

Grau en Enginyeria Informàtica de Gestió i Sistemes d’Informació

PLATAFORMA DE VISUALITZACIÓ D’ALERTES

SANITÀRIES

Memòria intermèdia

ERIK ESPUÑES JUBERÓ

TUTOR: EUGENI FERNÁNDEZ GONZÁLEZ

CURS 2020-2021



ABSTRACT

This project is focused on the development of a mobile application that warns the hospital staff when an event of their responsibility occurs. The product will be focused on two parts.

The first one will be a web server that will configure the users and the type of alert that they will receive.

The second one will be a mobile app that will be responsible for configuring and warning the user.

RESUM

Aquest treball se centra en el desenvolupament d’una aplicació de mòbil que avisi al personal d’un hospital quan succeeixi algun esdeveniment que sigui de la seva responsabilitat. El producte se centrarà en dues parts.

La primera, un servidor web que serà el que configuri els usuaris i el tipus d’alertes que puguin rebre.

La segona, una aplicació per a mòbils que serà l’encarregada de configurar l’alerta i avisar a l’usuari.

RESUMEN

Este trabajo se centra en el desarrollo de una aplicación de móvil que avise al personal de un hospital cuando suceda algún evento que sea de su responsabilidad. El producto se centrará en dos partes.

La primera, un servidor web que será el que configure los usuarios i los tipos de alertas que pueda recibir.

La segunda, una aplicación para móviles que será la encargada de configurar la alerta y avisar al usuario.

ÍNDEX

[ÍNDEX DE FIGURES III](#_Toc69985542)

[ÍNDEX DE TAULES V](#_Toc69985543)

[GLOSSARI DE TERMES VII](#_Toc69985544)

[CAPÍTOL 1. INTRODUCCIÓ 1](#_Toc69985545)

[CAPÍTOL 2. MARC TEÒRIC 3](#_Toc69985546)

[2.1. CONTEXT 3](#_Toc69985547)

[2.1.1. L’ERA DEL PAPER 3](#_Toc69985548)

[2.1.2. L’ERA DIGITAL 4](#_Toc69985549)

[2.2. ANTECEDENTS 5](#_Toc69985550)

[2.3. NECESSITATS DE LA INFORMACIÓ 5](#_Toc69985551)

[2.3.1. HEALTHCARE INFORMATION SYSTEM (HIS) 6](#_Toc69985552)

[2.3.2. MOTOR D’INTEROPERABILITAT 7](#_Toc69985553)

[2.3.3. PROTOCOLS ENTRE EL HIS I EL MOTOR D’INTEROPERABILITAT 7](#_Toc69985554)

[CAPÍTOL 3. OBJECTIUS I ABAST 11](#_Toc69985555)

[3.1. OBJECTIUS DEL CLIENT 11](#_Toc69985556)

[3.2. OBJECTIUS DEL PRODUCTE 11](#_Toc69985557)

[3.3. PÚBLIC POTENCIAL 11](#_Toc69985558)

[3.4. ABAST 11](#_Toc69985559)

[CAPÍTOL 4. METODOLOGIA 13](#_Toc69985560)

[CAPÍTOL 5. DEFINICIÓ DE REQUERIMENTS FUNCIONALS I TECNOLÒGICS 15](#_Toc69985561)

[5.1. REQUERIMENTS FUNCIONALS DEL SERVIDOR WEB 15](#_Toc69985562)

[5.2. REQUERIMENTS FUNCIONALS DE L’APLICACIÓ 15](#_Toc69985563)

[5.3. REQUERIMENTS TECNOLÒGICS 16](#_Toc69985564)

[CAPÍTOL 6. DESENVOLUPAMENT 17](#_Toc69985565)

[6.1. DISSENY 17](#_Toc69985566)

[6.1.1. BASES DE DADES 17](#_Toc69985567)

[6.1.2. SERVIDOR WEB 18](#_Toc69985568)

[6.1.3. APLICACIÓ MÒBIL 25](#_Toc69985569)

[6.2. SERVIDOR IRIS 35](#_Toc69985570)

[6.2.1. CONFIGURACIÓ 35](#_Toc69985571)

[6.2.2. CREACIÓ DEL HIS 35](#_Toc69985572)

[6.2.3. CREACIÓ DE L’API 36](#_Toc69985573)

[6.2.4. DEPLOY 38](#_Toc69985574)

[6.3. SERVIDOR WEB 38](#_Toc69985575)

[6.3.1. BACKEND 39](#_Toc69985576)

[6.3.2. REST API 39](#_Toc69985577)

[6.3.3. FRONTEND 39](#_Toc69985578)

[6.4. APLICACIÓ 39](#_Toc69985579)

[6.4.1. BACK END 39](#_Toc69985580)

[6.4.2. FRONT END 39](#_Toc69985581)

[CAPÍTOL 7. BIBLIOGRAFIA 41](#_Toc69985582)

# ÍNDEX DE FIGURES

[Figura 2.1: Esquema de les dades que emmagatzema l'HIS [12] 6](#_Toc69985583)

[Figura 2.2: Exemple de missatge de HL7 v2 [16, p. 7]. 8](#_Toc69985584)

[Figura 2.3: Exemple del missatge de HL7 v3 8](#_Toc69985585)

[Figura 2.4: Exemple d’una imatge generada pel protocol DICOM [17]. 9](#_Toc69985586)

[Figura 7.1: Diagrama ERD de la base de dades. 18](#_Toc69985587)

[Figura 7.2: Diagrama UML de classes del servidor web. 20](#_Toc69985588)

[Figura 7.3: Diagrama BPMN de la creació d’una institució mèdica. 21](#_Toc69985589)

[Figura 7.4: Diagrama BPMN de la gestió de les notificacions. 22](#_Toc69985590)

[Figura 7.5: Disseny de la UI de l’inici de sessió en el servidor web. 23](#_Toc69985591)

[Figura 7.6: Disseny de la UI de la pàgina d’inici en el servidor web. 24](#_Toc69985592)

[Figura 7.7: Dissenys de la UI de la llista, visualització i creació d’una institució mèdica en el servidor web. 25](#_Toc69985593)

[Figura 7.8: Diagrama UML de classes de l’aplicació mòbil. 27](#_Toc69985594)

[Figura 7.9: Diagrama BPMN de l’inici de sessió en l’aplicació. 28](#_Toc69985595)

[Figura 7.10: Diagrama BPMN de la visualització de la llista d’alertes en l’aplicació. 29](#_Toc69985596)

[Figura 7.11: Diagrama BPMN de la visualització d’una alerta en l’aplicació. 30](#_Toc69985597)

[Figura 7.12: Disseny de la UI de la llista d’alertes en l’aplicació mòbil. 34](#_Toc69985598)

[Figura 7.13 Figura 7.14: Disseny de la UI de la visualització d’una alerta en l’aplicació mòbil. 34](#_Toc69985599)

[Figura 7.15: Imatge de la configuració d’una API en Intersystems IRIS for Health. 38](#_Toc69985600)

# ÍNDEX DE TAULES

[Taula 7.1: Cas d’ús d’inici de sessió en l’aplicació mòbil. 32](#_Toc69985601)

[Taula 7.2: Cas d’ús de la llista d’alertes en l’aplicació mòbil. 32](#_Toc69985602)

[Taula 7.3: Cas d’ús de la visualització d’una alerta en l’aplicació mòbil. 33](#_Toc69985603)

# GLOSSARI DE TERMES

|  |  |
| --- | --- |
| ANDROID | Sistema operatiu de dispositius mòbils desenvolupat per Google. |
| API | Application Programming Interface. |
| CAP | Centre d’Atenció Primària. |
| E-HEALTH | Ajuda que tenen les institucions mediques en la tecnologia per a fer més accessible i segur els eu entorn. |
| FRAMEWORK | Entorn de treball |
| HIS | Health/Healthcare Information System (Sistema d’Informació Sanitari) |
| iOS | Sistema operatiu de dispositius mòbils desenvolupat per Apple. |
| IRIS | Motor d’interoperabilitat desenvolupar per InterSystems. |
| SCRUM | Metodologia àgil de planificació dels projectes centrat en fites i sprints. |
| SPRINT | Durada de la iteració en la metodologia scrum. Normalment sol ser d’una setmana a un mes. |
| TCP | Protocol per a la comunicació de dos dispositius en internet. |
| TDD | Metodologia de programació centrada en fer primer els tests que escriure el codi. |
| TFG | Treball de Final de Grau, |
| UML | Unified Modeling Language |

# INTRODUCCIÓ

En els anys on s’han vist marcats per una pandèmia mundial, s’ha pensat a fer una aplicació per ajudar tot el personal sanitari i millorar la tecnologia que s’usa als hospitals.

Es pretén aconseguir desenvolupar una aplicació de mòbil que enviï alertes a l’usuari sanitari que l’estigui usant, quan succeeixi qualsevol fet que sigui de la seva responsabilitat. Per exemple, si un gerent d’un hospital necessita que l’avisin quan l’ocupació dels llits per a persones ingressades per COVID-19 supera un valor, aquesta aplicació enviarà una alerta al mòbil del gerent perquè pugui intervenir de manera ràpida. Aquesta aplicació es comunicarà amb el motor d’integració que usi l’hospital, en el cas d’aquest TFG, es comunicarà amb una API que es generarà del software IRIS [1].

Aquesta aplicació ve acompanyada d’un servidor que estarà en línia i serà el que permeti configurar quin tipus d’alertes rep cada usuari de l’aplicació. Posant d’exemple el cas anterior, abans de poder rebre aquestes alertes, s’haurà de configurar que el gerent pugui rebre alertes d’ocupació de pacients de COVID-19. Després serà el gerent qui configurarà el rang que fa que aquell valor sigui crític perquè l’aplicació l’avisi o no.

# MARC TEÒRIC

## CONTEXT

Aquest TFG entra en el món de la medicina. Per tant per saber bé el context del producte a elaborar s’ha de veure l’evolució que ha tingut el tractament de dades en els hospitals. Podem diferenciar clarament dues eres on la revolució informàtica va permetre el salt d’una a altra. La primera, l’era en què totes les dades dels hospitals eren a paper, on es veurà quins problemes hi havia i el motiu pel qual es va fer la transició a l’era digital.

### L’ERA DEL PAPER

La primera vegada que es va escriure un document mèdic, que es té constància, és d’un papir de l’antic Egipte del 1600 Ac, on es reporten casos de tractaments de ferides de guerra.

Els documents mèdics van seguir evolucionant amb els grecs. Hipòcrates, el metge de l’Antiga Grècia més famós i considerat el pare de la medicina, va fundar la seva pròpia escola per aprendre medicina i va deixar documents escrits sobre els símptomes que tenien els pacients.

Tots aquests documents es van anar traduint a l’àrab perquè metges com al-Razi poguessin aprendre medicina en la gran era daurada islàmica entre el segle VIII i XIII. Els àrabs són els que van crear la idea de l’hospital. A més van ser els que van crear els historials mèdics, que eren escrits pels aprenents de metge sota la supervisió d’un metge titulat.

Aquests documents van seguint evolucionant en l’època de la llum a Europa. Primer a la França del segle XVII on les anàlisis de l’anatomia humana van canviar la perspectiva de la medicina. Després a inicis del segle XIII a Suècia van començar a refinar un sistema d’històries mèdiques a paper que van anar refinant fins al segle XX. [2]

A finals del segle XIX i inicis del XX, va haver-hi un canvi important, on als Estats Units es va començar a fer un historial mèdic dels pacients com es coneix en l’actualitat. Això inclou, un identificador del pacient, hàbits del pacient, malalties anteriors, historial familiar, anàlisis, entre molts altres aspectes més. Aquest historial va ser molt útil, i segueix sent-ho, per a ambulatoris o centres d’atenció primària. [3]

#### PROBLEMES DELS HISTORIALS MÈDICS EN PAPER

Amb els documents a paper sorgeixen una sèrie d’inconvenients que tenen a veure amb l’espai físic que ocupen, la cerca d’aquells documents i, també, la mobilitat que tenen.

El primer problema sorgeix amb l’espai que ocupa cada document i el temps que s’han de tenir arxivats. Si agafem dades d’un hospital universitari alemany [4], cada any ingressen entre 300.000 i 400.000 pacients. Això significa que es generen vuit milions d’impressions a paper, que equival a un volum d’un quilòmetre i mig en paper. A part del volum generat en un any, a Alemanya s’ha de guardar tot document mèdic un mínim de trenta anys. En canvi si aquests documents fossin digitals, ocuparien entre 10 i 15 terabytes.

Amb aquestes dades es pot veure que emmagatzemar les dades a paper requereix un espai d’emmagatzematge molt gran, i a més requereix un manteniment humà (empleats encarregats per gestionar aquelles dades) i físic (el magatzem). El temps per poder accedir a qualsevol document emmagatzemat també és molt elevat, fent que sigui bastant ineficient haver d’anar a consultar dades en aquests historials. Per acabar, tenir la documentació en paper dificulta la interconnexió amb diferents països. Per exemple si una persona viatja a un país com el Japó i ha de ser atès en un hospital d’aquella nació, tindran molt complicat, per no dir gairebé impossible, consultar el seu historial si aquesta està emmagatzemada en paper.

El motiu principal de la transició dels documents mèdics de paper als digitals va ser perquè amb el temps les màquines dels hospitals recol·lectaven més i més dades que han de ser emmagatzemades. Per tant, era més còmode emmagatzemar-les en digital que imprimir-les [5].

### L’ERA DIGITAL

Tot i que no se sap ben bé qui va fer el primer sistema de registres mèdics en digital o electrònic, segons Dalianis [6], durant els anys 60 i 70, els suecs van construir el primer sistema de registres electrònics per un entorn mèdic. Es va fer a l’Hospital Universitari de Karolinska.

Amb el que no hi ha discrepància és amb el primer sistema utilitzat de manera general. Va ser desenvolupat per la Universitat de Harvard i es deia COSTAR [7] (Computer-stored ambulatory record). Amb el temps van anar sorgint nous sistemes creats de manera interna a algun hospital o de manera pública. Més tard mentre progressaven l’electrònica aquests sistemes que estaven als hospitals es van anar fent més i més grans fins al punt de començar-se a comunicar-se entre ells fent que apareixes el termini HIS.

L’any 2005, l’OMS [8] va començar a posar normes perquè els HIS funcionessin de manera correcta, tinguessin la seguretat esperada i a més van predir que l'any 2015 tots els hospitals haurien de tenir-ne un. Segons Gillum [3], l'any 2011, més del 50 % dels metges van reportar que utilitzaven un HIS. A Amèrica del Nord el primer hospital completament digital va ser a Toronto l’any 2015.

## ANTECEDENTS

Els antecedents que es busquen en aquest apartat són aplicacions internes dels hospitals i que, preferiblement, es comuniquin amb el motor d’integració.

Durant la recerca d’aquests antecedents només es trobaven aplicacions públiques, i que per tant no són gaire útils per la nostra aplicació. La majoria d’aquestes són bases de dades de malalties. Un exemple d’aquestes aplicacions són UpToDate [9] o Dynamed [10].

Les aplicacions que són antecedents per l’aplicació que es farà han sigut impossible de trobar, ja que no hi ha aplicacions d’ús intern que siguin públiques.

## NECESSITATS DE LA INFORMACIÓ

Per poder entendre i desenvolupar l’aplicació s’han de saber algunes coses bàsiques del funcionament dels hospitals. Primer s’ha de saber que és l'HIS, quins components té i perquè és tan important. Després s’ha de saber que és el motor d’interoperabilitat, ja que és el pilar fonamental de l’aplicació a fer. Tot seguit s’ha de saber quins protocols usen l'HIS i el motor d’interoperabilitat per a comunicar-se, ja que pot ser bastant important a l’hora d’entendre algunes dades que pot rebre el producte. Per acabar es necessita saber quines dades s’emmagatzemen a l’HIS i es pot tenir accés amb el motor d’interoperabilitat. La majoria d’aquesta informació ha estat extreta del llibre anomenat Health Information Systems - Architectures and Strategies [11].

### HEALTHCARE INFORMATION SYSTEM (HIS)

El gran nucli d’informació que hi ha en un hospital és el Healthcare o Health Information System (HIS). Aquest sistema és l’encarregat de guardar i processar tota la informació que genera un hospital.

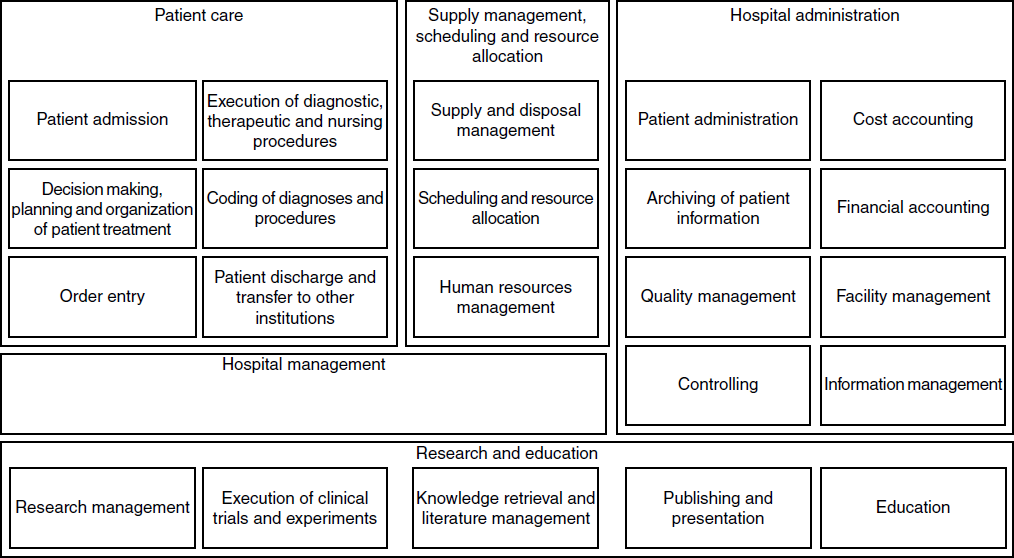


Figura .: Esquema de les dades que emmagatzema l'HIS [12]

En aquest esquema, es pot veure totes les dades que tractarà l'HIS, però pot observar-se que falten molts serveis que ofereix tant un hospital com el CAP, per exemple el servei de pediatria i de cardiologia d’entre d’altres. Tots aquests serveis que no es veuen a la imatge, formen part de l’HIS, però són un subsistema d’aquest.

En definitiva, l'HIS es pot definir com un conjunt de subsistemes d’informació dels diferents serveis que ofereix tota institució mèdica que emmagatzema, processa i pren decisions sobre les dades rebudes.

### MOTOR D’INTEROPERABILITAT

Perquè l'HIS pugui funcionar correctament, necessita dades d’altres hospitals per poder tenir tota la informació completa. Aquesta comunicació entre diferents HIS d’hospitals la proporciona el motor d’interoperabilitat que té cada institució mèdica. A més aquest motor d’interoperabilitat pot generar una API perquè aplicacions externes al HIS, com aplicacions web puguin comunicar-se. Per tant el motor d’interoperabilitat és l’encarregat de comunicar-se amb els elements externs al HIS que controla.

Aquest motor d’interoperabilitat és la base de l’e-health, ja que sense ell costaria molt fer una comunicació bona entre diferents HIS. A Espanya, segons un estudi fet per la Healthcare Information and Management Systems Society (HIMSS) l’any 2014 [13], el 93 % dels hospitals tenen un motor d’interoperabilitat integrat.

### PROTOCOLS ENTRE EL HIS I EL MOTOR D’INTEROPERABILITAT

Aquests protocols o estàndards que es mostraran a continuació també poden ser usats per la interoperabilitat dels HIS.

##### HL7

Health Level 7 és l’estàndard més usat per a la comunicació entre el HIS i el motor d’interoperabilitat, com entre motors d’interoperabilitat. Aquest estàndard envia missatges a un destinatari, però no inclou imatges [14, p. 2].

Aquest protocol té dues versions, la versió 2 i la versió 3. La principal diferencia que tenen aquestes versions és que la més nova, la versió 3, genera un missatge més fàcil de llegir, gràcies a la implementació del HL7 RIM (Reference Information Model). Per tant com es pot veure en les següents imatges, es pot veure que és molt més clar llegir un missatge de la versió 3, que de la versió 2.

El principal problema que tenen aquestes dues versions és que no són compatibles entre elles. Per tant si s’implementa alguna solució usant una de les dues versions, és difícil canviar-la [15].

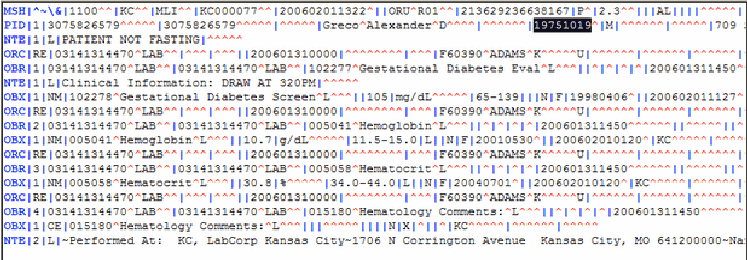


Figura .: Exemple de missatge de HL7 v2 [16, p. 7].

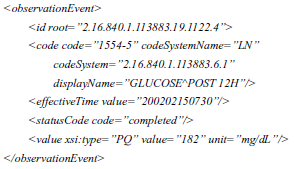


Figura .: Exemple del missatge de HL7 v3

##### DICOM

L’estàndard Digital Imaging and Communications in Medicine (DICOM) és el que gestiona l’intercanvi d’imatges que genera el HIS. Com pot ser una radiografia, una ressonància magnètica, etc.

A diferència del HL7, aquest estàndard necessita fer una connexió TCP-IP per a enviar la imatge. Aquest estàndard codifica i comprimeix la imatge per a facilitar el transport i la seguretat.



Figura .: Exemple d’una imatge generada pel protocol DICOM [17].

##### ISO/IEEE 11073

Aquest protocol, creat per la família d’estàndards internacional, és el que permet poder intercanviar senyals vitals o biomèdiques. Aquest protocol ofereix un conjunt de tecnologia per poder connectar-ho a un pacient i que funcioni sense cap configuració extra. És un protocol molt útil per al monitoratge dels pacients.

# OBJECTIUS I ABAST

En aquest apartat s’explicaran els objectius que té aquest treball, tant pel client com pel producte. A més s’inclourà quin és el públic objectiu i l’abast d’aquesta aplicació.

## OBJECTIUS DEL CLIENT

1. Demostrar que amb l’aplicació millora l’efectivitat de visualitzar un esdeveniment.
2. Fer una aplicació multiplataforma per arribar al màxim nombre d’usuaris.
3. Gestionar les alertes que disposin els usuaris a través d’una pàgina web.
4. Rebre notificacions quan s’ha superat el llindar del valor establert en l’alerta.
5. Integrar l’aplicació a l’hospital sense que suposi una dificultat extra pel client.

## OBJECTIUS DEL PRODUCTE

1. Dissenyar un sistema per a poder afegir hospitals i diferents motors d’interoperabilitat.
2. Tenir una seguretat adient d’una aplicació mèdica.
3. Evitar que la comunicació entre el servidor i l’aplicació mai es perdi per problemes del producte.
4. Seguir les lleis de protecció de dades dels usuaris.

## PÚBLIC POTENCIAL

Aquesta aplicació està pensada per a ser utilitzada per a qualsevol treballador d’un hospital o institució mèdica que demani accés al producte. Això vol dir que si estàs donat d’alta i el mateix hospital t’ha donat un usuari i contrasenya de l’aplicació, es podrà accedir i per tant usar-la.

## ABAST

Aquest producte està pensat per a ser utilitzat per a qualsevol hospital que vulgui usar aquesta aplicació independentment de la empresa del seu motor d’interoperabilitat.

Dit això, no vol dir que el disseny de la base de dades i del servidor no s’hagi pensat en el cas que es puguin afegir noves institucions mèdiques o usar altres empreses que disposin d’un motor d’interoperabilitat. Per tant l’abast d’aquest projecte pot ser tant “petit” com centrar-se en un únic motor d’interoperabilitat i un únic hospital, com fins a convertir-se en una aplicació que s’utilitza globalment.

# METODOLOGIA

Aquest TFG tindrà un procés cíclic en la part del producte. Com es pot saber utilitzar una metodologia *scrum* sent només un desenvolupador pot ser difícil, per tant s’usarà una petita variant. El principal objectiu d’aquesta metodologia és que en cada *sprint*, que serà d’una setmana, l’aplicació afegeixi funcionalitats sense trencar les anteriors.

En el control de versions que s’utilitzarà, hi haurà tres branques per defecte, *master*, *develop* i *documentation*. A part quan s’hagi d’afegir una funció al producte, es crearà una branca amb les sigles de la fita que li correspongui i el seu nom. Per exemple cs\_login, en el cas de fer la funció del servidor web d’iniciar sessió.

A més aquest projecte tindrà tres fites relacionades amb les entregues que hi ha de la documentació del projecte.

La primera ve relacionada amb l’entrega de l’avantprojecte, en aquesta fita s’haurà de dissenyar tot el producte. Aquest disseny ha de constar els diagrames UML de l’aplicació i del servidor, el diagrama relacional de la base de dades, el flux de l’aplicació i com es veurà visualment. Aquesta fita suposarà la transició de la preproducció del producte i l’inici de la producció.

La segona fita començarà la producció del servidor i està marcada amb l’entrega de la memòria intermèdia. Aquesta fita se separarà en tres *sprints* i, a arribar la data de la fita, s’haurà de tenir el servidor IRIS i el servidor Web. Aquests tres *sprints* corresponen al servidor web i es farà a tres setmanes de la data límit de la fita. Abans de fer aquesta web es crearà el servidor d’IRIS. El motiu de què, la producció del servidor IRIS, no estigui tallant en *sprints* és perquè en el moment de l’elaboració d’aquest apartat no se sap quan es trigarà a fer-ho. Així que s’ha deixat una longitud de temps equivalent a set *sprints*, per evitar inconvenients al futur.

La tercera i última fita coincideix amb la data de l’entrega de la memòria final i servirà per desenvolupar l’aplicació. Aquesta última fita estarà composta de set *sprints* que aniran des de connectar-se amb el servidor web fins a fer la interfície d’usuari.

# DEFINICIÓ DE REQUERIMENTS FUNCIONALS I TECNOLÒGICS

Aquest projecte necessita separar els requeriment funcionals del servidor i de l’aplicació, ja que són bastants diferents.

## REQUERIMENTS FUNCIONALS DEL SERVIDOR WEB

1. **Iniciar sessió al servidor web.**
2. **Tancar sessió al servidor web.**
3. **Crear noves institucions mèdiques.**
4. **Crear usuaris per l’aplicació i servidor.**
5. **Crear alertes per l’aplicació.**
6. **Crear rols per l’aplicació.**
7. **Assignar o eliminar alertes als rols.**
8. **Esborrar rols.**
9. **Esborrar institucions mèdiques.**
10. **Esborrar usuaris.**
11. **Poder visualitzar la llista d’usuaris, de rols, d’alertes i d’institucions mèdiques.**
12. **Poder filtrar la llista d’usuaris, de rols, d’alertes i d’institucions mèdiques.**
13. **Poder buscar en la llista d’usuaris, de rols, d’alertes i d’institucions mèdiques.**

Per provar que tots aquests requeriments funcionin, farà falta un ordinador amb accés a internet i que pugui obrir un navegador web.

## REQUERIMENTS FUNCIONALS DE L’APLICACIÓ

1. **Iniciar sessió a l’aplicació.**
2. **Tancar sessió a l’aplicació.**
3. **Poder visualitzar la llista d’alertes per usuari.**
4. **Buscar l’alerta desitjada.**
5. **Poder canviar la contrasenya del seu compte.**
6. **Poder demanar l’eliminació de l’usuari.**
7. **Rebre notificacions quan canvi de gravetat l’alerta.**

Per provar que tots aquests requeriments funcionin, farà falta un dispositiu mòbil amb accés a internet. El sistema operatiu ha de ser Android o iOS.

## REQUERIMENTS TECNOLÒGICS

Els requeriments tecnològics per poder executar, tant l’aplicació com el servidor web són bastant simples, en àmbit d’usuari.

Per començar el servidor web només té un requeriment tecnològic. Tenir un ordinador o dispositiu capaç de connectar-se a una pàgina web. Preferiblement que sigui un ordinador o que tingui una pantalla en mode horitzontal per a poder visualitzar bé les dades. Encara que s’intentarà que el servidor web es pugui visualitzar en els dispositius amb pantalles més petites.

Els requeriments tecnològics de l’aplicació és tenir un dispositiu mòbil, amb el sistema operatiu Android o iOS. Aquest dispositiu hauria de disposar de bona autonomia i connexió a internet.

En el cas de tractar-se del client, aquest necessita un requeriment tecnològic més que és disposar d’un servidor físic que tingui el HIS i el motor d’interoperabilitat per poder accedir a les dades d’aquell hospital.

# DESENVOLUPAMENT

En aquesta secció s’explicarà tota la producció del producte a produir en aquest TFG. Aquesta secció s’ha separat en quatre subapartats, que corresponen a les quatre tasques principals per fer aquest producte: disseny, servidor d’IRIS, servidor web i aplicació.

## DISSENY

Abans de començar a la producció del producte s’ha de dissenyar alguns aspectes. L’element principal a dissenyar és la base de dades, ja que serà la que unifiqui l’aplicació i els dos servidors. A més s’ha de dissenyar bé el servidor web i l’aplicació, per això s’han elaborat uns diagrames de classe UML i diagrames de flux BPMN. A més en el producte s’ha dissenyat una UI, tant pel servidor web com per l’aplicació mòbil.

### BASES DE DADES

Per poder fer la base de dades s’ha pensat en el futur i per si l’aplicació creix significativament.

Així que aquest diagrama relacional, té constància de les diferents institucions mèdiques que puguin usar l’aplicació. Per a fer això, per a cada institució mèdica es guarda la seva ubicació: el país on està situada i el seu nom, per exemple “Hospital de Mataró”. Per acabar es necessiten les dades perquè l’aplicació mòbil pugui accedir a l’API que es generarà al seu motor d’interoperabilitat. D’aquestes dades de l’API, només es guardarà l’URL, és a dir, la part comuna de totes les API que tingui aquella institució. Per exemple, com es veurà més endavant, aquest URL pot ser “https://www.tfginformatica.dev/rest/”. A més s’haurà de tenir desat per a cada institució les credencials per a poder accedir a aquella API.

A continuació hi ha la taula de rols, aquests rols poden ser els diferents serveis o grups de treballadors de cada institució mèdica que facin ús de les mateixes alertes. Per exemple, pot haver-hi un rol que sigui “Infermeria”. Aquest rol tindrà un conjunt d’alertes exclusives per aquest rol i uns empleats.

En la taula d’empleats, només es tindrà constància del seu rol, les seves credencials d’accés al producte i, per acabar, algunes dades personals, com pot ser el nom i cognoms.

En la taula d’alertes es necessita saber a quin rol pertany, i a continuació com poder accedir a l’URL de l’API especifica per aquell rol. En aquest cas ha de ser una continuació de l’URL proporcionat per la institució mèdica. Per acabar l’alerta també ha de saber quan un valor és correcte o incorrecte i també ha de tenir dades per a l’aplicació mòbil.

Aquesta alerta tindrà guardat un historial dels valors que ha proporcionat en el passat.

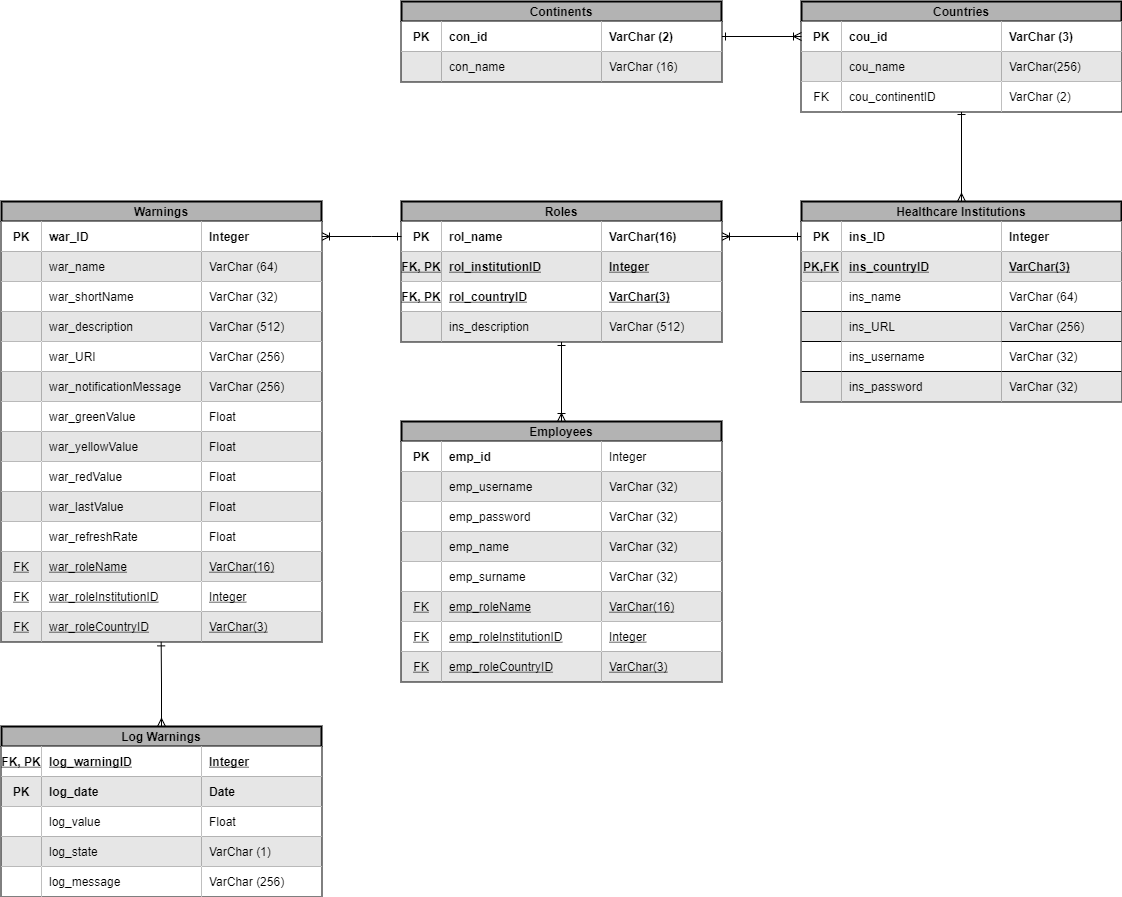


Figura .: Diagrama ERD de la base de dades.

### SERVIDOR WEB

El servidor estarà fet usant el *framework Spring*. El motiu d’utilitzar aquest framework és perquè és bastant utilitzat en el món de les pàgines web, és fàcil de trobar recursos que ajudin el desenvolupament d’un servidor web i, per acabar, utilitza el llenguatge de programació Java.

Per a poder programar aquest servidor, s’utilitzarà l’IDE IntelliJ de JetBrains. Aquest IDE és l’escollit, ja que és un dels IDE més utilitzats per a programar en Java i a més té una gran oferta d’IDE de diferents llenguatges que fa que canviar d’un llenguatge a un altre no suposi un canvi molt gran. Ja que les dreceres del teclat són comunes en tots els IDE d’aquesta empresa.

El servidor web serà l’encarregat de gestionar la configuració de l’aplicació. Això significa que serà l’únic que tindrà accés a la base de dades. Així que qualsevol canvi, o inserció d’alguna fila a la base de dades ho haurà de fer el servidor.

A més el servidor serà l’encarregat d’enviar les notificacions al mòbil dels usuaris. Per a poder fer això el servidor tindrà un *thread* per a cada alerta que hi hagi i aquest *thread* anirà informant els usuaris que estiguin assignats a aquesta alerta de si hi ha hagut un canvi de color. En el cas que hi hagi, s’enviarà una notificació a l’usuari.

#### DIAGRAMES

La part principal per a poder començar a programar és saber quin disseny i com es comportarà el servidor. Així que s’ha elaborat un diagrama de classes i un parell de diagrames BPMN.

##### DIAGRAMA UML DE CLASSES

Aquest diagrama mostrà com estaran comunicades les classes i a més quins mètodes i atributs tindrà.

És un diagrama de classe senzill, ja que el *framework Spring* proporciona tota la part de connectar-se a la pàgina web. D’aquesta manera només s’ha de gestionar del funcionament de la web.

Les classes que estan més amunt són les encarregades de la base de dades. La primera, DatabaseDAO, és la que farà les crides i la segona, DatabaseController, farà de façana.

Després hi ha els controladors web, el primer, APIController, serà el que gestioni l’API, que serà en la que es comunicarà l’aplicació. Les dues següents seran les que gestionin la web i estan separades en les que només ensenyen una pàgina HTML, WebGetController, i les que afegeixen dades a la base de dades, WebPostController.

Després hi haurà la classe TFGApplication, que és on està el main i és qui fa executar el servidor web. Tot seguit apareix la classe NotificationController, que serà qui creí les alertes i enviï les notificacions als usuaris de l’aplicació mòbil.

Per acabar hi ha les classes del domini, on es crearan els objectes, institució mèdica (HealthcareInstitution), rol (Role), empleat (Employee) i alerta (Warning).

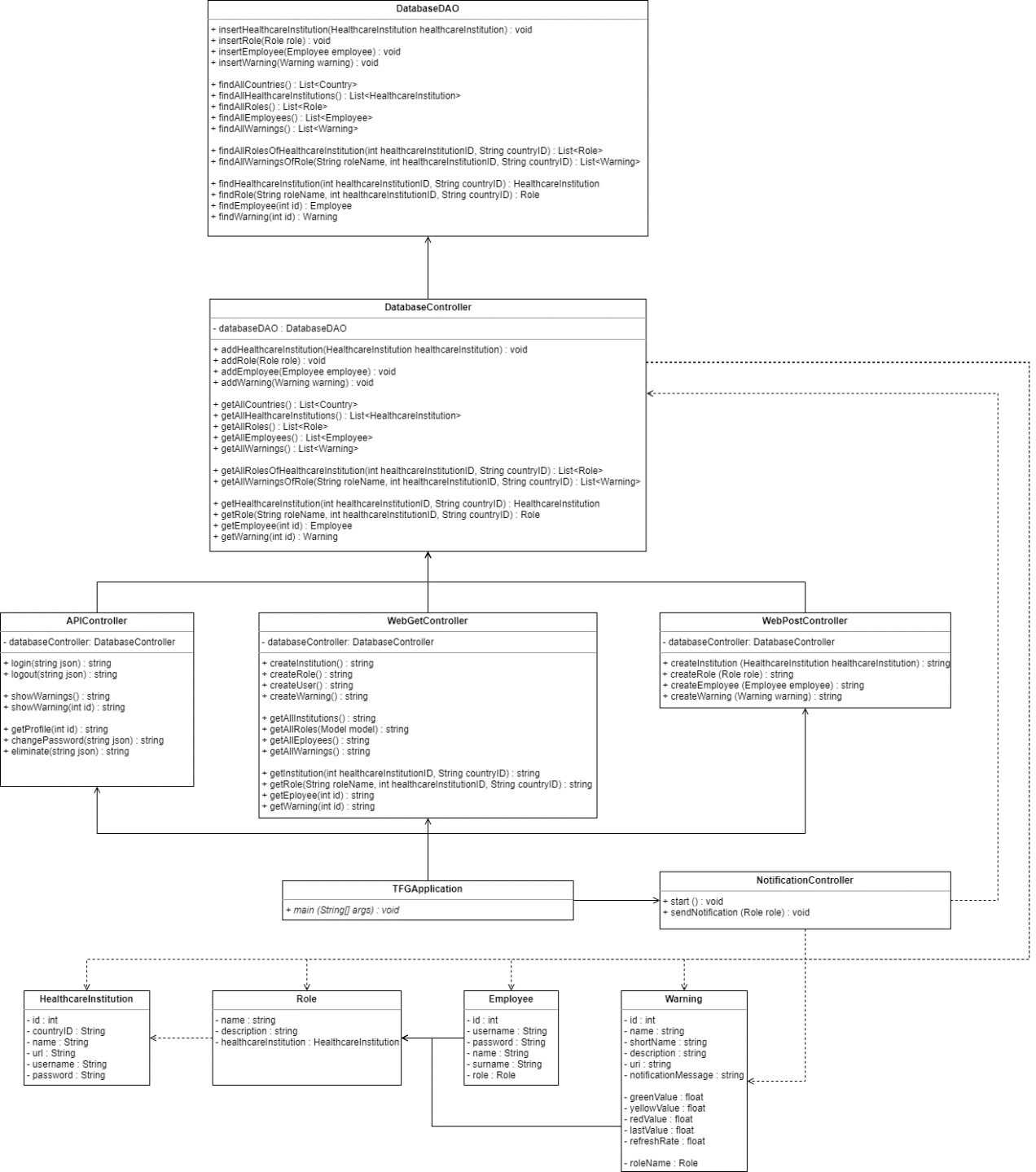


Figura .: Diagrama UML de classes del servidor web.

##### DIAGRAMES BPMN

A continuació es mostraran dos diagrames BPMN. El primer és per mostrar com funciona una creació d’una institució mèdica. S’ensenyarà només la creació, ja que per a tots els elements que utilitzen la base de dades des del servidor té el mateix funcionament.

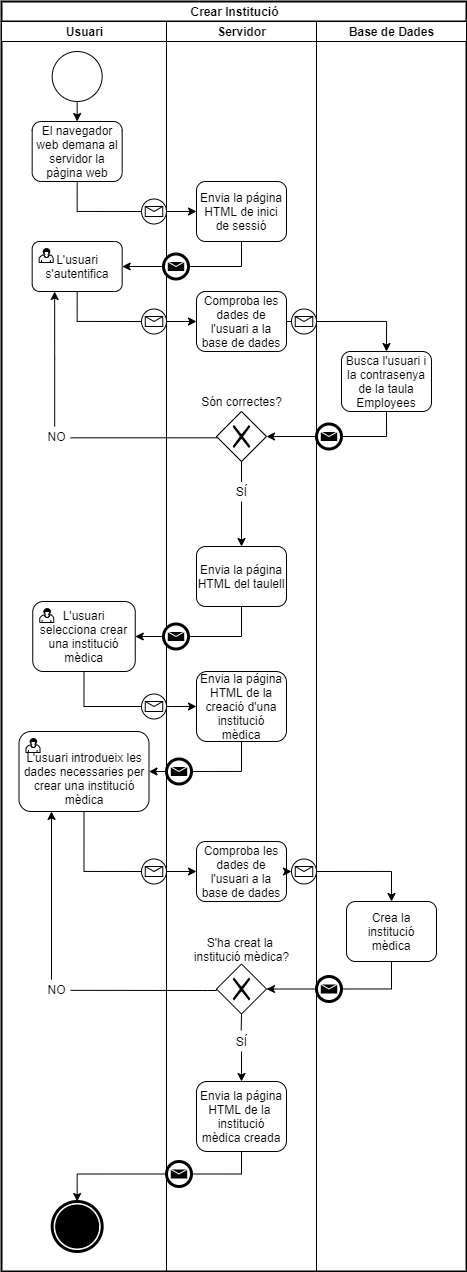


Figura .: Diagrama BPMN de la creació d’una institució mèdica.

El segon diagrama és referent a com es gestionen les notificacions. Com el diagrama anterior, només es mostrarà com funciona en una alerta. Així que la columna que es diu alerta n’hi haurà tantes alertes com alertes hi hagi a la base de dades.

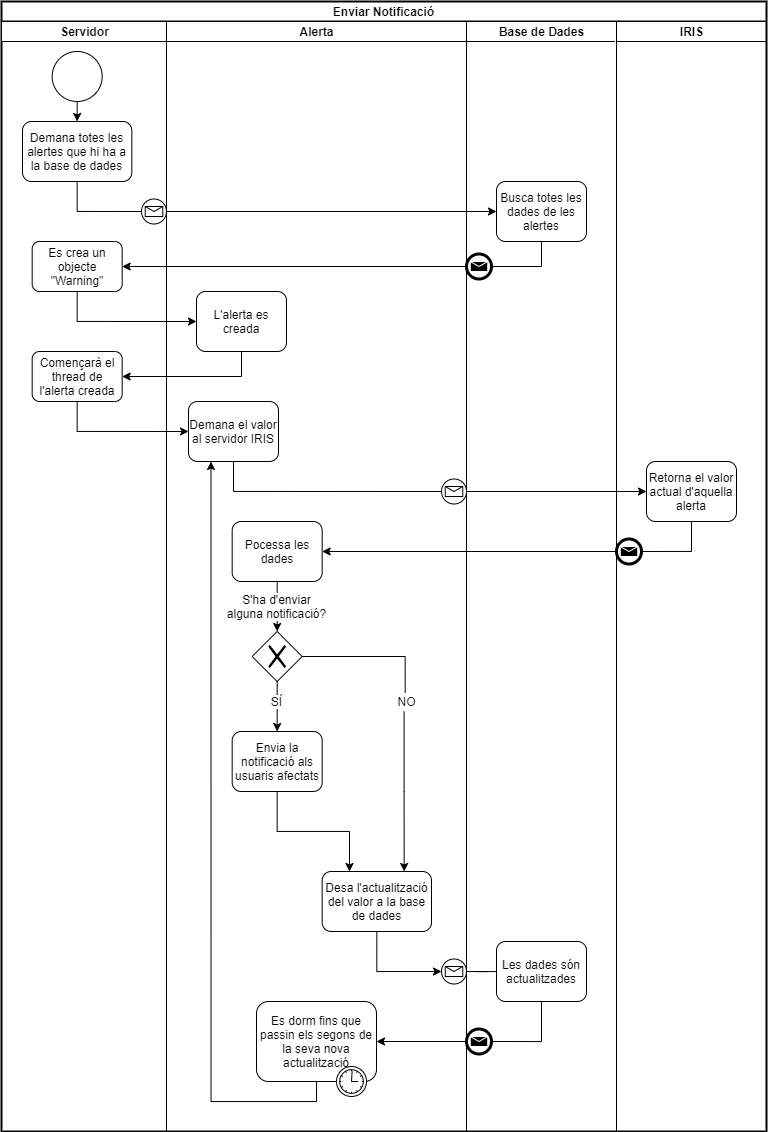


Figura .: Diagrama BPMN de la gestió de les notificacions.

#### UI

Per la UI s’han pensat en colors que fossin semblants als mèdics, aquests colors, després de buscar algunes referències, s’ha optat per un blau cel i blanc. En l’aplicació també seran els mateixos colors.

Per poder donar contrast en la creació o eliminació de dades, s’ha optat pel color verd i vermell respectivament.

Per a diferenciar els camps de textos amb els textos no editables, hi haurà un rectangle blanc i el text serà en negre. Com es pot veure a la imatge següent.



Figura .: Disseny de la UI de l’inici de sessió en el servidor web.

Una vegada iniciada la sessió es mostrarà la capçalera de la pàgina web, que constarà de quatre botons de navegació per a facilitar l’ús de l’aplicació. A més hi haurà un peu de pàgina amb algunes dades referent al TFG.

El següent disseny és de la pàgina d’inici, una vegada s’hagi iniciat sessió. S’ha pensat a posar botons per les llistes i creació.



Figura .: Disseny de la UI de la pàgina d’inici en el servidor web.

A continuació es veurà com són els dissenys per llistar, crear i veure les dades individualment. Per tal de fer-ho més senzill, només es mostraran els dissenys de les institucions mèdiques.



Figura .: Dissenys de la UI de la llista d’institucions mèdiques en el servidor web.

La llista d’institucions mediques tindrà un format de taula. A la capçalera d’aquesta taula hi haurà les eines de filtratge al costat del nom de la columna. Per diferenciar entre les files i columnes s’ha optat a fer un degradat amb un color de la mateixa família cromàtica que el blau cel, però, més fosc.

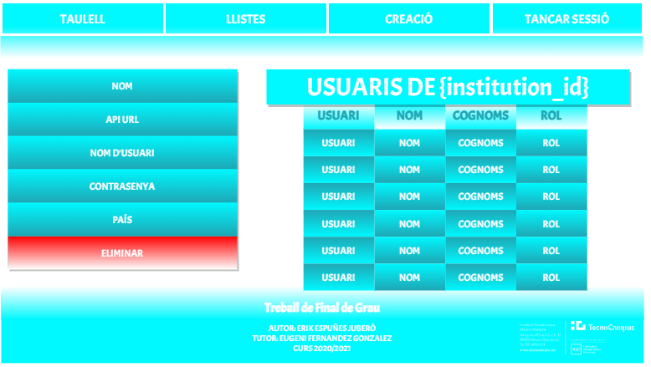


Figura .: Disseny de la UI de la visualització d’una institució mèdica en el servidor web.

A continuació es mostra la visualització d’una institució mèdica en detall. Aquí hi haurà totes les dades necessàries i es mostraran els usuaris que pertanyen a la institució mèdica. També es podran esborrar o editar les dades de la institució que s’estigui veient.



Figura .: Disseny de la UI de la creació d’una institució mèdica en el servidor web.

Per acabar hi ha el disseny de la creació d’una institució mèdica. Aquí hi haurà tots els camps necessaris per a poder crear la institució. Totes les altres creacions de dades, tindrà un aspecte molt semblant.

### APLICACIÓ MÒBIL

L’aplicació estarà feta usant el *framework React Native*. El motiu d’utilitzar aquest framework és perquè és bastant utilitzat en les majories d’aplicacions importants en l’àmbit del mòbil, com per exemple Instagram. A més és multiplataforma, això vol dir que no s’ha de fer el codi dues vegades, ja que és compatible amb iOS i Android. *React Native*, com també el seu pare *React*, utilitza el llenguatge de programació JavaScript.

Per a poder programar aquesta aplicació, s’utilitzarà l’IDE WebStorm de JetBrains. S’ha escollit aquest IDE perquè és un IDE de l’empresa JetBrains específic per JavaScript. I com s’ha comentat anteriorment la transició de diferents IDE de la mateixa empresa no suposa un canvi molt gran, respecte a la UI i les dreceres de teclat.

L’aplicació mòbil serà l’encarregada d’avisar a l’usuari que ha succeït alguna anomalia en alguna de les alertes que té assignades. A més serà d’ús propi per l’usuari, això vol dir que ho podrà consultar sempre que vulgui.

#### DIAGRAMES

A l’hora de poder començar a programar l’aplicació, és necessari saber com estaran organitzades les classes. A més s’ha de pensar bé el flux que tindran alguns casos d’ús, que es veuran en el següent subapartat, per a fer una bona gestió dels servidors web i d’IRIS.

##### DIAGRAMA UML DE CLASSES

Aquest diagrama UML està orientat a les classes o scripts que hi haurà en el desenvolupament de l’aplicació amb *React Native.*

L’element clau d’aquest diagrama són les classes que es comunicaran amb el servidor web i el servidor d’IRIS o de l’API corresponent. S’ha pensat que per millorar la seguretat de l’aplicació i del tractament de dades, de posar el patró façana. Així que les escenes, que són les pantalles que es veuran en el mòbil, es comunicaran directament amb una classe, que és el CommunicationManager, i aquesta distribuirà o gestionarà la comunicació amb el servidor web i l’API corresponent.

En la classe Controller, pot saltar a la vista l’únic mètode anomenat waitForResponse. Aquest mètode té la funció de bloquejar aquella classe fins que no arribi una resposta de l’API. És necessari fer un bloqueig, ja que les crides a una API són asíncrones, així que no se sap quan pot trigar a arribar les dades necessàries. Així que com és un mètode comú entre WebServerController i APIController, s’ha fet que aquestes dues classes siguin filles de la classe Controller.

Per acabar les classes d’escena seran com si fossin una pàgina HTML, així que només s’encarregaran de la part gràfica de l’aplicació i de rebre les comandes directament de l’usuari. Com aquestes classes tenen una variable i un mètode en comú, s’ha pensat en què han d’estendre d’una classe abstracta, anomenada Scene, per a tenir un bon disseny de l’aplicació i per no repetir codi innecessari.

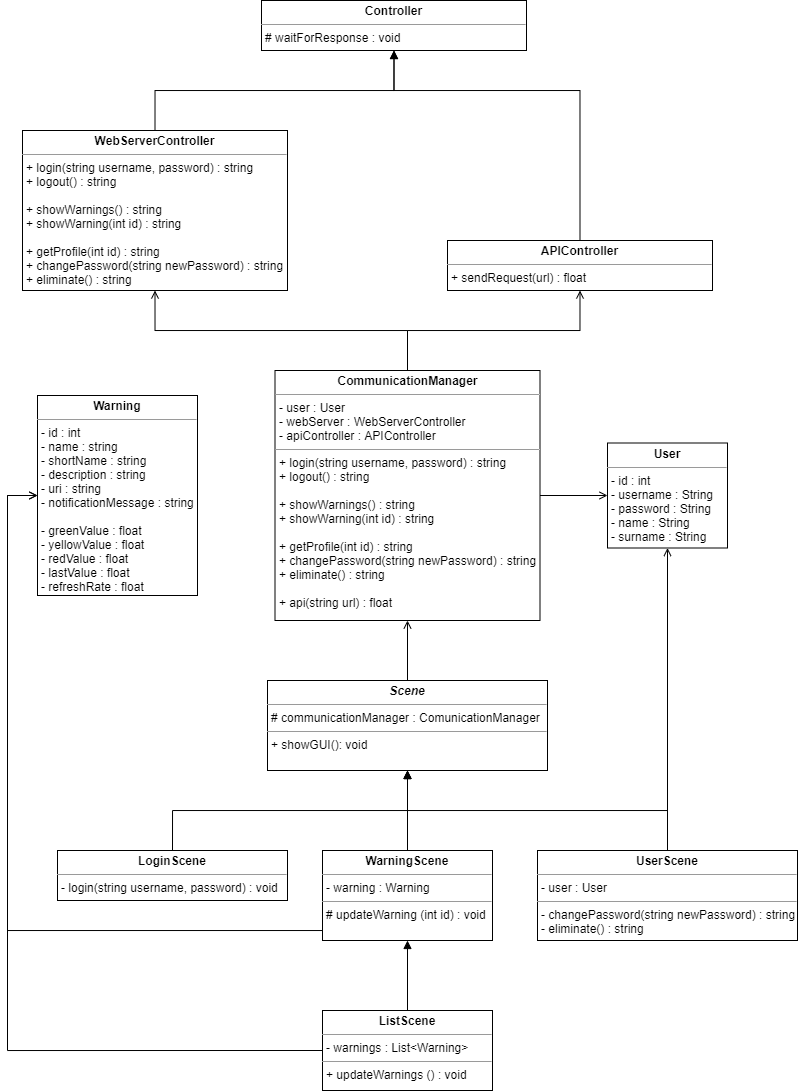


Figura .: Diagrama UML de classes de l’aplicació mòbil.

##### DIAGRAMES BPMN

A continuació es veuran tres diagrames BPMN.

El primer d’ells és per saber com funciona l’inici de sessió en l’aplicació. Aquest diagrama és comú per a tots aquells casos d’ús que no fa falta una comunicació directa amb IRIS. Per tant els casos d’ús de tancar sessió, canviar la contrasenya i demanar l’eliminació del compte seran semblants al següent diagrama.

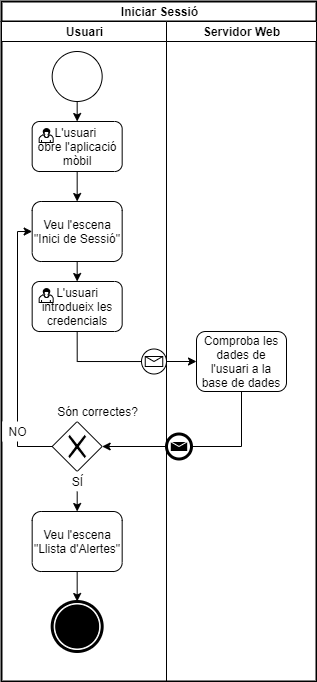


Figura .: Diagrama BPMN de l’inici de sessió en l’aplicació.

El segon diagrama és sobre la llista d’alertes. Aquesta llista, a part de comunicar-se amb el servidor web, s’ha de comunicar amb el servidor d’IRIS o de l'API que proporcioni la institució mèdica.

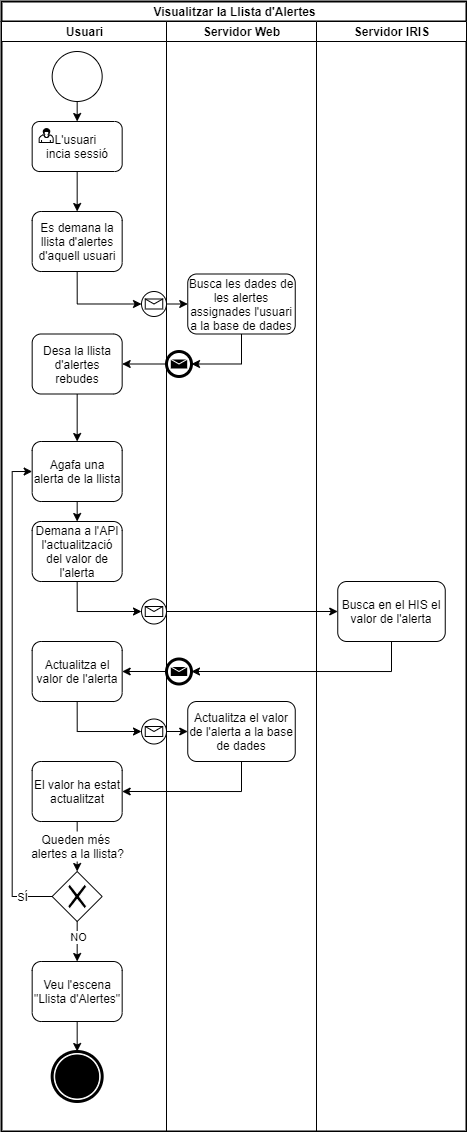


Figura .: Diagrama BPMN de la visualització de la llista d’alertes en l’aplicació.

El tercer i últim diagrama és sobre la visualització d’una alerta. És un diagrama bastant semblant a l’anterior però sense la iteració en la llista, ja que només s’actualitzarà una alerta.

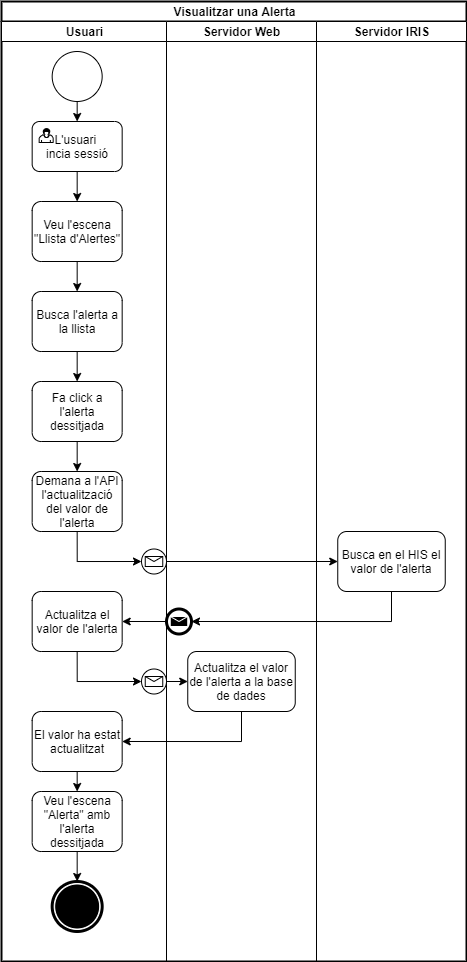


Figura .: Diagrama BPMN de la visualització d’una alerta en l’aplicació.

#### CASOS D’ÚS

En el cas de l’aplicació mòbil s’han elaborat tres casos d’ús per explicar el funcionament i el que ha de fer l’usuari per a poder utilitzar l’aplicació. Els tres casos d’ús que es veuran a continuació son les mateixes funcionalitats que s’ha vist en els tres diagrames BPMN.

|  |  |
| --- | --- |
| INICI DE SESSIÓ | |
| DESCRIPCIÓ | Aquest cas d’ús tracta de com s’inicia sessió a l’aplicació mòbil. |
| ACTOR | Qualsevol usuari amb accés a l’aplicació. |
| PRECONDICIONS | — |
| GARANTIES D’ÈXIT | — |
| FLUX BÀSIC | 1. L’usuari accedeix a l’aplicació mòbil. |
| 2. L’usuari veu l'escena “Inicia Sessió”. |
| 3. L’usuari introdueix l’usuari. |
| 4. L’usuari introdueix la contrasenya. |
| 5. L’usuari fa clic al botó d'“Iniciar Sessió”. |
| 6. L’usuari ha iniciat sessió. |
| 7. L’usuari veu l'escena “Llista d’Alertes”. |
| FLUX ALTERNATIU | 5A. La base de dades no troba el nom d’usuari. Torna al pas 2 explicant a l’usuari el motiu perquè no s’ha pogut iniciar sessió. |
| 5B. La contrasenya escrita per l’usuari no correspon a la que està a la base de dades. Torna al pas 2 explicant a l’usuari el motiu perquè no s’ha pogut iniciar sessió. |
| 5C. No s’ha pogut establir connexió amb el servidor web. Torna al pas 2 explicant a l’usuari el motiu perquè no s’ha pogut iniciar sessió. |
| 5D. No hi ha connexió amb la base de dades. Torna al pas 2 explicant a l’usuari el motiu perquè no s’ha pogut iniciar sessió. |
| REQUISITS ESPECIALS | Disposar de connexió a internet. |
| LLISTA DE TECNOLOGIA I VARIACIONS DE DADES | Telèfon intel·ligent, amb sistema operatiu Android o iOS, que disposi d’accés a internet. |

Taula 6.1: Cas d’ús d’inici de sessió en l’aplicació mòbil.

|  |  |
| --- | --- |
| VISUALITZACIÓ DE LA LLISTA D’ALERTES | |
| DESCRIPCIÓ | Aquest cas d’ús tracta de com es visualitza la llista d’alertes, una vegada s’ha iniciat sessió, a l’aplicació mòbil. |
| ACTOR | Qualsevol usuari registrat. |
| PRECONDICIONS | Haver executat el cas d’ús “Inici de sessió”. |
| GARANTIES D’ÈXIT | — |
| FLUX BÀSIC | 1. L’usuari veu l'escena “Llista d’Alertes”. |
| FLUX ALTERNATIU | 1A. S’ha perdut la connexió amb el servidor web. Torna al pas 1 explicant a l’usuari el motiu perquè no s’ha pogut visualitzar la llista d’alertes. |
| 1B. No hi ha connexió amb la base de dades. Torna al pas 1 explicant a l’usuari el motiu perquè no s’ha pogut visualitzar la llista d’alertes. |
| 1C. No s’ha trobat cap alerta per l’usuari. Torna al pas 1 explicant a l’usuari el motiu perquè no s’ha pogut visualitzar la llista d’alertes. |
| 1D. No s’ha pogut establir connexió amb el servidor IRIS. Els valors actuals de l’alerta no estaran actualitzats. |
| REQUISITS ESPECIALS | Disposar de connexió a internet. |
| LLISTA DE TECNOLOGIA I VARIACIONS DE DADES | Telèfon intel·ligent, amb sistema operatiu Android o iOS. |

Taula 6.2: Cas d’ús de la llista d’alertes en l’aplicació mòbil.

|  |  |
| --- | --- |
| VISUALITZACIÓ DETALLADA DE L’ALERTA | |
| DESCRIPCIÓ |  |
| ACTOR | Qualsevol usuari registrat. |
| PRECONDICIONS | Haver executat el cas d’ús “Inici de sessió”. |
| Haver executat el cas d’ús “Visualització de la llista d’alertes”. |
| GARANTIES D’ÈXIT | — |
| FLUX BÀSIC | 1. L’usuari veu l'escena “Llista d’Alertes”. |
| 1. L’usuari cerca l’alerta desitjada. |
| 1. L’usuari clica a sobre de l’alerta desitjada. |
| 1. L’usuari veu l'escena “Alerta”. |
| **FLUX ALTERNATIU** | 2A. L’usuari no troba l’alerta desitjada. Torna al pas 1. |
| 3A. S’ha perdut la connexió amb el servidor web. Torna al pas 1 explicant a l’usuari el motiu perquè no s’ha pogut visualitzar l’alerta. |
| 3B. No hi ha connexió amb la base de dades. Torna al pas 1 explicant a l’usuari el motiu perquè no s’ha pogut visualitzar l’alerta. |
| 3C. No s’ha pogut establir connexió amb el servidor IRIS. Els valor actual de l’alerta no estarà actualitzat. |
| **REQUISITS ESPECIALS** | Disposar de connexió a internet. |
| **LLISTA DE TECNOLOGIA I VARIACIONS DE DADES** | Telèfon intel·ligent amb sistema operatiu Android o iOS. |

Taula 6.3: Cas d’ús de la visualització d’una alerta en l’aplicació mòbil.

#### UI

Per la UI de l’aplicació s’ha optat pel mateix estil que el servidor web, així que tindrà un fons blanc i els elements on hi hagi les alertes o els camps on introduir dades serà en un color blau cel.

Per a poder representar bé els valors de les alertes, s’ha pensat a fer-ho com si fos un semàfor. Així que quan es vegi el llistat d’alertes, al costat del nom de l’alerta es veurà un quadrat amb el valor i el color, si és correcte, verd, si no és un valor òptim, groc, i si és un valor perillós, vermell.

La llista d’alertes serà l’escena per defecte en l’aplicació, una vegada s’hagi iniciat correctament l’aplicació.

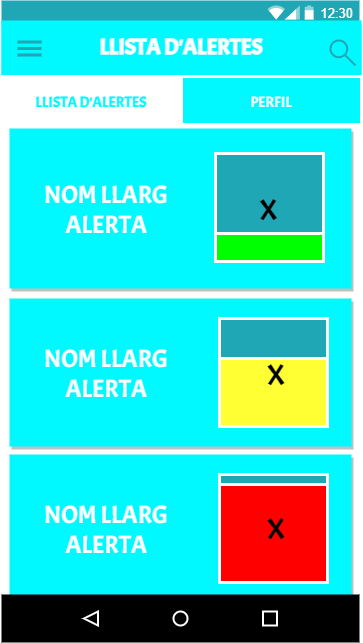


Figura .: Disseny de la UI de la llista d’alertes en l’aplicació mòbil.

Si es fa clic a sobre d’alguna alerta d’aquesta llista, es podrà veure la descripció d’aquesta alerta. A més, de nou, es podrà veure el color i el valor que té aquella alerta en aquell moment.

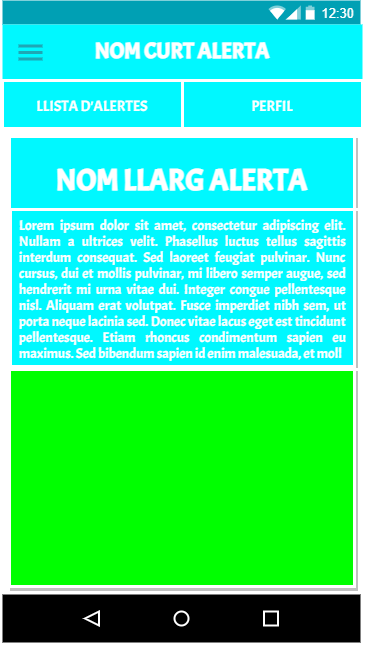


Figura . Figura .: Disseny de la UI de la visualització d’una alerta en l’aplicació mòbil.

## SERVIDOR IRIS

El primer pas en el procés de producció d’aquest TFG, és crear el servidor on estarà el motor d’interoperabilitat Intersystems IRIS for Health.

### CONFIGURACIÓ

Per poder començar a implementar el servidor d’IRIS, primer s’ha de configurar l’entorn de desenvolupament. A l’inici s’ha creat una màquina virtual amb el programa Hyper-V, que ja ve instal·lat amb el sistema operatiu Windows 10.

Aquesta màquina virtual té el sistema operatiu de Windows Server 2019. El motiu d’elegir Windows Server en comptes d’algun altre sistema operatiu que sol ser més usat per a fer servidors ha sigut per la corba d’aprenentatge. Amb el sistema operatiu Windows Server, la instal·lació del motor d’interoperabilitat IRIS for Health era bastant senzilla, ja que l’empresa et proporciona un instal·lador. El problema que hagués sorgit amb un altre sistema operatiu com pot ser Linux, és que aquesta instal·lació s’hagués hagut de fer a través d’un Docker. Això és un problema, ja que Windows, o almenys els processadors AMD, que és el que s’utilitza per elaborar aquest TFG, no permet la virtualització niuada. Així que es va optar per la solució més senzilla que era usar Windows Server 2019, amb l’instal·lador d’IRIS for Health.

### CREACIÓ DEL HIS

Per a fer el HIS, s’ha optat per fer una simulació. Això es deu a que el temps de desenvolupament per elaborar un HIS en condicions seria molt llarg, i això causaria que no donés temps a desenvolupar tant el servidor web com l’aplicació.

Així que directament s’han programat dues classes dintre d’IRIS.

La primera és com si fos l’objecte, que s’ha anomenat HIS. Aquest objecte està relacionat amb una taula en la base de dades, que s’ha afegit després de crear aquest objecte. Aquest objecte només té les variables de les tres alertes que est tenen pensades per aquest TFG. La primera és el percentatge d’ocupació dels llits d’un hospital, malalts per la COVID-19. La segona és el nombre de vacunes Pfizer per la COVID-19 que queden en el magatzem de la institució mèdica. La tercera i última alerta són el nombre d’ambulàncies que estan a l’hospital, és a dir, les ambulàncies que estan aparcades.

1. Class TFG.HIS Extends (%Persistent, %JSON.Adaptor) [ DdlAllowed ]
2. {
3. Property hisCovidBedsPercentage As %Float;
4. Property hisPfizerVaccineInStorage As %Integer;
5. Property hisAmbulancesWaiting As %Integer;
6. }

Com s’ha mencionat, després s’ha creat la taula a la base de dades. Aquesta taula té les mateixes variables que la classe i a part té un identificador. S’han generat cent mil dades diferents perquè, sempre doni un valor diferent.

### CREACIÓ DE L’API

Per a crear una API en IRIS for Health són necessari tres classes.

La primera és un JSON anomenat *spec* (Specification) i és l’encarregat de configurar les API. Això vol dir que en aquest JSON, hi haurà quina és la part de l'URI especifica per accedir a les seves dades i quins són els codis de resposta de cada API específica. A continuació hi ha un exemple per a obtenir el percentatge d’ocupació de llits amb malalts de COVID-19.

1. "/covidBed":{
2. "get":{
3. "operationId":"GetCovidBeds",
4. "produces":["application/json"],
5. "responses":{
6. "200":{"description":"Success"},
7. "404":{"description":"Not Found"},
8. "500":{"description":"Server error"}
9. }
10. }
11. }

Les següents classes s’autogeneren soles. Aquestes classes es diuen *disp* i *impl*. La classe *disp* (Dispatch), és l'encarregada de gestionar el servei REST. En el cas d’aquest TFG, s’ha deixat com està, ja que era la funció esperada. En el codi següent es pot veure com s’ha agafat el mètode GetCovidBeds, ja que els altres dos mètodes que queden són pràcticament idèntics, a excepció de la línia 6, que és la que està marcada amb verd, que canviaria la invocació del mètode.

1. ClassMethod GetCovidBeds() As %Status
2. {
3. Try {
4. Do ##class(%REST.Impl).%SetContentType("application/json")
5. If '##class(%REST.Impl).%CheckAccepts("application/json")' Do ##class(%REST.Impl).%ReportRESTError( ..#HTTP406NOTACCEPTABLE, $$$ERROR( $$$RESTBadAccepts)) Quit
6. Set response=##class(TFG.impl).GetCovidBeds()
7. Do ##class(TFG.impl).%WriteResponse(response)
8. } Catch (ex) {
9. Do ##class(%REST.Impl).%ReportRESTError( ..#HTTP500INTERNALSERVERERROR, ex.AsStatus(), $parameter("TFG.impl", "ExposeServerExceptions"))
10. }
11. Quit $$$OK
12. }

La següent i ultima classe programada per a crear l’API, és la *impl* (Implementation). Aquesta classe té la funció d'agafar les dades d’una base de dades o d’algun altre sistema que implementi el HIS. El que s’ha fet és molt simple, ja que com hi ha una sola taula, només s’ha d’accedir a una de les columnes de la taula cada vegada que es fa una crida a l’API. Així que s’ha generat un valor aleatori que va del 1 al 100.000, per a agafar una taula. Seguidament en cada mètode, GetCovidBeds, GetPfizerVaccineInStorage i GetAmbulancesWaiting, s’ha agafat la variable corresponent i s’envia en format JSON.

A continuació es mostrarà la implementació del mètode GetCovidBeds. D’igual manera que la classe *disp*, en els altres mètodes només varien en la línia 9, que és la que està marcada en color verd, i varia només l’atribut de retorn de la classe HIS.

1. ClassMethod GetCovidBeds() As %DynamicObject
2. {
3. Try {
4. Set id=$RANDOM(100000)+1
5. Set cm = ##class(TFG.HIS).%OpenId(id)
6. Do ..%SetStatusCode("200")
7. return {"value": (cm.hisCovidBedsPercentage)}
8. } Catch(ex) {
9. Do ..%SetStatusCode("500")
10. return {"errormessage": "Server error"}
11. }
12. }

Una vegada ja s’ha programat tot, només falta crear a IRIS, l’aplicació web, que seria l’API. En la següent imatge es pot veure com se li assigna un nom, que serà una part de l’URI, és a dir, l’API que s’ha fet tindrà la següent forma:

* + https://www.domini.com/rest/tfg/covidBed: Pel cas dels llits del percentatge d’ocupació de llits amb malalts de COVID-19.
  + https://www.domini.com/rest/tfg/pfizerStorage: Pel cas del nombre de vacunes Pfizer que queden en el magatzem.
  + https://www.domini.com/rest/tfg/ambulancesWaiting: Pel cas del nombre d’ambulàncies, de la institució mèdica, aparcades.

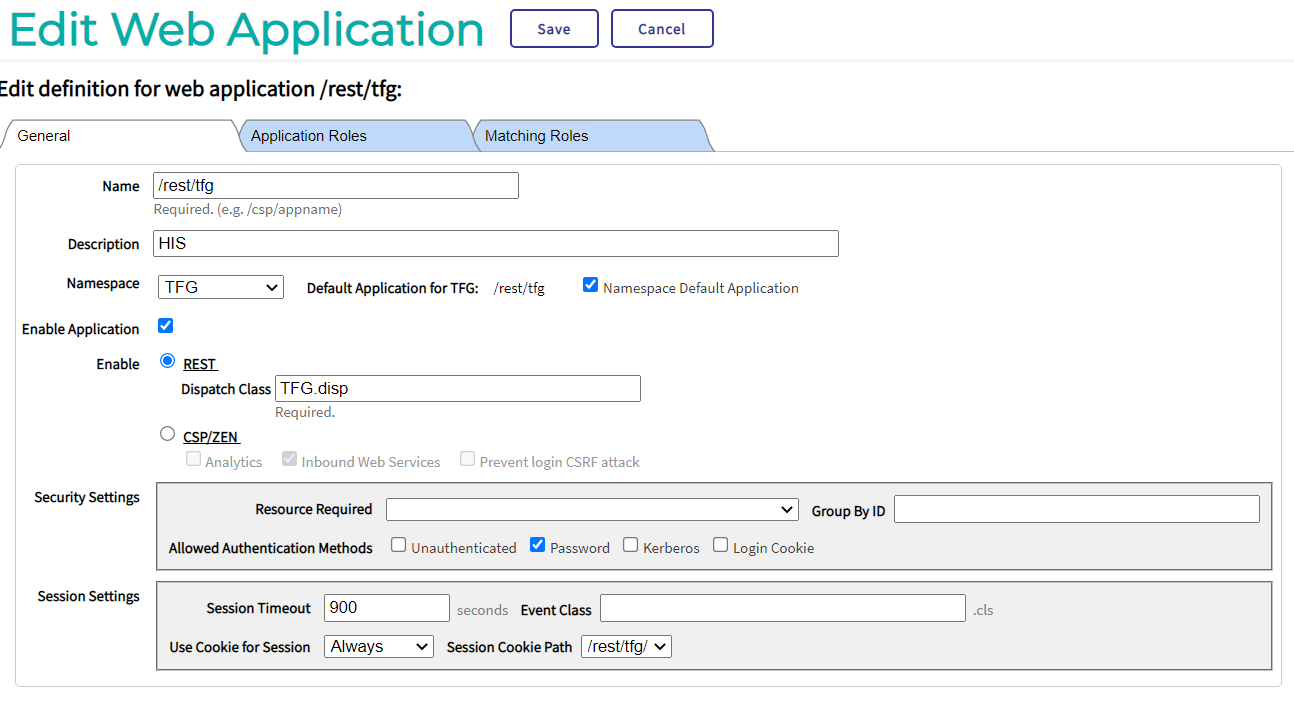


Figura .: Imatge de la configuració d’una API en Intersystems IRIS for Health.

Després del nom se li ha d’assignar quina classe d’enviament (*disp*) ha d’utilitzar. Per acabar només queda la seguretat i en el cas d’aquesta API, es necessitarà un usuari i contrasenya.

### DEPLOY

### PROBLEMES EN EL DESENVOLUPAMENT DEL SERVIDOR D’IRIS

## SERVIDOR WEB

El servidor web s’ha fet seguint el funcionament del *framework Spring*. Així que, com s’ha pogut veure al diagrama UML, s’han separat les classes que gestionen les crides a la pàgina web, de la base dades.

En els següents subapartats, el backend, la REST-API i el frontend, s’explicarà com funciona la creació d’una alerta, i com es mostra l’alerta, ja que al ser un codi simètric, les institucions mediques, els rols i els empleats, són bastant semblants.

### BACKEND

Per a fer el backend del servidor web, és necessiten crear tres paquets.

#### DOMINI

Primer paquet que s’ha de crear és el domini, on es guardaran les dades extretes de la base de dades en una classe Java, en aquest cas la classe *Warning*.

Aquestes classes son bastant simples, només tenen el seus atributs, dos constructors, un de buit i l’altre que inicialitza totes les dades i per acabar els seus gets i sets.

Un exemple d’una classe del domini seria la següent. Com s’ha mencionat a l’inici de l’apartat 6.3, es mostra la classe *Warning*, ja que tota l’explicació del servidor web es farà sobre com es crea i es mostren alertes.

1. public class Warning {
2. private int id;
3. private String name;
4. private String shortName;
5. private String description;
6. private String uri;
7. private String notificationMessage;
9. private float greenValue;
10. private float yellowValue;
11. private float redValue;
12. private float lastValue;
13. private float refreshRate;
15. private Role role;
17. public Warning(int id, String name, String shortName, String description, String uri, String notificationMessage, float greenValue, float yellowValue, float redValue, float lastValue, float refreshRate, Role role)
18. {
19. this.id = id;
20. this.name = name;
21. this.shortName = shortName;
22. this.description = description;
23. this.uri = uri;
24. this.notificationMessage = notificationMessage;
25. this.greenValue = greenValue;
26. this.yellowValue = yellowValue;
27. this.redValue = redValue;
28. this.lastValue = lastValue;
29. this.refreshRate = refreshRate;
30. this.role = role;
31. }
33. public Warning() {
34. }
36. // GETS I SETS
37. }

#### PERSISTÈNCIA

El segon paquet és el de la persistència i serà la que parli amb la base de dades. Aquest paquet tindrà dues classes. La primera, la que es comuniqui a través de queries amb la base de dades, DatabaseDAO. I l’altre la que fa de façana de la classe anterior i es comunica amb els controladors webs, DatabaseController.

Aquesta primera classe necessitarà, a part d’uns strings amb els queries, d’un mètode que converteixi una fila de la base de dades en un objecte del tipus *Warning*. Amb aquest mètode, agafar les dades de les files de la taula Warnings serà bastant més senzill, ja que no s’haurà de repetir codi.

A més el DatabaseDAO necessita un objecte de la classe JdbcTemplate, ja que serà qui farà les crides a la base de dades. Per saber quina base de dades ha de comunicar-se, s’ha hagut de crear un arxiu de propietats on hi ha totes les credencials d’aquesta base de dades.

Per acabar amb aquesta classe, hi ha tres mètodes desprès del warningMapper. El primer mètode és el que afegirà una fila a la taula Warnings. La segona, agafarà totes les files de la taula Warnings. I l’últim mètode agafarà la fila de la taula Warnings que tingui l’identificador que és passa per paràmetre del mètode.

1. @Repository
2. public class DatabaseDAO {
3. private final String INSERT\_WARNING = "INSERT INTO Warnings VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?);";
4. private final String FIND\_ALL\_WARNINGS = "SELECT \* FROM Warnings";
5. private final String FIND\_WARNING\_BY\_ID = "SELECT \* FROM Warnings WHERE war\_id=?";
6. private final String COUNT\_WARNINGS = "SELECT COUNT(war\_id) FROM Warnings";
8. private final JdbcTemplate jdbcTemplate;
10. public DatabaseDAO(JdbcTemplate jdbcTemplate) {
11. this.jdbcTemplate = jdbcTemplate;
12. }
14. //MAPPERS
15. private RowMapper<Warning> warningMapper = (resultSet, i)-> {
16. Warning warning = new Warning(
17. resultSet.getInt("war\_id"),
18. resultSet.getString("war\_name"),
19. resultSet.getString("war\_shortname"),
20. resultSet.getString("war\_description"),
21. resultSet.getString("war\_uri"),
22. resultSet.getString("war\_notificationmessage"),
23. resultSet.getFloat("war\_greenvalue"),
24. resultSet.getFloat("war\_yellowvalue"),
25. resultSet.getFloat("war\_redvalue"),
26. resultSet.getFloat("war\_lastvalue"),
27. resultSet.getInt("war\_refreshrate"),
28. new Role(resultSet.getString("war\_rolename"), resultSet.getInt("war\_roleinstitutionid"), resultSet.getString("war\_rolecountryid") )
29. );
30. return warning;
31. };
33. //CREATE
34. public int insertWarning(Warning warning) {
35. int id = jdbcTemplate.queryForObject(COUNT\_WARNINGS, new Object[]{}, Integer.class );
36. return jdbcTemplate.update(INSERT\_WARNING, id, warning.getName(), warning.getShortName(), warning.getDescription(), warning.getUri(), warning.getNotificationMessage(), warning.getGreenValue(), warning.getYellowValue(), warning.getRedValue(), 0, warning.getRefreshRate(), warning.getRole () );
37. }
39. //FIND ALL
40. public List<Warning> findAllWarnings() {
41. return jdbcTemplate.query(FIND\_ALL\_WARNINGS, new Object[]{}, warningMapper );
42. }
44. //FIND ONE
45. public Warning findWarning(int id) {
46. return jdbcTemplate.queryForObject(FIND\_WARNING\_BY\_ID, new Object[]{id}, warningMapper );
47. }
48. }

La classe DatabaseController és bastant més senzilla que la classe anterior, ja que crida directament als mètodes de la classe anterior. A continuació es mostra com s’ha fet la classe DatabaseController.

1. @Service("DatabaseController")
2. public class DatabaseController {
4. private final DatabaseDAO databaseDAO;
6. public DatabaseController(DatabaseDAO databaseDAO) {
7. this.databaseDAO = databaseDAO;
8. }
10. //CREATE
11. public void addWarning(Warning warning) {
12. databaseDAO.insertWarning(warning);
13. }
15. //GET ALL
16. public List<Warning> getAllWarnings() {
17. return databaseDAO.findAllWarnings();
18. }
20. // GET ONE
21. public Warning getWarning(int id) {
22. return databaseDAO.findWarning(id);
23. }
24. }

#### CONTROLADOR WEB

Per acabar es necessita el controlador web, aquest serà l’encarregat de gestionar les crides a la pàgina web. Està separat en dues classes, la classe WebGetController, que donada un URL, retornarà la pàgina HTML. I la classe WebPostController, que donada un URL o si es fa una crida POST, desarà les dades corresponents a la base de dades o farà la funció que toqui.

Per començar, la classe WebGetController, en el cas de les alertes, tindrà tres mètodes.

El primer el que retornarà la pàgina HTML per la creació d’una alerta, per crear una alerta s’ha d’afegir la llista de tots els rols.

El segon mètode serà el que mostri la llista d’alertes. Així que per poder-ho ensenyar, a part de la pàgina HTML, fa falta la llista d’alertes, fent una crida al mètode getAllWarnings de la DatabaseController.

El tercer i últim mètode retornarà la pàgina HTML per ensenyar les dades d’una alerta en específic. Així que s’ha de passar aquella alerta fent la crida al mètode pertinent de la DatabaseController.

1. @Controller
2. public class WebGetController {
3. private DatabaseController databaseController;
5. public WebGetController(DatabaseController databaseController) {
6. this.databaseController = databaseController;
7. }
9. //CREAR
10. @GetMapping("/warning/create")
11. public String createWarning(Model model) {
12. model.addAttribute("createWarning", new Warning());
13. model.addAttribute("roles", databaseController.getAllRoles());
14. return "warning/createWarning";
15. }
17. //GET ALL
18. @GetMapping("/warning/all")
19. public String getAllWarnings(Model model) {
20. model.addAttribute("warnings", databaseController.getAllWarnings());
21. return "warning/showWarnings";
22. }
24. //GET ONE
25. @GetMapping("/warning/{id}")
26. public String getWarning(@PathVariable int id, Model model) {
27. model.addAttribute("warning", databaseController.getWarning(id));
28. return "warning/showWarning";
29. }
30. }

A continuació es mostrarà el WebPostController. A diferència de la classe anterior només hi haurà un mètode, ja que l’única pàgina HTML que ha de fer una crida POST és la de crear una alerta. Així que, gràcies a Spring i Thymeleaf, que aquest últim es veurà en l’apartat del frontend, ja crea l’alerta automàticament amb les dades que dóna l’usuari. Per tant, el mètode s’ha d’encarregar d’afegir-ho a la base de dades i reencaminar a l’usuari perquè vegi les dades de l’usuari creat.

1. @Controller
2. public class WebPostController {
3. private DatabaseController databaseController;
5. public WebPostController(DatabaseController databaseController) {
6. this.databaseController = databaseController;
7. }
9. @PostMapping("/warning/create")
10. public String createWarning(@Valid Warning warning, Model model, RedirectAttributes redirectAttributes)
11. {
12. model.addAttribute("id", warning.getId());
13. databaseController.addWarning(warning);
14. return "redirect:/warning/{id}";
15. }
16. }

### REST-API

### GESTIÓ DE LES NOTIFICACIONS

### FRONTEND

### PROBLEMES EN EL DESENVOLUPAMENT DEL SERVIDOR WEB

## APLICACIÓ

### INICIAR SESSIÓ

### LLISTA D’ALERTES

### ALERTA

### PERFIL

### REBRE NOTIFICACIONS

### PROBLEMES EN EL DESENVOLUPAMENT DE L’APLICACIÓ

# RESULTATS

## RESULTATS DELS OBJETIUS

## PROPOSTES DE MILLORA

## VALORACIÓ FINAL

# BIBLIOGRAFIA

[1] InterSystems, *IRIS for Health*. https://www.intersystems.com/products/intersystems-iris-for-health/, 2018.

[2] H. Dalianis, «Chapter 2 - The History of the Patient Record and the Paper Record», en *Clinical Text Mining*, Cham: Springer International Publishing, 2018, p. 5-13.

[3] R. F. Gillum, «From Papyrus to the Electronic Tablet: A Brief History of the Clinical Medical Record with Lessons for the Digital Age», *AJM*, p. 5, oct. 2013, doi: 10.1016/j.amjmed.2013.03.024.

[4] A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, i F. Jahn, «2.4 Importance of Systematic Information Management», en *Health Information Systems - Architectures and Strategies*, Second Edition., Springer, 2011, p. 12-16.

[5] Reinhold Haux, «3.1. The 1st line: towards computer-based information processing tools», en *Health information systems — past, present, future*, 2006, p. 271.

[6] H. Dalianis, «Chapter 3 - User Needs: Clinicians, Clinical Researchers and Hospital Management», en *Clinical Text Mining*, Cham: Springer International Publishing, 2018, p. 13-21.

[7] G. O. Barnett *et al.*, «COSTAR—A computer-based medical information system for ambulatory care», *Proc. IEEE*, vol. 67, núm. 9, p. 1226-1237, 1979, doi: 10.1109/PROC.1979.11438.

[8] «eHealth Resolution», World Health Organization, Geneva, 58th World Health Assembly. Resolution 28, maig 2005. Consulta: gen. 01, 2021. [En línia]. Disponible a: https://www.who.int/healthacademy/media/WHA58-28-en.pdf.

[9] Wolters Kluwer Health, *UpToDate*. .

[10] EBSCO Information Services, *Dynamed*. .

[11] A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, i F. Jahn, *Health Information Systems: Architectures and Strategies*. London: Springer London, 2011.

[12] A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, i F. Jahn, «5.5 A Reference Model for the Domain Layer of Hospital Information Systems», en *Health Information Systems - Architectures and Strategies*, Second Edition., Springer, 2011, p. 70-71.

[13] HIMSS, «Strategic Interoperability in Germany, Spain & the UK». 2014, [En línia]. Disponible a: https://www.himss.eu/file/191/download?token=goNVaM8t.

[14] A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, i F. Jahn, «6.5.4.1 Health Level 7 (HL7) Version 2», en *Health Information Systems: Architectures and Strategies*, London: Springer London, 2011, p. 149-151.

[15] A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, i F. Jahn, «6.5.4.2 Health Level 7 (HL7) Version 3», en *Health Information Systems: Architectures and Strategies*, London: Springer London, 2011, p. 151-152.

[16] «HL7 in Personal Health System component’s integration for Mental Health Treatment».

[17] S. Subramanian, «DICOM Basics using Java - Extracting Image Data», oct. 06, 2014. https://saravanansubramanian.com/extractdicomimagedata/.

[18] Microsoft, *Hyper-V*. https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/, 2008.

[19] Google, *Android Studio*. https://developer.android.com/studio, 2013.

[20] Facebook, *React Native*. https://reactnative.dev/, 2015.