Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: EPSMap –Aplicació web de navegació interna pels edificis de la EPS

Document: Memòria

Alumne: Sergi Magret Goy

Tutor: Josep Blanes i Gimferrer
Cotutor: Òscar Flores Surós
Departament: ATC – Arquitectura i Tecnologia de Computadors
Àrea: ATC – Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): Setembre 2022

1. Agraïments

A en Josep Blanes i l'Òscar Flores, per el seu guiatge durant la realització d'aquest projecte.

Als meus pares, Asun i Josep Mª, pel recolzament i suport al llarg de tota la meva etapa estudiantil, sempre facilitant-me la feina a l'hora d'estudiar i ajudant amb tot el que han pogut i més per a que tot segueixi endavant.

A tots els meus amics, sense les llargues tardes i nits distraient-me dels estudis creieu-me que no hauria estat possible. Gràcies.

Finalment, a tots els professors i companys que m'he anat trobant al llarg del camí per ser tant propers i disposats a ajudar.

Moltes gràcies a tots.

2. Índex

3. Índex de figures

4. Introducció

L'objectiu d'aquest projecte és realitzar un sistema de navegació interna pels edificis de l'Escola Politècnica Superior en forma d'instruccions seqüencials conversacionals.

Igual que quan vas a una ciutat nova no saps ubicar els llocs i necessites algun sistema de guiatge per arribar-hi, quan arribes per primer cop a algun edifici de la Politècnica, tampoc et saps ubicar del tot. Aquest problema el pot tenir tant un alumne acabat d'arribar com professors, visitants, o inclús alumnes que ja fa un temps que hi estudien i tampoc sabrien trobar un despatx en concret. Aquest projecte té com a objectiu resoldre aquesta qüestió, tot realitzant un sistema de navegació interna pels diferents edificis de l'EPS, en forma d'instruccions seqüencials conversacionals.

Aquest sistema de navegació utilitzarà una aplicació web juntament amb una base de dades relacional. La pàgina web utilitzarà els llenguatges clàssics, és a dir, JavaScript, HTML i CSS per a construir el lloc i PHP i MariaDB per a gestionar la cerca de rutes i el model de base de dades. Per aconseguir la localització inicial de l'usuari, s'usaran unes etiquetes QR situades en llocs estratègics dels edificis (p. ex., a les entrades).

Per a poder navegar i donar direccions dins els edificis, s'haurà de fer un procés de *mapeig* d'aquests, amb això s'aconseguirà un graf dirigit on es podran buscar diferents camins. Aquest graf estarà format per zones d'arribada, punts d'intersecció, punts de destinació i segments.

Per tant, els **objectius** concrets d'aquest projecte es podrien resumir en el següent:

- Dissenyar una **estructura per a la base de dades** que permeti guardar la informació necessària: zones d'arribada, espais, punts de destinació, instruccions,...
- El sistema ha de ser **multiidioma**, per tant hi ha d'haver les mateixes instruccions traduïdes.
- Dissenyar un **sistema de classes** per a treballar amb tota la informació de forma àgil i fàcilment escalable.
- Implementar un algorisme de navegació dins un graf dirigit.
- Les instruccions i arestes s'han de poder reutilitzar en diferents camins.
- Dissenyar la **pàgina web** per a que l'usuari final pugui accedir a aquesta informació.
- L'aplicació final ha de ser el més senzilla possible, l'únic requeriment que necessita l'usuari és un navegador web.

Amb els objectius fixats, també és important deixar clars alguns aspectes que queden fora l'abast d'aquest projecte:

- El *mapeig* dels edificis de moment es farà manual i no s'implementarà cap sistema de *mapeig* automàtic.
- Per al visionat del graf existent s'utilitzarà un mapa manual, en cap moment es dibuixarà el graf sobre un mapa, ja que l'aplicació seria l'equivalent digital a preguntar a una persona les direccions per arribar a un lloc i aquesta persona ens respongués amb instruccions una darrera l'altra.

5. Viabilitat

Com s'ha comentat als objectius inicials, a l'hora d'implementar aquest projecte, s'ha començat amb la idea de que els recursos tecnològics necessaris per a la seva posada en marxa siguin els mínims, per tant l'únic que es necessitarà per a la posada en marxa de l'aplicació és un servidor amb PHP 7.4.X, MariaDB 5.5+ i Apache 2.4.6+.

En ser un projecte per a la Universitat de Girona el servidor utilitzat és de la UdG.

En quan a l'usuari final, només necessita d'un navegador web, connexió a internet i una càmera per a capturar el codi QR del punt d'inici.

Els costos humans són exclusivament en temps personal de l'autor, pel que no suposen cap impediment econòmic.

6. Metodologia

Per a la organització del projecte i desenvolupament del mateix he utilitzat el **Kanban**. El Kanban és una metodologia per definir, gestionar i millorar serveis o projectes. Ajuda a visualitzar la feina a realitzar, maximitzar l'eficiència i la millora continua. [1]

Hi intervenen dos elements principals:

- *Kanban Boards*: La pissarra és on es mostren totes les tasques. En la seva vesant més bàsica té 3 fases: Per Fer, En Progrés i Fet.
- *Kanban Cards*: En aquesta metodologia cada tasca a realitzar es representa amb una targeta. Aquestes targetes es van movent de fase en fase.

El funcionament de Kanban per als equips de treball és el següent:

Hi ha la primera fase on es van situant totes les tasques a realitzar, normalment la fase Per Fer (To-Do), ordenades de dalt a baix de més prioritàries a menys. El que ha de fer l'equip és anar agafant targetes de la part de dalt i movent-les cap a la fase de En Progrés (Doing). Un cop han acabat la tasca la mouen a la fase Fet (Done) i repeteixen el procés agafant la targeta de dalt de tot de la primera fase.

Aquest funcionament fa que sigui molt senzill visualitzar la quantitat de tasques que hi ha per realitzar, les que s'estan realitzant i les que ja s'han realitzat. També fa que pel coordinador de l'equip pugui modificar la prioritat de les tasques que hi ha per realitzar sense afectar a cap altre membre de l'equip.

En el meu cas, com es pot veure a la Figura 1, he utilitzat les tres fases descrites, Per Fer, En Progrés i Fet juntament amb la eina Trello. Per a identificar les diferents tasques que s'han de realitzar he creat 6 etiquetes:

- Informe: Tasques relatives a aquest informe.
- Web: Tasques relatives a la pàgina web que veurà l'usuari.
- Test: Tasques relatives a les diferents proves a realitzar.
- Core: Tasques relatives a les diferents classes en PHP i base de dades.
- Servidor: Tasques relatives a la instal·lació de l'aplicació al servidor de la EPS.
- Bug: Tasques relatives a errors que he anat trobant i corregint.

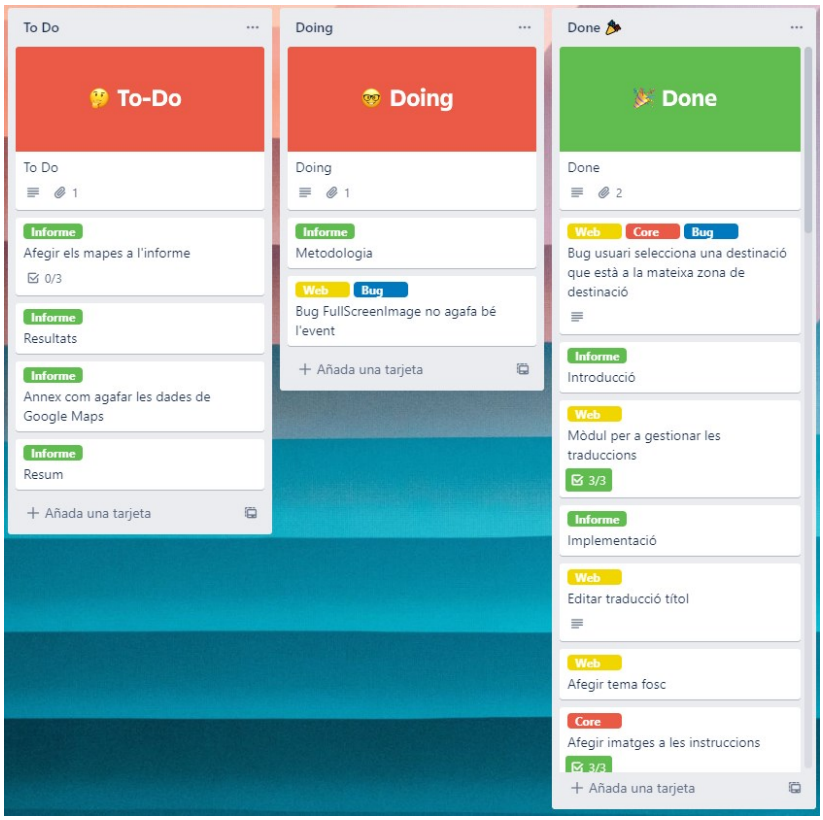


Figura 1 - Tauler Kanban

7. Planificació

Per a organitzar la planificació he utilitzat un **diagrama de Gantt** on es mostren les diferents passes que he realitzat per arribar al projecte final. Aquest diagrama es pot veure a la Figura 2.

Per començar s'havia de tenir clar quins eren els requisits inicials que se'm donaven pels tutors, aclarir els dubtes necessaris i estudiar-los per tal de prendre les decisions de disseny correctes. Aquest procés es va acabar a mitjans desembre del 2021.

Un cop definits els requisits, va ser moment de posar-me amb el disseny, tant de base de dades com de classes. Aquestes dues tasques van acabar a mitjans febrer del 2022. En aquesta secció també vaig fer una mica de disseny inicial de la pàgina web per tenir una idea de com havia de ser i a veure si als tutors els agradava. El disseny es va acabar a principis de març del 2022.

Fet el disseny de què s'havia de programar va ser l'hora de posar-me a programar l'aplicació. Vaig començar des de la base de dades, seguint per les classes en PHP i finalment la pàgina web. Gràcies a les fletxes es poden veure les tasques que en requerien d'altres. Per exemple, per a introduir les dades al sistema primer vaig necessitar agafar aquestes dades, però vaig poder començar a fer tests abans de tenir totes les dades introduïdes. Aquesta secció és la que hi havia més tasques en paral·lel, ja que mentre s'implementa sempre s'han d'anar fent tests i també s'ha de documentar tot el codi. L'aplicació va estar acabada a mitjans juliol del 2022, tot i que posteriorment s'han fet canvis al codi per arreglar certs problemes trobats.

Finalment el que queda és acabar d'escriure aquest informe i fer la presentació.

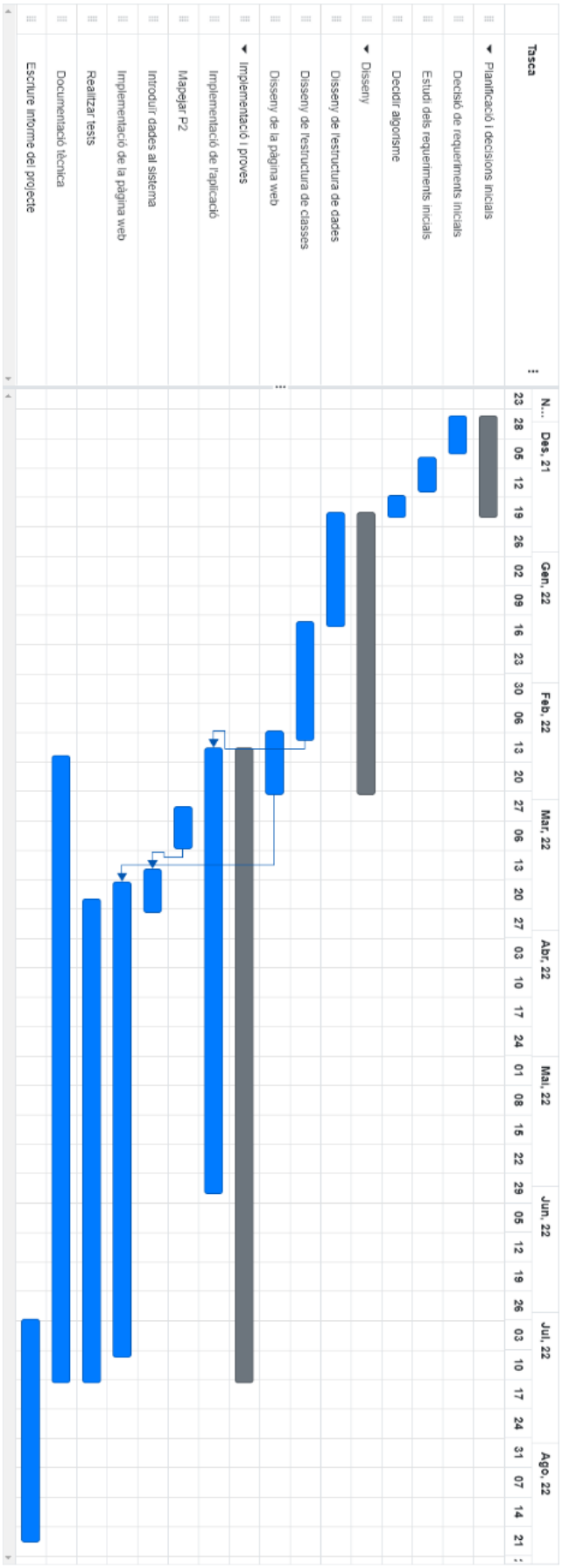


Figura 2 - Diagrama de Gantt amb les tasques realitzades, realitzat amb www.onlinegantt.com

8. Marc de treball i conceptes previs

8.1. Aplicacions Web

Una aplicació web és un programa que està guardat a un servidor remot i es distribueix als usuaris a través dels navegadors d'internet. Qualsevol component d'una pàgina web que faci una funció per a l'usuari es podria definir com a aplicació web.

No hi ha necessitat de descarregar les aplicacions web, ja que s'hi poden accedir simplement utilitzant un navegador com Google Chrome, Mozilla Firefox, Microsoft Edge o Safari, entre d'altres.

Perquè una aplicació web funcioni, necessita com a mínim 3 components: un **servidor web**, que s'encarrega de gestionar les peticions i el tràfic que prové dels clients; un **servidor per a l'aplicació**, que s'encarrega d'executar la funció final; i una **base de dades**, la qual s'encarrega de mantenir les dades i un estat al llarg del temps. Aquests 3 components, tot i que s'anomenin per separat, solen conviure dins el mateix servidor.

La majoria d'aplicacions web estan escrites en HTML5, CSS i JavaScript, o actualment amb algun framework[1] de JavaScript, tot i que al final sempre s'acaba traduint el codi escrit a aquests 3 llenguatges. Aquests es coneixen com a llenguatges de front-end, és a dir, construeixen l'estructura, l'aparença i el dinamisme que tindrà la pàgina final que veu l'usuari, respectivament.

Perquè l'aplicació web funcioni també s'ha de tenir en compte el back-end, que és on intervindran els 3 components mencionats anteriorment, servidor web, servidor per a l'aplicació i base de dades i on l'usuari final mai hi hauria de tenir accés directament, només el programador i administrador de l'aplicació. El back-end actualment sol estar programat amb algun dels següents llenguatges (i els seus respectius framework): JavaScript (Node.JS), PHP (Symfony, Laravel), Python (Django), Java (Spring, JSP), .NET (ASP). [2]

Per a entendre el funcionament intern de l'aplicació, primer s'ha de parlar dels **grafs** i dels diferents **algorismes d'encaminament** que existeixen per a buscar rutes dins d'aquests.

8.2. Grafs

Els grafs fan possible representar l'estructura d'un gran nombre de situacions d'una manera senzilla, especialment aquelles en les quals es té una col·lecció d'elements amb relacions binàries.

Un exemple clàssic és el de descriure una xarxa de comunicacions: ciutats unides per carreteres, línies de ferrocarril, enllaços telefònics, xarxes elèctriques... O en el nostre cas diferents punts geogràfics units per passadissos.

El problema que volem resoldre és d'optimització, on es tracta de fer mínima la distància recorreguda entre dos punts del graf. Exemples típics d'optimització són el problema del viatjant de comerç o els set ponts de Königsberg [3].

Un graf consta d'un conjunt V d'elements anomenats vèrtexs i un conjunt A de connexions entre aquests vèrtexs anomenades arestes. Gràficament, els vèrtexs es representen per punts del pla i

[1] Framework: Eina que proporciona components o solucions prefetes que poden ser personalitzades per tal d'agilitzar el procés de desenvolupament de software. [20]

les arestes per segments que connecten els vèrtexs corresponents. Es pot veure un exemple de graf a la Figura 3 [4].

$$V = \{1,2,3,4,5\}$$
$$A = \{\{1,2\},\{5,1\},\{5,4\},\{4,2\},\{3,4\},\{3,5\},\{2,3\}\}$$

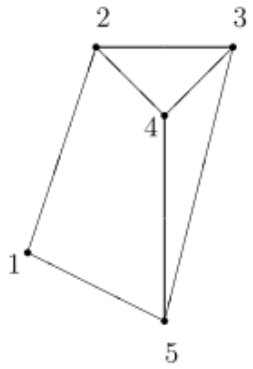


Figura 3 - Graf amb 5 vèrtexs i 7 arestes [4]

Per aquest projecte ens interessa una variant concreta dels grafs que són els **grafs dirigits**. No sol passar, però igual que hi poden haver passadissos que són només d'una direcció, també ens interessa que la seva aresta associada sigui només en un sentit. Un exemple d'aquesta utilitat seria quan hi ha portes d'un sol sentit, o quan als passadissos de l'EPS es recomanava circular només en un sentit per culpa de les restriccions de la COVID-19.

Amb un graf dirigit es pot aconseguir sense problema un graf no dirigit simplement fent que totes les arestes siguin bidireccionals. Com es veurà, per defecte a l'aplicació se suposarà que totes les arestes són bidireccionals, excepte si es diu el contrari.

L'altra variant que ens interessa són els **grafs ponderats**. Tota carretera porta associada la distància entre els pobles i ciutats que connecta. Aquesta distància és el que en teoria de grafs es coneix com al cost associat a l'aresta. En el nostre cas els pesos seran les distàncies en metres entre els diferents punts, o dit d'una altra manera, les llargades dels passadissos.

L'últim concepte important de teoria de grafs que s'ha de saber per entendre el projecte són els **camins**. Un camí és una seqüència de vèrtexs on cada vèrtex de la seqüència és adjacent[2] tant al vèrtex anterior com al següent. Es poden observar diferents camins de 1 a 3 a la Figura 4.

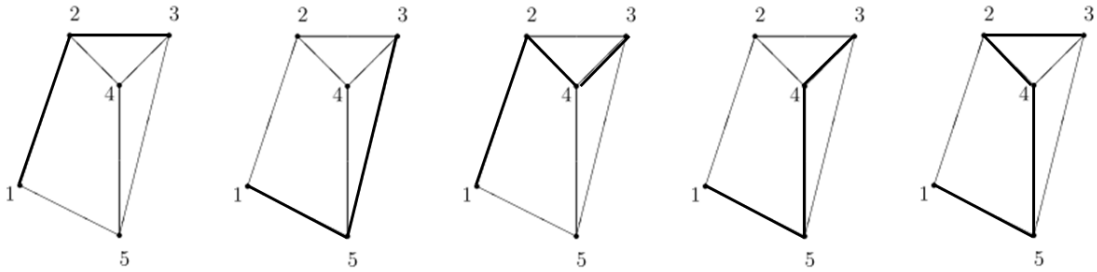


Figura 4 - Diferents camins per arribar del vèrtex 1 al 3

8.3. Algorismes d'encaminament

Per a poder guiar l'usuari final fins al seu punt de destinació ens interessarà portar-lo pel camí més curt, per a recórrer grafs hi ha diversos algorismes, els més coneguts són: Breadth First Search, Depth First Search, Dijkstra i A* (A-Star). [5]

[2] Vèrtex adjacent: Direm que dos vèrtexs $v_1, v_2 \in V$ d'un graf $G = (V, A)$, són adjacents si i només si existeix l'aresta $\{v_1, v_2\}$, és a dir, existeix una aresta al graf que uneix els dos vèrtexs.

8.3.1. Breadth First Search (BFS)

Aquest algorisme va ser publicat l'any 1959 per Edward Foorest Moore per a trobar el camí més curt dins un laberint. És un algorisme molt útil per exemple per a **trobar el punt més proper d'interès** en un sistema GPS ("Quina és la gasolinera més propera?"). El resultat d'aquest algorisme **garanteix que el camí trobat és el més curt**.

BFS (recorregut per amplada prioritària) explora totes les direccions alhora. És a dir que des del punt inicial es visiten tots els vèrtexs adjacents a aquest, després s'agafa el primer vèrtex adjacent i es visiten tots els seus adjacents, després el segon i també es visiten els seus adjacents i es torna a repetir el procés fins a arribar a la destinació.

8.3.2. Depth First Search (DFS)

Una versió d'aquest algorisme va ser estudiada per Charles Pierre Trémaux. És un algorisme molt útil per exemple per a **generar laberints o crear arbres de decisió**. El resultat ***no*** **garanteix que el camí trobar sigui el més curt**.

DFS (recorregut per fondària prioritària) explora el més lluny possible. És a dir, començant des del vèrtex inicial es camina tan lluny com sigui possible dins el graf sense formar un cicle[3], quan no es pugui avançar més es torna a la darrera bifurcació on hi hagués un o més camins i es prenen una de les arestes que no s'havien fet servir.

8.3.3. Dijkstra

Va ser publicat per Edsger Wybe Dijkstra l'any 1959 per a trobar el camí mínim entre els nodes d'un graf. Dijkstra ens permet prioritzar arestes, en comptes d'explorar totes les arestes per igual, com BFS i DFS, aquest algorisme **prioritza les arestes de menor pes**. És l'algorisme més conegut i el més utilitzat per **trobar els camins mínims entre un inici i múltiples destinacions**.

8.3.4. A* (A-Star)

Va ser publicat per Peter Hart, Nils Nilsson i Bertram Raphael l'any 1968. És una **optimització de Dijkstra per a una sola destinació**. A més de prioritzar les arestes amb menor pes (igual que Dijkstra, ja que és una extensió d'aquest), A* **prioritza les arestes que sembla que s'apropen més a la destinació final**.

És l'algorisme per defecte de la industria tecnològica quan es tracta de buscar un camí entre dos únics punts.

[3] Cicle: Camí en el que els vèrtexs inicial i final son el mateix.

9. Requisits del sistema

Al haver-se dissenyat l'aplicació des de l'inici el més senzilla possible la llista de requeriments és bastant curta.

9.1. Definicions

Al llarg del projecte i per als requeriments es fan servir uns **conceptes** que son importants entendre, aquests son:

- **Segment**: Un segment és un **camí únic i indivisible** per anar d'un punt a un altre. No es pot subdividir en altres segments. És l'equivalent a les arestes del graf.
- **Zona d'arribada**: És l'**àrea on es troba el punt de destinació**, de tal manera que aquest és accessible directament, sense que calguin més instruccions addicionals.
- **Punt de destinació**: És el **punt que ha triat l'usuari com a destinació final**. Està dins d'una zona d'arribada. En aquesta versió, serà o bé un nom de despatx o d'un espai (p. ex. "Despatx 136" o "Aula 04-A").
- **Punt de captura**: És un **punt físic amb un codi QR**, que té per funció localitzar la posició de l'usuari en el moment que aquest entra en el sistema per a fer la cerca de la destinació. Típicament es situen en els punts d'entrada als edificis, tot i que també n'hi ha d'haver dintre, perquè un usuari ha de poder començar a utilitzar el sistema en qualsevol moment. El format físic del punt ha d'observar les normes estilístiques de la resta d'etiquetes.

9.2. Requeriments funcionals

L'usuari final ha de poder realitzar de forma senzilla una acció: **seleccionar el punt de destinació final al que vol arribar**. Aquesta selecció i el llistat d'instruccions ha d'estar traduït a varis idiomes, de moment en català, castellà i anglès. Es seleccionarà el punt de partida gràcies als codis QR repartits pels passadissos.

9.3. Requeriments no funcionals

Els requeriments no funcionals inicials que se'm van donar van ser els següents:

- S'han de poder **reaprofitar les instruccions** per diversos segments.
- Les rutes d'un punt A a un punt B estaran formades per seqüències de segments.
- Cal que hi hagi el mínim de segments possible en tots els recorreguts.
- Hi ha d'haver la possibilitat **d'afegir opcionalment imatges a les instruccions** per tal d'ajudar a la seva comprensió.
- **El punt de destinació estarà dintre d'una zona d'arribada**. No és necessari que s'arribi fins al davant exacte del punt de destinació, sinó que només és necessari dur a l'usuari a una zona d'arribada. Un cop arribat, l'usuari pot buscar visualment el punt de destinació i no hauria de portar-li més de 30 segons.
- L'usuari inicia el seu trajecte davant d'un punt de captura i introdueix el punt de destinació.

10. Estudis i decisions

10.1. Software

Com s'ha comentat anteriorment, per al desenvolupament d'aquest projecte s'ha utilitzat **PHP**. S'ha usat aquest llenguatge de programació per diverses raons.

La primera i més important és que ja estava familiaritzat amb aquest llenguatge, ja que fa més de 2 anys que l'utilitzo diàriament i, per tant, m'és molt còmode treballar-hi i gràcies a això he pogut aplicar alguns conceptes que he après a la feina en aquest temps.

La segona raó és perquè tal com es mostra a la Figura 5, PHP històricament és dels llenguatges més utilitzats per al desenvolupament web, i tot i que molta gent cregui que el seu regnat s'està acabant, amb PHP 7.4 i sobretot les últimes versions de PHP 8.X, s'estan afegint moltes millores pel que fa a rendiment i funcionalitats noves com *typed properties*, *Just-In-Time Compilation*, *Fibers*... [6].

El fet que sigui tan popular, fa que hi hagi molta documentació i preguntes resoltes a internet, el que fa que resoldre problemes sigui més senzill que amb altres llenguatges de programació.

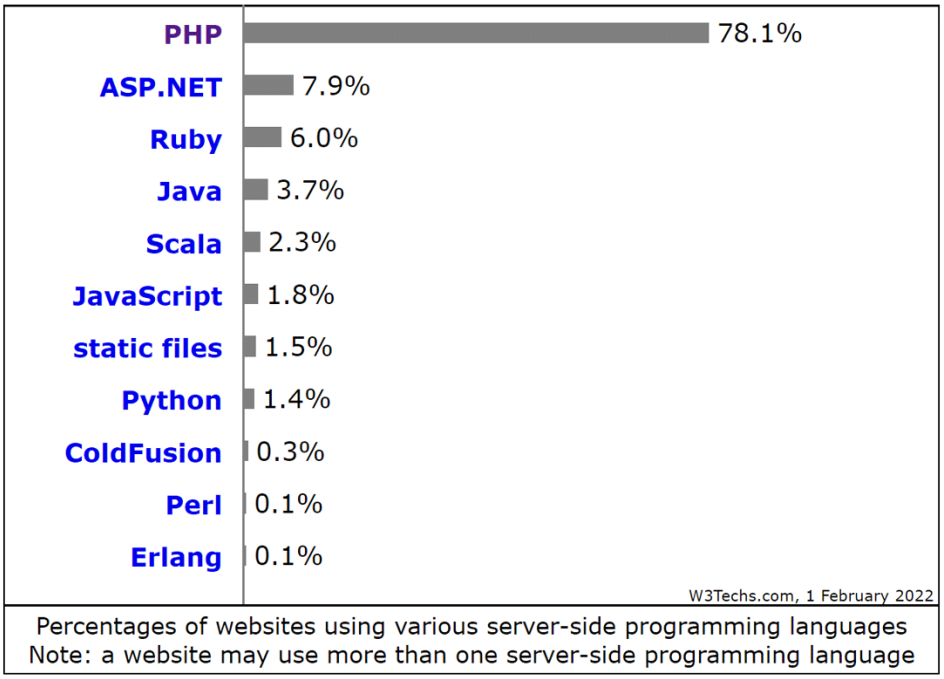


Figura 5 – Percentatges d'ús de diferents llenguatges web a 1 de febrer de 2022 amb PHP al capdavant [7]

10.2. Entorn de desenvolupament

Per al desenvolupament local de l'aplicació s'ha utilitzat XAMPP perquè és la forma més senzilla de tenir un entorn de desenvolupament que inclogui Apache, MariaDB i PHP.

XAMPP proporciona totes les eines necessàries per a tenir un servidor local, és de codi obert i està desenvolupat per *Apache Friends* amb una comunitat d'usuaris molt àmplia.

En descarregar XAMPP també es dona l'opció d'instal·lar l'eina *phpMyAdmin* la qual permet gestionar les bases de dades de forma àgil a través d'un portal web.

Per a programar he utilitzat un dels editors de codi més utilitzats actualment, el Visual Studio Code.

Per acabar amb el software utilitzat per al desenvolupament de l'aplicació, també s'ha de mencionar a Git i GitHub. Git és l'eina de control de versions més utilitzada actualment, això permet tenir un registre de tots els canvis que s'han anat efectuant al llarg del temps i qui els ha realitzat. GitHub permet guardar el codi font, utilitzant Git de rerefons, en un servidor al núvol per a poder-lo compartir amb altra gent. Tot el codi font d'aquest projecte es pot trobar sota el meu nom d'usuari penjat. [8]

10.3. Llicència

Conjuntament amb els tutors, Josep Blanes i Òscar Flores, hem decidit fer aquest projecte de codi obert amb la llicència **GNU GPLv3** [4].

El propòsit d'aquest projecte també inclou un possible futur desenvolupament per part d'altres desenvolupadors d'arreu. Pensem que pot ser un projecte interessant per a d'altres centres de tot tipus, i volem obrir al màxim aquest projecte per a que pugui esdevenir útil per a molta gent, fent això que el projecte s'enriqueixi amb les aportacions d'altres equips. A priori, sempre mantenint les decisions del projecte a la pròpia Universitat de Girona.

10.4. Llibreries externes

Per a dissenyar i programar la web he utilitzat tot un seguit de llibreries externes que faciliten la feina, aquestes son: jQuery 3.6, Bootstrap 4.6, FontAwesome 4.7 i Virtual Select 1.0.

10.4.1. jQuery

Tal com es pot veure a la Figura 6, jQuery és un dels frameworks més utilitzats per al desenvolupament web. És molt senzilla d'utilitzar, te una documentació molt extensa i ha estat altament testat.

En els últims anys ha anat perdent mercat per altres frameworks com poden ser React.js, Angular o Express; tot i això, com es pot observar encara hi ha moltíssims desenvolupadors que el fan servir en el seu dia a dia (28,57%), jo inclòs.

Aquesta ha estat la raó principal per a utilitzar aquest framework i no un altre. Igual que amb PHP, ja hi portava treballant un temps a l'hora de fer aquest projecte i, per tant, ha estat el més còmode amb el que treballar.

[4] GNU GPLv3: GNU General Public License v3.0. Aquesta llicència està condicionada a fer públic tot el codi font i les modificacions que se'n puguin fer, sota la mateixa llicència. La advertència de copyright i tipus de llicència s'han de mantenir. [25]

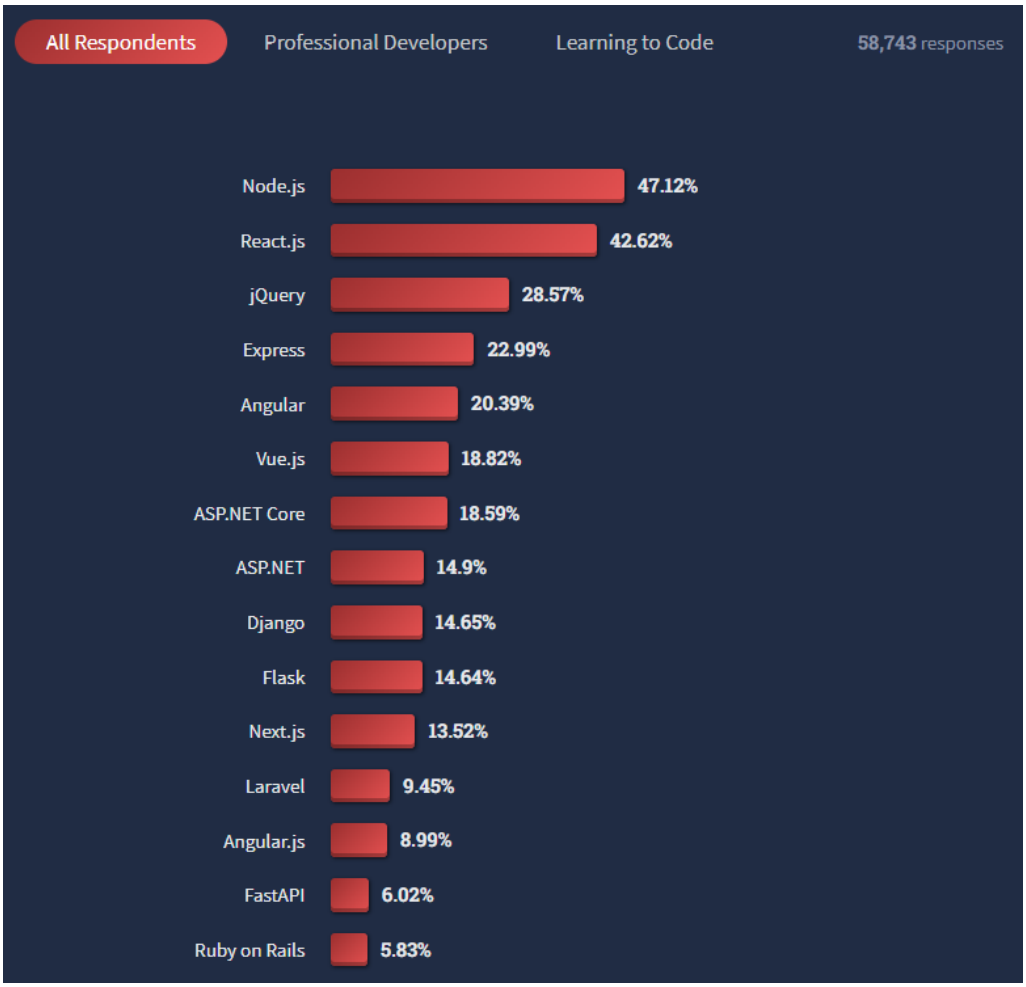


Figura 6 - Frameworks web més utilitzats, enquesta de Stack Overflow 2022 [9]

10.4.2. Bootstrap

Si jQuery m'ha facilitat la feina a l'hora de programar en JavaScript, Bootstrap m'ha ajudat a l'hora de fel el disseny visual.

Bootstrap té tot un seguit d'estils ja predefinits que estan pensats perquè la pàgina web final sigui responsiva[5]. Utilitzant els seus estils he estalviat haver de "reinventar la roda" pel que fa als estils de botons i disseny perquè quedi bé en tota mena de dispositius.

10.4.3. FontAwesome

Aquesta pàgina web es podria considerar una col·lecció d'icones en format SVG per a usar lliurement. En la versió utilitzada tenen més de 600 icones disponibles. Cada cop que es necessiti una icona per a un cas concret, en comptes d'anar a buscar a algun cercador i copiar el que interessa i guardar-lo simplement es pot anar a la seva pàgina, buscar la icona que ens interessa i copiar el codi indicat a la nostra pàgina web. Això fa que el fet de buscar icones sigui molt més ràpid i senzill. A més es poden redimensionar, canviar de color, animar... Tot dinàmicament.

[5] Disseny responsiu: És un concepte de desenvolupament web que es concentra en que les pàgines finals es vegin i comportin correctament en tot tipus de dispositius personals, des de ordinadors de sobretaula a mòbils i dispositius intel·ligents. [27]

10.4.4. Virtual Select

Aquesta llibreria segurament és la que menys gent coneix, ja que no és extremadament popular, però si és molt útil quan ens interessa tenir un cercador amb múltiples opcions dinàmiques a la nostra pàgina web. Gràcies a aquesta llibreria he pogut fer el cercador principal de forma senzilla.

10.4.5. Monolog

A diferència de les llibreries anteriors, aquesta no és per a la pàgina web final, sinó que és per al servidor de l'aplicació.

Tots sabem la importància de detectar els errors i poder-los corregir, per fer això es necessiten monitorar els errors. Una forma d'aconseguir això és escrivint tots els errors que puguin succeir a un fitxer, anomenat fitxer de logs d'error.

Aquesta llibreria s'encarrega d'això, qualsevol error que pugui aparèixer a l'aplicació p. ex. quan s'està cercant un camí, s'escriurà al fitxer necessari en un format correcte. A més de poder escriure al fitxer d'errors, si fes falta també es podria fer que els errors amb una gran urgència p. ex. si l'aplicació deixés de funcionar, s'enviessin directament a un correu concret.

10.5. Identitat corporativa UdG

En ser una aplicació molt lligada a la universitat, juntament amb els tutors vam pensar a utilitzar la identitat de marca d'aquesta, això vol dir seguir les guies que estan assenyalades pel Servei de Publicacions de la UdG, com per exemple la tipografia i els colors de la pàgina web.

Per a la tipografia de la pàgina, tal com està marcat, s'ha utilitzat la tipografia de sistema *Georgia* per als títols i *Arial* per a la resta de text. [10]

Perquè el color de l'aplicació ressalti respecte de la resta de colors de la UdG, vam decidir utilitzar com a color principal el *Pantone 7421 C*. Com a colors secundaris s'han utilitzat els *Pantone 7401 C*, *Pantone 7593* i *Pantone Reflex Blue C*, els pots trobar a la Figura 7. [11]

Figura 7 - Colors corporatius de la UdG. De dalt a baix: Pantone 7421 C, Pantone 7401 C, Pantone 7593 [11]

Juntament amb els colors també vam pensar el disseny que haurien de seguir els punts de captura amb els codis QR amb aquest color principal, vam pensar un disseny molt simple, però que fos fàcilment reconeixible, el pots veure a la Figura 8. Perquè en veure el punt de captura pensis automàticament en la pàgina web i viceversa, el disseny general de la pàgina web també ha seguit el mateix patró.

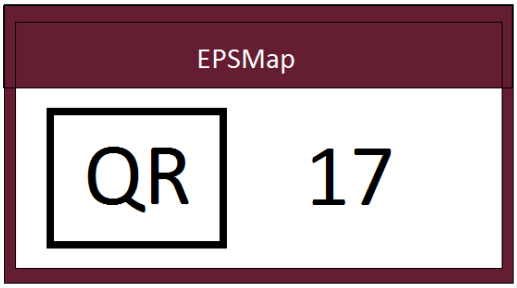


Figura 8 - Disseny inicial per als punts de captura

10.6. Algorisme d'encaminament

L'algorisme escollit per a fer la cerca de camins al graf és Dijkstra, es podria haver escollit A* sense problema, però amb Dijkstra estic més còmode treballant i al no tenir milers de nodes la diferència de rendiment serà negligible. Tot i això més endavant no hi hauria cap problema en modificar l'algorisme utilitzat ja que l'aplicació i el disseny de la persistència de dades s'ha fet independent d'aquest.

Es pot trobar el pseudocodi de Dijkstra a l'Annex 17.3 i la implementació d'aquest a l'aplicació a l'Annex 17.4.

11. Anàlisi i disseny del sistema

11.1. Disseny de la base de dades

El primer que vaig haver de fer un cop sabuts els requeriments inicials, va ser fer un disseny per a la persistència de les dades.

S'hauran de guardar principalment 7 elements:

- Arestes del graf.
- Nodes del graf.
- Zones d'arribada.
- Instruccions.
- Traduccions de les instruccions.
- Espais.
- Professors.

11.1.1. Graf

Per a guardar el graf en una base de dades relacional s'han fet servir dues taules: una de nodes i una altra d'arestes, la taula d'arestes connecta els diferents nodes.

Es pot trobar el graf intern a l'Annex 17.2, juntament amb els punts de captura i les zones de destinació. Aquests mapes els ha proporcionat el SOTIM[6]. També d'aquests mapes s'han extret els noms de les portes utilitzades.

Aquests mapes també es poden trobar en alta qualitat adjuntats a aquest informe.

11.1.1.1. Nodes

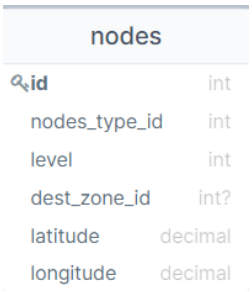


Figura 9 - Definició de la taula `nodes`

Cada node representa un punt físic en un mapa.

Com es pot veure a la Figura 9, cada node tindrà el seu identificador únic, quin tipus de node és, a quina planta està ubicat, depenent del tipus de node a quina zona d'arribada pertany i finalment la longitud i latitud d'on està ubicat el node.

Hi ha **5 tipus de node**:

1- Tipus escala: s'utilitza per a aquells nodes que estiguin ubicats a una escala.
2- Tipus espai: s'utilitza per a aquells nodes que representin un espai.
3- Tipus node del graf: aquests, son nodes interns que s'utilitzen per a les arestes i camins dins el graf.
4- Punt de captura: en aquest node hi ha ubicat un punt de captura.

[6] SOTIM: El Servei d'Oficina Tècnica i Manteniment és el responsable d'urbanisme, arquitectura, enginyeria i serveis generals en els Campus i edificis de la Universitat. [24]

100- Altres: si en un futur hi ha algun altre tipus de node que no es sàpiga com anomenar, es pot utilitzar aquest tipus de node però és preferible agafar el següent nombre disponible i crear un nou tipus de node.

El nivell ens dirà si està a nivell de carrer (0), si està a la primera planta de l'edifici (1), segona (2)...

Quan el tipus de node és un espai, el node tindrà una referència a quina zona d'arribada pertany l'espai que representa. És a dir, a quina zona d'arribada l'algorisme portarà a l'usuari final si selecciona l'espai.

Amb la latitud i longitud es pot saber exactament on està ubicat el node sobre un mapa i per tant, més endavant també es pot utilitzar per a dibuixar el graf sobre el mapa. Per agafar les coordenades he utilitzat el mètode descrit a l'Annex 17.1.

11.1.1.2. Arestes

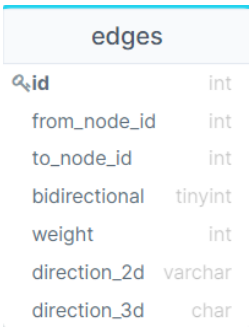


Figura 10 - Definició de la taula `edges`

L'altra part important per a la definició del graf son les arestes que connecten els nodes.

Aquestes arestes tindran com a origen i final uns nodes, normalment seran bidireccionals, tot i que hi pot haver algun cas en que no ho son, tindran un pes que serà la distància en metres entre els dos nodes i finalment tindran una direcció.

La direcció 2d serà útil per a l'hora d'arribar a una intersecció, mirant les direccions de les dos arestes, saber cap ha on a de girar l'usuari. La direcció 3d serà útil per al cas de les escales, saber si l'usuari ha de pujar o baixar una planta.

11.1.2. Zones d'arribada

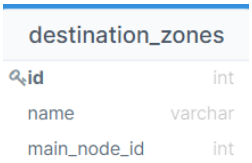


Figura 11 - Definició de la taula `destination_zones`

Les zones d'arribada és on es portarà a l'usuari quan demani per a un espai en concret. Cada zona d'arribada tindrà un o varis espais associats a ell (a través dels nodes) i tindrà un node principal que és el punt on hi haurà l'etiqueta i el punt exacte on es guiarà l'usuari.

11.1.3. Instruccions

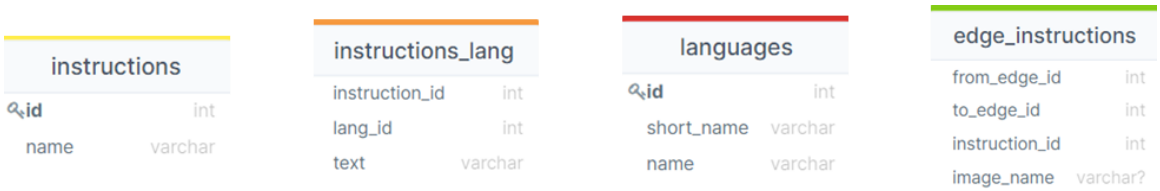


Figura 12 - Definició de les taules involucrades en guardar les instruccions: `instructions`, `instructions_lang`, `languages` i `edge_instructions`

Per a guardar les instruccions de les arestes hi intervenen 4 taules:

- `instructions`: taula on hi haurà les instruccions amb l'identificador únic. Gràcies a aquesta taula es podran reaprofitar les instruccions per a diferents interseccions.
- `instructions_lang`: taula on es guardaran les traduccions de les diferents instruccions.
- `languages`: taula on es guardaran els diferents idiomes disponibles.
- `edge_instructions`: taula on es guardarà quina instrucció s'ha de fer per a cada intersecció.

Les instruccions inicialment les vaig pensar per a ser utilitzades entre els nodes, però al començar la implementació vaig veure que hi havia alguna cosa que no quadrava. Això és perquè les instruccions han d'anar a les interseccions d'arestes, és a dir, als nodes, però aquesta instrucció ha de dependre des de quina aresta ve l'usuari i cap a quina aresta va l'usuari.

En molts casos no s'ha d'utilitzar la mateixa instrucció per anar de A a B que de B a A. Aquest cas es pot observar a la Figura 13, ja que per anar d'A a B l'usuari ha de girar a la dreta, però per anar de B a A, ha de girar a l'esquerra.

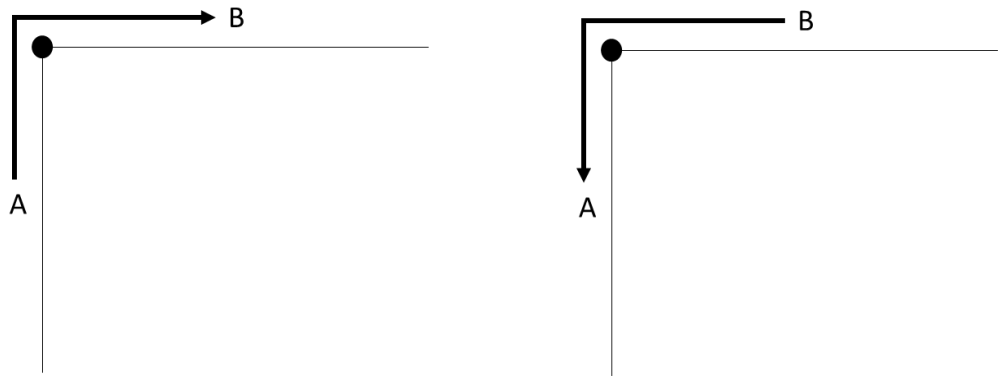


Figura 13 - Exemple on es mostra la necessitat de diferents instruccions per a les mateixes arestes

Un cop sabut això només s'ha de fer que la instrucció donada a l'usuari depengui de l'aresta entrant i l'aresta sortint.

A l'intervenir quatre taules en la definició de les instruccions donades a l'usuari i sabent que aquesta consulta serà realitzada múltiples vegades quan se sol·liciti una destinació. He creat una vista que agafa les columnes que interessen d'aquestes taules per a no realitzar un `JOIN` de quatre taules cada vegada que es vol una instrucció entre dos arestes en un determinat idioma.

11.1.4. Espais

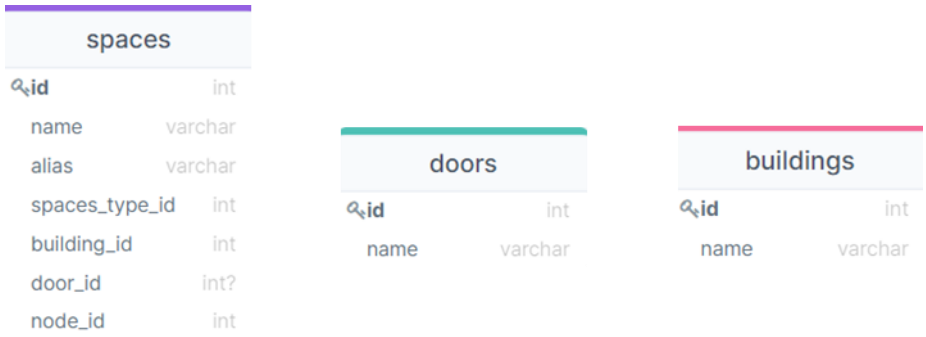


Figura 14 - Definició de les taules `spaces`, `doors` i `buildings`

L'altra part important de l'aplicació son els espais disponibles per a cercar. També s'han guardat els edificis disponibles, per a quan hi hagi dades d'espais de més d'un edifici, es puguin identificar i diferenciar fàcilment.

Cada espai tindrà un nom juntament amb un àlies, ja que a vegades els noms poden ser llargs. Quin tipus d'espai és, a quin edifici està situat, quina porta de les disponibles li correspon i quin node de tots els disponibles el representa al graf.

Com que hi ha espais que tenen més d'una porta d'entrada o sortida, la porta seleccionada per a un espai és la que es tindrà en compte a l'aplicació.

Actualment hi ha **7 tipus d'espais diferents**:

1- Classe
2- WC
3- Despatx
4- Laboratori
5- Aula informàtica
6- Hivernacle
7- Auditori

Com en el cas dels tipus de node, si en un futur es necessita un altre tipus d'espai, es pot agafar el següent nombre disponible.

11.1.5. Professors

Figura 15 - Definició de les taules `people` i `departments`

Els professors disponibles és una altra forma de buscar un espai. Al seleccionar un professor es portarà l'usuari final a l'espai assignat per a aquell professor. Clarament cada espai pot tenir més d'un professor. Si en un futur un professor es reubica, només s'haurà de canviar l'espai al que està assignat i automàticament l'aplicació portarà a l'usuari al nou espai corresponent.

S'han dividit els professors en els diferents departaments disponibles a l'Escola Politècnica Superior per facilitat de cerca i claredat de dades.

Amb tota la definició de les taules individuals, la base de dades completa amb totes les seves relacions es pot veure a la Figura 16.

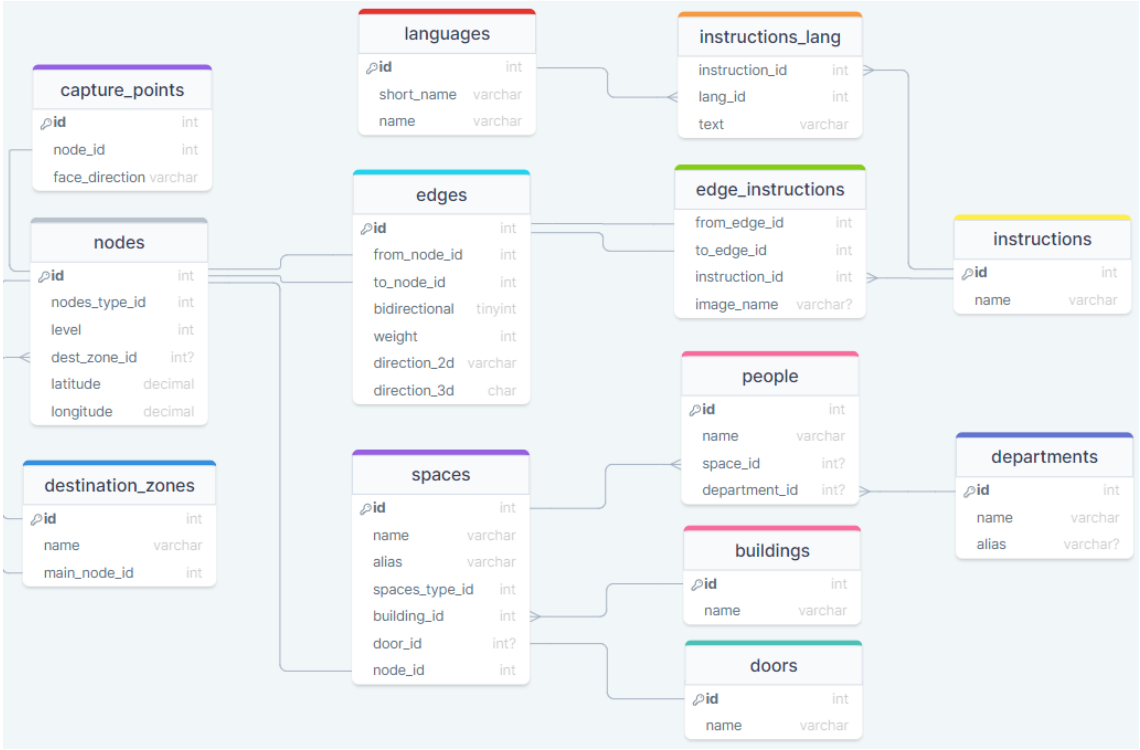


Figura 16 – Disseny final de la base de dades, creat amb DrawSQL [12]

11.2. Disseny de la pàgina web

Com s'ha anat repetint al llarg de l'informe, l'aplicació està pensada per a ser el més senzilla possible. Això sobretot inclou la pàgina web ja que serà la forma que l'usuari final tindrà de comunicar-se amb l'aplicació.

A la Figura 17 es pot veure quin era el disseny inicial. Si s'accedeix a l'aplicació final a través de https://epsapps.udg.edu/epsmap o mirant la Figura 41, podràs comprovar com des del disseny inicial fins al final no s'ha modificat en excés.

Sobretot hi ha dos parts principals:

1- Desplegable per a seleccionar la destinació a la que es vol anar.
2- Llistat d'instruccions a seguir un cop seleccionada la destinació.

El desplegable ha de servir per a seleccionar la destinació final però també per a cercar de forma senzilla mitjançant el nom o àlies d'un espai o professor.

El llistat d'instruccions ha de contenir les instruccions a seguir des de l'inici fins al final, podent incloure opcionalment imatges a les instruccions. Aquestes imatges s'han de poder obrir en pantalla completa per tal de poder-les veure correctament.

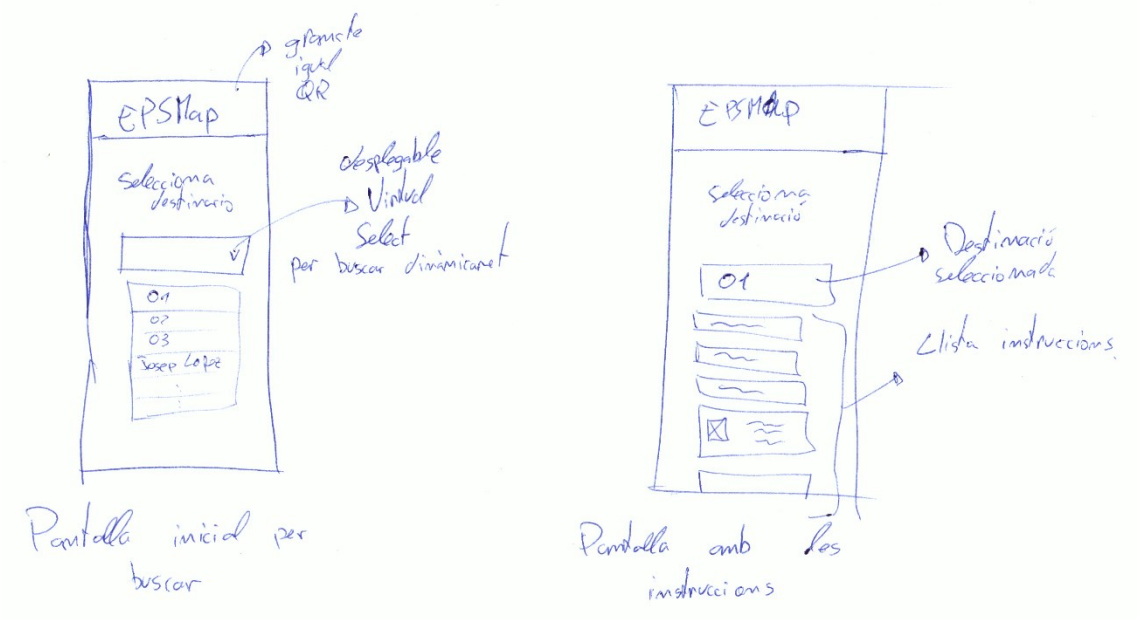


Figura 17 - Esbós inicial del disseny de la pàgina web

12. Implementació i proves

La implementació de l'aplicació es pot dividir en dos mòduls completament separats: client i servidor. La part del client està feta principalment en jQuery i la part del servidor en PHP.

12.1. Servidor

Per a treballar amb totes les dades esmentades anteriorment he dissenyat i implementat tot un seguit de classes. Aquestes classes en molts casos segueixen una estructura jeràrquica on van heretant mètodes i atributs de classes anteriors. Si et vols avançar i fer-te una idea general del disseny, el pots trobar complet a Figura 31. Començaré a explicar per la classe base i aniré baixant per l'arbre d'herències.

12.1.1. Inicialització de l'aplicació

Per entendre com es comuniquen entre elles les classes, primer s'ha de parlar com s'inicialitza l'aplicació. S'ha de recordar que **PHP és un llenguatge interpretat** i, per tant, cada cop que es fa una consulta, s'executen una llista de fitxers i un cop arribat al final es para. PHP no es manté sempre a l'escolta de peticions, el que fa això és el servidor web Apache i quan rep una petició l'envia cap al fitxer PHP que pertoca. Tot això també vol dir que a cada petició l'aplicació s'ha d'inicialitzar des de 0 carregant totes les classes.

L'aplicació sempre s'inicia des del AppInit.php i aquest fitxer s'encarrega d'incloure tota la resta de fitxers necessaris per a l'execució.

Inicialment, requereix l'*autoload* del composer[7] per tal de carregar les llibreries externes (1), a continuació carrega el fitxer de configuració (2) i el fitxer per a inicialitzar els *loggers* (3). Després s'assegura que l'include_path[8] sigui el correcte (4), requereix[9] tots els fitxers que contenen les diferents classes (5) i finalment inicialitza la connexió amb la base de dades (6) i crea l'objecte principal `EPS_Map` (7).

Per com funciona PHP, a partir del moment en què s'inclogui el fitxer AppInit.php mostrat a la Figura 18, hi haurà disponible la variable `$eps_map` arreu d'aquell fitxer. És a dir, si es té un fitxer A.php, des del punt en què es faci `require("AppInit.php")` fins al final del fitxer A.php, la variable `$eps_map` estarà disponible.

[7] Composer: Sistema de gestió de dependències externes al programa, en el meu cas el Monolog, tot i que en podrien ser més. El fitxer *autoload* és el que carrega totes les dependències descarregades a través del composer. [21]
[8] include_path: Llista de directoris on el *require* i el *include* busquen els fitxers sol·licitats. És important que hi hagi el directori on està instal·lada l'aplicació, sinó no trobarà els fitxers. [22]
[9] És important la diferència entre el *require* i el *include*: el *require* acabarà l'execució si no troba el fitxer sol·licitat, en canvi el *include* només llençarà un avís.

```php
<?php
require __DIR__ . '/vendor/autoload.php'; // (1)

require_once("conf/settings.php"); // (2)
require_once("conf/LogInit.php"); // (3)

// (4)
// If the BASE_FOLDER is not in the include path, the program can't be executed
$include_path = ini_get("include_path");
if(strpos(ini_get("include_path"), BASE_FOLDER) === false){
    // If the BASE_FOLDER is not included in the include_path try to include here
    ini_set("include_path", $include_path.":".BASE_FOLDER);
    $include_path = ini_get("include_path");
    if(strpos(ini_get("include_path"), BASE_FOLDER) === false) // If it was not added,
then die() the script
        die("You need to include the root path for the repository in the include path
through Apache config, .htacces, php.ini, .user.ini, etc\ninclude_path:
$include_path\nBASE_FOLDER: ".BASE_FOLDER);
}

// (5)
require_once("Core/Class.Logging.php");
require_once("Core/Class.DB_Access.php");
require_once("Core/Class.EPS_Map.php");
require_once("Core/Class.Graph.php");
require_once("Core/Class.Exceptions.php");
require_once("Core/Abstract.Class.DB_Object.php");
require_once("Core/Abstract.Class.Basic_Info.php");

require_once("Core/Class.Building.php");
require_once("Core/Class.CapturePoint.php");
require_once("Core/Class.Department.php");
require_once("Core/Class.Destination_Zone.php");
require_once("Core/Class.Door.php");
require_once("Core/Class.Edge.php");
require_once("Core/Class.Instruction.php");
require_once("Core/Class.Language.php");
require_once("Core/Class.Node.php");
require_once("Core/Class.Person.php");
require_once("Core/Class.Space.php");

$db_access = new DB_Access(DB_DRIVER, DB_HOST, DB_PORT, DB_NAME, DB_USER, DB_PASSWORD);
// (6)
$db_access->setLoggers(["error" => $error_logger, "debug" => $debug_logger]);
$eps_map = new EPS_Map($db_access); // (7)
$eps_map->setLoggers(["error" => $error_logger, "debug" => $debug_logger]);
?>
```

Figura 18 - Codi fitxer AppInit.php

El fitxer de configuració settings.php mostrat a la Figura 19, conté la definició d'una llista de constants necessàries per al correcte funcionament:

- `BASE_FOLDER`: És el directori base on està ubicat el programa.
- `INSTRUCTION_IMAGE_FOLDER`: Directori on es guardaran les imatges de les instruccions.
- `DEFAULT_LANGUAGE`: Si l'usuari no sol·licita cap idioma en específic s'utilitzarà aquest.
- `DB_DRIVER`: Driver per a la connexió a la base de dades.
- `DB_HOST`: Host on està ubicada la base de dades.
- `DB_PORT`: Port per a connectar-se a la base de dades.
- `DB_NAME`: Nom de la base de dades a utilitzar.
- `DB_USER`: Nom per a la connexió a la base de dades.
- `DB_PASSWORD`: Contrasenya per a la connexió a la base de dades.
- `LOG_ERROR_FILE`: Fitxer on s'escriuran els errors. És important tenir-hi accés d'escriptura.
- `LOG_DEBUG_FILE`: Fitxer on s'escriuran els missatges de *debug*.

```php
<?php

// Folders for the project
define("BASE_FOLDER", "C:/xampp/htdocs/EPSMap");
define("INSTRUCTION_IMAGE_FOLDER", BASE_FOLDER."/Core/images/");

define("DEFAULT_LANGUAGE", "ca");

// DB Configuration
define("DB_DRIVER", "mysql");
define("DB_HOST", "localhost");
define("DB_PORT", 3306);
define("DB_NAME", "eps_map");
define("DB_USER", "root");
define("DB_PASSWORD", null);

// Monolog configuration
define("LOG_ERROR_FILE", BASE_FOLDER."/Logs/error.log");
define("LOG_DEBUG_FILE", BASE_FOLDER."/Logs/debug.log");

?>
```

Figura 19 - Codi fitxer settings.php

12.1.2. Logging

Aquesta classe té com a funció principal mantenir els *loggers* de l'aplicació. Té només dos atributs públics: `$error_logger` i `$debug_logger`. Qualsevol classe que estengui aquesta, automàticament tindrà disponibles els dos *loggers* principals. El primer és l'utilitzat per a escriure els errors, el segon és utilitzat per a fer *debug* de l'aplicació.

Com s'ha vist a la Figura 18 els *loggers* s'inicialitzen al fitxer LogInit.php en fer un `require` d'aquest, a la Figura 20 es pot veure concretament com s'inicialitzen aquests dos *loggers*.

Primer es genera el format en el qual s'escriurà cada línia, en aquest cas només modifico el format de la data, després es crea un `IntrospectionProcessor` que és el que processarà els

errors. Aquest processador és interessant a l'hora d'escriure els logs, ja que inclou al missatge el fitxer, línia i funció on s'ha donat l'error. [13]

Després es genera el `Handler` per al *logger* dels errors utilitzant la constant del settings.php, que és qui s'encarrega d'escriure al fitxer i finalment es crea el `Monolog\Logger`. Es repeteix el mateix procés per al *logger* de *debug*. Necessiten `Handlers` diferents, ja que escriuen a dos fitxers diferents.

```php
<?php

$formatter = new Monolog\Formatter\LineFormatter(null, "d/m/Y h:i:s", false, true);
$processor = new Monolog\Processor\IntrospectionProcessor();


// Create and set the error logger
$handler = new Monolog\Handler\StreamHandler(LOG_ERROR_FILE, Monolog\Logger::ERROR);
$handler->setFormatter($formatter);

$error_logger = new Monolog\Logger('error', [$handler], [$processor]);


// Create and set the debug logger
$handler = new Monolog\Handler\StreamHandler(LOG_DEBUG_FILE);
$handler->setFormatter($formatter);

$debug_logger = new Monolog\Logger('debug', [$handler], [$processor]);

?>
```

Figura 20 - Codi fitxer LogInit.php

També per com funciona PHP, les variables `$error_logger` i `$debug_logger`, un cop inclòs aquest fitxer, sempre estan disponibles. És per això que al fitxer AppInit.php (Figura 18) es fan servir aquestes dues variables tot i que allà sembli que no estan definides.

Aquesta classe `Logging` l'estenen tant el `DB_Access` que és qui s'encarrega d'accedir a la base de dades, com el `EPS_Map` que és la classe principal de l'aplicació i des d'aquesta, la resta de classes poden accedir al *logger* necessari.

12.1.3. DB_Access

Aquesta classe és l'encarregada de fer totes les consultes a la base de dades corresponent. Internament, utilitza un connector que és anomenat **PDO** (PHP Data Objects). Anteriorment a PHP 5 se solia usar els mètodes mysqli per fer aquesta connexió, però eren complicats d'utilitzar, difícils de llegir i relativament senzill cometre errors de seguretat.

PDO permet utilitzar consultes preparades, és a dir, que se substitueixin certs valors sense haver de posar explícitament les variables. Això és el que fa el mètode `replaceTokens($queryStr, $substitutions)`. Se li passa un string amb la consulta i els valors a substituir i retorna la consulta preparada per a executar.

Es pot veure un exemple de consulta preparada a la Figura 21, on s'acabaran substituint els identificadors pels valors de les variables al moment de fer la consulta.

```
$queryStr = "INSERT INTO `nodes` (`nodes_type_id`, `level`, `dest_zone_id`)
             VALUES (:type_id, :level, :dz_id)";
$substitutions = [
    ":type_id" => $node_type,
    ":level" => $level,
    ":dz_id" => $destination_zone_id
];
```

Figura 21 - Exemple de substitució dels valors

Amb l'accés a base de dades hi ha dos grans diferenciats, el `SELECT`, que retorna una llista d'informació i els `INSERT`, `UPDATE` i `DELETE`, els quals només ens interessa saber si han anat bé o no.

Per al primer cas hi ha el `getResultArrayPrepared($queryStr, $substitutions)` on es passa la consulta i les substitucions i retornarà un array amb les files que compleixen la condició indexades pel nom de les columnes. A l'exemple de la Figura 22 es pot entendre millor que retorna una llista de llistes on la segona en comptes d'estar indexada de 0 a n, està indexada pel nom de la columna.

```
$queryStr = "SELECT `id`, `name` FROM `spaces`";
$resArr = $db->getResultArrayPrepared($queryStr);
/*
* Retornarà el següent:
* $resArr[0] = [
*       "id" => 1,
*       "name" => "foo"
* ];
* $resArr[1] = [
*       "id" => 3,
*       "name" => "bar"
* ];
*/
```

Figura 22 - Exemple informació retornada per el mètode getResultArrayPrepared

Per al segon cas hi ha el mètode `getResultPrepared($queryStr, $substitutions)` que en comptes de retornar un array amb els valors de la consulta retorna un booleà segons la consulta s'ha executat correctament o no.

12.1.4. EPS_Map

Aquesta és la classe principal i, per tant, molts cops el punt d'entrada per a fer altres operacions. Com a atributs de classe té la connexió amb la base de dades que es passa a l'instanciar l'objecte i una llista de noms de classes `$_classnames`.

Per què es manté una llista dels noms de les classes? Et preguntaràs. Per a poder-la modificar a un sol lloc i que s'actualitzi a la resta del codi automàticament. PHP permet instanciar classes amb mètodes estàtics a partir d'un string. [14] Això vol dir que el que es fa a la Figura 23 és

possible i funciona perfectament. A partir del nom de classe retornat pel mètode `getClassname("building")` es genera una instància de la classe que correspongui.

```php
/**
 * Get a building by its database ID
 *
 * @param int $id
 *
 * @return Building|null|false The Building instance, null if not found or
 * false on error
 */
public function getBuilding(int $id){
    return $this->getClassname("building")::getInstance($id, $this);
}
```

Figura 23 - Exemple on es mostra la instanciació de la classe Building de forma indirecta utilitzant els mètodes getClassname() i getInstance()

Dels mètodes `getInstance` i `getInstanceByData` se'n parlarà als apartats DB_Object i Basic_Info.

Aquesta classe guarda el més important, la instància del `DB_Access` per a connectar-se a la base de dades. Amb ella també es pot accedir a qualsevol altre objecte a través del seu identificador amb els *getters*, igual que els mètodes per a poder afegir qualsevol dels objectes principals a base de dades. Finalment, també té els mètodes de cerca per a aquells objectes que tenen un nom associat a ell p. ex. persones, espais, edificis... Es pot veure una llista completa dels mètodes a la Figura 24.

```
public function getDB(): DB_Access;
public function getClassname(string $class_name);

// Funcions per a crear nous objectes a la base de dades
public function addDoor(string $name);
public function addBuilding(string $name);
public function addDestinationZone(string $name, Node $main_node);
public function addDepartment(string $name, ?string $alias=null);
public function addNode(int $node_type, int $level, ?Destination_Zone
$destination_zone);
public function addPerson(string $name, ?Space $space=null);
public function addSpace(string $name, string $alias, int $space_type, Building
$building, Door $entrance_door, Node $associated_node);
public function addCapturePoint(Node $node, string $face_direction);

// Funcions per a recuperar objectes de la base de dades
public function getDoor(int $id);
public function getBuilding(int $id);
public function getDestinationZone(int $id);
public function getNode(int $id);
public function getPerson(int $id);
public function getSpace(int $id);
public function getDepartment(int $id);
public function getEdge(int $id);
public function getLanguage(int $id);
public function getLanguageByShortName(string $short_name);
public function getAvailableLanguages();
public function getInstruction(int $id);
public function getCapturePoint(int $id);

// Funcions per a cercar objectes a la base de dades
public function searchDoor(string $search_name, int $limit=50, int $offset=0);
public function searchBuilding(string $search_name, int $limit=50, int $offset=0);
public function searchDestinationZone(string $search_name, int $limit=50, int
$offset=0);
public function searchPerson(string $search_name, int $limit=50, int $offset=0);
public function searchSpace(string $search_name, int $limit=50, int $offset=0);
public function searchDepartment(string $search_name, int $limit=50, int $offset=0);
public function searchLanguage(string $search_name, int $limit=50, int $offset=0);
public function searchInstruction(string $search_name, int $limit=50, int $offset=0);

public function getAllNodes();
public function getAllEdges();
```

Figura 24 - Mètodes de la classe EPS_Map

12.1.5. JsonSerializable

En aquest cas no és una classe sinó que és una interfície, amb un sol mètode a implementar, el `jsonSerialize()`. Quan s'implementa aquesta interfície i el seu corresponent mètode s'aconsegueix convertir la instància de l'objecte a un objecte en notació JSON[10]. Gràcies a això es poden retornar les classes amb un `json_encode()` i serialitzarà el que retorni el mètode implementat. En la Figura 25 es pot veure un exemple de com està implementat aquest mètode a la classe `CapturePoint`.

```
public function jsonSerialize(): array {
    return [
        "id" => $this->getID(),
        "node" => $this->getNode(false)->jsonSerializeIDs(),
        "face_direction" => $this->getFaceDirection()
    ];
}
```

Figura 25 - Implementació del jsonSerialize a la classe CapturePoint

En la Figura 25 es pot veure com internament crida al mètode `jsonSerializeIDs()`. Això es fa per evitar recursivitat infinita, ja que si s'anessin retornant tots els objectes niats en format JSON, potser no s'acabaria mai, p. ex. cada zona de destinació té un node principal i cada node pot pertànyer a una zona de destinació, que a la vegada tindrà un node principal, que aquest node principal pertanyerà a una zona de destinació... Per evitar aquests casos, en el tercer nivell sempre es retornen els identificadors dels objectes en comptes dels objectes serialitzats. Del `jsonSerializeIDs()` també se'n parlarà a l'apartat DB_Object.

12.1.6. DB_Object

Aquesta classe abstracta és la base per a tot objecte que s'hagi de guardar a la base de dades. Té com a únic atribut una instància de `EPS_Map` que se li dona gràcies al `setEPSMap()`. Amb això tots els objectes que estenguin aquesta classe podran accedir-hi gràcies al `getEPSMap()` i en conseqüència a tots els seus mètodes.

En aquesta classe es declaren els 4 mètodes abstractes mostrats a la Figura 26.

```
protected abstract function jsonSerializeIDs(): array;
public abstract static function getTableName(): string;
public abstract static function getInstance(int $id, EPS_Map $eps_map);
public abstract static function getInstanceByData(array $resArr, EPS_Map
$eps_map): DB_Object;
```

Figura 26 - Mètodes abstractes de la classe DB_Object

El `jsonSerializeIDs()` és el mètode comentat anteriorment per a evitar la recursivitat infinita, en ser un mètode `protected` les diferents classes que estenguin aquesta, es podran cridar entre elles aquest mètode a l'hora de serialitzar-se sense problema.

Cada classe sol fer referència a una taula principal: la classe `Node` a la taula `nodes`, la classe `Building` a la taula `buildings`, etc. Per a mantenir sempre el mateix nom de taula arreu del

[10] JSON: JavaScrip Object Notation. És el format més utilitzat actualment per a comunicar-se entre el navegador i el servidor web.

codi hi ha el mètode `getTableName()` que retorna el nom de la taula a la qual fa referència aquella classe. Així, igual que amb el `getClassname()`, si es modifica a un lloc, es modificarà automàticament arreu.

Després tenim els dos mètodes `getInstance()` i `getInstanceByData()`, que cadascú ha d'implementar per tal d'instanciar-se a si mateix. En el primer cas és a partir d'un identificador únic i en el segon a partir de la fila retornada en realitzar una consulta a base de dades amb `getResultArrayPrepared()`.

Per tant, amb aquests dos mètodes, podem instanciar objectes sense necessitat de saber quina classe estem utilitzant.

Recordes la pregunta formulada a l'apartat EPS_Map? La resposta és la següent: imagina que d'aquí a un temps l'aplicació creix i els edificis passen a tenir molta més informació i per tal de no modificar el codi original es fa un `extends` de la classe `Building`.

Si les instàncies es creessin amb un `new Building()`, s'hauria d'anar arreu del codi canviant els `new Building()` per `new Building_Extended()`, cosa que seria molt carregosa. A més, en tenir més informació, s'haurà modificat la base de dades i a cada consulta es retornarà més informació, per tant, en modificar-se el constructor, s'hauran d'afegir els nous paràmetres a cada lloc on s'instanciï la classe.

En canvi, utilitzant l'estructura actual només s'han de fer un parell de modificacions perquè s'actualitzi tot el codi. Primer s'ha de canviar el nom de la classe de `$_classnames`, en comptes de ser "`Building`" serà "`Building_Extended`". Després, per a les noves columnes afegides a la taula, s'haurà de modificar el constructor per a afegir la nova informació i, aquí ve la gràcia, només s'haurà de modificar el `getInstanceByData()` ja que aquest mètode és l'únic lloc de tot el codi on es fa la instanciació *per se* de la classe.

Hem reduït haver de rebuscar per tot el codi els llocs necessaris on fer la modificació a només 3 llocs.

Es pot veure un exemple de la implementació del `getInstance()` a la Figura 27 i del `getInstanceByData()` a la Figura 28.

```php
/**
 * Get a CapturePoint instance by the Database id
 *
 * @param integer $id The unique ID for the CapturePoint
 * @param EPS_Map $eps_map Current EPS_Map instance
 *
 * @return CapturePoint|null|false The CapturePoint instance, null if not found
 * or false on error
 */
public static function getInstance(int $id, EPS_Map $eps_map){
    $db = $eps_map->getDB();
    $logger = $eps_map->error_logger;

    $tablename = self::table_name;

    $queryStr = "SELECT * FROM `$tablename` WHERE id = :id";
    $resArr = $db->getResultArrayPrepared($queryStr, [":id" => $id]);
    if($resArr === false){
        $logger->error($db->getErrorMsg());
        return false;
    }

    if(count($resArr) == 0) return null;

    return self::getInstanceByData($resArr[0], $eps_map);
}
```

Figura 27 - Implementació del mètode getInstance a la classe CapturePoint

```php
/**
 * Get a CapturePoint instance by a full database row
 *
 * @param array $resArr Database returned row
 * @param EPS_Map $eps_map Current EPS_Map instance
 *
 * @return CapturePoint
 */
public static function getInstanceByData(array $resArr, EPS_Map $eps_map): CapturePoint
{
    $instance = new self($resArr['id'], $resArr['node_id'], $resArr['face_direction']);
    $instance->setEPSMap($eps_map);

    return $instance;
}
```

Figura 28 - Implementació del mètode getInstanceByData() a la classe CapturePoint

Un cop entesos aquests dos mètodes si retrocedeixes fins la Figura 23, entendràs millor el perquè la instanciació de la classe Building es fa d'aquesta manera.

12.1.7. Basic_Info

Aquesta classe abstracta és utilitzada per a representar la informació bàsica de la majoria d'entitats que es guarden a base de dades, és a dir, l'identificador únic i el nom que els identifica de cara a l'usuari final.

Com pots observar al diagrama general de la Figura 31, la majoria d'entitats estenen aquesta classe. Gràcies a això s'evita repetir a totes aquestes classes el mateix tros de codi referent als identificadors i noms.

També implementa el mètode de cerca `searchByName()` i els dos `jsonSerialize()` i `jsonSerializeIDs()` per a la informació que guarda, ja que hi ha casos en què les entitats realment només tenen aquests dos atributs, com pot ser el cas dels edificis.

Aquesta classe te una peculiaritat en els mètodes `searchByName()` i `setName()` per tal de que aquests dos mètodes funcionin independentment de la classe que l'estengui.

Per explicar aquest cas hi ha tres paraules reservades de PHP que interessen entendre `$this`, `self` i `static`.

La primera és la més habitual quan estem parlant de programació orientada a objectes, ja que fa referència a la instància actual d'aquell objecte i només es pot utilitzar als mètodes no estàtics.

La segona també és habitual trobar-la en programació orientada a objectes, en aquest cas en els mètodes estàtics. Concretament en PHP fa referència a la classe actual on s'utilitzi aquesta paraula reservada.

Finalment, tenim `static`, que és la que ens interessa en aquest cas, és molt semblant a `self`, però en comptes de fer referència a la classe on s'utilitzi fa referència a la classe actual en temps d'execució. [15]

Amb l'exemple de la Figura 29 explicat s'entendrà millor.

```php
/**
 * Set the new name
 *
 * @param string $name
 *
 * @return boolean True on success, false on failure
 */
public function setName(string $name): bool {
    // Notice the use of static instead of self to use the implemented method in the
extended class
    $db = $this->getEPSMap()->getDB();
    $logger = $this->getEPSMap()->error_logger;

    $tablename = static::getTableName(); //          <--- AQUI ---

    // Each object will have a different table name thanks to the static
    $queryStr = "UPDATE `$tablename` SET `name` = :name WHERE `id` = :id";
    $res = $db->getResultPrepared($queryStr, [":id" => $this->getID(), ":name" =>
$name]);
    if($res === false){
        $this->error_logger->error($db->getErrorMsg());
        $logger->error($db->getErrorMsg());
        return false;
    }

    $this->_name = $name;
    return true;
}
```

Figura 29 - Implementació del mètode setName a la classe Basic_Info

A la Figura 29 hi ha la implementació del mètode `setName()` a la classe `Basic_Info`. A la línia marcada s'utilitza la paraula reservada esmentada, `static`. El mètode `getTablename()` aquí no està implementat, està deixat perquè la resta de classes l'implementin.

Si en aquest cas s'utilitzés `self::getTablename()` sempre s'executaria el mètode implementat a `Basic_Info`, utilitzant `static::getTablename()` sempre s'executarà el mètode de la classe de la instància actual. És a dir, que si es crida el mètode `setName()` sobre, p. ex. una instància de `Person`, s'acabarà executant el `Person::getTablename()` en comptes del `Basic_Info::getTablename()`. El mateix passarà amb la resta de classes.

12.1.8. Graph

La classe `Graph` és la que s'encarrega de gestionar el graf que representa el mapa de nodes i arestes i té un mètode principal: `findShortestPath()`.

Per a gestionar eficientment el graf, he utilitzat una estructura anomenada **matriu d'adjacències**. No és estrictament aquesta estructura, ja que hi poden haver files o columnes no existents, però la idea de rerefons continua essent la mateixa: una matriu de n x n on n són els nodes del graf i a la posició (i, j) hi ha l'aresta que connecta el node i amb el node j.

Amb la solució utilitzada **l'accés i consulta de nodes i arestes és constant**.

Internament, els arrays de PHP són tots *maps*, per tant, es pot guardar qualsevol mena d'informació clau-valor i el seu accés a aquests és constant.

Sabent això i per evitar haver de fer una cerca de les arestes i nodes cada cop a les llistes, he indexat tots els arrays de nodes i arestes pel seu identificador, d'aquesta manera `$nodes[$i]` conté el node amb identificador `$i`.

A l'Annex 17.4 hi pots trobar la implementació d'aquesta classe.

12.1.9. Altres classes

A part de les classes esmentades en els apartats previs hi ha totes les classes que representen les diferents entitats, professors, espais, edificis,... Aquestes no son menys importants que les explicades fins ara, però si és cert que no tenen cap peculiaritat especial.

Cada classe representa la seva respectiva taula a la base de dades i com a mínim té tants atributs com columnes té la taula. En els casos en què a la taula es guarda un identificador, a la classe corresponent també hi guardo un atribut per a guardar l'objecte al qual fa referència aquell identificador. Es podria considerar una cache.

Per exemple, cada zona de destinació té un node principal. A la base de dades es guarda l'identificador del node amb una clau forana, però un cop es carrega la informació en memòria a PHP, només se sol·licitarà la informació del node corresponent a base de dades un cop. Aquest primer cop es guardarà l'objecte que representa aquest node en memòria dins l'atribut corresponent.

Es pot veure aquest exemple a la Figura 30, inicialment al constructor es carrega l'identificador i la referència a l'objecte es deixa com a `null`. Quan es sol·licita el node, si encara no s'ha carregat en memòria, es cerca a la base de dades i es guarda a la classe per a que les properes consultes siguin més eficients.

La llista completa de classes es pot trobar a la Figura 31.

12.1.10. Constants

A part de les classes esmentades anteriorment que representen les diferents entitats, també hi ha 3 classes per a mantenir totes les constants que es necessiten. Aquestes són:

- `Node_Type`: conté la llista dels diferents tipus de node que hi ha disponibles, si es requereix un altre tipus de node, s'ha d'afegir a aquesta classe. També hi ha el mètode `isTypeValid()` per a comprovar de forma fàcil si un tipus donat és vàlid.
- `Space_Type`: conté la llista dels diferents tipus d'espais disponibles. També té el mètode per a comprovar si un tipus d'espai és vàlid `isTypeValid()`.
- `Directions`: conté la llista de direccions disponibles i de girs que es poden fer, nord, sud, sud-est, dreta, esquerra... També té dos mètodes per a saber quin és el gir a realitzar donades dues direccions, `turnDirection2D()` i `turnDirection3D()`, i quina és la direcció oposada a una direcció `getOppositeDirection()`.

```php
<?php

class Destination_Zone extends Basic_Info {

    private const table_name = "destination_zones";

    /** ID for the Node associated with this Destination_Zone
     * @var int */
    protected $_main_node_id;

    /** Attribute to act as a cache for the Node object to not request it every time
     * @var Node */
    protected ?Node $_main_node_obj;

    /**
     * Create a new Destination_Zone instance
     *
     * @param integer $id Unique database ID for the Destination_Zone
     * @param string $name Name for the Destination_Zone
     * @param int|null $espai_id ID for the Node associated with this Destination_Zone
     */
    public function __construct(int $id, string $name, int $main_node_id){
        parent::__construct($id, $name);
        $this->_main_node_id = $main_node_id;
        $this->_main_node_obj = null;
    }

    /**
     * Get the main Node associated with this Destination_Zone
     *
     * @param boolean $id Whether to return only the id of the Node or the full object
     *
     * @return Node|int|false The Node instance or the Node id depending on $id or
     * false on error
     */
    public function getMainNode(bool $id = false) {
        if($id === true) return $this->_main_node_id;
        else{
          if($this->_main_node_obj === null)
             $this->_main_node_obj = $this->getEPSMap()->getNode($this->_main_node_id);
          return $this->_main_node_obj;
        }
    }
}

?>
```

Figura 30 - Exemple d'utilització d'un atribut com a cache

Un cop descrites totes les classes que intervenen al sistema, ja podràs entendre l'estructura descrita a la Figura 31 on es mostren les relacions entre les diferents classes. En aquest diagrama he obviat molts mètodes d'algunes classes i només he deixat les parts importants que s'han explicat en els apartats anteriors.

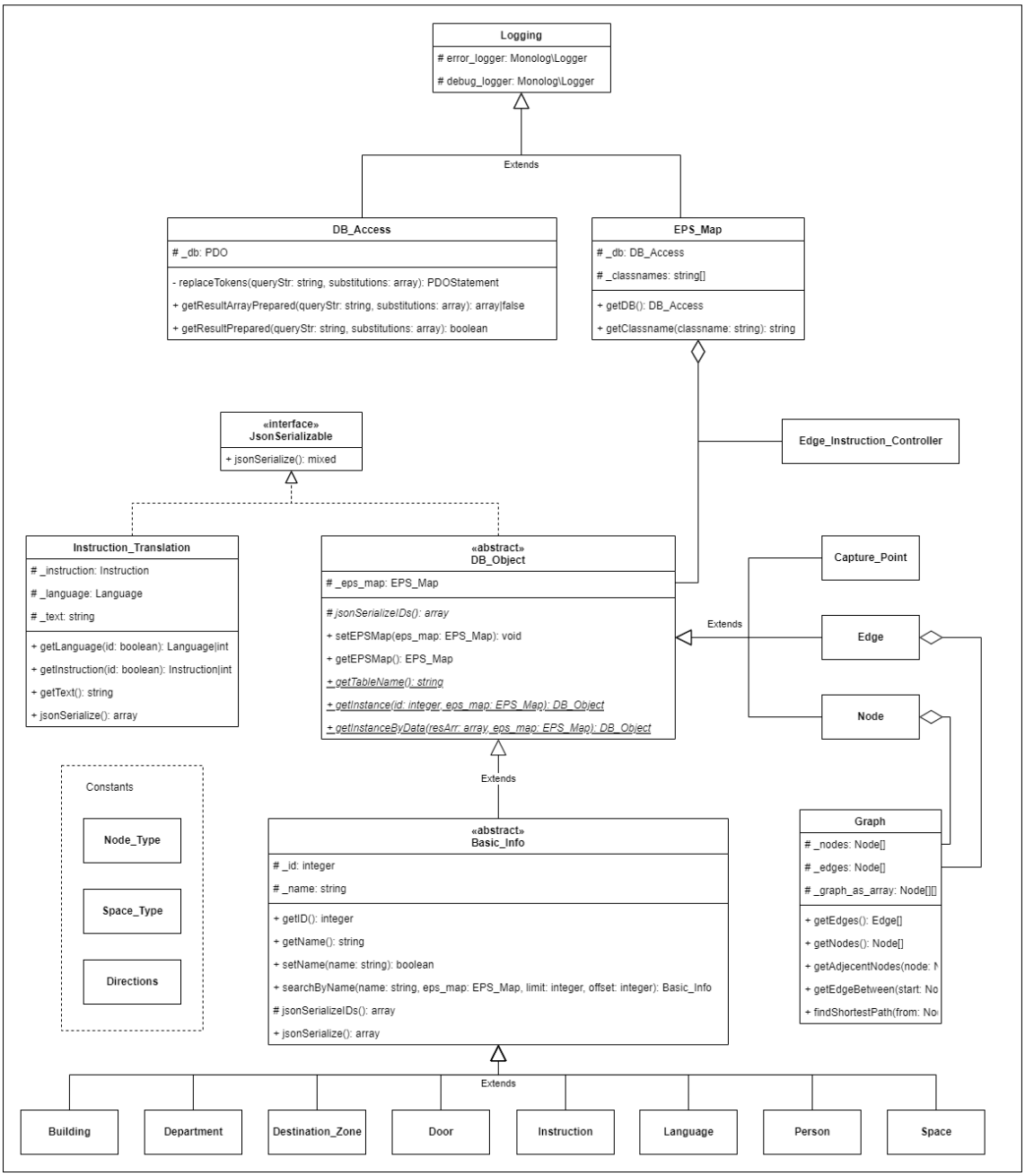


Figura 31 - Diagrama de classes complet

12.1.11. Accions

Tota aquesta estructura i disseny està molt bé, però fa falta alguna manera en què l'usuari final utilitzi totes aquestes dades. Per a aquest propòsit hi ha la carpeta actions. En aquesta carpeta hi ha les accions que es cridaran des del navegador de l'usuari.

12.1.11.1. action_functions.php

Aquest fitxer conté les funcions que les accions faran servir per a retornar informació, sigui d'èxit o d'error cap al navegador.

Pels casos d'error hi ha `endWithErrorCode()` on s'indica quin codi d'error HTTP es vol utilitzar i un missatge opcional per a la resposta.

Per als casos d'èxit hi ha la funció `endWithJSON()` que s'encarrega de fer un `json_encode()` de l'objecte rebut. Amb aquest mètode es pot retornar qualsevol objecte que implementi la interfície `JsonSerializable` i, per tant, tots els `DB_Object`.

12.1.11.2. search.php

Aquest fitxer és l'encarregat de fer la cerca inicial perquè l'usuari trobi la destinació final. Com es pot veure a la Figura 32 el primer que fa és inicialitzar l'aplicació amb el procediment explicat a l'apartat Inicialització de l'aplicació i inclou les funcions de les accions. Fet això es comprova què es vol buscar, espais o persones, i es fa la cerca del text que hagi introduït l'usuari a la base de dades.

Finalment es retornen els resultats en format JSON.

```php
<?php

require_once("../AppInit.php");
require_once("action_functions.php");

// Use FILTER_VALIDATE_BOOLEAN instead of FILTER_VALIDATE_BOOL for PHP 7.4
// since even though they are the same, the latter is only available from PHP 8.0
$include_space = filter_var($_GET['space'] ?? false,
                            FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE);
$include_people = filter_var($_GET['people'] ?? false,
                            FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE);

$limit = intval($_GET['limit'] ?? 20);

$response = [];
if($include_space)
    $response['space'] = $eps_map->searchSpace($_GET['text_search'], $limit, 0);
if($include_people)
    $response['people'] = $eps_map->searchPerson($_GET['text_search'], $limit, 0);

endWithJSON($response);

?>
```

Figura 32 - Fitxer search.php per a la cerca de destinacions

12.1.11.3. get_path.php

Aquest fitxer busca els famosos camins entre dos punts dins el graf, se li passa un node inicial o un punt de captura inicial i el node destí i retorna tot el llistat d'instruccions traduït.

Concretament retorna 6 atributs en l'objecte JSON:

- `instructions`: Llistat amb totes les instruccions a seguir. Conté la informació dels nodes i arestes intermèdies.
- `destination_zone`: Informació de la zona de destinació a la qual es portarà a l'usuari.

- `initial_turn`: Gir inicial que ha de fer l'usuari respecte el punt de captura per a posicionar-se mirant en la direcció de la primera instrucció.
- `total_cost`: Cost total en metres des del punt inicial fins al final.
- `in_place`: Per als casos en què l'usuari demana un punt final que està en la mateixa zona de destinació que el punt final.
- `only_edge`: La llista d'instruccions com s'ha vist a l'apartat Instruccions es dona entre dues arestes, però pot passar que entre el punt inicial i el final només hi hagi una aresta i, per tant, cap instrucció associada, en aquest cas aquest atribut conté aquesta aresta.

12.1.11.4. getInstructionImage.php

Finalment, aquest fitxer retorna la imatge per a una instrucció entre dues arestes. Tal com es mostra a la Figura 33, perquè els navegadors interpretin la imatge correctament s'ha de retornar amb el Content-Type[11] correcte, que en comptes de ser `text/html` per a la pàgina web o `application/json` per a les respostes JSON serà p. ex. `image/png` o `image/jpeg`. [16]

```
$instruction_ctrl = new Edge_Instructions_Controller($eps_map);
$image_path = $instruction_ctrl->getInstructionImageBetween($initial_edge,
$destination_edge);
if($image_path === false) endWithErrorCode(500, "Error getting image");
if($image_path === null) endWithErrorCode(404, "Image not found");

$mime = mime_content_type($image_path);
if($mime === false) endWithErrorCode(500, "Error getting mime type");
header("Content-Type: $mime");
echo file_get_contents($image_path);
```

Figura 33 - Retorn del mime type correcte al getInstructionImage.php

12.1.12. Proves de l'aplicació

Per a realitzar les proves unitàries hi ha el fitxer test.php on hi ha l'aplicació iniciada i es poden realitzar tota mena de crides als mètodes a través de la variable `$eps_map`.

Aquest fitxer està pensat per a fer les proves sobre el graf guardat i comprovar que la cerca de camins i instruccions és correcte. A la Figura 34 es pot veure l'inici del fitxer on s'inicialitza l'aplicació i, a més, es guarda el moment en el qual ha començat. D'aquesta manera, fent una resta al final de tot de l'execució es pot cronometrar l'eficiència de les crides que s'estan executant. Es pot veure un exemple de l'execució d'aquest fitxer de prova a la Figura 35.

En aquest exemple es pot veure com el recorregut és del node 20 fins al node 72, que és el node principal de la zona de destinació "I" que és on pertany l'espai "Despatx 204". A continuació es mostra la llista dels nodes pels quals s'ha de passar i finalment la llista d'instruccions que es donaran a l'usuari en l'idioma seleccionat.

Es pot veure com hi ha un cas concret en què falta una instrucció. Aquest cas concret és entre les arestes 37 i 35. Per a identificar-les millor hi ha entre parèntesis els nodes d'inici i final de cada aresta.

[11] Content-Type: Capçalera HTTP per a indicar el format MIME (Multipurpose Interner Mail Extensions) original del recurs retornat. En altres paraules, indica al navegador com mostrar el que rep del servidor, com una página web en forma HTML, una imatge, un PDF…

A sota de tot es mostra el temps total en segons que s'ha tardat per a calcular el trajecte seleccionat.

```php
$time_start = microtime(true);
$current_lang = "es";
require_once("AppInit.php");

$graph = new Graph($eps_map->getAllNodes(), $eps_map->getAllEdges());
echo "<pre>";

$ini_node = $eps_map->getNode(20);
$end_space = $eps_map->getSpace(5);
$end_node = $end_space->getDestinationZone()->getMainNode();

echo "Start: ".$ini_node->getID()."\n";
echo "Finnish: ".$end_space->getName()
                ." (".$end_space->getNode(true).") belonging to destination zone "
                .$end_space->getDestinationZone()->getName()." ("
                .$end_space->getDestinationZone(true).")";
echo " assigned to node ".$end_node->getID()."\n";
```

Figura 34 - Inici del fitxer test.php on es seleccionen els nodes inicials i finals

```
Start: 20
Finnish: Despatx 204 (83) belonging to destination zone P (16) assigned to
node 72

List of nodes to go by:
array(7) {
  [0]=>
  int(20)
  [1]=>
  int(21)
  [2]=>
  int(42)
  [3]=>
  int(70)
  [4]=>
  int(74)
  [5]=>
  int(73)
  [6]=>
  int(72)
}

Instructions to follow:
From 40 (20, 21) to 27 (21, 42): 3 meters -> Sube 1 piso
From 27 (21, 42) to 31 (42, 70): 10 meters -> Sube 1 piso
From 31 (42, 70) to 37 (70, 74): 10 meters -> Gira a la izquierda
From 37 (70, 74) to 35 (73, 74): 11 meters -> Missing instruction from 37
(70, 74) to 35 (73, 74)
From 35 (73, 74) to 34 (72, 73): 12 meters -> Sigue recto
11 meters

Final cost: 57 meters

Total time: 0.14300203323364 s
```

Figura 35 - Exemple d'execució del fitxer test.php

12.2. Client

En els navegadors el codi és en JavaScript i no hi ha classes com a tal, sinó que hi ha els mòduls. Per a aquesta aplicació he escrit 3 mòduls diferents. A part d'aquests 3 mòduls també hi ha guardades totes les traduccions dels textos estàtics de l'aplicació, perquè l'idioma de la pàgina web estigui en concordança amb l'idioma escollit.

12.2.1. Idiomes

He utilitzat un fitxer en format JSON per a cada idioma que es vulgui que hi hagi a l'aplicació. Es poden trobar aquests fitxers al directori languages. Aquest fitxer té format de clau valor i es pot veure un exemple per al català en la Figura 36.

```
{
    "title": "EPSMap",
    "subtitle": "Sistema de navegació intern pels edificis de la EPS",
    "select_destination": "Selecciona quina destinació vols buscar",
    "teachers": "Professors",
    "spaces": "Espais",
    "initially": "Inicialment",
    "and": "i",
    "and_then": "i després",
    "move_forward": "Avança",
    "meters": "metres",
    "you_will_have_reached_your_destination": "Hauràs arribat a la teva destinació",
    "destination_zone": "Zona de destinació",
    "interpret_results_title": "Com interpretar les instruccions?",
. . .
}
```

Figura 36 - Fitxer de traduccions al català per als textos estàtics

En carregar la pàgina web es comprova si l'usuari té seleccionat algun idioma i es carrega el fitxer corresponent a aquell idioma, si no en té cap de seleccionat, es carreguen les traduccions de l'idioma per defecte que hi hagi al fitxer settings.php.

Els continguts d'aquest fitxer es guarden a la variable `$lang_obj` i més endavant es traspassen a una altra variable JavaScript utilitzant un `const lang_obj = <?php echo json_encode($lang_obj) ?>`, després d'això es pot accedir a les traduccions a qualsevol part del codi JavaScript amb un `lang_obj.<clau>`, p. ex. amb `lang_obj.teachers`, s'obtindria "Professors". Aquesta explicació es pot veure traduïda al codi en la Figura 37.

```
<?php
require_once("AppInit.php");
$lang_str = null;
$lang_obj = null;
if(isset($_GET['lang'])) $lang_str = $_GET['lang'];
else $lang_str = DEFAULT_LANGUAGE;

$av_langs = $eps_map->getAvailableLanguages();

if(file_exists("languages/".$lang_str.".json"))
$lang_obj = json_decode(file_get_contents("languages/".$lang_str.".json"), true);
else
$lang_obj = json_decode(file_get_contents("languages/".DEFAULT_LANGUAGE.".json"),
true);
?>
. . .
<script>
const lang_obj = <?php echo json_encode($lang_obj) ?> || {};
</script>
```

Figura 37 - Càrrega del fitxer de traduccions

12.2.2. VirtualSelect_Searcher

Aquest mòdul és el que s'encarrega de generar el selector principal on s'indiqui, mostrar els resultats de la cerca de destinacions i realitzar la cerca del camí, és a dir, realitzar una crida AJAX[12] al fitxer search.php en el primer cas i al get_path.php en el segon.

Aquest mòdul rep 3 paràmetres inicials: un objecte de configuració, l'idioma seleccionat i l'objecte de traduccions descrit anteriorment.

L'objecte de configuració conté 5 atributs:

- `main`: Objecte jQuery on es carregarà el Virtual Select per a les cerques.
- `people`: Objecte jQuery que conté el checkbox[13] per si es volen cercar persones o no.
- `spaces`: Objecte jQuery que conté el checkbox per si es volen cercar espais o no.
- `capture_point_id`: Identificador del punt de captura del que es parteix.
- `onSelectDestination`: Funció que s'executarà en seleccionar una destinació. A aquesta funció es passarà l'identificador del punt final seleccionat i la llista d'instruccions retornada pel get_path.php.

Aquesta última funció és la que s'encarrega de mostrar la llista d'instruccions a l'usuari final i es pot trobar la implementació a l'Annex 17.5.

[12] AJAX: Asyncrhonous Javascript and XML. Terme utilitzat per a les consultes al servidor que es fan gràcies a la interacció de l'usuari amb la pàgina web sense haver de recarregar aquesta.
[13] Checkbox: Tipus d'element per a l'etiqueta HTML <input> per a quan es vol seleccionar un conjunt d'elements d'una llista.

12.2.3. FullScreenImage

Aquest petit mòdul s'encarrega de gestionar quan s'obre i es tanca la imatge d'una instrucció per a obrir en pantalla completa. Si es prem sobre la imatge, aquesta s'obrirà en pantalla completa i quan està oberta, si es prem a qualsevol altre lloc de la pantalla que no sigui la mateixa imatge, es tancarà. Es pot veure la implementació a la Figura 38.

Aquest mòdul es crida de forma automàtica just després de crear-se i, per tant, el *listener*[14] estarà sempre actiu.

```
/**
 * Attach an event listener to open the images in full screen and close them
 * once the user tap anywhere else than the image
 */
var FullScreenImage = function(){
    const close_image = (event) => {
        if(event.target != $("#fullscreen-image img")[0]){
            $("body").removeClass("noscroll");
            $("#dark-background").addClass("d-none");
            $("#fullscreen-image").addClass("d-none");
            $(document.body).off('click', close_image);
        }
    };

    $("#path-instructions").on("click", ".instruction-image img", event => {
        $("body").addClass("noscroll");
        $("#dark-background").removeClass("d-none");
        const fullscreen_image = $("#fullscreen-image");
        fullscreen_image.removeClass("d-none");
        fullscreen_image.empty();
        fullscreen_image.append($("<img/>", {
            src: event.currentTarget.src
        }));

        // Small timeout to attach the close listener so the user doesn't click
        // twice accidentally
        setTimeout(() => {
            $(document.body).on('click', close_image);
        }, 100);
    });
}();
```

Figura 38 - Implementació del mòdul FullScreenImage

[14] Listener: Funció que s'executarà quan es produeixi un esdeveniment, anomenats *events*, en concret. Aquests *events* normalment son deguts a la interacció de l'usuari, p. ex. quan fa *click*, selecciona o modifica un element. [23]

12.2.4. ChangeLanguage

Aquest mòdul s'encarrega d'actualitzar l'idioma de la pàgina.

Com es mostra a la Figura 39, bàsicament té un *listener* que cada cop que es canviï l'element seleccionat de la llista disponible d'idiomes, tornarà a carregar la pàgina afegint l'idioma que pertoqui a l'URL.

Per exemple si se selecciona l'espanyol, recarregarà la pàgina afegint `lang=es`. Aquest `lang` es llegirà a l'inici de la pàgina com s'ha explicat a l'apartat Idiomes i farà que es carregui el nou idioma.

```
/**
 * Listener for when the user wants to change the language of the
application
 */
var ChangeLanguage = function(){
    $("#lang").on('change', event => {
        const params = new URLSearchParams(location.search);
        params.set("lang", $(event.currentTarget).val());
        // Reload the same page with the new param
        window.location.href = window.location.pathname + "?" + params;
    });
}();
```

Figura 39 - Implementació del mòdul ChangeLanguage

13. Resultats

Els resultats d'aquest projecte es poden trobar entrant a https://epsapps.udg.edu/epsmap/ o escanejant els codis QR de la Figura 40. Aquí hi ha implementada l'aplicació esmentada per l'edifici **Politècnica II (P-II)**. Si es llegeix el codi QR de la Figura 40 (Esquerra), les instruccions començaran des de l'entrada principal, és a dir, aquest QR haurà d'estar ubicat a l'entrada, si es llegeix el QR de la Figura 40 (dreta), llavors les instruccions es donaran des del punt de captura 5, ubicat al mig del passadís del primer pis de P-II. Els diferents punts de captura que hi ha preparats es poden trobar a l'Annex 17.2 marcats amb color verd clar.

També consultant l'Annex 17.2 es poden veure els mapes de P-II proporcionats pel SOTIM amb els diferents apartats:

- En color verd fosc hi ha els nodes amb els seus corresponents identificadors únics i les arestes que els connecten.
- En color groc hi ha les portes utilitzades per a cada espai.
- En color vermell hi ha les diferents zones de destinació i els diferents espais que inclouen.
- En verd clar hi ha els punts de captura conjuntament amb la direcció on ha d'estar mirant l'usuari per a que la instrucció inicial sigui la correcta.

Com s'ha explicat al llarg de l'informe, cada espai té un node assignat, i aquest pertany a una zona de destinació, que té un node principal. En els mapes de l'Annex 17.2 es pot veure tota aquesta informació, p. ex. la zona de destinació "I" té com a node principal el 34 i conté els espais associats als nodes 45, 46 i 47. Això vol dir que si es sol·licita anar a l'espai associat al node 45, l'aplicació et portarà fins al node 34.

Tot i que no surti explícitament als mapes, els nodes de les escales dels diferents pisos també estan connectats. Per tant, els nodes 44 (nivell 0) i 37 (nivell 1) estan connectats i d'igual manera també la resta de nodes d'escales entre si.

En la Figura 41 es mostra un exemple de funcionament partint des del punt de captura 1 (porta d'entrada), fins a seleccionar la destinació final (Aula II-04B).

Figura 40 - Codis QR d'inici. Esquerra: punt de captura 1, dreta: punt de captura 5

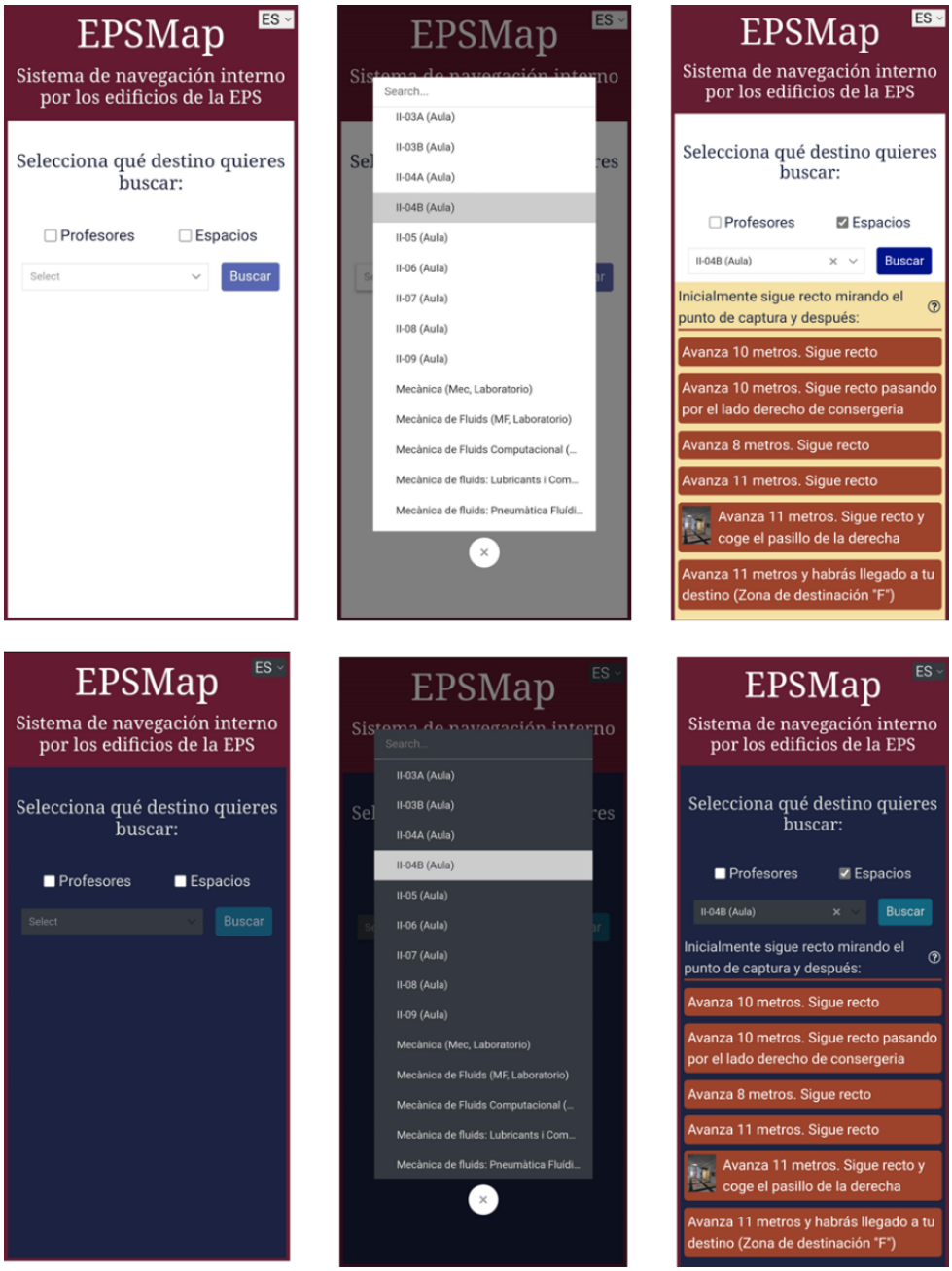


Figura 41 - Disseny final de la pàgina web, a dalt en tema clar i a baix en tema fosc. D'esquerra a dreta: pantalla inicial, pantalla de cerca i pantalla d'instruccions

Pel que respecte a la LOPD[15] i la LSSICE[16], **l'aplicació no guarda cap informació sensible** ni de la universitat, ni dels professors (els noms estan disponibles públicament al directori de la UdG), ni de l'usuari que accedeix a fer ús de l'aplicació.

[15] LOPD: Llei Orgànica de Protecció de Dades
[16] LSSICE: Llei de Serveis de la Societat de la Informació i Comerç Electrònic

14. Conclusions

Si ens fixem en els objectius inicials d'aquest projecte, podrem comprovar com **s'han assolit tots els objectius**:

- Dissenyar una estructura per a la base de dades que permeti guardar la informació necessària: zones d'arribada, espais, punts de destinació, instruccions,...

Tota la informació necessària es guarda a la base de dades, excepte les imatges de les instruccions que es guarden a disc, tot i que es guarda el nom de la imatge a la base de dades per saber quina imatge de les disponibles retornar. (Apartat 11.1)

- El sistema ha de ser multiidioma, per tant hi ha d'haver les mateixes instruccions traduïdes.

Tota l'aplicació és multiidioma. Les instruccions es poden traduir en els idiomes que faci falta. Només s'ha d'afegir el nou idioma a la base de dades i introduir les noves traduccions per a cada instrucció. (Apartat 11.1.3)

A part d'això, la pàgina web també és totalment multiidioma gràcies a guardar els textos als diferents fitxers. (Apartat 12.2.1)

- Dissenyar un sistema de classes per a treballar amb tota la informació de forma àgil i fàcilment escalable.

Amb el disseny creat s'han afegit tots els mètodes necessaris per a poder gestionar la informació de la base de dades sense haver de saber exactament com està estructurada.

Per afegir una nova entitat només s'ha d'afegir al fitxer d'inicialització. Hi ha les classes `DB_Object` i `Basic_Info` per a facilitar la feina de futur desenvolupament. (Apartats 12.1.6 i 12.1.7)

Si es necessita ampliar una classe, només s'ha de modificar el `classnames` per a canviar la instància que es genera arreu del codi. (Apartat 12.1.4)

- Implementar un algorisme de navegació dins un graf dirigit.

S'ha implementat l'algorisme de Dijkstra per a fer cerques dins un graf dirigit. Aquesta implementació només està dins un mètode i per tant si en un futur és necessari canviar-lo o actualitzar-lo, només s'haurà de modificar aquest mètode. (Apartat 12.1.8, Annexos 0 i 17.4)

- Les instruccions i arestes s'han de poder reutilitzar en diferents camins.

Gràcies a la definició individual d'instruccions, aquestes es poden reutilitzar en més d'un encreuament d'arestes.

- Dissenyar la pàgina web per a que l'usuari final pugui accedir a aquesta informació.

L'usuari té a la seva disposició la pàgina web https://epsapps.udg.edu/epsmap per a utilitzar l'aplicació. (Apartat 11.2)

- L'aplicació final ha de ser el més senzilla possible, l'únic requeriment que necessita l'usuari és un navegador web.

L'usuari només necessita un navegador web que pugui executar JavaScript, i opcionalment, un lector de codis QR per a llegir el punt de captura inicial si és que vol iniciar el recorregut des d'algun altre punt que no sigui l'entrada de P-II.

Aquest projecte es pot comparar amb tres tecnologies existents actualment:

- **Google Maps** i similars com Apple Maps, Bing Maps, etc.
- **Path Guide**
- **Posicionament Wi-Fi**/Bluetooth

Les primeres son les més conegudes i les que faria servir gairebé tothom si se li demanés com arribar a un lloc, però tenen el principal inconvenient que al recolzar-se amb la tecnologia GPS, si la cobertura dins d'un edifici és dolenta, que ho sol esser, llavors l'aplicació no serveix perquè no té la precisió suficient per a determinar on està l'usuari.

La segona opció és un projecte de Microsoft que va ser publicat per primer cop al 2017, que s'assembla molt més a l'enfoc d'aquest projecte. Ja té en compte que la cobertura i precisió dins l'edifici son dolentes, per tant, ho enfoca amb un **model "líder/seguidor"**.

Per a arribar a qualsevol lloc sempre hi ha hagut d'haver algun líder que hagi guardat el recorregut a fer des del punt inicial fins al final, i llavors els seguidors poden reproduir aquest camí guardat. En aquest cas s'haurien de fer tots els possibles camins al menys un cop per tal de guardar el recorregut. [17] [18]

Al final, aquesta opció acaba utilitzant molts sensors del mòbil, sobretot a l'hora de guardar el trajecte, però també quan s'està seguint. Tot i que en general tots els dispositius mòbils solen tenir els mateixos sensors, cada fabricant és un món i cada dispositiu mòbil un altre. Per exemple, al descarregar l'aplicació per a provar-la, la primera informació que m'ha sortit és que al no tenir un baròmetre com a sensor, els canvis d'altura no es podran registrar.

Encara que utilitzi el camp geomagnètic en comptes del GPS, segurament no té prou precisió per a diferenciar dos espais diferents amb 5 metres de diferència. De fet, aquesta és la mateixa raó per la que l'aplicació mai porta l'usuari a l'espai final, sinó a una zona de destinació més àmplia.

En l'últim cas s'utilitzen els sensors Wi-Fi/Bluetooth dels dispositius mòbils per a saber les potències de cada senyal i determinar la posició de l'usuari. Però aquestes senyals son fàcilment afectades per interferències d'altres Wi-Fi, ràdios o parets. [19]

Per tant, l'enfoc que més s'assembla a aquest projecte és el Path Guide. A la Figura 42 he fet una taula per a veure millor els pros i contres de cada aplicació.

EPSMap vs. Path Guide		
Funcionalitat	EPSMap	Path Guide
Enregistrament de rutes	L'administrador de l'aplicació ha d'entrar manualment els nodes, les arestes i les traduccions de les instruccions. Un cop entrat qualsevol d'aquests paràmetres es pot reutilitzar tants cops com faci falta.	El líder ha d'enregistrar totes les possibles rutes un cop anant d'inici a fi. No es poden reaprofitar les rutes ja que sempre son d'un punt A a un punt B.
Detecció de l'usuari	No hi ha detecció de l'usuari durant el recorregut. Però si l'usuari s'ha perdut, pot anar escanejant els punts de captura que es trobi pel camí per tal de tornar a començar.	Detecta la posició de l'usuari utilitzant els sensors del mòbil, principalment el sensor per a mesurar el camp magnètic
Manteniment de camins	No fa falta mantenir els camins com a tal. Sempre i quan el graf sigui vàlid, l'aplicació trobarà el camí més curt. Si	Cada trajecte és un camí diferent.

	es vol afegir un camí més curt, s'ha d'afegir una nova aresta entre dos nodes.	
Espais finals	Hi ha tota una llista d'espais finals d'on seleccionar-ne un. Es pot modificar el node al que està assignat un espai fàcilment.	No hi ha concepte d'espai final. Només hi ha el concepte de trajecte. Per a buscar com arribar a un espai concret s'hauria de buscar el trajecte. Si es vol modificar on està l'espai s'ha de refer tot el trajecte.
Disseny de les característiques	Nosaltres sempre podem modificar les característiques necessàries de l'aplicació. Afegint detalls i explicacions concretes on sigui necessari.	S'està limitat per el per la tecnologia que s'utilitzi, ja sigui GPS, sensor geomagnètic, baròmetre, etc i pel disseny de l'aplicació que altres han fet.

Figura 42 - Comparació de l'aplicació del projecte EPSMap amb l'aplicació Path Guide de Microsoft

Com es pot observar, cadascun té els seus avantatges i inconvenients. **La principal fortalesa del Path Guide és que pot detectar la posició de l'usuari mentre està fent el camí** i actualitzar la instrucció mostrada en vers aquesta informació. En canvi, EPSMap només pot mostrar tot el llistat d'instruccions des de l'inici fins al final.

Però en quan a la facilitat de gestionar les dades guardades, **EPSMap manté separats els trajectes dels punts inicials i finals**, i per a modificar els camins només s'ha de modificar el graf. Per exemple, afegint una connexió entre dos nodes, automàticament es poden actualitzar molts camins ja que sempre es busca el camí més curt.

Aquests i altres aspectes també es tracten al següent apartat com a aspectes a millorar pel treball futur.

15. Treball futur

En tota aplicació sempre hi ha aspectes a millorar i noves funcionalitats a implementar, en aquest apartat citaré les més importants:

- Com s'ha comentat en l'apartat anterior, un dels punts més febles de l'aplicació és la **localització de l'usuari**. Per tant s'hauria d'implementar una forma de localització a través d'algun dels sensors dels dispositius mòbils, a través de les senyals Wi-Fi internes que hi ha a la Universitat o qualsevol altre mètode que pugui aparèixer més endavant.
- Actualment per interactuar amb el graf s'ha de fer directament accedint a la base de dades. No és possible **visualitzar el graf intern de l'aplicació sobre un mapa**.
  Això facilitaria molt la feina a l'hora de crear nous nodes i arestes. Mitjançant les coordenades guardades de cada node, s'haurien de dibuixar els diferents nodes, amb les seves corresponents arestes, sobre un mapa com per exemple Google Maps o HERE Maps.
  Afegint les opcions de poder modificar el graf, i que les modificacions que es facin es guardin a la base de dades.
- Molts cops la EPS celebra **esdeveniments temporals**, i els assistents no saben exactament on han d'anar i per tant, tampoc saben l'espai final al que han d'anar. Es podria afegir una taula d'àlies o esdeveniments que durant un període estan disponibles al cercador, i simplement "apunten" a l'espai final que s'ha assignat per a aquell esdeveniment.
- Al tenir els espais dividits per categories, es podria **afegir un altre mètode a la classe `Graph` per a que busqui el node més proper d'un cert tipus** utilitzant BFS o similar. P. ex. el lavabo més proper, la cafeteria més propera,...
- Actualment quan s'afegeix una imatge a una instrucció només es guarda aquella imatge, però quan s'està mostrant en petit no és necessari retornar la imatge amb la resolució completa. Es podria afegir un sistema que **al pujar una imatge, generi un parell de versions d'aquesta de menys qualitat i més petites**. Per tant, quan es mostrés la imatge a la llista d'instruccions es retornaria menys informació, i només es retornaria la imatge amb la màxima qualitat quan sigui necessari, p. ex. quan s'obra en pantalla completa.

16. Bilbiografia

[1] D. Radigan, «Kanban - A brief introduction,» [En línia]. Available: https://www.atlassian.com/agile/kanban. [Últim accés: 03 08 2022].

[2] TechTarget Contributor, «Web application (Web app),» [En línia]. Available: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app. [Últim accés: 08 07 2022].

[3] Wikipedia, «Teoria de grafs - Problemes de teoria de grafs,» [En línia]. Available: https://ca.wikipedia.org/wiki/Teoria_de_grafs#Problemes_de_teoria_de_grafs. [Últim accés: 04 07 2022].

[4] Departament d'Informàtica i Matemàtica Aplicada, «Introducció a la teoria de grafs,» 2018.

[5] Hybesis, «Pathfinding algorithms : the four Pillars,» 20 01 2020. [En línia]. Available: https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b. [Últim accés: 04 07 2022].

[6] A. Karunaratne, «PHP Versions,» 2022. [En línia]. Available: https://php.watch/versions. [Últim accés: 30 06 2022].

[7] S. Ravoof, «The Definitive PHP 7.2, 7.3, 7.4, 8.0, and 8.1 Benchmarks (2022),» Kinsta, 30 03 2022. [En línia]. Available: https://kinsta.com/blog/php-benchmarks/. [Últim accés: 30 06 2022].

[8] S. M. Goy, «EPSMap – Aplicació web de navegació interna pels edificis de la EPS,» [En línia]. Available: https://github.com/sergiMagret/EPSMap.

[9] Stack Overflow, «Stack Overflow Developer Survey 2022,» 05 2022. [En línia]. Available: https://survey.stackoverflow.co/2022/#section-most-popular-technologies-web-frameworks-and-technologies. [Últim accés: 11 07 2022].

[10] Universitat de Girona, «Identitat corporativa. Tipografia,» [En línia]. Available: https://www.udg.edu/ca/coneix/identitat-corporativa/marca-udg/tipografia. [Últim accés: 24 03 2022].

[11] Universitat de Girona, «Identitat corporativa. Colors,» [En línia]. Available: https://www.udg.edu/ca/coneix/identitat-corporativa/marca-udg/colors. [Últim accés: 24 03 2022].

[12] DrawSQL, «DrawSQL - Database schema diagrams,» [En línia]. Available: https://drawsql.app/home.

[13] Seldaek, «Handlers, Formatters and Processors,» 24 07 2022. [En línia]. Available: https://github.com/Seldaek/monolog/blob/main/doc/02-handlers-formatters-processors.md#processors. [Últim accés: 31 07 2022].

[14] The PHP Group, «Static Keyword,» [En línia]. Available: https://www.php.net/manual/en/language.oop5.static.php. [Últim accés: 27 07 2022].

[15] The PHP Group, «Late Static Bindings,» [En línia]. Available:

https://www.php.net/manual/en/language.oop5.late-static-bindings.php. [Últim accés: 28 07 2022].

[16] MDN Contributors, «MIME types (IANA media types),» 14 07 2022. [En línia]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types. [Últim accés: 30 07 2022].

[17] Y. Shu, B. F. Karksson, Y. Lin i T. Moscibroda, «Path Guide: Plug-and-play Indoor Navigation,» Microsoft, [En línia]. Available: https://www.microsoft.com/en-us/research/project/path-guide-plug-play-indoor-navigation/. [Últim accés: 09 08 2022].

[18] Y. Shu i B. F. Karlsson, «Path Guide: A New Approach to Indoor Navigation,» Microsoft, 14 07 2017. [En línia]. Available: https://www.microsoft.com/en-us/research/blog/path-guide-new-approach-indoor-navigation/. [Últim accés: 09 08 2022].

[19] InfSoft, «Indoor Positioning, Tracking and Indoor Navigation with Wi-Fi,» [En línia]. Available: https://www.infsoft.com/basics/positioning-technologies/wi-fi/. [Últim accés: 09 08 2022].

[20] R. Ranjan, «What is a Framework in Programming & Why You Should Use One,» 07 10 2021. [En línia]. Available: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/. [Últim accés: 08 07 2022].

[21] Composer, «Composer,» [En línia]. Available: https://getcomposer.org/. [Últim accés: 26 07 2022].

[22] The PHP Group, «Description of core php.ini directives,» [En línia]. Available: https://www.php.net/manual/en/ini.core.php#ini.include-path. [Últim accés: 26 07 2022].

[23] MDN contributors, «Event reference,» 17 05 2022. [En línia]. Available: https://developer.mozilla.org/en-US/docs/Web/Events. [Últim accés: 07 30 2022].

[24] Universitat de Girona, «Servei d'Oficina Tècnica i Manteniment,» [En línia]. Available: https://www.udg.edu/ca/estructura/serveis/servei?ID=93. [Últim accés: 10 08 2022].

[25] GitHub, «GNU General Public License v3.0,» 3 10 2020. [En línia]. Available: https://choosealicense.com/licenses/gpl-3.0/. [Últim accés: 10 08 2022].

[26] Departament d'Informàtica i Matemàtica Aplicada, «Grafs: Recorreguts i camins mínims,» 2018.

[27] MDN Contributors, «Responsive web design,» 22 04 2022. [En línia]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Responsive_web_design. [Últim accés: 11 07 2022].

17. Annexos

17.1. Recull de les coordenades dels nodes i pesos de les arestes

Per a saber coordenades geogràfiques i les distàncies entre aquests punts he utilitzat Google Maps.

Les coordenades d'un punt concret es poden saber prement el botó dret just del punt que volem recollir les dades, al menú contextual que es veu a la Figura 43 es mostren les coordenades de l'entrada principal de P2.

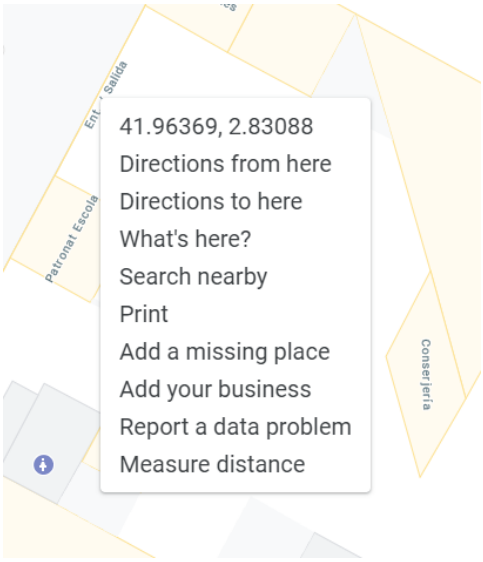
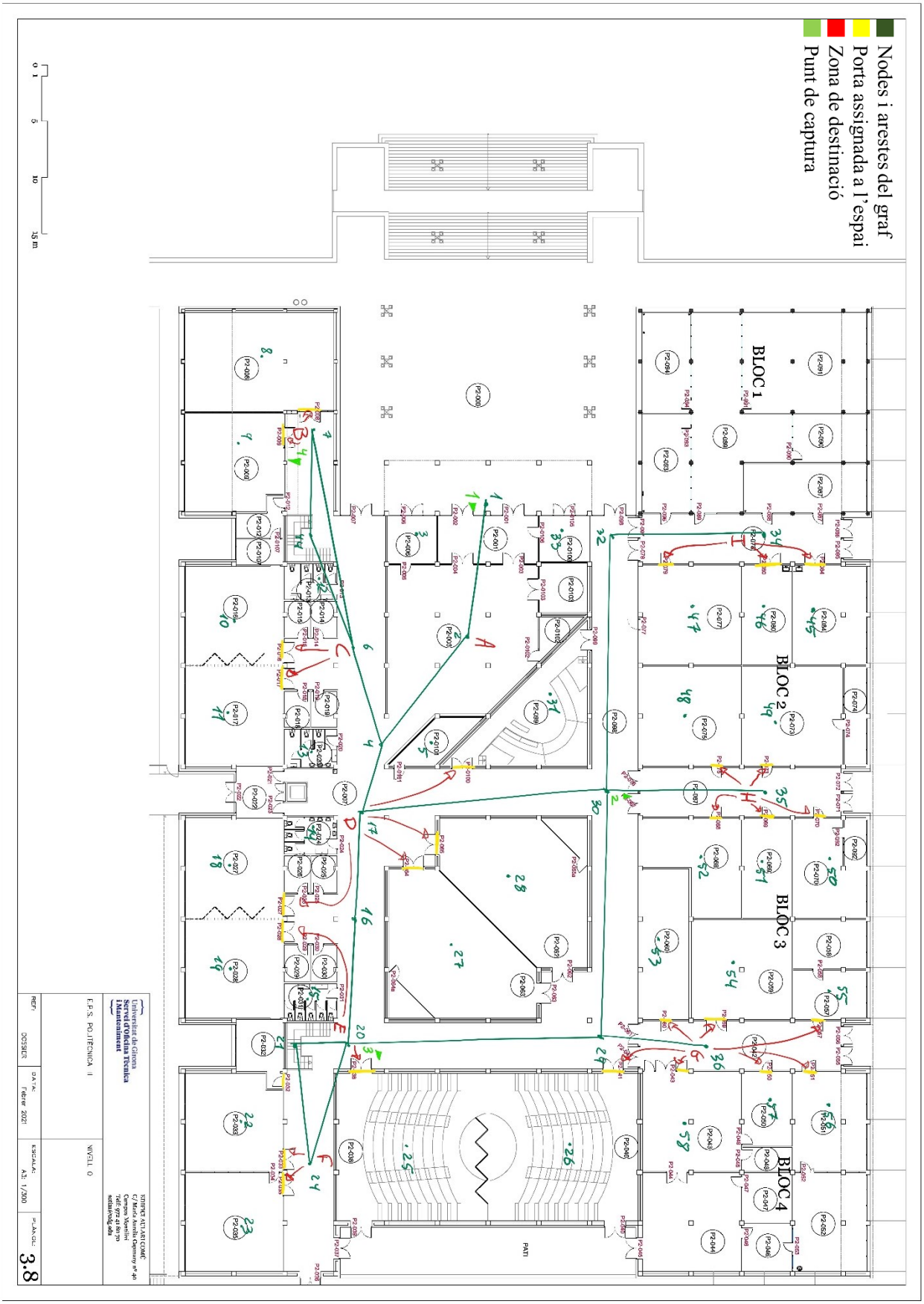


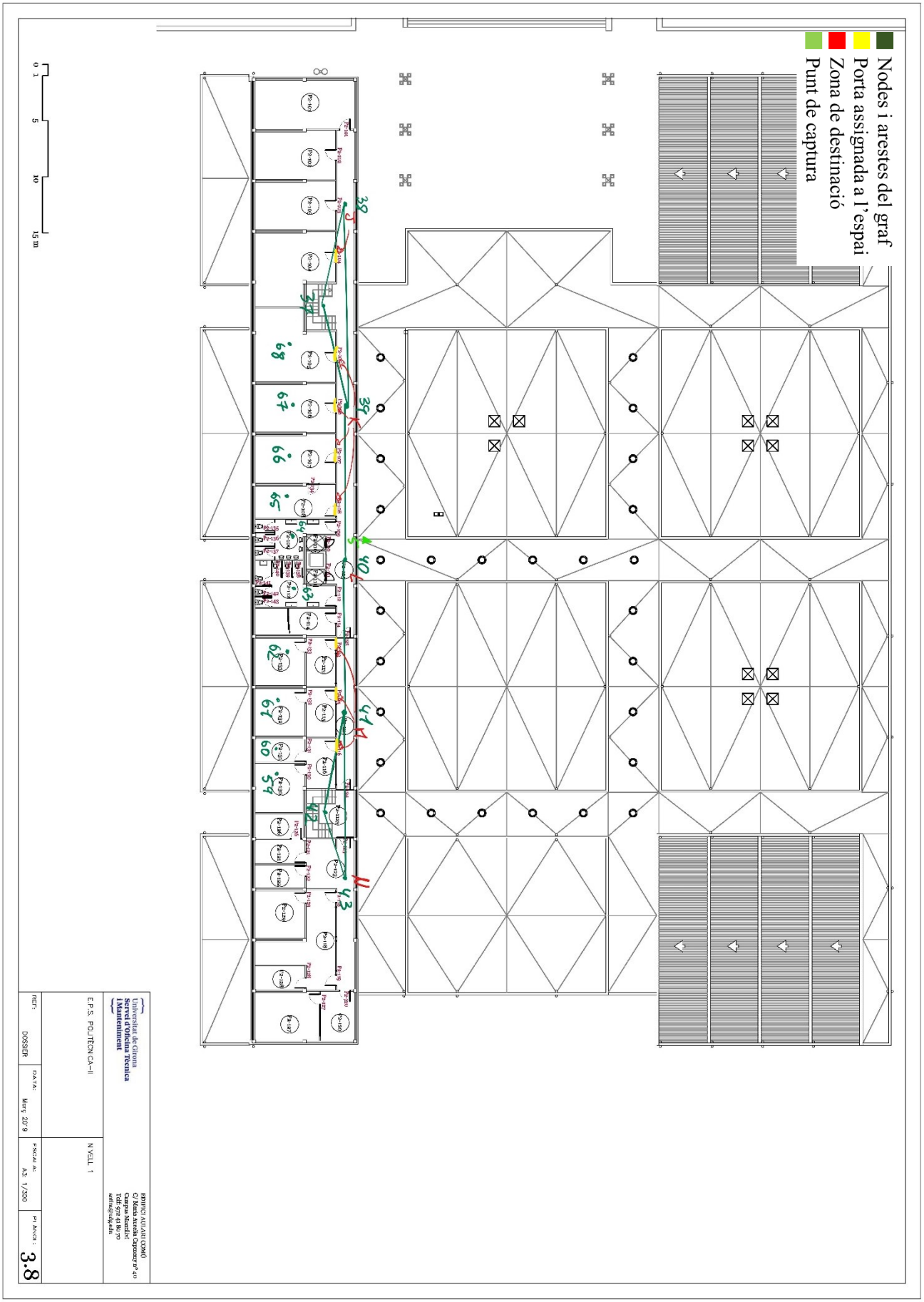
Figura 43 - Menú contextual de Google Maps

En el cas de les distàncies entre dos punts, es fa de forma similar. Del mateix menú contextual que es mostra a la Figura 43, s'ha de seleccionar l'última opció "Measure distance" (el text pot variar segons l'idioma). Un cop seleccionada l'opció, demanarà que triem un altre punt del mapa. Just quan es triï el segon punt, apareixerà un regle amb la distància entre els dos punts seleccionats. A la Figura 44 es mostra la distància entre l'entrada de P2 i consergeria.

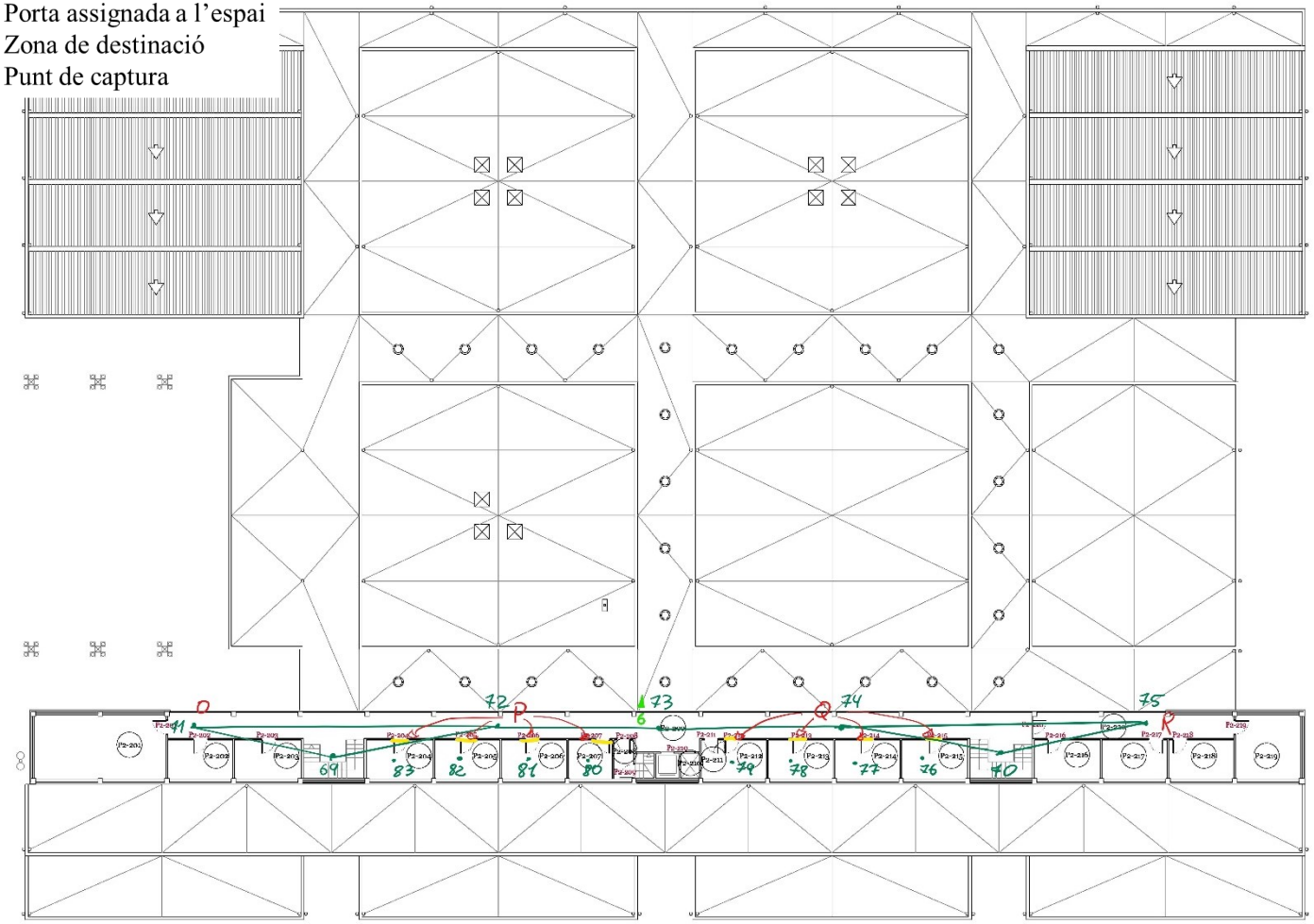
Figura 44 - Distància entre l'entrada de P2 i consergeria mesurada amb Google Maps

17.2. Mapes edifici Politècnica II




Nodes i arestes del graf
Porta assignada a l'espai
Zona de destinació
Punt de captura
0 1 5 10 15 m
E.P.S. POLITÈCNICA-II
NIVELL 1
DOSSIER
Març 2019
1/320
3.8

- Nodes i arestes del graf
- Porta assignada a l'espai
- Zona de destinació
- Punt de captura



Nº Espai	Ús	Sup.Util m2
P2-200	Circulació	117,00 m2
P2-201	Despatx professor	35,85 m2
P2-202	Administració	11,45 m2
P2-203	Administració	11,70 m2
P2-204	Despatx professor	11,70 m2
P2-205	Despatx professor	11,45 m2
P2-206	Despatx professor	11,45 m2
P2-207	Despatx professor	7,80 m2
P2-208	Concessionari	5,65 m2
P2-210	Local tècnic	2,10 m2
P2-211	Despatx professor	3,40 m2
P2-212	Despatx professor	7,80 m2
P2-213	Despatx professor	11,45 m2
P2-214	Despatx professor	11,45 m2
P2-215	Despatx professor	11,70 m2
P2-216	Despatx professor	11,70 m2
P2-217	Despatx professor	11,45 m2
P2-218	Despatx professor	11,70 m2
P2-219	Despatx professor	19,15 m2
P2-220	Circulació	21,25 m2

Universitat de Girona
Servei d'Oficina Tècnica i Manteniment

EDIFICI AULARI COMÚ
C/ Maria Aurèlia Capmany nº 49
Campus Montilivi
Telf: 972 41 80 70
sotim@udg.edu

E.P.S. POLITÈCNICA–II

NIVELL 2

REF:	DATA:	ESCALA:	PLÀNOL:
DOSSIER	juny 2017	A3: 1/300	3.8

17.3. Pseudocodi de Dijkstra

```
Dijkstra(G, s, t):
for i = 1 to n do
   if i≠ s then
      dist[i]:=∞;
   else
      dist[i]:=0;
   end if
   pred[i]:=0;
   def[i]:=0;
end for
vertact:=s;
while def[t]=0 and vertact≠ 0 do
   mindist:=∞, vertact:=0;
   for i = 1 to n do
      if def[i]=0 and dist[i]<mindist then
         mindist:=dist[i];
         vertact:=i;
      end if
   end for
   if vertact≠ 0 then
      def[vertact]:=1;
      for i = 1 to gr(vertact) do
         vertadj:=llistadj(vertact,i);
         if def[vertadj]=0 and dist[vertadj]>dist[vertact]+cost(vertact,i) then
            dist[vertadj]:=dist[vertact]+cost(vertact,i);
            pred[vertadj]:=vertact;
         end if
      end for
   end if
end while
```

on:
s és el vèrtex de sortida.
t és el vèrtex destí.
n és el nombre de vèrtexs del graf.
pred[*i*], *i*=1,...,*n*. Vector de sortida que emmagatzema el predecessor de cada vèrtex *i*.
dist[*i*], *i*=1,...,*n*. Vector de sortida que emmagatzema la distància entre vèrtex *s* i el vèrtex *i*.
def[*i*], *i*=1,...,*n*. Vector que emmagatzema si el vèrtex *i* ja té la distància definitiva.
gr(*i*), *i*=1,...,*n*. Funció que retorna el grau del vèrtex *i*.
llistadj(*i*,*j*), *i*=1,...,*n*; *j*=1,...,gr(*i*). Funció que retorna el vèrtex adjacent número *j* del vèrtex *i*.
cost(*i*,*j*), *i*=1,...,*n*; *j*=1,...,gr(*i*). Funció que retorna el cost entre el vèrtex número *i* i el vèrtex número *j*.

Figura 45 - Pseudocodi per a la implementació de Dijkstra [26]

17.4. Implementació classe Graph

```php
<?php
/**
 * Class to store a Graph and perform operations over it.
 */
class Graph {

    /** List of Nodes indexed by their ID
     * @var Node[] */
    protected array $_nodes;

    /** List of Edges indexed by their ID
     * @var Edge[] */
    protected array $_edges;

    /** List of Edges indexed by [node_id_start][node_id_end]
     * @var Edge[][] */
    public array $_graph_as_array;

    /**
     * Create a Graph to operate over, and find paths on it
     *
     * @param Node[] $nodes List of Nodes for the Graph
     * @param Edge[] $edges List of Edges for the Graph
     */
    public function __construct(array $nodes, array $edges) {
        $this->_nodes = [];
        foreach($nodes as $n) $this->_nodes[$n->getID()] = $n;

        $this->_edges = [];
        foreach($edges as $e) $this->_edges[$e->getID()] = $e;

        $this->_graph_as_array = [];
        foreach($this->_edges as $e)
            $this->_graph_as_array[$e->getEdgeStart(true)][$e->getEdgeEnd(true)] = $e;
    }

    /**
     * Get all the edges from the graph
     * @return Edge[]
     */
    public function getEdges(): array {
        return $this->_edges;
    }

    /**
```

```
     * Get all the nodes from the graph
     * @return Node[]
     */
    public function getNodes(): array {
        return $this->_nodes;
    }

    /**
     * Get the adjecent nodes for a given node
     *
     * @param Node $node
     * @return Node[]
     */
    public function getAdjecentNodes(Node $node): array {
        $nodes = [];

        // The adjecent nodes are the ones with edges from $node to other nodes
        // AND from orther nodes to $node, but with a bidirectional edge
        foreach($this->_nodes as $n){
            if(isset($this->_graph_as_array[$n->getID()][$node->getID()])
                && $this->_graph_as_array[$n->getID()][$node->getID()]->isBidirectional()){
                // From other nodes to $node
                $nodes[] = $this->_graph_as_array[$n->getID()][$node->getID()]->getEdgeStart();
            }else if(isset($this->_graph_as_array[$node->getID()][$n->getID()])){
                // From $node to other nodes
                $nodes[] = $this->_graph_as_array[$node->getID()][$n->getID()]->getEdgeEnd();
            }
        }

        return $nodes;
    }

    /**
     * Get the Edge between $start and $finish.
     * The edge does not necessarily go from $start to $finish,
     * this method gets the edge (if exists) between $start and $finish
     * but if the edge is bidirectional, then the returned Edge might go from
     * $finish to $start
     *
     * @param Node $start
     * @param Node $finish
     * @return Edge|null The Edge or null if there is no Edge between $start and $finish
     */
    public function getEdgeBetween(Node $start, Node $finish): ?Edge {
        // First check if there is an edge directly from $start to $finish
        if(isset($this->_graph_as_array[$start->getID()][$finish->getID()]))
            return $this->_graph_as_array[$start->getID()][$finish->getID()];
```

```
        // Then check if there is an edge from $finish to $start and that edge
        // is bidirectional
        if(isset($this->_graph_as_array[$finish->getID()][$start->getID()])
            && $this->_graph_as_array[$finish->getID()][$start->getID()]->isBidirectional())
            return $this->_graph_as_array[$finish->getID()][$start->getID()];

        // If neither of the above is true, then return null since there is no edge
        // from $start to $finish
        return null;
    }

    /**
     * Find the shortest path between $from and $to Nodes using Dijkstra.
     *
     * The returned result is a list of the Edges the user has to go by.
     * The Edges are considered bidirectional.
     *
     * You should call this method with:
     *  list($edges_path, $cost) = Graph->findShortestPath(Node, Node)
     *
     * @param Node $from
     * @param Node $to
     * @return array|null The returned array contains the list of Edges the user has
     * to go by in the first position and the total cost in the second position
     */
    public function findShortestPath(Node $from, Node $to): ?array {
        $dist = []; // Stores the distances between the node $from and the node <i>
        $prev = []; // $prev[i] sotres the previous node to <i>
        $def = [];  // Stores whether the node <i> has the definitive distance in $dist[i]
        $nodes = $this->getNodes();

        // Initialize all the arrays
        foreach($nodes as $n){
            $dist[$n->getID()] = INF;
            $prev[$n->getID()] = null;
            $def[$n->getID()] = false;
        }

        // Initialize the distance for the initial node
        $dist[$from->getID()] = 0;

        // Current node
        $vert_act = $from;

        while($def[$to->getID()] == false && $vert_act != null){
            $min_dist = INF; $vert_act = null;
            foreach($nodes as $n){ // Search the node with the min distance
                if($def[$n->getID()] == false && $dist[$n->getID()] < $min_dist){
```

```
                    $min_dist = $dist[$n->getID()];
                    $vert_act = $n;
                }
            }

            if($vert_act != null){ // If it's null we have finnished
                $def[$vert_act->getID()] = true;
                foreach($this->getAdjecentNodes($vert_act) as $adj){
                    $tmp_weight = $dist[$vert_act->getID()] +
                                  $this->getEdgeBetween($vert_act, $adj)->getWeight();
                    if($def[$adj->getID()] == false && $dist[$adj->getID()] > $tmp_weight){
                        $dist[$adj->getID()] = $tmp_weight; // Distance to the adjecent node
                        $prev[$adj->getID()] = $vert_act; // Set the previous node
                    }
                }
            }
        }

        if($def[$to->getID()] == null) return null; // No path was found

        // The path is created from the finnish to the start i.e. $to -> $from
        $path = [$to];
        $vert_act = $to;
        while($vert_act->getID() != $from->getID()){
            $vert_act = $prev[$vert_act->getID()];
            $path[] = $vert_act;
        }

        // You can return the list of nodes here
        // return [array_reverse($path), $dist[$to->getID()]];

        // But we are interested in the edges since the instructions are between the edges
        $edges_path = [];
        $prev_n = $path[0];
        $act_n = null;
        for($i=1; $i < count($path); $i++){
            $act_n = $path[$i];
            $edges_path[] = $this->getEdgeBetween($prev_n, $act_n);
            $prev_n = $act_n;
        }

        return [array_reverse($edges_path), $dist[$to->getID()]];
    }
}
?>
```

17.5. Implementació onSelectDestination

```
function onSelectDestination(destination, path_information) {
    const path = path_information.instructions;
    const total_cost = path_information.total_cost;
    const destination_zone = path_information.destination_zone;
    const initial_turn = path_information.initial_turn;
    const only_edge = path_information.only_edge;
    const already_in_place = path_information.in_place;

    const initialTurnText = {
        'F': lang_obj.initial_turn_forward,
        'B': lang_obj.initial_turn_backward,
        'R': lang_obj.initial_turn_right,
        'L': lang_obj.initial_turn_left,
    };

    let path_instructions = $("#path-instructions");
    path_instructions.removeClass("d-none");
    path_instructions.empty();
    if(initial_turn){
        path_instructions.append($("<div/>", {
                class: "initial-turn-instruction"
            }).append(
                $("<div/>", {
                    class: "instruction-text"
                }).text(lang_obj.initially + " " +
                        initialTurnText[initial_turn].toLowerCase() + " " +
                        lang_obj.and_then + ":"),
                $(`<a tabindex="0" role="button" data-toggle="popover" class="btn-inter-results"
                 style="color: inherit;"><i class="fa fa-question-circle-o" aria-hidden="true"
                 title="${lang_obj.interpret_results_title}"></i></a>`)
            )
        );

        $('.btn-inter-results').popover({
            trigger: 'focus',
            content: lang_obj.interpret_results_content,
            placement: "left",
            html: true
        })
    }

    let append_move_forward_text = true;
    // List of instructions whose following instruction won't
    // include the text to move forward X meters
    const instructions_not_move_forward_text = ["go_1_floor_upstairs", "go_1_floor_downstairs"];

    if(already_in_place){ // The destination is in the same destination zone
```

```
        const already_in_place_text = $("<div/>", {
            class: "final-instruction-text"
        }).text(
            `${lang_obj.already_in_place} (${lang_obj.destination_zone}
"${destination_zone.name}")`
        );

        path_instructions.append(already_in_place_text);
    }else if(!path){ // The destination is right next to the user
        const move_forward_text = $("<div/>", {
            class: "final-instruction-text"
        }).text(
            `${lang_obj.move_forward} ${only_edge.weight} ${lang_obj.meters} ${lang_obj.and}
            ${lang_obj.you_will_have_reached_your_destination.toLowerCase()}
            (${lang_obj.destination_zone} "${destination_zone.name}")`
        );

        path_instructions.append(move_forward_text);
    }else{
        path.forEach(edge => {
            const from_edge = edge.from;
            const to_edge = edge.to;
            const instruction_translation = edge.instruction_translation ||
                    `Missing instruction from ${from_edge.id}
                    (${from_edge.from_node.id}, ${from_edge.to_node.id}) to
                    ${to_edge.id} (${to_edge.from_node.id}, ${to_edge.to_node.id})`;
            const has_image = edge.has_image;
            const instruction_image = has_image ? $("<div/>", {
                class: "instruction-image"
            }).append($("<img/>", {
                src:
"actions/getInstructionImage.php?initial_edge_id="+from_edge.id+"&destination_edge_id="+to_edge.id
            })) : null;

            const full_instruction_text = append_move_forward_text ?
                                          lang_obj.move_forward + " " + from_edge.weight + " " +
                                          lang_obj.meters + ". " + (instruction_translation.text ||
                                          instruction_translation) :
                                          instruction_translation.text
            const instruction_text = $("<div/>", {
                class: "instruction-text"
            }).text(full_instruction_text);

            // Update the variable for the next instruction
            // Since the move forward is done before the instruction we want to not append
            // the move forward in the next instruction
            if((typeof instruction_translation).toLowerCase() == "object")
append_move_forward_text =
```

```
!instructions_not_move_forward_text.includes(instruction_translation.instruction.name);
            else append_move_forward_text = true;

            const instruction_wrapper = (
                $("<div/>", {
                    class: "instruction-wrapper" + (has_image ? " has-image" : "")
                }).append(instruction_image, instruction_text)
            );

            path_instructions.append(instruction_wrapper);
        });

        const move_forward_text = $("<div/>", {
            class: "final-instruction-text"
        }).text(
            `${lang_obj.move_forward} ${path[path.length -1].to.weight} ${lang_obj.meters}
                ${lang_obj.and} ${lang_obj.you_will_have_reached_your_destination.toLowerCase()}
                (${lang_obj.destination_zone} "${destination_zone.name}")`
        );

        path_instructions.append(move_forward_text);
    }
}
```

18. Manual d'usuari i instal·lació

18.1. Instal·lació de l'aplicació

Per a la instal·lació de l'aplicació a un servidor es necessita PHP 7.4+, MariaDB 5.5+, Apache 2.4.6+ i un usuari per a connectar-te a la base de dades.

El primer és descarregar el codi de GitHub, el trobaràs a https://github.com/sergiMagret/EPSMap, recomano fer un `clone` de Git en comptes de descarregar-lo com a ZIP, però això depèn de tu.

Un cop descarregat tot el codi hauràs de crear la base de dades, en el meu cas anomenada `eps_map`. Per a fer això pots utilitzar un dels dos fitxers que trobaràs a la carpeta mysql. Aquí hi ha un fitxer per a crear només l'estructura o per a crear l'estructura i introduir les dades per a l'edifici Politècnica 2, és a dir, amb els resultats d'aquest treball.

Un cop hagis creat la base de dades, has de configurar l'aplicació. Per això fes una còpia del fitxer conf/settings.fill.php i canvia el nom a conf/settings.php. Aquest últim fitxer és el que es s'ha de modificar per a canviar la configuració de l'aplicació. No està vigilat per Git, per tant res del que modifiquis aquí es guardarà al repositori.

En aquest mateix fitxer trobaràs les instruccions i recomanacions a l'hora de configurar l'aplicació. Aquest fitxer és el mateix que el mostrat a la Figura 19.

18.2. Manual d'usuari

Per entrar a l'aplicació ho pots fer a través de https://epsapps.udg.edu/epsmap o escanejant un dels codis QR. Recomano entrar-hi escanejant un dels codis QR ja que així les instruccions es donaran des del punt on hagis capturat el QR. Sinó, es donen des de l'entrada de l'edifici.

Només d'entrar a l'aplicació el primer que se't demanarà serà a quina destinació vols anar, pots filtrar per espai (p. ex. si has d'anar a un laboratori, despatx o aula concreta) o per professor (p. ex. tens una reunió amb aquest professor i no saps quin és el despatx).

Un cop seleccionada una de les dues opcions si prems sobre el desplegable apareixerà la llista de espais o professors disponibles. Pots cercar per nom o àlies. Un cop trobat el pots seleccionar prement un cop sobre l'opció i es tancarà el selector. Si ara prems a "Cercar" et mostrarà totes les instruccions a seguir per arribar fins la zona d'arribada on es troba aquest espai final.

Si al llistat d'instruccions hi veus una imatge, pots prémer sobre ella i s'obrirà en pantalla completa, si tornes a prémer a algun lloc fora de la imatge, es tancarà.

Si necessites canviar l'idioma a dalt a la dreta hi ha un desplegable amb els idiomes disponibles a l'aplicació.

Pots veure un exemple de funcionament a la Figura 41.