Roger Muntané Serra

SISTEMA AUTOMÀTIC DE CREACIÓ DE FITXES D'HABITACIÓ PER LA PÀGINA WEB D'AMIMIR.COM

TREBALL DE FI DE GRAU

dirigit per Marc Sanchez Artigas

Grau d'Enginyeria Informàtica/Telemàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

Resum

El projecte que s'explica a continuació detalla les diferents etapes de disseny i implementació d'un sistema automàtic de creació de fitxes d'habitació a partir de peticions a APIs de proveïdors externs. El motiu d'aquest projecte és per aconseguir un millor nivell de detall i descripció de característiques del producte que s'ofereix a la pàgina web d'Amimir.com, una plataforma de cerca d'hotels online. Aquest sistema recorrerà tots els hotels amb els quals treballa l'empresa de Viajes Para Ti, i intentarà generar una fitxa d'habitació on es mostren les diferents fotografies i es descriu els diferents serveis. Per implementar aquest sistema s'han utilitzat diferents tecnologies del servei núvol d'Amazon i tecnologies de desenvolupament web com són el PHP per al back-end i una combinació de JavaScript, HTML i CSS per al front-end. Gràcies a aquest projecte s'ha aconseguit oferir un millor servei als usuaris de la pàgina web i reduir temps i costos als treballadors de l'empresa que s'encarreguen de crear aquestes fitxes manualment.

Resumen

El proyecto que se explica a continuación detalla las diferentes etapas de diseño e implementación de un sistema automático de creación de fichas de habitación a partir de peticiones a APIs de proveedores externos. El motivo de este proyecto es conseguir un mejor nivel de detalle y descripción de las características del producto que se ofrece en la página web de Amimir.com, una plataforma de búsqueda de hoteles online. Este sistema recorrerá todos los hoteles con los cuales trabaja la empresa Viajes Para Ti, e intentará generar una ficha de habitación donde se muestren las diferentes fotografías y se describan los distintos servicios. Para implementar este sistema se han utilizado diferentes tecnologías del servicio en la nube de Amazon y tecnologías de desarrollo web como PHP para el back-end y una combinación de JavaScript, HTML y CSS para el front-end. Gracias a este proyecto se ha conseguido ofrecer un mejor servicio a los usuarios de la página web y reducir tiempo y costos a los trabajadores de la empresa que se encargan de crear estas fichas manualmente.

Abstract

The project described below details the different stages of design and implementation of an automatic room sheet creation system based on requests to external providers' APIs. The purpose of this project is to achieve a better level of detail and description of the product features offered on the Amimir.com website, an online hotel search platform. This system will go through all the hotels that the company Viajes Para Ti works with and will try to generate a room sheet showing the different photographs and describing the various services. To implement this system, different Amazon cloud service technologies and web development technologies have been used, such as PHP for the back-end and a combination of JavaScript, HTML, and CSS for the front-end. Thanks to this service, it has been possible to offer better service to the website users and reduce time and costs for the company's employees who are responsible for creating these sheets manually.

Agraïments

Vull expressar el meu més sincer agraïment a totes les persones que han fet possible la realització d'aquest treball de final de grau.

En primer lloc, vull agrair a la meva família pel seu suport incondicional, la seva paciència i la seva confiança en mi durant tot el període de la carrera.

També vull donar les gràcies als meus amics i la meva parella, per estar sempre al meu costat, per animar-me en els moments difícils i per celebrar amb mi els èxits.

A l'empresa on he realitzat les pràctiques, Viajes Para Ti, estic profundament agraït per l'oportunitat que m'han brindat de posar en pràctica els meus coneixements i aprendre de professionals excel·lents. En concret, al meu tutor Marc Farré i tot l'equip de producte, la vostra guia i suport han estat imprescindibles per al desenvolupament d'aquest projecte.

Finalment, vull agrair a la universitat i al meu tutor Marc Sanchez per ajudar a estructurar correctament aquest projecte i aclarir tots els dubtes que han anat sorgint.

A tots vosaltres, moltes gràcies.

Índex

Índex il·lustracions

1 Introducció

En aquest escrit es mostrarà una explicació i documentació del treball de fi de grau de l'alumne de quart d'enginyeria informàtica Roger Muntané.

Bé, doncs, per començar m'agradaria parlar de mi. Com bé he dit abans, soc el Roger, tinc 25 anys i he cursat els estudis d'enginyeria informàtica a la URV de Tarragona.

Com a estudis anteriors al grau, vaig cursar el batxillerat tecnològic i un Cicle formatiu de grau superior de Sistemes informàtics i de telecomunicació a l'institut Pere Martell.

Amb l'entrega d'aquest TFG faig tancament de la meva etapa estudiantil a la universitat per a continuar desenvolupant la meva carrera professional al món laboral.

El meu TFG l'he desenvolupat a l'empresa Viajes Para Ti (VPT), una agència de viatges especialitzada en la venda Online propietària de la marca "Esquiades.com", líder de vendes de packs d'esquí i esports d'aventura a Espanya. També és propietària de les següents marques:

- **BuscoUnChollo** : Web pionera en la venda flash de paquets turístics.

- **Amimir**: Buscador d'hotels a escala mundial.

- **Jump2Spain**: Destinat al turista internacional per oferir les millors ofertes per visitar Espanya

Il·lustració 1 Logotip d Viajes Para Ti

A part de ser una empresa líder al sector turístic a Espanya, també destaca en l'àmbit tecnològic, aquesta forma part del Clúster Tic Catalunya Sud (TIC).

El TIC és una agrupació de les principals empreses tecnològiques i institucions del territori amb l'objectiu de potenciar el creixement i la competitivitat del sector TIC de la província de Tarragona.

A continuació una breu explicació de la història de l'empresa.

L'empresa VPT va néixer el 2002 amb la pàgina **Esquiades**, fundada per Rafa Fuertes i com a únic treballador en aquell moment, estava orientada a oferir viatges a la neu, en concret a Port Ainé.

Il·lustració 2 Rafa Fuertes CEO de l'empresa

De mica en mica les opcions de viatge van anar augmentant i l'empresa va anar creixent fins que al 2010 va néixer **BuscoUnChollo**, amb l'objectiu d'oferir "chollos" (ofertes), que durin poc temps però que siguin les millors del mercat.

Al 2016 es crea la nova marca **Amimir**, el portal per buscar i fer reserves arreu del món. En aquest cas es troba que Amimir te molts competidors molt forts com és el cas de Booking.com, Trivago.es ... Tot i això, la pàgina també acaba sent un èxit.

Per últim, al 2022 neix **Jum2Spain** amb l'objectiu d'explotar el producte dins del país i oferir les millors ofertes *flash* al públic internacional.

L'empresa any rere any ha incrementat els seus èxits i beneficis, així com els seus treballadors i oficines.

Actualment, l'empresa compta amb més de 130 treballadors. La mitja d'edat d'aquests és de 27 anys, una mitja bastant baixa tenint en compte la quantitat de treballadors. Això ve donat que l'empresa dona moltes oportunitats als joves per començar les seves primeres experiències al món laboral a la vegada que aporten valor a l'empresa.

L'empresa està ubicada al TecnoParc de Reus, un viver d'empreses on hi ha oficines para a diferents empreses del sector. El TecnoParc és propietat de l'empresa local de Reus anomenada Redessa.

Il·lustració 3 Parc tecnològic de Redessa

Dins del TecnoParc, VPT disposa de diverses oficines bastant àmplies, així com sales externes per fer reunions, gravar espots publicitaris i d'oci. A principis de 2024 ha ampliat a més oficines per tal de separar tots els treballadors d'IT amb la resta de l'empresa.

Quant a dades oficials tenim que l'any 2022 va facturar més de 125.000.000€ i que va tenir més de 265.000 reserves entre les seves pàgines web.

Una vegada fet una petita explicació de l'empresa, és rellevant destacar que prèviament de fer el conveni i l'estança de TFG a VPT, jo ja havia estat fent les pràctiques del grau allí.

Gràcies a haver estat aquests mesos de pràctiques previs al TFG, ja coneixia la infraestructura tecnològica, així com el funcionament de l'empresa, cosa que va agilitzar el meu treball.

2 Descripció general del projecte

2.1 Context

Com hem pogut veure, VPT, és una empresa plenament enfocada al turisme i el seu negoci és totalment Online.

Qualsevol pàgina web que ofereix algun producte, si vol que aquest tingui èxit, haurà d'elaborar una bona descripció del producte així com disposar de fotografies o vídeos que ajudin a la descripció.

Partint d'aquesta premissa i transportada al món del turisme, si volem que els hotels que s'ofereixen a la pàgina web tinguin un nombre elevat de reserves, s'haurà d'invertir un esforç a detallar bé totes les característiques d'aquest així com detallar una descripció de l'habitació, ja que així l'usuari pot veure exactament que està reservant i que es trobarà.

Per a Amimir i Esquiades, quan un client vol fer una reserva d'un hotel, pot veure una fitxa detallada de l'hotel (Serveis, fotografies, ubicació, pàrquings, etc.), aquesta informació permet a l'usuari decantar-se cap a un hotel o un altre.

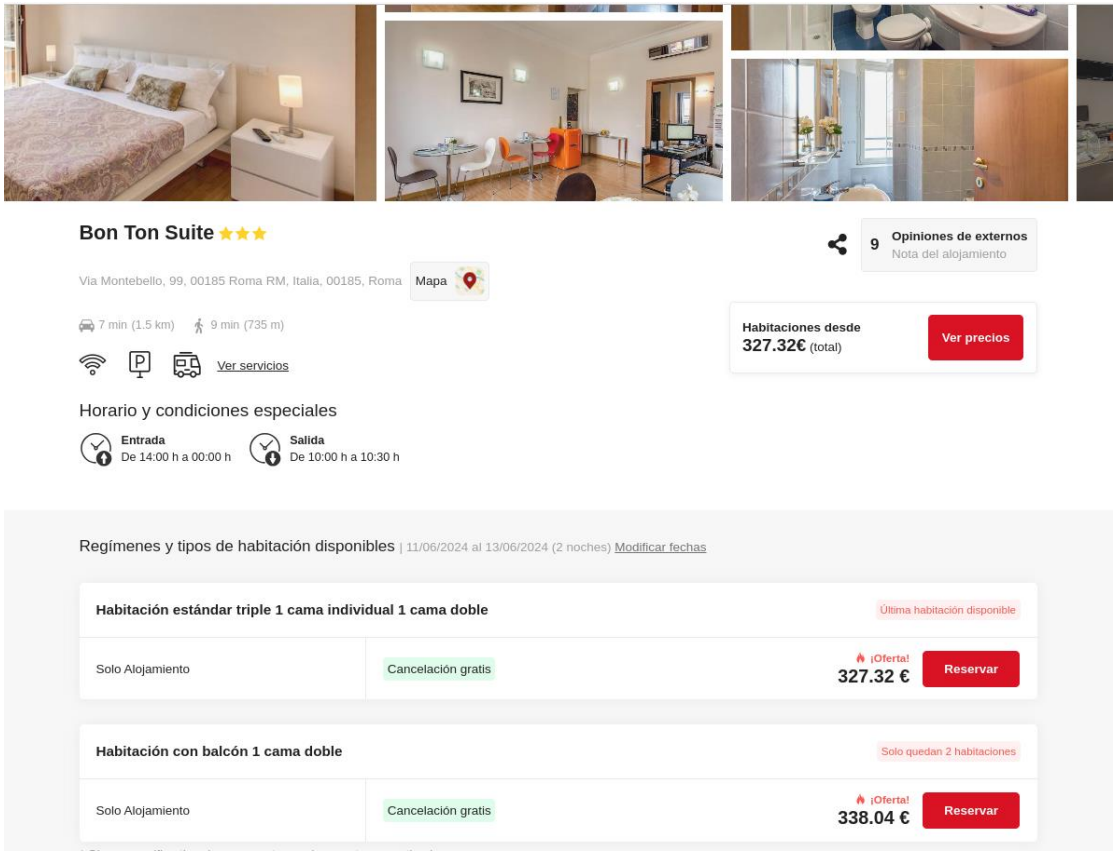
L'inconvenient està quan l'usuari vol consultar la informació de l'habitació exacta d'on s'allotjarà, i és que només hi ha 600 de 500.000 d'hotels amb alguna fitxa d'habitació creada, el que proporcionalment és el 0.12% d'hotels amb fitxes d'habitacions creades, una quantitat molt baixa.

Quin és el motiu d'aquest percentatge tan reduït?

El motiu és que les fitxes d'habitació s'han de crear manualment per un treballador de l'empresa, la qual cosa no es una feina ràpida, ja que, ha de recopilar informació d'aquella habitació i després emplenar tots els camps de la fitxa (m^2, serveis, fotografies, distribució de llits, text descriptiu, etc.)

Si com a càlcul aproximant fiquem que un treballador triga 5 minuts a fer una fitxa d'habitació, hi ha 500.000 hotels al sistema i cada hotel té de mitjana, 3 tipus d'habitació (n'hi ha que tenen moltes més), el treballador trigaria 5208 dies en crear totes les fitxes, una dada monstruosa.

A continuació, a la il·lustració 4, podem veure la *landing* de compra d'un hotel a la pàgina web d'Amimir. En aquesta es pot observar que només surt el nom de l'habitació i el preu, la qual cosa és una descripció bastant pobra.



Il·lustració 4 Landing compra hotel Amimir

2.2 Objectius

Com s'ha pogut observar anteriorment, la creació de fitxes manual, comporta un gran cost humà i econòmic, per la qual cosa, neix la necessitat d'un sistema que pugui crear aquestes fitxes d'una forma automàtica.

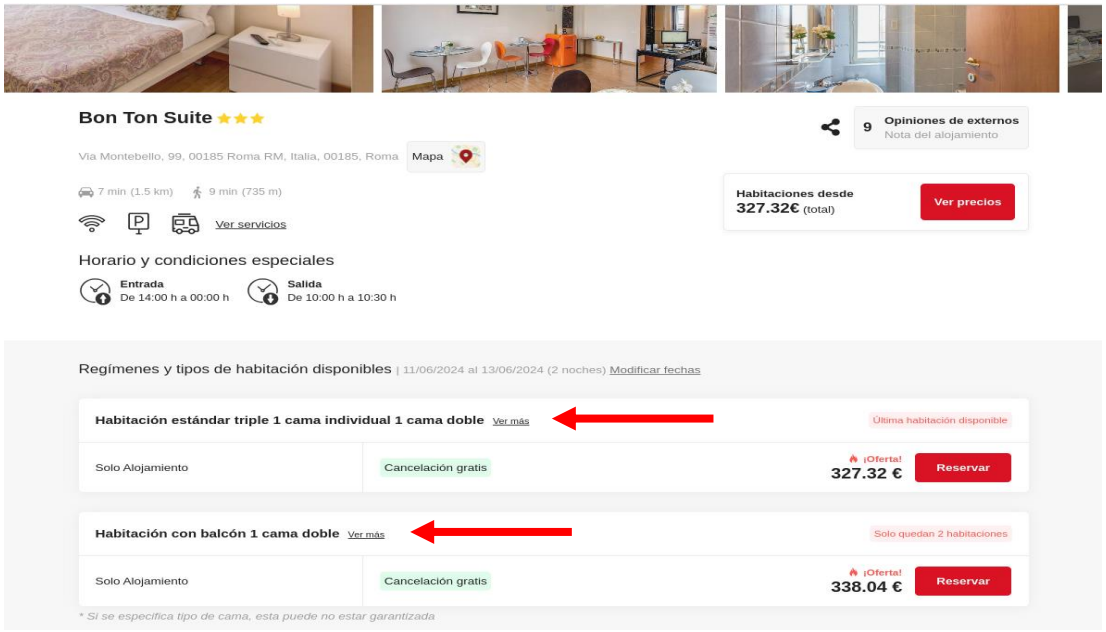
> Així doncs, l'objectiu principal d'aquest TFG, és dissenyar i implementar un sistema automàtic de creació i actualització de fitxes d'habitació a partir d'informació que proporcionen diferents proveïdors externs amb crides a les seves APIs.

Amb el desenvolupament d'aquest sistema, s'aconseguirà un millor tractament i configuració del producte que s'ofereix a la web, així com, millorar l'experiència dels usuaris a l'hora de navegar entre els diferents hotels disponibles.

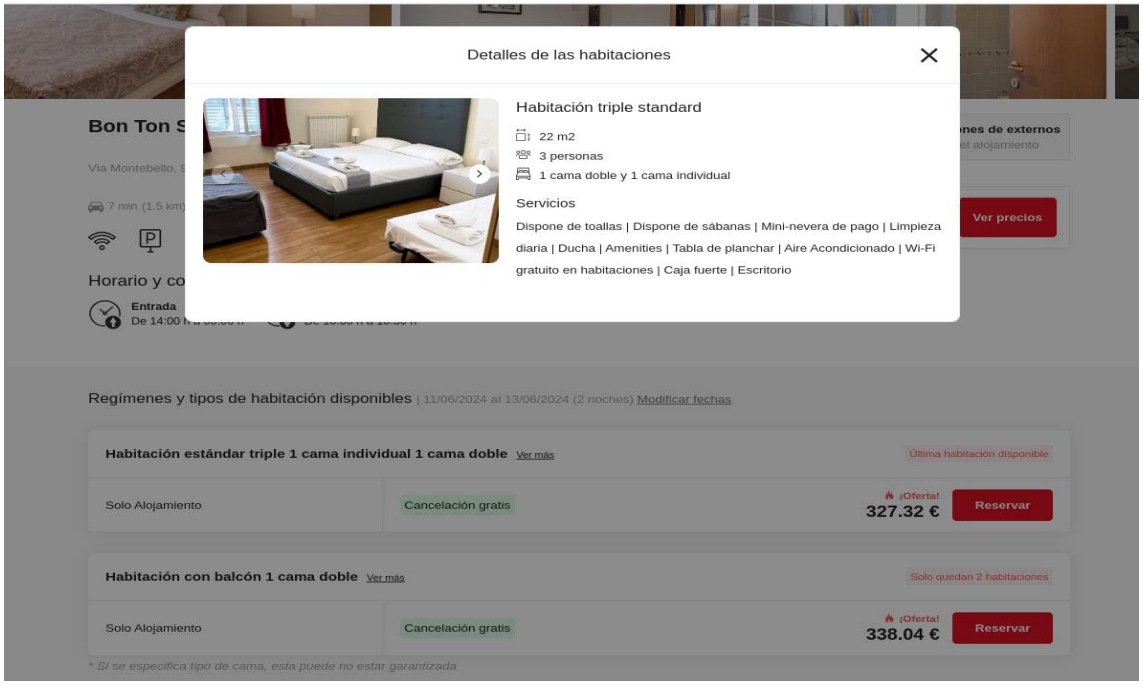
A continuació podem veure dues imatges de com es veu el mateix hotel, una vegada que ha passat pel sistema automàtic de creació de fitxes d'habitació.

A la primera imatge (Il·lustració 5) podem observar com ha aparegut l'opció de "Ver más", on si l'usuari fa clic, se li desplegarà un mòdul amb tota la informació d'aquella habitació (Il·lustració 6)

Personalment, crec que amb aquesta millora, la pàgina web fa un salt qualitatiu en vers al servei que oferim.



Il·lustració 5 Opcions "Ver Más"



Il·lustració 6 Fitxa d'habitació a la pagina d'Amimir

A més de l'objectiu principal d'aquest Treball de Fi de Grau, és important destacar que hi ha altres objectius relacionats amb l'aprenentatge personal i l'adquisició d'experiències, contribuint així, al meu desenvolupament com a enginyer informàtic.

Entre aquests podem trobar:

- Aprenentatge de diferents tecnologies de desenvolupament web.
- Treball en equip
- Resolució de problemes
- Conèixer la infraestructura i organització d'una empresa real
- Metodologies de gestió de projectes/equips

I moltes més que crec que han sigut i seran clau per al meu futur professional.

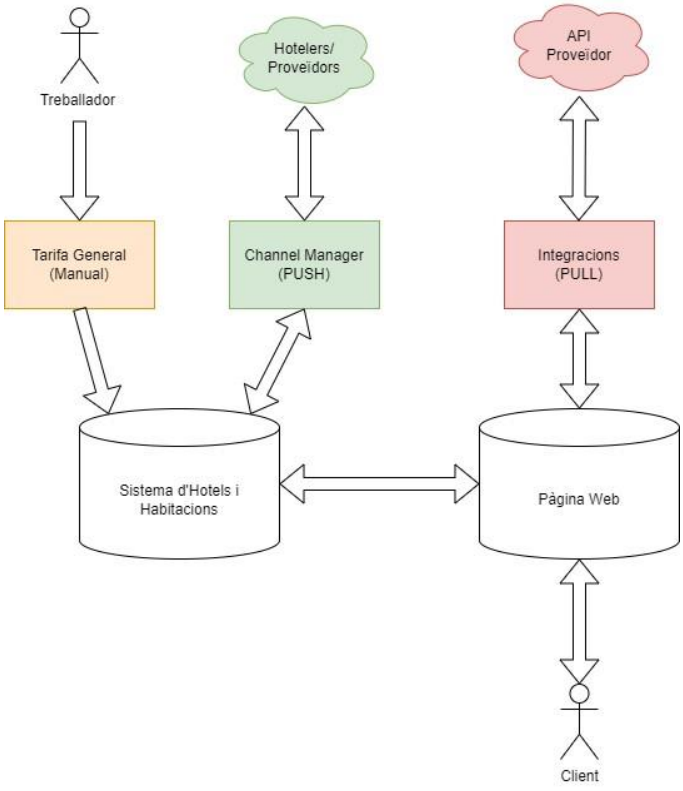
3 Sistema d'informació previ

Per poder passar a definir els requisits funcionals cal explicar breument el sistema d'informació de l'empresa previ al meu desenvolupament.

3.1 Mètodes de càrrega de producte

Considero important explicar com és carrega el producte al sistema de l'empresa, ja que està molt relacionat amb les habitacions i cal tenir alguns conceptes clars per entendre algunes funcions o explicacions de més endavant.

El producte principal d'Amimir i Esquiades són habitacions en allotjaments i forfets (en el cas d'Esquiades). A part de 4 o 5 hotels que són propietat de l'empresa, la resta ve donat per diferents proveïdors/ hotelers externs que ens envien diferents tarifes. Per oferir aquests preus als clients s'ha de carregar al sistema. Com que l'empresa i la infraestructura va evolucionant i cada hoteler/empresa treballa d'una forma diferent, existeixen 3 vies de càrrega de producte.



Il·lustració 7 Càrrega de producte

3.1.1 Tarifa General

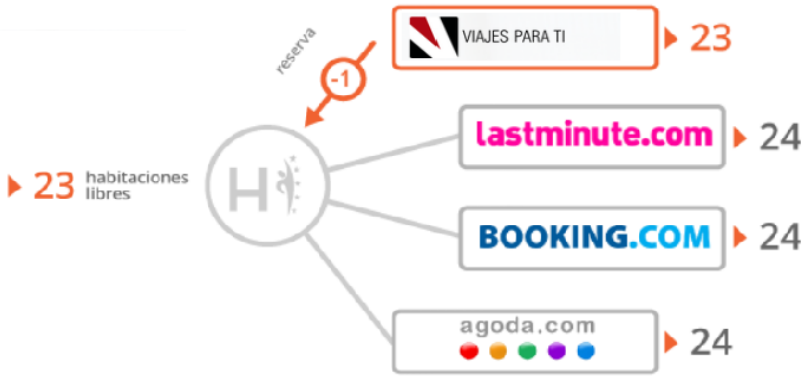
Aquest mètode és el més antic i més lent, però a la vegada el més fiable. Bàsicament, els treballadors de la secció "Producte" de l'empresa es comuniquen amb els hotels i pacten una tarifa de preus i habitacions, l'hoteler li envia les tarifes (normalment amb Excel) i aquest les carrega al sistema intern d'hotels-habitacions. Tot i que es un sistema fiable, ja que els treballadors tenen en tot moment el control de la disponibilitat i preus, pràcticament no es fa servir pel fet que es massa lent per fer-ho per molts hotels.

3.1.2 Channel Manager (Push)

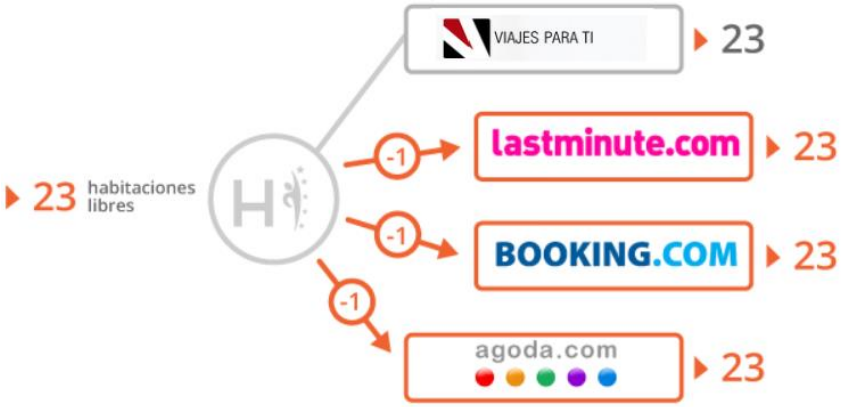
Aquest ja és un sistema informàtic més elaborat amb el qual s'automatitzen molts dels processos. Bàsicament, els proveïdors i hotelers externs, es connecten a nosaltres a través d'un intermediari anomenat "Channel Manager" per carregar la seva disponibilitat i preus sobre el sistema intern d'hotels-habitacions de l'empresa (Il·lustració 8).

Llavors quan un client fa una reserva, aquesta passa pel Channel i arriba a l'hoteler, el qual registra les dades(Il·lustració 9). Un mateix hotel pot estar connectat a diferents empreses de venda d'hotels a la vegada a través d'un Channel, llavors, si es fa la reserva d'una habitació en una empresa, el Channel actualitza la disponibilitat a la resta (Il·lustració 10).

Il·lustració 8 Càrrega disponibilitat



Il·lustració 9 Reserva al Channel



Il·lustració 10 Actualització disponibilitat

Amb aquest mètode, podem tenir controlats els preus de les habitacions en tot moment, ja que, primer es carreguen al sistema intern i després s'ofereix al client. Actualment, hi ha uns 1000 hotels que funcionin pel "Channel", una xifra bastant baixa. El motiu de què aquesta xifra sigui baixa, és que connectar-se per aquesta via implica un esforç al proveïdor extern per connectar-se al sistema del Channel de Amimir, i això fa que moltes empreses no vulguin utilitzar aquest mètode.

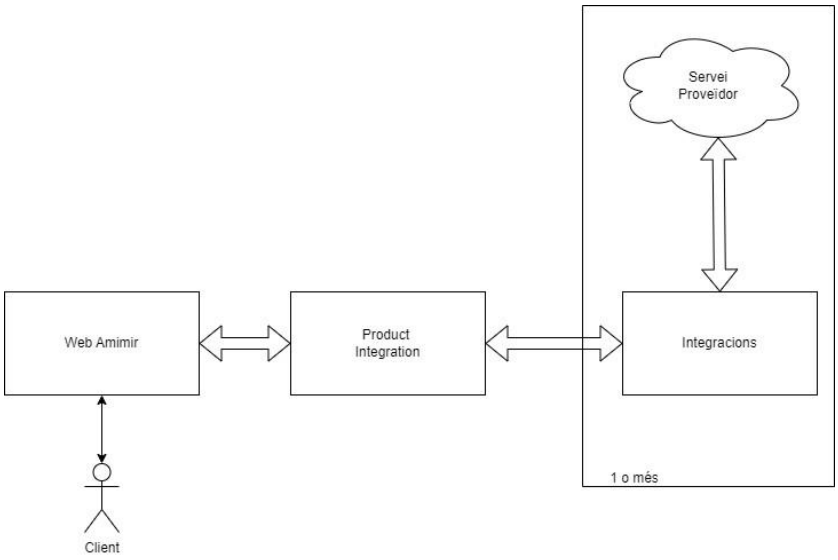
3.1.3 Integracions (Pull)

En aquest últim sistema, no s'emmagatzema cap informació referent a preus o tarifes d'habitacions dels hotels prèviament. Si no que, quan l'usuari realitza una cerca a la web, es llencen una o més peticions a les APIs dels proveïdors externs i aquest responen amb la llista de preus de les habitacions disponibles i les seves polítiques de cancel·lació, no retornen característiques descriptives de l'habitació més enllà del nom.

L'inconvenient d'aquest sistema és que depens del temps de resposta i latència del servidor del proveïdor, i donat que, la petició es fa en el moment en què el client fa la cerca, es busca que la web ofereixi resultats com més aviat millor.

El gran avantatge és que facilita molt connectar-se als proveïdors, ja que ells no han de fer cap esforç. Simplement fent una petició als seus "End Points" amb les dades de dates i ocupants amb el format que ells requereixen, et retornen tota la informació de les habitacions i preus al moment.

Com cada proveïdor disposa d'una infraestructura tecnològica diferent, s'han de fer programes fets a mesura per fer les connexions amb cada un d'ells. Per un costat s'han d'adaptar les dades abans fer la petició, per altre costat, s'han de tractar i estructurar les dades que ens retornen ells per poder utilitzar-les com a productes a la pàgina web.



Il·lustració 11 Sistema Pull

Al diagrama anterior (Il·lustració 11), podem veure el procediment del sistema pull. Per començar el client fa una cerca a la web d'Amimir, la web fa una petició al projecte "Product Integration" (**PI),** el qual s'encarrega de buscar quins proveïdors ofereixen el/s hotel/s buscats i fan peticions en paral·lel als proveïdors. El projecte que s'encarrega de fer la petició i estructurat de dades s'anomena "Integracions".

Així doncs, des de PI s'aixequen les diferents instàncies dels serveis que es comuniquen amb les Apis externes. A mesura que els servidors dels proveïdors van responent, es van validant i formatant les dades, per tal de poder tractar-les de manera igual indiferentment del proveïdor origen. PI va rebent aquestes dades i comparant-les entre elles per oferir el preu més baix, llavors es retornen a la web d'Amimir i es mostren al client.

Un dels inconvenients d'aquest mètode, és la pèrdua del control de les tarifes/preus, ja que com s'ofereix al client al moment i no s'emmagatzema, és més difícil controlar que tot està realment bé. És molt important que al projecte Integracions hi hagi un bon control d'errors i diferents casuístiques per tenir un sistema robust i no tenir errors greus.

Com hem comentat, aquest sistema ofereix un gran abast a l'hora de connectar-se amb molts proveïdors, per aquest motiu, representa la majoria de producte que es ven a la pàgina d'Amimir.

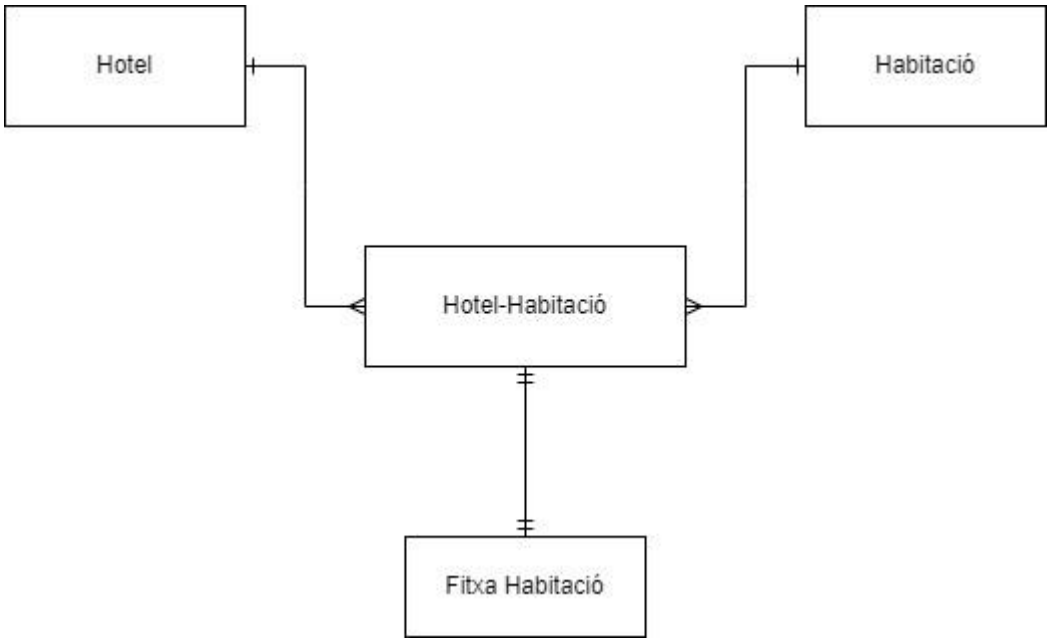
3.2 Sistema d'hotels i habitacions

Anteriorment, hem parlat dels proveïdors, aquests poden ser empreses del tipus "**BedBank**", les quals no tenen cap portal web de venda de producte i només ofereixen disponibilitat de producte per a altres webs com poden ser exemples Tour10, Smyrooms, HotelBeds…

O, per altra banda, empreses que tenen el seu propi portal web on venen el seu producte i a més a més també ofereixen hotels a altres pàgines web per la seva comercialització. Exemples: Vacanceole, Booking, Expedia...

Per poder oferir el producte dels proveïdors, indiferentment del mètode de càrrega de producte, necessitem tenir l'hotel creat al sistema. D'aquest, s'emmagatzemarà, entre altres, les dades referents a la fitxa d'hotel. Aquesta, és tota la informació que veurà el client i que descriu l'hotel, les dades de la fitxa d'hotel són: Fotografies, Serveis, Horaris, Adreça, Pàrquings, etc.

A part de la informació de l'hotel, també podem tenir dades de quines habitacions té aquell allotjament i com són. Només trobarem dades d'habitacions en el cas que siguin producte de **Tarifa General** o **Channel Push**, ja que, es necessita tenir habitacions creades per poder carregar els preus que ens envien els proveïdors sobre aquestes. Normalment un hotel que és ven a través del Channel, només l'ofereix un únic proveïdor, per tant, aquest és el que ens comunica quantes habitacions s'han de crear i de quin tipus són, ja que s'ha negociat el preu d'aquestes prèviament.



Il·lustració 12 Relació Hotel-Habitació

A la il·lustració anterior (Il·lustració 12) podem veure la relació entre Hotel i Habitació. Un hotel pot tenir 0 o més habitacions, per poder unir aquestes entitats s'utilitza la relació Hotel-Habitació. En el cas que existeixi una fitxa d'habitació, aquesta, fa referència a l'entitat Hotel-Habitació. Això és perquè l'entitat Habitació, és un tipus d'habitació genèrica és a dir "Habitació doble", no és fins quan s'enllaça a un hotel amb la relació Hotel-Habitació, que no és l'habitació concreta d'aquell hotel i, per tant, pot tenir una fitxa d'habitació que la descrigui.

En canvi, en els hotels que es venen a través d'**Integracions Pull**, és molt habitual la casuística de què un mateix hotel l'ofereixin 1 o més proveïdors, i més, no hi ha cap negociació de preus entre la nostra empresa i el proveïdor, per tant, no es crea cap informació referent a les habitacions.

3.3 Actualització dades d'hotel

Actualment, tenim informació registrada de més de 500.000 hotels al sistema. Per poder mantenir aquestes fitxes d'hotels actualitzades existeix un sistema el qual va recorrent tots els hotels que tenim, i, a través de scrapping a Google i peticions a Apis de proveïdors, va recollint informació d'aquell hotel en concret, la tracta i l'actualitza a la base de dades.

Aquest és un sistema que està en funcionament sempre i no depèn en res del sistema de compra de Amimir o Esquiades. Bàsicament, va recorrent tots els hotels que tenim enregistrats i els va actualitzant. El projecte s'anomena "**External Info Collector**" i més endavant en parlaré en extensió, ja que gran part del meu projecte està ubicat en aquest sistema.

3.4 Identificació d'hotels entre proveïdors

En el context de la col·laboració amb diversos proveïdors, hi ha una problemàtica d'identificació de cada hotel, ja que, al nostre sistema utilitzem un ID únic per un hotel en concret, però al sistema del proveïdor és un ID totalment diferent. Donat que es treballa amb molts proveïdors i que hi ha molts hotels disponibles, ens trobem amb una gran variació de codis.

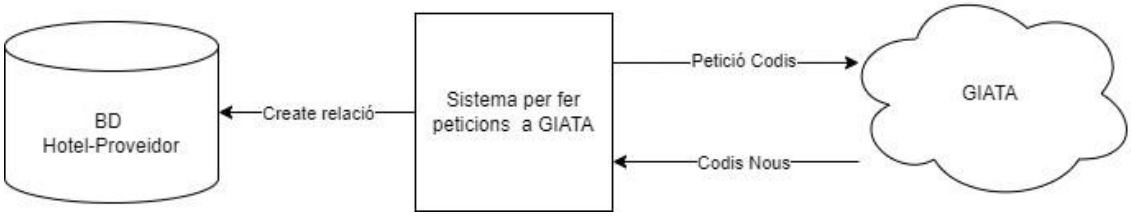
És necessari saber l'ID de l'hotel del sistema del proveïdor, donat que quan es fa una petició (Pull), s'ha d'identificar l'hotel amb el seu codi i no amb el nostre, ja que, ells no tenen constància dels nostres codis interns.

Per abordar aquesta complexitat, VPT, i la majoria d'empreses del sector turístic treballen amb una empresa externa la qual facilita la comunicació i normalització de dades entre empreses anomenada GIATA.

GIATA és una empresa alemanya que especialitza en la gestió i distribució de dades de la indústria turística. Proporciona una àmplia gamma de serveis relacionats amb la gestió de dades d'hotels i destinacions turístiques. D'aquesta empresa utilitzem l'eina anomenada "**GiataMulticodes**"

GiataMulticodes permet la normalització de la informació mitjançant la gestió de múltiples codis d'hotels. Aquesta eina és essencial per estandarditzar la comunicació i l'intercanvi de dades amb els diversos proveïdors, assegurant una integració fluida i

eficient. El funcionament d’aquest (Il·lustració 13) es basa a connectar-nos a ells cada poc temps i fer una petició de tots els codis nous dels proveïdors amb els quals treballem. Aquests codis els emmagatzemem a la BD interna del sistema i així sempre que hem de fer una petició d’un hotel a un proveïdor, es consulta l’ID corresponent a l’Api externa.



Il·lustració 13 MultiCode de GIATA

És important destacar que GIATA opera exclusivament a nivell d'hotel i, per tant, no ofereix la capacitat de relacionar codis específics d'habitacions. Aquesta limitació es deu a diversos factors, entre els quals destaca la variabilitat en les pràctiques de gestió de dades entre els proveïdors.

Els sistemes d'identificació d'habitacions poden variar significativament d'un proveïdor a un altre, i en molts casos, els proveïdors no tenen les seves habitacions identificades amb codis únics o estàndard. Això crea una complexitat addicional en la integració i la gestió de dades, ja que no hi ha una correspondència directa entre els codis d'habitacions utilitzats pels diferents proveïdors.

Com a resultat d'aquesta situació, el nostre sistema actual es centra exclusivament en la identificació i gestió d’hotels, sense la capacitat de relacionar codis d'habitacions entre diferents proveïdors. Això pot representar un repte en l'harmonització de la informació i en la integració de dades, especialment quan es tracta de proporcionar una experiència coherent i precisa als clients.

4 Especificacions

Els requisits són funcions, característiques o restriccions que ha de complir el sistema que estem desenvolupant.

Marcar els requisits del projecte des d'un principi és fonamental per al desenvolupament del sistema. Serveixen com a guia per al procés de disseny/implementació i poder estudiar si s'estan complint com el client o l'equip havia especificat.

A l'hora de parlar de requisits podem distingir de 2 tipus:

·Requisit Funcional (RF): Defineix una funció del sistema o dels seus components, estableix un comportament d'aquest sistema. Descriuen interaccions entre Software i Usuari, així com el comportament del Software.

·Requisit No Funcional (RNF): Aquests complementen als requisits funcionals, defineix les qualitats i característiques del sistema. Normalment, van relacionats a paràmetres com Seguretat, Rendiment, Eficiència, etc.

4.1 Requisits Funcionals

RF 1. Un usuari pot crear relació entre un Id d'hotel, un Id de proveïdor, un codi d'habitació del proveïdor i una habitació del nostre sistema de manera manual.

RF 2. Un usuari pot veure les relacions Hotel-Habitació-Proveïdor creades.

RF 3. Un usuari pot modificar una relació Hotel-Habitació-Proveïdor.

RF 4. Un usuari pot eliminar una relació Hotel-Habitació-Proveïdor.

RF 5. Un usuari pot aplicar filtres per cercar l'habitació del sistema a l'hora de crear la relació.

RF 6. Un usuari pot veure de manera ràpida i clara quins codis d'habitació existeixen per aquell hotel-proveïdor i quins s'han relacionat i quins no.

RF 7. Un usuari pot crear una fitxa d'habitació de manera automàtica fent una petició al proveïdor de la relació.

RF 8. Mostrar la fitxa d'habitació (si està creada) per al producte que ens envien els proveïdors per **Pull.**

RF 9. Un usuari pot crear una relació de TextHabitació-Proveïdor-Habitació

RF 10. Un usuari pot Llistar, Filtrar, Eliminar i modificar les relacions TextHabitació-Proveïdor-Habitació

RF 11. Que el sistema ExternalInfoCollector rebi i processi les dades d'habitacions dels proveïdors, les compari i busqui si hi existeix alguna relació d'aquella habitació del proveïdor amb alguna del nostre sistema. Si existeix, que enviï les dades al sistema de creació de fitxes d'habitació.

RF 12. Que el sistema de creació de fitxes pugui crear fitxes a partir de la informació del ExternalInfoCollector i relacioni la fitxa amb l'habitació del proveïdor.

RF 13. Que l'usuari pugui observar Logs del comportament del sistema.

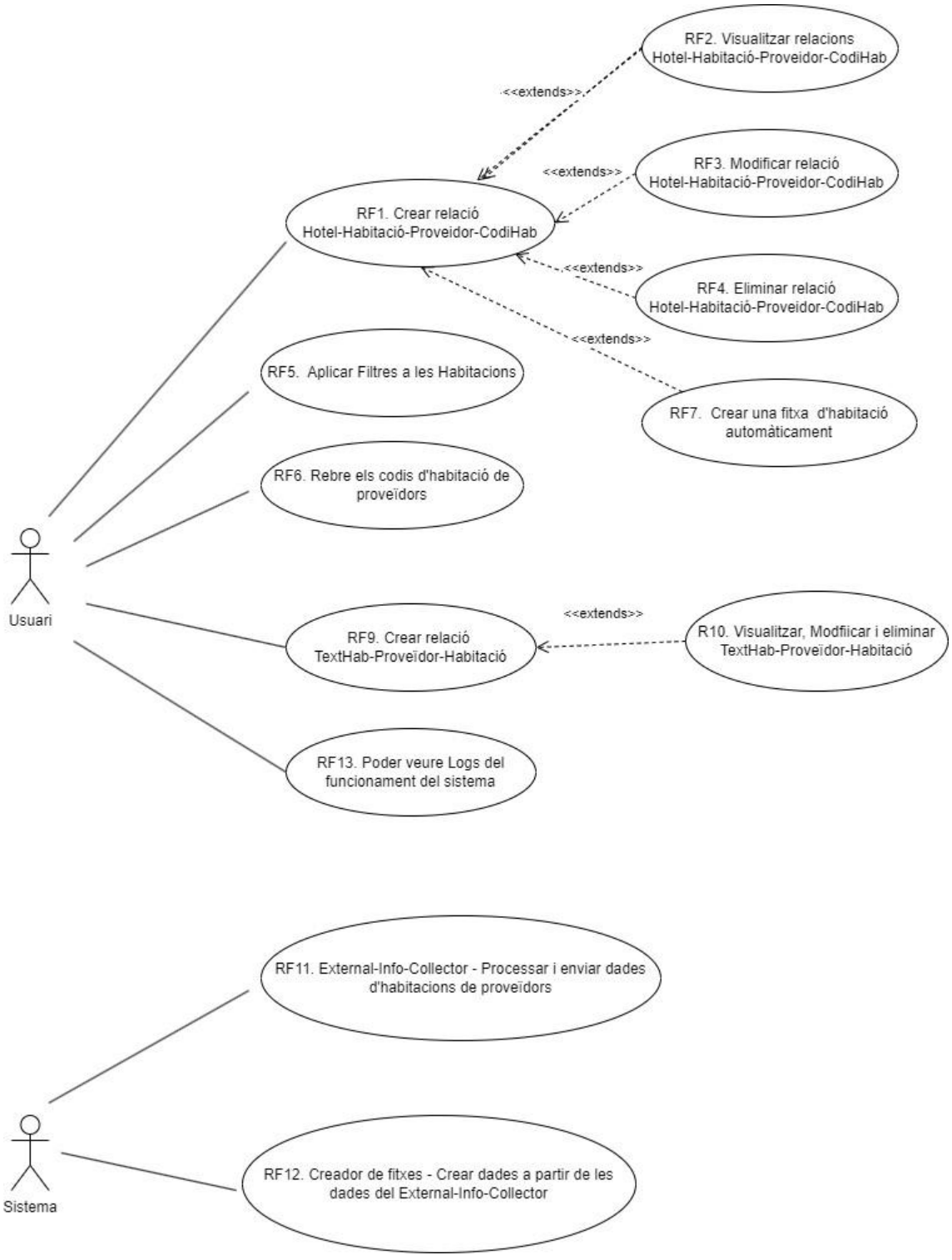
4.2 Requisits no funcionals

RNF 1. El procés de crear una fitxa d'habitació automàtica ha de ser bastant ràpid, no hauria de tardar més de 20 segons.

RNF 2. Les pantalles de l'Admin han de tenir una interfície ràpida i intuïtiva.

RNF 3. El sistema no ha de generar uns elevats costos a l'empresa.

4.3 Diagrama Cas d'ús



Il·lustració 14 Diagrama casos d'ús

4.4 Casos d'ús textuals

Una vegada vist els requisits i el diagrama de casos d'ús de la il·lustració 14, estudiarem els casos d'ús de manera textual. Amb això es pot descriure la interacció de l'usuari i el sistema d'una forma més rigorosa.

4.4.1 Cd'ú 01. Creació Relació Hotel-Proveïdor-Habitació-CodiHabitacióProveïdor

Resum de la funcionalitat: Crear una entitat per poder relacionar el codi d'habitació d'un hotel i un proveïdor amb una habitació del nostre sistema.

Paràmetres d'entrada: Id hotel, Id Proveïdor, Id habitació, Codi habitació proveïdor.

Paràmetres de sortida: Cap.

Actors: Usuari

Precondició: Que l'hotel, el proveïdor i l'habitació estiguin creats al sistema.

Postcondició: S'ha creat una nova relació a base de dades.

Procés normal:

1. L'usuari accedeix a la pantalla.
2. El sistema carrega totes les habitacions i proveïdors existents dins d'uns selectors.
3. L'usuari selecciona les dades a través dels selectors.
4. El sistema comprova que existeixen l'hotel, el proveïdor i l'habitació.
5. El sistema comprova si ja existeix aquesta relació.
6. El sistema detecta l'usuari que està fent l'acció i també la data i hora.
7. El sistema comprova si existeix una relació Hotel-Habitació i si no és així la crea.
8. El sistema crea la relació a base de dades juntament amb l'usuari i la data.
9. El sistema notifica l'usuari el resultat de la creació.

Procés alternatiu:

3a. El sistema prem al botó de Filtrar Habitacions

3a1. Cd'05

4a. El sistema no troba alguna de les entitats (hotel, habitació, proveïdor)

4a.1 El sistema notifica l'error i torna al pas 1.

5a. El sistema comprova que ja existeix aquesta relació exacta.

5a.1 El sistema notifica i torna al pas 1.

4.4.2 Cd'ú 02. Llistar relacions

Resum de la funcionalitat: Poder visualitzar totes les relacions creades que té un hotel.

Paràmetres d'entrada: Id hotel.

Paràmetres de sortida: Llista de relacions.

Actors: Usuari

Precondició: Que l'hotel existeixi.

Postcondició: Cap.

Procés normal:

1. L'usuari entra dins de la pantalla de "Relacions" d'aquell hotel en concret
2. El sistema rep el Id d'hotel.
3. El sistema comprova que existeix l'hotel.
4. El sistema comprova si l'hotel té alguna relació creada.
5. El sistema retorna les relacions de Hotel-Proveïdor-Habitació-CodiHabitació.

Procés alternatiu:

4a. El sistema no troba cap relació amb aquell hotel

4a.1 El sistema retorna una llista buida.

4.4.3 Cd'ú 03 Modificar relació

Resum de la funcionalitat: Modificar algun atribut de la relació.

Paràmetres d'entrada: Id relació Hotel-Proveïdor-Habitació-CodiHabitació.

Paràmetres de sortida: Cap.

Actors: Usuari

Precondició: Que existeixi la relació.

Postcondició: La relació és modificada.

Procés normal:

1. Cd'ú 02.
2. L'usuari selecciona una de les relacions de la llista i prem el botó de "Modificar"
3. El sistema valida que la relació existeixi.
4. El sistema demana a l'usuari els nous camps:
    a. Habitació del sistema
    b. Proveïdor
    c. Codi Habitació Proveïdor
5. El sistema valida que les entitats Habitació i Proveïdor existeixin.
6. El sistema valida que aquesta nova relació modificada no existeixi.
7. El sistema modifica la relació a base de dades
8. El sistema notifica l'estat de la creació.

Procés alternatiu:

3a. El sistema no troba la relació que es vol modificar.

3a.1 El sistema notifica i torna al pas 1.

5a. El sistema no troba alguna de les entitats (habitació, proveïdor)

5a.1 El sistema notifica l'error i torna al pas 1.

6a. El sistema troba que ja existeix la combinació després de la modificació.

6a1. El sistema notifica i torna al pas 1.

4.4.4 Cd'ú 04. Eliminar relació

Resum de la funcionalitat: Esborra una relació ja creada

Paràmetres d'entrada: Id relació Hotel-Proveïdor-Habitació-CodiHabitació.

Paràmetres de sortida: Cap.

Actors: Usuari

Precondició: Que la relació existeixi.

Postcondició: La relació és borra de base de dades

Procés normal:

1. Cd' 02
2. L'usuari selecciona la relació de la llista i prem el botó Esborrar.
3. El sistema comprova que la relació existeix.
4. El sistema esborra la relació de base de dades.
5. El sistema notifica a l'usuari.

Procés alternatiu:

3a. El sistema no troba la relació

3a.1 El sistema notifica l'error i torna al pas 1.

4.4.5 Cd'ú 05. Aplicar filtres d'habitacions

Resum de la funcionalitat: Possibilitat d'aplicar un filtratge per seleccionar l'habitació adequada per la relació, donat que existeixen moltes al sistema

Paràmetres d'entrada: Ocupació, Ocupació Min, Ocupació Max, Num nens, Num adults, NomesNens, NomesAdults.

Paràmetres de sortida: Llistes d'habitacions.

Actors: Usuari

Precondició: Cap

Postcondició: S'actualitzarà el selector d'habitacions

Procés normal:

1. L'usuari prem el botó d' "Aplicar Filtres"
2. L'usuari introdueix els paràmetres d'entrada per al filtratge.
3. El sistema fa un filtratge a base de dades amb les dades de l'usuari.
4. El sistema retorna la llista d'habitacions i actualitza al selector d'habitacions del client.

Procés alternatiu:

4a. El sistema no troba cap habitació amb el criteri de cerca.

4a1. El sistema retorna una llista buida.

4.4.6 Cd'ú 06 Rebre els codis d'habitacions dels proveïdors

Resum de la funcionalitat: Permetre a l'usuari rebre els codis d'habitació de l'hotel seleccionat per un proveïdor determinat i diferenciar els que ja estan mapejats dels que no.

Paràmetres d'entrada: Id hotel, Id proveïdor.

Paràmetres de sortida: Llista habitacions del proveïdor.

Actors: Usuari, Api Proveïdor

Precondició: Que existeixi l'hotel i el proveïdor

Postcondició: Cap.

Procés normal:

1. L'usuari selecciona un proveïdor.
2. El sistema valida que el proveïdor existeixi.
3. El sistema valida que el proveïdor tingui habilitada la petició per demanar informació d'un hotel, anomenada **getHotel**.
4. El sistema mostra el botó "Veure habitacions" a l'usuari.
5. L'usuari prem el botó "Veure habitacions".
6. El sistema fa una petició "**getHotel**" al proveïdor per aquell hotel.
7. El servidor del proveïdor retrona la informació d'aquell hotel.
8. El sistema processa les dades de l'hotel del proveïdor i extreu la informació de les habitacions.
9. El sistema comprova si existeix alguna relació amb algun codi que ha retornat el proveïdor.
10. El sistema retorna a l'usuari una taula amb els noms de les habitacions del proveïdor, els seus respectius codis i una *check* que determina si s'ha mapejat o no.

Procés alternatiu:

2a. El sistema no troba el proveïdor.

2a1. El sistema notifica l'error i torna al pas 1.

3a. El proveïdor extern no té habilitada la petició **getHotel**.

3a1. NO es mostra el botó "Veure habitacions".

3a2. Torna al pas 1.

6a. La comunicació amb el proveïdor falla.

6a1. El sistema notifica l'error i torna al pas 1.

4.4.7 Cd'ú 07. Crear fitxa d'habitació automàticament

Resum de la funcionalitat: Fer una petició a l'Api d'un proveïdor i crear una fitxa d'habitació del sistema amb les dades rebudes de manera automàtica.

Paràmetres d'entrada: Id relació Hotel-Proveïdor-Habitació-CodiHabitació.

Paràmetres de sortida: Cap.

Actors: Usuari, Api proveïdor

Precondició: Que existeixi la relació Hotel-Proveïdor-Habitació-CodiHabitació

Postcondició: S'haurà creat una nova fitxa i enllaçat amb una habitació del sistema.

Procés normal:

1. Cd'ú 02.
2. L'usuari selecciona una de les relacions i prem el botó de "Carregar Fitxa"
3. El sistema fa una crida "**getHotel"** al proveïdor de la relació.
4. L'Api del proveïdor retorna la informació de l'hotel
5. El sistema cerca el codi d'habitació de la relació dins de la resposta del **getHotel.**

6. El sistema formata i estructura les dades de l'habitació del proveïdor
7. El sistema comprova si ja existeix una fitxa d'habitació per aquella relació Hotel-Habitació.
8. El sistema crea una fitxa d'habitació amb les dades formatades del proveïdor i es registra a base de dades.
9. El sistema notifica el resultat de l'operació a l'usuari.

Procés alternatiu:

4a. L'Api retorna un error

4a1. El sistema notifica a l'usuari i torna al pas 1.

5a. L'Api no retorn l'habitació amb el codi de la relació

5a1. El sistema notifica a l'usuari i torna al pas 1.

7a. El sistema troba que ja existeix aquella una fitxa per aquella habitació.

7a1. Si la fitxa es va crear amb aquell proveïdor, l'actualitza

7a2. Si la fitxa es va crear per un altre proveïdor o a mà, notifica i torna al pas 1

4.4.8 Cd'ú 08. Mostrar opció "Veure Fitxa"

En aquest cas explicaré la funcionalitat perquè crec que serà més entenedor que amb el diagrama. Originalment, només es mostraven fitxes d'habitació del producte que arribava del **Channel Pull**, el qual, ja es trobava carregat al sistema intern d'Hotel-Habitacions. Però com bé hem explicat abans, el producte de **Channel** tot i ser més controlat, és molt més reduït que el de **Integracions Pull**, per tant, s'havia de modificar el sistema per poder mostrar l'opció de "Veure Fitxa" per al producte que ens arribes pel **Pull**.

Així doncs, l'objectiu d'aquest Cas d'ús es poder relacionar, durant el procés de compra del client, les habitacions disponibles que retorna l'Api externa, amb les habitacions del sistema.

Procés Normal:

1. El client selecciona un hotel a la pàgina d'Amimir
2. El sistema fa una petició de disponibilitat a l'Api del proveïdor.
3. L'Api externa retorna la disponibilitat
4. El sistema formata les dades i cerca els IDs de les habitacions (si és que n'hi ha)
5. El sistema busca la relació per aquell Hotel Proveïdor i Codi d'habitació
6. El sistema mostra a l'usuari la fitxa d'habitació que hi ha enllaçada a la relació anterior

Procés Alternatiu:

5a. No existeix cap relació per aquell ID d'habitació.

5a1. No es mostra el botó de "Ver Fitxa".

4.4.9 Cd'ú 09. Crear relació Habitació-TextTipusHabitació-Proveïdor

Resum de la funcionalitat: Crear una entitat que pugui relacionar una habitació del sistema, un text de tipus d'habitació ("Habitació doble vistes mar") i un proveïdor.

Paràmetres d'entrada: Id habitació, Id proveïdor, text habitació.

Paràmetres de sortida: Relació creada.

Actors: Usuari

Precondició: Que el proveïdor i l'habitació estiguin creats al sistema.

Postcondició: S'ha creat una nova relació a base de dades.

Procés normal:

1. El sistema demana les dades a l'usuari.
2. L'usuari introdueix les dades
3. El sistema comprova que existeixen el proveïdor i l'habitació.
4. El sistema comprova si ja existeix aquesta relació.
5. El sistema crea la relació a base de dades
6. El sistema notifica l'usuari el resultat de la creació.

Procés alternatiu:

3a. El sistema no troba alguna de les entitats

3a.1 El sistema notifica l'error i torna al pas 1.

4a. El sistema comprova que ja existeix aquesta relació exacta.

4a.1 El sistema notifica i torna al pas 1.

4.4.10 Cd'ú 10. Llistar modificar i eliminar relacions Habitació-TextTipusHabitació-Proveïdor

He agrupat en un únic cas d'ús les accions de llistar, modificar i eliminar relacions, donat que, ja he explicat el seu cas d'ús individual per l'entitat "Hotel-Habitació-CodiHabitació-Proveïdor" i en ser les mateixes accions per a entitats semblants no crec que sigui necessari repetir detalladament el procés de cada una d'elles.

El procediment es pot trobar explicat en els casos d'ús 2, 3 i 4.

4.4.11 Cd'ú 11. External-Info-Collector processar habitacions dels proveïdors

Resum de la funcionalitat: L'external Info Collector és el sistema automàtic que va actualitzant les dades del hotels constantment. Ara ha de tractar les dades deld proveïdors, comparar-les i seleccionar-les per crear una nova fitxa.

Paràmetres d'entrada: Id hotel

Paràmetres de sortida: Dades d'habitacions.

Actors: Sistema, Api proveïdor

Precondició: Que l'hotel existeixi i l'Api del proveïdor tingui dades d'aquest.

Postcondició: S'han emmagatzemat les dades de les habitacions.

Procés normal:

1. Entra un hotel a la cua del sistema
2. El sistema fa una crida getHotel per als 2 proveïdors amb els quals funciona
3. Els proveïdors retornen la informació de l'hotel
4. El sistema formata i emmagatzema les dades de les habitacions

5. El sistema comparà les dades dels dos proveïdors i escull les “millors”.
6. Es registren uns Logs amb les dades i quin proveïdor s’ha escollit.
7. El sistema envia les dades de totes les habitacions d’aquell hotel a un servei que les intentarà enllaçar amb les habitacions del nostre sistema i crear les fitxes (Cd’u 12).
8. El Cd’ú 12 retorna que ha fet amb cada una de les habitacions.
9. Es guarden els Logs a base de dades

Procés alternatiu:

3a. Cap proveïdor retrona informació d’aquell hotel

3a.1 Es creen uns logs i es passa al següent hotel.

5a. Només hi ha les dades d’un proveïdor

5a1. Es seleccionen aquelles dades i continua el procés.

4.4.12 Cd’ú 12. Crear fitxes a partit de dades del External-Info-Collector

Resum de la funcionalitat: El servei rebrà una llista amb les habitacions d’un hotel d’un proveïdor i el sistema intentarà enllaçar aquestes a les habitacions del nostre sistema. Per fer-ho buscarà el nom de cada habitació (Ex:Habitació triple accés piscina) dins de les relacions Habitació-Proveïdor-TextHabitació. En el cas de què trobi la relació i aquesta tingui una habitació del sistema assignada, crearà l’habitació, la fitxa i l’enllaçarà amb l’hotel.

Paràmetres d’entrada: Llista habitacions.

Paràmetres de sortida: Llista id’s habitacions i que s’ha fet amb cada una d’elles.

Actors: Sistema

Precondició: Tenir relacions Habitació-Proveïdor-TextHabitació creades.

Postcondició: Fitxes d’habitació creades/actualitzades i/o noves relacions Habitació-Proveïdor-TextHabitació.

Procés normal:

1. El sistema rep una llista d’habitacions i les seves característiques.

2. Per cada una d'elles el sistema comprova si ja tenim una fitxa d'habitació creada al sistema.
3. El sistema comprova si tenim algun registre Habitació-Proveïdor-TextHabitació amb aquell nom d'habitació i proveïdor.
4. El sistema crea l'habitació a la qual fa referència la relació i crea la fitxa amb les dades rebudes i enllaça l'habitació creada amb l'hotel.
5. Es registren Logs de la creació.
6. Retorna una llista indicant que s'ha fet amb cada habitació (s'ha creat fitxa, s'ha actualitzat fitxa, s'ha creat relació Habitació-Proveïdor-TextHabitació o s'ha afegit 1 als intents d'una relació)

Procés alternatiu:

2a. Ja existeix una fitxa d'habitació

2a1. S'actualitza la fitxa amb les dades.

2a2. Es registren logs i passa a la següent habitació Pas 2

3a. Troba una relació Habitació-Proveïdor-TextHabitació, però no té cap habitació assignada.

3a1. Suma 1 al comptador d'intents

3a2. Es registren logs i passa a la següent habitació Pas 2

3b. No troba cap relació amb aquell nom i proveïdor.

3b1. Crea una relació només amb el nom de l'habitació i el proveïdor (però sense cap habitació del sistema) i el comptador d'intents a 1.

3b2. Es registren logs i passa a la següent habitació Pas 2

4.4.13 Cd'ú 13. Consultar els Logs del procés

Resum de la funcionalitat: Consultar informació del procés de creació automàtic de fitxes d'habitació.

Paràmetres d'entrada: Dates de cerca

Paràmetres de sortida: Llista d'hotels que han passat per l'External-Info-Collector

Actors: Usuari

Precondició: Que s'hagi processat algun hotel.

Postcondició: Cap.

Procés normal:

1. L'usuari accedeix a la pantalla "Informació External Info Collector".
2. L'usuari introdueix les dates en les quals vol buscar els Logs.
3. El sistema consulta a base de dades els Logs de tots els hotels d'aquelles dates.
4. El sistema mostra un llista amb tots els hotels.
5. L'usuari selecciona un hotel de la llista
6. En seleccionar l'hotel, es despleguen tots els Logs del procés, entre ells, quin proveïdor s'ha seleccionat, quines habitacions ha retornat el proveïdor i que s'ha fet amb cada habitació.

Procés alternatiu:

3a. No hi ha cap informació referent a aquelles dates

3a1. No es mostrà cap hotel.

3a2. Pas 2.

4.5 Diagrama de Seqüències

Amb tots els casos d'ús ja explicats, a continuació s'il·lustraran els diagrames de seqüències d'aquests.

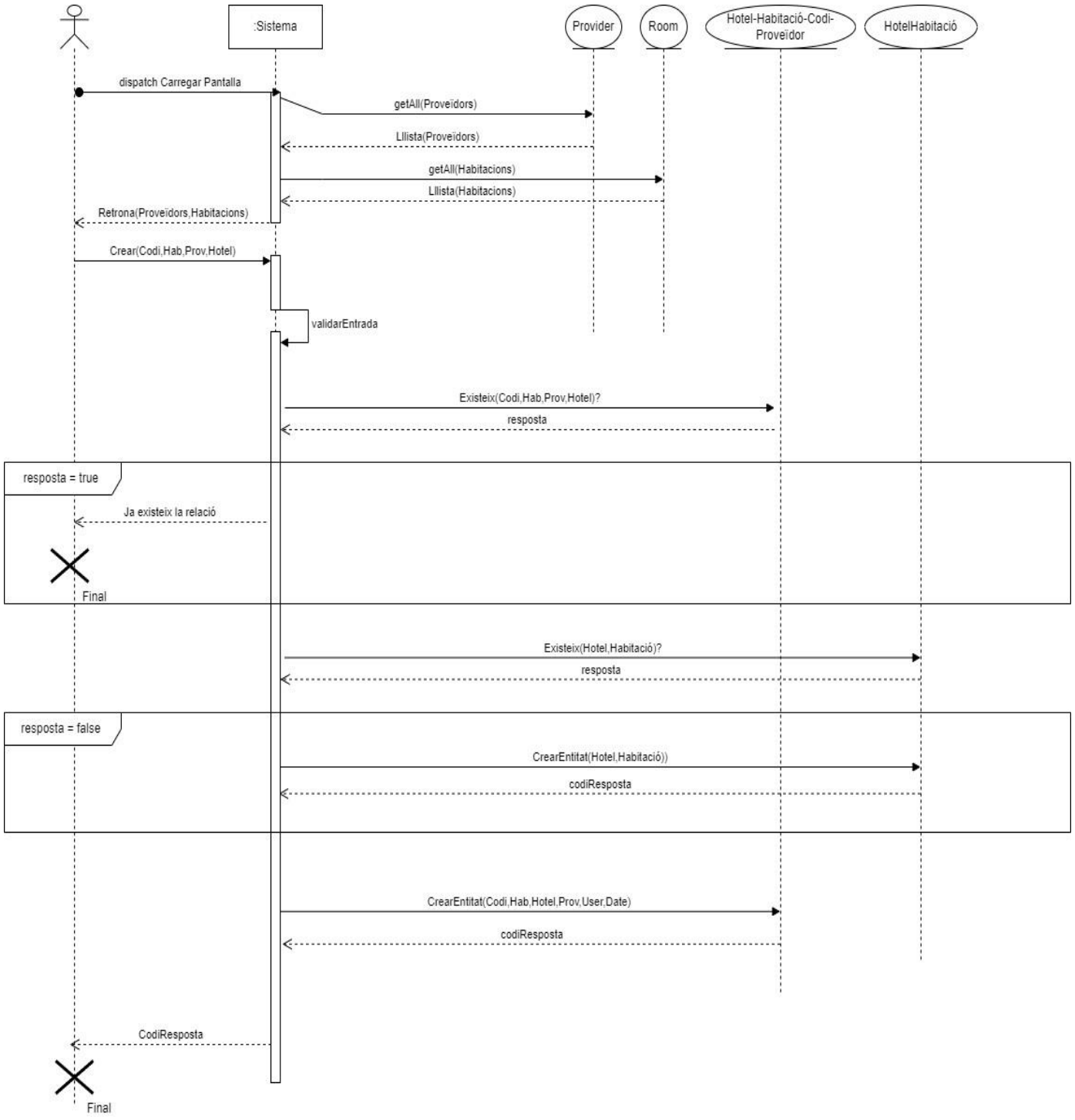
Un diagrama de seqüències és una representació gràfica que mostra la interacció entre objectes dins d'un sistema o entre sistemes al llarg del temps. És una eina fonamental en el modelatge i disseny de sistemes de programari i permet visualitzar el flux de missatges entre els diferents elements del sistema durant l'execució d'un escenari o procés determinat.

Només mostraré els diagramés d'alguns dels casos d'ús:

- D'algun cas d'ús senzill com crear o modificar entitats

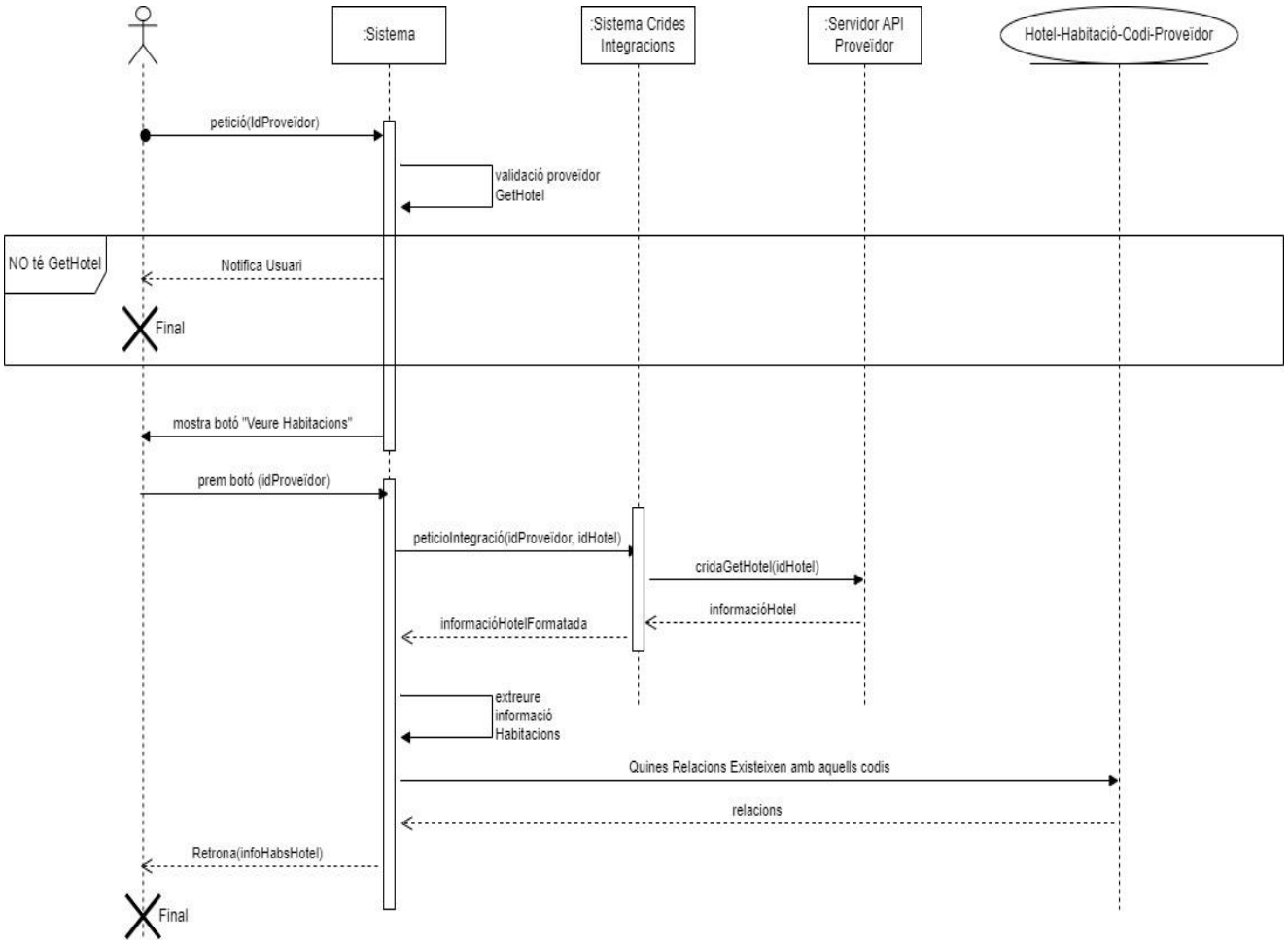
- Dels casos d'ús més complexes, els quals, necessitin una representació gràfica per a una millor comprensió.

4.5.1 Diagrama Seqüències Cd'ú 01. Crear relació



Il·lustració 15 Diagrama de Seqüències Cd'ú 1

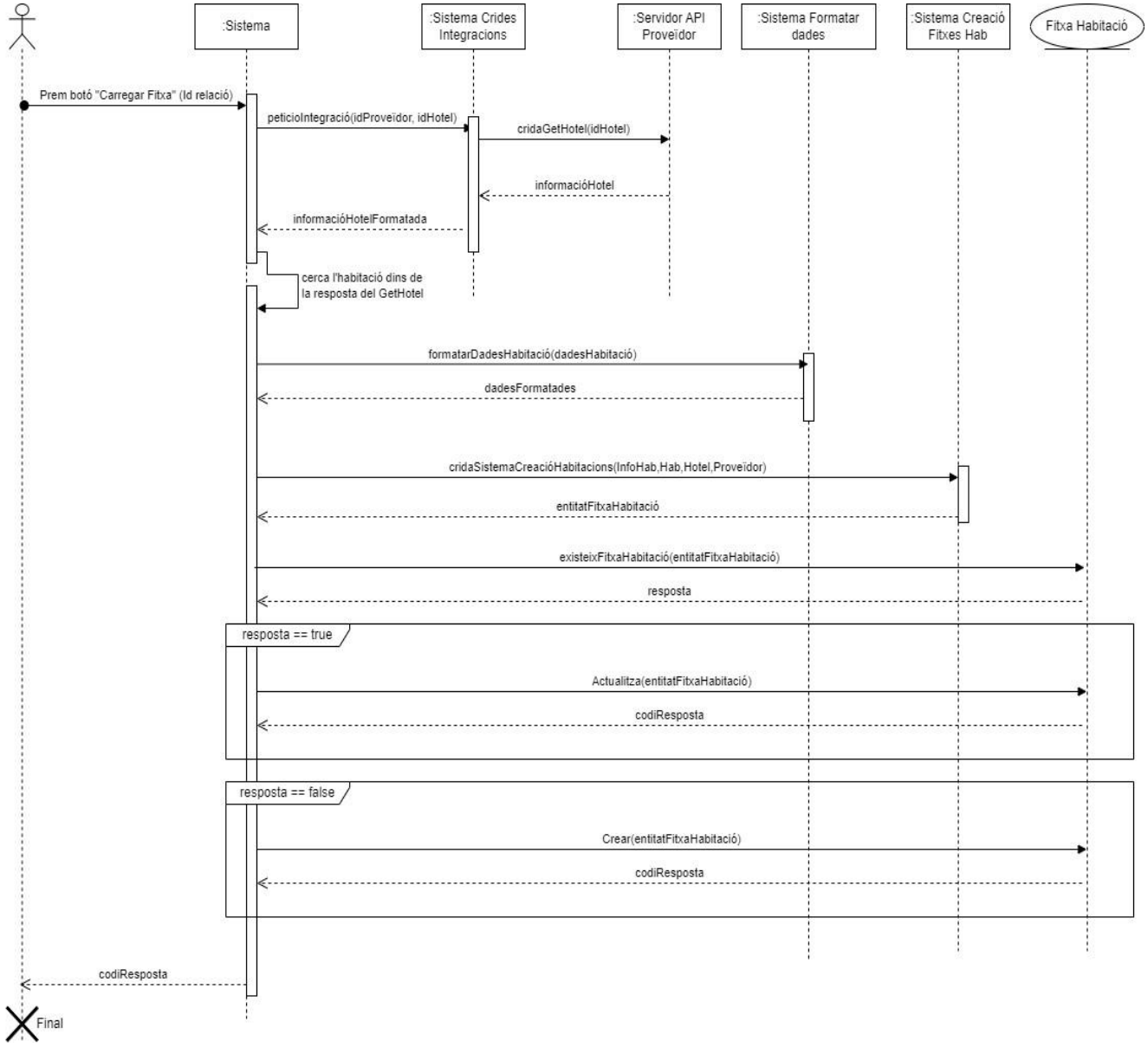
4.5.2 Diagrama Seqüències Cd'ú 06. Rebre els codis d'habitacions dels proveïdors



Il·lustració 16 Diagrama de Seqüències Cd'ú 6

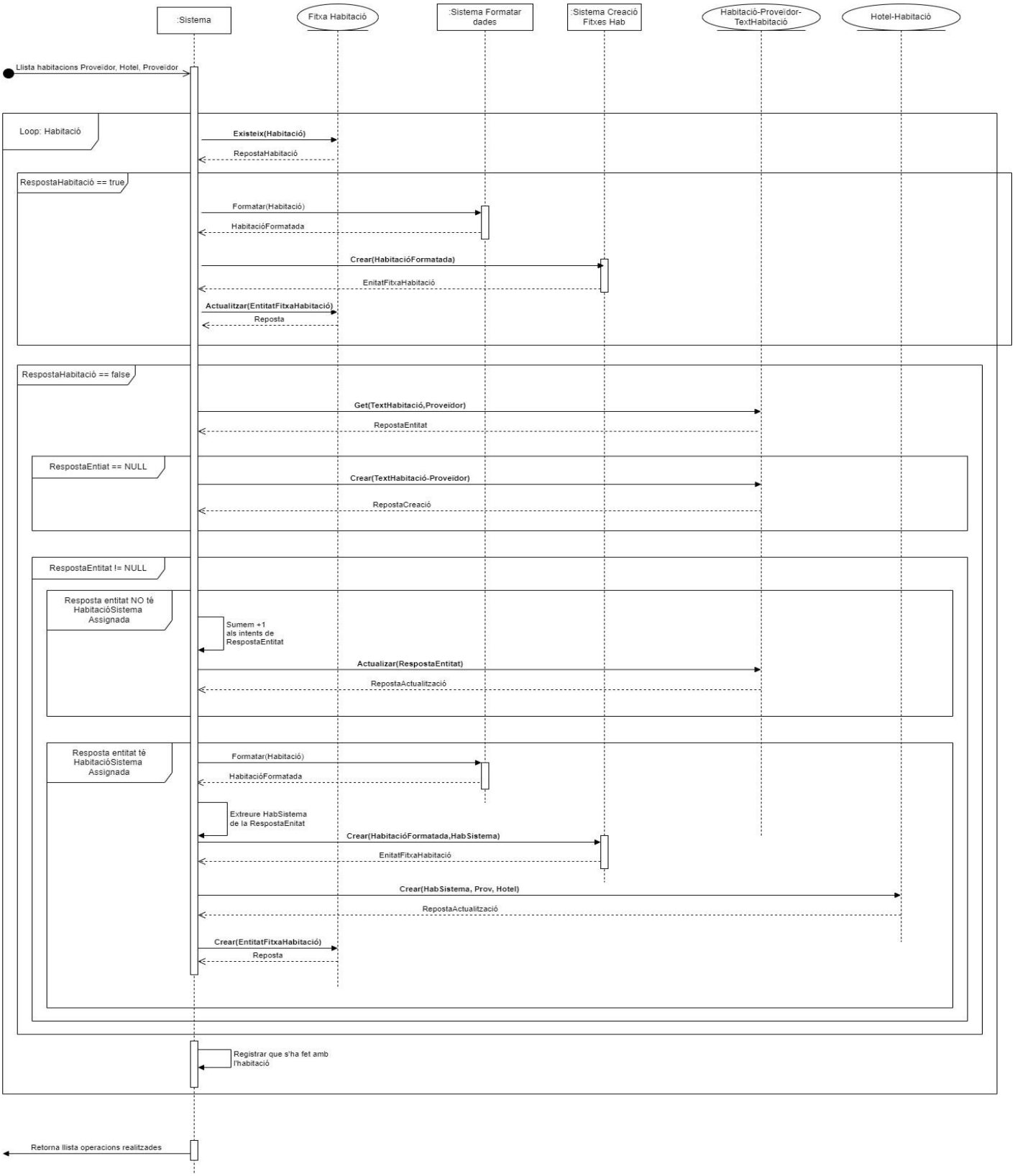
4.5.3 Diagrama Seqüències Cd'ú 07. Crear fitxa d'habitació automàticament

Aquest procediment necessita prèviament el Cas d'ús 2 (Llistar relacions), ja que, perquè l'usuari pugui clicar el botó de "Carregar Fitxa" de l'habitació desitjada, prèviament s'han d'haver llistat les relacions.



Il·lustració 17 Diagrama de Seqüències Cd'ú 7

4.5.4 Diagrama Seqüències Cd'ú 12. Tractar habitacions External-Info-Collector



Il·lustració 18 Diagrama de Seqüències Cd'ú 12

5 Disseny

El disseny de l'aplicació és una etapa crucial en el desenvolupament del projecte, ja que determina la forma en què els usuaris interactuaran amb el sistema i com satisfarà les seves necessitats. En aquest apartat, es descriurà l'arquitectura de l'aplicació, així com els dissenys de les interfícies d'usuari i de les bases de dades.

5.1 Arquitectura general de l'aplicació

A continuació podem veure els diferents elements que componen l'arquitectura de l'aplicació:

1 **Frontend (Client):**

- És la part de l'aplicació amb la qual els usuaris interactuen directament.
- Inclourà les interfícies d'usuari que s'utilitzaran per crear les diferents relacions d'habitacions i crear fitxes.

2 **Backend (Servidor):**

- És la part de l'aplicació que gestiona la lògica de negoci, l'accés a les dades, així com, les validacions d'aquestes.
- Gestionarà les peticions del client, la interacció amb la base de dades i la comunicació amb les APIs dels proveïdors.
- També és on està allotjada tota la lògica de l'External-Info-Collector

3 **Base de Dades:**

- Emmagatzemarà la informació sobre hotels, habitacions, proveïdors i altres dades necessàries per al funcionament de l'aplicació.
- És una part imprescindible per la persistència de les dades del negoci.

4 **API de Proveïdors:**

- L'API del proveïdor és una interfície que permet al nostre sistema comunicar-se amb el sistema del proveïdor extern per accedir a les seves dades i funcionalitats. Aquesta API pot proporcionar diversos punts d'entrada, anomenats "**endpoints**", als quals podem fer crides per obtenir informació específica o realitzar accions determinades El **getHotel**, que

s'ha mencionat en repetides ocasions, és un exemple de crida a l'endpoint d'una Api.

Per fer aquesta aplicació s'ha seguit un **patró d'arquitectura** anomenat **Model Vista Control** (MVC), el qual, és el més utilitzat en el disseny d'aplicacions web.

El MVC, separar les dades i la lògica de negoci de la presentació i el mòdul encarregat de gestionar els esdeveniments i les comunicacions.

El MVC, com el seu nom indica, està format per tres elements:

- Model (**Base de dades)**: Representa les dades i la lògica del negoci de l'aplicació. En aquesta capa, es defineixen les estructures de dades i les operacions que es poden realitzar sobre aquestes dades. El model no sap res sobre la interfície d'usuari ni com es presenten les dades.
- Vista (**Front-End**): En aquest component només hi ha les interfícies d'usuari. La vista s'encarrega de presentar la informació del model, és a dir, mostrar les dades que és sol·liciten al model. Habilita als usuaris a interactuar amb les dades i crear-ne de noves.
- Control (**Back-End)**: El controlador és un component de l'aplicació que actua com a pont entre la vista i el model. És responsable de rebre les interaccions de l'usuari de la vista i de coordinar les accions necessàries per actualitzar el model o la vista en conseqüència. En altres paraules, el controlador interpreta les accions de l'usuari, com ara clics de botons o entrades de formularis, i decideix com processar-les. A continuació, actualitza el model segons sigui necessari i actualitza la vista per reflectir els canvis realitzats en el model.

Alguns dels seus principals avantatges són:

- Menor acoblament
- Major cohesió
- Major flexibilitat i agilitat
- Més claredat de disseny
- Facilita el manteniment
- Major escalabilitat.

5.2 Arquitectura de les “Sub-Aplicacions”

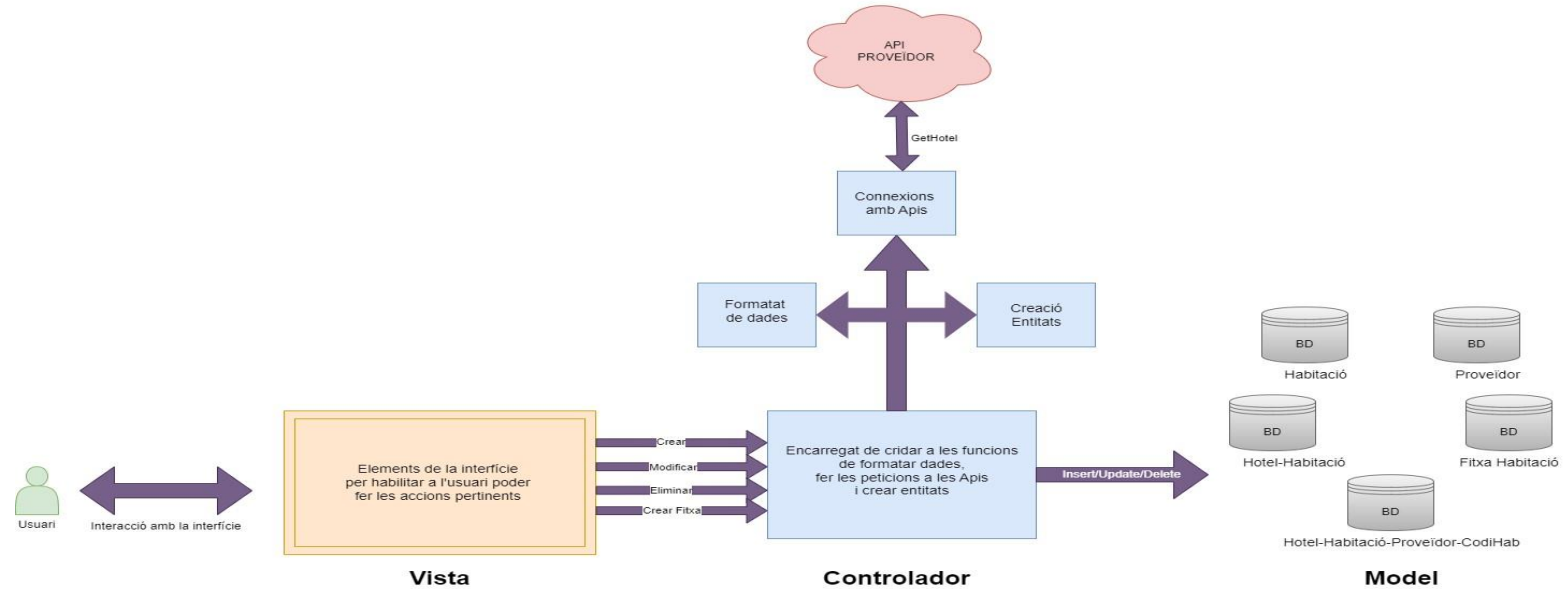
L’aplicació del TFG es podria dividir en 3 “Sub-Aplicacions” que ofereixen 3 funcionalitats diferents. Cal explicar bé les tres parts per separat per poder comprendre bé el sistema en la seva totalitat

5.2.1 App de Gestió relacions Hotel-Habitacions-CodiHab-Proveïdor

El primer sistema que s’havia de desenvolupar era una nova pantalla a l’administrador per poder configurar les habitacions que tenia un hotel en concret i relacionar aquestes habitacions amb ID’s d’habitacions de diferents proveïdors. Amb això si un proveïdor, durant el procés de compra, retorna preus de l’habitació amb l’ID XXX, si tenim aquest ID relacionat amb una habitació del sistema, podrem mostrar la fitxa de l’habitació (si n’hi ha). Per fer aquestes relacions es necessiten saber quins IDs d’habitació té aquell hotel pel proveïdor X, així que seria necessari poder veure els codis que ens retorna el proveïdor per crear les relacions de manera més fàcil.

Per altra banda, ja que en aquesta pantalla tenim relacionada una habitació del proveïdor amb una nostra del sistema, seria útil una funció que pogués fer una petició demanant les característiques d’aquella habitació (serveis, fotos, mida, ocupació, etc.) al proveïdor, i crear una fitxa d’habitació al nostre sistema amb aquestes dades.

Totes aquestes funcions s’han de poder fer des de la mateixa pantalla, i l’arquitectura seria la que podem observar a la il·lustració 19



Il·lustració 19 Esquema Primera App

5.2.2 Arquitectura app Gestió relacions Habitació-Proveïdor-TextHabitació

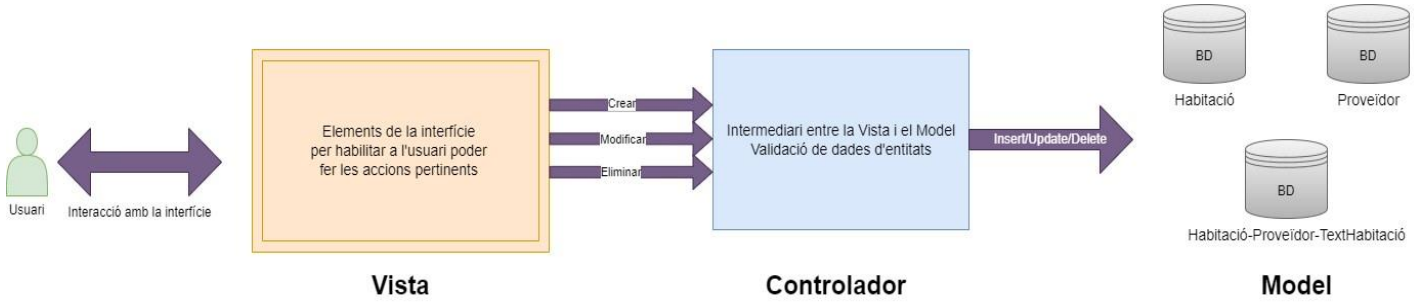
Una vegada amb el sistema de mapeig de codis d'habitació manual desenvolupat, es va pensar com es podria automatitzar aquest sistema per anar relacionant codis de proveïdor i creant fitxes al sistema. Automatitzar aquesta tascà seria un repte pel fet que els proveïdors no retornen un codi de "tipus d'habitació" que sigui genèric.

Els mapejos que havia fet jo utilitzant l'aplicació del punt anterior, els havia realitzat mirant el nom i codi de l'habitació que ens retornava el proveïdor i llavors jo seleccionava quina habitació del nostre sistema seria la més adequada. Per exemple el proveïdor retornava el text "Habitación doble vista lateral mar 2 camas" i jo relacionava el seu id amb l'habitació nostra anomenada "Habitación doble vista mar lateral"

Un factor comú que retornen la majora de proveïdors és el nom de l'habitació, així doncs s'hauria d'utilitzar aquest paràmetre com a referència per fer les relacions d'habitacions del sistema amb les dels proveïdors.

Així doncs, la següent aplicació a desenvolupar, seria una pantalla a l'administrador on, independentment de l'hotel, es pogués fer una relació de NomHabitacióProveïdor - Habitació Sistema – Proveïdor. Des d'aquesta pantalla es poden crear i modificar les relacions que més endavant consultarà el sistema de creació automàtic per saber quina habitació del sistema escollir. A la relació també s'afegiran 2 paràmetres els qual són Ocupació i Comptador d'Intents. L'ocupació s'afegeix perquè els proveïdors, poden retornar el mateix nom d'habitació amb ocupacions diferents, per tant, cal diferenciar la "Habitació Estàndard" de 2 persones i la de 4 persones.

El paràmetre "Comptador" ens permetrà veure quins noms d'habitacions està intentant enllaçar el sistema automàtic, però no troba cap habitació associada a la relació i, per tant, suma +1 al comptador.



Il·lustració 20 Esquema segona App

5.2.3 Arquitectura app Sistema automàtic creació de fitxes d'habitació

Finalment, estudiarem l'aplicació principal la qual és el sistema automàtic que crearà les fitxes d'habitació. Aquesta, és una aplicació basada en **microserveis**.

El disseny del procés d'actualització d'informació d'hotels es basa en la utilització d'un sistema d'orquestració de serveis juntament amb l'ús de microserveis per a la implementació de les diferents funcionalitats. Amb aquest sistema d'orquestració podem crear un flux de treball complex enllaçant els diferents elements.

El sistema s'alimenta en base a una cua d'ids d'hotel, cada un d'aquests, passarà per totes les fases de l'esquema que podem veure a la il·lustració 21.

El primer micro-servei comprova si existeix algun codi GIATA per algun dels dos proveïdors, si és així, fa les peticions, estructura les dades i les emmagatzema per cada proveïdor.

El següent micro-servei recupera les dades d'hotels dels dos proveïdors i les compara entre elles, seleccionant la "millor" segons diferents criteris de comparació (més nombre d'habitacions, de fotos, o establir prioritats entre proveïdors). Emmagatzema les dades del proveïdor "guanyador".

El següent microservei s'encarregarà, per un costat, d'actualitzar tota la informació genèrica de l'hotel, per altre costat, cridarà el servei de "Gestió d'habitacions" per gestionar la llista d'habitacions del proveïdor.

Dins d'aquest Sistema de Gestió d'Habitacions, s'iterarà per cada habitació de la llista que es rep per paràmetre i s'efectuarà una operació o un altre depenent de la informació que tinguem al sistema.

Accions:

- Si l'habitació ja té una fitxa creada al sistema, es formaten les dades de l'habitació del proveïdor i s'actualitza la fitxa amb aquestes.

Else

- Si tenim una relació Habitació-Proveïdor-TextHabitació creada per aquell nom d'habitació i ocupació, amb una habitació del sistema assignada, es formaten les dades de l'habitació del proveïdor, es crea una fitxa d'habitació i s'enllaça a la

relació Hotel-Habitació (Si no existeix, es crea). També es crea una relació Hotel-Habitació-Proveïdor-CodiHabitació per tenir els IDs relacionats.

Else

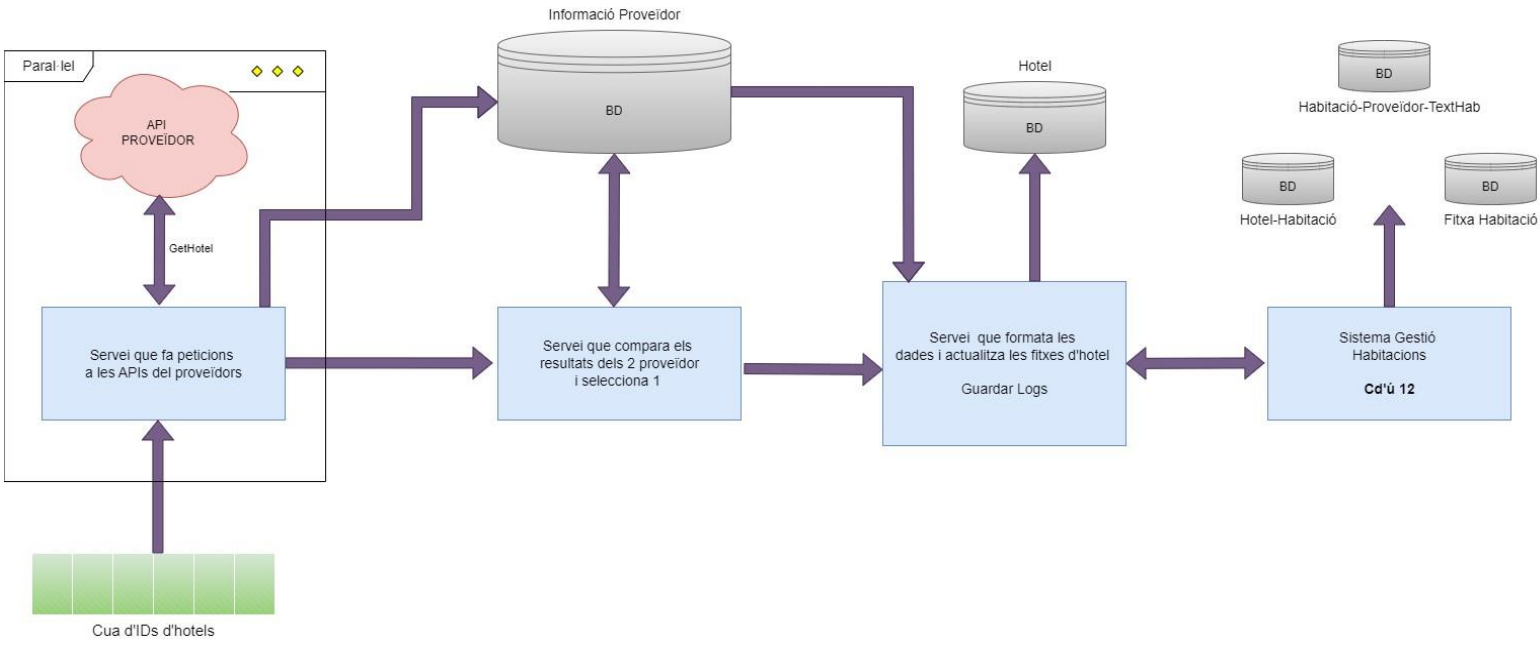
- Si tenim una relació Habitació-Proveïdor-TextHabitació creada per aquell nom d'habitació i ocupació **sense** habitació del sistema assignada, és suma +1 al comptador d'intents.

Else

- Si no tenim cap relació Habitació-Proveïdor-TextHabitació per aquell nom, es crea una nova entitat amb el nom de l'habitació, l'ocupació i el proveïdor, i el comptador d'intents a 1.

Per últim, es retorna amb una llista dels ID's de les habitacions i quina acció s'ha fet amb cada una d'elles. S'emmagatzemen els Logs de les accions de les habitacions i els de la resta del procés. El següent hotel de la cua entra al sistema.

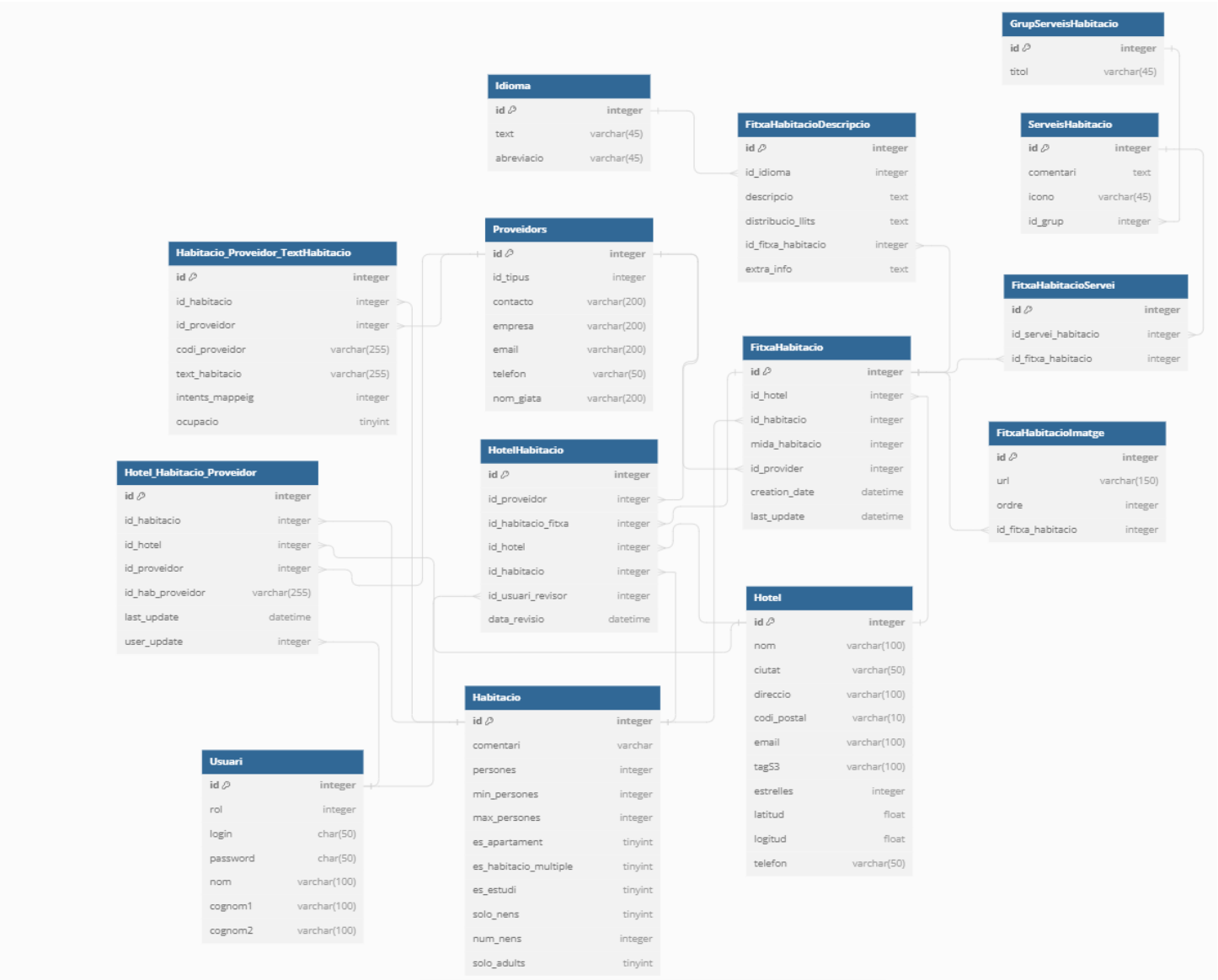
Amb aquest sistema, podem anar actualitzant i creant fitxes d'habitacions, així com, portar un recompte de quins noms d'habitació s'estan intentant enllaçar i no poden. Graciés a l'aplicació de l'apartat **5.2.2**, podrem visualitzar quins noms tenen un major nombre d'intents d'enllaç (ja que es poden ordenar) i assignar manualment a quina habitació del sistema hauria de fer referència



Il·lustració 21 Esquema External-Info-Collector

5.3 Disseny de la Base de dades

A continuació es pot observar l'esquema de les entitats i relacions de la base de dades.



Il·lustració 22 Esquema base de dades

La majoria d'entitats ja estaven creades i, per tant, les vaig aprofitar per al meu sistema. Les noves entitats, que no existien i vaig crear jo són "Habitacio_Proveidor_TextHabitacio" i "Hotel_Habitacio_Proveidor".

A part, també vaig modificar l'entitat FitxaHabitacio, vaig afegir dates de creació i actualització i el proveïdor amb el qual s'havia creat.

Només s'han indicat les dades de les entitats que són importants o que estan relacionades amb l'aplicació. Les entitats Hotel o Usuari per exemple, tenen molts més atributs a base de dades, però, no tenen res a veure amb el sistema de fitxes.

5.3.1 Hotel

Entitat on s'emmagatzema tota la informació relacionada a la ubicació, contacte, puntuació i més d'un hotel. Entre els atributs podem observar "tagS3", aquest camp fa referència al directori on és troben totes les fotografies de l'hotel dins del núvol d'Amazon S3.

5.3.2 Habitació

Aquesta entitat serveix per registrar un tipus d'habitació. En aquesta hi ha el nom (comentari), l'ocupació, la qual pot ser fixa indicada amb el paràmetre "ocupació" o variable amb els atributs "min_persones" i "max_persones". També hi ha diferents atributs per indicar característiques de l'habitació (apartament, estudi, només adults ...).

Com ja he mencionat en apartats anteriors, aquesta entitat és una habitació genèrica, com pot ser "Habitació Doble" i no és una habitació d'un hotel en concret.

5.3.3 HotelHabitació

Relació d'un Hotel amb una Habitació, amb això estem indicant que un hotel en concret té una sèrie d'habitacions dins d'aquest. A diferència de la relació anterior, això ja és una habitació en concret d'un hotel. La relació comptava amb dos paràmetres que no és fan servir a dia d'avui, els quals són "id_proveidor" i "id_usuari_revisor".

Jo he aprofitat el id_proveidor per indicar si la relació s'ha creat per un treballador de forma manual (indicant com a id_proveidor el de la nostra empresa) o bé si la relació s'ha creat amb el meu sistema automàtic (id_proveidor el del proveïdor el qual hàgim agafat les dades).

I el id_usuari_revisor per a quan un treballador vulgui indicar que ha revisat la fitxa d'habitació que s'ha creat i relacionat a aquesta entitat.

5.3.4 FitxaHabitació

Entitat de la fitxa d'habitació, aquesta emmagatzema l'hotel i l'habitació la qual fa referència.

S'han afegit els camps id_provider, creation_date i last_update per veure de quin proveïdor s'han agafat les dades de la fitxa i un control de la data de creació i actualització

Les característiques de la fitxa d'habitació com són les descripcions, imatges i serveis, es troben relacionades a les següents 3 entitats.

5.3.5 FitxaHabitacióImatge

Entitat on es relaciona una fitxa d'habitació amb les imatges que aquesta té. Les imatges de les habitacions, com les dels hotels, també es troben emmagatzemades al núvol S3 d'Amazon, per això la relació té el camp "URL", el qual indica la ubicació de la imatge dins de S3. També hi ha el camp "ordre" per indicar l'ordre d'aparició de les imatges dins de la fitxa.

5.3.6 FitxaHabitacióDescripció

En aquesta entitat trobem els diferents textos que descriuen l'habitació. En total són 3 textos per cada fitxa i aquests, estan traduïts als 7 idiomes de la pàgina web.

Els textos són:

- Descripció: Descripció general de l'habitació en format de text
- Distribució_llits: En aquest text és descriu quins tipus de llits hi ha a l'habitació. Ex: Dues lliteres i un futon doble.
- Extra_info: Informació addicional.

Així doncs cada entrada en aquesta base de dades indicarà per una fitxa d'habitació i un idioma en concret els 3 textos.

5.3.7 FitxaHabitacióServei

Aquesta és una entitat intermèdia per fer una relació **Many To Many** entre fitxa d'habitació i servei d'habitació. Aquesta té id_fitxa_habitacio i id_servei_habitacio per fer referència a les entitats.

5.3.8 ServeisHabitació

En aquesta taula es guarden tots els serveis d'habitació, com poden ser Televisió, Banyera, Nevera, Servei de despertador, etc. Per cada registre s'emmagatzema el nom (comentari), el nom de la icona i el id_grup al qual pertany.

5.3.9 GrupServeisHabitació

Entitat per enregistrar els grups generals de tipus de serveis d'habitació. Aquests poden ser:

- Tipus llit
- *Amenities*
- Serveis de cuina
- Serveis generals
- Lavabo

5.3.10 Idioma

S'emmagatzemen els diferents idiomes amb els quals està disponibles la pàgina web.

Actualment són 7 idiomes: Espanyol, Català, Anglès, Portuguès, Francès, Alemany i Italià.

D'aquests se'n registra el nom sencer i l'abreviació ES, CAT, PT, ENG...

5.3.11 Proveïdor

Aquí registrem tots els proveïdors externs amb els quals treballa l'empresa. Emmagatzemem diferents atributs de contacte així com el "nom_giata" el qual ens serveix per identificar-lo al sistema MultiCode de GIATA explicat anteriorment.

5.3.12 Usuari

Entitat per guardar els treballadors de l'empresa.

5.3.13 Hotel-Habitació-Proveïdor

Entitat fonamental de la primera aplicació. Amb aquesta taula, podem relacionar codis d'habitació de proveïdors amb habitacions del nostre sistema per a un hotel en concret.

L'entitat està formada per un id_habitació del nostre sistema, un id_hotel, id_proveidor i un identificador d'habitació del sistema del proveïdor. També hi ha un atribut per registrar quan s'ha modificat per última vegada el registre i quin usuari l'ha modificat.

Gràcies a aquesta relació podem carregar la fitxa d'una habitació amb el botó de "Carregar Fitxa" de la primera aplicació. També fa possible que durant el procés de compra pugui mostrar la fitxa d'habitació per a producte de proveïdors (sempre que hi hagi la relació feta).

5.3.14 Habitació-Proveïdor-TextHabitació

Entitat encarregada de relacionar noms d'habitacions de diferents proveïdors amb habitacions del nostre sistema indiferentment de l'hotel.

Els atributs que conformen la relació són el id_habitacio del nostre sistema i el id_proveidor, per altre costat, tenim un codi/text habitació del proveïdor; tenim dos atributs per aquesta funció perquè alguns proveïdors a part del nom de l'habitació, també retornen un codi de tipus d'habitació.

També tenim un atribut "ocupació", ja que a vegades els proveïdors retornen el mateix nom d'habitació per habitacions que tenen ocupaciós diferents, per exemple "Habitació Estàndard", no és el mateix una habitació estàndard d'ocupació 1 que d'ocupació 3.

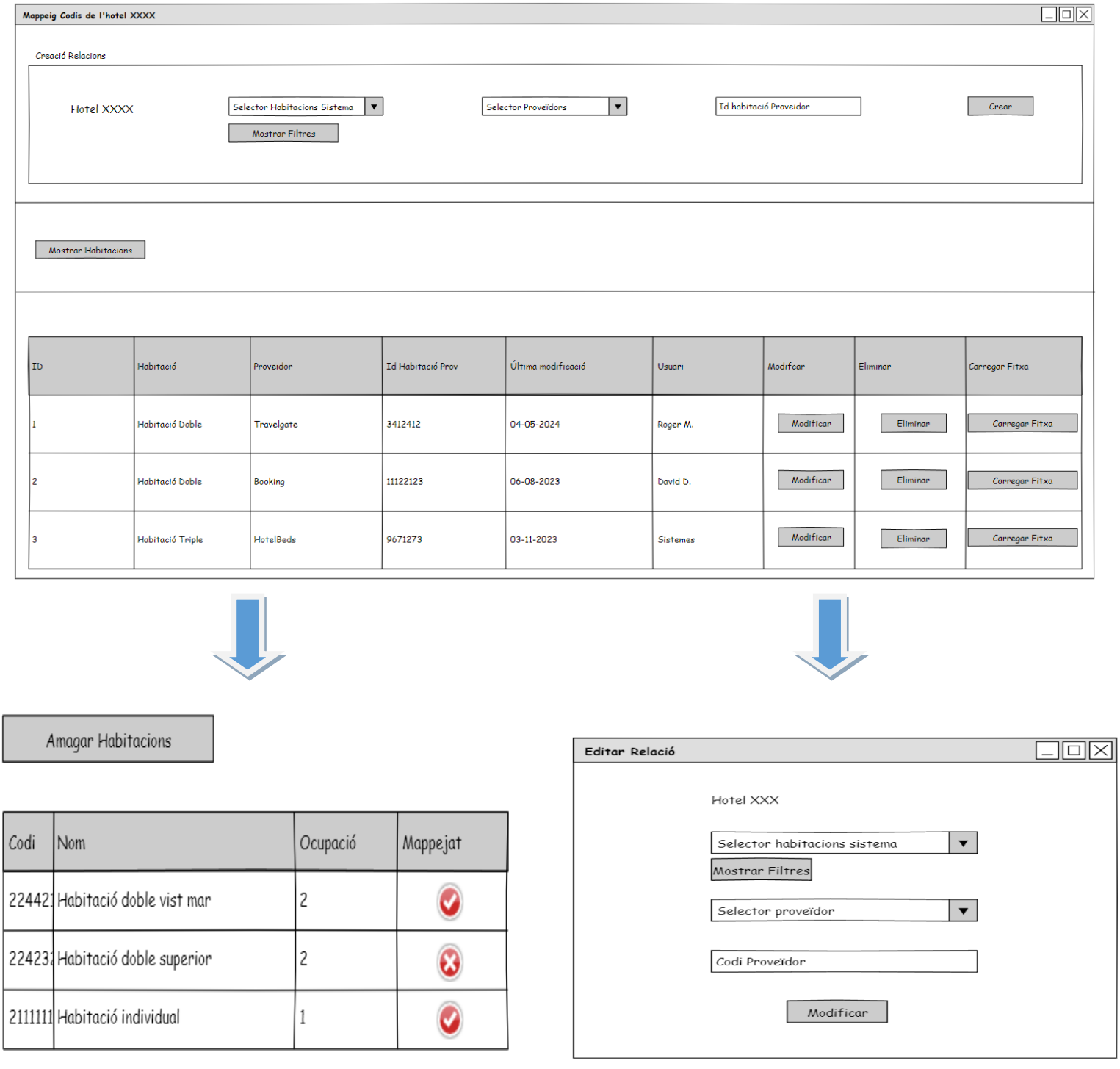
Per últim, tenim l'atribut "intents_mappeig" el qual es un comptador de quantes vegades s'està intentant enllaçar un nom d'habitació d'un proveïdor, però no ho aconsegueix perquè no té cap habitació del sistema relacionada. Amb aquest atribut podem veure quins noms es repeteixen més i afegir nosaltres la relació.

El sistema de creació automàtic de fitxes consulta aquesta entitat per crear les habitacions i les seves fitxes.

5.4 Interfícies d'usuari

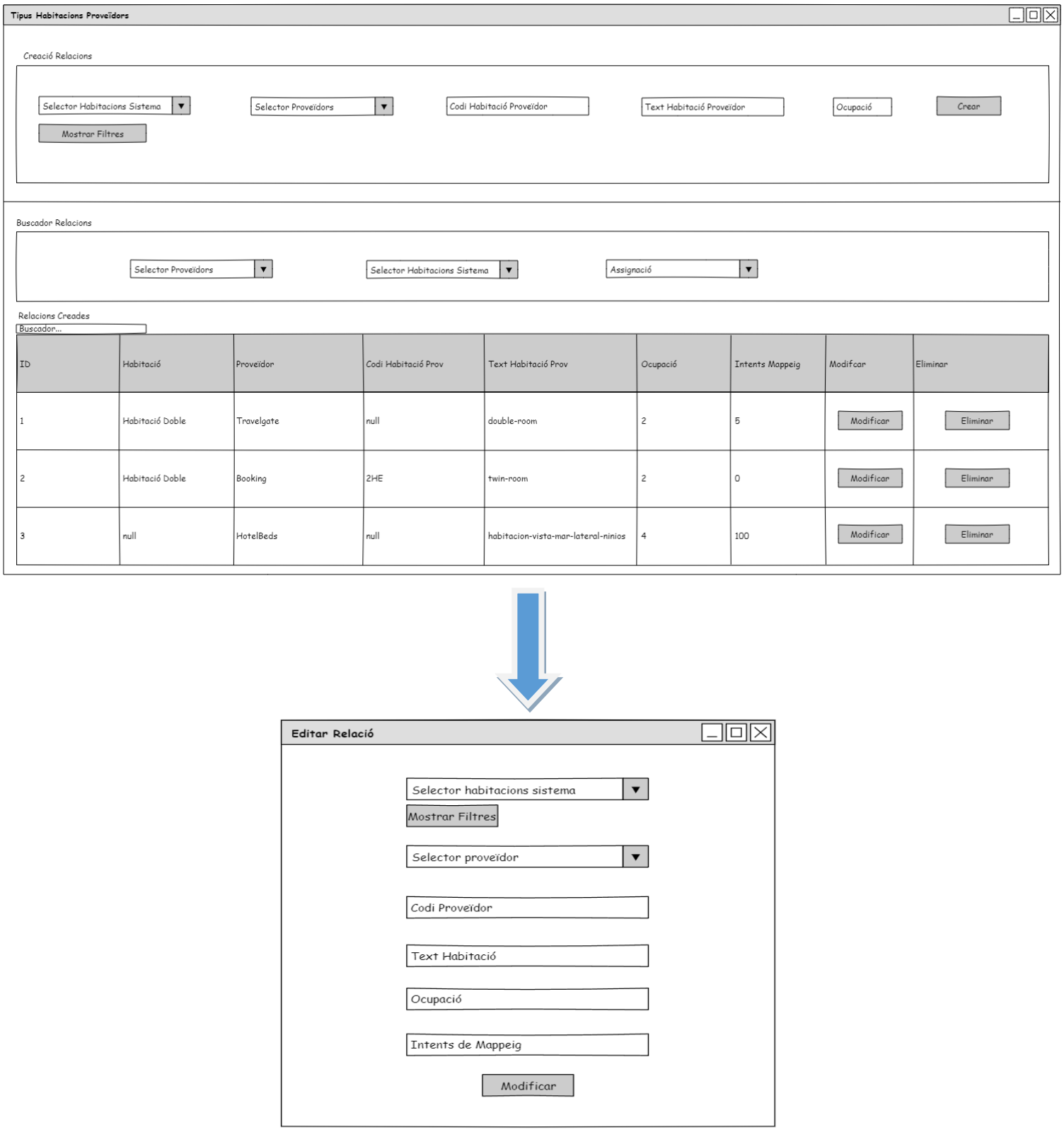
En aquest apartat podrem veure els dissenys de les interfícies d'usuari de les pantalles de l'admin des d'on es poden gestionar les entitats "Hotel-Habitació-Proveïdor" i "Habitació-Proveïdor-TextHabiació".

5.4.1 Interfície gràfica pantalla Hotel-Habitació-Proveïdor



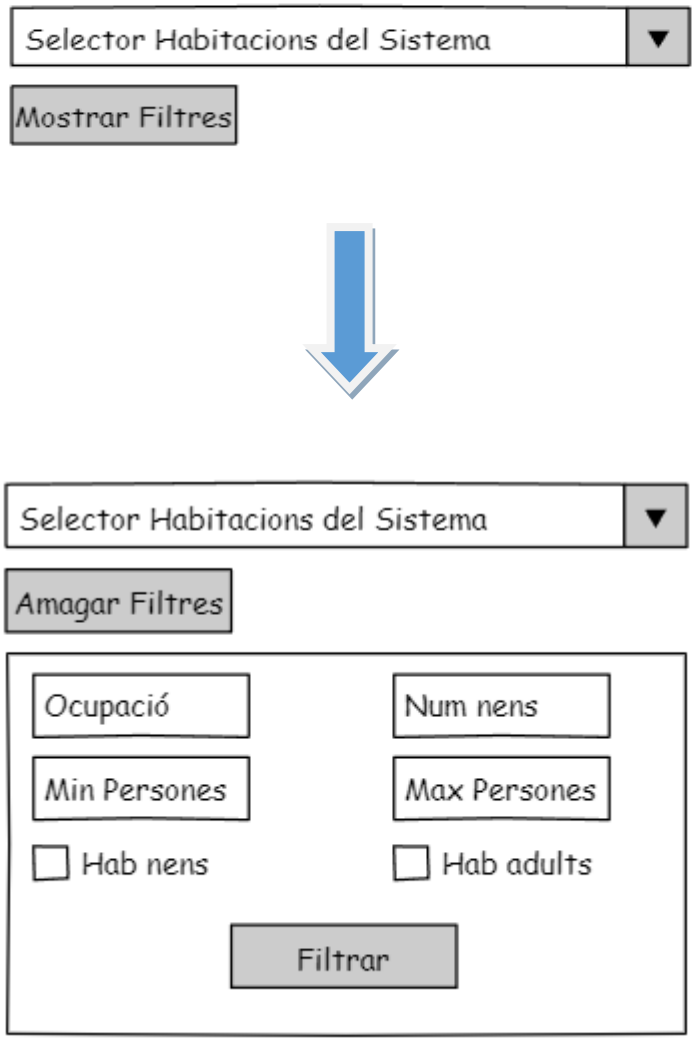
Il·lustració 23 Interfície gràfica primera App

5.4.2 Interfície gràfica pantalla Habitació-Proveïdor-TextHabitació



Il·lustració 24 Interfície gràfica segona App

5.4.3 Interfície gràfica mòdul filtres d'habitació



Il·lustració 25 Interfície gràfica mòdul filtres

6 Implementació

En aquest apartat explicaré com s'ha implementat la solució del dissenys explicats en l'anterior apartat i les tecnologies emprades.

6.1 Tecnologies

6.1.1 Back-End

Les tecnologies utilitzades per implementar el back-end, es a dir, la part del servidor, son:

Php / Symfony

El PHP (*Hypertext Preprocessor*) és un llenguatge de programació d'alt nivell molt popular enfocat al desenvolupament web. El seu tipatge és dinàmic, la qual cosa el fa ser molt flexible, també és un llenguatge senzill i accessible per a programadors amb poca experiència.

Il·lustració 26 Logotip PHP

Juntament amb el PHP es fa servir Symfony, un *framework* de PHP, el qual segueix un patró de disseny Model-Vista-Controlador i està dissenyat per simplificar i accelerar el procés de desenvolupament web proporcionant una estructura organitzada i reutilitzable.

Il·lustració 27 Logotip Symfony

Per gestionar les interaccions del nostre codi amb les entitats de base de dades, utilitzarem Doctrine, una biblioteca de persistència de dades de PHP. Doctrine,

proporciona una capa d'abstracció per interactuar amb la base de dades de manera ràpida, sense haver de fer crides SQL directament.

Il·lustració 28 Logotip Doctrine

Gràcies a la flexibilitat del llenguatge i els diferents frameworks/llibreries que faciliten el desenvolupament de codi, m'ha sigut bastant fàcil adaptar-me a aquestes tecnologies amb les quals està muntada la major part de l'infraestructural de l'empresa i que jo no havia utilitzat mai.

Kotlin

Kotlin és un llenguatge de programació de propòsit general que s'executa a la màquina virtual de Java (JVM), així com també pot compilar-se a JavaScript o a codi natiu. Va ser desenvolupat per JetBrains i llançat el 2011. Kotlin està dissenyat per ser interoperable amb Java, el que significa que pot coexistir i treballar fàcilment amb codi Java existent.

El seu tipatge és estàtic, la qual cosa facilita la detecció i prevenció d'errors i millora el rendiment. També proporciona una millor claredat a l'hora de llegir el codi en comprendre quin tipus de variable estem guardant en tot moment.

Tot i que la major part del codi de l'empresa està fet en PHP, hi ha una nova tendència a implementar les noves integracions amb els proveïdors amb Kotlin, donat que és un llenguatge més segur i més ràpid, característiques molt importants ja que, es fan moltes peticions a les Apis dels proveïdors i les dades que ens envien han de conservar la integritat.

Il·lustració 29 Logotip Kotlin

Amazon Web Services

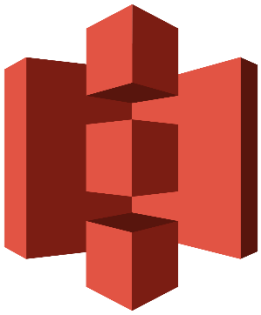
AWS és una plataforma de serveis en el núvol oferida per Amazon.com que proporciona una àmplia gamma de serveis informàtics, incloent-hi emmagatzematge, potència de càlcul, bases de dades, anàlisi, aprenentatge automàtic, Internet de les coses (IoT), seguretat i més. AWS permet a les empreses i als individus utilitzar la infraestructura del núvol d'Amazon per construir i executar aplicacions i serveis de manera flexible i escalable, sense haver de invertir en maquinari físic costós. És una de les plataformes de serveis en el núvol més grans i àmpliament utilitzades al món.

Il·lustració 30 Logotip Amazon Web Services

Dins d'aquesta gran gamma de serveis, els que més he utilitzat han sigut:

-S3: Amazon Simple Storage Service es un sistema d'emmagatzematge multi propòsit basat en el núvol que permet al usuaris emmagatzemar i recuperar una quantitat il·limitada de dades. Entre les seves característiques es troben :

- Facilitat d'us → Té una interfície intuïtiva i una Api molt completa
- Seguretat → Redundància de dades, les dades estan replicades a diferents centres.
- Escalabilitat → És pot oscil·lar el volum de les unitats de manera rapida i còmoda.



Il·lustració 31 Logotip Amazon S3

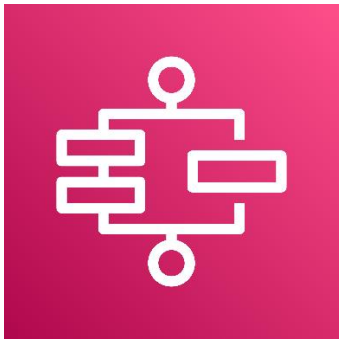
-Lambdes: És un servei *Serverless*, permet executar codi sense haver de preocupar-te d'aprovisionar i gestionar un servidor. És un sistema que s'autogestiona i per tant te la capacitat d'escalar si la necessitat de computació ho demana. Al ser un servei *Serverless* només es pagarà el temps que s'estigui executant, per tant, és molt important tenir un codi òptim i eficient per reduir costos. És una eina molt utilitzat per al processat de dades ja que es pot encadenar amb S3 de manera molt senzilla i el servei només s'executarà quan sigui necessari

Il·lustració 32 Logotip Amazon Lambda

-Step Functions: Un sistema d'orquestració i administració de flux de treball que permet als desenvolupadors coordinar diferents components d'aplicacions distribuïdes així com automatitzar processos i crides a micro-serveis. Per exemple es poden concatenar invocacions de funcions Lambda, executar contenidors d'Amazon ECS, crides a serveis web, etc.

L'avantatge d'aquest sistema és que te una interfície molt visual de com s'executarà cada procés, per tant és pot observar el *work-flow* de manera clara.

He utilitzat i modificat el sistema de StepFunctions i Lambdes ja existent a l'empresa per desenvolupar el Sistema Automàtic de creació de fitxes d'habitació.



Il·lustració 33 Logotip Amazon Step Functions

6.1.2 Front-End

Les tecnologies per implementar el front-end, és a dir, la part del client són les següents:

HTML (HyperText Markup Language)

L'HTML és el llenguatge de marcatge utilitzat per estructurar el contingut de les pàgines web. Bàsicament, és un conjunt d'etiquetes que el navegador interpreta i s'utilitzen per definir el text i altres elements que compondran l'esquelet de la pàgina web , com ara imatges, llistes, taules, vídeos, etc.

El llenguatge HTML serveix per descriure l'estructura bàsica d'una pàgina i organitzar la forma en què es mostrarà el seu contingut.

Il·lustració 34 Logotip HTML

CSS (Cascading Style Shets)

Bàsicament, és el llenguatge que gestiona l'estil, disseny i presentació dels components visuals

de la pàgina web. Funciona juntament amb l'HTML.

Gràcies al CSS es poden crear les experiències d'usuari satisfactòries i generar interfícies intuïtives.

A l'empresa existeixen classes "*customitzades*" de CSS per tal de no repetir codi innecessàriament. Aquestes classes s'anomenen VPT *Assets* (Viajes Para Ti), amb això també s'aconsegueix que les diferents pantalles de l'administrador i de la web pública disposin dels mateixos estils d'elements i, per tant, una interfície més elaborada, neta i intuïtiva.

Il·lustració 35 Logotip CSS

JavaScript (JS)

JavaScript és un llenguatge de programació d'alt nivell, interpretat i dinàmic, que s'utilitza principalment per crear interactivitat dins de les pàgines web. Va ser desenvolupat inicialment per Netscape i és un dels tres llenguatges bàsics de la web, juntament amb HTML i CSS, sent JS, l'únic llenguatge de programació com a tal dels tres.

Gràcies al JS, he pogut afegir dinamisme i interactivitat a les pantalles de l'administrador, utilitzant aquest per fer crides "**AJAX**" (Asynchronous JavaScript and

XML), les quals, permeten fer crides al backend, en el meu cas al Controlador de PHP, i actualitzar diferents elements de la pàgina web sense haver de recarregar la pàgina al complet.

Il·lustració 36 Logotip Javascript

Twig

Twig és un motor de plantilles per PHP que permet als desenvolupadors crear i gestionar la presentació de les seves aplicacions de manera eficient i estructurada. Va ser creat per Fabien Potencier, el fundador de Symfony.

Característiques principals de Twig:

1. **Sintaxi clara i llegible**: Twig utilitza una sintaxi intuïtiva i fàcil d'entendre, la qual facilita la creació i el manteniment de les plantilles.
2. **Separació de la lògica i la presentació**: Twig fomenta una clara separació entre la lògica de l'aplicació (PHP) i la presentació (HTML), el que ajuda a mantenir un codi més net i organitzat.
3. **Reutilització de codi**: Permet la creació de plantilles modulars i la reutilització de fragments de codi en múltiples parts d'una aplicació.

Il·lustració 37 Logotip TWIG

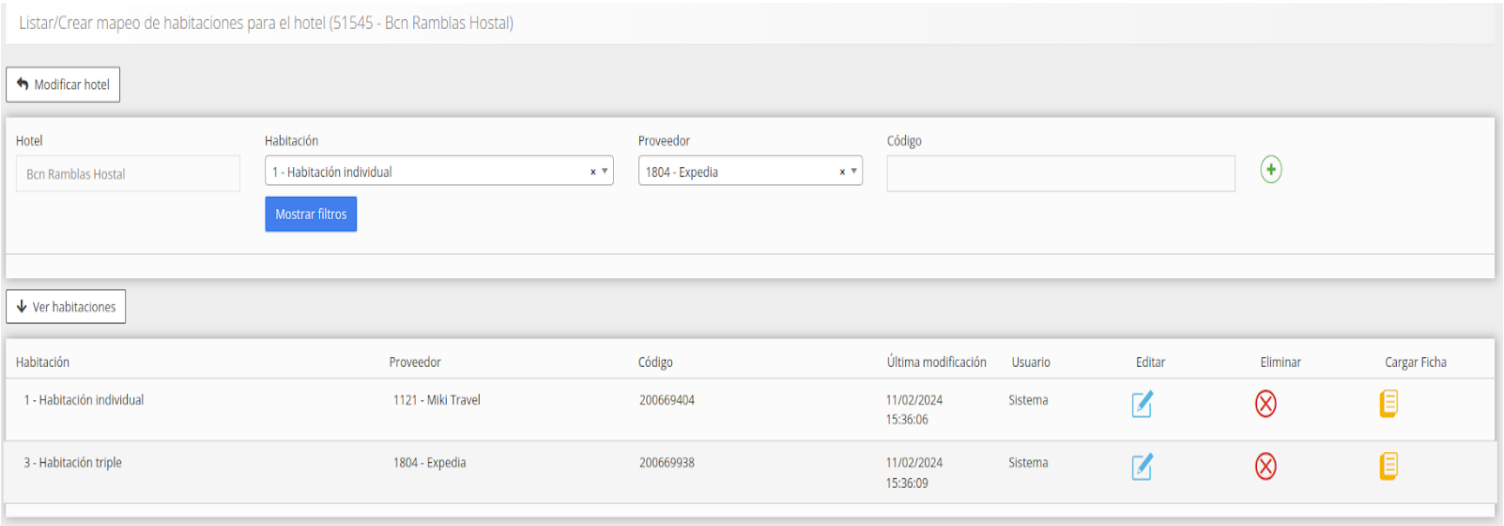
6.2 Implementació de les Funcionalitats

A continuació s'il·lustraran imatges on es veu el resultat final de les diferents aplicacions juntament amb l'explicació de com s'ha fet la implementació.

6.2.1 Relacions Hotel-Habitació-Proveïdor

Per començar mostraré la primera aplicació que vaig desenvolupar, el sistema per poder relacionar IDs d'habitacions de proveïdors amb habitacions del nostre sistema sobre un hotel concret.

A la il·lustració 38 es pot observar el resultat final de la pantalla, la part superior està orientada a la creació de noves relacions. Per crear una nova relació, s'ha de seleccionar una de les habitacions del sistema, un proveïdor i introduir un Id d'habitació del proveïdor.

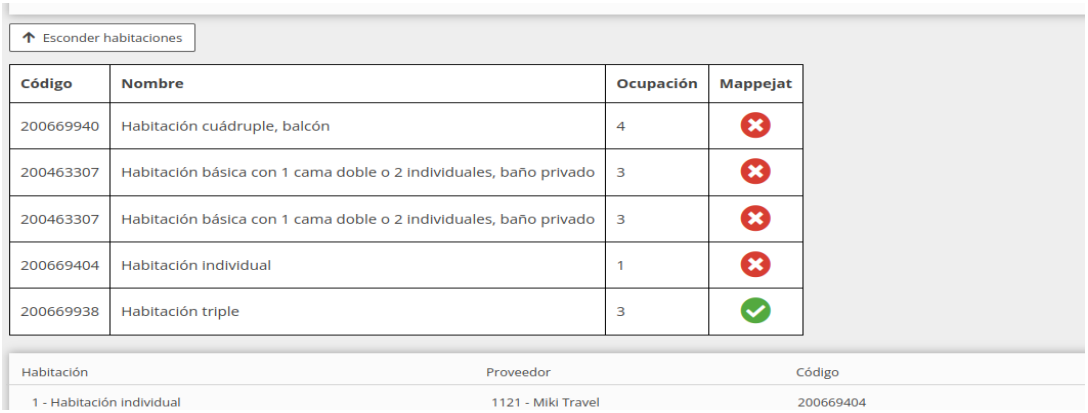


Il·lustració 38 Pantalla Administrador HotelRoomProvider

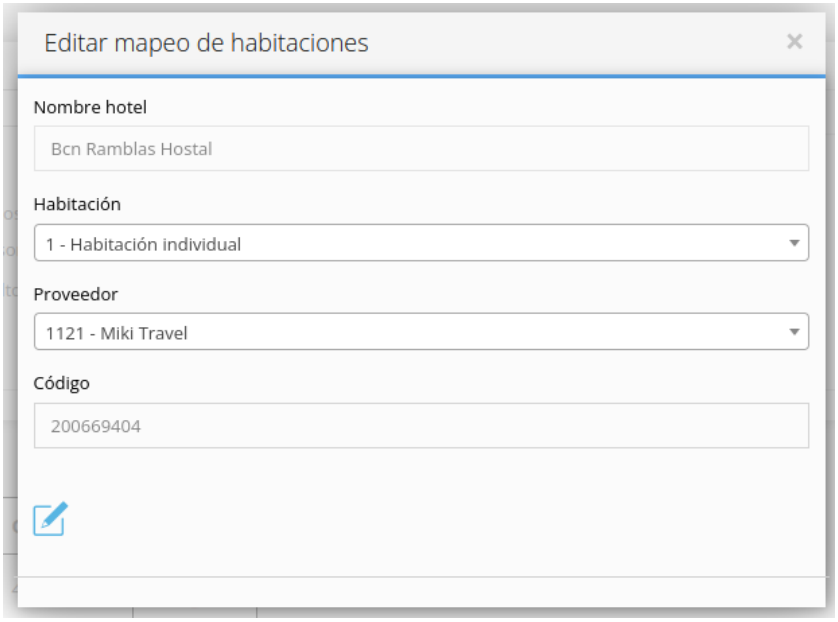
Per poder veure els IDs d'habitació que té disponible un proveïdor sobre aquell hotel en concret, vaig fer la funcionalitat de "Ver Habitaciones". Aquest botó, depenent del

proveïdor que estigui seleccionat al selector de Proveïdor, farà una petició a la seva API i mostra una taula amb les habitacions que té registrades en aquell hotel com podem veure a la il·lustració 39.

La part de baix de la pàgina és una taula per gestionar les relacions creades per aquell hotel, des d'on podrem modificar/eliminar les relacions (a la il·lustració 40 podem veure el *modal* d'edició) i carregar un fitxa de manera automàtica sobre aquella habitació.



Il·lustració 39 Taula habitacions proveïdor



Il·lustració 40 Modal d'edició de relació

Els diferents botons de la pantalla (Crear, Eliminar, Filtrar...) estan implementats amb JavaScript els quals, fan peticions Ajax al controlador de php "HotelRoomProviderController".

Amb això aconseguim tractar i emmagatzemar les dades, amb les quals l'usuari interactua, d'una manera fàcil i dinàmica.

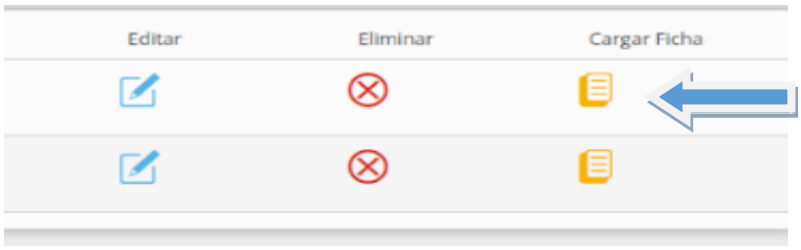
Per als casos de Crear, Eliminar, Modificar i Filtrar, la part del servidor (back-end) és limita a rebre les dades del front-end, validar-les i registrar-les a la base de dades amb l'ajuda dels "*Repository Interfaces*" de Doctrine. Aquestes, són interfícies que faciliten la interacció amb les entitats que tenim a base de dades, incorporant una sèrie d'atributs i mètodes per simplificar la connexió (en comptes de SQL directament).

Un altre gran avantatge que ens aporta Symfony, és la "**Injecció de Dependències**", aquesta permet crear dependències com poden ser objectes de classe o serveis i poder "injectar-los" en un altre classe sense haver de crear-los internament. Amb això s'aconsegueix un gran desacoblament i modularitat al codi.

Amb les accions de Carregar Fitxa i Mostrar habitacions, el back-end es complica una mica més, ara a part de registrar la informació a base de dades, s'ha de connectar a una API de proveïdor externa i fer una petició. Per explicar-ho en detall, utilitzaré l'exemple de Carregar Fitxa.

6.2.2 Carregar fitxa d'habitació

La finalitat d'aquesta funció és, a partir d'una relació Hotel-Room-Provider, fer una petició al proveïdor, i amb la informació retornada per aquest, crear una fitxa d'habitació.



Il·lustració 41 Botó de crear fitxa d'habitació automàtic

1- Per començar quan l'usuari prem el botó de "Carregar Fitxa" d'una de les relacions de la taula de la pantalla (Il·lustració 41), es llança un Ajax al endpoint "ChargeRoomDataSheet" del controller amb una "*Request*" que conté l'id de la relació.

2- Al controller recuperem l'id de la relació i amb les entitats de doctrine, recuperem tots els atributs de la relació (hotel, proveïdor, habitació i **codi hab proveïdor**).

3- Es fa una petició cURL a un dels endpoints del projecte IntegrationsPull, el qual és l'encarregat de fer les comunicacions amb les APIs dels proveïdors. Al cURL, indiquem el id del proveïdor i el codi de l'hotel que volem preguntar.

4- Es fa la petició al endpoint encarregat de retornar tota la informació d'un hotel a l'Api del proveïdor corresponent.

5- Retornem al controller de php un Json amb totes les dades de l'hotel en qüestió. Descodifiquem el json amb la funció "json_decode()" de la llibreria pròpia de php, i ara que tenim tot el Json en un Array Associatiu, extrèiem la part que fa referència a les habitacions.

6- Una vegada tenim totes les habitacions, fem una cerca per buscar l'habitació que tingui com a id el codi d'habitació que tenim a la relació HotelRoomProvider (PAS 2).

7- Quan tenim l'habitació, formatem les dades al format corresponent per al servei de creació de fitxes i l'executem perquè aquest emmagatzemi tota la informació de l'habitació a les entitats corresponents de bases de dades que conformen la fitxa d'habitació (serveis, imatges, textos)

El format de les dades que ens envien els proveïdors amb la petició “getHotel”, varia segons el proveïdor, a continuació mostraré una possible resposta a la il·lustració 42. Podem observar que és un Json, on s’indica la informació general de l’hotel, direcció, nom, telèfon, serveis i fotos.

```
{
   "getHotel":{
      "items":1,
      "result":[
         {
            "name":"Hotel Bonic",
            "code":"234432",
            "info":{
               "category":"1.0",
               "address":{
                  "street":"Ctra general de libertad",
                  "cp":"43001",
                  "city":"La Massana",
                  "infoExtra":{
                     "coordinates":{
                        "altitude":41.575894,
                        "longitude":2.518983
                     },
                     "description":[
                        "Hotel con vistas a la montaña"
                     ],
                     "email":null,
                     "tlf":"34-976 101 311",
                     "checkin":"15:00",
                     "checkout":"10:00",
                     "services":[
                        {
                           "id":"1073743",
                           "name":"Aparcamiento al aire libre"
                        }
                     ],
                     "photos":[
                        {
                           "caption":"Imagen principal",
                           "hero_image":"true",
                           "url":"https://i.url.jpg"
                        }
                     ]
                  ],
                  "roomDetails":[
                     {
```


Il·lustració 42 Resposta Json del getHotel (info del Hotel)

Al final, podem observar el paràmetre “roomDetails”, aquest, és una llista on hi ha totes les habitacions que existeixen en aquell hotel, a la il·lustració 43 podem observar l’estructura d’una habitació.

```
"roomDetails":[
   {
      "name":"Habitación doble",
      "description":"Hotel muy bonito con vistas al mar"
         ""occupation"":3,
         ""dimensions"":{
            ""square_meters"":"15",
            ""square_feet"":"161"
         },
         ""services"":[
            {
               "10737430":""Toallas""
            }
         ],
         ""images"":[
            {
               ""caption"":""Habitación"",
               ""hero_image"":"true",
               ""url"":""https":"
            }
         ],
         ""bedTypes"":[
            {
               "37341":"2 camas individuales"
            },
            {
               "37316":"1 cama doble"
            }
         ],
         ""views"":[

         ],
         ""roomId"":"21514513"
```


Il·lustració 43 Resposta Json del getHotel (info de una Habitació)

Una habitació conté un nom, ocupació, dimensions (m^2), llista de serveis d'habitació, llista d'imatges, tipus de llits i un Id.

Com cada proveïdor retorna les dades d'un format diferent vaig haver de crear una sèrie de classes "Formatters" per tal d'adaptar el format que ens retornava cada integració a un genèric per poder tractar les dades per igual. Per fer això vaig crear una classe pare "**AbstractDefaultDataSheetFormatter**" la qual implementava una "**DefaultDataSheetFormatterInterface**" on estava indicat el mètode per formatar les dades "**dataToCreateRequest**".

Per cada proveïdor vaig crear un fill per poder implementar el codi de formatat corresponent. Dins d'aquest *formatter* fill, agafava les dates del Json de habitacions del proveïdor i ho estructurava amb un format genèric. A part, també recorria tots els IDs serveis d'habitació que ens enviaven, i buscava la relació amb els nostres serveis d'habitació, ja que, cada proveïdor fa servir els seus propis IDs.

A més a més, com la web està en 6 idiomes diferents, la informació de les fitxes d'habitació hauria d'estar traduïda als 6 idiomes. Així que vaig implementar una funció a la classe pare AbstractDefaultDataSheetFormatter, una funció per traduir els textos de les distribucions de llits. Per això, vaig utilitzar el servei de traduccions de Google Services, el qual disposava d'un Endpoint per fer traduccions. Llavors, a la vegada que formatava les dades, també afegia les traduccions.

Per poder buscar la relació de IDs de serveis d'habitació hi ha una taula a base de dades que relaciona un Proveïdor – Un ID de servei d'habitació del proveïdor – Un ID d'habitació del nostre sistema. Prèviament, vaig fer peticions als endpoints del proveïdor que retornaven tots els serveis d'habitació disponibles i els vaig mappejar a la base de dades.

Inicialment, abans que comences a desenvolupar el meu TFG, les fitxes només es podien crear de forma manual des d'un *form* de PHP al admin on el treballador anava omplint els camps de manera manual. Llavors es recollien les dades del formulari i es cridava el servei de creació de fitxes anomenat "**HotelRoomDataSheetCreateUseCase**" el qual rebia per paràmetre un *formData* on estaven totes les variables del formulari i emmagatzemava tota la informació a les entitats de base de dades corresponents i les imatges al Servei de S3 d'Amazon.

Jo per poder reutilitzar part del codi del servei de creació de fitxes, vaig modificar el paràmetre que rebia com argument, ara en comptes d'un **formData**, rebria un **DTO** (Data Transfer Object) que vaig crear anomenat "**HotelRoomDataSheetCreateRequest**" on hi ha tots els atributs per la creació de fitxa d'habitació, il·lustració 44.

```
15 usages   Roger Muntane +2 *
class HotelRoomDataSheetCreateRequest
{
    2 usages
    private HotelRoomDataSheet $hotelRoomDataSheet;
    2 usages
    private int $roomSize;
    2 usages
    private array $bedDistribution;
    2 usages
    private string $extraInfo;
    2 usages
    private string $description;
    2 usages
    private array $services;
    2 usages
    private array $images;
    2 usages
    private array $imageOrder;
```

Il·lustració 44 DTO HotelRoomDataSheetCreateRequest

Com que el formulari de l'admin per crear i editar fitxes manualment havia de seguir funcionant, vaig haver d'adaptar les dades del formulari al nou objecte, per fer això vaig crear un nou fill al "**AbstractDefaultDataSheetFormatter**" anomenat "AdminDataSheetFormatter" per tal formatar les dades.

Per últim, vaig crear una classe anomenada "**HotelRoomDataSheetFormatterFactory**" amb un mètode "create", el qual depenent del ID que li enviaves et retornava un dels formatters fills (Patró de disseny **Simple Factory**), el ID en qüestió seria el del proveïdor i en el cas del admin el ID de l'empresa d'esquiades.

Amb aquest codi, indiferentment del proveïdor que ens enviï les dades, o de si un usuari les ha introduït manualment, podem generar l'objecte "**HotelRoomDataSheetCreateRequest"** per poder cridar el servei de creació de fitxes i crear una fitxa d'habitació al sistema.

6.2.3 Relacions Habitació-Proveïdor-TextHabitació

La segona aplicació també consisteix en un controlador php anomenat "RoomTypeController" i unes plantilles de Twig que contenen html css i javascript.

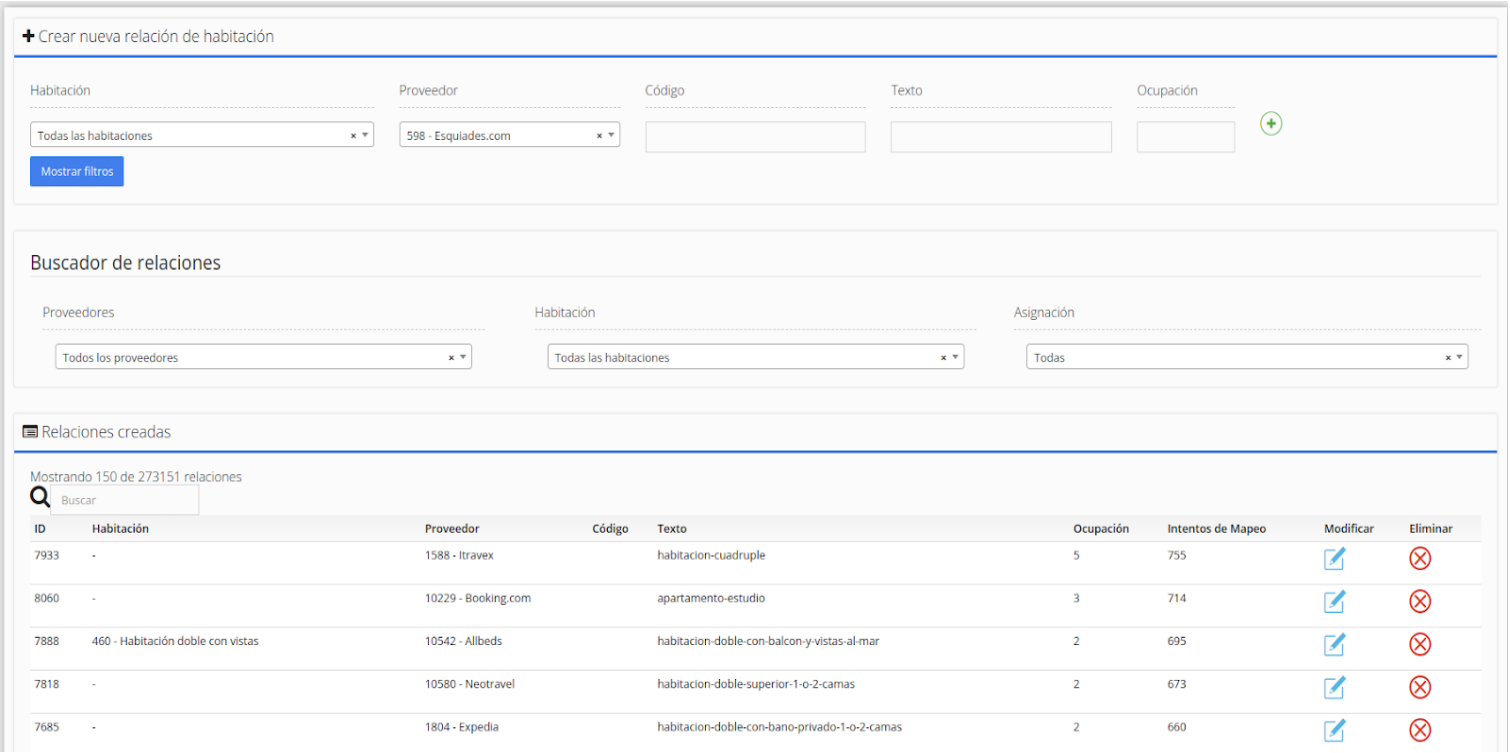
Quan un usuari accedeix a la pantalla, s'executa la primera funció del controller anomenada "indexAction", com podem veure a la il·lustració 45, la qual recupera de les entitats de base de dades amb doctrine el continguts dels selectors i les relacions creades. Amb totes les dades de BD carregades, renderitza la plantilla de Twig on es troba l'estructura principal de la pàgina.

```
Roger Muntane
public function indexAction(Request $request)
{
    $allRooms = $this->roomRepository->findAll();
    $count = $this->roomTypeProviderRepository->countAll();
    $roomProviderRelation = $this->roomTypeProviderRepository->findBy([],['mappingAttempts' => 'DESC'],self::QUERY_LIMIT);
    $allProviders = $this->integrationOptionsPullRepository->findAll();


    return $this->render( view: '@pages_admin_path/roomTypeProvider/indexRoomType',
        [
            'rooms' => $allRooms,
            'integrationProviders' => $allProviders,
            'roomsRelation' => $roomProviderRelation,
            'totalCount' => $count,
            'count' => count($roomProviderRelation)
        ]);
}
```

Il·lustració 45 Funció índex del RoomTypeController

Una vegada es renderitza la plantilla de Twig, l'usuari ja pot veure i interaccionar amb la pantalla que es pot observar a la il·lustració 46



Il·lustració 46 Pantalla de l'Admin de RoomTypeProvider

El disseny és molt semblant al de la primera aplicació, la part superior està orientada a la creació de noves relacions. A l'apartat del selector d'habitacions hi ha el filtre que vaig crear a la primera pantalla, originalment, el vaig ubicar al Twig de la primera pantalla, però quan vaig començar a programar aquesta interfície, vaig extreure tot el codi relacionat amb els filtres (Html, Css i Js) i vaig crear una nova plantilla Twig per tal de poder-lo reutilitzar.

A la part del mig vaig crear 3 selectors a mode de filtres per poder cercar les relacions en més precisió. Les relacions es poden filtrar per Proveïdor, Habitació del sistema i si ja tenen una habitació assignada o no.

La part inferior de la pàgina, està orientada a la visualització i edició de relacions, aquesta vegada, però, vaig crear una nova plantilla Twig on es trobava tot el **body** de la taula de relacions amb l'objectiu de facilitar el filtratge. En comptes de tenir tot el codi del body al Twig principal, ara, fem un @include de la nova plantilla i passem com a paràmetre la llista de relacions que ens ha retornat el Controller com podem observar en la il·lustració 47.

Dins de la plantilla de la taula s'itera per cada element de $roomsRelation i es va imprimint cada atribut de la taula i generant els botons d'Editar i Eliminar.

```
<table id="relation-table" class="table table-condensed">
    <thead>
    <tr>
        <th>ID</th>
        <th>Habitación</th>
        <th>Proveedor</th>
        <th>Código</th>
        <th>Texto</th>
        <th>Ocupación</th>
        <th>Intentos de Mapeo</th>
        <th>Modificar</th>
        <th>Eliminar</th>
    </tr>
    </thead>
    <tbody id="relations-body-content">
    {% include '@pages_admin_path/roomTypeProvider/relationsTable.html.twig' with {'roomsRelation':roomsRelation} %}
    </tbody>
</table>
```

Il·lustració 47 Include d'una plantilla Twig

Llavors quan es modifica un dels tres selectors de filtratge, que es poden veure a la imatge de la pantalla completa (Fig X), el JavaScript detecta aquest canvi i fa una petició a una funció del Controller de PHP anomenada FilterAction (Fig X), la qual rep el contingut dels selectors, del atribut Request, i fa una cerca personalitzada a la base de dades utilitzant el Repository Interface de l'entitat.

Quan tenim el resultat de filtratge al Controller, retornem un render de la plantilla de la taula d'entitats juntament amb una variable amb les relacions. Al JavaScript fem un "load" sobre l'element HTML el qual conté el include de la plantilla Twig, i per tant, actualitzem el seu contingut amb els resultats de cerca.

```
$("select#roomFilterSelector, select#providerFilterSelector, select#roomAssigned").on('change',function() {
    let providerId = $("#providerFilterSelector").val();
    let roomId = $("#roomFilterSelector").val();
    let assignedRoom = $("#roomAssigned").val();
    var data = {providerId,roomId,assignedRoom};

    $('#relations-body-content').load("{{ path('admin.room.room_type_filter') }}",data);

});
```

Il·lustració 48 Codi Javascript per filtrar relacions

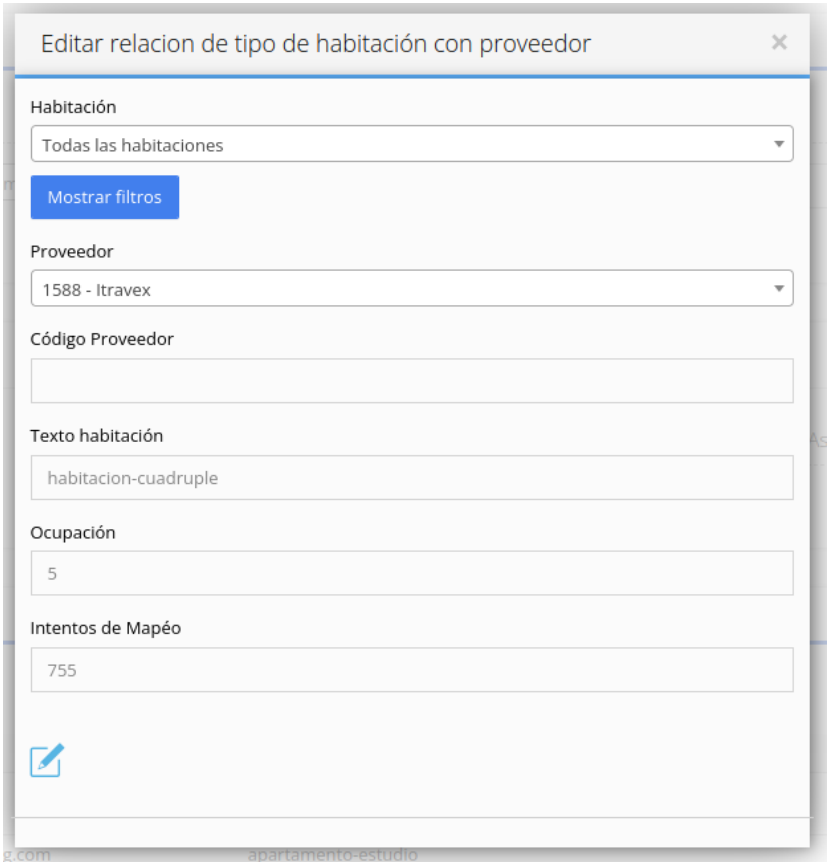
La resta d'accions de la pàgina són Crear i Eliminar, les quals són com les de la primera aplicació, bàsicament, amb els diferents botons fem crides Ajax als diferents mètodes del Controller per crear i esborrar les entitats de base de dades.

Per el cas de modificar, el botó té un atribut href (Il·lustració 49), on el *path* apunta a la ruta del controller de “ModifyIndex”, el qual ens carrega la informació dels selectors i renderitza la template del modal.

```
<td name="providerCodeRow"> {{ roomRelation.providerCode }} </td>
<td name="roomTextRow"> {{ roomRelation.roomText }} </td>
<td name="occupationRow"> {{ roomRelation.occupation }} </td>
<td name="mappingAttemptsRow"> {{ roomRelation.mappingAttempts }}</td>
<td><a href="{{ path('/room/room_type_modify',{
        'idRoomTypeProvider' : roomRelation.id
    }) }}" class="pointer launch-modal"><i
                class="ci ci-edit no-underline fa-2x info"></i></a></td>
<td><a data-relation-id="{{ roomRelation.id }}" name="deleteRelation"
       class="pointer no-underline"><i
                class="ci ci-times-circle no-underline fa-2x danger"></i></a>
</td>
```

Il·lustració 49 Codi per carregar el Modal d'edició

A continuació el resultat de com es mostra el modal d’edició de les relacions.



Il·lustració 50 Modal edició de relacions

Sempre que fem un crida al controller, ja sigui des d'un botó HTML o amb un Ajax de Js, utilitzem unes rutes que s'han de crear prèviament per poder accedir al contingut.

Symfony utilitza un sistema d'enrutament per mapejar URLs a controladors. Aquestes rutes es poden definir en YAML, XML, PHP o anotacions. En el meu cas utilitzem un fitxer YAML (YAML Ain't Markup Language), el qual és un format de serialització de dades per definir, comunament, configuracions d'aplicacions.

Un exemple de definició de l'URL d'una funció del controller, podria ser el de la il·lustració 51 on és mostra com estic creant la ruta per crear relacions de RoomTypeProvider. Aquí, estic indicant que la ruta "admin.room.room_type_create" apuntarà a la funció "create" del RoomTypeController.

```
admin.room.room_type_create:
    path: /room/room_type_create
    defaults: { _controller: 'AdminBundle:Room\RoomType:create' }
```

Il·lustració 51 Definició ruta de Symfony

6.2.4 Step Functions del sistema de creació automàtic de fitxes

El sistema de creació de fitxes (anomenat internament External-Info-Collector) és una Step Function d'Amazon, la qual ja estava creada i funcionant abans que jo fes el meu TFG. El seu funcionament consistia en recórrer tots els hotels dels sistema i actualitzar tota la informació que teníem emmagatzemada d'ells a partir de peticions a 2 proveïdors (els quals anomenaré Provider1 i Provider2 per confidencialitat) que retornen abundant informació i de qualitat en cada petició.

Només s'actualitza la informació a nivell d'hotel, per tant, no es tractava cap informació a nivell d'habitació, així doncs, jo vaig implementar tota la lògica per processar la informació d'habitacions.

Al ser una Step Function que combina crides Lambda en Python i execució de commands de PHP, tot comunicat amb cues SQS de Amazon, és un sistema bastant complex i gran, així que, intentaré resumir el funcionament i implementació prèvia i després el que vagi modificar jo.

Selecció d'hotels per actualitzar

Per començar hi ha un command de PHP, el qual, va comprovant a la base de dades quins hotels són els que porten més temps sense actualitzar-se i els va agafant i introduint a la primera cua SQS del sistema anomenada "external-info-collector". A la cua només s'afegeix l'id del hotel.

```
{
   "hotelId": "<idHotel>"
}
```

Processament del missatge encuat

Quan l'Id d'hotel entra a la cua, s'executa un "Worker" el qual és un Command de PHP i la seva finalitat és recollir els paràmetres necessaris d'aquell hotel i veure si son suficients per continuar amb el procés d'actualització.

Els paràmetres són els següents:

```
{
   "hotelData": {
        "hotelId": "<hotelId>",
        "keyword": "<nombre hotel> <nombre ciudad>",
        "latitude": "<latitud>",
        "longitude": "<longitude>",
        "Provider1HotelId": "<provider1HotelId> o <null>",
        "Provider2HoteId": "<provider2HotelId> o <null>",
        "hotelCountry": "<acronim del pais del hotel> o
<null>"
   }
```

Els paràmetres de providerHotelId 1 i 2 són els respectius codis d'aquell hotel al sistema del proveïdor. Si tenim alguns dels dos IDs, el missatge anterior s'encua a la següent cua "hotel-update-request-queue".

Step function

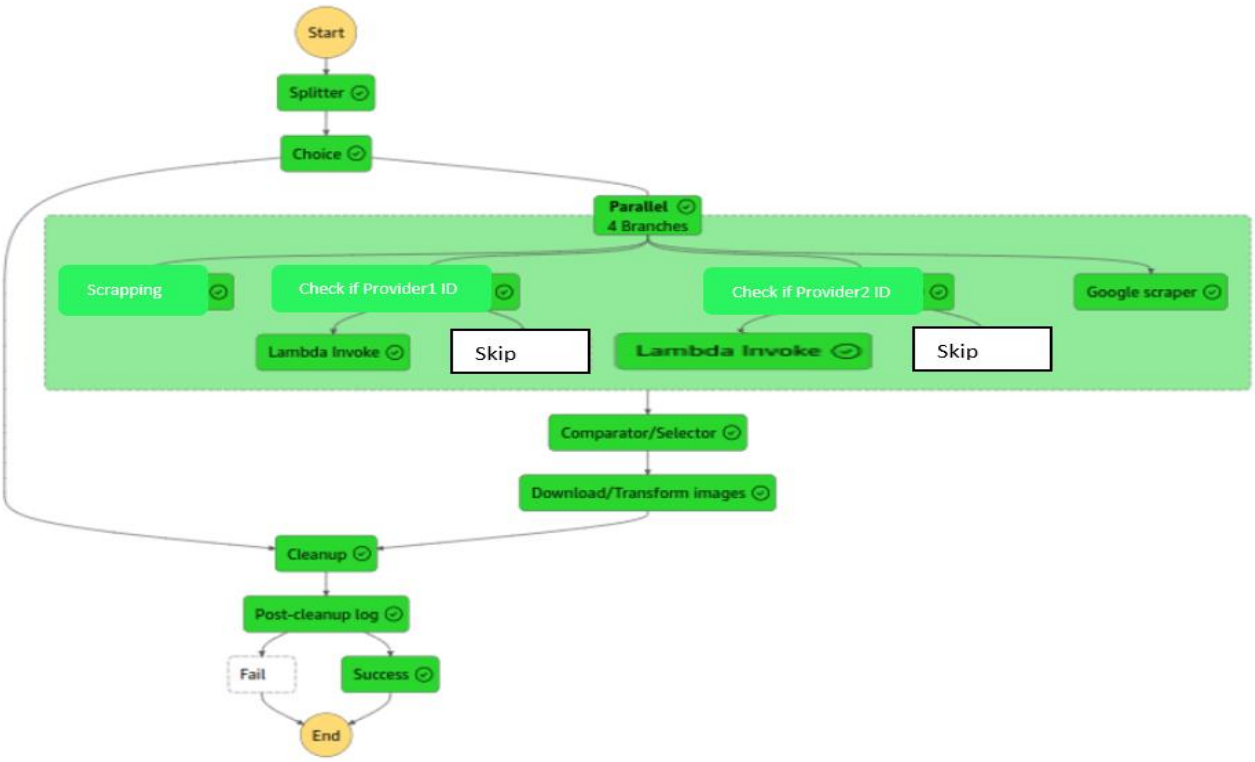
A través d'una Lambda intermèdia, s'agafen els missatges de l'anterior cua i aixeca una instància de la Step Function per cada missatge que rep (amb un límit de 5).

Podem veure representades les diferents parts/elements de la Step Function a la il·lustració 52.

El primer pas, és la Lambda "**Splitter**" la qual agafa un únic missatge amb els paràmetres de l'hotel de la cua "hotel-update-request-queue" i el passa al següent Stetp.

Al "**Choice**" comprova si encara hi ha missatges per agafar o no.

El següent bloc, s'executa en paral·lel i conté diferents lambdes per recopilar informació de fonts externes.



Il·lustració 52 Step Function Amazon

- Scrapping: Fa Scrapping a la pàgina de l'hotel, si en té, per obtenir informació.
- Provider 1 i Provider 2: Per cada un dels proveïdors, es fa una comprovació per si dins dels paràmetres hi ha el codi d'hotel d'aquell proveïdor. En cas de que si, s'executa una Lambda per fer una petició a l'API del proveïdor, al seu corresponent endpoint de GetHotel per aconseguir la màxima informació de l'hotel. Aquesta petició no es fa directament a la Lambda, sinó que, utilitzant les "**Lambda-Layers"** (un servei per importar dependències i recursos a la Lambda d'Amazon) s'importa un servei per fer crides al projecte intern "Integrations-Pull-Kotlin", i és aquest el que fa la petició a l'Api del proveïdor.
- GoogleScrapper: Lambda que fa Scrapping a Google per extreure informació general de l'hotel (Adreça, Latitud, Longitud, Contacte)

Cada una d'aquestes lambdes les quals estan implementades en Python, després d'executar el Scrapping o petició, emmagatzemen les dades aconseguides en un Json dins d'un *bucket* de S3, sota el directori content/{proveïdor}/{hotelId}/info.

A part també es separa en diferents fitxers les dades recollides, inicialment eren 4 fitxers:

-InfoGeneral, UrlsImatges, ServeisHotel i CheckInCheckOutInfo

Una vegada tenim tota la informació dins del bucket de S3, passem a la següent fase, "**Comaparator/Selector"**, el qual executa un command de PHP que recupera tota la informació de S3 d'aquell hotel i la comparar per seleccionar la "millor" en base a certs criteris per cada tipus de dada com per exemple quin proveïdor té més fotos, serveis, etc.

El següent pas és el **"Download/Transform Images"**, una Lamba en Python que s'encarrega de fer cURLs per cada imatge de l'hotel per obtenir el contingut de la imatge, aplicar un format i pujar el contingut d'aquesta a S3. També passa la informació de la cua de sortida del comparator a l'última cua SQS anomenda "hotel-info-collector-finalize-queue"

"Clean-Up" és una Lamba que s'encarrega d'eliminar el missatge d'aquell hotel de la primera cua per tal que no torni a entrar al sistema.

Finalment, una vegada ja tenim el Json amb la informació bona de l'hotel, és crida l'últim command de PHP "DataBaseUpdaterFinalizerCommand", agafa la informació del Json del hotel i actualitza tots aquesta informació a la base de dades utilitzant entitats de Doctrine.

Una vega explicat tot el funcionament del External-Info-Collector, explicaré que vaig afegir per tal de poder treballar la informació de les habitacions.

Modificacions al External-Info-Collector

Les primeres modificacions, les vaig realitzar a les Lambdes de Python del Provider1 i Provider2, encarregades de fer la comunicació amb les Apis dels proveïdors.

El getHotel ja retornava la informació de les habitacions, però no s'estava processant. Així que he creat una estructura de dades en Python anomenada roomsDetails la qual és un Array de Rooms, i a la vegada aquesta està format per diferents paràmetres com es pot veure a la il·lustració

```
room = {
    'id' : roomId,
    'name' : name,
    'occupation' : occupation,
    'size' : size,
    'bedTypes' : bedTypes,
    'description' : description,
    'images' : imageCollection['images'],
    'services' : servicesToSave['services']
}

roomsDetails['rooms'].append(room)
```

Il·lustració 53 Estructura Room Python

Així doncs, he afegit una funció a l'execució principal de la Lambda, anomenada "processRooms" on li arriba per paràmetre la part de les habitacions dels Json del getHotel, i amb un bucle vaig recorrent aquest i omplint l'estructura de dades de les habitacions.

Per últim, quan s'ha recorregut totes les habitacions, s'emmagatzema el roomsDetails al bucket de S3, aquest codi s'executa dins un **try/catch**, com podem observar a la il·lustració 54, per portar un bon control d'excepcions i registrar a les variables "errors" i "standardExecution" el comportament dels diferents submòduls de la Lambda.

```
try:

    body = json.dumps(roomsDetails)

    clientS3PutObject(body, "content/Provider1/{}/info/roomsDetails.json".format(hotelId))
    standardExecution.append('Se ha subido correctamente el archivo de la informacion de las habitaciones')

except Exception as e:
    print(e)
    errorException = traceback.format_exc()
    errors.append('Se ha producido un error al intentar subir el archivo de la informacion de las habitaciones. Motivo: ' + errorException)

return {
    'standardExecution': standardExecution,
    'errors': errors
}
```

Il·lustració 54 Codi per emmagatzemar el Json a S3

El següent canvi ha sigut **"Comparator"**, el command de PHP on es comparen els diferents fitxers de dades (infoGeneral, serveis, imatges ...) de S3 de cada proveïdor. Aquí, he afegit una nova funció de PHP per recuperar el fitxer de roomDetails.json dels dos proveïdors i els convertia en objectes DTOs per poder treballar amb les seves variables.

La comparació es basa en un parell de funcions PHP, on, es fa un recompte del nombre de serveis i imatges que tenim en total de cada habitació i s'estableix una priotirtat en base això. Una vegada s'ha seleccionat la informació "guanyadora", s'emmagatzema en un altre bucket de S3.

En el **"Download/Transform Images"** no processem cap informació de les fotos de les habitacions, així que no he afegit res.

Finalment, he modificat l'últim command del sistema, el qual s'encarrega d'emmagatzemar la informació del fitxer de S3 resultant a la base de dades.

El problema és que, així com els hotels ja estan creats al sistema de la base de dades i, per tant, només fa falta actualitzar les dades, la gran majoria d'habitacions no estan creades al sistema, i s'han de crear i relacionar a l'hotel. El gran repte aquí, és saber quina habitació genèrica del sistema s'ha d'assignar a l'habitació que ens envia el

proveïdor i una vegada tinguem una habitació del sistema seleccionada, cridar el servei de creació de fitxes **HotelRoomDataSheetCreateUseCase**, que hem mencionat anteriorment, per crear la fitxa.

Tot el codi del sistema d'actualització d'hotels està ubicat en un projecte de PHP anomenat "ExtenralInfoCollector", en canvi, el servei de creació de fitxes d'habitació està ubicat dins del projecte de "Esquiades" on s'ubica la major part del codi de l'admin del sistema, per tant, no podria cridar el servei d'un projecte a un altre.

Per resoldre aquestes casuístiques, hi ha una API interna anomenada **BackOffice API** (ubicada dins del projecte d'esquiades), la qual ofereix una interfície amb la qual es poden comunicar la gran part de projectes en base a crides API REST, i gràcies a això poder utilitzar els mètodes d'aquest projecte.

Així doncs, vaig crear un endpoint dins del BackOffice anomenat "**ManageRoomDetailsUseCase**", el qual rebia per paràmetre un objecte "**ManageRoomDetailsUseCaseReques**" on hi havia 3 atributs (HotelId, ProveïdorEscollit, LlistaHabitacions) i retornava un objecte "**ManageRoomDetailsResponse"** el qual retorna una llista amb l'id de cada habitació que ha rebut i un codi de resposta per indicar que s'ha fet amb cada una d'elles.

Des de l'últim Command de la Step Function, es fa una petició al ManageRoomDetailsUseCase.php on per cada habitació del proveïdor s'intentarà actualitzar o crear la fitxa. L'algoritme de decisions està indicat a l'apartat de disseny de l'External-Info-Collector.

Per implementar-ho he utilitzat PHP juntament amb els *repository interfaces* de les entitats RoomTypeProvider i HotelRoomProvider que vaig crear a les primeres etapes de l'aplicació, i amb això poder veure si l'habitació ja té una fitxa creada (en base l'ID de l'habitació del proveïdor), i si no en té si existeix una relació RoomTypeProvider (en base al nom de l'habitació) i, per tant, una relació que assignin aquell text d'habitació amb una habitació genèrica del sistema sobre la qual poder carregar la fitxa d'habitació.

Per cada possible opció que s'ha fet amb cada habitació que ha rebut el servei, s'assigna una constant de retorn per indicar al InfoCollector que s'ha fet amb cada una d'aquestes i poder registra-ho als logs.

Així doncs, per acabar, dins de l'últim command de la Step Function, es recorre *l'Array* de resposta i amb una estructura **Switch**, com podem observar a la il·lustració 55, es decideix quin text s'afegeix als logs en base les constants de resposta creades prèviament.

```
switch ($responseBackOffice['code']) {
    case Response::HTTP_OK:
        foreach ($responseBackOffice['msg'] as $key => $roomLog) {
            switch ($roomLog){
                case self::UPDATED_DATASHEET:
                    $logMessage .=  $key. " -> La habitacion del proveedor ha actualizado su ficha correctamente; ";
                    break;
                case self::CREATED_DATASHEET:
                    $logMessage .= $key. " -> La habitacion del proveedor ha creeado su ficha correctamente; ";
                    break;
                case self::CREATED_ROOM_TEXT:
                    $logMessage .= $key. " -> No existe una relacion del nombre de la habitacion se ha creado una entrada en la tabla de relaciones sin habitacion; ";
                    break;
                case self::INCREMENT_ROOM_TEXT_MAPPING_ATTEMPTS:
                    $logMessage .= $key. " -> Ya existe una relacion con el nombre  de la habitacion  se ha sumado 1 al intento de Mapeo; ";
                    break;
                case self::DATASHEET_EXISTS_FOR_ANOTHER_PROVIDER:
                    $logMessage .= $key. " -> Ya existe una ficha para esta habitación de otro proveedor, no se actualiza";
                    break;
                case self::EXISTING_ESQUIADES_DATASHEET:
                    $logMessage .= $key. " -> No se creará ninguna ficha, ya que existen por Esquiades ";
                    break;
                case self::MANUAL_MODIFIED_DATASHEET:
                    $logMessage .= $key. " -> No se actualizará la ficha, ya que se ha modificado manualmente";
                    break;
                case self::EXISTING_ACTIVE_RATEPLAN:
                    $logMessage .= $key. " -> No se creará la ficha, ya que hay tarifas de channel";
                    break;

                default:
                    $logMessage .= $key. " -> Error en esta habitacion; ";
            }
```

Il·lustració 55 Sistema de Logs de la creació de fitxes

7 Proves i resultats

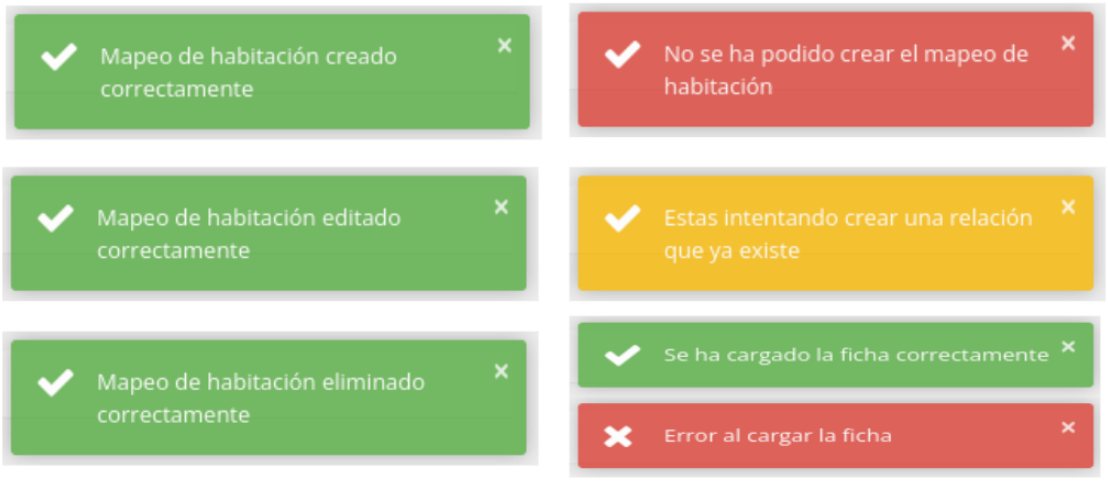
Una vegada explicada tota la implementació de l'aplicació, és important mencionar quines proves vaig realitzar per comprovar el correcte funcionament d'aquest.

7.1 Proves realitzades

Cada vegada que creava una nova funcionalitat, testejava casuístiques diferents per tal d'evitar errors de producció en un futur.

Respecte a les dues pantalles que vagi crear a l'administrador, la manera de testejar era a través dels codis de resposta de les diferents crides Ajax que vaig desenvolupar. També validava si els canvis realitzats sobre les entitats (Creació, Modificació, Eliminació) es reflectissin a base de dades. Si durant el procés de creació d'una entitats, algun dels paràmetres era obligatori, realitzava una comprovació per JavaScript per tal de validar que no estigués buit.

Depenent del codi de resposta mostraria una notificació per pantalla a l'usuari per indicar el resultat de l'operació. Les notificacions poden tenir 3 colors diferents per indicar el "gravetat" del problema. Verd (*Succes*) tot correcte, Groc hi hagut algun problema com per exemple que la entitat que vols crear ja existeix (*Warning*) i per últim el color Vermell per indicar quan hi ha hagut algun error intern al codi o ha fallat la petició al proveïdor (Error). Podem veure exemples de aquestes notificacions a la següent imatge.



Il·lustració 56 Exemples de resposta del BackEnd

A part dels codis de resposta, sempre corroborava la informació amb l'estat de les entitats a la base de dades. Utilitzava l'eina del phpMyAdmin (Il·lustració 57) per veure els registres de base de dades d'una manera més visual, ràpida i fàcil.

	id	id_room	id_hotel	id_provider	provider_code	last_update	user_update
Edit Copy Delete	185988	12320	123	1804	314029	2024-06-05 18:03:16	382
Edit Copy Delete	185989	16252	123	270	2013998	2024-02-21 11:55:37	185

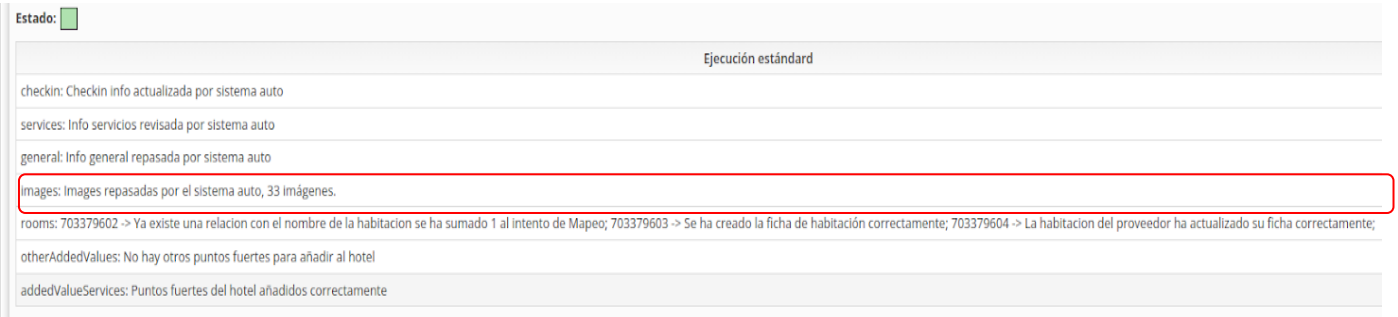
Il·lustració 57 Entitat HotelRoomProvider al phpMyAdmin

Amb el sistema del InfoCollector, vaig fer moltes proves en local dels diferents components de la StepFunction, ja que la plataforma de AWS té un perfil "dev" per tal de fer proves en local.

El problema és que no vaig aconseguir fer funcionar el sistema automàtic d'orquestració complet en local, així que, havia d'encuar jo manualment els missatges a les cues SQS i executar manualment les Lambdes de Python i els commans de PHP.

Per comprovar el correcte funcionament del sistema, validava si es creaven els fitxers de dades als corresponents buckets de S3 i si les dades de l'interior d'aquests eren correctes.

Com vaig crear un sistema de codis de resposta en base a què fa el sistema de "ManageRoomDetailsUseCase", a través de DynamoDB (base de dades NoSQL gestionada per AWS), on es guarden diferents logs del InfoCollector i on vaig afegir els meus, puc controlar el comportament del sistema. A la següent imatge es pot veure els diferents logs del procés, entre ells, es pot veure que el proveïdor ha retornat 3 habitacions, i per cada una s'ha fet una acció diferent. També corroborava que les fitxes es creessin a base de dades i es visualitzessin correctament a la web.



Il·lustració 58 Logs del InfoCollector

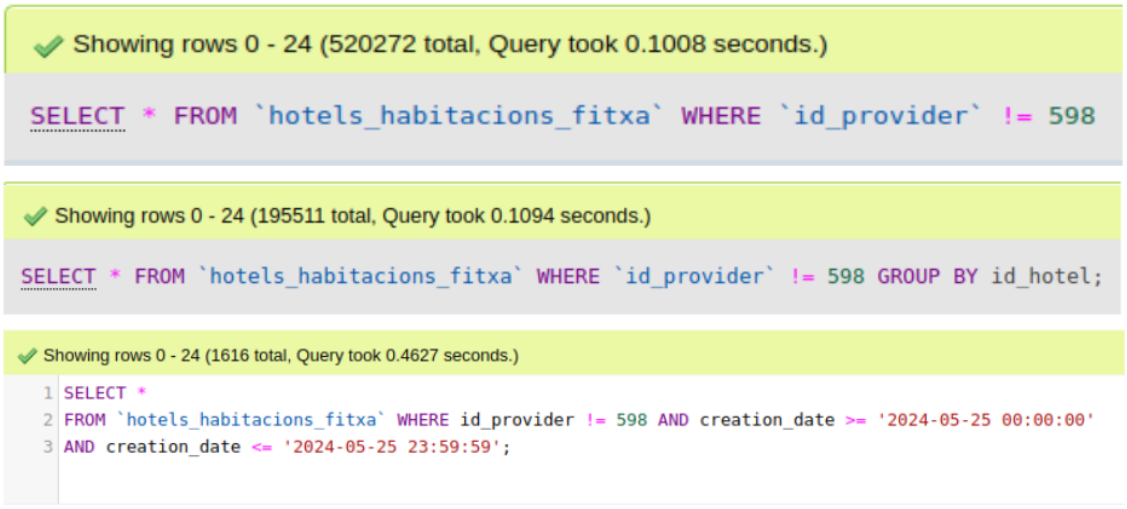
7.2 Resultats

Els resultats obtinguts amb les proves anteriorment explicades van ser positius. Amb cada mètode implementat, vaig realitzar diferents testos i millores fins a aconseguir que els codis de resposta i els registres a bases de dades fossin els correctes.

El dia d'avui, el sistema de creació de fitxes d'habitació porta un total de 520272 fitxes creades, a nivell d'hotel en tenim 195511 amb, com a mínim, una fitxa creada, ja que, un hotel pot tenir més d'una fitxa.

El nombre de fitxes que el sistema crea per dia són unes 1500 de mitjana, el que són unes 60 fitxes per hora o 1 fitxa per minut. Un exemple és el dia 25-05-2024, on va crear un total de 1616 fitxes.

Totes les dades exposades les he extret de base de dades a partir de les següents *querys*. El id_provider != 598 és per no tenir en compte les fitxes que s'han creat manualment, ja que, 598 és el id de l'empresa.



Il·lustració 59 Querys SQL resultats fitxes

Un altre paràmetre important és si aquestes fitxes d'habitació, estan sent utilitzades pels clients, és a dir, si estan prement el botonet de "Ver más" al costat de l'habitació.

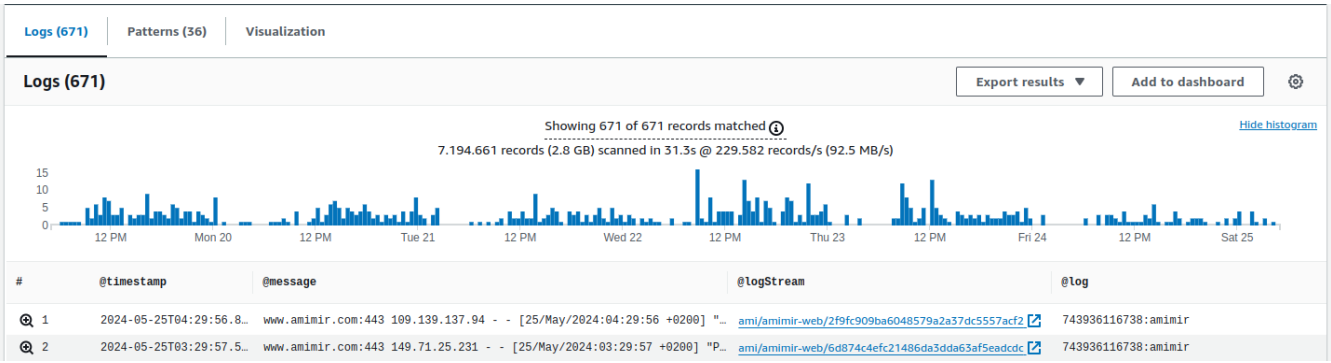
Per poder fer això, he utilitzat el **Logs Insights** d'Amazon **CloudWatch** el qual és una funcionalitat avançada de AWS que permet analitzar i buscar de manera interactiva els registres (logs) generats per les aplicacions i la infraestructura. És una eina poderosa per

monitorar, depurar i optimitzar aplicacions i sistemes, proporcionant una capacitat de cerca i anàlisi ràpida i eficaç sobre grans volums de dades de registre.

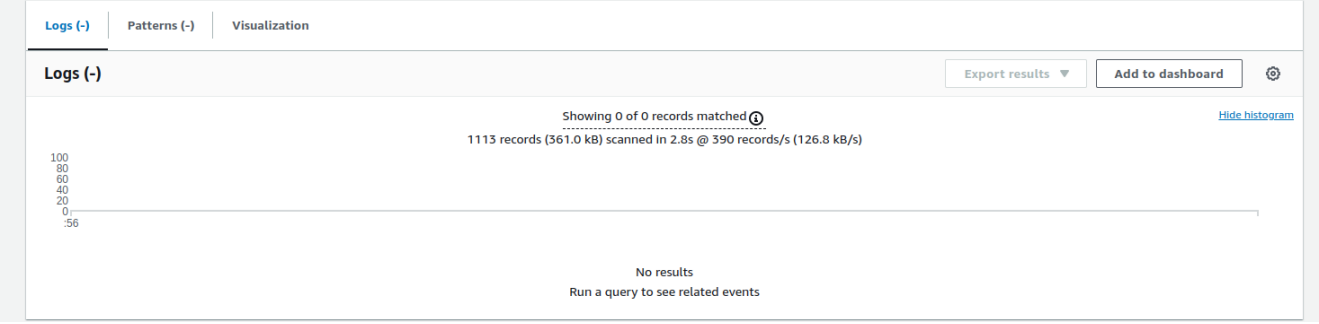
He fet una cerca personalitzada per veure durant una setmana, la del 20 al 26 de maig del 2024, per visualitzar quantes vegades s'ha clicat el botó "Ver más" a la pàgina web d'Amimir. També he fet la cerca per veure els resultats en un dia concret (el dijous 23 de maig) i un altre d'una setmana abans de pujar a producció el meu projecte.

Els resultats són:

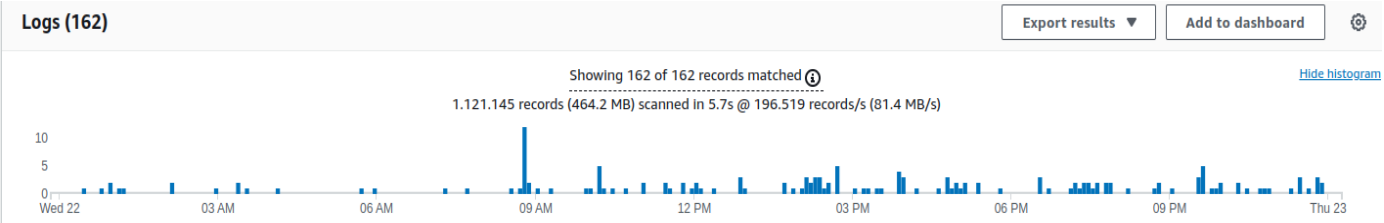
- 671 clics en una setmana. Il·lustració 60
- 0 clics abans de pujar el projecte. Il·lustració 61
- 162 clics un dia. Il·lustració 62



Il·lustració 60 Nombre de clics en una setmana



Il·lustració 61 Nombre de clics en una setmana (abans)



Il·lustració 62 Nombre de clics en un dia

Es pot observar una clara millora, ja que abans que jo fes el projecte, les fitxes d'habitació era un element que només es treballava de forma manual a la pàgina web d'esquiades.com, perquè és un producte molt més reduït i controlat.

Tenint en compte que s'han creat fitxes per a uns 195000 hotels, 671 clics setmanals no son tants, això és degut a que el botó de "Ver más" només és mostra per producte que ens envia el proveïdor amb el qual hem mapejat la fitxa, ja que, s'ha fet una relació a nivell d'Ids d'habitació. Un millora que estic implementat de cara al futur, és poder relacionar en directe durant el procés de compra, totes les fitxes que ja tenim creades, amb les habitacions dels diferents proveïdors durant la petició de disponibilitat.

Aquesta millor, multiplicaria el nombre de clics per setmana, ja que, en un funcionament òptim, el botó de "Ver más" es mostraria per molts més proveïdors. Però bé, com bé he dit és una millora a futur.

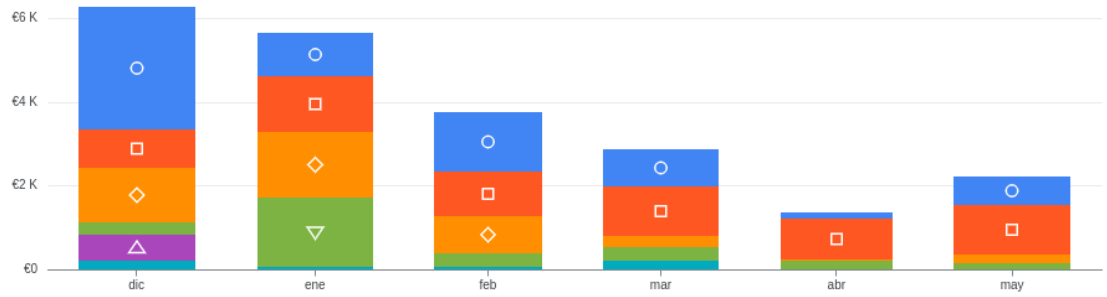
7.3 Costos

Una mètrica molt important, són els costos que està generant el total d'aquest sistema a nivell d'utilització de serveis externs com Amazon o Google.

Quant als serveis d'Amazon, no es va detectar un increment important en els costos, també és veritat que la StepFunction del InfoCollector és un projecte que està en constant millora i implementant noves funcionalitats a la vegada que jo realitzava el meu projecte, per tant, és difícil estimar els costos exactes de la meva part.

En canvi, per part de Google, en concret al servei de traduccions, sí que es van elevar els costos a causa del meu projecte. Com he explicat anteriorment, el sistema tradueix les distribucions de llits als 7 idiomes de la pàgina web. Aquests textos són molt curts, però quan es tradueixen per un nombre molt elevat d'habitacions els costos son notoris.

A continuació podem observar les gràfiques del *pricing* dels diferents serveis que utilitzem de Google. En concret el color verd és el dedicat a les traduccions.



Il·lustració 63 Gràfica costos serveis de Google

He d'aclarir que no tots els costos son relacionats al meu sistema, ja que altres aplicacions de l'empresa utilitzen aquests serveis.

En concret el gener es va llançar un sistema de creació massiu d'hotels el qual va generar un cost molt elevat, el qual no té res a veure amb el meu sistema.

A finals de desembre el meu sistema va començar a funcionar amb una quantitat limitada d'hotels, i es pot veure una petita franja verda on a part de la resta de costos de l'empresa, també estan els meus.

A gener vaig augmentar el límit dels hotels del sistema i a part també va haver-hi moltes tasques de traducció externes a mi, la qual cosa es veu reflectida amb un augment de la franja verda.

A febrer i març els costos es van normalitzar.

A l'abril vaig fer proves amb el traductor de DeepL en comptes del de Google, però el *pricing* (preu) era semblant al de Google.

Finalment, a mitjan abril vaig implementar una millora per guardar internament les diferents distribucions de llits i les seves respectives traduccions. Aquest nou sistema, només traduïa els textos en cas de no trobar cap traducció a base de dades. A part, cada vegada que cridi al traductor extern, s'emmagatzemarà la traducció per la següent vegada.

Amb aquest canvi, els canvis de març a abril es van quasi a la meitat.

8 Pujada a producció

8.1 Organització del projecte a l'empresa

En aquest apartat explicaré com s'ha organitzat el desenvolupament de les diferents tasques del projecte.

Durant tota l'estança del TFG a l'empresa VPT, he estat dins d'un dels seus equips de treball, en concret en l'equip de "Producte". Aquest equip està orientat a desenvolupar i millorar les connexions amb els diferents proveïdors, el que entenem com integracions. Per tant, és l'encarregat de proporcionar als treballadors de l'empresa diferents vies de càrrega de producte per a les pàgines web.

Tots els equips de programació de l'empresa treballen amb SCRUM, el qual és un marc de gestió de projectes àgil per gestionar el desenvolupament de projectes complexos, especialment en àmbit de Software. La idea principal de les metodologies "Agile", és permetre fer un canvi de prioritats dins de cada fase del projecte segons els objectius i necessitats del client, ja que, estan orientades a obtindre resultats tangibles des d'un principi.

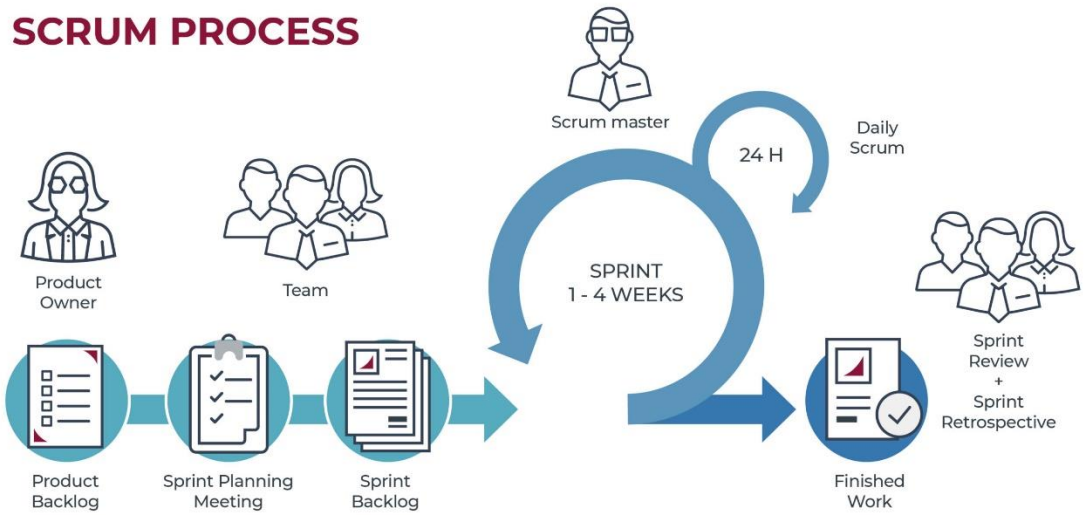
Dins d'aquestes metodologies hi ha 3 rols principals:

- **Product Owner**: Encarregat de maximitzar el valor del producte i del treball de l'equip de desenvolupadors. Defineix i prioritza les diferents tasques que és fan a cada **Sprint**.
- **Scrum Master**: Especialista en Scrum que s'encarrega que es compleixin les regles del marc de treball. Intenta eliminar o reduir els impediments que poden sorgir durant el desenvolupament del Sprint. Dona consell tant al Product Owner com a l'equip de treball.
- **Developer Team**: Equip de desenvolupadors que realitzarà les tasques del projecte. Són els únics que poden puntuar les diferents tasques per assignar un valor numèric sigui per dificultat o temps que s'ha d'invertir.

Així doncs, el *product owner*, juntament amb la resta de treballadors, aniran creant tasques per als diferents projectes que existeixen. Totes aquestes tasques van a parar al "**Product Backlog**", el qual és un llistat dinàmic per agrupar totes les tasques relacionades a un projecte/equip. Del *Backlog* és d'on s'agafaran totes les tasques per construir els *Sprints*, els quals explicaré a continuació.

Una vegada hem vist els diferents rols del scrum, mencionaré els elements que conformen el "Scrum Process", els quals podem observar a la Il·lustració 64.

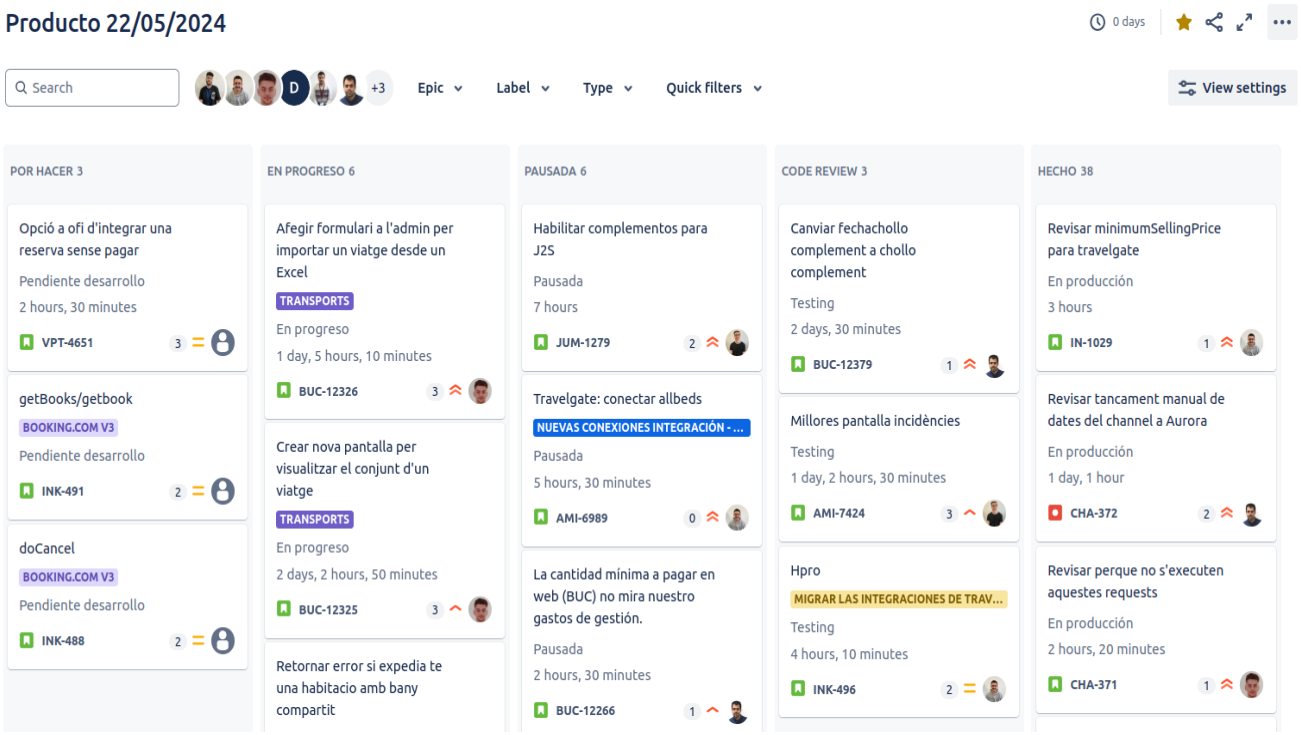
- **Sprint**: Interval de temps en el que l'equip treballa per finalitzar totes les tasques que s'han establert per aquest període. La duració d'un Sprint normal sol ser de 1-4 setmanes. En el cas del meu equip són 2 setmanes.
- **Daily Scrum**: Reunió diària d'uns 15 minuts com a màxim per veure com va l'estat del projecte.
- **Sprint Planning**: Reunió que es fa just al principi del Sprint per identificar quines tasques s'agafaran del Backlog i es realitzaran durant el període de treball. També s'assignaran prioritats a les tasques per establir un ordre segons les demandes del product owner.
- **Task Valoration**: Reunions per revisar les tasques que hi ha al Backlog i afegir-li una puntuació en vers a temps o dificultat. A vegades s'inclou dins del propi Sprint Planning.
- **Sprint Retrosprective**: Reunió que es fa al final del Sprint per reflexionar sobre com ha anat el període de treball**,** quins problemes o interrupcions hi ha hagut, etc. Normalment, es fa amb dinàmiques de grup per fer-ho més fàcil.



Il·lustració 64 Esquema metodologia SCRUM

Per facilitar la correcta implementació del SCRUM, l'empresa utilitza JIRA, una eina software d'Atlassian, utilitzada per la gestió de projectes.

Amb el JIRA, els diferents usuaris poden crear les tasques que en un principi aniran al Backlog i posteriorment al Sprint. Tots els desenvolupadors tenim un compte, i podem assignar-nos les tasques que realitzem així com modificar-les. A la il·lustració 65 es pot observar el panell de "Active Sprint" on apareixen totes les tasques del Sprint i el seu estat (To do, In progress, Code Review o Done)



Il·lustració 65 Panell Active Sprint del JIRA

Per cada nou Sprint, jo creava tasques relacionades amb el meu TFG i les afegia amb la resta de les tasques del Sprint de Producte. Així, jo també seguia la mateixa dinàmica de grup que la resta, i podia veure com avançaven la resta de projectes de l'empresa. També em va servir per marcar-me uns objectius i establir la duració dels Sprints per acabar-los.

8.2 Desplegament a producció

Per desenvolupar les noves funcionalitats als diferents projectes, s'utilitza el repositori de Git per al control de versions. Cada vegada que s'implementa una nova tasca, s'ha de crear una nova branca de git amb l'Id de la tasca del JIRA com a nom de la branca (Ex: AMI-1423).

Una vegada considero que tinc la funcionalitat de la tasca acabada i amb proves/test realitzats, creo un **"Merge Request**" i li assigno a algun company de l'equip perquè me'l revisi.

Les diferents webs i aplicacions de l'empresa estan allotjades a la plataforma de EC2 d'Amazon. Amazon EC2 (Elastic Compute Cloud) és un servei de computació escalable en el núvol, que permet als usuaris desplegar i gestionar aplicacions en una infraestructura segura i flexible.

Per garantir que el procés de desplegament del codi sigui eficient i fiable, utilitzem GitLab CI/CD (Continuous Integration/Continuous Deployment). GitLab CI/CD és una eina poderosa que permet automatitzar els processos de construcció, test i desplegament del codi, assegurant que cada canvi fet al codi font es desplegui de manera consistent i sense errors.

El flux de treball per desplegar el codi a producció implica diversos passos que s'executen de manera automàtica a través de **pipelines** definides a GitLab CI/CD. A continuació, es detallen les etapes clau del procés:

- **Commit i Push del Codi**: Els desenvolupadors realitzen canvis al codi localment i, un cop validats, aquests canvis es commiten i es pugen al repositori de GitLab.
- **Execució de Pipelines**: Cada cop que es fa un push al repositori, GitLab CI/CD inicia una pipeline. Aquesta pipeline està composta per diverses etapes, com ara la construcció del codi, l'execució de tests automatitzats i la verificació de la qualitat del codi.
- **Construcció del Codi**: En aquesta etapa, el codi es compila i es preparen els artefactes necessaris per al desplegament.
- **Tests Automatitzats**: Es realitzen una sèrie de tests automatitzats per assegurar que els canvis no introdueixen errors o regressions. Això inclou tests unitaris, tests d'integració i altres verificacions de qualitat.

- **Desplegament a Producció**: Si totes les etapes anteriors es completen amb èxit, es procedeix al desplegament del codi a l'entorn de producció. Aquesta etapa implica copiar els artefactes construïts als servidors EC2 d'Amazon, actualitzant l'aplicació en viu amb els darrers canvis.

Aquest procés d'automatització mitjançant GitLab CI/CD pipelines no només redueix el temps necessari per desplegar canvis al codi, sinó que també minimitza el risc d'errors humans i assegura una alta qualitat del codi desplegat. A més, permet als desenvolupadors enfocar-se en la creació de noves funcionalitats i millores, mentre que el procés de desplegament es gestiona de manera eficient i consistent.

9 Conclusions

Per concloure el meu treball, m'agradaria expressar que aquest projecte, ha sigut una gran oportunitat per posar-me a prova tant a mi com als meus coneixements adquirits durant els anys de la carrera.

Per començar, he pogut experimentar com funciona i es gestiona internament la secció d'IT d'una empresa important en el sector. He après dinàmiques de coordinació d'equips com el SCRUM i eines per la gestió de tasques com el JIRA, coses que durant la universitat no es mencionen, però considero importants, ja que, són la base del treball en equip. També conèixer la infraestructura tecnològica que engloba una empresa tan gran.

En segon lloc, he après bastants noves tecnologies que no coneixia abans de començar el projecte, com per exemple el llenguatge PHP juntament amb Symfony. He tingut l'oportunitat d'utilitzar diferents serveis de AWS com S3, Lambes i StepFunctions, tecnologies que són molt interessants i ofereixen una gran varietat d'utilitats. Per altre costat, considero que he millorat alguns coneixements adquirits a la carrera, com per exemple, HTML, CSS, JavaScript i interacció amb bases de dades tant amb SQL com amb entitats de persistència.

He experimentat que es sent quan puges el teu codi a producció, la felicitat de veure que la teva funcionalitat està funcionant correctament i oferint un nou servei útil per l'empresa. Com a contrapart he experimentat el estrès i nerviosisme de veure com alguna part del codi que s'ha pujat a producció ha provocat que la pàgina web deixes de funcionar, revertint els canvis ràpidament.

Estic content amb resultat final del meu treball, ja que, totes les funcionalitats i objectius s'han desenvolupat correctament.

En resum, considero que ha sigut una molt bona experiència i els coneixements adquirits seran claus per al meu futur com a enginyer informàtic.

10 Referències

[1] Entrevista història empresa VPT. https://www.rtve.es/play/audios/son-4-dies/entrevista-rafa-fuertes-director-fundador-desquiadescom/6826723/

[2] Informació de TWIG. https://www.acens.com/comunicacion/wp-content/images/2014/06/twig-plantillas-wp-acens.pdf

[3] Informació sobre PHP. https://www.php.net/manual/es/intro-whatis.php

[4] Informació de Kotlin. https://www.plainconcepts.com/es/kotlin-android/

[5] Funcionament de Doctrine. https://openwebinars.net/blog/que-es-doctrine-y-como-funciona/

[6] Informació Channel Manager. https://www.ifema.es/noticias/turismo/que-es-un-channel-manager-y-para-que-sirve

[7] Patrons d'arquitectura. MVC. https://rjcodeadvance.com/patrones-de-software-patron-mvc-ejemplo-parte-4/

[8] Informació HTML, CSS. https://www.vadavo.com/blog/html-que-es-y-para-que-sirve/#%C2%BFQue_es_HTML_y_para_que_sirve

[9] Explicació SCRUM. https://www.vadavo.com/blog/html-que-es-y-para-que-sirve/#%C2%BFQue_es_HTML_y_para_que_sirve

[10] Metodologies Àgils. https://www.escueladenegociosydireccion.com/revista/business/scrum-framework-agiliza-trabajo-equipo/

[11] Dubtes globals sobre programació. https://stackoverflow.co/