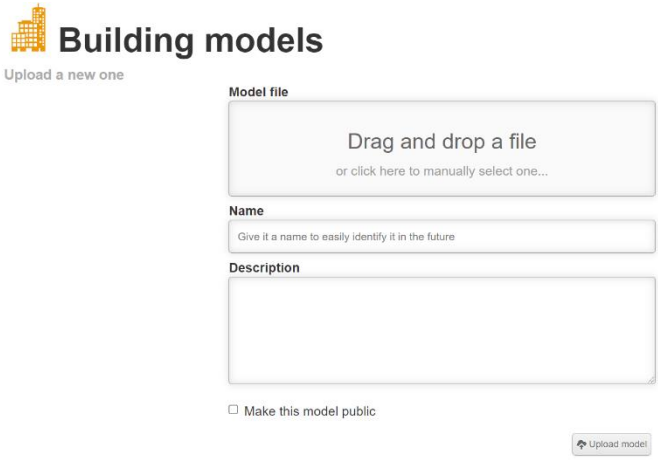


FIGURA 11. FORMULARI A OPTISIM PER PERMETRE ESPECIFICAR EL FITXER IDF A IMPORTAR

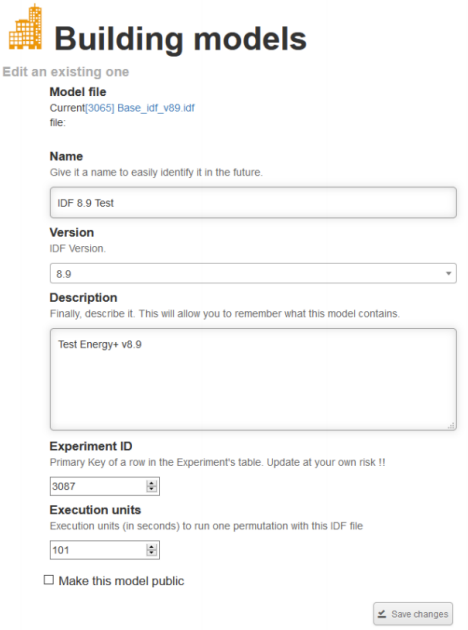


FIGURA 12. FORMULARI DE MODIFICACIÓ DE LES DADES DE L'IDF A OPTISIM

En el primer formulari (Figura 11) es realitzen diverses tasques. La primera és guardar a la base de dades una nova instància de model d'edifici amb les dades mostrades en aquest formulari d'alta. La segona és importar tots els materials i solucions constructives existents en l'IDF i inserir-los en la taula corresponent a la base de dades. Per això és necessari recórrer tot el fitxer IDF (parcejar-lo) buscant el que ens interessa: els materials i solucions constructives; les solucions constructives son estructures com ho pot ser una paret o un sostre que estan compostos per el materials també descrits en l'IDF. La resta del contingut del fitxer IDF és ignorat per Optisim (tot i que sí que ho farà servir Energy Plus quan es llanci una simulació).

NECADA fa servir una base de dades relacional per a guardar tota la informació necessària. Bàsicament, tenim una taula per guardar els models anomenada IDFModels. En l'aplicació web Optisim, aquesta taula es transforma en una classe, IDFModels, que es gestiona mitjançant un ORM (Object Relational Mapping) anomenat Entity Framework (EF).

Per a guardar els materials es disposa d'una taula Materials que correspon amb diverses classes Materials (i subclasses). També existeix una altra taula per a les solucions constructives, amb la seva corresponent classe a Optisim.

Per a contextualitzar la necessitat d'adaptar els sistemes que utilitzen Energy Plus per adoptar aquest format epJSON de fitxer cal tenir en compte l'evolució i les actualitzacions que es faran a Energy Plus respecte els tipus de fitxers d'entrada suportats actualment IDF i epJSON publicades el 7 d'agost del 2017 (47), on actualment quan es redacta aquest projecte la última versió disponible es la 9.5:

- Versió **EnergyPlus 8.8**: l'entrada d'epJSON s'utilitzarà internament i com a entrada experimental (per permetre canvis d'esquema i comentaris dels usuaris sobre els noms de les claus).
- Versió **EnergyPlus 8.9**: epJSON es converteix en ciutadà de primera classe juntament amb IDF. S'inclou el format d'entrada JSON i refactor InputProcessor.
- Versions **EnergyPlus 9.0 - 9.3**: L'entrada d'IDF obsoleta (s'afegeixen avisos d'abandonament d'aquest format a la documentació, missatge d'advertència des de la línia d'ordres quan s'utilitza IDF i warnings de compilació), però encara es te traducció automàtica a EnergyPlus.
- Versió **EnergyPlus 10.0**: Finalment en aquesta versió s'elimina la possibilitat d'entrada d'un fitxer IDF, es congela IDD (Input Data Dictionary) / IDF i es treu el programa de traducció fora d'EnergyPlus.

Aquest canvi fet esglaonadament a mesura que avancen les actualitzacions d'un format IDF cap a un epJSON te la intenció de minimitzar l'impacte en els fluxos de treball existents en sistemes com NECADA que facin ús d'Energy Plus així proporcionant tres anys per a fer les transicions necessàries (sempre que es vulgui utilitzar la versió més actualitzada d'EnergyPlus).

Per donar solució a NECADA a aquesta modificació de canvi de format de fitxer d'entrada en aquest projecte es realitza una aplicació de consola (que no té interficicie web) de tipus .NET Framework amb Microsoft Visual Studio. L'aplicació de consola que s'ha desenvolupat en aquest projecte te la funció de poder importar la informació que interessa a NECADA a la seva base de dades (que es gestionarà amb SQL Server Management Studio tot utilitzant l'ORM per aplicacions .NET anomenat Entity Framework) en dos formats diferents de fitxer d'entrada: IDF i epJSON (tot simulant la funcionalitat d'Optisim en una aplicació de consola) tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou format de fitxer. En aquest projecte ens centrem sobretot en l'exploració del procés d'importació (també tenint en compte l'exportació d'elements de la base de dades de NECADA cap a aquest nou format) del format epJSON i en com adaptar el sistema per tal de que es puguin gestionar aquests dos tipus de formats de forma correcte, ja que Optisim ja gestiona fitxers en format IDF. Pel que fa a la part de gestió del epJSON l'aplicació de consola rebrà el nom i path complet d'un fitxer en format epJSON on amb una llibreria de JSON per C# es llegirà el contingut del fitxer, es validarà amb un esquema, s'extrauran els materials i solucions constructives i es guardarà tot en una nova base de dades (Figura 13).

El model inicial de dades serà exactament el mateix que es té actualment a Optisim i es modificarà per tal de poder permetre importar fitxers tant IDF com epJSON.

L'objectiu final d'aquesta aplicació de consola es que el seu codi es pugui incorporar a Optisim fent els menors canvis possibles. La decisió d'optar per la creació d'una aplicació de consola per assolir aquest objectiu recau en intentar eliminar complexitat en el desenvolupament podent-se centrar exclusivament amb l'objectiu de l'aplicació de consola. Aquesta manera de procedir on primer es crea una aplicació de consola per després poder adaptar el codi a Optisim ja s'ha efectuat en altres projectes de NECADA com ho és el de ResultParser per eliminar complexitat en les primeres fases de desenvolupament de les noves funcionalitats.

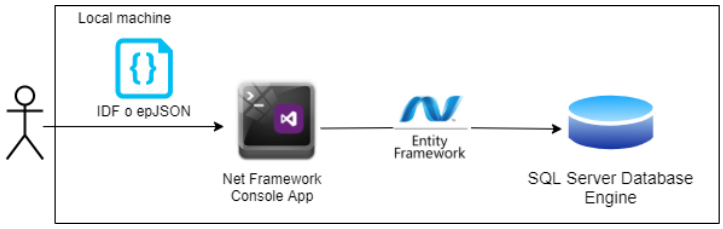


FIGURA 13. DIAGRAMA APLICACIÓ DE CONSOLA

1.4. Abast del projecte

1.4.1. Objectius

El projecte esta dividit en tres objectius, tots tres relacionats amb l'etapa ja presentada de la preparació de l'anomenat escenari de simulació:

- Objectiu 1: Investigar i dur a terme el procés d'obtenció d'un BIM (BEM) simplificat de la BRGF tant en format IDF com en epJSON (tenint en compte l'Objectiu 2) que serveixi com a paràmetre d'entrada per a la simulació a NECADA.
- Objectiu 2: Explorar la necessitat d'adaptar part del sistema NECADA perquè aquest pugui acceptar BIMs amb el nou format epJSON tenint en compte els avantatges que implica conjuntament amb el ja suportat format IDF.
- Objectiu 3: Desenvolupar una aplicació de consola que tingui la funció de poder importar la informació que interessa a NECADA a la seva base de dades en dos formats diferents de fitxer d'entrada: IDF i epJSON, tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou fitxer. De tal forma que fent les menors modificacions possibles es pugui adaptar a NECADA.

1.4.2. Stakeholders

Hi ha diferents parts implicades que tenen un interès en el desenvolupament i resultat d'aquest projecte. Aquests stakeholders es poden dividir en dos grups diferents en funció de la interacció i els beneficis que tinguin amb el propi TFG.

- **Els stakeholders que tenen una interacció directa amb el projecte són:**

L'autor d'aquest TFG, Carles Cidraque Sala, és el responsable de la planificació, desenvolupar i documentar el projecte, així com experimentar, analitzar i treure conclusions.

També forma part d'aquest grup d'stakeholders en Pau Fonseca i Casas com a director del projecte.

Els diferents integrants de l'equip de NECADA també s'inclouen com a stakeholders amb interacció directe amb el projecte ja que com a membres del projecte de recerca NECADA de PolyhedraTech de la UPC treballen en el mateix projecte al qual pertany aquest TFG i poden estar interessats en el seu desenvolupament i els resultats que s'obtinguin. En concret, en José Luis Pérez Vila, integrant de l'equip de NECADA, a part del director del projecte, guiarà i supervisarà l'estudiant per al correcte desenvolupament de determinades tasques de desenvolupament.

- **Stakeholders que no interactuen directament amb el projecte però que reben avantatges si es du a terme correctament:**

La UPC com a institució forma part d'aquest grup d'stakeholders, ja que la pròpia Polyhedra Tech i per tant NECADA sorgeix de la UPC BarcelonaTech. Tot intent de millora que es realitza en aquest TFG del sistema NECADA pot ser d'interès per la UPC. A més, poder obtenir informació per a plantar les bases per optimitzar l'eficiència energètica de la BRGF i així poder adaptar-se a possibles millores energètiques en aquest edifici pot incloure també com a stakeholders d'aquest grup a més de la UPC, als usuaris de la BRGF.

1.4.3. Potencials riscos i obstacles

Compatibilitat entre formats de fitxers. Un dels principals esculls que es pot trobar en el desenvolupament d'aquest projecte es que tindrem versions de formats de fitxers amb els que estiguem treballant que no son compatibles amb les versions d'aquest format de fitxer que utilitzen altres softwares que haurem d'utilitzar i per tant serà un dels esculls principals a corregir. Aquest problema es un dels principals motius pels quals aquest projecte esdevé un projecte d'Enginyeria del Software, ja que a través d'aquest TFG s'aniran solucionant problemes informàtics que estan vinculats amb tot el procés per arribar a completar els objectius marcats.

Canvis en els requisits. Durant el desenvolupament del projecte poden aparèixer nous requisits inesperats i activitats de refactorització. La metodologia Agile utilitzada en aquest projecte pot arribar a mitigar aquest tipus de problemes.

Data límit del projecte. Hi ha un termini per al lliurament del projecte que s'ha de complir. Per tant, haurà d'haver un bon pla i complir amb els terminis especificats per poder acabar el projecte tal com estava previst.

Inexperiència en l'entorn de treball i programació. L'autor d'aquest TFG haurà de passar temps acostumant-se a utilitzar i entendre les noves tecnologies i a adaptar-se, en algunes fases del projecte, a la utilització de llenguatges de programació que mai ha utilitzat.

Llicències d'eines. Pot haver-hi processos en la realització de determinades tasques que es podrien simplificar o agilitzar mitjançant eines de les quals l'alumne no en té la llicència, i que per tant, s'hauran de buscar solucions alternatives que donin el mateix resultat de forma gratuïta.

Impossibilitat de reunir-se presencialment amb el tutor i parts interessades. Mentre les autoritats recomanin les reunions no presencials a causa de la COVID-19, no hi hauria reunions presencials entre tutor, parts interessades i estudiant i es duran totes a terme mitjançant Skype substituint així les reunions presencials per a reunions telemàtiques ja més que habituals.

Problemes de maquinari. Aquest risc té en compte la situació que podria passar en cas que el portàtil utilitzat per desenvolupar aquest projecte es trenqui o sigui robat. Gràcies al fet que tots els documents i el codi s'emmagatzemen al núvol, en el cas de pèrdues aquestes seran gairebé indetectables.

1.5. Eines i metodologia

1.5.1. Metodologia Agile i eines utilitzades

S'han analitzat les característiques de la tesi per triar la metodologia de treball més adequada. Aquest projecte requereix revisions i supervisió constants per assolir el seu objectiu. A més, el projecte pot patir canvis a mesura que es va avançant, per tant, usant la metodologia Agile ens basarem en aquesta adaptabilitat als possibles canvis en els requisits i al seguiment d'aquests en cada procés com a mitjà per augmentar les possibilitats d'èxit del projecte.

La metodologia Agile, al contemplar un desenvolupament iteratiu ens permet testejar cada pas endavant que fem així evitant donar-ne menys d'erronis en comparació amb els que podríem arribar a donar amb un desenvolupament tradicional en cascada.

Les metodología Agile és més ràpida i proporciona una millor comunicació amb els diferents integrants del projecte, tant amb el director del projecte com amb els diferents integrants de l'equip de NECADA que puguin tenir un afecte directe o indirecte en el desenvolupament del TFG, també permet retroalimentació contínua, transparència i flexibilitat en les tasques fetes i per fer en comparació amb les metodologies tradicionals de desenvolupament de programari.

Per tal de que les característiques d'una metodología Agile anteriorment mencionades es compleixin la comunicació amb el director del projecte és constant a través de reunions periòdiques en línia (ja que la situació COVID-19 no ha permès fer-ne de presencials) i intercanvis de correu electrònic.

Skype s'utilitza en el cas de reunions en línia i per a la comunicació via xat en cas de necessitat. Pel que fa a l'administració del projecte de forma personal, l'eina utilitzada per fer-ho serà Trello. Amb aquesta eina fem servir el sistema kanban per al registre d'activitat amb targetes virtuals, de manera que podrem organitzar les diferents tasques segons el seu estat (To Do, Doing, Done i Bloquejat) i adjuntar-hi arxius i vincular-hi tota la documentació exterior necessària, Trello també serveix com a punt d'entrada al projecte.

Tot i tenir disponible Trello per una planificació més personal del projecte, l'equip de NECADA utilitza Azure DevOps Server, un producte de Microsoft que proporciona control de versions (amb Team Foundation Version Control (TFVC) o Git), informes, gestió de requisits, gestió de projectes (tant per a metodologies Agile com per desenvolupament en cascada), versions automatitzades, proves i capacitats de gestió de versions. Cobreix tot el cicle de vida de l'aplicació i habilita les capacitats de DevOps.

Aquesta eina serà l'eina utilitzada quan en el projecte s'hi realitzin tasques que siguin d'interès a ser seguides per a l'equip de NECADA, creant sprints amb les corresponents tasques tot ordenant-les tal i com faríem amb Trello segons els seu estat (To Do, Doing i Done).

S'utilitzarà el repositori de codi que ofereix Azure DevOps dins del projecte NECADA com a eina per al control de versions, cosa que ens permetrà facilitar la disponibilitat del codi i la recuperació de versions anteriors en cas d'errors crítics. S'utilitzaran diferents branques al mateix repositori per tal de tenir una organització ben estructurada del que es fa o està en desenvolupament. Azure DevOps està dissenyat per a Microsoft Visual Studio, aquesta eina serà l'emprada en el moment de gestionar i generar codi pel sistema NECADA.

El contingut de tots els sprints o iteracions estarà condicionat pel progrés realitzat en els anteriors. Es faran planificacions, ressenyes i reunions retrospectives de cada sprint.

Tota la documentació de la memòria del projecte s'ha realitzat amb Microsoft Word, tenint en tot moment la documentació compartida a Microsoft OneDrive amb el director del projecte.

Tota documentació addicional d'interès per a l'elaboració del projecte es penjarà en una carpeta a Google Drive.

Totes aquestes eines mencionades que s'han utilitzat per a dur a terme el projecte tenen la característica de poder ser col·laboratives, per tant, tots els integrants del projecte podran accedir per consultar l'estat del projecte en tot moment en el cas que se'ls hi concedeixi accés.

1.5.2. Requisits Funcionals i No Funcionals

Requisits Funcionals

Model BIM:

- Procés d'obtenció detallat del model BIM en format IDF i epJSON.
- Fàcil escalabilitat del model BIM (BEM) per tal de poder-hi afegir complexitat amb facilitat.
- El model ha de ser simplificat per tal de que esdevingui una definició de l'edifici més estable i fàcil de mantenir.

- Facilitat en l'obtenció i comprensió dels resultats i passos intermedis. Cada funcionalitat ha de ser clara, concisa i no deixar que apareguin ambigüitats.
- Model interoperable entre diferents softwares de modelat (BEM en format gbXML).

Aplicació de consola:

- Bon *feedback* per tal d'assegurar-se de quin es el problema en l'execució o si s'ha completat amb èxit la importació del BIM.
- Importació de la informació requerida a emmagatzemar per NECADA del BIM a la base de dades tant en format IDF com epJSON.
- Tractament del format epJSON aprofitant els seus avantatges respecte el format IDF.

Requisits No Funcionals

- **Usabilitat i facilitat d'utilització**. Al ser una aplicació de consola la seva execució simplement comporta executar l'executable des de la terminal, en cas de dubte es pot consultar al manual d'execució.
- **Disponibilitat:** L'eina ha d'estar sempre disponible. Al ser una aplicació de consola que s'executa de forma local en la nostra màquina l'eina sempre esta disponible, per tant, en qualsevol moment es pot executar i modificar per consultar el seu comportament per a adaptar l'aplicació a NECADA.
- ***Look and feel*.** Tot i ser una aplicació de consola i per tant no tenir interfície gràfica i que no tingui com a fi la seva utilització com a aplicació de consola sinó que el seu objectiu final sigui adaptar-la al sistema NECADA, aquesta ha de tenir un diàleg amb l'usuari que sigui natural i informatiu.
- **Adaptabilitat**. L'aplicació de consola ha de ser adaptable a NECADA realitzant les menors modificacions possibles (per exemple en el model de dades) al sistema per tal de facilitar l'adaptació i eliminant així complexitat en el procés d'importació com en el d'exportació.
- **Escalabilitat**. L'aplicació de consola ha de ser fàcilment escalable a noves versions de fitxers BIM en format IDF I epJSON i a nous materials i informació a importar.
- **Documentació:** La funcionalitat de l'aplicació i el procés d'execució ha de ser entès correctament mitjançant la documentació proporcionada en aquest document. També pel procés d'obtenció del BIM en format IDF i epJSON.
- **Seguretat i privadesa**: Al ser una aplicació de consola que s'executa localment en la nostra màquina no es té especial en compte aquest aspecte, tot i així l'aplicació ja està preparada perquè el model BIM com el projecte al qual s'hi importi aquest model que s'entra com a paràmetre en la aplicació de consola pugui ser públic o no depenent si es vol restringir l'accés i visibilitat a aquests a determinats rols d'usuaris del sistema NECADA.

2. Planificació temporal del projecte

2.1. Introducció

El Treball de Fi de Grau consta de 18 crèdits ECTS, cada crèdit ECTS comporta un total de 25-30 hores de treball (48).Es fixen les 30 hores de treball per crèdit per tal d'estimar les hores màximes contemplades en un inici per a desenvolupar tot el projecte a 540 hores totals.

Es fixa com a data d'inici del projecte el dia 1 de febrer de 2021, data en la qual es va començar ja a treballar en el projecte. Es treballarà el projecte fins a la setmana final de juny i principis de juliol del 2021 ja que es tria com a torn de Lectura el torn de juny, que comprèn les dates des de el dilluns dia 28 de juny fins el divendres 2 de juliol del 2021, per tant es posa com a data final del projecte el dia 15 de juny , ja que la presentació final del projecte es contempla cap a finals d'aquest mes i principi del següent.

Tenint en compte el nombre de dies que s'ha fixat que compondran el projecte (135 dies) i el nombre d'hores que en un inici comportarà (540 hores) podem estimar el nombre d'hores diàries a dedicar al projecte a un total de 4 hores per dia.

2.2. Descripció de les tasques

A continuació, es presenta el bloc de tasques que es duran a terme al llarg del projecte. Es descriuen, la durada i les dependències d'aquestes tasques. El resultat d'aquest projecte formarà part d'un projecte més gran i per tant, les tasques i l'abast del treball són força fixes. Tenint en compte que el projecte segueix la metodologia àgil, les tasques d'implementació es divideixen en sprints.
La primera reunió oficial va ser el 4 de febrer i fins a la finalització del projecte es faran reunions setmanals.
A continuació es divideixen i es detallen les tasques a realitzar en dos grans grups: primerament les tasques vinculades a la gestió del projecte i posteriorment les tasques de desenvolupament del projecte que son les relacionades amb la part tècnica del projecte. Tant les tasques de gestió de projecte com les de desenvolupament son descrites i temporitzades d'acord amb la temporització real duta a terme en el projecte.

2.2.1. Tasques de gestió de projecte

TG0. Contextualització i abast
Definició: S'elabora la part del document que contindrà la contextualització del projecte, l'abast del projecte i les eines i metodologies utilitzades per elaborar el projecte.
Duració: 25 hores
Dependències: -

TG1. Planificació
Definició: S'elabora la part del document que contindrà la descripció de les tasques que es realitzaran, es faran estimacions sobre la duració d'aquestes tasques i juntament amb els seus *timings* s'elaborarà un diagrama de Gantt. Juntament amb aquesta planificació de les tasques s'elaborarà una gestió de riscos que en el que s'explicaran plans alternatius i possibles obstacles.
Duració: 20 hores
Dependències: TG0

TG2.Gestió econòmica i sostenibilitat
Definició: S'elabora la part del document amb l'estimació del pressupost de costos i de gestió, en aquesta tasca també es realitzarà un informe de sostenibilitat del projecte.
Duració: 25 hores
Dependències: TG1

TG3. Definició de la fase inicial del projecte
Definició: S'elabora la part del document que integrarà els continguts dels documents realitzats en les tasques TG0, TG1 i TG2 amb la integració de possibles millores.
Duració: 5 hores
Dependències: TG2

TG4. Documentació de la fase de seguiment del projecte
Definició: S'elabora una entrega que integrarà els continguts realitzats fins al moment del projecte i es realitzarà una reunió amb el director del projecte per a avaluar l'estat i el progrés fins llavors del projecte. La data màxima per fer el seguiment es fixa el dia 28 de maig.
Duració: 35 hores
Dependències: TG3

TG5. Documentació de la fase final del projecte
Definició: S'elabora la documentació necessària per a realitzar l'entrega final del projecte.
Duració: 45 hores
Dependències: TG4

TG6. Preparació per la Lectura del projecte
Definició: Preparació del material necessari per a la presentació final del projecte.
Duració: 30 hores
Dependències: TG5

TG7. Meetings
Definició: Les reunions amb el director del projecte seran de mitja setmanals amb la finalitat d'assegurar que la tesi evoluciona correctament i complint el pla de temps estimat i acordat.
Duració: El temps estimat per reunió es de 30 minuts.
Dependències: -

2.2.2. Tasques de desenvolupament del projecte

Sprint 1
TD0. Obtenció del BIM simplificat de l'edifici de la BRGF en format IDF.
Definició: El primer pas per a la parametrització del model a NECADA consisteix en obtenir el BIM, BEM de l'edifici de la BRGF. Aquest model tridimensional de l'edifici s'obtindrà a partir d'un procés en el qual s'utilitzaran diferents eines que permetran anar fent transformacions del fitxer base inicial fins a obtenir el fitxer de la representació tridimensional de la biblioteca en format IDF d'Energy Plus que servirà com a punt de partida per a la parametrització del model i les seves simulacions a la plataforma de NECADA i com a model base per anar-hi afegint més complexitat en un futur.
Duració: 77,625 hores
Dependències: -

Sprint 2
TD1. Avaluació de si realitzar una migració del model en format IDF cap a epJSON.
Definició: En aquesta tasca es durà a terme un anàlisi per veure si es factible realitzar la migració del model des de el format IDF inicial cap a un format epJSON determinant les necessitats d'actualització del sistema NECADA respecte el software EnergyPlus que utilitza i les seves implicacions.
Aquesta tasca pren especial importància en el projecte ja que segons es decideixi continuar amb l'ús del format IDF o es decideixi migrar cap al format epJSON la seqüencia i metodologia de treball a utilitzar de les següents tasques a realitzar variarà segons es triï una opció o l'altre (TD2 o TD3). Finalment es va optar per TD3.
Duració: 77,625 hores
Dependències: TD0

Sprint 3

TD2. Adaptació de NECADA per a poder incorporar la manipulació de fitxers IDF a la versió més actual disponible.

Definició: En cas de que es decideixi optar per l'alternativa de no migrar el fitxer de format IDF cap a JSON, primer el que s'haurà de veure es si la versió del fitxer IDF actualment permet l'execució directe a NECADA sense haver de fer cap modificació en el sistema o per el contrari el que s'haurà de fer es adaptar el sistema NECADA perquè pugui suportar l'execució del model de la BRGF amb la versió en la qual s'ha obtingut.

Duració: 77,625 hores

Dependències: TD1

TD3. Primera aproximació a l'adaptació del sistema NECADA a poder manipular fitxers epJSON conjuntament amb els IDF ja suportats mitjançant una aplicació de consola.

Definició: Per donar solució a NECADA a aquesta modificació de canvi de format de fitxer d'entrada en aquest projecte es realitza una aplicació de consola de tipus .NET Framework amb Microsoft Visual Studio. L'aplicació de consola que s'ha desenvolupat en aquest projecte te la funció de poder importar la informació que interessa a NECADA a la seva base de dades tant en format IDF com en epJSON tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou format fitxer.

Duració: 77,625 hores

Dependències: TD1

Sprint 4

TD4. Migració BIM de IDF a epJSON

Definició: S'explorarà i es realitzarà el procés d'obtenció en format epJSON del BIM de la BRGF.

Duració: 34,5 hores

Dependències: TD1

Sprint 5

TD5. Acabar de finalitzar tasques TD0, TD3 i TD4.

Definició: S'acaben de finalitzar correctament les tasques TD0, TD3 i TD4. Tasques que en els *sprints* on s'han inicialitzat no han sigut suficients per finalitzar-les ja que han anat patit millores al llarg del seu desenvolupament.

Duració: 77,625 hores

Dependències: TD1

2.3. Recursos

Es divideixen els recursos necessaris per poder organitzar i dur a terme correctament les diferents tasques del projecte detallades anteriorment 3 grups diferents: humans, hardware i software.

Recursos Humans: En aquest projecte es poden identificar a quatre persones que formen part dels recursos humans.

En primer lloc l'estudiant Carles Cidraque Sala, responsable de la planificació, desenvolupament i documentació del projecte.

En Pau Fonseca i Casas com a director del projecte es el responsable de guiar a l'estudiant cap al correcte desenvolupament del projecte.

En Jose Luis Pérez Vila es l'enginyer especialista que porta la part relacionada amb el desenvolupament web de NECADA i donarà un cop de mà a l'estudiant en determinades etapes del projecte.

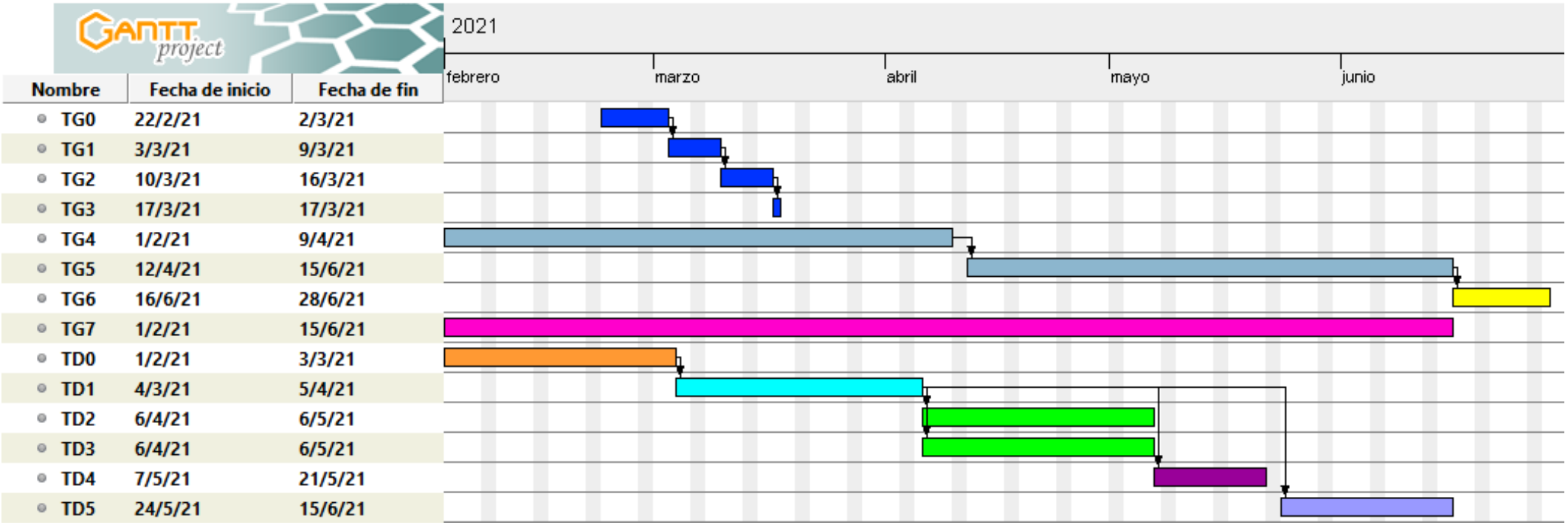
I finalment el tutor de GEP, Fernando Barrabes Naval, és l'encarregat d'ajudar a l'estudiant per a dur a terme la gestió del projecte correctament durant el primer mes del quadrimestre en el que el TFG s'elabora.

Recursos Hardware: El recurs hardware essencial per aquest projecte es un ordinador. L'estudiant utilitzarà el seu propi ordinador personal, un Dell XPS 13 9380 Procesador Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz. A més, tots els recursos per a la connexió a la xarxa (per exemple el router) s'han de tenir en compte. És possible que en algun moment calgui fer una còpia del projecte en una memòria USB.

Recursos Software: Per al desenvolupament del projecte es necessiten diversos softwares que cadascun d'ells anirà destinat a resoldre diferents tasques. Primerament es necessitaran programes com Autodesk Civil 3D, Autodesk Revit, OpenStudio o SketchUp entre d'altres per a obtenir el BIM de la BRGF. Altres softwares que s'utilitzaràn per implementar el projecte seran Visual Studio, Visual Studio Code, SSMS i el propi NECADA. Azure DevOps Server cobreix tot el cicle de vida de l'aplicació de NECADA i habilita les capacitats de DevOps. Les reunions es faran amb Skype. El diagrama Gantt es realitzarà amb GanttProject. Finalment s'utilitza Microsoft Word i Microsoft OneDrive per a la documentació i compartició del projecte.

2.4. Estimacions i Gantt

En el següent diagrama Gantt (Taula 2) veiem els diferents sprints diferenciats per colors. Cal mencionar que les tasques relacionades amb la gestió del projecte (TG0, TG1, TG2 i TG3) i les de documentació d'aquest (TG4 i TG5) pertanyen al sprint al qual li pertoca segons als sprints amb els quals son contemporanis, per tant, podem veure com per exemple els Meetings (TG7) pertanyen a tots als sprints ja que es realitzarà un cada setmana com ja s'ha mencionat anteriorment.



Nombre	Fecha de inicio	Fecha de fin
TG0	22/2/21	2/3/21
TG1	3/3/21	9/3/21
TG2	10/3/21	16/3/21
TG3	17/3/21	17/3/21
TG4	1/2/21	9/4/21
TG5	12/4/21	15/6/21
TG6	16/6/21	28/6/21
TG7	1/2/21	15/6/21
TD0	1/2/21	3/3/21
TD1	4/3/21	5/4/21
TD2	6/4/21	6/5/21
TD3	6/4/21	6/5/21
TD4	7/5/21	21/5/21
TD5	24/5/21	15/6/21

TAULA 2. DIAGRAMA GANTT. ELABORACIÓ PRÒPIA AMB GANTPROJECT

2.5. Gestió del risc: Plans alternatius i obstacles

Atès que el projecte seguirà una metodologia àgil, és molt probable que la planificació experimenti alguns canvis al llarg del projecte (tal i com s'ha detallat en l'apartat 2.2). A més, hi ha altres factors més enllà del de la metodologia que podrien afectar i retardar els temps estimats en aquest document.

A continuació, a la Taula 3 podem veure els diferents riscos i obstacles amb els que ens podem trobar durant la realització del projecte juntament amb el pla alternatiu per a mitigar aquest risc.

Risc	Pla alternatiu
Compatibilitat entre formats de fitxers i les seves diferents versions amb les que es treballarà	S'adaptaran els sistemes necessaris per fer que hi hagi compatibilitat de versions o bé es migrarà cap a altres formats
Inexperiència en l'entorn de treball i programació (tant l'entorn de treball per obtenir el BIM de l'edifici de la BRGF com en menor mesura pel desenvolupament de la aplicació de consola)	Al estimar les hores de treball que comporten la utilització de noves eines s'han estimat conjuntament amb les hores estimades per aprendre el seu funcionament
Llicències d'eines	Buscar solucions alternatives que donin el mateix resultat de forma gratuïta
Planificació temporal errònia i possibles canvis en els requisits	La metodologia agile fa que al final de l'sprint, durant la retrospectiva, validem les tasques a fetes durant aquell sprint, i si hi ha hagut variacions respecte a la planificació inicial es modifica per al següent sprint
Possibles problemes de maquinari	S'emmagatzemen tots els documents i el codi s'emmagatzemen al núvol, en el cas de pèrdues aquestes seran gairebé indetectables

TAULA 3. RISCOS I PLANS ALTERNATIUS DEL PROJECTE

3. Pressupost i Sostenibilitat

3.1. Pressupost

3.1.1. Identificació i estimació de costos

Un cop feta la planificació del projecte, es calculen les despeses que suposaria el desenvolupament d'aquest projecte. En el càlcul del pressupost del projecte no s'han tingut en compte els costos relacionats amb els desplaçaments dels recursos humans a causa del fet que aquest projecte s'ha realitzat encara en fase de pandèmia COVID-19 i la major part del treball s'ha realitzat a casa com a teletreball.

Com ja s'ha mencionat en aquest document en l'apartat 2.3*,* pel correcte desenvolupament del projecte son necessaris una sèrie de recursos humans, hardware i software que tenen els seus costos vinculats.

Finalment, a més a més d'aquests recursos ja mencionats s'han de tenir en compte els recursos indirectes per a poder estimar correctament el cost del projecte (com ho poden ser els costos vinculats amb l'ús de les instal·lacions on s'elabora el treball) i els costos de contingències i d'imprevistos.

3.1.1.1. Costos de recursos humans

En aquest projecte podem identificar sis rols clars per als recursos humans, cadascun amb el seu respectiu cost per hora (49)(Taula 4): *project manager*, *junior developer*, *junior* researcher, *tester*, *analyst* i *technical writer*.

El ***project manager*** assumeix la plena responsabilitat d'iniciar, signar, planificar, controlar, executar, supervisar i tancar el projecte amb èxit. L'estudiant assumirà principalment aquesta responsabilitat, però el tutor del GEP i el director del projecte també faran aquest paper.

El projecte serà implementat per una sola persona, l'estudiant, per tant, serà ell el que realitzarà el paper de ***junior developer***. L'estudiant també ha de fer el paper de ***tester*** per tal de verificar el correcte funcionament del codi desenvolupat com a *developer*. A més, l'estudiant també pren el paper d'investigador, ***junior researcher*** i ***analyst*** pel seu compte, ja que estudiarà, experimentarà, analitzarà i traurà conclusions.

Finalment l'estudiant també prendrà el rol de ***technical writer***, ja que haurà de redactar manuals d'instruccions i altres documents d'una manera que comuniqui la informació tècnica i complexa en un llenguatge menys tècnic, d'una forma ordenada i comprensible per al potencial públic del projecte.

ROL	COST (€)/H
PROJECT MANAGER	23
JUNIOR DEVELOPER	22
JUNIOR RESEARCHER	15
TESTER	8
ANALYST	13
TECHNICAL WRITER	16

TAULA 4. COST (€/H) PELS DIFERENTS SIS ROLS

A la Taula 5 es mostra el salari anual de cadascun dels perfils professionals (49) tinguts en compte per a l'elaboració del projecte així com també es mostra amb el capital destinat a la SS (aproximadament un 23%).

ROL	SALARI ANUAL	SS INCLOSA
PROJECT MANAGER	42320	52053,6
JUNIOR DEVELOPER	40480	49790,4
JUNIOR RESEARCHER	27600	33948
TESTER	14720	18105,6
ANALYST	23920	29421,6
TECHNICAL WRITER	29440	36211,2

TAULA 5. SALARI ANUAL DELS DIFERENTS ROLS I AMB SS INCLOSA. S'HA ESTIMAT QUE EL NOMBRE D'HORES TREBALLADES PER ANY ES DE 1840 HORES.

En la següent taula (Taula 6) podem veure el temps estimat per cada tasca i el rol/s que en prendrien part.

TASCA	HORES	PROJECT MANAGER	JUNIOR RESEARCHER	JUNIOR DEVELOPER	TESTER	ANALYST	TECHNICAL WRITER
CONTEXTUALITZACIÓ I ABAST	25	-	25	-	-	-	-
PLANIFICACIÓ	20	20	-	-	-	-	-
GESTIÓ ECONÒMICA I SOSTENIBILITAT	25	12,5	12,5	-	-	-	-
DOCUMENTACIÓ (TG3, TG4, TG5, TG6)	115	-	-	-	-	39	76
MEETINGS	10	10	-	-	-	-	-
TD0	77,625	-	31,05	34,93125	5,82188	5,82187	-
TD1	77,625	-	54	-	-	23,625	-
TD2/**TD3**(S'ACABA REALITZANT TD3, ES DE LA QUAL ES CALCULEN ELS COSTOS)	77,625	-	23,2875	42,69375	5,82188	5,82187	-
TD4	34,5	-	25	3,5	1	5	-
TD5	77,625	-	23,2875	42,69375	6,21	5,43375	-

TAULA 6. TEMPS ESTIMAT PER TASCA I ROL

Finalment podem calcular el cost estimat de les diferents tasques (Taula 7) que formen el projecte a partir de les dades que s'han proporcionat a la Taula 4 i a la Taula 6, i a la Taula 8 es mostra quin seria el cost de la contractació, conegut com a CPA dels recursos humans dividits pel seu rol en el projecte i també el total en cost d'hores i capital del projecte.

TASCA	COST (€)
CONTEXTUALITZACIÓ I ABAST	375
PLANIFICACIÓ	460
GESTIÓ ECONÒMICA I SOSTENIBILITAT	475
DOCUMENTACIÓ (TG3, TG4, TG5, TG6)	1723
MEETINGS	230
TD0	1356,5
TD1	1117,125
TD2/**TD3** (TD3 ES LA QUE S'HA ACABAT REALITZANT)	1410,83
TD4	525
TD5	1408,89375

TAULA 7. COST ESTIMAT PER TASCA

ROL	HORES	COST
PROJECT MANAGER	42,5	977,5
JUNIOR DEVELOPER	123,8188	2724,0136
JUNIOR RESEARCHER	194,125	2911,875
TESTER	18,85376	150,83
ANALYST	84,70249	1101,1324
TECHNICAL WRITER	76	1216
TOTAL	540	9081,351

TAULA 8. COST TOTAL DELS RECURSOS HUMANS

3.1.1.2. Costos de recursos hardware

El hardware que es considera a utilitzar en un inici per a la implementació del projecte es l'ordinador personal de l'alumne. Cal tenir en compte llavors l'amortització del recurs hardware utilitzat.

La fórmula per calcular l'amortització és la següent (Taula 9) :

$$Amortització = \frac{Preu\ HW\ (€)}{vida\ útil\ (anys) \times dies\ laborables\ al\ any \times ús\ diari\ (hores)} \times duració\ del\ projecte\ (hores)$$

TAULA 9. FÓRMULA CÀLCUL DE L'AMORTITZACIÓ

Tal i com es va indicar en l'apartat de Planificació temporal del projecte, els dies de treball que contempla el projecte i per tant l'ús de l'ordinador son un total de 135 dies. La mitjana d'hores diàries es va fixar a 4 hores per dia, ja que les hores totals estimades que comporten el desenvolupament complet del projecte son de 540 hores. L'esperança de vida de l'equip es fixa a 5 anys i els dies laborables a l'any a 220 dies. A la Taula 10 es mostra l'amortització del hardware utilitzat.

HARDWARE	COST (€)	ESPERANÇA DE VIDA (ANYS)	AMORTITZACIÓ (€)
PORTÀTIL DELL XPS13	1500	5	184,01
TOTAL	-	-	184,01

TAULA 10. AMORTITZACIÓ DEL HARDWARE UTILITZAT

3.1.1.3. Costos de recursos software

Tots els recursos software utilitzats han estat totalment gratuïts o s'utilitzen les seves respectives versions gratuïtes. Per això el total del cost en recursos software és de 0 €.

3.1.1.4. Costos indirectes

Pel fet de que estem en situació de la pandèmia COVID-19 i la major part del treball s'efectuarà i es pot fer a casa com a teletreball, els costos relacionats amb l'electricitat, la connexió a Internet i l'aigua es calcularan a partir dels serveis contractats i de l'import total de temps treballat cada mes i no es tindrà en compte tampoc per tant el cost en desplaçaments que s'hauria de tenir en compte en cas de no tenir la situació de la pandèmia.

Cost en electricitat: El cost real del kWh és de 0,1199 amb "One Luz" Tarifa Endesa. Només es comptabilitzen les despeses en utilitzar el hardware utilitzat. L'únic hardware que es té en compte per tant és el portàtil utilitzat. S'utilitzarà aproximadament totes les 540 hores estimades per a desenvolupar el projecte, per tant, si es te carregador de 45W, el total el consum rondarà els 24,3 kWh generant un cost elèctric de 24,3kWh * 0,1199 €/kWh = 2,91€.

Cost d'Internet: La tarifa d'Internet costa 110 € al mes, si fixem els dies que té un mes a 31, el cost diari d'Internet es de 3,55 € per dia (110 € /31 dies per mes = 3,55 € per dia). Tenint en compte que el projecte té una durada de 135 dies i que les hores de treball diàries es fixen a 4, el cost d'Internet és de 135 dies * 3,55 € * (4 / 24) = 79,875€.

Cost de l'aigua: El cost de l'aigua a la zona de l'estudiant costa al voltant de 30€ per persona al mes. Fent la mateixa equació del cost d'Internet, el cost de l'aigua rondarà els 135 dies * (30€/31) * (4 / 24) = 21,77€.

Per tant, els costos genèrics, és a dir, el cost de l'electricitat, d'Internet i de l'aigua augmenten el cost del projecte en (2,91 € + 79,875 € + 21,77 €) = **104,555 €**.

3.1.1.5. Cost de contingència

El cost de contingència són diners que s'afegeixen al pressupost per a aquells imprevistos que no s'han anticipat. Aquest cost es calcula com un percentatge del valor total del pressupost. El percentatge sol venir determinat pel sector i el nivell de detall del pressupost. En projectes informàtics aquest percentatge sol estar entre el 10 i el 20%, sent 15% el marge escollit per aquest projecte.

Per tant, el **cost de contingència del projecte es de 1405,4874 €** ((Cost recursos humans + Cost recurs hardware + Cost software + Costos indirectes)*0,15 = 1405,4874 €).

3.1.1.6. Cost d'imprevistos

En l'apartat 2.5 s'han especificat els riscos que s'han detectat, així com les accions a prendre per solucionar-los en cas que apareguin. Aquestes accions tindran un impacte econòmic si ocorren, resumits en la Taula 11, és per això que s'ha de calcular quants diners s'hi ha d'afegir al pressupost destinat a costejar les accions a realitzar en cas de risc.

Inexperiència en l'entorn de treball i programació. En els *timings* entre els diferents 5 sprints que componen el projecte ja s'han tingut en compte possibles imprevistos, però en cas de que aquests problemes siguin majors en total s'afegiran 7 hores per sprint. Això acabarà sumant un total de (7 hores * 5 sprints * 22 €) = 770 €. Aquest cost s'ha calculat amb el preu per hora del Junior Developer. El cost s'ha de multiplicar per la probabilitat de risc de que es produeixi l'esdeveniment. El risc és del 30%, per tant, el cost seria de 231 €.

Riscs com **no tenir llicència de determinades eines** que podrien agilitzar el procés, **la compatibilitat entre formats de fitxers i les seves diferents versions amb les que es treballarà** o **possibles problemes de maquinari** seran riscs que faran que aparegui el risc de la **planificació temporal errònia i possibles canvis en els requisits,** que es marcarà que pot arribar a endarrerir el projecte en els pitjors dels casos en 2 setmanes de treball, és a dir, 14 dies * 4 hores diàries = 56 hores addicionals. El preu de les hores també es calculat amb el cost per hora del Junior Developer ja que serà en el cas de que apareguin aquests riscos el que passi més hores executant els diferents plans alternatius mencionats en l'apartat 2.5.

RISC	PROBABILITAT (%)	TEMPS (HORES)	COST (€)
INEXPERIÈNCIA EN L'ENTORN DE TREBALL I PROGRAMACIÓ	30	35	231
PLANIFICACIÓ TEMPORAL ERRÒNIA I POSSIBLES CANVIS EN ELS REQUISITS	50	56	616
TOTAL	-	-	847

TAULA 11. TAULA COST D'IMPREVISTOS

3.1.1.7. Pressupost final

Per tant, es conclou que el cost total del projecte s'estima a un total de 11817,53€ tal i com es resumeix en la següent Taula 12:

DESCRIPCIÓ DEL COST	COST (€)
COSTOS RECURSOS HUMANS	9081,351
COSTOS RECURSOS HARDWARE	184,01
COSTOS RECURSOS SOFTWARE	0
COSTOS INDIRECTES	104,555
COST DE CONTINGÈNCIA	1405,4874
COST D'IMPREVISTOS	847
TOTAL	11622,4034

TAULA 12. COST TOTAL DEL PROJECTE

3.1.2. Control de gestió

Quan s'ha calculat el pressupost final del projecte en els anterior apartats s'han fet estimacions de temps i pressupost i per tant, aquestes dades no seran 100% exactes. En conseqüència, es defineix un model per controlar les desviacions potencials del pressupost per conèixer en tot moment si les prediccions que s'han fet s'estan complint o està passant alguna desviació inesperada.

Aquests model per controlar les desviacions potencials del pressupost contempla mecanismes i indicadors detallats a continuació que seran consultats i actualitzats al llarg del projecte per així poder observar les desviacions que van succeint a mesura que aquest avança.

Desviació cost d'hores per tasca = (Hores estimades - Hores reals) * Cost real

Desviació dels costos en recursos humans per tasca = (Cost estimat - Cost real) * Hores reals

Desviació total costos generals = Cost general estimat - cost general real

Desviació total costos imprevistos = Cost imprevist estimat - cost imprevist real

Desviació total hores = Hores totals estimades - Hores totals reals

Desviació total costos = Cost total estimat - Cost total real

3.2. Sostenibilitat

Per tal d'analitzar l'autoavaluació de l'actual domini de la sostenibilitat, el curs GEP va proporcionar una enquesta. Aquesta enquesta d'autoavaluació m'ha ajudat a recordar que s'han de tenir sempre en compte tres dimensions quan es parla de la sostenibilitat en qualsevol tipus de projecte: la dimensió econòmica, la dimensió ambiental i la social, fins i tot es podria arribar a afegir-n'hi una que no deixa de ser important i es la dimensió cultural.

Desenvolupar els projectes de manera que cap d'aquestes 3-4 dimensions no es vegi afectada es de vital importància. Les 3 dimensions estan molt relacionades entre elles i el fet de deixar alguna de banda pot fer que el projecte acabi tenint un impacte negatiu en les altres dimensions i per tant en el conjunt del projecte.

Ja era conscient de molts dels aspectes que es plantegen a l'hora de realitzar l'enquesta gràcies a que en altres assignatures del Grau en Enginyeria Informàtica a la FIB els introdueixen i ens fan ser conscients dels potencials impactes que poden arribar a tenir les nostres accions com a potencials enginyers. Tot i això, també em fa adonar-me que es un aspecte que mai li he donat la importància que mereix, ja que a l'hora d'elaborar determinats projectes sense tenir una cura especial en la dimensió econòmica, ambiental i social pot arribar a tenir conseqüències greus i inesperades en aquestes 3-4 dimensions.

A continuació es donen resposta a un seguit de preguntes inicials sobre les 3 dimensions esmentades que l'autor del projecte s'ha de plantejar.

3.2.1. Dimensió ambiental

Referent a PPP. He estimat l'impacte ambiental que tindrà la realització del projecte? M'he plantejat minimitzar l'impacte, per exemple, reutilitzant recursos?

La documentació es en format digital per estalviar paper, de la mateixa manera, els correus electrònics o altres documents no s'imprimiran quan no sigui del tot necessari. En aquesta secció hem de tenir en compte el consum energètic causat per l'ús del nostre PC (energia i radiació sense fils) i altres tecnologies (servidors, etc.).

Referent a la Vida Útil. Com es resol actualment el problema que es vol abordar (estat de l'art) ? En què milloraria ambientalment la meva solució respecte a les ja existents?

Un dels objectius que té el software NECADA i per tant aquest projecte concretant en la BRGF es la millora del medi ambient i fer front a l'actual canvi climàtic accelerat que patim a causa principalment de les emissions de CO2 i dels residus radioactius que es generen per a alimentar el consum d'edificis com el de la BRGF.

Amb aquest projecte s'aconseguirà poder plantar les bases per obtenir dissenys òptims i per tant, millor solucions, obtingudes amb més informació per així poder plantejar modificacions en l'edifici de la biblioteca per tal d'intentar reduir el consum energètic, així reduint emissions de CO2 i potencials residus radioactius. Amb l'ajuda de l'entorn cloud NECADA obtenim rapidesa, potencialitat de càlcul i escalabilitat (utilització de servidors en paral·lel al cloud) per a obtenir els escenaris òptims d'eficiència energètica que altres eines actualment no poden oferir. Al explorar la necessitat d'adaptació i millora d'una part del sistema NECADA s'està ajudant a que aquest sistema continuï proveint i millorant el serveix que ofereix que va entre d'altres aspectes estrictament lligat amb la millora del medi ambient.

3.2.2. Dimensió econòmica

Referent a PPP. He estimat el cost de la realització del projecte (recursos humans i materials)?

A l'apartat 3.1 d'aquest document, el lector pot trobar descrit el cost d'aquest projecte, tenint en compte els costos dels diferents recursos utilitzats, incloent-hi les potencials desviacions que poden aparèixer durant el transcurs del treball.

Referent a la Vida Útil. Com es resol actualment el problema que es vol abordar (estat de l'art)? En què millorarà econòmicament la meva solució a las ja existents?

Econòmicament, pel que respecte al desenvolupament del projecte, aquest pot arribar a sortir més barat que d'altres pel simple fet de que el principal autor i desenvolupador del treball es estudiant del Grau en Enginyeria Informàtica de la FIB i per tant el sou estimat en l'apartat de "Pressupost" es un sou d'enginyer Junior, menor al sou d'un enginyer que no ho sigui. A més pel que respecte a altres recursos utilitzats com ho pot ser el recurs hardware no s'hi ha hagut d'invertir un capital inicial per adquirir-lo.

Pel que fa a la potencial millora econòmica que pot proporcionar la solució del projecte es força clara, ja que l'objectiu principal del projecte, com ja s'ha mencionat força vegades, es el de plantar les bases per obtenir escenaris òptims de consum energètic de la BRGF de la UPC tot començant a actualitzar part del sistema NECADA que ho permetrà fer-ho, per tant, estalviant capital actualment invertit en el consum energètic de la biblioteca.

A més, els programes de renovació que poden arribar a sorgir d'aquest projecte en un futur donen un impuls al sector de la construcció que condueix a la creació de més ocupació local i creixement econòmic.

3.2.3. Dimensió social

<u>Referent a PPP</u>. Que crec que m'aportarà a nivell personal la realització d'aquest projecte?

A nivell personal crec que aquest projecte em pot enriquir en molts sentits, no només amb coneixements sobre el desenvolupament de software, sinó també, en determinades ocasions, aprendre com funciona un equip de desenvolupament en una empresa real i com aplicar-hi metodologies Agile.

A més a més, el projecte es situa en un context gairebé completament nou per a mi i per tant, hi veig una gran oportunitat per veure quins son els objectius, dificultats i oportunitats del sector. Crec que al tractar-se a més d'un projecte real força pioner pel que respecte al seu objectiu i la forma d'arribar-hi em pot enriquir i oferir diferents punts de vista que en un inici no tenia.

<u>Referent a la Vida Útil</u>. Com es resol actualment el problema que es vol abordar (estat de l'art)? En què millorarà socialment (qualitat de vida) la meva solució respecte a les ja existents?

Una de les grans millores que pot proporcionar aquest projecte es l'estalvi de diners destinats al consum energètic de la BRGF (tant plantant les bases del model BEM de la BRGF com en plantejament i aproximació de l'actualització de part del sistema NECADA) i que per tant, poden anar destinats a altres projectes amb l'objectiu de continuar millorant la qualitat de vida dels ciutadans. Per tant, a més, la pobresa energètica, en gran mesura, es reduirà a llarg termini quan de forma global s'implementin estratègies de renovació en els edificis perquè esdevinguin energèticament més eficients.

L'altre millora més visible es la de millora del medi ambient i fer front a l'actual canvi climàtic accelerat que patim a causa principalment de les emissions de CO2 i dels residus radioactius que es generen per a alimentar el consum d'edificis com ho son la BRGF, objectiu que en el seu compliment afavoreix la millora en la salut de la població.

A més a més, l'eficiència energètica, i en el cas d'aquest projecte optimitzada, contribueix al benestar dels ciutadans que fan servir la instal·lació tant siguin treballadors com visitants d'aquesta.

<u>Referent a la Vida Útil</u>. Existeix una necessitat real del projecte?

Tal i com s'ha detallat en l'apartat de 1.2 del projecte, actualment tenim com a societat una necessitat imperiosa de reduir el consum energètic dels edificis. De no prendre mesures i no millorar la seva eficiència, l'Agència Internacional de l'Energia (IEA) estima que la demanda global d'energia augmentarà un 50% abans de 2050 situació que faria empitjorar i substancialment la situació econòmica, social i ambiental dels països.

La mateixa IEA ha assenyalat que cal destinar el 76% de les inversions necessàries per assolir els objectius de l'Acord de París han d'anar destinades a millorar l 'eficiència energètica dels edificis ja que el Buildings Performance Institute Europe (BPIE) va trobar que només el 3% dels edificis de la UE es poden valorar com a altament eficients energèticament el 2017, deixant l'altre 97% restant amb necessitat de renovació energètica abans 2050 (50), (3).

Tal i com també s'ha mencionat en l'apartat mencionat anteriorment, actualment s'intenta arribar a solucions de disseny on es millora l'eficiència energètica dels edificis, però no son dissenys del tot optimitzats en la gran majoria, per tant no s'estan obtenint els dissenys més òptims per al consum energètic dels edificis així com sí que ho pot oferir NECADA, i concretament ho pot arribar a oferir amb la BRGF de la UPC on aquest TFG en concret hi ajuda tot plantant les bases del BEM de la biblioteca i de l'exploració i l'inici d'actualització de part del sistema NECADA tenint en compte les actualitzacions que pateix el motor de simulació que fa servir.

4. Desenvolupament

4.2. Obtenció BIM de la BRGF

En aquest apartat es detallen els passos seguits per arribar a obtenir el BIM de la biblioteca en format IDF i epJSON. Tal i com s'ha presentat en l'apartat 1.3.1 s'han utilitzat dos camins diferents els quals fan servir eines diferents però a la vegada també hi ha passos comuns i eines comunes. En cadascun dels següents subapartats de continuació es descriuen els passos tenint en compte la seva enumeració presentada en l'apartat 1.3.1 on podem identificar si correspon a un pas comú entre els dos camins o d'un únic camí en particular.

4.2.1. Obtenció fitxer KML de la BRGF. Cadastre

El primer pas per a la parametrització del model a NECADA consisteix en obtenir la representació tridimensional de l'edifici de la BRGF. Per obtenir aquesta representació tridimensional de la BRGF el que fem es accedir a la web de la seu electrònica del cadastre del govern espanyol (51).

Ens situem sobre l'opció de "Buscador de inmuebles y visor cartográfico" encerclat en vermell a la Figura 14.

FIGURA 14. PLANA INICIAL DE LA SEU ELECTRÒNICA DEL CADASTRE DEL GOVERN ESPANYOL (39)

Allà cerquem la BRGF o bé mitjançant el mapa (Figura 16) o bé amb la seva adreça (Universidad Politécnica de Cataluña, Carrer de Jordi Girona, 1-3, 08034 Barcelona) (Figura 15).

FIGURA 15. CERCADOR D'IMMOBLES A LA WEB DEL CADASTRE (39)

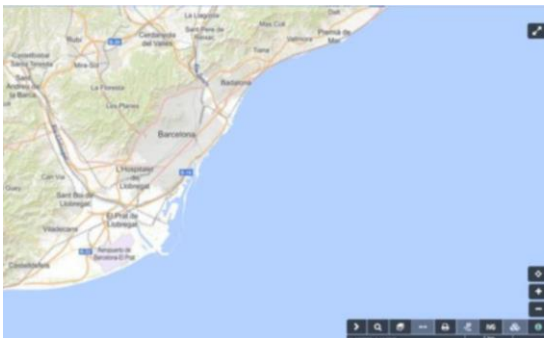


FIGURA 16. CERCADOR D'IMMOBLES AMB EL VISOR CARTOGRÀFIC DEL CADASTRE (39)

El que obtenim al cercar la BRGF es una referència cadastral amb identificador "5924301DF2852D0001DZ" que referencia a una parcel·la formada per diferents edificis adjacents a la BRGF (Figura 17).

FIGURA 17. PARCEL·LA CADASTRAL 5924301DF2852D0001DZ

Com veiem a la Figura 17 se'ns ofereixen diferents formats de la parcel·la per a descarregar. Triem el format KML (Keyhole Markup Language). KML utilitza una estructura basada en etiquetes amb atributs i elements aniuats i està basat en l'estàndard XML. Amb aquest format tindrem la informació necessària per a representar l'edifici en 3D. Un cop descarregat el fitxer KML l'obrim amb Visual Studio Code per tal d'entendre millor el codi i extreure la informació vinculada amb l'edifici que ens interessa. A continuació s'expliquen algunes de les etiquetes bàsiques i d'interès del fitxer KML descarregat per tal d'obtenir informació més específica de la BRGF.

Primerament tenim els anomenats *Placemark*, aquests permeten marcar una posició sobre la superfície de la Terra. El *Placemark* més senzill inclou només un únic element de punt (<Point>), que especifica la ubicació del *Placemark* en qüestió. També se li pot especificar un nom i elements geomètrics entre d'altres etiquetes.

Gràcies a l'extensió extensió *Geo Data Viewer* que ofereix VS Code (Figura 18, Figura 19) podem visualitzar en 2D i 3D tota la parcel·la a la que fa referència el KML que estem analitzant i ens permet identificar els Placemarks que formen la BRGF i que per tant ens interessen. En concret son els Placemarks identificats com a (-I+IV) i (-I+POR) que es poden visualitzar a les il·lustracions Figura 20 i Figura 21 respectivament.

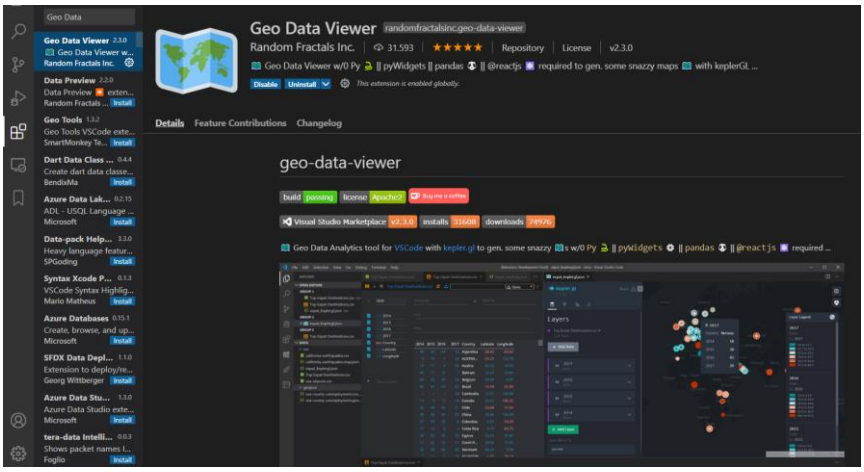


FIGURA 18. EXTENSIÓ GEO DATA VIEWER DE VISUAL STUDIO CODE

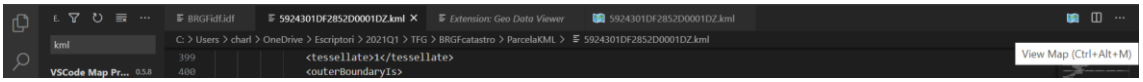


FIGURA 19. OPCIÓ PER VISUALITZAR EL CODI KML AMB L'EXTENSIÓ GEO DATA VIEWER

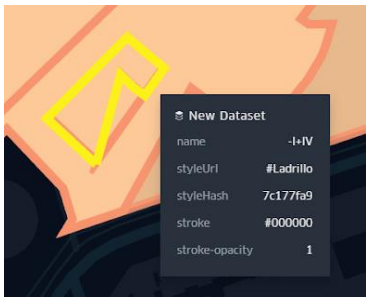


FIGURA 20. PLACEMARK -I+IV EN GEO DATA VIEWER DE VSCODE

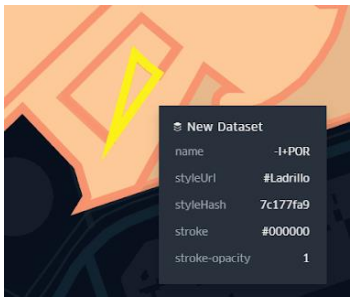


FIGURA 21. PLACEMARK -I+POR EN GEO DATA VIEWER DE VSCODE

Un cop identificats els *Placemarks* que ens interessen ja podem identificar-los dins el fitxer KML i obtenir més informació rellevant sobre l'edifici. A continuació mostrem el codi vinculat al *Placemark* "-I+IV" a mode d'exemple per explicar les diferents etiquetes que ens interessa analitzar (Figura 22).

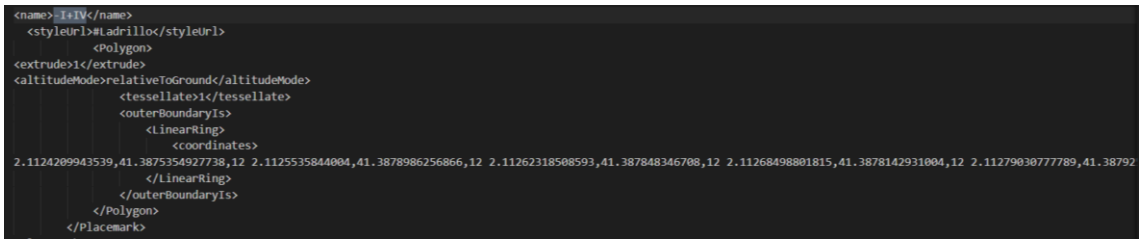


FIGURA 22. CODI EN KML DEL PLACEMARK -I+IV

Els polígons **(<Polygon>)** es poden utilitzar per crear edificis simples i altres formes, dins d'aquesta etiqueta s'hi aniuen d'altres per donar més forma i informació als diferents polígons. Un polígon es defineix per un límit exterior i 0 o més límits interiors. Els límits, al seu torn, estan definits per *LinearRings*. Quan es fa ***extrude*** (*<extrude>1</extrude>*) d'un polígon, els seus límits es connecten a terra per formar polígons addicionals, cosa que dona l'aspecte d'un edifici.

L'etiqueta ***<altitudMode>*** especifica com s'interpreten els components d'altitud de l'element *<coordinates>.* En el nostre cas, en els dos Placemarks pren valor de *relativeToGround* que estableix l'altitud de l'element en relació amb l'elevació del terreny real (del terra sobre el que es construeix, no sobre el nivell del mar) d'una ubicació concreta. L'element ***<coordinates>*** està compost per quatre o més tuples, cadascuna formada per valors de coma flotant per a la longitud, la latitud i l'altitud en aquest ordre del polígon en qüestió. Les coordenades només s'expressen en graus decimals.

Un cop obtingut el KML del cadastre i analitzat i entès la part del codi que ens pot interessar per a la representació en tres dimensions de la BRGF ja podem passar al pas referent a la conversió del model en format KML a format IDF d'Energy Plus i epJSON.

4.2.2. Obtenció DXF. CADMapper

Per facilitar la feina de l'obtenció del fitxer DXF a partir del format KML i que aquest es pugui visualitzar correctament al Autodesk Revit i SketchUp s'ha utilitzat una aplicació web auxiliar anomenada Cadmapper (52) que transforma dades de fonts públiques com OpenStreetMap, NASA i USGS en fitxers CAD ben organitzats. Aquesta aplicació ha permès descarregar-se directament el model de la BRGF en format DXF.

Per obtenir aquest DXF primerament cliquem sobre "GET STARTED CREATING A FILE" des de la pàgina principal de Cadmapper (encerclada amb color vermell a la Figura 23). Un cop redirigits a la finestra de "Create Map" cal marcar l'opció de AutoCAD com a design program i marcar el checkbox de "3D Buildings (if available)" tal i com es mostra en la Figura 24, el resultat a la Figura 25.

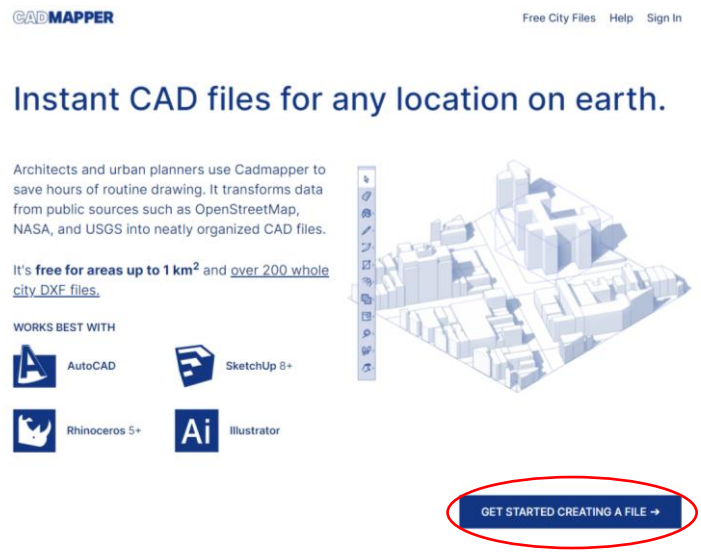


FIGURA 23. PLANA INICIAL DE CADMAPPER (40)

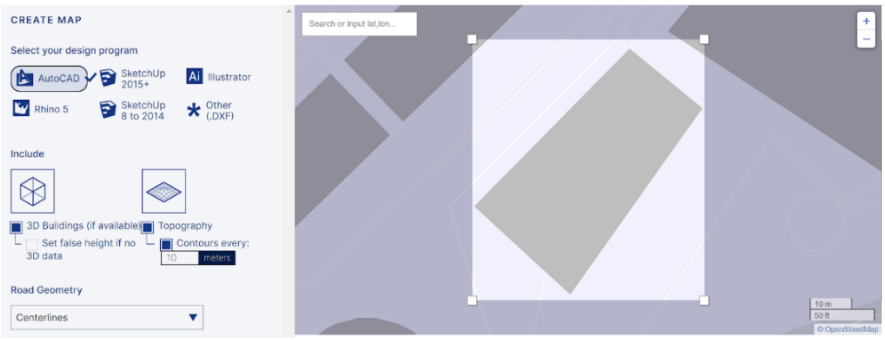


FIGURA 24. FINESTRA DE CREATE MAP DE CADMAPPER SOBRE LA BRGF

FIGURA 25. FINESTRA DE EDIT MAP DE CADMAPPER DE LA BRGF

4.2.3. Neteja DXF i comparació amb KML. Autodesk Civil 3D

El que hem de veure es si realment aquest model en format DXF obtingut amb CADMapper en el pas anterior s'ajusta a les especificacions que el KML ens ofereix i si el KML ens pot complementar informació al model descarregat en DXF. Per fer-ho obrim el DXF descarregat del CADMapper al Autodesk Civil 3D on farem una sèrie de mesures sobre la geometria de l'edifici i la seva orientació.

En el cas de que no haguem pogut enquadrar del tot bé l'àrea a processar per CADMapper per tal de que només hi aparegui la BRGF al fitxer DXF i haguem enquadrat parts d'edificis que no ens interessin

(encerclades en vermell a la Figura 26) llavors cal primer netejar el fitxer tot eliminant-los un cop el tinguem obert al Civil 3D. Per fer-ho senzillament seleccionem les parts que no formen part de la BRGF i les suprimim amb *Supr*.

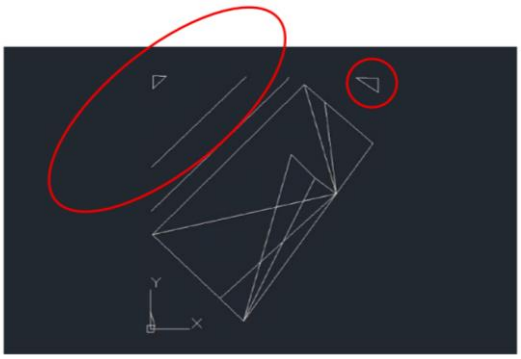


FIGURA 26. VISTA EN PLANTA DEL FITXER DXF A CIVIL 3D ON LES PARTS QUE NO PERTANYEN A LA BRGF ESTAN ENCERCLADES AMB VERMELL

Un cop en el fitxer DXF hi tinguem únicament la BRGF procedim a contrastar la seva orientació. Veiem que les dades recollides per CADMapper amb l'ajuda d'OpenStreetMap pel que fa a l'orientació de la biblioteca es la correcta. Comparem també visualment l'orientació de l'edifici respecte Google Earth i veiem que també concorden (tant en la Figura 27 com en la Figura 28 la brúixola encerclada amb vermell es troba en la mateixa posició).

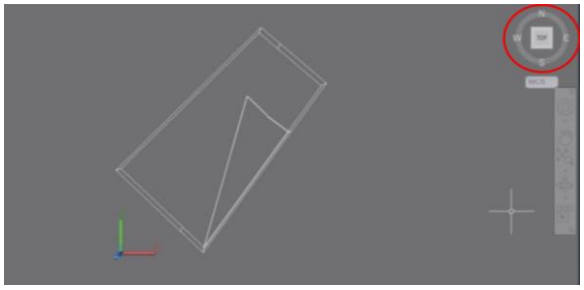


FIGURA 27. BRGF VISTA DE PLANTA A CIVIL 3D

FIGURA 28. BRGF VISTA DE PLANTA GOOGLE EARTH

A continuació procedim a contrastar la geometria que ens ofereix el fitxer KML amb el DXF de CADMapper. Pel que fa al model en DXF, l'altura que veiem que CADMapper ens calcula de l'edifici es de 12 metres, on en tota l'estructura es la mateixa, això ho podem comprovar amb el Civil3D utilitzant la comanda DIST (Figura 29), i clicant sobre el punt d'inici i final entre els quals volem mesurar la distància.

FIGURA 29. COMANDA DIST A AUTODESK CIVIL 3D

Veiem que en el KML la Placemark -I+IV (Figura 22), que es la principal que forma el cos de la biblioteca veiem que totes les tuples de les coordenades també tenen una altura de 12 metres (la tercera coordenada de cada tupla dins de l'etiqueta <coordinates> indica l'altura respecte el terra).

El següent aspecte a contrastar serien les dimensions de llargada i amplada de l'edifici.
Primerament el que fem es mesurar l'amplada i la llargada de l'edifici a del fitxer DXF generat per CADMapper amb el Civil3D amb la comanda DIST com hem fet anteriorment amb l'alçada. Veiem que la part frontal de l'edifici te una amplada de 21.7764 metres (Figura 30).

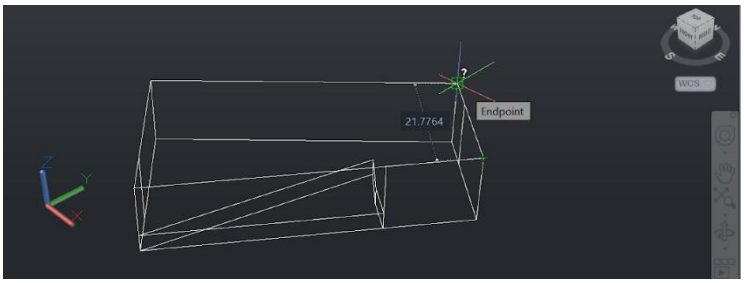


FIGURA 30. MESURA DE LA PART FRONTAL DE LA BRGF AMB DIST A CIVIL 3D

La part posterior de l'edifici te una amplada de 30.4084 metres (Figura 31).

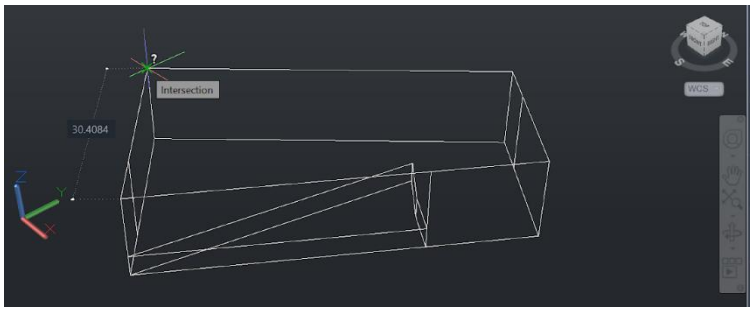


FIGURA 31. MESURA DE LA PART POSTERIOR DE LA BRGF AMB DIST A CIVIL 3D

La llargada de la part lateral que esta de cara al nord-oest mesura 51.6744 metres (Figura 32).

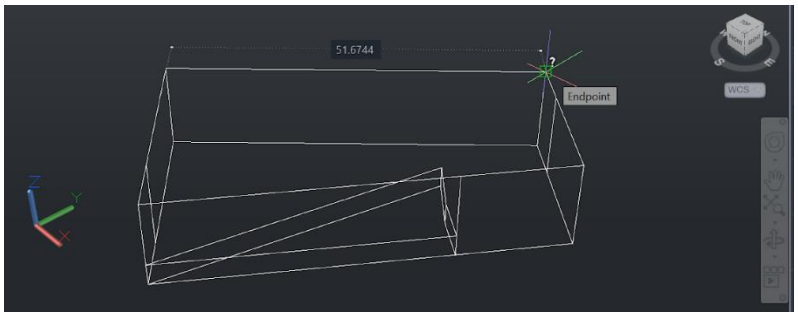


FIGURA 32. MESURA DE LA PART LATERAL NORD-OEST DE LA BRGF AMB DIST A CIVIL 3D

Veiem que les distàncies mesurades del fitxer DXF si que coincideixen amb les del fitxer KML gràcies a les coordenades que aquest últim ofereix. Amb l'ajuda d'un script per a calcular distàncies a partir de punts latitud/longitud (53), hem pogut calcular la distància entre dues coordenades donades, i gràcies a la possible visualització del KML amb l'extensió de *Geo Data Viewe*r de VSCode s'ha pogut

comprovar on es trobava físicament cadascuna de les coordenades oferides en el KML tot situant-nos amb el cursor sobre la posició de la que volem saber aquestes coordenades.

Si passem a analitzar el segon *placemark* que forma l'estructura de la BRGF veiem que les distàncies de llargada i amplada s'ajusten a les que ofereix el KML.

Veiem per exemple que la llargada son uns 38.0180 metres (Figura 33), que coincideix amb la distància marcada per les coordenades donades en el KML del Placemark -I+POR.

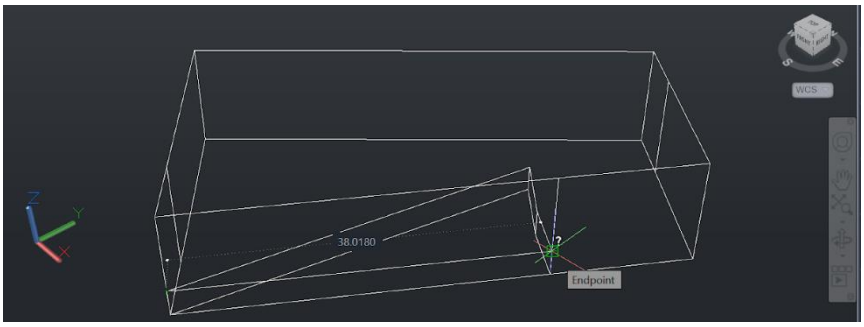


FIGURA 33. MESURA DE LA LONGITUD DEL PLACEMARK -I+POR DE LA BRGF AMB DIST A CIVIL 3D

L'únic problema que li trobem a aquest *placemark* -I+POR tant en el KML com en el DXF obtingut a partir de CADMapper es que el seu volum es troba situat arran de terra, quan es tot el contrari a la realitat, on aquest es troba just al sostre, semblant a la forma d'un porxo tal i com es mostra a la Figura 34.

FIGURA 34. CAPTURA DEL PLACEMARK -I+POR DE L'STREET VIEW DE GOOGLE MAPS

A més també podem comprovar que l'altura assignada a aquest element de l'estructura es únicament d'1,5 metres en el KML, això queda lluny de la realitat on podem veure gràcies al mesurament d'altura ofert per Google Earth que l'altura d'aquest Placemark -I+POR té un 38,89% respecte l'altura de l'edifici, que traslladat a l'altura que ens proporciona el DXF de CADMapper com el KML representaria una altura de 4,67 metres.

Per tant el que hem hagut de fer es pujar aquest element amb l'altura augmentada fins que el seu sostre quedi a l'altura del sostre de l'estructura principal de la biblioteca (Placemark -I+IV) de tal forma que sota el Placemark -I+POR quedi un espai buit.

4.2.4. DXF a gbXML. Autodesk Revit.

El següent pas consisteix en importar el model en format DXF netejat i contrastat cap al Autodesk Revit, allà recrearem la geometria del model per posteriorment, des de el mateix Revit exportar-lo com a un fitxer amb format gbXML. Per tal de que el model pugui ser exportat al format gbXML necessitem recrear la geometria del model en format DXF de tal forma que hi generem delimitadors d'espais com el sostre, les parets o el terra de l'edifici, elements necessaris per al modelat energètic d'aquest.

Per fer aquest següent pas el que fem es obrir el Autodesk Revit i crear un nou projecte amb el *template* de *Metric-Construction Template* (Figura 35).

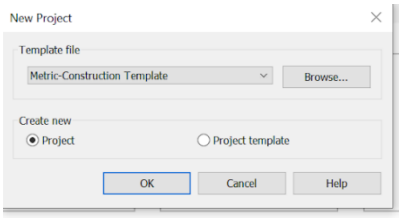


FIGURA 35. FINESTRA DE CREACIÓ D'UN NOU PROJECTE A AUTODESK REVIT

Un cop el tenim creat importem el fitxer DXF, per fer-ho seguim els següents pasos:

- Primer ens situem al Level1, on hi obrirem el model.
- Un cop situats linkem el model DXF amb l'opció de "link CAD" de la pestanya "insert" (Figura 36).

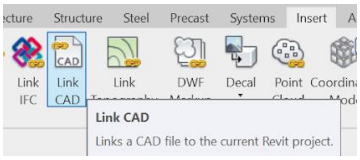


FIGURA 36. OPCIÓ LINK CAD A AUTODESK REVIT

- Obrim el DXF de la BRGF amb el paràmetre "Import units" amb valor de *meter*, ja que el DXF obtingut de CADMapper utilitza els metres com a unitat de mesura de la geometría del model (Figura 37).

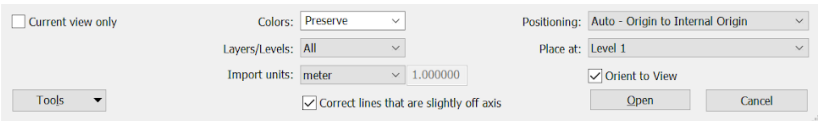


FIGURA 37. FINESTRA D'IMPORTACIÓ DEL FITXER CAD AMB ELS DIFERENTS PARÀMETRES DISPONIBLES

Un cop obert el fitxer CAD en format DXF podem triar l'opció de "Default 3D View" i poder veure el model de la BRGF en 3D (Figura 38).

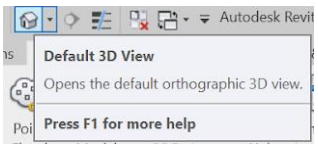


FIGURA 38. OPCIÓ DE DEFAULT 3D VIEW DE AUTODESK REVIT

Ara ja podem començar a recrear la geometria de l'edifici calcant la geometria importada del DXF.

El primer pas a fer es ubicar el terra de la BRGF al Level1, que es el nivell a 0 metres d'altura del terreny i així, al exportar-lo al format gbXML la base de la biblioteca es trobarà a una altura de 0 metres sobre el terra (Figura 39).

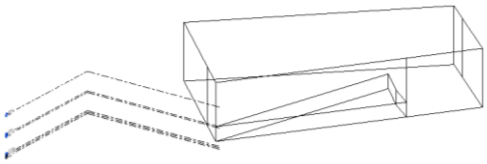


FIGURA 39. CAPTURA DEL MODEL EN FORMAT DXF IMPORTAT A AUTODESK REVIT

Per tal de poder moure el model de la BRGF cal fer un "*unpin*" tot clickant sobre la icona que apareix quan seleccionem el model tal i com es mostra a la Figura 40.

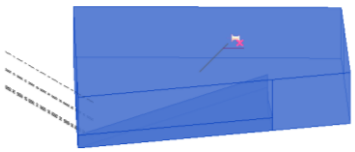


FIGURA 40. UNPIN DEL MODEL PER TAL DE PODER-LO MOURE

Ara ja podem començar a recrear la geometria de l'edifici tot utilitzant l'eina de *"In-Place Mass"* ubicada a la pestanya *"Massing & Site"* tal i com es mostra a la Figura 41. Crearem dos *mass*, una per el *placemark* -I+IV i un altre per el -I+POR. Aquests dos *placemarks* al tenir dues altures diferents se'ns serà més fàcil de recrear la geometria si creem dos *mass* per separat, les modelem i després les unim.

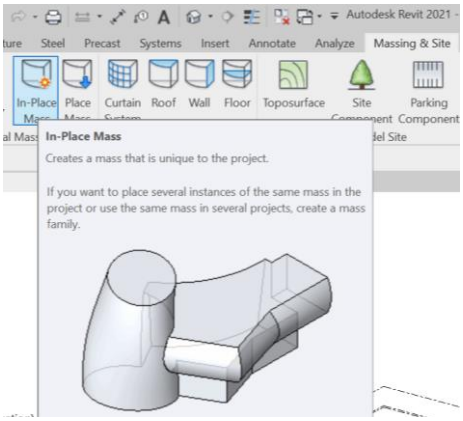


FIGURA 41. EINA DE PLACE MASS A AUTODESK REVIT

Per crear aquestes *mass* el que fem es utilitzar l'eina "*Line*" per a formar el perímetre de l'estructura per posteriorment utilitzant l'opció de *"Create Solid Form"* poder donar-li l'altura corresponent a cada una de les dues *mass* creades (Figura 42).

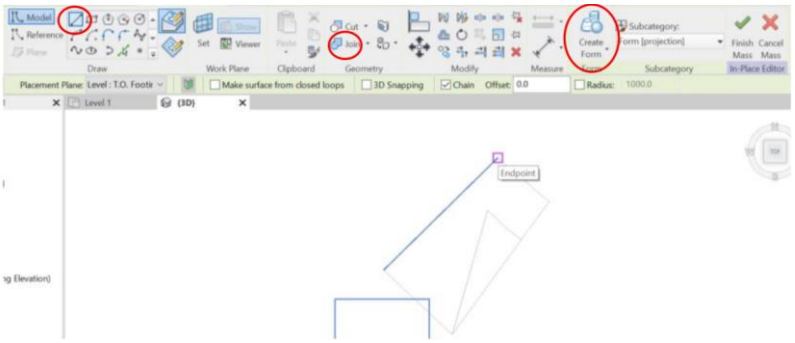


FIGURA 42. CREACIÓ DE LA MASS AMB L'EINA LINE PER POSTERIORMENT UTILITZAR CREATE SOLID FORM

Un cop creades les dues *mass* sent calcades a la geometria corresponent al DXF inicial però amb la variació comentada anteriorment sobre l'altura i la posició del *placemark* -I+POR, ja les podem unir amb l'eina *"join"* i podem crear-hi les corresponents parets, sostres i terres utilitzant les opcions de *"Create Floor by Face", "Create Roof by Face" i "Create Wall by Face"* respectivament seleccionant les cares de l'estructura segons el que volem que representin (Figura 43).

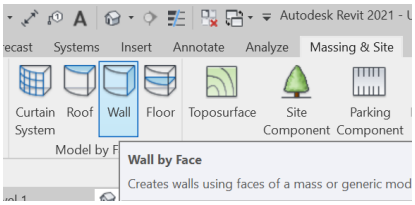


FIGURA 43. OPCIÓ DE CREATE WALL BY FACE A AUTODESK REVIT

Posteriorment, per a crear l'*energy model* del model s'han de crear els espais corresponents de l'edifici delimitats per les parets, terra i sostre creats anteriorment. Per fer-ho creem les *rooms* necessàries que delimitaran l'espai que s'analitzarà en la creació de l'*energy model*. El nivell d'inici de la *room* es el terra de la BRGF i el límit superior es el *Level 2* (nivell on es troba situat el sostre de l'edifici) (Figura 44). Les *Rooms* son utilitzades pels arquitectes per contenir informació sobre els acabats i el seu contingut.

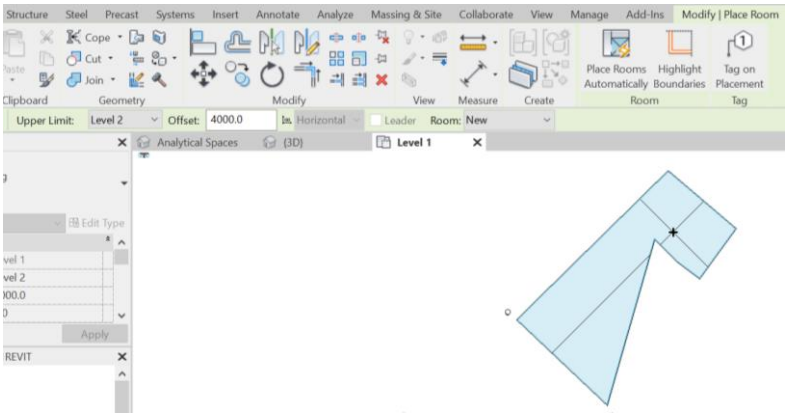


FIGURA 44. CREACIÓ ROOM DE L'EDIFICI

Posteriorment, un cop creada la *room* procedim a la creació de les anomenades *Zones*. Aquestes *Zones* definiran les zones HVAC que es tindran en compte a l'hora de generar l'*energy model.* Abans però de crear aquesta *Zone* s'han de crear els espais (*spaces*) als quals estaran associats, procedim de la mateixa forma que en la creació de la *room (*Figura 45*)*. Els enginyers fan servir *spaces* per fer un seguiment de la informació elèctrica, d'il·luminació, d'energia i de climatització de l'edifici.

Per crear la *Zone* senzillament seleccionem l'eina de *create zone* i seleccionem l'*space* creat anteriorment (Figura 46).

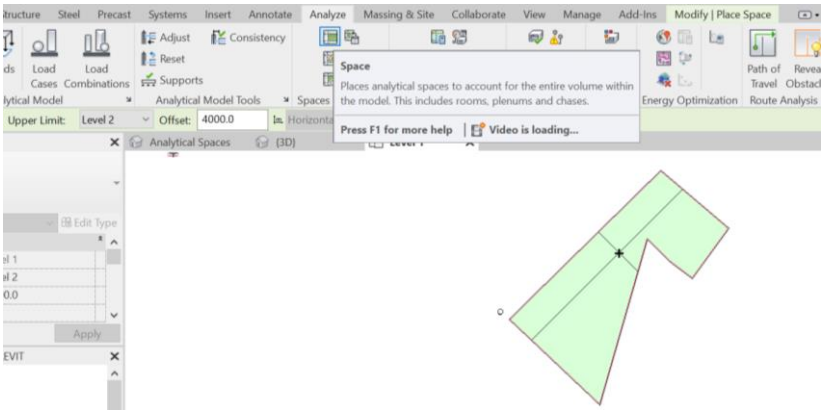


FIGURA 45. CREACIÓ SPACES DE L'EDIFICI

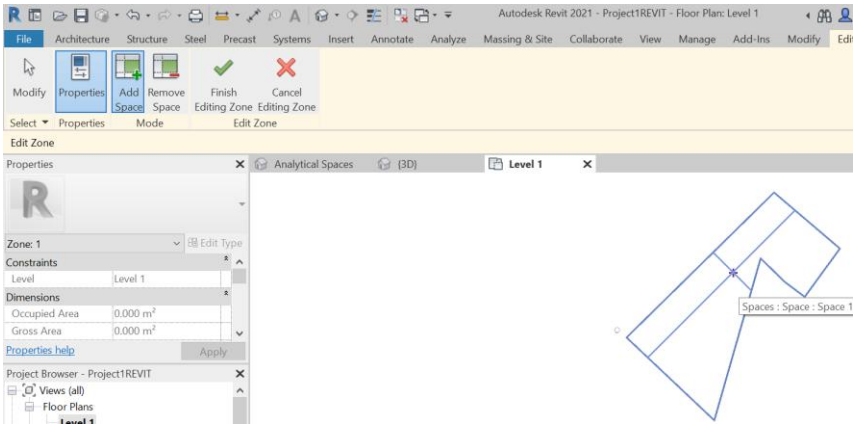


FIGURA 46. CREACIÓ ZONES DE L'EDIFICI

Un cop creades, utilitzant l'opció de *"Visual Style: Shaded with Edges"* (Figura 47) podem visualitzar d'una millor manera l'estructura de la BRGF que finalment hem obtingut (Figura 48).

FIGURA 47. OPCIÓ DE VISUAL STYLE: SHADED WITH EDGES

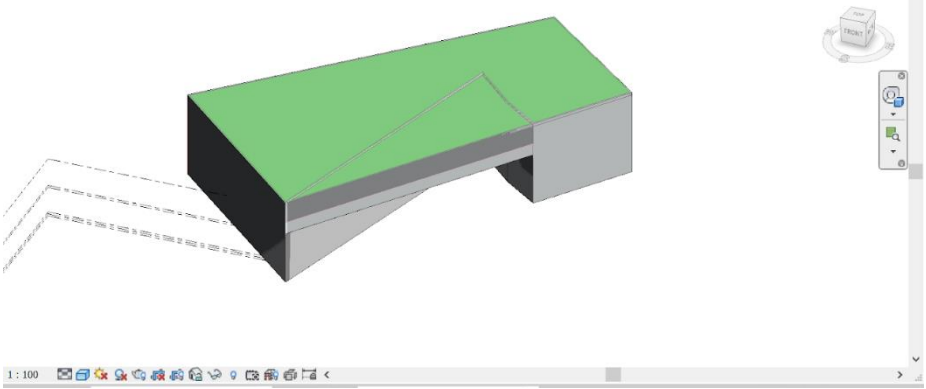


FIGURA 48. ESTRUCTURA FINAL DE LA BRGF A AUTODESK REVIT

Per tal de poder exportar el model en format gbXML necessitem abans utilitzar l'eina de *"Create Energy Model"*, aquesta eina està situada a la pestanya *"Analyze"*. Amb aquesta eina es crea un model d'anàlisi energètic simplificat (EAM) amb les dades que tingui el model fins al moment.

Un cop creat aquest model sen's mostren la geometria que es tindrà en compte a l'hora de exportar-la a gbXML (Figura 49):

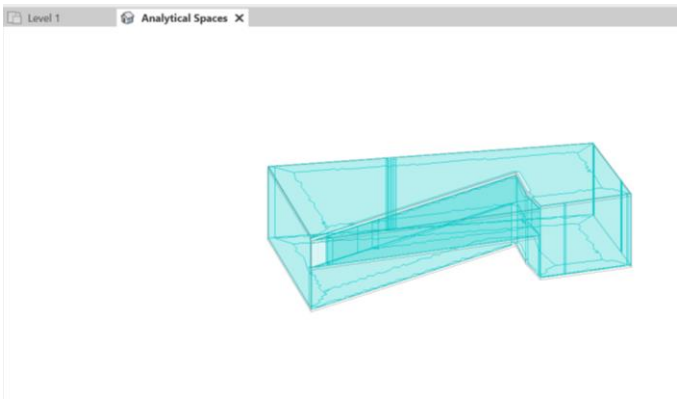


FIGURA 49. FINESTRA D'ANALYTICAL SPACES DESPRÉS DE CREAR L'ENERGY MODEL

Se'ns presenten dos mètodes per exportar el model al format gbXML (Figura 50), o bé "Use Energy Settings" o "Use Room/Space Volumes"(Figura 51).

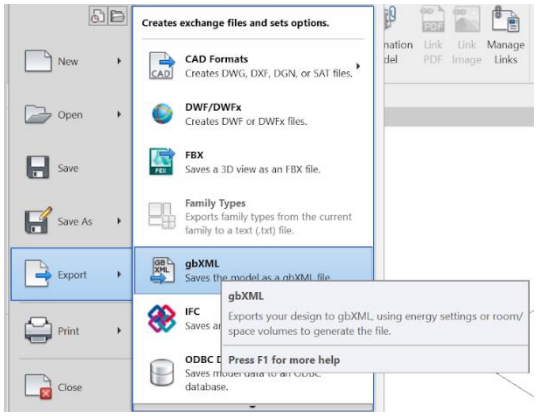


FIGURA 50. OPCIÓ D'EXPORTAR CAP A GBXML A AUTODESK REVIT

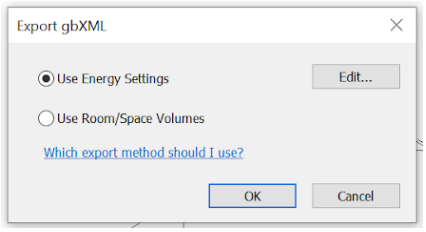


FIGURA 51. MÈTODES PER EXPORTAR EL MODEL AL FORMAT GBXML

Utilitzem l'opció de "Use Energy Settings" tot modificant en cas que sigui necessari els paràmetres per defecte (veiem que alguns ja ens van bé, per exemple el *Ground Plane* es el *Level 1* que es on hem situat la base de l'edifici (Figura 52)), cliquem a "Ok" i obtenim el model en format gbXML. S'utilitza aquesta opció de "Use Energy Model" per exportar cap a gbXML ja que aquest mètode tot i no necessitar les *Rooms* ni els *Spaces* creats, si estàn creats els identifica i també els te en compte

a l'hora d'exportar cap a gbXML, a més, aquesta opció crea un gbXML més acurat que l'opció de "Use Room/Space Volumes" (54).

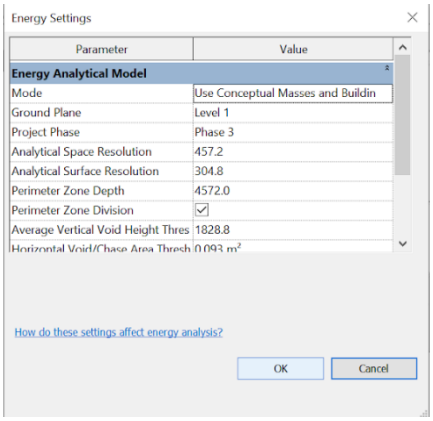


FIGURA 52. FINESTRA DE CONFIGURACIÓ DE PARÀMETRES DEL MÈTODE "USE ENERGY SETTINGS"

4.2.5. gbXML a IDF. OpenStudio

Finalment hem de generar el fitxer IDF d'Energy Plus corresponent a aquest gbXML obtingut amb Autodesk Revit. Per a poder obtenir aquest fitxer IDF fem ús de l'aplicació OpenStudio.

Obrim l'applicació d'OpenStudio, en ella creem un nou projecte, i a la secció de *"Geometry"* ens ubiquem a la pestanya de *"Editor",* allà escollim com a opció de *"Geometry Type"* el gbXML (Figura 53).
Fem l'import del gbXML de la BRGF obtingut amb l'Autodesk Revit i un cop fet cliquem sobre el *"Merge with current OSM"*, canviem de pestanya cap a la de *"3D View"* per poder veure la biblioteca en 3D (Figura 54).
Un cop hem vist que l'import s'ha efectuat correctament ja podem exportar el model de la BRGF cap al format IDF d'Energy Plus anant a *"File" -> "Export" -> "IDF"*.
La versió del fitxer IDF obtingut correspon amb la versió 9.3 d'IDF d'Energy Plus.

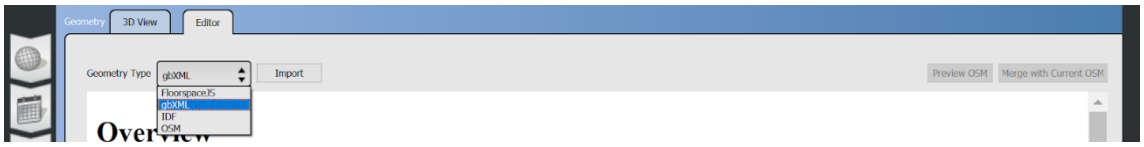


FIGURA 53. SELECCIÓ DE GBXML COM A GEOMETRY TYPE A LA FINESTRA EDITOR A OPENSTUDIO

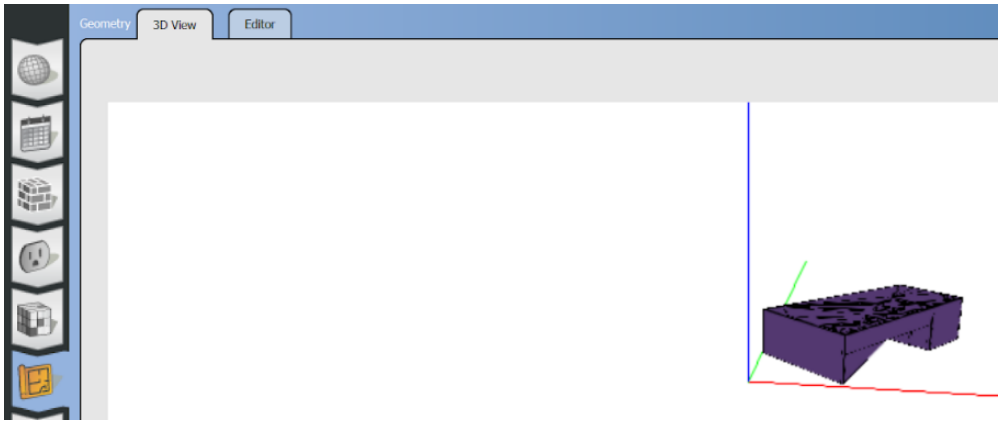


FIGURA 54. FINESTRA VISUALITZACIÓ 3D DEL MODEL EN LA SECCIÓ DE GEOMETRY A OPENSTUDIO

4.2.6. DXF a IDF i gbXML. SketchUp

Amb SketchUp 2019 Pro i el plugin disponible per aquest d'Open Studio s'ha realitzat el desenvolupament de la geometria bàsica simplificada del DXF de CADMapper ja comprovat i després s'hi han afegit els elements necessaris per poder crear un BEM de tal forma que pugui ser acceptat per executar una simulació a Energy Plus. A continuació s'explica aquest camí alternatiu respecte al explicat anteriorment mitjançant Autodesk Revit.

Primerament el que fem es crear la geometria a SketchUp ja de tal forma que sigui ben interpretada per Energy Plus, per fer-ho primerament importem el model DXF de la BRGF a SketchUp (Figura 55).

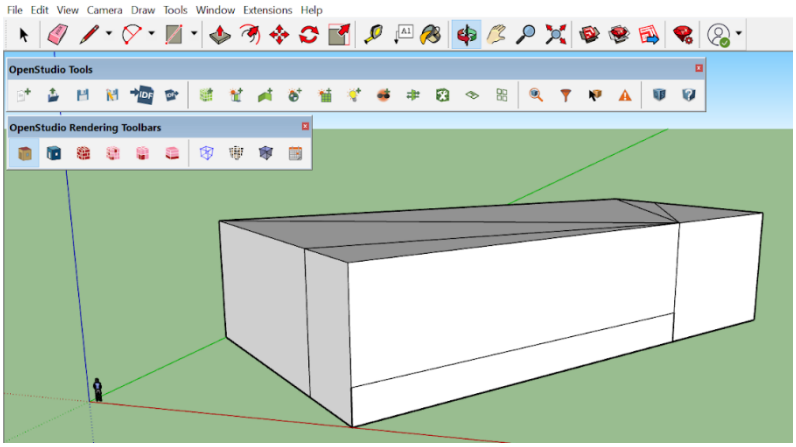


FIGURA 55. DXF DE LA BRGF IMPORTAT A SKETCHUP

Un cop importat amb l'eina de dibuix "Line" (encerclada en vermell a la Figura 56) creem el polígon que forma la base de la biblioteca (sense el porxo), el resultat es mostra a la Figura 57.

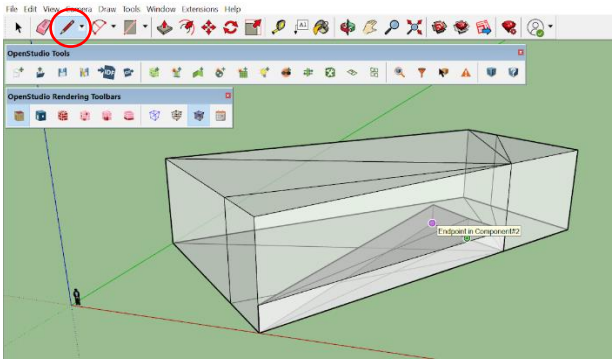


FIGURA 56. AMB L'EINA DE DIBUIX "LINE" CREEM EL POLÍGON QUE FORMA LA BASE DE LA BIBLIOTECA

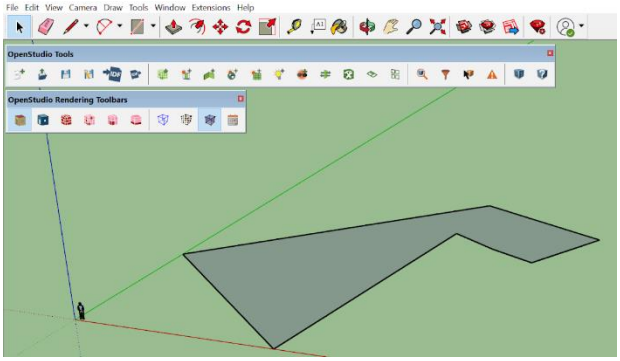


FIGURA 57. POLÍGON REPRESENTANT LA BASE DE LA BRGF

Un cop tenim ja la base creada ja podem seleccionar-la i utilitzar la eina de “Create Spaces from diagram” del Open Studio plugin (encerclada en vermell a la Figura 58) i creem l’*space* amb l’altura de 12 metres i senzillament un *floor* (una planta) (es un *simple box model*) (Figura 59).

FIGURA 58. EINA “CREATE SPACES FROM DIAGRAM” DEL OPEN STUDIO PLUGIN

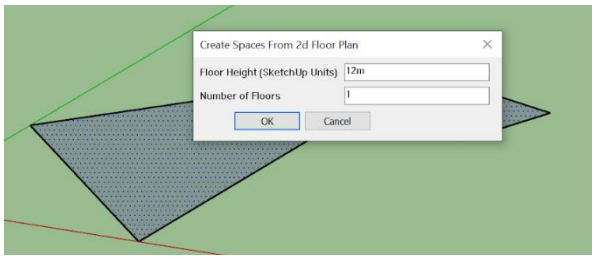


FIGURA 59. FINESTRA DE CONFIGURACIÓ DE L'EINA "CREATE SPACES FORM DIAGRAM"

Obtenint el resultat següent (Figura 60):

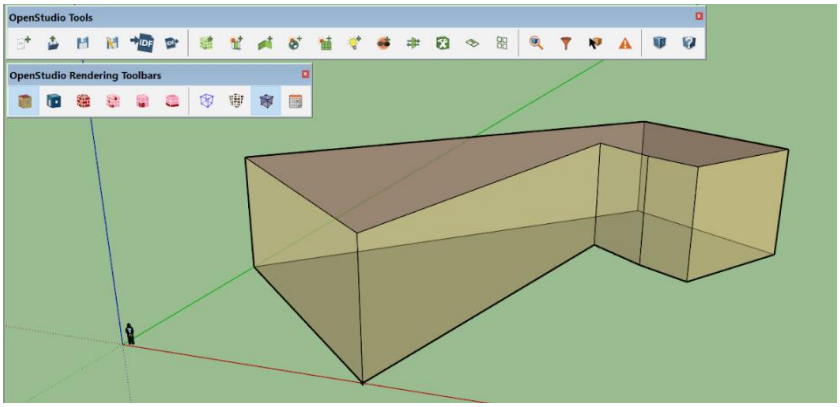


FIGURA 60. MODEL SIMPLIFICAT GEOMETRICAMENT DE LA BRGF A SKETCHUP

Si ens fixem i utilitzem l’opció de “Render By Boundary Spaces” veiem com totes les parets inclòs el sostre tenen frontera amb l’exterior, menys el terra que està de color crema que té frontera amb el propi sol (Figura 61).

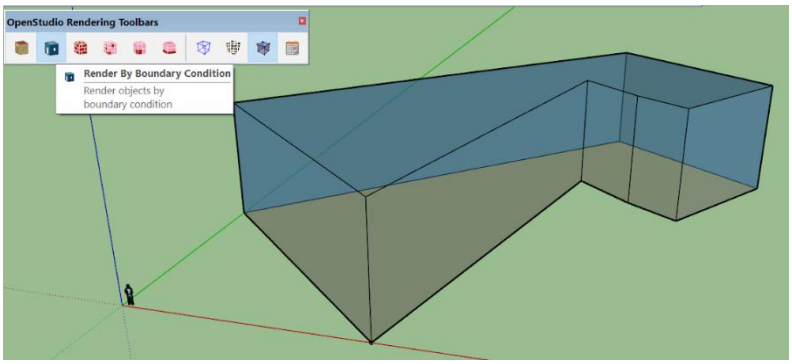


FIGURA 61. OPCIÓ DE “RENDER BY BOUNDARY SPACES” DE OPENSTUDIO PLUGIN

Ara però falta afegir els elements necessaris per tal de poder ser executat en una simulació a Energy Plus, és a dir perquè esdevingui un BEM simplificat de la biblioteca. Per fer-ho aquest model ha de tenir els elements bàsics d'un BEM, un *space* i una *zone* associada a aquest.

El primer que fem es comprovar utilitzant l'eina OpenStudio Inspector, que el model conté l'space creat, però podem comprovar que en l'apartat de "OpenStudio HVAC" no hi tenim cap ThermalZone definida (Figura 62, senyalat en vermell tant l'*space* com el *thermal zone*).

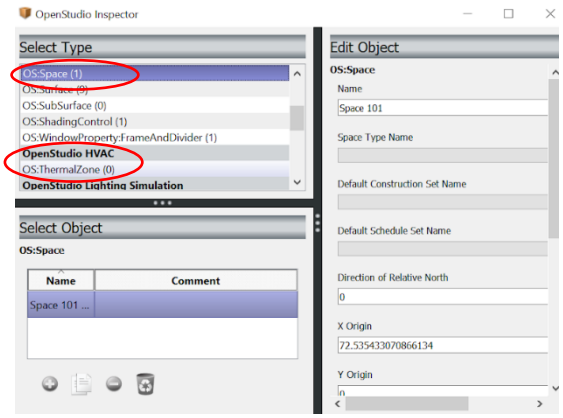


FIGURA 62. OPENSTUDIO INSPECTOR ON ENCARA NO HI HA CAP THERMALZONE DEFINIDA

Per tant, el que hem de fer es utilitzar l'opció de "Add New Thermal Zone For Spaces With No Thermal Zone" per tal d'afegir una *thermal zone* a l'*space* ja creat (Figura 63).

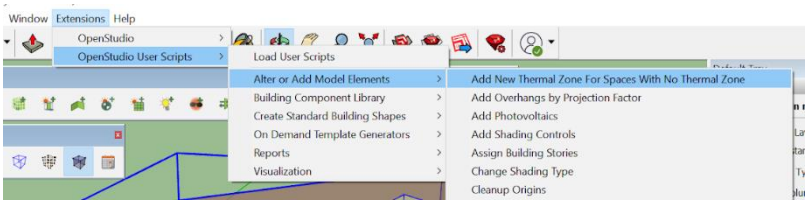


FIGURA 63. OPCIÓ DE "ADD NEW THERMAL ZONE FOR SPACES WITH NO THERMAL ZONE"

Ara ja podem comprovar en el OpenStudio Inspector que ja hem definida la *thermal zone*. El problema però es que aquesta *thermal zone* no està especificat a quin *space* correspon en el seu nom (Figura 64), en aquest cas només en tenim un, però se li ha d'especificar a quin *space* correspon per tal de generar el model correctament i el motor de simulació l'entengui, per tant, per fer-ho utilitzem l'eina "Raname Thermal Zones Based On Space Names" (Figura 65).

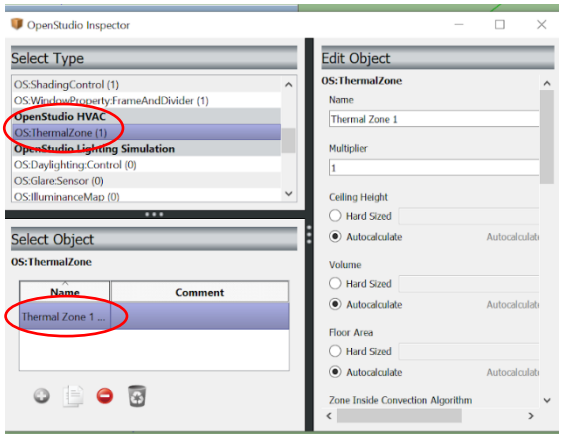


FIGURA 64. OPENSTUDIO INSPECTOR ON THERMAL ZONE NO ESTÀ ESPECIFICAT A QUIN SPACE CORRESPON

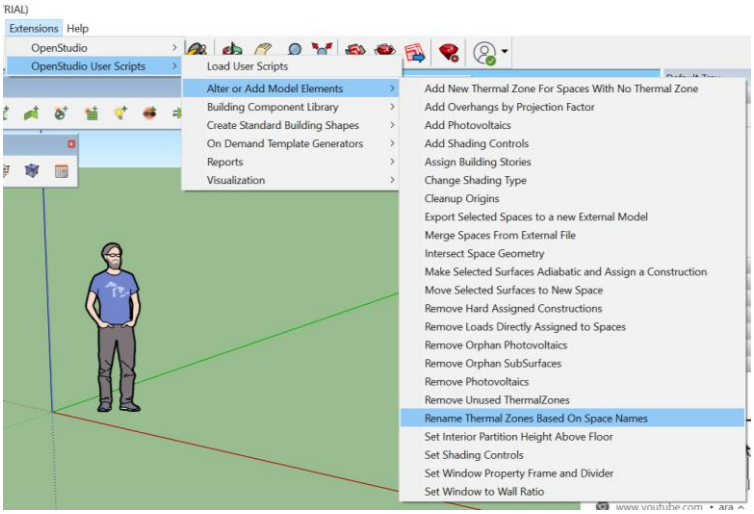


FIGURA 65. OPCIÓ "RANAME THERMAL ZONES BASED ON SPACE NAMES"

Ara ja podem comprovar a OpenStudio Inspector que la Thermal Zone definida ja te el mateix nom que l'space al qual correspon (Figura 66).

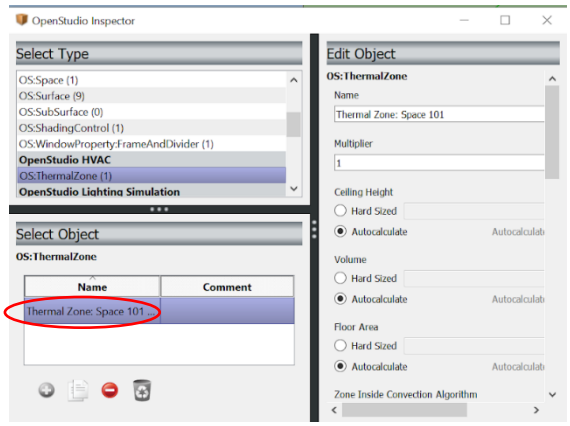


FIGURA 66. OPENSTUDIO INSPECTOR QUE LA THERMAL ZONE DEFINIDA JA TE EL MATEIX NOM QUE L'SPACE AL QUAL CORRESPON

El següent pas es assignar el tipus d'espai que representa aquest model. Per fer-ho utilitzem l'opció de "Space Type and Construction Set Wizard" (Figura 67).

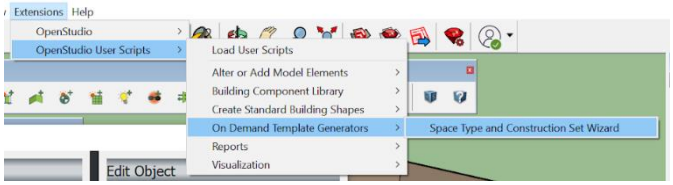


FIGURA 67. OPCIÓ DE "SPACE TYPE AND CONSTRUCTION SET WIZARD"

El que més se li apropa es el tipus "Office" ja que no hi ha la "Library" disponible i escollim una Climate Zone. Definint el que ofereix aquesta opció (Figura 68) ja ens defineix spaces types, Construction sets, etc. tot el necessari per en un futur poder continuar afegint-li complexitat al model i tanmateix poder executar una simulació amb el model simplificat a Energy Plus.

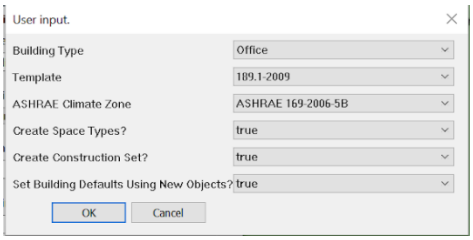


FIGURA 68. FINESTRA DE "SPACE TYPE AND CONSTRUCTION SET WIZARD"

Ara ja podem exportar cap al format IDF amb l'opció "Export Energy Plus idf" (també ens ofereix l'opció de gbXML com a Revit, interessant si es vol importar a algun altre software de modelat) (ambdues opcions encerclades en vermell a la Figura 69).

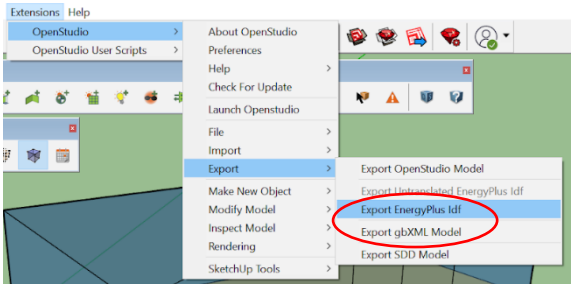


FIGURA 69. OPCIONS "EXPORT ENERGYPLUS IDF" I "EXPORT GBXML MODEL" A SKETCHUP

4.2.7. IDF a epJSON. EnergyPlus

L'últim pas del procés de l'obtenció del BIM de la BRGF en format IDF i epJSON es el pas relacionat amb l'obtenció del model de la BRGF en format epJSON. Si partim de l'IDF obtingut anteriorment es força més ràpid i senzill, ja que la definició del model ja la contindrà el fitxer en IDF i EnergyPlus dona les eines necessàries per a fer la transformació del format IDF a epJSON amb facilitat.

Tal i com es veu a la Figura 70, si executem la comanda *>energyplus.exe -h* veiem que hi trobem una opció *-c, --convert* (encerclada en vermell a la Figura 70) que ens informa que si la utilitzem podem generar la conversió de IDF cap a epJSON i a la inversa segons l'input que rebi.

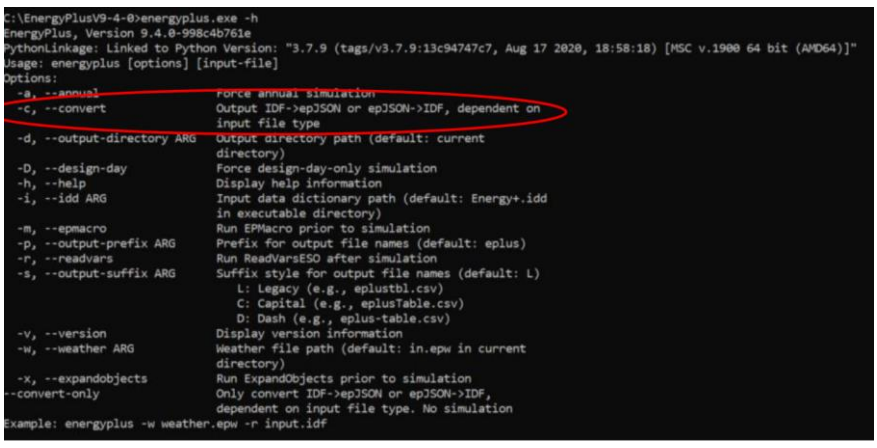


FIGURA 70. EXECUCIÓ COMANDA ENERGYPLUS.EXE -H QUE ENS PERMET VEURE OPCIONS D'EXECUCIÓ

Per efectuar la conversió de IDF cap a epJSON s'ha d'executar una simulació a EnergyPlus amb el *flag –convert* mencionat, per tant, com ja s'ha comentat en els anteriors apartats de desenvolupament del model a aquest ja no li hem de fer cap més modificació perquè ja l'hem preparat perquè es pogués executar simulacions sobre ell.

Per efectuar la conversió de IDF cap a epJSON mitjançant l'execució d'una simulació a EnergyPlus amb aquest flag mencionat cal que la versió de l'IDF o el epJSON el qual es vulgui transformar estigui

en la mateixa versió que l'EnergyPlus sobre el que s'executa la comanda. En el cas de que la versió de l'IDF obtingut en un inici no correspongui amb la versió descarregada d'EnergyPlus, aquest ofereix una eina de transició de versió cap a la que té l'EnergyPlus descarregat amb l'opció *Transition version* (Figura 71).

Aquesta utilitat de conversió pot trigar entre un i tres minuts per convertir un edifici definit per exemple amb v7.0 a la versió v8.5. Utilitzant tècniques més modernes, com ara l'aplicació d'esquemes i la validació proporcionada per JSON i JSON Schema, aquest procés de conversió es pot efectuar molt més ràpid.

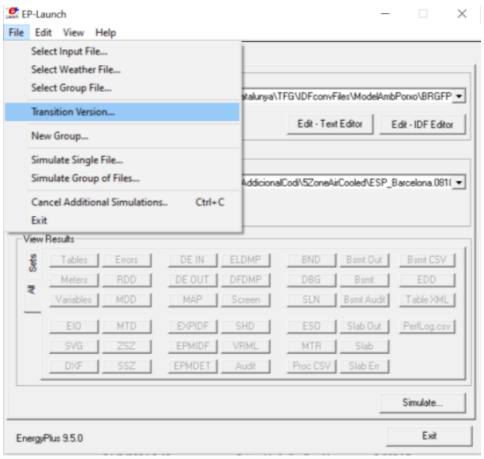


FIGURA 71. OPCIÓ DE TRANSITION VERSION A ENERGYPLUS

Cal fer notar que EnergyPlus fins la versió 10.0 donarà suport tant a la conversió de models en format IDF cap a epJSON i si es necessari també pot transformar la versió d'IDF d'entrada cap a la versió d'EnrgyPlus que s'estigui utilitzant (tal com es menciona en l'apartat 1.3.2). Per tant, es interessant que tot i que softwares com NECADA puguin continuar processant fitxers en format IDF de projectes *legacy* aquests es vagin convertint cap a format epJSON, tot fent l'actualització de l'IDF amb l'opció *Transition version* mencionada en cas de necessitat per fer la conversió, ja que com es detallarà en l'apartat 4.3 aquest nou format aporta beneficis en molts aspectes. A la Figura 72 es mostra un exemple del procés a seguir on la versió d'EnergyPlus es la 9.5 i com els projectes amb format IDF segons la seva versió es poden convertir a epJSON per obtenir els avantatges que aquest format ofereix. D'aquesta forma NECADA podrà treballar amb les noves versions d'EnergyPlus aprofitant així els seus avantatges sense que hi hagi projectes que quedin enrere i desactualitzats.

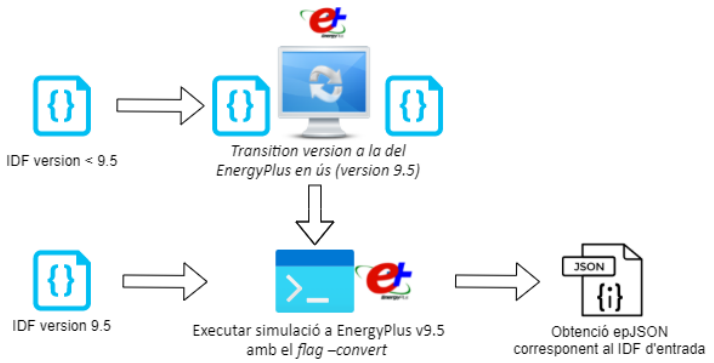


FIGURA 72. EXEMPLE DEL PROCÉS A SEGUIR ON LA VERSIÓ D'ENERGYPLUS ES LA 9.5 I COM ELS PROJECTES AMB FORMAT IDF SEGONS LA SEVA VERSIÓ ES PODEN CONVERTIR A EPJSON

4.3. Aplicació de consola

4.3.1. Introducció

L'aplicació de consola que s'ha desenvolupat en aquest projecte te la funció de poder importar la informació que interessa a NECADA a la seva base de dades en dos formats diferents de fitxer d'entrada: IDF i epJSON, tot tenint en compte que també s'haurà de gestionar l'exportació de modificacions cap a aquest nou fitxer. L'usuari decidirà en el moment d'execució quin format de fitxer desitja importar (Figura 73) tal i com ja s'ha introduït en l'apartat 1.3.2 en la Figura 13.

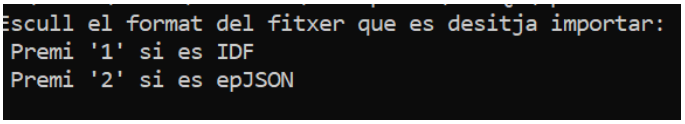


FIGURA 73. CAPTURA DE L'EXECUCIÓ DE L'APLICACIÓ DE CONSOLA EN EL MOMENT DE DECIDIR EL FORMAT

En aquest projecte ens centrem sobretot en l'exploració del procés d'importació del format epJSON i en com adaptar el sistema per tal de que es puguin gestionar aquests dos tipus de formats de forma correcte, ja que Optisim ja gestiona fitxers en format IDF. Pel que fa a la part de gestió del epJSON l'aplicació de consola rebrà el nom i path complet d'un fitxer en format epJSON on amb una llibreria de JSON per C# es llegirà el contingut del fitxer, es validarà amb un esquema, s'extrauran els materials i solucions constructives i es guardarà tot en una nova base de dades.

Per a realitzar l'aplicació de consola es crearà el que s'anomena dins l'entorn de Visual Studio un projecte. Quan creem una aplicació o un lloc web a Visual Studio, comencem amb un projecte. Un projecte conté tots els fitxers que es compilen en un executable, llibreria o lloc web. Aquests fitxers poden incloure codi font, icones, imatges, fitxers de dades, etc. Un projecte també conté paràmetres del compilador i altres fitxers de configuració que poden necessitar diversos serveis o components amb els quals el nostre programa es comuniqui.

Si ens fixem en la part inferior de la Figura 74 podem observar un fitxer anomenat NECADA.sln. Aquest fitxer es el que s'anomena en l'entorn de Visual Studio una solució. Un projecte es troba dins d'una solució. Malgrat el seu nom, una solució no és una "resposta", simplement és un contenidor per a un o més projectes relacionats, juntament amb informació de compilació, configuració de la finestra de Visual Studio i qualsevol fitxer divers que no estigui associat a un projecte concret. La aplicació de consola que es desenvolupa en aquest TFG es compartirà com un projecte de la solució de NECADA.

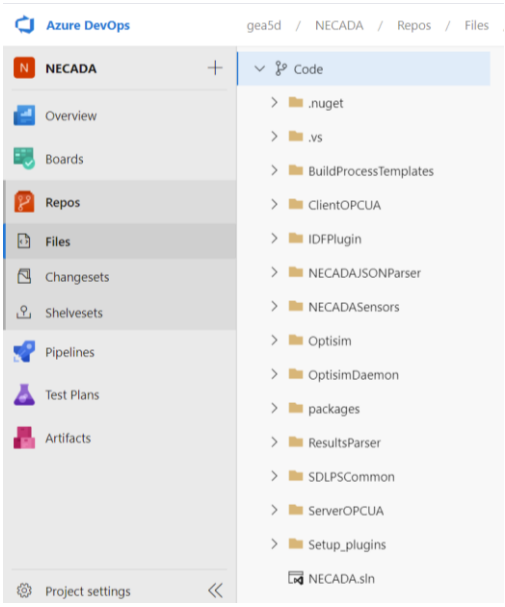


FIGURA 74. REPOSITORI DE CODI AZURE DEVOPS DE NECADA

Aquest nou projecte creat es del tipus “Console App (.NET Framework)” ja que si es crees com 'Net Core' (l’altre possibilitat que ofereix Visual Studio d’una aplicació de consola en C#) tindria incompatibilitats per adaptar aquest codi a Optisim web (NECADA).

Per començar a adaptar el sistema perquè pogués acceptar el format epJSON primerament es va dur a terme una preparació prèvia on partint del codi d’Optisim disponible en el repositori de codi d’Azure DevOps de NECADA (Figura 74) es va crear una aplicació de consola que permetia importar un fitxer IDF. Tot fent diversos canvis en el codi del projecte Optisim es van copiar les classes vinculades que formen el model de dades i tots els fitxers representant les migracions al nou projecte i executar-les sobre una nova base de dades, de tal forma que sobre aquesta aplicació de consola es poguessin fer les modificacions de les classes necessàries i generar les corresponents noves migracions necessàries per poder adaptar el sistema a acceptar el format epJSON.

Per a fer tota la gestió en local de la nova base de dades s’ha utilitzat l’SQL Server Management Studio (SSMS) (55). SSMS és un entorn integrat per gestionar qualsevol infraestructura SQL, des de SQL Server fins a Azure SQL Database. SSMS proporciona eines per configurar, supervisar i administrar instàncies de SQL Server i bases de dades. El framework ORM (Object Relational Mapping) emprat es el Entity Framework (EF), a continuació es fa un breu resum d’aquest ja que en el desenvolupament se’n farà referencia a algunes de les seves característiques utilitzades.

EF és un framework ORM open-source per a aplicacions .NET suportades per Microsoft. Permet als desenvolupadors treballar amb dades mitjançant objectes de classes específiques de domini sense centrar-se en les taules i columnes de base de dades subjacents on s’emmagatzemen aquestes dades. Amb l'Entity Framework, els desenvolupadors poden treballar amb un nivell d'abstracció superior quan tracten amb dades així eliminant complexitat en la gestió de la base de dades. La següent figura (Figura 75) il·lustra on s’ubica Entity Framework dins d’una aplicació.

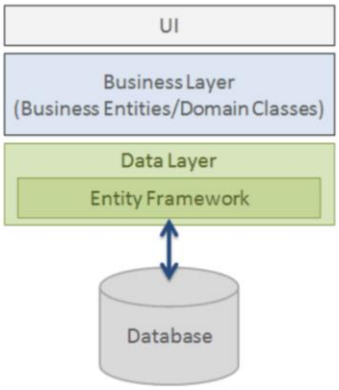


FIGURA 75. EN VERD LA UBICACIÓ D'EF DINS UN SISTEMA (56)

Les característiques principals d'EF que s'aprofiten en aquest projecte son:

1) Modelització: EF (Entity Framework) crea un EDM (Entity Data Model) basat en entitats POCO (Plain Old CLR Object) amb propietats get / set de diferents tipus de dades. EF utilitza aquest model quan ha de consultar o desar dades d'entitats a la base de dades subjacent.
2) Consultes: EF ens permet utilitzar queries LINQ (C # / VB.NET) per recuperar dades de la base de dades subjacent. El proveïdor de base de dades traduirà aquestes consultes LINQ al llenguatge de consulta específic de la base de dades (SQL per a una base de dades relacional). EF també ens permet executar consultes SQL directament des de el codi a la base de dades.
3) Seguiment de canvis: EF fa un seguiment dels canvis produïts en instàncies de les nostres entitats que cal enviar a la base de dades.
4) Desar: EF executa ordres INSERT, UPDATE i DELETE a la base de dades en funció dels canvis produïts a les nostres entitats quan es crida al mètode SaveChanges (). EF també proporciona el mètode asíncron SaveChangesAsync ().
5) Transaccions: EF realitza una gestió automàtica de transaccions mentre consulta o desa dades. També proporciona opcions per personalitzar la gestió de transaccions.
6) Migracions: EF proporciona un conjunt d'ordres de migració que es poden executar a la NuGet Package Manager Console o a la Command Line Interface per crear o gestionar l'esquema de base de dades que tinguem.

4.3.2. Format IDF

Actualment l'usuari quan utilitza el sistema NECADA defineix i introdueix el BIM de l'edifici del qual es volen executar simulacions i que serveix com a principal fitxer d'entrada per a dur-les a terme amb el format IDF (Input Data File), cal tenir en compte les seves característiques bàsiques per tal de contextualitzar millor les millores que proporciona el nou format epJSON.

Els format IDF pretén ser un format que sigui fàcilment llegible per humans, principalment per finalitats de desenvolupament i depuració. La majoria d'usuaris d'Energy Plus mai veuen un IDF, interactuen amb EnergyPlus mitjançant una interfície gràfica d'usuari (GUI), que escriu el IDF per a ells. En el cas de NECADA, l'usuari final un cop insereixi l'IDF a través de l'aplicació web aquest ja no es podrà modificar. No obstant això, un desenvolupador de mòduls és un tipus d'usuari especial. El desenvolupador del mòdul haurà de crear una porció d'un IDF manualment molt aviat en el procés de desenvolupament per començar a provar el mòdul en desenvolupament. Per tant, és important entendre la sintaxi IDF i utilitzar comentaris per crear fitxers IDF llegibles.

L'IDF es un fitxer amb un format com el CSV (Comma Separated Variable). L'ordre en el que estan situats els diferents elements que formen aquest fitxer no és important i per tant les dades poden aparèixer en qualsevol ordre i es recuperaran i ordenaran segons sigui necessari mitjançant la simulació EnergyPlus per mòduls (la diferència més gran entre EnergyPlus i el processament de fitxers d'entrada més tradicional és que EnergyPlus és modular en el seu emplenament real dels

detalls de la simulació. A causa d'aquesta estructura modular d'EnergyPlus, cada mòdul és responsable d'"obtenir" la seva pròpia entrada).

A continuació es detallen algunes de les característiques principals del format IDF i un petit fragment d'un exemple d'aquest tipus de fitxer (Figura 76):

- La línia inicial d'un objecte d'entrada ha de tenir una coma o un punt i coma.
- Els camps no s'estenen per sobre dels límits de la línia. Normalment, si una coma o un punt i coma (si escau) no és l'últim valor de camp d'una línia, s'inserirà un. Per descomptat, poden aparèixer diversos camps en una sola línia sempre que estiguin separats per comes. (I l'últim podria anar seguit d'un punt i coma).
- Les comes delimiten els camps; per tant, cap camp pot tenir comes incrustades. No es produirà cap error, però no s'obtindrà el resultat desitjat.
- Es permeten línies en blanc.
- El caràcter de comentari és una exclamació "!". Qualsevol cosa que hi hagi en una línia després de l'exclamació s'ignora per al processament del fitxer.
- Els registres d'entrada (també coneguda com a longitud de línia d'entrada) poden tenir una longitud de fins a 500 caràcters. Si s'excedeix aquesta longitud, no es produirà cap error, però no obtindrem el que volem.
- Cada cadena alfa del fitxer de dades d'entrada pot tenir una longitud de fins a 100 caràcters. Tot allò més enllà es veu truncat. Això s'aplicaria a tots els noms (com ara el nom de la zona, el nom del node, el nom del schedule, ...) dels camps.
- Cada cadena alfa (incloses les paraules clau de secció i de classe / objecte) es mapeja a majúscules durant el processament, tret que el flag "retaincase" marqui el camp a l'IDD (Input Data Dictionary). L'inconvenient principal d'això és que els missatges d'error que surten del processador d'entrada es mostraran en majúscules i minúscules i poden no aparèixer exactament com a l'entrada.
- Els caràcters especials, com ara els tabuladors no s'han d'incloure al fitxer. No obstant això, els tabuladors per exemple es poden allotjar i es poden convertir en espais.
- Tots els números es poden introduir de manera flexible i es processen en variables de precisió individuals (és a dir, 1.0, 1.000, 1, .1E + 1 es processen per igual).

```
BuildingSurface:Detailed,
  Building_Roof,            !- Name
  Roof,                     !- Surface Type
  IEAD Non-res Roof,        !- Construction Name
  TopFloor_Plenum,          !- Zone Name
  Outdoors,                 !- Outside Boundary Condition
  ,                         !- Outside Boundary Condition Object
  SunExposed,               !- Sun Exposure
  WindExposed,              !- Wind Exposure
  AutoCalculate,            !- View Factor to Ground
  4,                        !- Number of Vertices
  49.9110,0.0000,11.8872,   !- X,Y,Z ==> Vertex 1 {m}
  49.9110,33.2738,11.8872,  !- X,Y,Z ==> Vertex 2 {m}
  0.0000,33.2738,11.8872,   !- X,Y,Z ==> Vertex 3 {m}
  0.0000,0.0000,11.8872;    !- X,Y,Z ==> Vertex 4 {m}
```

FIGURA 76. EXEMPLE DE FORMAT IDF ON DEFINEIX UN OBJECTE (PRIMERA LINIA), SEGUIT DE LES PROPIETATS D'AQUEST OBJECTE

L'Input Data Dictionary (IDD) és un fitxer ASCII (text) que conté una llista de tots els possibles objectes EnergyPlus i una especificació de les dades que requereix cada objecte, per tant, el fitxer IDD és el que marca la sintaxi acceptada en l'IDF. Si hi ha canvis en aquest fitxer IDD en les noves i diferents versions, caldrà anar modificant els diferents sistemes que com NECADA necessiten llegir els diferents elements de l'IDF per ser guardats en la base de dades pertinent per poder adaptar-se als canvis i poder-los reflectir.

4.3.3. Motiu de la transició d'IDF cap a epJSON

Els motius principals pels quals EnergyPlus va decidir cercar una alternativa al format IDF son els següents:

1) Dificultat de parsejat
   - E+ parser personalitzat.
   - Cal tornar a implementar el parsejat d'E + per a qualsevol eina de tercers (NECADA).
2) Els camps son referenciats per índex.
3) Només pot tenir un grup de camps extensibles per objecte.
   - Per exemple, *BuildingSurface:Detailed* només pot tenir *vertices.*
4) És difícil validar IDF mitjançant IDD sense executar EnergyPlus (cal implementar validador).
5) Prioritat de l'equip EnergyPlus i del Departament d'Energia (DOE) dels USA.
   - La necessitat de fer el canvi estava dins el *Top* 10 de suggeriments del usuaris.
   - Proporcionarà una millor estructura del codi intern i una millor manteniment.

Les principals millores de les quals es beneficia EnergyPlus (i per tant tots els softwares de tercers que l'utilitzen) a l'hora de substituir IDF per epJSON son les següents:

Avantatges vinculats amb el fitxer d'entrada epJSON:

- Basat en la clau / valor, no posicional.
- Camps extensibles de longitud il·limitada.
- Gairebé tots els idiomes admeten el parsejat de JSON.
- Fàcil d'afegir i eliminar camps.
- Acceleració en el processament del fitxer d'entrada de '1,6x a 5,4x.

Avantatges vinculats amb el epJSON Schema (automàticament generat a partir de l'IDD):

- La majoria de llenguatges admeten la validació mitjançant un esquema JSON.
- No cal escriure un validador personalitzat.
- En el futur permetrà facilitar validacions més complexes a mesura que els fitxers representant un BEM d'un edifici que es fan servir d'entrada al sistema cada cop continguin més informació i complexitat.

Tal i com indica un estudi titulat *EnergyPlus Performance Improvements Via JSON Input Refactoring* (57), un refactor important de l'InputProcessor d'EnergyPlus (comentat en el següent apartat 4.3.4. Format epJSON (JSON)) per utilitzar un nou format d 'entrada JSON proporciona una velocitat general de 1,6x a 5,4x. Aquesta gran velocitat prové d'un parsejat del fitxer d'entrada més ràpid, de la reducció de la complexitat algorítmica de les funcions d'InputProces, i de la reducció completa de la necessitat verificar que els noms son únics deguts al compliment de la pròpia estructura JSON.

En aquest mateix estudi es conclou que gracies a aquest refactor mencionat es produeixen unes millores de rendiment en fitxers de mostra on es produeix una reducció del 14% en les lectures d'schemas i una reducció del 60% en les lectures dels fitxers d'entrada, una reducció del 36-62% a l'hora de processar el fitxer d'entrada i entre un 24-99% de reducció de les consultes per a funcions que fan servir el nou format. A més, utilitzant JSON i JSON Schema, els usuaris poden utilitzar un gran nombre de llenguatges per crear, manipular i validar fitxers d'entrada epJSON. En utilitzar un format d'entrada modern i estandarditzat, EnergyPlus està ben posicionat durant els propers 20 anys de desenvolupament facilitant als softwares que utilitzin EnergyPlus aprofitar aquest bon posicionament.

Aquestes millores en rendiment s'aniran comentant al llarg dels següents apartats de forma més detallada.

4.3.4. Format epJSON (JSON)

EnergyPlus va fer esforços de refactorització al voltant del InputProcessor EnergyPlus per donar suport de forma nativa a JavaScript Object Notation (JSON), permetre la validació mitjançant l 'ús d'un validador d'esquemes JSON, convertir i validar els fitxers típics de fitxers de dades d'entrada EnergyPlus (* .idf) a fitxers JSON i millorar mètriques relacionades com ara el temps de lectura per als fitxers d'esquema i entrada, analitzar el temps per processar l'entrada i el temps per consultar / retornar dades de l'estructura de dades interna d'EnergyPlus.

JSON (JavaScript Object Notation) és un estàndard obert basat en text dissenyat per un intercanvi de dades llegible per humans basat en els estàndards *RFC 7159* i *ECMA-404* (58). Deriva del llenguatge script JavaScript, per a representar estructures de dades simples i llistes associatives, anomenades objectes transmetent aquests objectes de dades mitjançant parells de clau/valor. Actualment és el format de dades més utilitzat per a la comunicació asíncrona de navegador/servidor. Els fitxers JSON solen utilitzar l'extensió *.json i substitueixen en gran part el XML (eXtensible Markup Language). Actualment hi ha eines existents per llegir i manipular fitxers JSON en gairebé tots els llenguatges de programació.

EnergyPlus va investigar dos formats JSON per identificar-ne un de similar al format IDF per heredar la seva funció en el sistema. Un format analitza, consulta i valida ràpidament (Figura 77), mentre que l'altra (Figura 78) és molt més similar al format IDF (Figura 76). Finalment la necessitat de que el temps necessari per validar fos el més ràpid possible i la necessitat d'utilitzar un format que sigui adequat a l'esquema JSON, van fer que l'opció de la Figura 77 fos la que prosperés.

FIGURA 77. FORMAT EPJSON, FORMAT PER A L 'ANÀLISI RÀPID, LA CONSULTA I LA VALIDACIÓ. ELS OBJECTES ENERGYPLUS ARA SÓN CLAUS ARREL SEGUIDES DE CLAUS DE NOM I, FINALMENT, CLAUS DE NOM DE CAMP AMB ELS SEUS VALORS ASSOCIATS. ELS VALORS SÓN ELS MATEIXOS QUE ELS DE L'IDF DE LA FIGURA 78.

FIGURA 78. FORMAT EPJSON MÉS SIMILAR A L'IDF ORIGINAL QUE EL MOSTRAT EN LA FIGURA 77

En el tradicional *.idf les relacions entre camps de dades es defineixen per la seva posició, tal i com es mostra a la Figura 76. Amb el nou format utilitzat * .epJSON, les dades de nivell de camp d'un objecte EnergyPlus són definides per el parell clau/valor tal com es descriu a la Figura 77 i a l'esquema de la Figura 79. L'emmagatzematge Clau/valor té diversos avantatges:

1) Traducció entre versions és més fàcil.
2) Permet un ús més fàcil dels valors predeterminats, ja que no s'han de definir al fitxer d'entrada.
3) És més fàcil llençar-hi consultes.

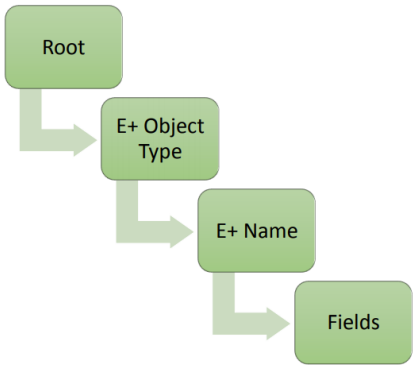


FIGURA 79. ESTRUCTURA PARELL CLAU-VALOR DE EPJSON (JSON)

Com ja s'ha mencionat, actualment hi ha eines per llegir i analitzar fitxers JSON en gairebé tots els llenguatges de programació, cosa que redueix el cost de manteniment de l'anàlisi personalitzat i programes complexos per llegir i/o traduir un fitxer d'entrada EnergyPlus. Els usuaris també poden afegir marcatge personalitzat a l'entrada fitxer per a ús de tercers o eines personalitzades; EnergyPlus ignorarà els camps de dades no definits pel diccionari de dades d'EnergyPlus.

EnergyPlus té el ja mencionat diccionari de dades d'entrada (IDD) que va ser traduït a un esquema epJSON. El JSON Schema s'utilitza per al processament de l'esquema epJSON per habilitar els usuaris i desenvolupadors de softwares de tercers validar el seu fitxer * .epJSON contra el * .schema.epJSON mitjançant qualsevol validador JSON en el llenguatge de programació que es faci servir. Mentre que aquesta validació detectarà qualsevol problema estructural, de tipus de dades, de conveni de

denominació de noms o d'interval de valors, no capturarà aspectes relacionats amb la lògica de l'edifici, és a dir, podríem dir que l'Schema epJSON validaria sintàcticament el BIM en epJSON però no semànticament, on si hi ha incongruències en aquesta última, aquestes seran capturades durant la validació quan EnergyPlus llegeixi el * .epJSON. Tot i que abans d'arribar a que EnergyPlus pugui detectar errors en el BEM hi ha diferents moments per trobar errors en aquest abans de ser obtingut en format IDF o epJSON, ja sigui en el propi programa BIM (e.g. Revit), programes especialitzats en BEM (e.g. IES-VE (59)) o en noves eines com el gbXML validator (60) desenvolupat com a part del *ASHRAE research project (RP-1810))*.

4.3.4.1. Refactorització del InputProcessor d'EnergyPlus

Com a modificació necessària per canviar el tipus de fitxer d'entrada a EnergyPlus, es va necessitar un parser modificat per traduir el nou fitxer d 'entrada a l'estructura d'emmagatzematge de la descripció de l'edifici que s'emmagatzemarà a la base de dades per executar amb èxit les simulacions. El processador d'entrada d'EnergyPlus (InputProcessor) es va reestructurar i optimitzar per augmentar el rendiment d'EnergyPlus en la tasca de parsejat.

Quan EnergyPlus s'executa, s'executa un mòdul l'anomenat InputProcessor. A causa de la naturalesa modular del motor de simulació EnergyPlus, cada mòdul és responsable d'obtenir les dades d'entrada que ha d'executar. Aquesta funció és realitzada per l'InputProcessor.

Per minimitzar el nombre de modificacions necessàries per adaptar el sistema al nou format de fitxer d'entrada, l'adaptació utilitza les mateixes signatures de les funcions del InputProcessor que les versions anteriors d'EnergyPlus. Per tant, l'InputProcessor només es va modificar per llegir, analitzar i consultar/retornar dades JSON, però les dades es proporcionen a cadascun dels mòduls EnergyPlus de la mateixa manera que les versions anteriors.

Dins del InputProcessor, una funció coneguda com a *GetObjectItem* és responsable de la cerca en l'estructura de dades interna d'EnergyPlus per retornar dades rellevants d'un objecte determinat. Una altra funció del InputProcessor, coneguda com a *VerifyName*, és responsable de verificar que cada tipus d'objecte EnergyPlus o grup de tipus tenen noms únics. Aquesta funció es va reescriure per acomodar el fitxer estructura addicional proporcionada per l'emmagatzematge per parell clau / valor de JSON.

Gràcies a l'adopció del format epJSON es va comprovar que la complexitat teòrica d'aquests dos mètodes utilitzats anomenats de *GetObjectItem* i *VerifyName* es va reduir d' O(n^2) a O (n) mitjançant un map no ordenat estàndard C ++, que ha amortitzat la cerca a O (1). Això pot tenir un fort efecte en temps d'execució al analitzar un edifici amb moltes superfícies o altres objectes. Les crides a *VerifyName* es van eliminar ja que la funció només comprovava que els noms eren únics dins del mateix tipus d'objecte EnergyPlus, i aquesta capacitat queda subsumida per l'estàndard JSON que aplica claus úniques (aspecte que també s'aprofita a l'hora de llegir i guardar els materials i solucions constructives del epJSON a l'aplicació de consola com veurem en els següents apartats).

4.3.5. Desenvolupament de l'aplicació de consola

En aquest apartat de la documentació es realitza una breu explicació de l'estructura de l'aplicació, del model de dades utilitzat i del codi tot seguint el seu flux d'execució.

4.3.5.1. Introducció

Per tal de posar les bases que serveixen per seguir l'explicació de codi i entendre'l amb més fluïdesa primerament es mencionen els canvis fets sobre el diagrama de classes de l'aplicació de consola inicial (mencionada en l'apartat 4.3.1. Introducció) que permet solament la importació del format IDF per tal d'adaptar-hi la importació d'un model en un format epJSON.

Ens centrem únicament en les classes i les seves corresponents taules a la base de dades que es vinculen i que se'n fan ús a l'hora d'importar un nou BIM a NECADA a través d'Optisim. El diagrama de classes complet de l'aplicació de consola inicial es mostra a la Figura 80.

Un cop es van analitzar les diferents classes i la seva funció pel que fa a la importació dels materials i solucions constructives que conté un fitxer IDF d'entrada es va concloure que per dur a terme l'aplicació de consola que permet la importació del seu contingut en els dos tipus de format de fitxer d'entrada ens havíem de fixar en les classes encerclades en vermell en la Figura 80 i analitzar quines modificacions calia fer per assolir l'objectiu.

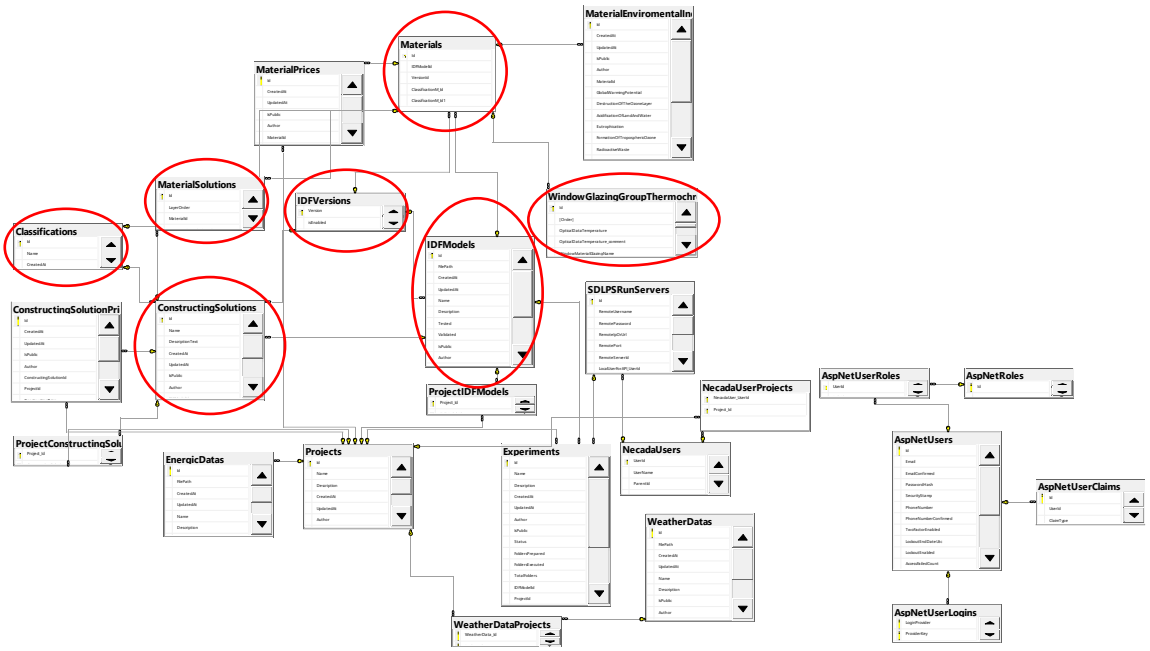


FIGURA 80. DIAGRAMA DE CLASSES DE L'APLICACIÓ DE CONSOLA INICIAL

La taula "IDFModels" emmagatzema informació general sobre el fitxer que conté el model BIM: *FilePath, CreatedAt, UpdatedAt, Name, Description, Tested, Validated, IsPublic, Author, GeneratedBy, IsUserModified, ExperimentId, ExecutionUnits i VersionId* que actua com a *foreign key*.

La taula "Materials" emmagatzema informació de tots els tipus de materials que conté el fitxer del model BIM juntament amb tots els seus possibles atributs, tot indicant com a *foreign key* a quin model, versió i classificació de material pertanyen. La taula "WindowGlazingGroupThermochromicMaterialDatas" va vinculada a un tipus especial de material que pot arribar a estar format per més d'un tipus de material i aquesta contindrà tota la informació vinculada a aquests indicant amb *foreign key* a quin material fan referència.

La taula "ConstructingSolutions" emmagatzema totes les solucions constructives que conté el fitxer del model BIM juntament amb tots els seus possibles atributs, tot indicant com a *foreign key* a quin model, versió i classificació de solució constructiva pertanyen. La taula "MaterialSolutions" emmagatzema tots aquells materials utilitzats en una solució constructiva, indicant amb *foreign key* a quin material i a quina solució constructiva fan referència i la taula "Classifications" conté la informació vinculada als dos tipus de classificacions existents que son "ClassificationMaterial" i "ClassificationConstructingSolution".

Finalment la taula "IDFVersions" emmagatzema les versions disponibles i suportades pel sistema d'IDF.

Les classes que es tradueixen a la base de dades coma a "Materials", "WindowGlazingGroupThermochromicMaterialDatas", "MaterialSolutions", "Classifications" i "ConstructingSolutions" no s'han modificat ja que son les que contenen la informació dels materials i solucions constructives, informació que es la mateixa tant per IDF com per epJSON, d'aquesta forma s'aprofiten les classes ja existents a Optisim.

Al no modificar aquestes darreres classes mencionades ha sigut necessari però modificar-ne d'altres, en concret la classe "IDFModel" i la classe "IDFVersion".

Si ens fixéssim únicament en la importació dels material i solucions constructives la classe "IDFModel" tal i com estava en un inici ja ens faria servei; però si ens fixem en que Optisim cal que permeti també la exportació tant cap a un fitxer IDF com en un epJSON aquí ja hauríem d'identificar a quin format pertany el model al qual hi volem exportar informació per tal de fer-la correctament.

Per solucionar aquest aspecte mencionat el que s'ha fet es canviar el nom d'aquesta classe "IDFModel" a "Model" (un nom més genèric que engloba els dos formats) i procedir a la creació de dues noves subclasses anomenades "IDFModel" i "EpJSONModel".

Actualment EF permet tres tipus d'herència diferents: Table Per Hierarchy (TPH), Table Per Type (TPT) i Table Per Concrete Type (TPC) (61).

Quan es va desenvolupar Optisim l'únic tipus d'herència que suportava EF era la TPH. Actualment totes les classes del model que utilitzen herència es mapejen a la base de dades d'acord amb el concepte de TPH.

En la TPH s'utilitza una taula per representar totes les classes de la jerarquia. S'utilitza una columna "discriminator" per discriminar entre diferents tipus, aquesta taula pren el nom de la classe base o de la propietat DbSet associada per defecte.

Per haver de fer les menors modificacions possibles del codi de l'aplicació de consola cap a Optisim s'opta per utilitzar aquesta tipologia d'herència també per solucionar el problema esmentat de la taula "IDFModels", a més aquesta tipologia d'herència per aquesta classe en concret ens escau molt bé ja que tots els atributs de les subclasses son compartits i per tant no tindrem una taula amb camps superflus en una tupla d'una determinada subclasse. El que obtenim llavors es una taula "Model" que conté exactament els mateixos camps que la seva successora "IDFModel" ja que aquests camps son compartits per ambdues subclasses ("IDFModel" i "EpJSONModel"), però amb un camp addicional *Discriminator* que podrà tenir o bé el valor IDFModel o bé EpJSONModel segons el format de fitxer del qual s'hagi obtingut el model en qüestió. A la Figura 81 es veu la taula "IDFModels" original i a la Figura 82 la nova taula "Models" amb el nou camp *Discriminator* a la part més inferior.

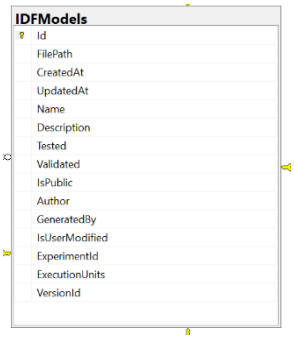


FIGURA 81. TAULA IDFMODELS ORIGINAL A SSMS

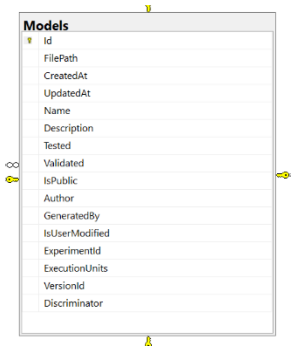


FIGURA 82. TAULA MODELS A SSMS

Pel que fa a la classe IDFVersion i la seva taula vinculada a la base de dades te la funció de informar al sistema de quines versions estan disponibles en aquest i per tant poder identificar quins fitxers IDF son suportats per Optisim i quins no segons la seva versió.

Quan entrem la nova variable de poder importar fitxers en un format diferent que son importats i validats de forma diferent es pot tenir el cas de que hi hagi versions suportades per IDF que no ho estiguin per epJSON i a la inversa.

Per solucionar aquest aspecte, al poder tenir versions que passin a no estar suportades pel sistema perquè l'administrador així ho decideixi o que un numero de versió en concret estigui disponible per ambdós formats, si entréssim en la possibilitat d'aplicar la mateixa solució que a "Models" no obtindríem el resultat desitjat, ja que en aquest cas teníem que un Model en concret o bé era IDFModel o bé EpJSONModel (*disjoint*) mentre que pel que fa a les versions ens trobem que es *overlapping*, és a dir, una mateixa versió pot estar disponible tant per IDF com per epJSON.

La solució per la que s'opta es no tractar la classe IDFVersion com si pogués tenir dos tipologies diferents, realment en té només una, la versió en concret que el sistema suportarà. A aquesta classe se li afegeixen dos nous atributs de tipus *bool* anomenats "SupportIDF" i "SupportEpJSON" que informaran de si aquella versió en concret està suportada per IDF, per epJSON, las dues o cap.

A més, Si es dissenya d'aquesta forma la gestió de versions passa a ser més simplificada de que si haguéssim optat per la creació de dues subclasses de IDFVersion tal i com s'ha realitzat amb Model.

A la Figura 83 es veu la taula "IDFVersions" original i a la Figura 84 es veu la mateixa taula on se li han afegit els dos nous atributs "SupportIDF" i "SupportEpJSON" explicats.

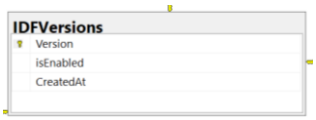


FIGURA 83. TAULA IDFVERSIONS ORIGINAL

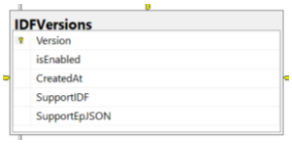


FIGURA 84. TAULA IDFVERSIONS AMB ELS DOS NOUS CAMPS AFEGITS

4.3.5.2. Migrations

Tal i com s'ha mencionat en l'apartat d'Entity Framework, aquest té la característica de gestionar migrations.

En els projectes del món real, els models de dades canvien a mesura que s'implementen noves funcionalitats: s'afegeixen i s'eliminen noves entitats o propietats i cal canviar els esquemes de base de dades en conseqüència per mantenir-los sincronitzats amb l'aplicació. La funció de migracions a EF proporciona una manera d'actualitzar de manera incremental l'esquema de la base de dades per mantenir-lo sincronitzat amb el model de dades de l'aplicació mentre es conserven les dades existents a la base de dades.

Resumint en un nivell alt, les migracions funcionen de la següent manera:

- Quan s'introdueix un canvi de model de dades, el desenvolupador utilitza les eines EF per afegir una migració corresponent que descriu les actualitzacions necessàries per mantenir l'esquema de la base de dades sincronitzat. EF compara el model actual amb una instantània del model antic per determinar les diferències i genera fitxers font de migració; es pot fer un seguiment dels fitxers creats a la carpeta Migrations que es crea dins el nostre projecte (Figura 85).
- Un cop s'ha generat una nova migració, es pot aplicar a una base de dades de diverses maneres. EF registra totes les migracions aplicades en una taula d'historial especial que li permet saber quines migracions s'han aplicat i quines no (per tant, quins canvis estan pendents i quins no).

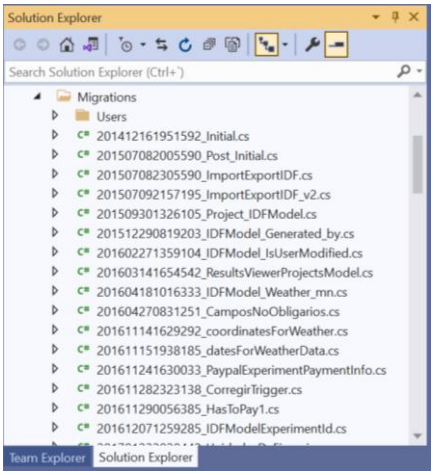


FIGURA 85. CARPETA MIGRATIONS DEL PROJECTE

Les noves migrations realitzades per adaptar la base de dades als canvis realitzats son les tres següents:

1) En la primera Migration es realitzada la gestió per els nous dos atributs de la classe IDFVersion. A més, s'actualitzen les 6 versions disponibles d'IDF fins el moment segons aquests nous dos atributs i s'insereix la versió 9.5 per epJSON (actualment la única versió disponible en el sistema ja que es la única de la qual es disposa del seu schema) tot fent servir la característica d'EF de poder utilitzar queries LINQ (C#).
2) La segona Migration es on es gestiona el Update del rename de la classe "IDFModel" a "Model".
3) En la ultima Migration afegida es on hi ha la informació vinculada a la creació de les noves dues subclasses "IDFModel" i "EpJSONModel". En forma de Discriminator a la taula "Models".

4.3.5.3. Gestió de versions

En el cas de que es disposi d'un fitxer en una versió que no estigui en el sistema encara que sigui anterior a la més actual suportada per aquest o estigui entre dues versions suportades, aquest fitxer no s'importarà ja que poden haver-hi canvis de requisits entre diferents versions dels fitxers BIM tant en IDF com en epJSON sobretot en la obligatorietat dels diferents camps dels materials i solucions constructives així com els seus noms i possibles valors que farien que la importació es realitzés incorrectament, per tant, si es vol importar un fitxer tant IDF com epJSON cal que aquella versió en concret estigui suportada pel sistema ja que serà amb aquella versió amb la que s'executarà la simulació.

La raó es que la validació pel que fa el fitxer epJSON es fa mitjançant un schema epJSON de validació proporcionat en la mateixa carpeta d'instal·lació d'EnergyPlus. Aquest schema només garantitza la correcte validació per la versió en concret amb la qual ha estat descarregat EnergyPlus. A més, Optisim té en aquest sentit una base de dades no dinàmica, és a dir, no té lògica de control per segons quina versió s'estigui important o exportant i identificar quins camps son obligatoris o no per exemple en els materials o solucions constructives entre d'altres matisos, per tant, es decideix que tots els camps siguin no obligatoris en la base de dades, així estalviant-nos incoherències entre versions i deixant tota la feina de validació i correctesa de les dades del fitxer d'entrada epJSON al schema de validació per tal de que posteriorment l'aplicació reculli els camps desitjats del fitxer de forma correcte perquè s'adeqüi a la base de dades.

4.3.5.4. Incorporació i donada de baixa de versions

En cas de que es vulgui donar suport a una versió determinada en un format de fitxer en concret i aquesta ja tingui una tupla assignada a la taula IDFVersions, s'haurà de fer el Update de la tupla d'aquella versió actualitzant el valor del camp SupportIDF o SupportEpJSON segons correspongui.

En cas de que la versió no tingui encara una tupla assignada a la taula IDFVersions de la base de dades es procedirà a fer un Insert d'una nova tupla indicant la nova versió i els formats els quals soporta.

En ambdós casos cal afegir aquests Updates i Inserts en una Migració per tal de que quan fem l'actualització de la base de dades els tingui en compte i aquesta quedi actualitzada correctament.

En el cas de voler de deixar de donar suport a una versió en concret d'IDF o epJSON cal afegir afegir Update i Delete en una Migration per tal de que quan fem l'actualització de la base de dades els tingui en compte i aquesta quedi actualitzada correctament de la mateixa forma que s'ha procedit en la incorporació de noves versions.

En el cas de voler incorporar una nova versió per epJSON caldrà afegir l'schema amb un nom identificatiu a la carpeta "Utils\EPJSON\epJSON_Versions\" de l'aplicació. Aquest nom identificatiu que se li posa a l'schema dins l'aplicació servirà per escollir més fàcilment dins el codi quin esquema utilitzar segons la versió del fitxer epJSON d'entrada.

4.3.5.5. Explicació del codi

En aquest apartat es centra l'atenció en el tractament del fitxer epJSON tot tenint en compte l'adaptació feta en el model de dades per tal de poder acceptar tant el format epJSON com l'IDF original. En la Figura 95 es mostra a mode de resum un diagrama de seqüència de l'activitat de l'aplicació de consola desenvolupada centrat en el tractament del fitxer epJSON d'entrada.

Paràmetres d'entrada. Tal i com ja s'ha mencionat aquesta aplicació de consola permet dos tipus de format de fitxer d'entrada, un cop l'usuari ha escollit el format del fitxer d'entrada, tant si s'escull el format IDF com el epJSON se li demana al usuari que proporcioni a la aplicació dos paràmetres adicionals. El primer dels dos es el nom del fitxer que conté el model en IDF o epJSON del qual volem fer la importació. El segon paràmetre es el path complet cap a aquest fitxer per tal de que l'aplicació sigui capaç d'ubicar-lo hi extreure el seu contingut quan sigui necessari. En cas de que l'usuari no triï un dels formats disponibles o no introdueixi aquests dos paràmetres correctament el programa retorna una excepció informant a l'usuari de la seva incorrectesa.

Connexió base de dades. Un cop recollits els valors dels dos paràmetres d'entrada mitjançant using (62) inicialitzem una instància de DatabaseContext, classe que hereda de DbContext (63), DbContext s'utilitza amb un tipus derivat que conté propietats DbSet (64) per a les entitats arrel del model. Aquests conjunts s'inicialitzen automàticament quan es crea la instància de la classe derivada. Utilitzant un using i la nova instancia de DatabaseContext iniciem una transacció amb la base de dades tot utilitzant BeginTransaction (65), aquesta ens retorna un objecte del tipus SqlTransaction Class (66) que representa una transacció Transact-SQL que es farà en la nostra base de dades SQL Server. Sobre aquest objecte SqlTransaction hi farem Commit o Rollback segons vulguem finalment guardar els diferents materials i solucions constructives a la nostra base de dades o no.

La base de dades a la qual ens volem connectar i fer servir s'indica en el fitxer de configuració de l'aplicació de consola App.config on en camp "connectionstring" hi indiquem el servidor al qual ens volem connectar i el nom de la instancia de base de dades que volem fer servir d'aquest.

Parsejat i Validació.

Per tal de contextualitzar el parsejat i validació d'un fitxer JSON cal presentar el namespace Newtonsoft.Json.Linq que proporciona classes que s'utilitzen per implementar LINQ a JSON.

LINQ (Language Integrated Query) és una sintaxi de consulta uniforme en C # y VB.NET utilitzat per guardar i recuperar dades de diferents fonts, en el nostre cas un fitxer epJSON (JSON).

Un cop parsejat el fitxer d'entrada mitjançant Newtonsoft.Json.Linq podrem gestionar-lo i fer-hi consultes utilitzant LINQ sobre algunes de les diferents JSON Classes disponibles: **JObject** (representa un objecte JSON i ajuda a analitzar les dades JSON i aplicar-hi consultes (LINQ) per filtrar les dades necessàries), **JArray** (representa un JSON Array), **JToken** (representa un JSON Token, és una classe de la qual deriven altres com JObject, JArray, JProperty, JValue, etc.) i **JSchema** (representació d'un Schema JSON utilitzat per ajudar a definir i validar estructura de JSON).

- Per tal de poder validar el fitxer d'entrada epJSON amb l'schema d'Energy Plus primer cal parsejar el epJSON que indiquem amb el path com a segon paràmetre de l'aplicació de consola ja que JSON és un format que codifica objectes en un string. La serialització significa convertir un objecte en aquest string i la deserialització és la seva operació inversa (convertir string -> objecte). Per fer-ho utilitzem el mètode JObject.Parse (67) de Newtonsoft.Json.Linq (68). Aquest mètode rep un string (string complet del contingut del fitxer epJSON) i ens retorna el seu JObject equivalent sobre el qual ja podrem operar i gestionar més fàcilment (és a dir, deserialitzem per recuperar l'estructura del BIM com a objecte JSON). L'string es obtingut a partir del mètode LoadJson (69) al qual li passem el path complet del fitxer d'entrada que conté la definició del model en epJSON i ens retorna el seu contingut complet en format d'string. Un cop tenim el contingut del fitxer en un JObject ja podem començar a fer el tractament per a fer la seva validació amb l'schema per a epJSON obtingut d'Energy Plus.

  Un esquema permet definir l'estructura d'un document al qual cal complir; en aquest cas, la descripció de l'edifici, les propietats dels seus materials, construccions i els horaris dels d'equips d'aquest que formen part del fitxer d'entrada EnergyPlus i per tant de NECADA han de seguir unes certes regles. La utilització d'un esquema també permet eines automatitzades per validar un fitxer específic i proporcionar errors descriptius si el format, els intervals de dades o altres propietats del fitxer d'entrada violen l'esquema contra el qual s'està validant.

  Primerament però llegim en quina versió està el fitxer epJSON que passem com a input per tal de fer la validació amb un schema o un altre (Figura 86), o en el cas de que no es tingui adaptat el sistema per a una determinada versió (l'aplicació no te inclòs l'schema per a aquella versió) llavors s'informa a l'usuari per consola que la versió del fitxer d'input no està suportada pel sistema (Figura 87). Abans de passar a fer la validació

necessitem parsejar l'schema epJSON com a JSchema amb el mètode JSchema.Parse (70) per tal de que poguem fer-lo servir com a schema de validació.

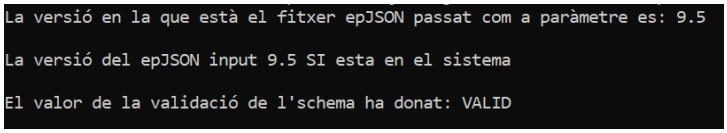


FIGURA 86. MISSATGE PER CONSOLA INFORMANT A L'USUARI DE LA VERSIÓ EN LA QUE ESTÀ EL FITXER D'ENTRADA EPJSON, SI AQUESTA ESTA SUPORTADA PEL SISTEMA I SI LA SEVA VALIDACIÓ TÉ ÈXIT

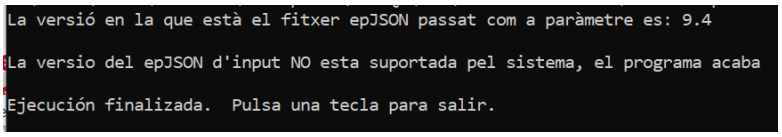


FIGURA 87. MISSATGE INFORMANT A L'USUARI DE LA VERSIÓ EN LA QUE ESTÀ EL FITXER D'ENTRADA ON AQUESTA NO ES TROBA EN EL SISTEMA I PER TANT EL PROGRAMA ACABA

Aquesta consulta de disponibilitat es fa directament a la base de dades on cada cop que integrem un nou schema per a suportar una nova versió haurem d'afegir una nova tupla a la taula IDFVersion de la base de dades per tal d'indicar-ho, indicant si dona suport a epJSON i/o a IDF amb els atributs "SupportEpJson" i "SupportIDF". A la Figura 88 es mostra un exemple de fragment de fitxer que conté la definició d'un model en format epJSON on es mostra la versió d'aquest (versió 9.5):

```
"Version": {
    "Version 1": {
        "version_identifier": "9.5"
    }
},
```

FIGURA 88. ESTRUCTURACIÓ DE LA INFORMACIÓ VINCULADA AMB LA VERSIÓ D'UN FITXER EN FORMAT EPJSON

- La validació es realitza mitjançant el mètode IsValid (71) que determina si un JToken es vàlid o no segons un JSchema passat com a paràmetre.
- Si el model ha estat correctament validat amb l'schema el programa continuarà. En cas contrari acabarà i informarà a l'usuari que el fitxer que conté la definició del model en format epJSON no s'ha pogut validar correctament i l'error que ha succeït. (Figura 86 i Figura 89).
- Un cop validat el fitxer epJSON es crea una instància de la classe epJsonImporter la qual gestionarà tota la lectura importació dels diferents materials i solucions constructives que contingui el model.

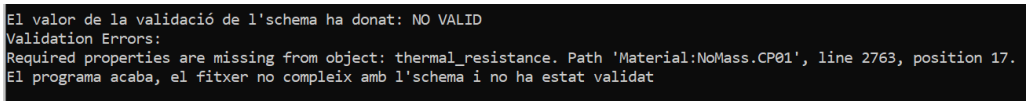


FIGURA 89. MISSATGE MOSTRAT A L'USUARI INDICANT QUE EL FITXER D'INPUT NO ES VÀLID I L'ERROR DE VALIDACIÓ SUCCEÏT

Tal i com s'ha mostrat el parsejat i la validació mitjançant l'schema epJSON es realitza fàcil i ràpidament, aspecte totalment contrari en el IDF on el procés es molt més complex.

EpJson Importer.

La classe EpJsonImporter gestiona la importació dels materials i solucions constructives del fitxer ubicat en el path complet passat com a paràmetre d'execució a l'aplicació.

La classe epJsonImporter conté els diferents atributs que contindràn la informació necessaria per a importar el contingut que ens interessi del fitxer d'input. Els atributs amb els que es crearà una nova instància d'aquesta classe son bàsicament un EpJSONModel (la ja mencionada subclasse de Model), el DatabaseContext, el userName, un ClassificationConstructingSolution, un ClassificationMaterial, el el JObject del propi fitxer epJSON ja validat i òbviament parsejat, un IDFVersion amb un string representant la versió, dos llistes, una que contindrà el conjunt de Materials (List) i l'altre les Solucions Constructives (List) i finalment totes les llistes necessaries que contindràn cada una d'elles un subtipus de material. Quan creem un epJsonImporter li donem valor a tots els atributs que li passem com a paràmetre i es creen noves llistes buides que s'aniran omplint segons si els elements llegits es tracten de Materials o Solucions Constructives i subtipus de Materials.

Primer es farà el tractament del Materials que contingui el fitxer epJSON i després es realitza el tractament de les solucions constructives que contingui aquest mateix fitxer. Es procedeix d'aquesta forma ja que les solucions constructives estan formades per materials que es guarden a la base de dades com a MaterialSolutions, és a dir, materials que son utilitzats per alguna solució constructiva llegida, aquestes MaterialSolutions han de fer referencia com a *foreign key* als materials de la taula Materials llegits prèviament (Figura 90).

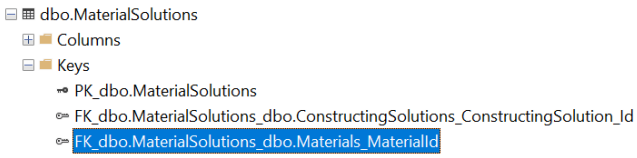


FIGURA 90. TAULA MATERIALSOLUTIONS AL SSMS ON ES MOSTREN LES FK CAP A LA CONSTRUCTINGSOLUTION I MATERIAL AL QUE FAN REFERÈNCIA

Tractament dels materials.

El tractament del materials es fa mitjançant una funció que agafa el epJSON complet representant la definició del model i buscarà cada tipus de material per key que representa el seu nom. Un cop els trobem aquests tipus de material com a JObject, aquests contindran totes les instàncies d'aquell tipus de material que recorrerem per obtenir els seus atributs.

Tant els Materials com les Solucions Constructives estan estructurades de la mateixa forma seguint el format epJSON. A continuació es mostra un exemple (Figura 91) d'un tipus de Material i les diferents instàncies d'aquest (en concret un tipus de material anomenat "Material NoMass") :

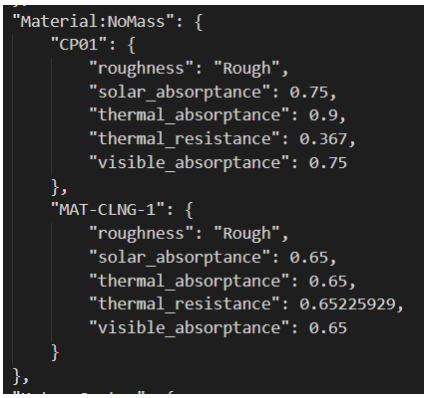


FIGURA 91. EXEMPLE DEL TIPUS DE MATERIAL "MATERIAL:NOMASS" I LES SEVES INSTÀNCIES EN UN FITXER EN FORMAT EPJSON

Veiem que el tipus de Material es un Object en JSON (delimitat per "{}") el qual a la seva vegada conté en aquest el cas d'exemple dos Objects més, aquest cop aquests dos representen ja dues instàncies del tipus de Material que s'especifica en el nom del Object ("Material:NoMass"). Els noms dels dos Materials que formen part d'aquest subtipus "Material:NoMass" son "CP01" i "MAT-CLNG-1". Ara ja cada Object que representa una instància d'un tipus de Material ja conté un seguit de parelles clau-valor indicant el nom i el valor de cada camp representat les característiques de cada material. En el schema per epJSON que ens proporciona EnergyPlus podem identificar quins camps son obligatoris i quins no segons el tipus de Material o Solució Constructiva, per tant, no sempre hi haurà el mateix nombre de camps en un determinat tipus de material o solució constructiva a no ser que tots els possibles camps que pot tenir aquell Material siguin obligatoris.

Com a exemple, a la següent Figura 92 podem veure un fragment de l'schema epJSON referent a la versió 9.5 que informa dels camps i les característiques que poden tenir aquests en el tipus de material "Material:NoMass" (exemple mostrat en la Figura 92) així com la seva obligatorietat (tal i com es veu al camp "required" inferior del fragment de codi on només el camps "roughness" i "thermal_resistance" son obligatoris per cada instància d'aquest tipus de material).

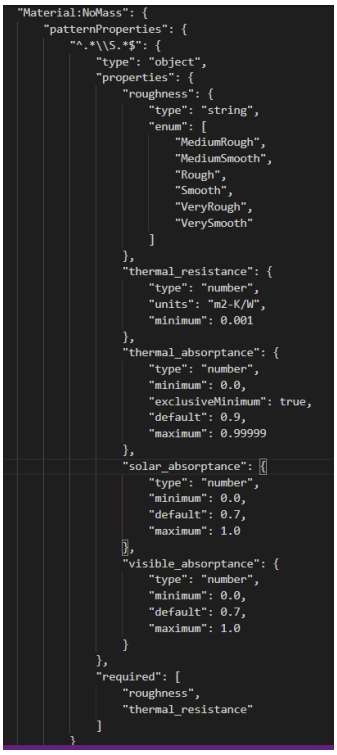


FIGURA 92. FRAGMENT REFERENT AL TIPUS DE MATERIAL "MATERIAL:NOMASS" DE L'SCHEMA EPJSON

Per tant, a l'hora d'obtenir la informació del fitxer epJSON primer cal obtenir l'Object que representa el tipus de Material que ens interessa i posteriorment recorre'l per poder iterar sobre cada una de les instàncies d'aquest i recollir els seus camps. Per cada tipus de Material procedim de la mateixa manera:

- Primerament obtenim l'Objecte JSON que representa el tipus de Material mitjançant la cerca pel seu nom.
- Per a cada instància d'aquest tipus de material segons el tipus de material del que es tracti (es crearà una nova instància d'aquest material i es cridarà a la funció que inicialitzarà tant la superclasse Material com la subclasse representant el tipus de Material amb els atributs que es recullin de la instància del material passada com a paràmetre.
- Cada funció d'inicialització d'una instància de tipus de material es diferencia en tres parts:
    1) Inicialització de la superclasse Material amb el nom passat en el paràmetre. Aquesta funció no varia. S'inicialitzen els atributs propis de la superclasse Material i per tant comuns en tots els tipus de Material.
    2) La segona part es la vinculada amb la inicialització dels valors dels atributs de la subclasse de Material que estiguem analitzant. Cada subtipus de material s'inicialitza amb una funció diferent així diferenciant clarament quins camps s'han de llegir en un tipus de material i en un altre. Es procedeix recorrent l'Object de la instancia del tipus de material mitjançant el tipus JProperty (72). Cada JProperty representa una parella clau-valor de l'Object instancia tipus material. Tal com s'ha mencionat anteriorment hi ha camps que son obligatoris i camps que no ho són, però en aquest moment no hem de fer cap tractament diferent, ja que en el cas de que hi hagués un camp obligatori no present en aquest Object el fitxer epJSON no hauria estat validat i per tant no hauríem arribat al seu tractament. Per tant, es comproven tots els possibles noms dels camps que pot arribar a tenir un tipus de material en concret i en cas de que es

trobi se li assigna el seu valor al atribut corresponent a la instància representant el tipus de material en concret.

3) Finalment, un cop inicialitzat tan la superclasse Material com la subclasse tipus de Material el qual s'estigui tractant s'afegeix la instancia Material a la llista que conté tots els Materials que s'han anat llegint i s'afegeix la instancia del tipus de submaterial a la llista que conté tots els tipus de submaterials d'aquell tipus.

Tractament de les solucions constructives.

El tractament de les solucions constructives es fa mitjançant una funció que cerca i inicialitza les diferents solucions constructives que es troben en el fitxer epJSON d'entrada.

Es procedeix de la mateixa forma que amb Materials però en aquest cas només cerquem per la key "Construction" que representarà el JObject que contindrà totes les instàncies de ConstructingSolution que ens interessin.

A continuació (Figura 93) es mostra un exemple de solucions constructives i les seves diferents instàncies.

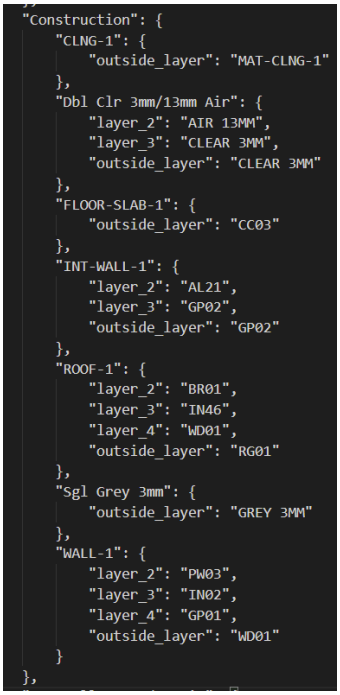


FIGURA 93. EXEMPLE DE SOLUCIONS CONSTRUCTIVES "CONSTRUCTION" I LES SEVES DIFERENTS INSTÀNCIES EN UN FITXER EN FORMAT EPJSON

Els noms d'aquestes instàncies de solucions constructives son els noms de materials ja llegits anteriorment per l'aplicació i l'únic camp obligatori mencionat per l'schema en la versió 9.5 es el camp "outside_layer" (Figura 94).

FIGURA 94. FRAGMENT REFERENT A LES SOLUCIONS CONSTRUCTIVES ("COSNTRUCTION") DE L'SCHEMA EPJSON

Per cada instància de solució constructiva es cridarà a una funció que primerament donarà valor als atributs de la nova instància de solució constructiva per posteriorment crear una nova instància de MaterialSolution representant el material utilitzat per un dels camps de la solució constructiva que s'estigui analitzant. Cada parella clau-valor d'una solució constructiva representa un material utilitzat (tal i com es veu en l'exemple), per tant, aquests s'aniran afegint a la llista de "MaterialSolutions" de la solució constructiva al qual pertanyin. Finalment un cop recorregut totes les parelles clau-valor d'una solució constructiva determinada, aquesta s'afegirà a la llista de ConstructingSolutions perquè posteriorment sigui guardada a la base de dades.

Un cop vist el tractament dels materials i de les solucions constructives amb el format epJSON podem veure que ens ofereix un gran avantatge en cost de cerca d'aquests, ja que si procedíssim de la mateixa forma que amb l'IDF faríem una cerca a través de tots els Objects de primer nivell d'aniuament del fitxer a través d'una iteració buscant un *match* amb algun dels noms que ens interessen, en canvi ara amb l'estructura JSON ja podem fer una cerca i accés per clau.

La diferencia de costos es que la primera opció té un cost de O(n) on "n" es el nombre de Objects de primer nivell del fitxer epJSON parsejat d'entrada sobre el que iteraríem (el pitjor dels casos es que hi hagués un Object que ens interesses i que aquest estigués situat l'últim i per tant hauriem de recòrrer tot el fitxer per fer el *match*) i en el segon cas on cerquem i accedim per clau senzillament tenim un cost O(1). Aquesta avantatge la aconseguim gràcies a la pròpia estructura de JSON on un tipus d'Objecte conté aniuades totes les instàncies d'aquest, en canvi l'IDF pot contenir les instancies disperses per tot el fitxer.

En el tractament del fitxer en format IDF quan estem cercant i tractant els materials recorrent tot el fitxer IDF també cerquem a la vegada les solucions constructives per evitar haver de recórrer tot el fitxer posteriorment per trobar-les. En canvi amb un format JSON es possible fer una cerca per clau i per tant no es necessari recórrer el fitxer i per tant tampoc buscar solucions per disminuir el cost com la que es fa de buscar les solucions constructives mentre es cerquen els materials en el cas de l'IDF.

Cal tenir en compte però, que aquesta disminució de cost comentada a l'hora d'importar la informació que interessa a NECADA a la seva base de dades es una disminució en el cost del ja mencionat procés de preparació de l'escenari de simulació. El cost vinculat amb la preparació de l'escenari de simulació es totalment marginal en el conjunt del sistema NECADA ja que el gran cost ve determinat per l'execució de les simulacions (l'ús del format epJSON també ajudaria a obtenir millor eficiència en el simulador i per tant a NECADA tal i com s'ha comentat al llarg de l'apartat 4.3.5. Desenvolupament de l'aplicació de consola). Tot i això, no acaba de ser menyspreable, ja que la preparació de l'escenari es quan realment l'usuari està en contacte amb el sistema restant a l'espera per poder llençar les simulacions desitjades i en cas de entrar per exemple un BIM d'una àrea urbana complexa amb molta informació doncs acabaria tenint certa rellevància.

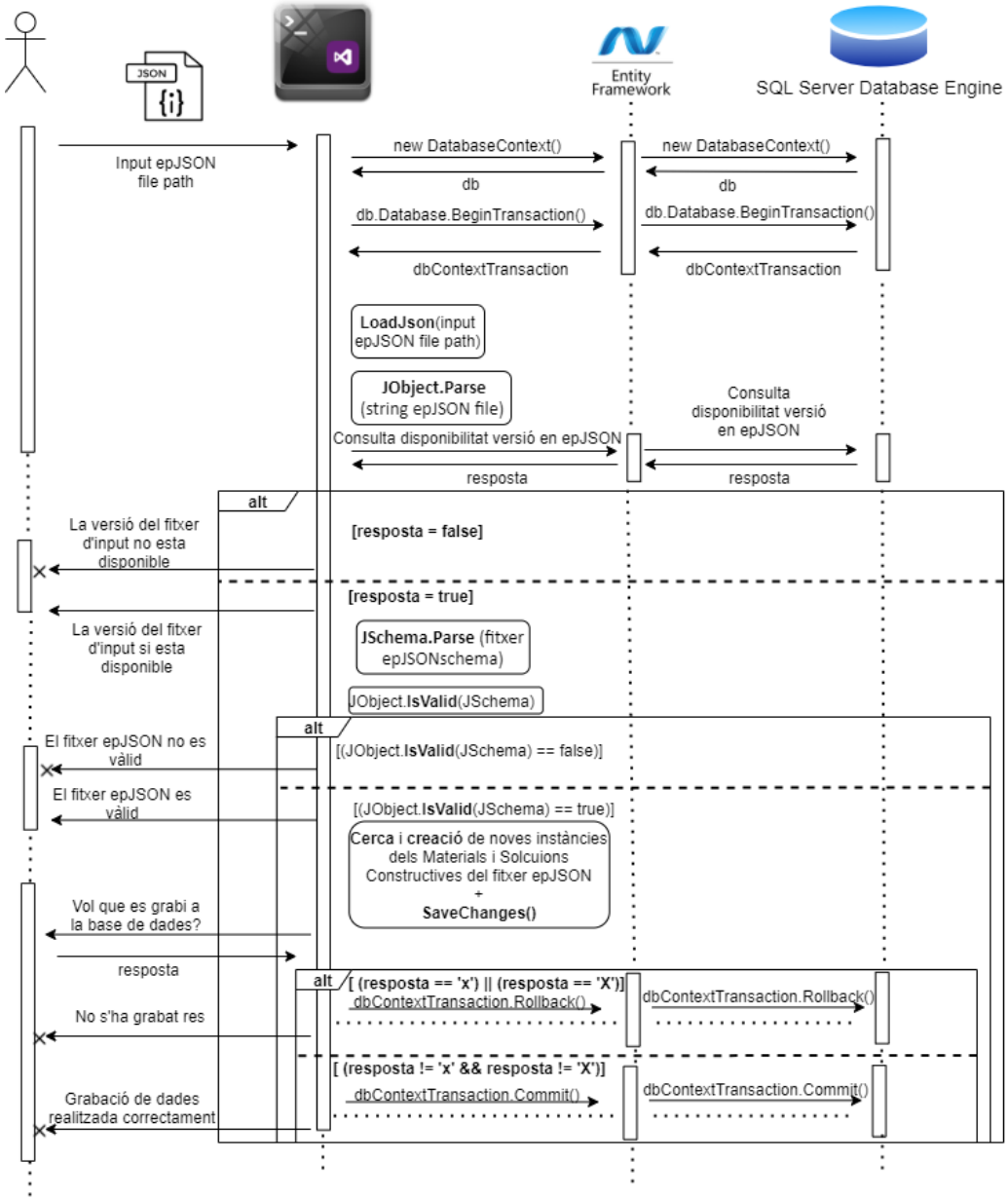


FIGURA 95. DIAGRAMA DE SEQÜÈNCIA DE L'ACTIVITAT DE L'APLICACIÓ DE CONSOLA DESENVOLUPADA CENTRAT EN EL TRACTAMENT DEL FITXER EPJSON D'ENTRADA

EpJsonImporter. Opció d'importació mitjançant la deserialització del fitxer d'entrada utilitzant les classes c# existents de materials i solucions constructives.

Una altre opció que també s'ha emprat per a dur a terme la importació dels materials i solucions constructives del fitxer epJSON d'entrada a la base de dades es la d'utilitzar una classe c# per tal de que serveixi per a deserialitzar la informació que ens interessa del fitxer d'entrada (materials i solucions constructives) així obtenint els .NET objectes que ens permetran gestionar-los des de la aplicació (Figura 96), d'aquesta forma ja no es necessari primer parsejar tot el document com a JObject per després cercar per key els elements que ens interessa importar a la base de dades com s'ha explicat anteriorment.

No deserialitzem tot el fitxer ja que no te sentit per a la importació, únicament necessitem gestionar dins l'aplicació el que interessa a Optisim. El que fem es crear una classe c# nova que representarà el fitxer epJSON deserialitzat i que contindrà tots els materials i solucions constructives que aquest contingui i que per tant s'hauran d'importar a la base de dades.
Com a exemple es mostra el següent fragment de codi de la nova classe creada que conté un Dictionary el qual contindrà, en aquest cas tots els materials de tipus "MassMaterial" (identificats en el fitxer epJSON com a Material):

```
public class epJSONFile
{
        public Dictionary<string, MassMaterial> Material { get; set; }
        (… tots els altres Dictionary vinculats amb els tipus de materials restants i
        les solucions constructives)
}
```

Cada parella clau-valor del dictionary Material simbolitza una instància del tipus de material "MassMaterial" trobat, deserialitzat del fitxer epJSON d'entrada, on la key es un string fent referència al nom de la instancia de "MassMaterial" en concret i MassMaterial el valor del "MassMaterial" amb aquest nom (per tant, també fem servir la pròpia classe c# del tipus de material i solució constructiva per a fer la deserialització).

El namespace System.Text.Json proporciona funcionalitats per serialitzar i deserialitzar des de la notació JSON.

Per tant, el que fem es llegir el contingut del fitxer epJSON com a un string i deserialitzar-lo mitjançant el mètode *Deserialize* de *System.Text.Json.JsonSerializer* utilitzant la nova classe *epJSONFile* creada per tal d'obtenir una instància d'aquesta classe i ja poder recollir els materials i solucions constructives que desitgem d'aquest fitxer d'entrada epJSON.

Per cada tipus de material i solució constructiva procedim de la mateixa manera que ho hem fet anteriorment, la diferencia es que ara ja els tenim deserialitzats, és a dir, com a instàncies de les classes que ens interessa gestionar en la aplicació. Per tant, ara no és necessari cercar per key. Senzillament per exemple, afagem el Dictionary ja deserialitzat que conté totes les instàncies del tipus de material MassMaterial, recorrerem totes les instancies de MassMaterial del fitxer epJSON tot creant i inicialitzant les noves instàncies d'aquest tipus de material i la seva superclasse Material amb els valors deserialitzats del fitxer epJSON d'entrada. On de cada parella clau-valor del Dictionary, entry.Key serà el nom de la instància del tipus de material i el entry.Value la instància del tipus de material sencera:

```
Dictionary<string, MassMaterial> nomassmaterials = epjsonfile.Material;
foreach (KeyValuePair<string, MassMaterial> entry in nomassmaterials)
```

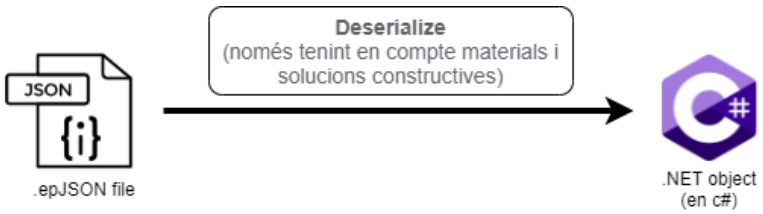


FIGURA 96. DIAGRAMA D'IMPORTACIÓ MITJANÇANT LA DESERIALITZACIÓ DEL FITXER D'ENTRADA UTILITZANT LES CLASSES C# EXISTENTS DE MATERIALS I SOLUCIONS CONSTRUCTIVES

Procés d'exportació de Materials i Solucions Constructives a un fitxer epJSON.

Tot i que es més elegant i lògic fer servir els avantatges de JSON per la seva serialització i deserialització utilitzant directament les classes que tenim dins el model de dades, en la importació no es estrictament necessari utilitzar-les (en la primera importació descrita inicialment es fa però no directament en les classes que tenim dins el model de dades, sinó com a JObjects). En el cas de l'exportació però, es del tot necessari perquè no esdevingui un procés innecessàriament complex. Podem veure per tant que gràcies al fàcil parsejat que ofereix JSON l'exportació també esdevé una millora respecte els fitxers IDF.

Quan es du a terme la importació de materials i solucions constructives des de un fitxer epJSON cap a NECADA no es necessari tenir en compte el conjunt del document restant per deserialitzar-lo. En canvi, en el cas de l'exportació, en la deserialització, si que hem de tenir en compte tot el document epJSON representant el BIM de l'edifici per tal de que a l'hora de tornar a serialitzar el contingut d'aquest document (amb les modificacions fetes) cap al fitxer epJSON d'origen, el document quedi intacte menys òbviament la part afegida o modificada segons els materials i solucions constructives que haguem decidit exportar.
De no fer-ho així senzillament sobreescriuríem el fitxer epJSON original al qual hi volíem exportar materials o solucions constructives amb únicament la part deserialitzada que ens interessava en la importació (materials i solucions constructives), eliminant la resta del contingut del fitxer epJSON representant el BIM de l'edifici.

Per tant, tal i com hem fet amb la importació creem una classe, però en aquest cas, a més de tenir els *Dictionary's* que contindran cadascun dels tipus de materials i solucions constructives (les seves classes c# per a deserialitzar, i en aquest cas de la exportació també serialitzar) que pertoqui cal afegir en aquesta deserialització la resta del document. Com que la resta del document segueix l'estructura del epJSON doncs tenim que podem deserialitzar com un únic *Dictionary<string, object>* tota la resta del document. Per informar la resta del document que no s'ha mencionat anteriorment en la classe que fem servir per a fer la deserialització utilitzem [JsonExtensionData] on es deserialitza tota la informació en el nou *Dictionary* "RestaDelDocument" (ja que senzillament no tenim classes en l'aplicació per a que aquesta informació sigui deserialitzada (no interessa a Optism)):

```
public class EpJSONfile
{

   public Dictionary<string, MassMaterial> Material { get; set; }

      (… tots els altres Dictionary vinculats amb els tipus de materials restants i
      les solucions constructives)

   [JsonExtensionData]
   private Dictionary<string, object> restaDelDocument { get; set; }
}
```

Per tant, de la mateixa forma que hem deserialitzat el fitxer epJSON a importar també necessitem deserialitzar el fitxer epJSON al qual hi volem exportar informació. Procedim de la mateixa forma, però en aquest cas tenint en compte ja tot el fitxer.
Per altre banda, per dur a terme l'exportació de materials i/o solucions constructives cap a un determinat fitxer epJSON cal obtenir aquests materials i solucions constructives de la base de dades de NECADA tal i com es fa en el cas de l'exportació cap a fitxers IDF. Senzillament igual que hem fet amb la versió (per dur a terme si està suportada pel sistema o no) declarem una instancia del tipus de material i/o solució constructiva que volem exportar i li donem valor amb el valor que tingui a la base de dades tot utilitzant l'objecte Database ja presentat i els DbSet<TipusDeMaterial> que ens permeten consultar i obtenir objectes d'una determinada classe amb facilitat.
Tant el fitxer epJSON que conté la definició completa del BIM com el material i/o solució constructiva que recollim de la base de dades han d'estar en la mateixa versió per tal de que no hi hagi incoherències en el fitxer resultant de l'exportació.

A continuació es mostra un exemple on *epjsonfile* es el fitxer epJSON ja deserialitzat al qual es vol exportar informació i en aquest cas que es mostra s'afegeix un tipus de material en concret recollit de la base de dades al *Dictionary* que li correspon segons el tipus de material del fitxer epJSON deserialitzat d'entrada. Finalment es serialitza tot el fitxer epJSON d'entrada que ha estat ja modificat i el sobreescrivim de tal forma que quedi actualitzat:

```
epjsonfile.Material.Add(massMaterialaExportar.Name, massMaterialaExportar);
string jsonstring = JsonConvert.SerializeObject(epjsonfile, Formatting.Indented);
File.WriteAllText(pathCompletepJSONCapAlQueExportar, jsonstring);
```

La deserialització i serialització es fa d'acord amb el nom del atributs de les classes utilitzades per a la deserialització/serialització i el nom dels camps del JSON, per tant, necessiten tenir el mateix nom. En cas de no tenir el mateix nom i no voler canviar-lo en les classes (els atributs de C# s'han de nomenar mitjançant PascalCase (73) i en el fitxer epJSON doncs no es compleix la mateixa nomenclatura), podem utilitzar l'atribut [JsonPropertyName] indicant el nom en que l'atribut s'ha de deserialitzar i serialitzar cap al fitxer epJSON. Així podrem seguir utilitzant els noms originals de les classes ja existents i no haver d'actualitzar la base de dades innecessariament. Exemple on tenim que en el fitxer epJSON el camp "conductivity" comença en minúscula però l'atribut de la classe c# a l'aplicació i a Optisim comença en majúscula):

```
[JsonPropertyName("conductivity")]
public double Conductivity { get; set; }
```

Un altre aspecte a tenir en compte en la serialització es que per defecte es serialitzen tots els atributs de la classe feta servir per a la deserialització i que per tant també es fa servir per a la seva serialització. Com que tenim classes que tenen atributs els quals no volem que es serialitzin, s'exportin cap al fitxer epJSON, doncs utilitzem l'atribut [JsonIgnore] abans de la declaració d'aquest dins la classe c#. (Per la deserialització aquest aspecte no te rellevància ja que només es deserialitzarien els camps que tingués el epJSON, per tant, no ens hem de preocupar per els atributs que només hi son en les classes c# de l'aplicació i Optisim, però com també s'ha de dur a terme l'exportació i per tant serialització doncs cal fer-ne ús afegint aquest atribut a tots els atributs de les classes d'Optisim que no s'hagin d'exportar cap a un fitxer epJSON).

Exemple on en la classe MassMaterial hi tenim atributs de comentaris, aquest no volem que s'exportin al fitxer epJSON ja que aquest no els contempla en el seu esquema:

```
[JsonIgnore]
public string SpecificHeat_comment { get; set; }
```

El procés d'exportació de materials i/o solucions constructives cap a un fitxer epJSON queda resumit en la següent Figura 97:

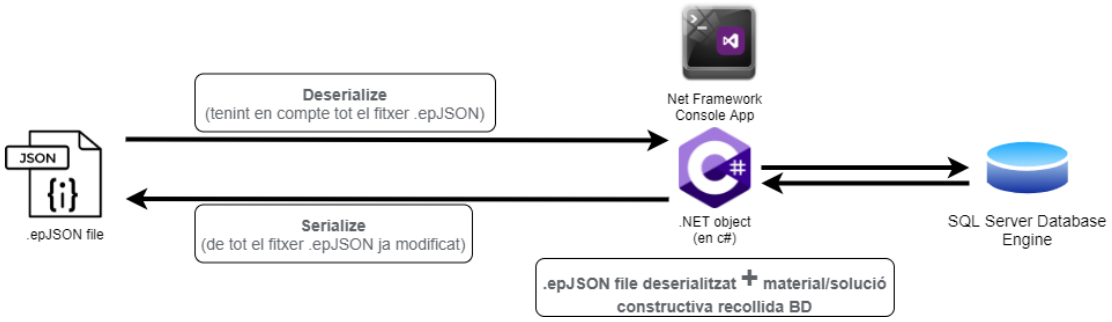


FIGURA 97. DIAGRAMA DEL PROCÉS D'EXPORTACIÓ DE MATERIALS I SOLUCIONS CONSTRUCTIVES A UN FITXER EPJSON

4.3.6. Guia d'execució de l'aplicació de consola

4.3.6.1. Prerequisits

L'aplicació de consola no la podem executar directament ja que utilitza una base de dades local, primer necessitem definir una database i indicar al fitxer App.Config quina es la base de dades que volem fer servir, per tant haurem d'acabar compilant per tal de que l'executable tingui en compte aquest canvi. A continuació es detallen els requisits per tal de poder compilar l'aplicació i poder-la executar tot seguint el manual d'execució proporcionat en l'apartat 4.3.6.3. Manual d'execució

1) **VisualStudio2019** (VS2019): Amb VS2019 obrim la solució del projecte de l'aplicació de consola (fitxer amb extensió .sln).

2) Un cop obert per tal de que puguem compilar l'aplicació de consola i així poder generar l'executable necessitem abans tenir instal·lats els següents paquets, que en cas de no tenir-los els instal·lem mitjançant el **NuGet package manager** (utilitzant o bé "Package Manager Console" o bé "Manage NuGet Packages for Solution" (Figura 98)):

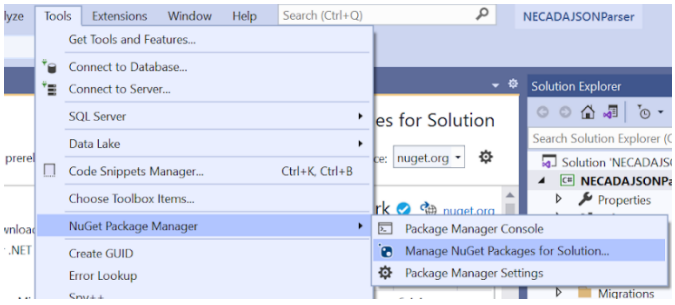


FIGURA 98. EINES "PACKAGE MANAGER CONSOLE" I "MANAGE NUGET PACKAGES FOR SOLUTION" A VS2019

- Name: **EntityFramework**. Version: **6.4.4.**
  Description: Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.

- Name: **Microsoft.AspNet.Identity.Core** Version: **2.2.3**
  Description: Core interfaces for ASP.NET Identity.

- Name: **Microsoft.AspNet.Identity.EntityFramework** Version: **2.2.3**
  Description: ASP.NET Identity providers that use Entity Framework.

- Name: **Microsoft.AspNet.Mvc** Version: **5.2.7**

Description: This package contains the runtime assemblies for ASP.NET MVC. ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup.

- Name: **Microsoft.AspNet.Razor** Version: **3.2.7**
  Description: This package contains the runtime assemblies for ASP.NET Web Pages. ASP.NET Web Pages and the new Razor syntax provide a fast, terse, clean and lightweight way to combine server code with HTML to create dynamic web content.

- Name: **Microsoft.AspNet.WebPages** Version: **3.2.7**
  Description: This package contains core runtime assemblies shared between ASP.NET MVC and ASP.NET Web Pages.

- Name: **Microsoft.Web.Infrastructure** Version: **1.0.0**
  Description: This package contains the Microsoft.Web.Infrastructure assembly that lets you dynamically register HTTP modules at run time.

- Name: **Newtonsoft.Json** Version: **12.0.3**
  Description: son.NET is a popular high-performance JSON framework for .NET.

- Name: **Newtonsoft.Json.Schema** Version: **3.0.14**
  Description: Json.NET Schema is a complete and easy-to-use JSON Schema framework for .NET.

- Name: **System.Configuration.ConfigurationManager** Version: **5.0.0**
  Description: Provides types that support using configuration files.
  Commonly Used Types:
  System.Configuration.Configuration
  System.Configuration.ConfigurationManager
  When using NuGet 3.x this package requires at least version 3.4.

- Name: **System.Runtime.Serialization.Json** Version: **4.3.0**
  Description: Provides classes for serializing objects to the JavaScript Object Notation (JSON) and deserializing JSON data to objects.
  Commonly Used Types:
  System.Runtime.Serialization.Json.DataContractJsonSerializer
  When using NuGet 3.x this package requires at least version 3.4.

- Name: **System.Security.AccessControl** Version: **5.0.0**
  Description: Provides base classes that enable managing access and audit control lists on securable objects.
  Commonly Used Types:
  System.Security.AccessControl.AccessRule
  System.Security.AccessControl.AuditRule
  System.Security.AccessControl.ObjectAccessRule
  System.Security.AccessControl.ObjectAuditRule
  System.Security.AccessControl.ObjectSecurity
  When using NuGet 3.x this package requires at least version 3.4.

- Name: **System.Security.Permissions** Version: **5.0.0**
  Description: Provides types supporting Code Access Security (CAS).
  When using NuGet 3.x this package requires at least version 3.4.

- Name: **System.Security.Principal.Windows** Version: **5.0.0**

Description: Provides classes for retrieving the current Windows user and for interacting with Windows users and groups.
Commonly Used Types:
System.Security.Principal.WindowsIdentity
System.Security.Principal.SecurityIdentifier
System.Security.Principal.NTAccount
System.Security.Principal.WindowsPrincipal
System.Security.Principal.IdentityReference
System.Security.Principal.IdentityNotMappedException
System.Security.Principal.WindowsBuiltInRole
System.Security.Principal.WellKnownSidType
When using NuGet 3.x this package requires at least version 3.4.

- Name: **System.Text.Json** (I totes les seves dependencies)Version: **5.0.2**
  Description: Provides high-performance and low-allocating types that serialize objects to JavaScript Object Notation (JSON) text and deserialize JSON text to objects, with UTF-8 support built-in. Also provides types to read and write JSON text encoded as UTF-8, and to create an in-memory document object model (DOM), that is read-only, for random access of the JSON elements within a structured view of the data.

3) Un cop instal·lats els paquets necessaris ja podríem compilar l'aplicació i generar l'executable, abans però necessitem fer modificacions en el fitxer **App.config** (que equival al 'web.config' d'una aplicació web):

- Cal indicar a quin server i instància de database ens volem connectar i per tant fer servir. Exemple:

```
<connectionStrings>
  <add name="DefaultConnection"
connectionString="Server=CARLESXPS13\SQLEXPRESS;Database=ConsoleAppIDFepJSON;In
tegrated Security=true" providerName="System.Data.SqlClient" />
</connectionStrings>
```

4) En cas d'haver de crear una nova database procedim de la forma descrita en l'apartat 4.3.6.2. Creació de la Base de Dades

5) Si la aplicació compila sense errors ja podem executar les **migrations** des de la **Package Manager Console** (idèntiques a les que tenia Optisim en versió web (que ja dona suport a IDF) més les noves migrations creades per donar suport a la importació de fitxers epJSON) sobre la nova base de dades:

```
> Update-Database -ConfigurationTypeName Optisim.Migrations.Users.Configuration
> Update-Database -ConfigurationTypeName Optisim.Migrations.Configuration
```

A l'aplicació Optisim (web) hi ha dos contexts:

a) Un es el que gestiona els usuaris i els permisos que tenen aquests (la base de dades s'actualitza amb aquesta informació a partir del primer update-database mostrat). Crea les taules 'AspNetUsers', 'AspNetUserRoles', 'AspNetUserLogins', 'AspNetUserClaims' i 'AspNetRoles'.

b) El segon es el que gestiona les pròpies dades de l'aplicació, crea la resta de les taules (de materials, solucions constructives, versions suportades, etc.).També crea un trigger que quan s'inserten dades a 'AspNetUsers' fa un insert a 'NecadaUsers'.

A l'aplicació de consola necessitem els dos contextos mencionats per tal de que no falli cap migration, però només utilitzem les taules que es creen en el punt b), per tant, mai accedirem a les taules 'AspNetUsers', 'AspNetUserRoles', 'AspNetUserLogins', 'AspNetUserClaims' ni 'AspNetRoles', donat que no fem login en cap moment (Optisim web si que ho fa òbviament), ens centrem únicament en la gestió dels fitxers d'entrada (IDF i epJSON) obviant la gestió d'usuaris (el mateix passa amb alguns dels paquets instal·lats prèviament).

Ara ja tenim actualitzada la base de dades de tal forma que ja podem executar l'aplicació de consola correctament d'acord amb l'actualització de la base de dades feta a partir de l'execució de les migracions que conté l'aplicació.

4.3.6.2. Creació de la Base de Dades

- Obrim el programa Microsoft SQL Server Management Studio (SSMS) (Figura 99).

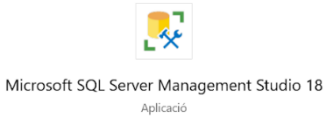


FIGURA 99. MICROSOFT SQL SERVER MANAGEMENT STUDIO (SSMS)

- Un cop entrem ens connectem al servidor que vulguem fer servir (amb l'opció d'autentificació de "Windows Authentication" se'ns omplen les credencials automàticament amb l'usuari Windows que estiguem fent servir) (Figura 100):

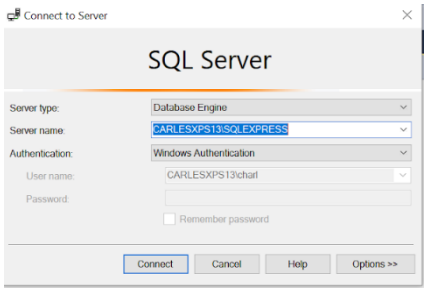


FIGURA 100. DIÀLEG DE CONNEXIÓ A SERVIDOR. SSMS

- Creem la nova data base (Figura 101):

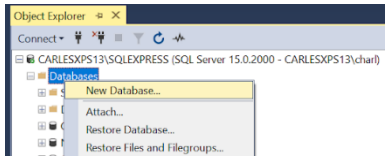


FIGURA 101. OPCIÓ NEW DATABASE A SSMS

- A la finestra que se'ns obre escrivim senzillament el nom que li volem donar a la database i que farem servir per identificar-la en el fitxer App.config de l'aplicació de consola i cliquem 'Ok' (Figura 102).

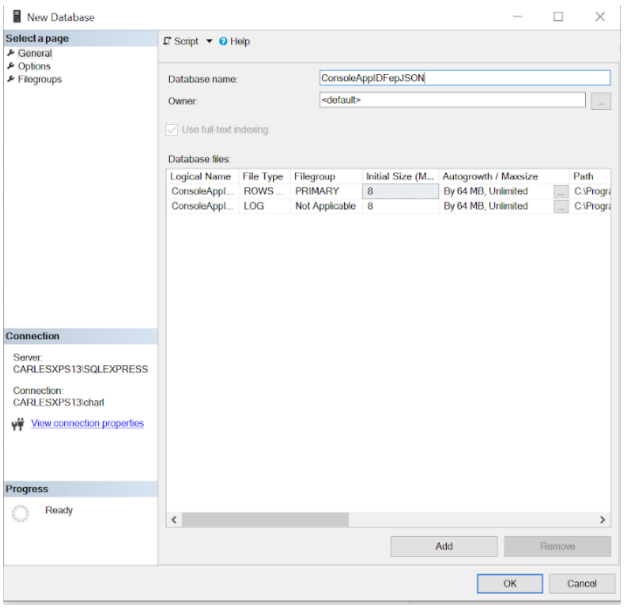


FIGURA 102. DIÀLEG DE CREACIÓ D'UNA NOVA BASE DE DADES AL SERVIDOR DESITJAT. SSMS

4.3.6.3. Manual d'execució

Es una aplicació de consola per tant, ens situem des de la consola al directori on hi trobem l'executable (\Optisim\bin\Debug). L'executable de l'aplicació té el nom "Optisim". Per tant, executem Optisim.exe (Figura 103).

```
\bin\Debug>Optisim.exe
```

FIGURA 103. EXECUCIÓ DE L'APLICACIÓ DE CONSOLA

El primer que ens pregunta es quin tipus de fitxer volem importar, en cas de voler importar un fitxer IDF introduirem el número '1' per consola, en cas de voler importar un fitxer en format epJSON introduirem el número '2' (Figura 104).

```
Escull el format del fitxer que es desitja importar:
 Premi '1' si es IDF
 Premi '2' si es epJSON
```

FIGURA 104. DIÀLEG DE DECISIÓ DE QUIN FORMAT DE FITXER ES DESITJA IMPORTAR. APLICACIÓ DE CONSOLA

Un cop escollit els tipus de fitxer a importar tant si hem escollit IDF com epJSON haurem de proporcionar primer el nom del fitxer (amb extensió inclosa: .idf o .epJSON) i posteriorment proporcionar el path complet de la carpeta on està ubicat el fitxer en qüestió:

- Cas epJSON (Figura 105).

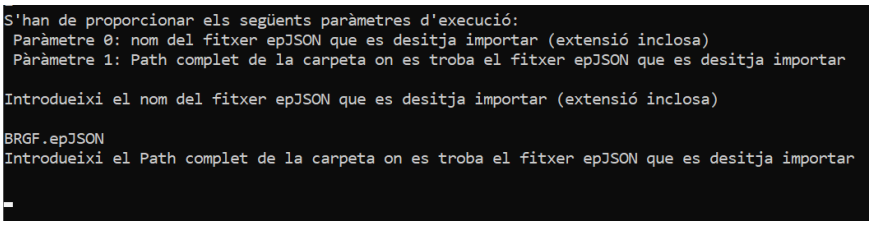


FIGURA 105. DIÀLEG ON L'USUARI PROPORCIONA EL NOM DEL FITXER EPJSON I POSTERIORMENT PROPORCIONA EL PATH COMPLET DE LA CARPETA ON ESTÀ UBICAT EL FITXER EN QÜESTIÓ. APLICACIÓ DE CONSOLA

- Cas IDF (Figura 106).

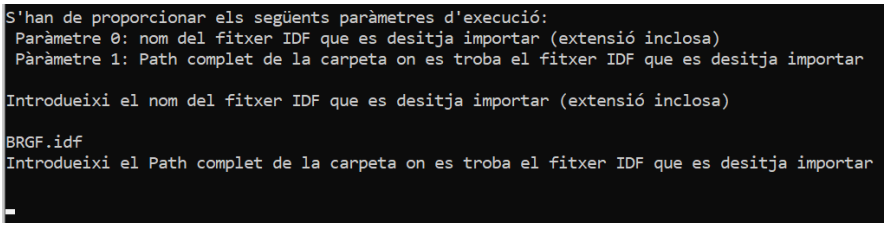


FIGURA 106. DIÀLEG ON L'USUARI PROPORCIONA EL NOM DEL FITXER IDF I POSTERIORMENT PROPORCIONA EL PATH COMPLET DE LA CARPETA ON ESTÀ UBICAT EL FITXER EN QÜESTIÓ. APLICACIÓ DE CONSOLA

Amb aquestes entrades per consola per part de l'usuari l'aplicació ja té la informació necessària per dur a terme el seu objectiu.

4.3.6.4. Resultats i *feedback*

L'aplicació informarà a l'usuari de si ha succeït algun error en la seva execució o pel contrari la seva execució s'ha efectuat correctament.

En concret pel cas de la importació d'un fitxer epJSON primerament informarà de la versió en que està el fitxer epJSON que es desitja importar i si aquella versió està suportada pel sistema. En cas de que no ho estigui el programa acaba (Figura 107 i Figura 108).

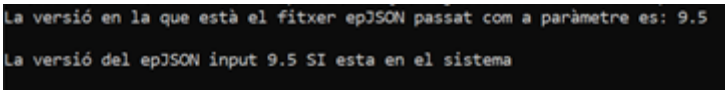


FIGURA 107. DIÀLEG ON ES MOSTRA LA VERSIÓ DEL FITXER D'ENTRADA I AQUESTA ESTÀ SUPORTADA PEL SISTEMA. APLICACIÓ DE CONSOLA

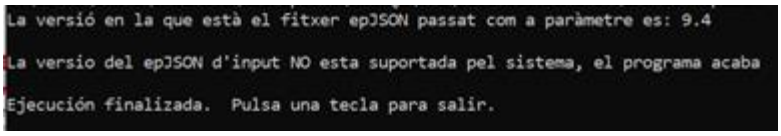


FIGURA 108. DIÀLEG ON ES MOSTRA LA VERSIÓ DEL FITXER D'ENTRADA I AQUESTA NO ESTÀ SUPORTADA PEL SISTEMA. APLICACIÓ DE CONSOLA

Un cop informat a l'usuari de si la versió del fitxer d'entrada es troba suportada pel sistema o no, se'l informarà de si ha estat validat correctament o no, i en cas de que no, s'informarà a l'usuari de l'error que ha succeït en la seva validació (Figura 109 i Figura 110).

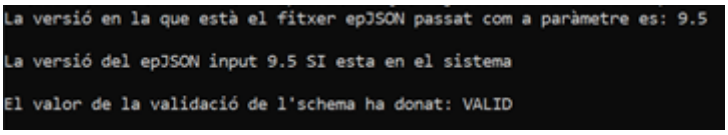


FIGURA 109. DIÀLEG ON ES MOSTRA QUE EL FITXER D'ENTRADA SI HA ESTAT VALIDAT CORRECTAMENT. APLICACIÓ DE CONSOLA

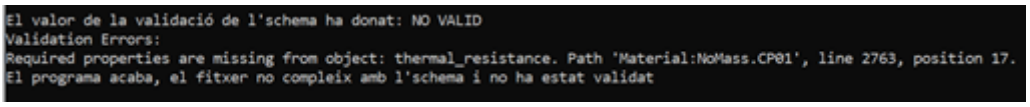


FIGURA 110. DIÀLEG ON ES MOSTRA QUE EL FITXER D'ENTRADA NO HA ESTAT VALIDAT CORRECTAMENT I L'ERROR DE VALIDACIÓ. APLICACIÓ DE CONSOLA

Aquests dos passos anteriors informatius per a l'usuari no es mostren en el cas de l'IDF ja que aquest ja està completament adaptat a Optisim i per tant, no es necessari facilitar informació addicional d'aquest tipus per consola, sinó que senzillament s'adapta el codi perquè es pugui executar com una aplicació de consola conjuntament amb el format epJSON.

Finalment tant per la importació d'un fitxer IDF com per la importació d'un fitxer epJSON se li dona la possibilitat a l'usuari de desar finalment les dades recollides del fitxer d'entrada a la base de dades o no (Figura 111).
Podem comprovar des de SSMS el valor de les taules actualitzades que ens interessi consultar. Per efectuar una consulta ràpida podem utilitzar l'opció "Select Top 1000 Rows" (Figura 112) per tal de que es generi i s'executi automàticament una query SQL sobre la taula desitjada consultant tots els seus atributs de les 1000 primeres tuples (Figura 113).

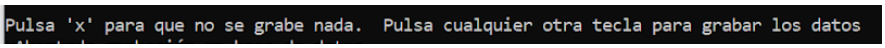


FIGURA 111. DIÀLEG ON ES DONA L'OPCIÓ A L'USUARI DE DESAR FINALMENT LES DADES RECOLLIDES DEL FITXER D'ENTRADA A LA BASE DE DADES. APLICACIÓ DE CONSOLA

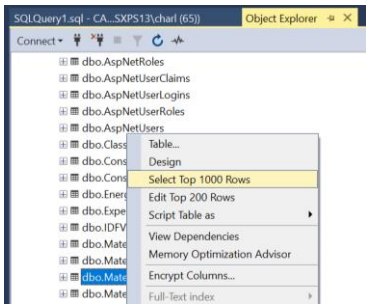
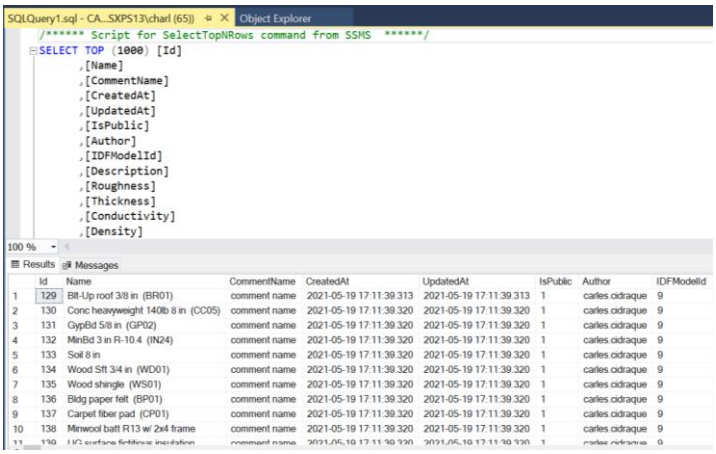


FIGURA 112. OPCIÓ "SELECT TOP 1000 ROWS" SOBRE LA TAULA MATERIALS A SSMS



	Id	Name	CommentName	CreatedAt	UpdatedAt	IsPublic	Author	IDFModelId
1	129	Blt-Up roof 3/8 in (BR01)	comment name	2021-05-19 17:11:39.313	2021-05-19 17:11:39.313	1	carles.cidraque	9
2	130	Conc heavyweight 140lb 8 in (CC05)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
3	131	GypBd 5/8 in (GP02)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
4	132	MinBd 3 in R-10.4 (IN24)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
5	133	Soil 8 in	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
6	134	Wood Sft 3/4 in (WD01)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
7	135	Wood shingle (WS01)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
8	136	Bldg paper felt (BP01)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
9	137	Carpet fiber pad (CP01)	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
10	138	Minwool batt R13 w/ 2x4 frame	comment name	2021-05-19 17:11:39.320	2021-05-19 17:11:39.320	1	carles.cidraque	9
11	139	UG surface finitious insulation	comment name	2021-05-19 17:11:39.330	2021-05-19 17:11:39.330	1	carles.cidraque	9

FIGURA 113. FRAGMENT DEL RESULTAT D'EXECUCIÓ DE LA CONSULTA GENERADA EN LA FIGURA 112

4.3.7. Guia per adaptar l'aplicació de consola a Optisim

En aquest apartat s'explica de forma resumida els passos i modificacions necessàries a realitzar per tal de poder adaptar l'aplicació de consola desenvolupada en aquest projecte a Optsim. Aquesta adaptació d'Optisim es recomana dur-la a terme de forma local (i provant tot executant Optisim de forma local) així evitant possibles modificacions no desitjades per exemple en la base de dades de NECADA.

Arquitectura general Optisim

A Optisim, a l'usuari se li mostra el formulari de creació d'un model IDF. Ha de posar un nom i una descripció, i afegir un arxiu IDF (tal i com es menciona en l'apartat 1.3.2 a la Figura 11).

Quan es prem el botó de "Upload Model", s'executa la *action* Create del controller IDFModelsController.

Aquesta *action* valida unes quantes coses, i li cedeix el control a una classe IDFModelsBL, en el que anomenem la capa de negoci (BL ≡ Bussiness Layer).

Aquesta capa BL es recolza en classes específiques (IdfParserNecada, IdfImporter, IdfExporter i IdfExporterAbstract) que s'encarreguen de parsejar el fitxer IDF, importar els seus materials i solucions constructives, i també permet exportar aquests materials cap a altres fitxers IDF. Tot això respectant la versió concreta que tingui el fitxer IDF.

Modificacions al formulari d'alta d'un model a Optisim

Com ja s'ha explicat en la memòria l'aplicació web Optisim gestiona l'execució de simulacions. Per tant, necessita incorporar fitxers IDF per a realitzar aquestes simulacions. Tal i com es va introduir en l'apartat 1.3.2 Optisim no permet editar el contingut d'un fitxer IDF, només afegir-lo com fitxer. Per afegir un fitxer IDF tenim un formulari per permet especificar el fitxer a importar (formulari presentat en l'apartat 1.3.2 en la Figura 11), així com el nom que se li donarà dins Optisim, i una descripció. Un cop importat, hi ha un altre formulari per modificar les dades de l'IDF (però no el contingut de l'IDF, que ja no canviarà mai).

En el primer formulari, el d'afegir un fitxer IDF, es realitzen diverses tasques. La primera és inserir una fila a la taula IDFModels amb les dades mostrades en aquest formulari d'alta. La segona és importar tots els materials i solucions constructives existents en l'IDF, i inserir-los en la taula de materials i solucions constructives. Per a això és necessari recórrer tot el fitxer IDF (parsejar-lo)

buscant el que ens interessa: els materials i solucions constructives. La resta del contingut de el fitxer IDF és ignorat per Optisim (sí que ho farà servir Energy Plus quan es llanci una simulació).

Per tant, caldrà modificar el formulari de creació d'un model IDF (Figura 11). Ara aquesta pantalla que conté el formulari d'alta d'un model a NECADA haurà de permetre triar el format de l'arxiu a importar (IDF o epJSON).

Una modificació addicional respecte la gestió de versions es que actualment a Optisim existeix una pantalla on es fa aquest manteniment, on s'afegeixen o es treuen les versions IDF suportades per NECADA. Com que entrem un nou tipus de fitxer i el sistema pot suportar versions IDF que no es suportin en epJSON i a la inversa, aquest formulari de gestió de versions haurà de variar per tal d'identificar si passem a suportar o deixar de suportar una determinada versió en epJSON, IDF o les dos.
A més, caldrà afegir a Optisim els schemas epJSON amb noms identificatius segons la versió a la qual donin suport per tal de dur a terme la validació a Optisim de la mateixa forma que s'ha dut a terme en l'aplicació de consola.

Model de dades.

Tal i com s'ha mencionat en l'apartat 4.3.5. Desenvolupament de l'aplicació de consola s'han intentat fer els mínims canvis possibles per tal de no afegir complexitat a l'hora de poder adaptar el codi de l'aplicació de consola a Optisim web.

El primer canvi realitzat sobre el model de dades es el vinculat amb la taula IDFModels, que ara ha passat a ser una jerarquia on hi tenim una taula base anomenada Models (amb la seva classe vinculada Model) i dues classes que hereden de la classe Model (IDFModel i EpJSONModel) que es tradueixen al model de dades com a IDFModels i EpJSONModels (en forma de discriminator a la taula Models), tot gestionat mitjançant el ORM Entity Framework, tal i com s'ha detallat en l'apartat 4.3.5.1. Introducció

El segon canvi vinculat amb el model de dades es el vinculat amb la classe IDFVersion, aquesta se li han afegit dos nous atributs (SupportIDF i SupportEpJSON) per identificar si la versió està suportada per epJSON i/o per IDF.

Les altres classes del model de dades (com Material i ConstructingSolution) no s'han modificat i per tant no les hem de tenir en compte per possibles adaptacions a Optisim. Tot i això en el cas de voler dur a terme en un fitxer epJSON l'exportació i importació utilitzant la deserialització i serialització amb les classes existents a Optisim cal afegir en les classes referents als tipus de material i a les solucions constructives els canvis mencionats al final de l'apartat on s'explica el procés d'exportació (bàsicament el indicar el nom en que l'atribut s'ha de deserialitzar i serialitzar cap al fitxer epJSON i la ignoració d'altres per tal de que no siguin exportats), modificació que serà necessària en futures iteracions per a l'adaptació completa de NECADA al nou format epJSON.

Classe model i versió: IDFModel i IDFVersion
Tal i com ja s'ha presentat s'ha creat una nova classe Model amb dues subclasses IDFModel i EpJSONModel. Cal modificar Optisim per tal que treballi amb Model + IDFModel (com a filla de Model).

Cal dir que el fet d'haver anomenat a la subclasse IDFModel amb el mateix nom que ja tenia anteriorment i que els seus atributs son exactament els mateixos ja que els hereda tots de la classe Model, les modificacions en altres fitxers serà mínima, únicament caldrà modificar aquelles linies de codi on es faci una consulta directa a la base de dades consultant a la taula IDFModels utilitzant EF, on ara s'haurà de reemplaçar per una consulta a Models tenint en compte el discriminator.

El mateix passa per IDFVersions, el nom no s'ha modificat, les úniques linies que caldrà canviar seràn les consultes directes a la base de dades que ara caldrà no només cercar el numero de la versió si no, a més a més consultar el valor de l'atribut SupportIDF. A més, tal i com s'ha mencionat en l'apartat 4.3.5.4. Incorporació i donada de baixa de versionsquan afegim o eliminem una versió hem de tenir en compte de quin format passa a ser suportada o donada de baixa.

Modificacions en el Controlador

Quan es prem el botó "Upload Model", s'executarà la action Create del controller. Per tant, en aquesta action caldrà diferenciar si l'usuari ens envia un fitxer IDF, o epJSON. Si s'envia un IDF, fem el mateix que fins ara, i si ens envia un JSON, s'executa el camí paral·lel desenvolupat per la aplicació de consola per importar el fitxer epJSON.

Classe BL (Business Layer): IDFModelsBL
Aquesta classe no tindria canvis. Però cal crear una altra classe, EpJSONModelsBL, que faci el mateix, però prenent un fitxer epJSON en comptes d'un fitxer IDF.

IdfParserNecada, IdfImporter, IdfExporter e IdfExporterAbstract
La classe IDFModelsBL es recolza en les classes IdfParserNecada, IdfImporter, IdfExporter i IdfExporterAbstract. Exceptuant IdfParserNecada, que ja no seria necessària (ja que el parsejat del epJSON s'efectua de forma més ràpida i senzilla utilitzant una llibreria), la resta de classes cal que siguin duplicades (EpJSONImporter, EpJSONExporter) perquè importessin (guardessin a la base de dades) els materials i solucions constructives. Per tant EpJSONModelsBL es recolzarà en les classes EpJSONImporter, EpJSONExporter i EpJSONExporterAbstract.

5. Conclusions

L'eficiència energètica en els edificis és i serà un dels principals reptes als que ens haurem d'enfrontar en els pròxims anys a nivell mundial. El problema energètic dels edificis afecta tots els països de diferents maneres, no només a termes de contaminació o emissions atmosfèriques sinó també pel que fa a la preservació de fonts d'energia i l'ús racional d'aquesta mateixa. Aquest TFG s'ha elaborat dins de l'entorn de NECADA, eina sorgida a la UPC davant la necessitat imperiosa de donar solució a aquest problema.

Aquest projecte ha tingut els objectius de posar les bases per a obtenir dissenys òptims de consum energètic de l'edifici de la Biblioteca Rector Gabriel Ferraté (BRGF) de la UPC tot plantant les bases del model que representarà digitalment la biblioteca i de l'exploració i l'inici d'actualització de part del sistema NECADA tenint en compte les actualitzacions que pateix el motor de simulació que fa servir anomenat EnergyPlus. Objectius estretament lligats amb la preparació de l'escenari de simulació.

La primera tasca ha anat vinculada amb l'obtenció del *Building Information Model* (BIM) de la BRGF. NECADA en l'actualitat suporta únicament fitxers BIM en format IDF, però EnergyPlus en la actualitat ja accepta un altre tipus de format anomenat epJSON el qual acabarà sent el dominant i l'únic que acabarà sent suportat per aquest motor de simulació d'eficiència energètica. Per tant, en aquest projecte per tal d'estar al dia amb EnergyPlus, s'han plantat les bases del BIM de la biblioteca en format de fitxer IDF com en epJSON, els dos formats que accepta Energy Plus en l'actualitat tot explorant diferents camins per aconseguir-ho, camins que en el cas de l'obtenció del BIM en epJSON pot servir per actualitzar projectes legacy en format IDF a NECADA perquè aquest sistema es pugui actualitzar a la última versió d'EnergyPlus sense perdre suport als antics projectes als quals donava suport.

La segona gran tasca del projecte ha detectat, explorat i donat solució a la necessitat d'adaptar el sistema NECADA per tal de permetre l'entrada i la vinculada importació i validació d'aquest nou tipus de fitxer en format epJSON representant el BIM d'un edifici o zona urbana. Per donar solució a NECADA a aquesta modificació de canvi de format de fitxer d'entrada en aquest projecte s'ha realitzat una aplicació de consola de tipus .NET Framework amb Microsoft Visual Studio. L'aplicació de consola que s'ha desenvolupat en aquest projecte té la funció de poder importar la informació que interessa a NECADA a la seva base de dades en dos formats diferents de fitxer d'entrada: IDF i epJSON, tenint en compte també la gestió de l'exportació de modificacions cap a aquest nou format de fitxer epJSON. L'objectiu final d'aquesta aplicació de consola es que el seu codi es pugui incorporar a NECADA fent els menors canvis possibles, així eliminant complexitat tot aprofitant els beneficis que aporta les noves versions d'EnergyPlus disponibles amb suport del nou format epJSON. Aquesta manera de procedir on primer es crea una aplicació de consola per després poder adaptar el codi a NECADA ja s'ha efectuat en altres projectes vinculats a aquest sistema per tal d'eliminar complexitat en les primeres fases de desenvolupament.

En conclusió, els objectius d'aquest projecte (1.4.1) s'han complert i tota la feina feta està preparada perquè es puguin dur a terme futures modificacions, extensions i adaptacions per part dels diferents integrants de l'equip de NECADA com per a qualsevol persona que es pugui veure vinculada en cercar solucions per millorar el sistema NECADA i també més en concret cercar solucions d'eficiència energètica per a la BRGF.

Bibliografía

1. **IDEAS.RePEc. [En línia] 2 / Març / 2021. https://ideas.repec.org/a/eee/rensus/v43y2015icp843-862.html.**

2. El periódico de la energía. [En línia] 2 / Març / 2021. https://elperiodicodelaenergia.com/mas-del-84-de-los-edificios-en-espana-son-energeticamente-ineficientes/.

3. European Alliance of Companies for Energy Efficiency in Buildings (EuroACE). [En línia] 2 / Març / 2021. https://euroace.org/.

4. International Energy Agency. IEA. [En línia] 2 / Març / 2021. https://www.iea.org/.

5. LEED rating system. [En línia] 2 / Març / 2021. https://www.usgbc.org/leed.

6. Comprehensive Assessment System for Built Environment Efficiency (CASBEE). [En línia] 2 / Març / 2021. https://www.ibec.or.jp/CASBEE/english/.

7. DIRECTIVA (UE) 2018/844 DEL PARLAMENTO EUROPEO Y DEL CONSEJO. [En línia] 2 / Març / 2021. https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32018L0844&from=ES.

8. Attia, S, Grattia, E i De Herde, A. Simulation-based decision support tool for early stages of zero-energy building design. ScienceDirect. [En línia] Juny / 2012. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/pii/S037877881200045X.

9. American Institute of Architecture (AIA), Energy Modeling Working Group i Building Green. AIA. Architect's guide to integrating energy modeling in the design process. [En línia] 2012. [Data: 2 / Març / 2021.] https://www.aia.org/resources/8056-architects-guide-to-integrating-energy-modeli.

10. Hensen, J. Research TUE. Towards more effective use of building performance simulation in design. [En línia] 2004. [Data: 2 / Març / 2021.] https://research.tue.nl/en/publications/towards-more-effective-use-of-building-performance-simulation-in-.

11. NECADA. Optimization for Sustainable Architecture. [En línia] [Data: 2 / Març / 2021.] https://necada.com/.

12. Polyhedra Tech: Simplifying simulation, for a better environment. [En línia] [Data: 2 / Marzo / 2021.] http://polyhedra.tech/.

13. Hofmann, E i Rüsch, M. Industry 4.0 and the current status as well as future prospects on logistics. [En línia] 2017. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/abs/pii/S0166361517301902?via%3Dihub.

14. Villani, V, Pini, F i Secchi, C. Survey on human–robot collaboration in industrial settings: Safety. [En línia] 2018. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/abs/pii/S0957415818300321?via%3Dihub.

15. Boyes, H, et al. The industrial internet of things (IIoT): An analysis framework. [En línia] 2018. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/pii/S0166361517307285?via%3Dihub.

16. *Roland Berger Digitization in the Construction Industry: Building Europe's Road to "Construction 4.0".* Berger, R. Munich : s.n., 2016.

17. MD ASLAM HOSSAIN, ABID NADEEM. TOWARDS DIGITIZING THE CONSTRUCTION . [En línia] 2019. https://www.researchgate.net/publication/334670417_TOWARDS_DIGITIZING_THE_CONSTRUCTION_INDUSTRY_STATE_OF_THE_ART_OF_CONSTRUCTION_40.

18. Woodhead, R, Stephenson, P i Morrey, D. Digital construction: From point solutions to IoT ecosystem. [En línia] 2018. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/abs/pii/S0926580517309512?via%3Dihub.

19. Bock, T. The future of construction automation: Technological disruption and the upcoming ubiquity of robotics. [En línia] 2015. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/abs/pii/S092658051500165X?via%3Dihub.

20. De Soto, B.G, et al. Productivity of digital fabrication in construction: Cost and time analysis of a robotically built wall. [En línia] 2018. [Data: 2 / Març / 2021.] https://www.sciencedirect.com/science/article/abs/pii/S092658051731124X?via%3Dihub.

21. Oesterreich, T. D i Teuteberg, F. Understanding the Implications of Digitisation and Automation in The Context of Industry 4.0: A Triangulation Approach and Elements of a Research Agenda for The Construction Industry. [En línia] 2016. [Data: 2 / Març / 2021.] https://www.researchgate.net/publication/308909169_Understanding_the_implications_of_digitisation_and_automation_in_the_context_of_Industry_40_A_triangulation_approach_and_elements_of_a_research_agenda_for_the_construction_industry.

22. *Design Principles for Industrie 4.0 Scenarios.* Hermann, M, Pentek, T i Otto, B. 49th Hawaii International Conference on System Sciences (HICSS) : s.n., 2016.

23. *Construction Automation: Research Areas, Industry Concerns and Suggestions for Advancement, Automation in Construction.* Chen, Q, García de Soto, B i Adey, B. T. 2018.

24. *Component Based Engineering of a Mobile BIM-Based Augmented Reality System, Automation in Construction.* Meža, S, Turk, Ž i Dolenc, M. 2014.

25. Hossain, M i Yeoh, J. K. W. BIM for Existing Buildings: Potential Opportunities and Barriers. [En línia] 2018. [Data: 2 / Març / 2021.] https://iopscience.iop.org/article/10.1088/1757-899X/371/1/012051.

26. *Cloud-Based BIM Governance Platform Requirements and Specifications: Software Engineering Approach Using BPMN and UML.* Alreshidi, E, Mourshed, M i Rezgui, Y. 2016.

27. Ernstrom B, Hanson D, Hill D, Clark J, Holder M, Turner D, Sundt D, Barton L, Barton T. The contractors' guide to BIM. Associated General Contractors of America. [En línia] 2006.

https://scholar.google.com/scholar_lookup?title=The%20contractors%27%20guide%20to%20BIM&author=B.%20Ernstrom&publication_year=2006.

28. Eastman C, Fisher D, Lafue G, Lividini J, Stoker D, Yessios C. An outline of the building description system. [En línia] https://scholar.google.com/scholar_lookup?title=An%20outline%20of%20the%20building%20description%20system&author=C.%20Eastman&publication_year=1974.

29. H Gao, C Koch, Y Wu. Building information modelling based building energy modelling: A review. [En línia] 2019. https://scholar.google.es/scholar?q=building+information+modelling+based+building+energy+modelling+a+review&hl=ca&as_sdt=0&as_vis=1&oi=scholart.

30.] Bragança L, Vieira SM, Andrade JB. Early stage design decisions: the way to achieve sustainable buildings at lower costs. [En línia] 2014. https://scholar.google.com/scholar_lookup?title=Early%20stage%20design%20decisions%3A%20the%20way%20to%20achieve%20sustainable%20buildings%20at%20lower%20costs&author=L.%20Bragan%C3%A7a&publication_year=2014.

31. Zimmerman A, Eng P. Integrated design process guide. [En línia] 2006. https://scholar.google.com/scholar_lookup?title=Integrated%20design%20process%20guide&author=A.%20Zimmerman&publication_year=2006.

32.] Filzmoser M, Kovacic I, Vasilescu D-C. Development of BIM-supported integrated design processes for teaching and practice. [En línia] 2016. https://img1.wsimg.com/blobby/go/d0dd54db-2225-42e2-a2f9-e42e4b1f907b/downloads/Vol62Filzmoser.pdf?ver=1561843628572.

33.] Echenagucia TM, Capozzoli A, Cascone Y, Sassone M. The early design stage of a building envelope: multi-objective search through heating, cooling and lighting energy performance analysis. [En línia] 2015. https://scholar.google.com/scholar_lookup?title=The%20early%20design%20stage%20of%20a%20building%20envelope%3A%20multi-objective%20search%20through%20heating%2C%20cooling%20and%20lighting%20energy%20performance%20analysis&author=T.M.%20Echenagucia&publica.

34. ASHRAE 209-2018. [En línia] 2018. [Data: 2 / Març / 2021.] https://www.techstreet.com/ashrae/standards/ashrae-209-2018?gateway_code=ashrae&product_id=2010483.

35. bigladdersoftware. [En línia] [Data: 2 / Març / 2021.] https://bigladdersoftware.com/epx/docs/9-3/essentials/title.html.

36. *SDL illustrated - visually design executable models.* Doldi, L. 2001.

37. EnergyPlus web. [En línia] [Data: 2 / Març / 2021.] https://energyplus.net/.

38.] Crawley DB, Lawrie LK, Winkelmann FC, Buhl WF, Huang YJ, Pedersen CO. EnergyPlus. EnergyPlus: creating a new-generation building energy simulation program. [En línia] 2001. https://www.sciencedirect.com/science/article/abs/pii/S0378778800001146.

39. Anon. GenOpt. [En línia] 2011. [Data: 2 / Març / 2021.] https://simulationresearch.lbl.gov/GO/index.html.

40. —. The DAKOTA Project. [En línia] 2011. [Data: 2 / Març / 2021.] https://dakota.sandia.gov/.

41. Fonseca i Casas, P i Fonseca i Casas, A. NECADA. OPTIMIZATION SOFTWARE FOR SUSTAINABLE ARCHITECTURE. [En línia] [Data: 2 / Març / 2021.] https://www.researchgate.net/publication/286445990_NECADA_OPTIMIZATION_SOFTWARE_FOR_SUSTAINABLE_ARCHITECTURE.

42. *Formal Simulation Model to Optimize Building Sustainability, Advances in Engineering Software.* Fonseca i Casas, P, et al. Vol. 69, p. 62–74.

43. loraxllc. [En línia] https://www.loraxllc.com/simple-box-model-its-as-simple-as-it-sounds/.

44. Wikipedia. [En línia] https://en.wikipedia.org/wiki/Digital_twin.

45. *Tansforming IFC Architectural View BIMS for Energy Simulation: 2011.* Hitchcock, R. J i Wong, J. 12th Conference of International Building Performance. Sidney : s.n., 2011, Vol. pp. 1089–1095.

46. *Annex 60: New Generation Computational Tools for Building & Community Energy Systems.* Wetter, M i Van Treeck, C. 2017.

47. [En línia] 2017. [Data: 2 / Març / 2021.] https://github.com/ORNL-BTRIC/IBPSA-BuildingSim-2017-JSON.

48. Grau en Enginyeria Informàtica. Treball de Fi de Grau. [En línia] https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/treball-de-fi-de-grau.

49. Payscale. [En línia] https://www.payscale.com/.

50. Buildings Performance Institute Europe: BPIE. [En línia] [Data: 14 / Marzo / 2021.] http://www.bpie.eu/publication/97-of-buildings-in-the-eu-need-to-be-upgraded/.

51. Sede Electrónica del Catastro. [En línia] https://www.sedecatastro.gob.es/.

52. CADMAPPER. [En línia] https://cadmapper.com/.

53. Calculate distance, bearing and more between Latitude/Longitude points. [En línia] https://www.movable-type.co.uk/scripts/latlong.html.

54. Daniel John Stine, Jeff Hanson. Autodesk Revit 2021 Architectural Command Reference. [En línia] https://books.google.es/books?id=JLnoDwAAQBAJ&pg=SA2-PA54&lpg=SA2-PA54&dq=difference+between+%E2%80%9CUse+Energy+Settings%E2%80%9D+and+%E2%80%9CUse+Room/Space+Volumes%22&source=bl&ots=zTVcSfwJWx&sig=ACfU3U0Ne-3Ka4jj0m1LiIPu80EnCxwzKw&hl=ca&sa=X&ved=2ahUKE.

55. SQL Server Management Studio (SSMS). [En línia] [Data: 2 / Març / 2021.] https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15.

56. Entity Framework Tutorial. [En línia] https://www.entityframeworktutorial.net/what-is-entityframework.aspx.

57. *EnergyPlus Performance Improvements via JSON Input Refactoring.* Adams. B, Mark i New, Joshua Ryan. United States : s.n., 2018.

58. ECMA International. [En línia] [Data: 2 / Març / 2021.] https://www.ecma-international.org/publications-and-standards/standards/ecma-404/.

59. IESVE. [En línia] https://www.iesve.com/.

60. gbXML online validator. [En línia] https://validator.gbxml.org.

61. Inheritance in Entity Framework. [En línia] https://www.learnentityframeworkcore.com/inheritance.

62. Microsoft Docs .NET using statement (C# Reference). [En línia] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/using-statement.

63. Microsoft Docs .NET DbContext Class. [En línia] https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbcontext?view=entity-framework-6.2.0.

64. Microsoft Docs .NET DbSet Class. [En línia] https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbset?view=entity-framework-6.2.0.

65. Microsoft Docs .NET SqlConnection.BeginTransaction Method. [En línia] https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlconnection.begintransaction?view=dotnet-plat-ext-5.0.

66. Microsoft Docs .NET SqlTransaction Class. [En línia] https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqltransaction?view=dotnet-plat-ext-5.0.

67. JObject.Parse Method (String). [En línia] https://www.newtonsoft.com/json/help/html/M_Newtonsoft_Json_Linq_JObject_Parse.htm.

68. Newtonsoft.Json.Linq Namespace. [En línia] https://www.newtonsoft.com/json/help/html/N_Newtonsoft_Json_Linq.htm.

69. p5.js | loadJSON() Function. GeeksforGeeks. [En línia] https://www.geeksforgeeks.org/p5-js-loadjson-function/.

70. Json.NET Schema Documentation Newtonsoft. [En línia] https://www.newtonsoft.com/jsonschema/help/html/M_Newtonsoft_Json_Schema_JSchema_Parse.htm.

71. Complete JSON Schema framework for .NET. [En línia]
https://www.newtonsoft.com/jsonschema.

72. JProperty Class Newtonsoft. [En línia]
https://www.newtonsoft.com/json/help/html/T_Newtonsoft_Json_Linq_JProperty.htm.

73. techterms.com. [En línia] https://techterms.com/definition/pascalcase.

74. N, Larsson. The integrated design process; history and analysis. International Initiative for a Sustainable Built Environment. [En línia] 2009.
https://scholar.google.com/scholar_lookup?title=The%20integrated%20design%20process%3B%20history%20and%20analysis&author=N.%20Larsson&publication_year=2009.