Enginyeria del Programari

TechCare – Empresa de Software

Document SRS: SeniorLife

Versió 1.0

22/11/24

Historial de revisions

Data	Versió	Descripció	Autor
22/11/24	1.0	Anàlisi de requisits i disseny sistema	TechCare
05/12/24	2.0	Disseny i implementació	TechCare

Taula 1. Historial de versions.

Sobre aquest document

Aquest document recull els requisits i dissenys tècnics per al desenvolupament del sistema SeniorLife, una aplicació destinada a persones grans per a la gestió de la seva salut i benestar. El contingut d'aquest document és el resultat d'una col·laboració entre diversos stakeholders, incloent-hi l'emprenedora Carla Qurban, l'enginyera en informàtica Alejandra Popa, el metge Pascual Peña, el CFO Eduardo Gasch i Antonio Garcia, amic i familiar d'un usuari potencial.

El document està dividit en quatre seccions principals:

1. **Anàlisi de Requisits:** Aquesta secció inclou els requisits funcionals i no funcionals que defineixen el comportament esperat del sistema, així com els possibles conflictes i dependències entre requisits. També s'hi inclouen els casos d'ús a nivell d'usuari que descriuen les interaccions dels usuaris amb el sistema.

2. **Disseny del Sistema:** Aquí es detallen els diagrames tècnics que descriuen l'arquitectura del sistema, com els diagrames de casos d'ús (DCU), diagrames d'activitat, seqüència i classes. També s'inclouen els wireframes i el disseny visual, per proporcionar una visió clara de la interfície d'usuari.

3. **Implementació:** Aquesta secció aborda els patrons de disseny seleccionats per a la implementació del sistema i la fase inicial de desenvolupament, incloent les proves i la validació del sistema.

4. **Aprovació del document:** Finalment, el document recull les aprovacions formals i qualsevol comentari addicional rellevant per al seguiment del projecte.

Aquest document servirà com a guia per al desenvolupament del sistema SeniorLife i com a referència per a les futures etapes de disseny i implementació. A mesura que es vagi avançant en el projecte, els detalls d'aquest document es podran actualitzar per reflectir canvis i noves decisions preses per l'equip.

Taula de continguts

1. Anàlisi de requeriments

Aquesta secció identifica i defineix els requisits, tant funcionals com no funcionals, dels sistema. Aquests es troben resumits en la Taula 1. A més, exposa possibles conflictes i dependències entre ells.

Requisits funcionals (F)	Requisits no funcionals (NF)
F001 – L'aplicació ha de permetre la comunicació entre la gent gran, família, amics i els voluntaris.	NF001 - Les accions principals s'ha de realitzar en menys de 10 segons.
F002 – L'aplicació ha de monitoritzar constants vitals o altres paràmetres relacionats.	NF002 - L'aplicació ha de complir el Reglament General de Protecció de Dades (GDPR).
F003 – Si algun dels paràmetres no és l'esperat l'aplicació llançarà una alarma.	NF003 - L'aplicació ha de ser dissenyada per a tablets Android i mòbils iOs i Android.
F004 – L'aplicació ha de permetre que l'usuari creï un perfil mèdic, introduir la seva medicació i la programació d'una videoconferència amb el personal mèdic per validar les dades.	

Taula 2. Requisits funcionals i no funcionals.

1.2. Requisits funcionals (F)

En aquest apartat es mostra una llista de quatre requisits funcionals recollits de l'usuari. Es mostra la descripció d'aquest, juntament amb el motiu pel qual l'stakeholder el desitja, en format d'història d'usuari.

- F001: Com a directora executiva, l'aplicació ha de permetre la comunicació entre la gent gran, la seva família, el seu cercle d'amistats i amb els grups de voluntaris de SeniorLife. La comunicació es farà en format de missatge de text, fotografia i videoconferència.
- F002: Com a directora executiva, l'aplicació ha de monitoritzar constants vitals o altres paràmetres relacionats amb l'activitat o salut de l'usuari mitjançant l'ús de 'wearables' o altres dispositius IoT.
- F003: Com a directora executiva, si algun dels paràmetres no és l'esperat l'aplicació llançarà una alarma que rebrà el personal mèdic, els familiars propers i, si aquesta es considera crítica, s'enviarà directament als serveis d'emergència.

- F004: Com a directora executiva, l'aplicació ha de permetre que l'usuari creï un perfil mèdic mitjançant un formulari inicial, la introducció de dades sobre la seva medicació actual i la programació d'una videoconferència amb el personal mèdic per validar les dades introduïdes.

1.3. Requisits no funcionals (NF)

En aquest apartat es mostren tres requisits no funcionals de l'usuari, recollits en forma de taula. A més, es mostra la descripció d'aquests, juntament amb l'stakeholder que els desitja.

Requisit	NF001: Les accions principals s'ha de realitzar en menys de 10 segons.
Descripció	La plataforma ha de permetre que les accions principals com l'enviament de missatges o inicis de videotrucades es realitzin en menys de 10 segons, ja que es vol ser èmfasi en la usabilitat.
Tipus	NF de rendiment: dinàmic.
Stakeholders	Carla Qurban, emprenedora i directora executiva.

Taula 3. NF001.

Requisit	NF002: L'aplicació ha de complir el Reglament General de Protecció de Dades (GDPR).
Descripció	L'aplicació ha de complir el GDPR a alt nivell, assegurant la confidencialitat de les dades personals i mèdiques, prohibint la seva cessió sense consentiment.
Tipus	NF de restriccions de disseny: acompliment d'estàndards.
Stakeholders	Alejandra Popa, enginyera en informàtica i CTO (Chief Technology Officer, responsable tècnica).

Taula 4. NF002.

Requisit	NF003: L'aplicació ha de ser dissenyada per a tablets Android i mòbils iOs i Android.
Descripció	L'aplicació ha d'estar dissenyada per a tablets Android, per a la gent gran, per a mòbils, compatible tant amb smartphones iOS com Android per a la resta.
Tipus	NF de restriccions de disseny: limitacions de hardware.
Skateholders	Alejandra Popa, enginyera en informàtica i CTO (Chief Technology Officer, responsable tècnica).

Taula 5. NF003.

1.4. Conflictes del sistema

A continuació, es mostren els requisits que estan en conflicte i els stakeholders interessats en cada un.

Cas 1: Privacitat vs Monitoratge constant:
D'una banda, és essencial que les persones grans tinguin la capacitat de gestionar qui les monitoritza i quan, juntament amb l'opció de desactivar funcions específiques si es percep com una violació de la seva privadesa. Això aborda les preocupacions sobre la privadesa i fomenta una sensació d'autonomia entre els usuaris i els seus familiars. Per contra, el sistema ha de fer un seguiment constant dels indicadors mèdics per identificar irregularitats i activar alertes quan sorgeixen amenaces per a la salut. Aquest seguiment és vital per al personal mèdic i els cuidadors, que depenen d'aquesta capacitat per salvaguardar el benestar dels qui assisteixen. El conflicte sorgeix en intentar conciliar aquests requisits, ja que la vigilància continuada pot ser vista com una intrusió i provocar resistència per part de l'usuari, mentre que la disminució de la supervisió podria posar en perill la seguretat i dificultar la prevenció de situacions crítiques.

Cas 2: Usabilitat senzilla vs Funcionalitats avançades:
L'aplicació està dissenyada principalment per a persones grans amb poca experiència tecnològica, i és fonamental que sigui intuïtiva i permeti realitzar les accions principals en menys de deu segons. Aquesta simplicitat és clau per als usuaris finals, que necessiten una experiència d'ús sense complicacions. Tanmateix, l'aplicació també ha d'oferir funcionalitats complexes com videoconferències, recerca de grups d'interès, monitoratge de dades mèdiques i personalització de paràmetres, que són imprescindibles per a altres usuaris com familiars i personal mèdic. Aquestes funcionalitats, tot i ser avançades, poden complicar el disseny de la interfície i dificultar l'accés a les tasques bàsiques. Per tant, hi ha un conflicte intrínsec entre mantenir la simplicitat per a un col·lectiu amb necessitats específiques i proporcionar una aplicació potent que acompleixi els requisits d'altres stakeholders.

1.5. Dependència de requisits

Si analitzem els casos d'ús de la reunió, observem que hi ha requisits dependents entre si. Per exemple, la creació i ús del perfil mèdic de l'usuari per al monitoratge remot de la seva salut.

Tal i com explica en Pascual Peña, l'usuari ha de crear-se un perfil mèdic omplint un formulari sobre la seva condició mèdica, juntament amb medicació, malalties i historial mèdic. Doncs, el primer requisit necessari per a la monitorització és la **creació d'un perfil**

mèdic ja que sense aquest el personal mèdic no podria configurar un monitoratge adequat i personalitzat en base a la salut del pacient.

Tal i com explica la Carla Qurban i en Pascual Peña, una vegada creat el perfil, el personal mèdic assignat valida i configura els paràmetres específics que seran monitoritzats amb els dispositius "wearables". Per tant, el segon requisit necessari és **l'activació de la secció de monitoratge mèdic i assignació de paràmetres**. Sense aquests paràmetres configurats, els dispositius "wearables" no tenen instruccions per mesurar ni detectar anomalies, i no poden monitoritzar correctament.

L'Antonio Garcia i la Carla Qurban detallen que un cop establerts els paràmetres mèdics, el sistema ha de tenir capacitat per enviar notificacions i alertes als familiars i al personal mèdic en cas que es detectin valors fora dels paràmetres establerts. Doncs, considerem que el tercer requisit necessari és la **notificació i comunicació de les dades** ja que sense una estructura que enviï alertes el monitoratge no seria útil.

Finalment, l'Antonio Garcia exposa que l'aplicació ha de permetre que l'usuari i els seus familiars tinguin accés al perfil mèdic i a l'historial de monitoratge, amb la capacitat de configurar la privadesa de qui i quan poden veure aquestes dades. Des de TechCare identifiquem **l'accés de l'usuari i els familiars al monitoratge mèdic** com el quart requisit necessari per a la revisió del monitoratge i establir preferències de privadesa, tot i que només es pot proporcionar quan el sistema està funcionant. Donem importància a aquest requisit ja que garanteix l'autonomia i privadesa de l'usuari, la qual és essencial per a la seva tranquil·litat i confiança del sistema de seguiment.

En conclusió, els requisits esmentats formen un flux coherent que és necessari i essencial per al funcionament complet del sistema de monitoratge remot de la salut de la gent gran. Aquests proporcionen les bases necessàries per configurar, activar i utilitzar el monitoratge de manera efectiva i segura per a cada usuari.

1.6. Casos d'ús a nivell d'usuari

En aquest apartat es presenten els casos d'ús identificats a partir dels requisits i la informació recollida durant les entrevistes amb els stakeholders. Els casos d'ús representen interaccions concretes entre els actors principals (persona gran, familiars, metges, voluntaris, directors) i l'aplicació. Cada cas d'ús descriu una funcionalitat específica que permet assolir un objectiu determinat dins el sistema.

Els casos d'ús estan dissenyats per cobrir un ampli ventall de necessitats dels usuaris, des de la interacció social fins al seguiment mèdic, i garantir que totes les funcionalitats

siguin accessibles. A continuació, es presenta la taula amb alguns dels casos d'ús identificats:

CASOS D'ÚS	
Descripció	**Actor inicial**
Iniciar una videoconferència amb un familiar	Persona gran
Completar el formulari inicial per al registre mèdic	Persona gran o familiar
Validar la informació mèdica proporcionada per l'usuari	Metge
Sol·licitar ajuda d'un voluntari	Persona gran
Programar una cita amb el metge	Persona gran
Configurar els paràmetres de monitoratge de salut amb dispositius wearables	Metge
Consultar el resum d'activitat diària de la persona gran	Familiar
Configurar alertes i notificacions per a paràmetres crítics	Metge
Desactivar temporalment el monitoratge de dispositius	Persona gran
Consultar estadístiques sobre el temps d'ús de l'aplicació per cada tipus d'usuari, les funcionalitats més usades i la interacció amb els anuncis.	CFO
Canviar alguna dada de la posologia	Metge

Taula 6. Casos d'ús a nivell d'usuari.

2. Disseny del sistema

Aquesta secció detalla la documentació visual i tècnica dels requisits mitjançant diagrames UML, especificacions de casos d'us i dissenys de l'aplicació.

2.1. Diagrama de casos d'ús i descripcions DCUS

A continuació es descriuen 5 DCUS. També s'exposa el diagrama de casos d'ús.

CAS D'ÚS 1:	Completar el registre mèdic		
Versió	1.0	**Data**	21/11/2024
Autors	TechCare		
Descripció	Aquest cas d'ús descriu com l'usuari pot completar un registre mèdic, omplint un formulari inicial. Inclou la validació posterior per part del metge i l'activació de les alertes mèdiques.		
Actors	Persona gran o familiar, Metge.		
Precondició	L'usuari ha d'estar registrat al sistema amb accés actiu per poder iniciar sessió en l'aplicació.		
Flux Principal	1. La persona gran o el familiar inicia sessió i selecciona l'opció de "registre mèdic" 2. Omple els camps requerits (historial mèdic, medicacions, dades personals). 3. El sistema valida el formulari per verificar que totes les dades són completes. 4. La persona gran selecciona una data disponible per a una videoconferència amb el metge. 5. Durant la videoconferència, el metge revisa i valida la informació proporcionada. 6. El metge configura alertes mèdiques i paràmetres de monitoratge 7. El metge aprova el registre mèdic de la persona gran.		
Subfluxos	El sistema pot oferir ajuda automàtica per guiar l'usuari durant el registre. Es proposa de manera automàtica quan la validació és incorrecta.		
Fluxos alternatius	- Si el formulari està incomplet, el sistema notifica l'usuari perquè el revisi i el completi. - Si el metge detecta errors o mancances en el resum, pot rebutjar el registre i notifica a l'usuari perquè aquest demani una cita presencial.		

Postcondicions	El registre mèdic queda complet i s'activen les alertes configurades pel metge.
Requeriments no funcionals	La interfície ha de ser intuïtiva i accessible per persones amb discapacitats visuals, ha de complir com a mínim amb les indicacions del W3C.
Prioritat	Urgent: és essencial per habilitar les funcionalitats mèdiques del sistema.
Comentaris	Ha de garantir la privacitat de les dades.

Taula 7. Cas d'us 1: CU1.

CAS D'ÚS 2:	Sol·licitar ajuda d'un voluntari		
Versió	1.0	**Data**	21/11/2024
Autors	TechCare		
Descripció	Aquest cas d'ús descriu com una persona gran pot sol·licitar ajuda a un voluntari registrat al sistema.		
Actors	Persona gran, Voluntari.		
Precondició	L'usuari ha iniciat sessió i existeixen voluntaris disponibles.		
Flux Principal	1. La persona gran inicia sessió i accedeix al menú de comunicació social. 2. Selecciona l'opció "Voluntari". 3. El sistema mostra la llista de voluntaris disponibles. 4. L'usuari selecciona un voluntari i envia la sol·licitud. 5. El voluntari rep la notificació i accepta o rebutja la sol·licitud.		
Subfluxos	El sistema suggereix voluntaris segons la ubicació i la disponibilitat. S'encarrega l'algorisme de cerca.		
Fluxos alternatius	- Si no hi ha voluntaris disponibles, el sistema informa l'usuari i permet deixar una sol·licitud en espera. Igualment, el sistema es manté actiu cercant altres voluntaris i oferint altres possibilitats (per si l'ajuda és urgent).		
Postcondicions	La sol·licitud d'ajuda queda registrada, i el voluntari seleccionat pot intervenir. A més, el voluntari esdevé "no disponible".		
Requeriments no funcionals	La interfície ha de ser intuïtiva i accessible per persones amb discapacitats visuals, ha de complir com a mínim amb les indicacions del W3C.		
Prioritat	Normal: és una funció social important però no crítica.		
Comentaris	Els voluntaris han de ser verificats prèviament per l'organització, però és aliè al cas d'ús ja que això s'ha de tenir en compte en el registre dels usuaris voluntaris. D'altra banda, la disponibilitat dels usuaris no depèn de si una persona demana ajuda o no. Per		

	tant, el cas d'ús no es responsabilitza d'aquesta part (i se suposa precondició).

Taula 8. Cas d'us 2: CU2.

CAS D'ÚS 3:	Programar una cita amb el metge		
Versió	1.0	**Data**	21/11/2024
Autors	TechCare		
Descripció	Aquest cas d'ús descriu com una persona gran pot programar una cita amb el metge assignat.		
Actors	Persona gran.		
Precondició	El metge ha de tenir disponibilitat d'horaris al sistema.		
Flux Principal	1. La persona gran inicia sessió i selecciona "Monitorització de la salut". 2. A continuació selecciona "contactes i consultes". 3. El sistema mostra els metges disponibles assignats. 4. L'usuari selecciona el botó "sol·licitar cita" i selecciona un horari i confirma la cita. 5. El sistema verifica la disponibilitat i reserva la cita. 6. El sistema envia una notificació al metge i a l'usuari.		
Subfluxos	El sistema pot enviar recordatoris abans de la cita.		
Fluxos alternatius	- Si no hi ha disponibilitat, l'usuari pot sol·licitar una notificació quan hi hagi noves franges.		
Postcondicions	La cita queda registrada i visible tant per al metge com per a l'usuari.		
Requeriments no funcionals	La interfície ha de ser intuïtiva i accessible per persones amb discapacitats visuals, ha de complir com a mínim amb les indicacions del W3C.		
Prioritat	Normal: pot dependre de la urgència de la cita.		
Comentaris	S'hauria de permetre reprogramar o cancel·lar cites de forma senzilla.		

Taula 9. Cas d'us 3: CU3.

CAS D'ÚS 4:	Configurar alertes i notificacions per a paràmetres crítics		
Versió	1.0	**Data**	21/11/2024
Autors	TechCare		
Descripció	Aquest cas d'ús descriu com el metge configura les alertes i notificacions per a paràmetres crítics monitoritzats pels dispositius wearables.		
Actors	Metge.		

Precondició	Els dispositius wearables estan vinculats i monitoritzant paràmetres actius.
Flux Principal	1. El metge accedeix al perfil de l'usuari i selecciona la icona "Emergència". 2. El sistema mostra els paràmetres disponibles i els llindars actuals. 3. El metge defineix nous llindars per a paràmetres crítics. 4. El sistema activa les alertes i notifica al metge i familiars autoritzats.
Subfluxos	El sistema suggereix llindars recomanats segons les condicions mèdiques de l'usuari.
Fluxos alternatius	- Si el metge configura llindars incorrectes, el sistema mostra un avís de validació.
Postcondicions	Les alertes estan actives i funcionals segons la configuració.
Requeriments no funcionals	La interfície ha de ser intuïtiva i accessible.
Prioritat	Alta: impacta la seguretat de l'usuari.
Comentaris	Cal garantir que les alertes es comuniquin amb rapidesa i fiabilitat.

Taula 10. Cas d'us 4: CU4.

CAS D'ÚS 5:	Consultar estadístiques sobre el temps d'ús de l'aplicació per cada tipus d'usuari, les funcionalitats més usades i la interacció amb els anuncis.		
Versió	1.0	**Data**	21/11/2024
Autors	TechCare		
Descripció	Aquest cas d'ús descriu com el CFO consulta estadístiques sobre l'ús de l'aplicació per prendre decisions estratègiques.		
Actors	CFO		
Precondició	Els dispositius wearables estan vinculats i monitoritzant paràmetres actius.		
Flux Principal	1. El CFO inicia sessió i accedeix al panell de control d'estadístiques. 2. Selecciona els informes desitjats (temps d'ús, funcionalitats més utilitzades, interacció amb anuncis). 3. El sistema genera i mostra els informes seleccionats. 4. El CFO pot descarregar els informes o visualitzar gràfics detallats.		
Subfluxos			
Fluxos alternatius	- Si les dades no estan disponibles per a un període concret, el sistema mostra una notificació d'error.		

Postcondicions	El CFO obté la informació necessària per prendre decisions estratègiques.
Requeriments no funcionals	S'ha de complir el Reglament General de Protecció de Dades.
Prioritat	Normal
Comentaris	Les dades han d'estar ben protegides per salvaguardar el dret a la privacitat dels usuaris.

Taula 10. Cas d'us 5: CU5.

El diagrama de casos d'ús presentat s'alinea amb els casos d'ús descrits. A continuació, s'explica en paràgrafs com es relacionen els elements del diagrama amb els casos d'ús.

El Cas d'ús 1, **Completar el registre mèdic**, està representat dins del paquet "Registre Mèdic". Inclou els actors principals: la persona gran o el familiar que inicia el registre, i el metge, que valida i finalitza el procés. Aquest cas d'ús inclou dues dependències principals: "Realitzar Videoconferència amb el Metge" i "Validar Dades", com es descriu al flux principal. Durant el procés, la persona gran selecciona una data per a una videoconferència, on el metge revisa i valida les dades proporcionades. Un cop completat, s'activen les alertes configurades pel metge mitjançant el cas d'ús "Activar Alertes". Això assegura que el procés sigui complet i s'ajusta a les necessitats de monitoratge mèdic.

El Cas d'ús 2, **Sol·licitar ajuda d'un voluntari,** està representat dins del paquet "Ajuda Social". La persona gran inicia el procés i pot seleccionar un voluntari d'una llista disponible al sistema. En cas que no hi hagi voluntaris disponibles, el cas d'ús s'estén al cas "Notificar falta de voluntaris", que permet gestionar situacions en què no es pot oferir ajuda immediata. Això reflecteix els fluxos alternatius descrits al cas d'ús, assegurant que l'usuari rebi una resposta.

El Cas d'ús 3, **Programar una cita amb el metge,** està situat dins del paquet "Gestió de Cites". La persona gran inicia el procés seleccionant un metge disponible i un horari adequat. Aquest cas d'ús s'estén al cas "Gestionar disponibilitat" per cobrir situacions en què no hi ha horaris disponibles, oferint flexibilitat a l'usuari. Aquesta extensió garanteix que l'usuari pugui trobar una solució.

El Cas d'ús 4, **Configurar alertes i notificacions per a paràmetres crítics**, està dins del paquet "Alertes Crítiques". El metge és l'actor principal i pot configurar els llindars dels paràmetres monitoritzats pels dispositius. Aquest cas d'ús inclou validar les dades per assegurar que les alertes estan ben configurades. També proporciona la capacitat de personalitzar notificacions segons les necessitats de l'usuari i els seus familiars. Això es

relaciona amb el flux principal del cas d'ús, on el sistema activa les alertes configurades pel metge. Aquesta funcionalitat és crítica per garantir la seguretat de l'usuari.

El Cas d'ús 5, **Consultar estadístiques sobre l'ús de l'aplicació**, està representat dins del paquet "Estadístiques". El CFO inicia el procés accedint al panell de control per consultar dades detallades sobre l'ús de l'aplicació i la interacció amb els anuncis. Aquest cas d'ús s'estén al cas "Descarregar Informes", que permet al CFO descarregar informació específica en formats que poden utilitzar-se per a la planificació estratègica.

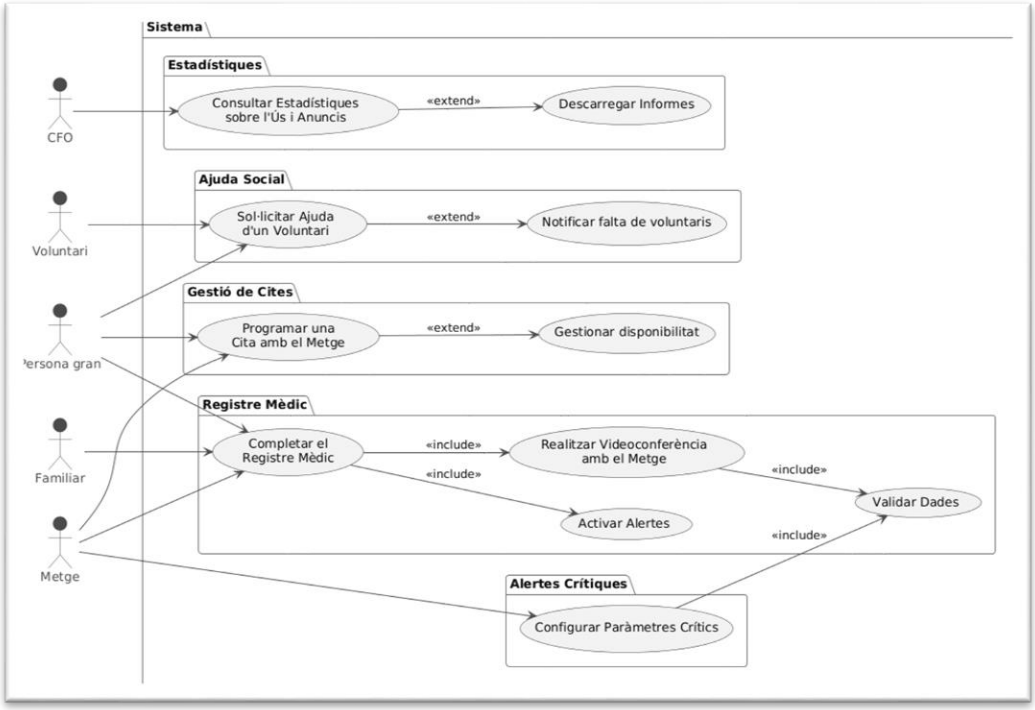

Diagrama Cas d'Ús d'usuari.

2.2. Diagrames d'activitats

En aquest diagrama d'activitat s'expandeix el cas d'ús **Completar el registre mèdic** i es descriuen amb detall totes les activitats realitzades pels diferents actors del cas d'ús.

Cal comentar que hem afegit la validació del formulari per part del sistema, aquest afegit no es considera pels clients però és de sentit comú que si no has omplert algun dels camps del formulari, el sistema no et deixi continuar amb la validació del personal mèdic. D'aquesta manera crees una espècie de "filtre" per evitar que formularis incomplets siguin enviats a aquest personal mèdic.

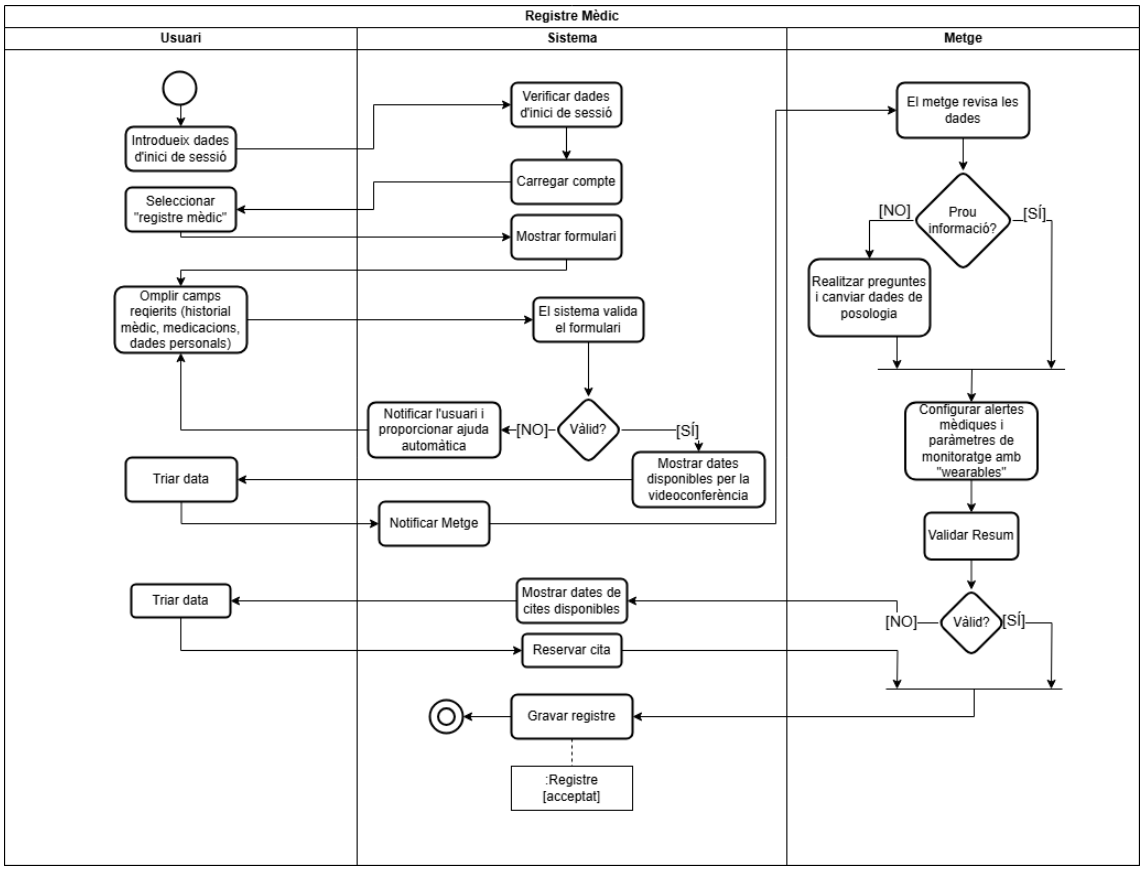

Diagrama d'Activitat 1. CU1: completar registre mèdic.

En aquest diagrama d'activitat s'expandeix el cas d'ús **Programar una cita amb el metge** i es descriuen amb detall totes les activitats realitzades pels diferents actors del cas d'ús.

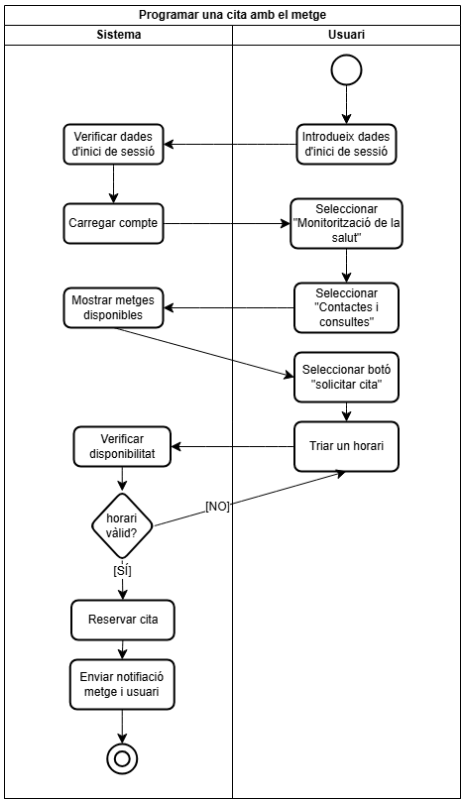

Diagrama d'Activitat 2. CU3: Programar una cita amb el metge.

En aquest diagrama d'activitat s'expandeix el cas d'ús **Configurar alertes i notificacions per a paràmetres crítics** i es descriuen amb detall totes les activitats realitzades pels diferents actors del cas d'ús.

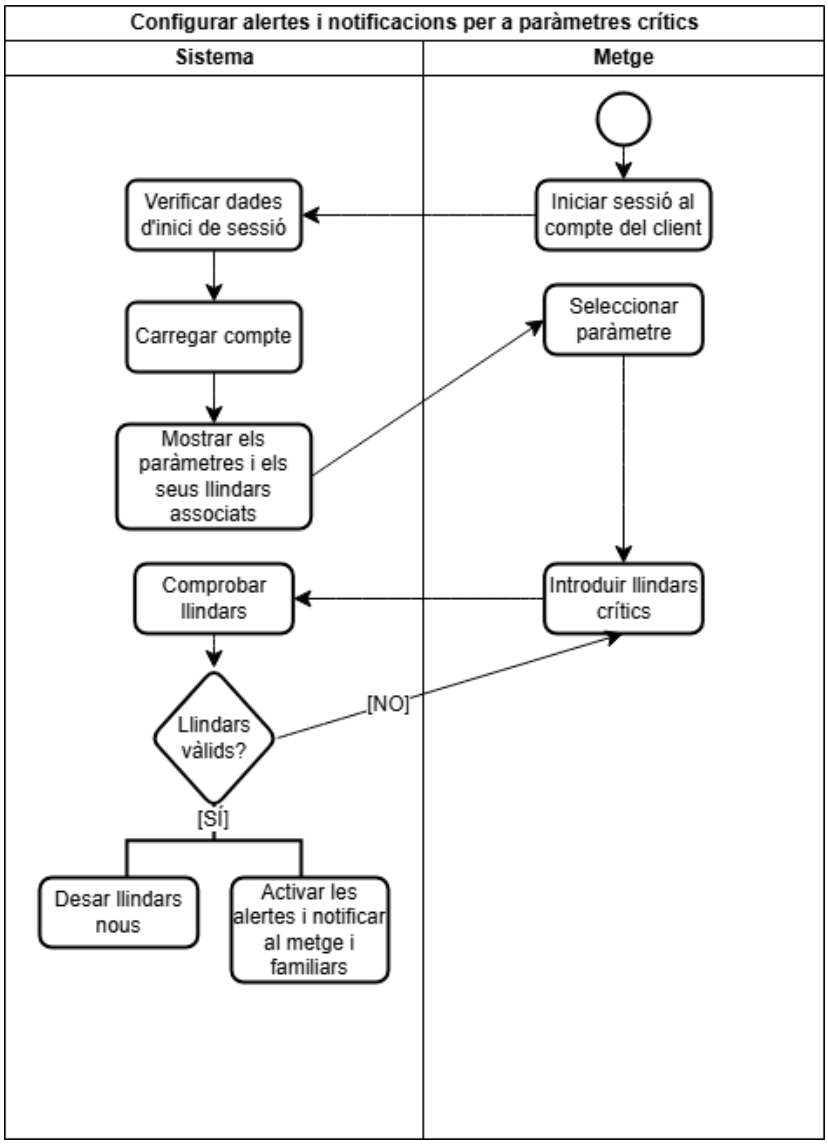

Diagrama d'Activitat 3. CU4: Configurar alertes i notificacions per a paràmetres crítics

2.3. Diagrames de seqüència

En aquest apartat hem desenvolupat els diagrames de seqüència de tres DCUS, concretament els CU1, CU2 i CU3. Els diagrames contemplen el flux principal, subfluxos i fluxos alternatius.

El Diagrama de Seqüència 1, il·lustra el cas d'ús 1. Doncs, hem dissenyat com un usuari (persona gran o un familiar) pot completar el registre mèdic bàsic, destacant les interaccions principals entre els actors i el sistema. Un dels motius pels qual hem escollit dissenyar aquest cas d'ús és perquè és essencial per habilitar les funcionalitats mèdiques de l'aplicació i garantir un seguiment adequat i segur de la salut dels usuaris.

En primer lloc, es mostra l'inci de sessió per part de l'usuari. Si el sistema no valida les dades introduïdes, l'usuari no podrà accedir a l'aplicació i continuar amb el procés. Per tal d'il·lustrar el bucle, hem especificat el símbol "*", el qual indica que fins que la sessió no sigui correcta ([sessió=sí]) no canviarà de pantalla. Aquest ens permet mantenir la seguretat de l'aplicació.

En segon lloc, el sistema carrega el compte/perfil de l'usuari i aquest escull l'opció adient. En aquest cas, l'opció és "Registre Mèdic". Destaquem doncs, que per a qualsevol acció dins de l'aplicació, s'hauran de realitzar aquests tres primers passos: inici, validació i escollir opció.

En tercer lloc, el sistema mostra el formulari i l'usuari l'emplena. D'igual forma, el sistema valida les dades introduïdes. En cas que estiguin incompletes, l'usuari ha de modificar-les tornant a omplir el formulari. Aquesta iteració no cessarà fins que el sistema consideri que les dades són correctes, p.e. tipus de dades o tots els camps obligatoris omplerts. Doncs, el sistema torna a actuar de filtre, en aquest cas per mantenir la integritat de les dades.

En quart lloc, el sistema concerta una cita amb el metge proposant-li els horaris disponibles a l'usuari i notifica al personal mèdic. A continuació, el metge revisa el registre del formulari.

En cinquè lloc, el metge realitza una conferència amb l'usuari per tal d'aprovar el registre. Només en cas de falta d'informació ([registre=rebutjat]) es que li realitza preguntes, les quals actualitza en el sistema. Seguidament, configura les alertes i els paràmetres.

Finalment, abans d'acabar la conferència, valida el resum. En cas que no sigui vàlid, mitjançant l'app, sol·licita una cita presencial. Per tant, l'usuari triarà la cita segons els

horaris disponibles del metge. En cas que sigui vàlid, es grava el registre i queda configurat.

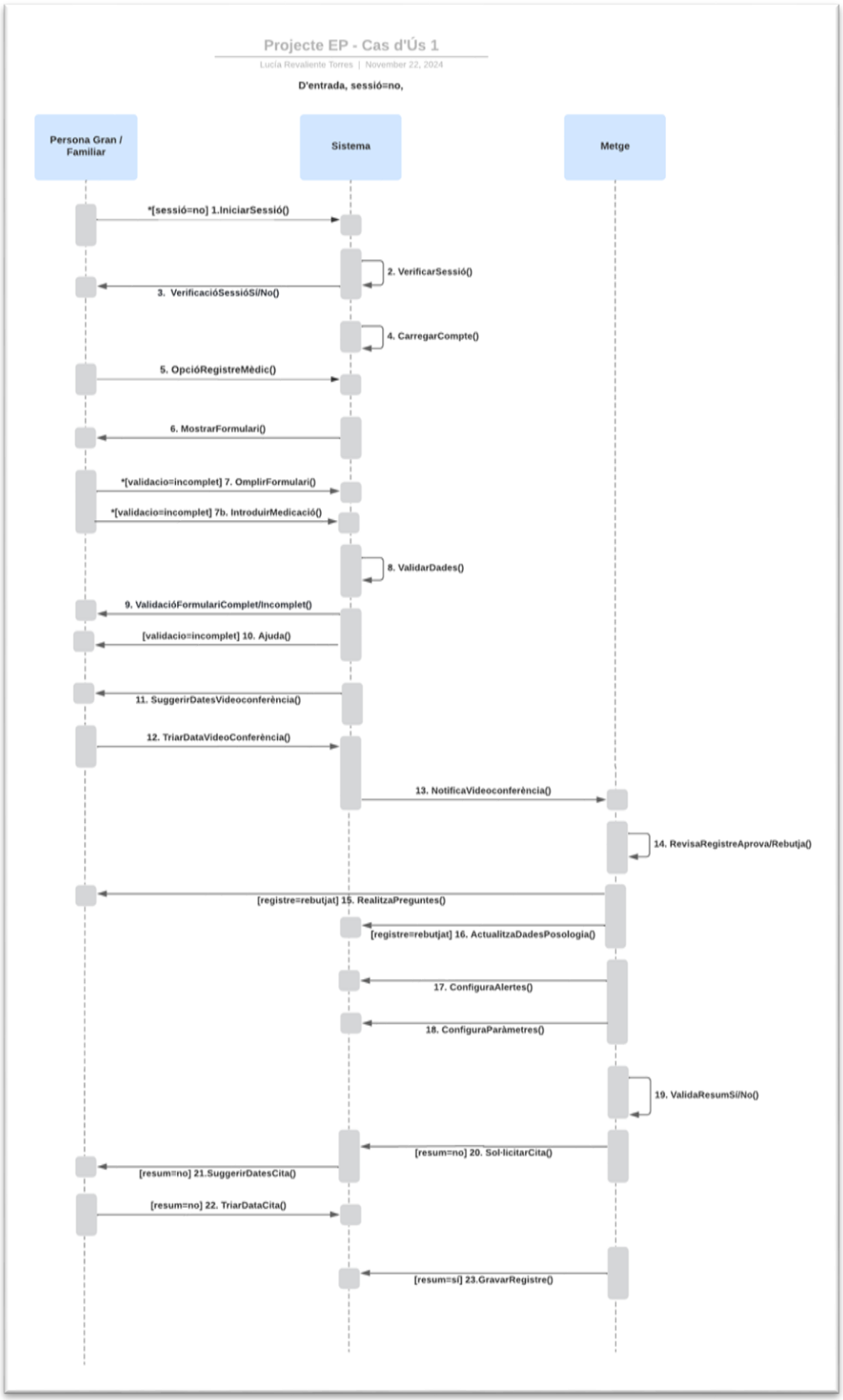

Diagrama de Seqüència 1. CU1: completar registre mèdic.

El Diagrama de Seqüència 2, il·lustra el cas d'ús 2 Doncs, hem dissenyat com una persona gran pot demanar ajuda a un voluntari, destacant les interaccions principals entre els actors i el sistema. Un dels motius pels qual hem escollit dissenyar aquest cas d'ús és perquè és essencial la comunicació entre els voluntaris i la gent gran, sobre tots amb persones que tenen mobilitat reduïda. D'aquesta manera, garantim que les necessitats bàsiques de les persones grans estiguin complertes i no se sentin sols.

En primer lloc, es mostra l'inci de sessió per part de l'usuari. Si el sistema no valida les dades introduïdes, l'usuari no podrà accedir a l'aplicació i continuar amb el procés. D'igual forma en el diagrama anterior, el bucle ens permet mantenir la seguretat de l'aplicació.

En segon lloc, el sistema carrega el compte/perfil de l'usuari i aquest escull l'opció adient. En aquest cas, l'opció és "Voluntari". Recordem que per a qualsevol acció dins de l'aplicació, s'hauran de realitzar aquests tres primers passos: inici, validació i escollir opció.

En tercer lloc, el sistema busca els voluntaris disponibles segons la ubicació i disponibilitat. Aquesta informació no s'especifica en el diagrama perquè és inherent de l'algorisme de cerca. A més, els voluntaris notifiquen al sistema quan són disponibles, però no és un missatge que necessàriament es realitza quan es duu a terme aquest cas d'ús.

En cas que no hi hagi cap persona voluntària disponible, el sistema es manté actiu cercant voluntaris. Doncs, hi ha un bucle amb la condició corresponent: *[voluntaris=indisponible]. Això assegura l'eficàcia de la nostra aplicació, perquè sempre acabarà trobant a algú. En cas que hi hagi voluntaris disponibles, el sistema mostra les persones amb opció de realitzar el servei, l'usuari escull a qui necessita i el sistema notifica al voluntari.

En quart lloc, el voluntari accepta o rebutja el servei. En cas que sigui denegat, el sistema informa a la persona gran i aquesta tornarà a demanar els voluntaris disponibles. Per tant, torna a il·lustrar-se un bucle "*" en el pas 5b, quan [sol·licitud=rebutja]. Escollim aquest pas i no el 9 perquè potser en el moment de reescollir, hi ha usuaris que estan ocupats fent un altre servei. Doncs, seria incorrecte i hauria una incoherència en el sistema.

En cas que l'usuari accepti el servei, també s'informaria a l'usuari però ara s'enregistra la sol·licitud i s'actualitza la disponibilitat de l'usuari en el sistema.

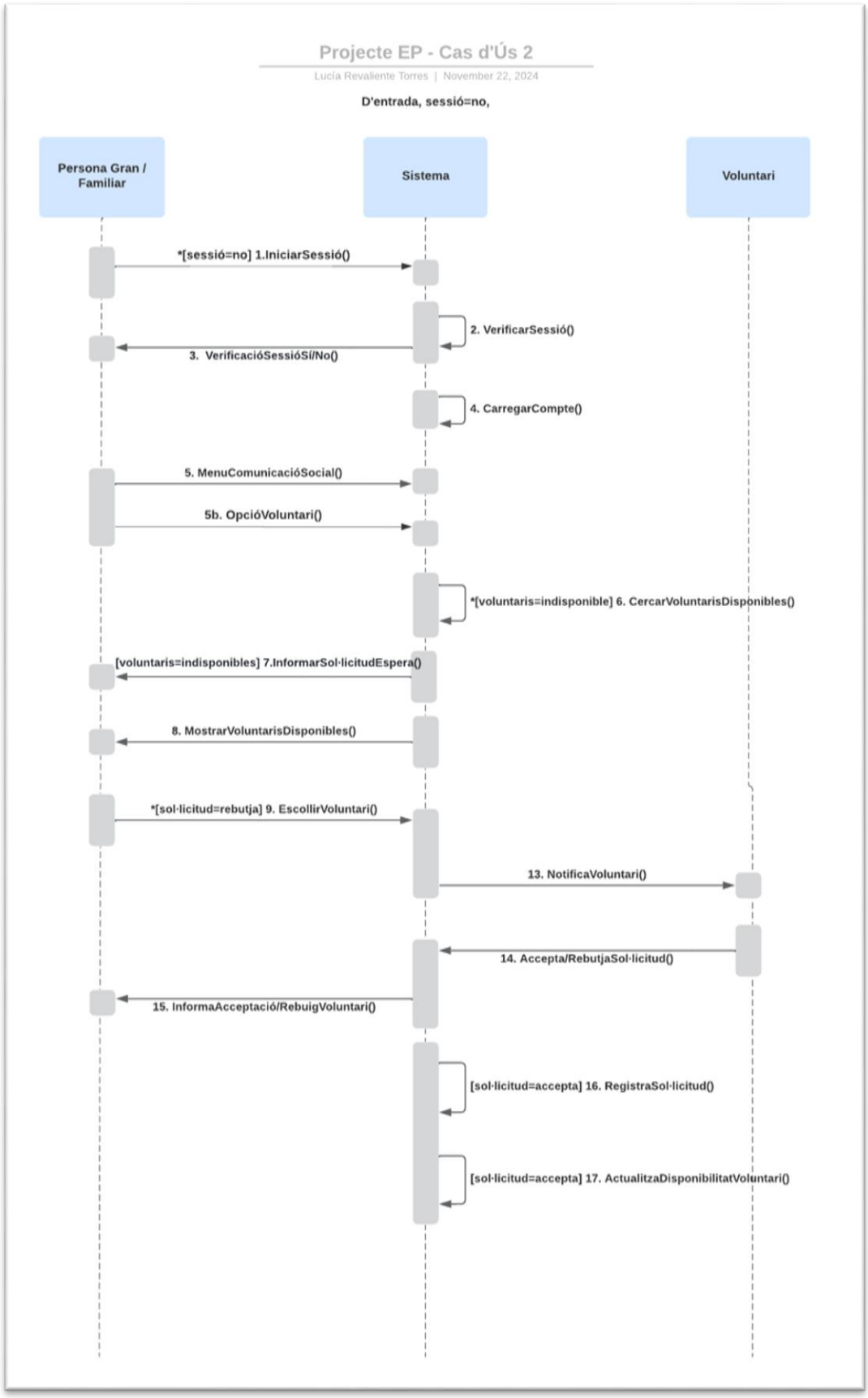
Projecte EP - Cas d'Ús 2
Lucía Revaliente Torres | November 22, 2024
D'entrada, sessió=no,
Persona Gran / Familiar
Sistema
Voluntari
*[sessió=no] 1.IniciarSessió()
2. VerificarSessió()
3. VerificacióSessióSí/No()
4. CarregarCompte()
5. MenuComunicacióSocial()
5b. OpcióVoluntari()
*[voluntaris=indisponible] 6. CercarVoluntarisDisponibles()
[voluntaris=indisponibles] 7.InformarSol·licitudEspera()
8. MostrarVoluntarisDisponibles()
*[sol·licitud=rebutja] 9. EscollirVoluntari()
13. NotificaVoluntari()
14. Accepta/RebutjaSol·licitud()
15. InformaAcceptació/RebuigVoluntari()
[sol·licitud=accepta] 16. RegistraSol·licitud()
[sol·licitud=accepta] 17. ActualitzaDisponibilitatVoluntari()

Diagrama de Seqüència 2. CU2: sol·licitar ajuda a un voluntari.

El Diagrama de Seqüència 3, il·lustra el cas d'ús "Programar una cita amb el metge". Hem dissenyat com un usuari (persona gran) pot sol·licitar i gestionar una cita amb el seu metge assignat, destacant les interaccions principals entre l'usuari, el sistema i el metge. Un dels motius pels quals hem escollit dissenyar aquest cas d'ús és perquè és essencial facilitar l'accés a l'atenció mèdica, garantint que el procés sigui intuïtiu, segur i eficient per a persones grans.

En primer lloc, es mostra el procés d'inici de sessió per part de l'usuari. Si el sistema no valida les dades introduïdes, l'usuari no podrà accedir a l'aplicació ni continuar amb el procés de sol·licitud de la cita. Per tal d'il·lustrar la validació, hem destacat aquest pas inicial com una interacció clau per garantir la seguretat de l'aplicació i protegir les dades sensibles dels usuaris.

En segon lloc, un cop validat l'inici de sessió, el sistema carrega el perfil de l'usuari, que posteriorment selecciona l'opció "Monitorització de la salut" i després "Contactes i consultes" per accedir a la funcionalitat de programació de cites. Destaquem que aquest flux inicial és comú a totes les accions dins l'aplicació, ja que inclou els passos essencials d'autenticació, validació i selecció d'opció.

A continuació, el sistema mostra la llista dels metges assignats i disponibles per a l'usuari. Aquest pot visualitzar les opcions i seleccionar el botó "sol·licitar cita" per indicar el metge i l'horari desitjat. Aquest pas és fonamental per assegurar que l'usuari pugui triar de manera fàcil i clara el professional mèdic que millor s'ajusti a les seves necessitats.

Seguidament, el sistema verifica automàticament la disponibilitat de l'horari seleccionat. Si l'horari és vàlid, el sistema reserva la cita i envia notificacions tant a l'usuari com al metge per assegurar que tots dos estan informats. Per altra banda, si l'horari no és vàlid, l'usuari té l'opció de sol·licitar una notificació quan hi hagi noves franges disponibles, mantenint així la comunicació activa i evitant frustracions.

Finalment, el sistema confirma la cita i envia notificacions amb els detalls finals, incloent-hi la data, l'hora i el metge assignat. Aquesta funcionalitat no només assegura que les dues parts estan al corrent, sinó que també garanteix que el procés de programació és àgil, clar i segur. D'aquesta manera, el diagrama subratlla la importància de mantenir una experiència intuïtiva per a persones grans, alhora que es respecten els estàndards d'accessibilitat i seguretat.

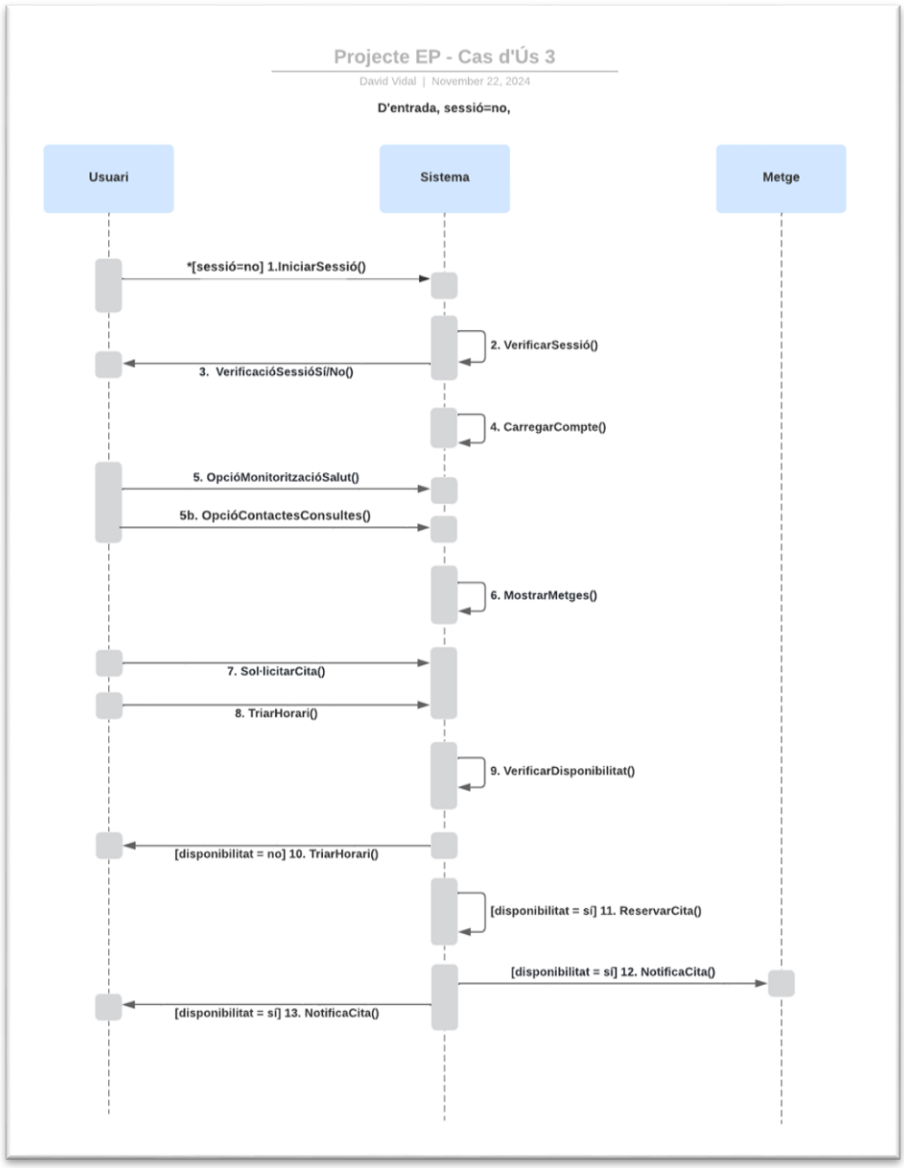

Diagrama de Seqüència 3. CU3: Programar una cita amb el metge.

2.4. Diagrames de classes per a xarxes socials en SeniorLife

El diagrama de classes següent representa el model conceptual del sistema SeniorLife, dissenyat per gestionar les interaccions socials i sanitàries de les persones grans. Aquest model integra diferents tipus d'usuaris, com familiars, amics i personal sanitari, i permet registrar visites mèdiques, així com la participació en xarxes socials. Mitjançant relacions ben definides i atributs clau, el diagrama assegura una estructura clara i escalable, adaptable a futures necessitats del sistema.

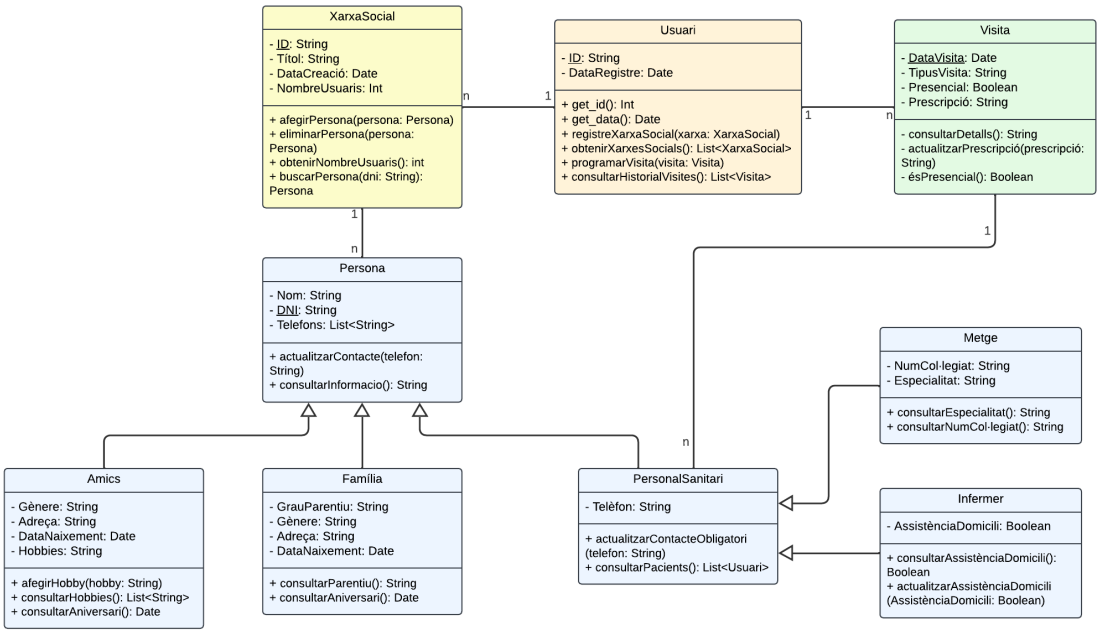

Diagrama de Classes 1. DC1: Diagrames de classes per a xarxes socials en SeniorLife.

2.4.1. Classes, atributs i mètodes

El sistema proposat necessita tant atributs per emmagatzemar informació com mètodes per realitzar accions concretes i mantenir el flux d'informació entre les classes. A continuació, es detalla la funció de cada classe, la justificació dels atributs i els mètodes associats:

1. Usuari
    a. Funció: Representa una persona registrada al sistema, que pot formar part de xarxes socials i interactuar amb professionals sanitaris.
    b. Atributs:

i. ID: Identificador únic de l'usuari. Justificació: Necessari per establir relacions amb altres entitats com XarxaSocial i Visita.
ii. DataRegistre: Data en què l'usuari es va registrar. Justificació: Ajuda a mantenir un registre cronològic dels usuaris i les seves activitats.

c. Mètodes:
i. get_id(): Retorna l'identificador de l'usuari. Justificació: Permet accedir a informació bàsica en altres parts del sistema.
ii. get_data(): Retorna la data de registre de l'usuari. Justificació: Facilita la consulta de l'historial de l'usuari.
iii. registreXarxaSocial(xarxa: XarxaSocial): Assigna una xarxa social a l'usuari. Justificació: Permet a l'usuari unir-se a una xarxa.
iv. programarVisita(visita: Visita): Registra una nova visita amb un professional sanitari. Justificació: Facilita la gestió d'interaccions sanitàries.
v. consultarHistorialVisites(): Retorna totes les visites associades a l'usuari. Justificació: Ofereix una visió completa de les interaccions sanitàries.

2. XarxaSocial
a. Funció: Representa un grup d'usuaris que comparteixen interessos i poden col·laborar entre ells.
b. Atributs:
i. ID: Identificador únic de la xarxa. Justificació: Necessari per identificar cada xarxa dins del sistema.
ii. Títol: Nom descriptiu de la xarxa social. Justificació: Permet distingir i donar significat a cada xarxa.
iii. DataCreació: Data de creació de la xarxa. Justificació: Ajuda a rastrejar l'activitat i l'antiguitat de la xarxa.
iv. NombreUsuaris: Nombre d'usuaris en la xarxa (derivat). Justificació: Proporciona una mesura directa del volum d'usuaris.
c. Mètodes:
i. afegirPersona(persona: Persona): Afegeix una persona a la xarxa. Justificació: Permet gestionar els membres de la xarxa.
ii. eliminarPersona(persona: Persona): Elimina una persona de la xarxa. Justificació: Manté la xarxa actualitzada.
iii. obtenirNombreUsuaris(): Calcula i retorna el nombre d'usuaris. Justificació: Proporciona una visió general de l'escala de la xarxa.
iv. buscarPersona(dni: String): Retorna una persona dins de la xarxa a partir del seu DNI. Justificació: Facilita la cerca específica de membres.

3. Persona

   a. Funció: Representa un individu que pot formar part d'una xarxa social.
   b. Atributs:
      i. Nom: Nom de la persona. Justificació: Identificació personal dins del sistema.
      ii. DNI: Document Nacional d'Identitat. Justificació: Necessari per assegurar la unicitat de cada individu.
      iii. Telèfons: Llista de números de contacte. Justificació: Permet contactar amb la persona per a diverses necessitats.
   c. Mètodes:
      i. actualitzarContacte(telèfon: String): Afegeix o modifica un número de telèfon. Justificació: Assegura que la informació de contacte estigui actualitzada.
      ii. consultarInformacio(): Retorna la informació personal. Justificació: Proporciona una visió general de la persona.
4. Família (subclasse de Persona)
   a. Funció: Representa un membre de la família d'un usuari.
   b. Atributs:
      i. GrauParentiu: Tipus de relació familiar. Justificació: Proporciona informació rellevant per a les relacions socials.
      ii. Gènere: Gènere del familiar. Justificació: Per finalitats estadístiques o personalitzades.
      iii. Adreça: Domicili del familiar. Justificació: Permet contactar-hi físicament si cal.
      iv. DataNaixement: Data de naixement. Justificació: Fomenta interacció (p. ex., felicitacions d'aniversari).
   c. Mètodes:
      i. consultarParentiu(): Retorna el grau de parentiu. Justificació: Facilita l'organització de les relacions familiars.
      ii. consultarAniversari(): Retorna la data de naixement. Justificació: Proporciona informació per fomentar la interacció.
5. Amics (subclasse de Persona)
   a. Funció: Representa un amic d'un usuari dins de la xarxa social.
   b. Atributs:
      i. Adreça: Domicili de l'amic. Justificació: Permet interactuar-hi físicament si cal.
      ii. Hobbies: Interessos compartits amb l'usuari. Justificació: Ajuda a identificar punts en comú.
   c. Mètodes:
      i. afegirHobby(hobby: String): Afegeix un hobby a l'amic. Justificació: Actualitza interessos compartits.

ii. consultarHobbies(): Retorna la llista d'hobbies. Justificació: Facilita la connexió entre usuaris.

6. PersonalSanitari (subclasse de Persona)
    a. Funció: Representa un professional sanitari que interactua amb usuaris.
    b. Atributs:
        i. Telèfon: Número de contacte obligatori. Justificació: Facilita la comunicació directa en casos sanitaris.
    c. Mètodes:
        i. consultarPacients(): Retorna la llista d'usuaris atesos. Justificació: Proporciona un registre dels pacients associats.
7. Metge (subclasse de PersonalSanitari)
    a. Funció: Representa un metge que presta serveis als usuaris.
    b. Atributs:
        i. NumCol·legiat: Número de col·legiació. Justificació: Assegura la identificació professional.
        ii. Especialitat: Àrea d'expertesa mèdica. Justificació: Permet assignar metges segons la necessitat.
    c. Mètodes:
        i. consultarEspecialitat(): Retorna l'especialitat del metge. Justificació: Facilita l'accés a serveis mèdics adequats.
8. Infermer (subclasse de PersonalSanitari)
    a. Funció: Representa un infermer que presta serveis als usuaris.
    b. Atributs:
        i. AssistènciaDomicili: Indica si presta serveis a domicili. Justificació: Facilita la gestió de serveis mèdics a casa.
    c. Mètodes:
        i. consultarAssistènciaDomicili(): Retorna si l'infermer realitza assistència a domicili. Justificació: Permet organitzar serveis personalitzats.
9. Visita
    a. Funció: Registra una interacció entre un usuari i un professional sanitari.
    b. Atributs:
        i. DataVisita: Data de la visita. Justificació: Registra quan va tenir lloc l'interacció.
        ii. TipusVisita: Tipologia de la visita. Justificació: Classifica les visites (p. ex., consulta mèdica, seguiment).
        iii. Presencial: Indica si la visita és presencial o virtual. Justificació: Registra la modalitat de l'interacció.
        iv. Prescripció: Informació de medicaments prescrits. Justificació: Registra recomanacions mèdiques.
    c. Mètodes:

i. consultarDetalls(): Retorna els detalls de la visita. Justificació: Facilita el seguiment mèdic.
ii. actualitzarPrescripció(prescripció: String): Modifica la prescripció. Justificació: Permet actualitzar la informació mèdica associada.

2.5.2. Relacions

El diagrama mostra diverses relacions entre les classes per modelar les interaccions entre els diferents elements del sistema. A continuació, es detallen aquestes relacions amb la seva justificació:

1. Relació Usuari - XarxaSocial
    a. Tipus de relació: Associació (1 ↔ n)
    b. Descripció: Un usuari pot estar associat a una o més xarxes socials, mentre que una xarxa social pot tenir múltiples usuaris.
    c. Justificació: Això reflecteix que cada usuari és membre d'almenys una xarxa social, fomentant la interacció social.
2. Relació Usuari - Visita
    a. Tipus de relació: Associació (1 ↔ n)
    b. Descripció: Un usuari pot tenir diverses visites registrades amb professionals sanitaris, mentre que cada visita està associada a un sol usuari.
    c. Justificació: Permet portar un historial complet de les visites mèdiques d'un usuari.
3. Relació Visita - PersonalSanitari
    a. Tipus de relació: Associació (n ↔ 1)
    b. Descripció: Una visita pot involucrar un únic professional sanitari, mentre que un professional sanitari pot atendre múltiples visites.
    c. Justificació: Facilita la identificació del professional sanitari responsable de cada interacció.
4. Relació XarxaSocial - Persona
    a. Tipus de relació: Associació (1 ↔ n)
    b. Descripció: Una xarxa social pot tenir diverses persones com a membres, incloent familiars, amics i personal sanitari, mentre que cada persona pot pertànyer a una sola xarxa social.
    c. Justificació: Modela la composició de les xarxes socials amb diferents tipus de membres.
5. Relació Persona - Família
    a. Tipus de relació: Generalització/Herència
    b. Descripció: La classe "Família" hereta de "Persona".

   c. Justificació: Permet tractar els familiars com un tipus específic de persona, afegint atributs i comportaments específics.
6. Relació Persona - Amics
   a. Tipus de relació: Generalització/Herència
   b. Descripció: La classe "Amics" hereta de "Persona".
   c. Justificació: Permet definir els amics com un tipus especialitzat de persona amb atributs addicionals, com els hobbies.
7. Relació Persona - PersonalSanitari
   a. Tipus de relació: Generalització/Herència
   b. Descripció: La classe "PersonalSanitari" hereta de "Persona".
   c. Justificació: Permet tractar el personal sanitari com un tipus específic de persona amb atributs addicionals, com el telèfon obligatori.
8. Relació PersonalSanitari - Metge
   a. Tipus de relació: Generalització/Herència
   b. Descripció: La classe "Metge" hereta de "PersonalSanitari".
   c. Justificació: Defineix els metges com un tipus especialitzat de personal sanitari amb atributs específics com el número de col·legiat.
9. Relació PersonalSanitari - Infermer
   a. Tipus de relació: Generalització/Herència
   b. Descripció: La classe "Infermer" hereta de "PersonalSanitari".
   c. Justificació: Defineix els infermers com un tipus especialitzat de personal sanitari amb atributs específics com l'assistència domiciliària.

Observació sobre cardinalitats:

1. Les cardinalitats es representen de la següent manera:
   a. 1 ↔ n: Un element de la primera classe pot estar relacionat amb diversos elements de la segona classe, però cada element de la segona classe està associat amb un únic element de la primera.
   b. n ↔ 1: Diversos elements de la primera classe poden estar relacionats amb un únic element de la segona classe.

2.5.3. Observacions

2.4.4. Observacions

Durant el disseny del diagrama de classes i l'anàlisi de les relacions, s'han realitzat diverses decisions per optimitzar la modularitat, escalabilitat i mantenibilitat del sistema. A continuació, es detallen algunes observacions rellevants:

1. **Relació entre Persona i subclasses (Família, Amics, PersonalSanitari):**

Inicialment, es va considerar no incloure una classe "Persona" i definir cadascun dels rols (Família, Amics, PersonalSanitari) com a entitats separades. Tanmateix, per evitar duplicació de atributs comuns (com `Nom`, `DNI`, i `Telefons`), es va optar per crear una relació d'herència que permet reutilitzar els atributs compartits de manera eficient.

2. **Modularitat en PersonalSanitari:**
   Les subclasses `Metge` i `Infermer` refinen el concepte general de personal sanitari. Aquest disseny permet estendre el model fàcilment per afegir altres tipus de professionals, si calgués, mantenint la consistència de la classe base.
3. **Decisió sobre atributs immutables en Visita:** els atributs com `DataVisita` i `TipusVisita` es van definir com immutables, ja que un cop registrades, les visites no haurien de poder modificar-se. Això garanteix la integritat de l'historial mèdic i evita inconsistències.
4. **Relació Usuari-XarxaSocial:** Es va establir una cardinalitat obligatòria d'`1 ↔ n`, assegurant que cada usuari ha de formar part d'almenys una xarxa social. Això reflecteix el requisit funcional que els usuaris del sistema estiguin connectats socialment.
5. **Inclusió del telèfon obligatori en PersonalSanitari:** Es va decidir que aquest atribut sigui obligatori per garantir que qualsevol professional sanitari pugui ser contactat en cas de necessitat. Això respon a requisits de seguretat i comunicació immediata.

2.5. Diagrames de classes per la monitorització de la salut

El diagrama de classes UML presentat representa un sistema complex per monitoritzar constants vitals mitjançant dispositius IoT i gestionar alertes. Tot en aquest diagrama està justificat per satisfer els requeriments d'un sistema mèdic funcional i escalable. A continuació, s'explica i justifica cada part del diagrama.

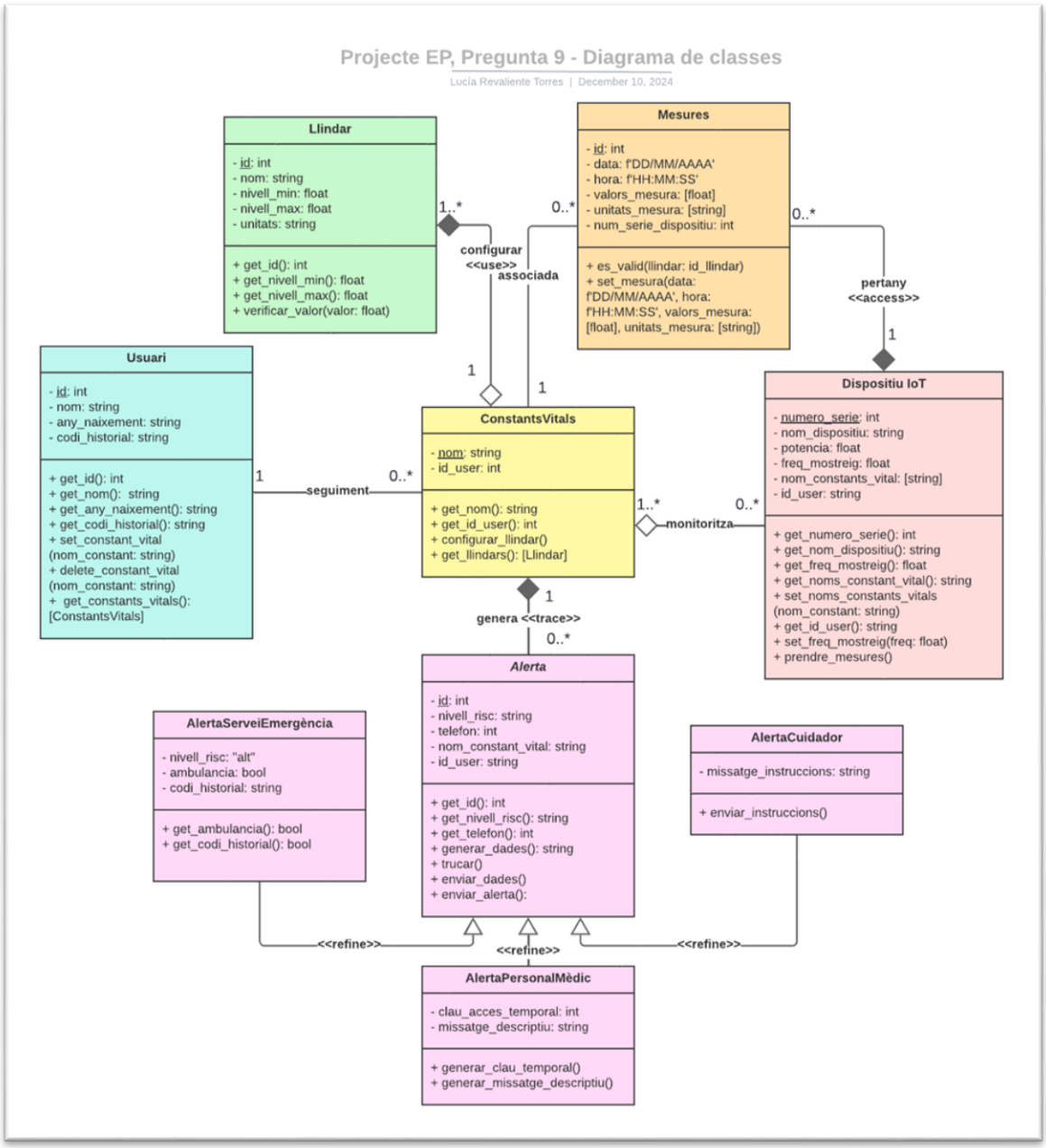

Diagrama de Classes 2. DC2: Diagrama de classes per la monitorització de la salut.

2.5.1. Classes, atributs i mètodes

El sistema proposat no només necessita atributs per emmagatzemar dades, sinó també mètodes per realitzar accions concretes i mantenir el flux d'informació entre les classes. A continuació, es detalla la funció de cada classe, justificació dels atributs, i la inclusió dels mètodes per cada classe:

1. **Usuari:**
    a. **Funció:** Representa una persona usuària del sistema, que pot ser monitoritzada o rebre alertes.
    b. **Atributs:**
        i. **id:** Identificador únic per a cada usuari. Justificació: Necessari per identificar un usuari en les relacions amb altres classes, com ConstantsVitals o Dispositiu IoT.
        ii. **nom:** Nom de l'usuari. Justificació: Facilita la identificació humana i el maneig de la informació personal.
        iii. **any_naixement:** Any de naixement de l'usuari. Justificació: Permet calcular l'edat, un factor clau per interpretar dades mèdiques (p. ex., freqüència cardíaca normal segons l'edat).
        iv. **codi_historial:** Codi que enllaça l'usuari amb el seu historial mèdic. Justificació: Proporciona accés a informació mèdica addicional, com malalties preexistents o tractaments en curs.
    c. **Mètodes**:
        i. **get_id(), get_nom(), get_any_naixement(), get_codi_historial():** Retornen la informació bàsica de l'usuari. Justificació: Faciliten la consulta de dades per altres components del sistema.
        ii. **set_constant_vital (nom_constant: string) i delete_constant_vital (nom_constant: string):** Permet afegir i eliminar constants a monitorar associades a l'usuari. Justificació: L'evolució del monitoreig dels pacients pot canviar i potser es necessiten afegir o deixar de mesurar constants.
        iii. **get_constants_vitals():** Retorna les constants vitals associades a aquest usuari. Justificació: Connecta l'usuari amb els elements que s'estan monitoritzant, permetent accedir a dades de salut.
2. **Constants vitals:**
    a. **Funció**: Representa les constants vitals d'un usuari i els seus límits associats.
    b. **Atributs**:
        i. **nom:** Nom de la constant vital. Identificador únic.
        ii. **id_user:** Enllaça la constant vital amb un usuari. Justificació: Com la clau primària es nom, per poder filtrar segons el pacient, hem afegit la id_user. Doncs, és una clau forana.
    c. **Mètodes**:

i. **get_nom():** Retorna el nom de la constant vital. Justificació: Proporciona identificació per operacions i alertes.
ii. **get_id_user():** Retorna l'ID de l'usuari associat. Justificació: Necessari per traçar la informació al propietari de la dada.
iii. **configurar_llindar():** Enllaça la constant vital amb un límit (instància de Línidar). Justificació: Permet ajustar els límits per a una constant específica.
iv. **get_llindars():** Retorna els límits associats a aquesta constant vital. Com pot haver, més d'un llindar, es retorna una llista de classes Llindar, és a dir, amb tota la informació continguda. Justificació: Facilita l'accés als límits per a verificacions i validacions.

3. **Dispositius IoT:**
   a. **Funció:** Representa un dispositiu físic que recull dades de constants vitals.
   b. **Atributs:**
      i. **num_serie:** Identificador únic del dispositiu. Justificació: Necessari per identificar dispositius individuals en sistemes amb múltiples sensors.
      ii. **nom_dispositiu:** Nom descriptiu del dispositiu (p. ex., "Tensiòmetre"). Justificació: Ajuda a identificar fàcilment el tipus de dispositiu.
      iii. **potencia:** Estat de la potència del dispositiu. Justificació: Per assegurar que els dispositius estiguin operatius.
      iv. **freq_mostreig:** Freqüència amb què el dispositiu recull dades. Justificació: Permet adaptar la freqüència de mostreig segons les necessitats mèdiques.
      v. **nom_constants_vitals:** Llista de constants vitals monitoritzades pel dispositiu. Justificació: Un dispositiu pot monitoritzar múltiples constants, com un dispositiu que mesura temperatura i freqüència cardíaca.
      vi. **id_user:** Identificador de l'usuari associat al dispositiu. Justificació: Permet enllaçar un dispositiu a una persona. Doncs, és una clau forània.
   c. **Mètodes:**
      i. **get_numero_serie():** Retorna el número de sèrie del dispositiu. Justificació: Necessari per identificar dispositius en registres i bases de dades.
      ii. **get_nom_dispositiu():** Retorna el nom del dispositiu. Justificació: Proporciona identificació per manteniments i operacions.

iii. **get_freq_mostreig():** Retorna la freqüència de mostreig actual. Justificació: Permet verificar si el dispositiu està configurat correctament.
iv. **set_freq_mostreig(freq: float):** Configura una nova freqüència de mostreig. Justificació: Permet personalitzar el ritme de captura de dades segons les necessitats, ja que aquest pot canviar al llarg de l'estudi mèdic.
v. **prendre_mesures():** Captura una mesura i l'enregistra. Justificació: Funció essencial per obtenir dades del dispositiu.

4. **Mesures:**
   a. **Funció:** Representa les dades recollides per un dispositiu en un moment concret.
   b. **Atributs:**
      i. **id:** Identificador únic de la mesura. Justificació: Necessari per diferenciar mesures individuals, especialment en grans bases de dades.
      ii. **data:** Data en format DD/MM/AAAA. Justificació: Permet registrar el moment en què es va fer la mesura.
      iii. **hora:** Hora en format HH:MM:SS. Justificació: Afegir precisió temporal a les mesures.
      iv. **valors_mesura:** Una llista de valors flotants obtinguts pel dispositiu. Justificació: Cada dispositiu pot recollir múltiples valors alhora.
      v. **unitats_mesura:** Les unitats dels valors obtinguts. Justificació: Assegura una interpretació correcta dels valors (p. ex., °C per temperatura).
      vi. **num_serie_dispositiu:** Número de sèrie del dispositiu que va generar la mesura. Justificació: Identifica quin dispositiu va fer la mesura, cosa que és útil en cas d'errors tècnics o calibració.
   c. **Mètodes:**
      i. **es_vàlid(línidar: Línidar):** Comprova si una mesura és vàlida segons un límit donat. Justificació: Permet validar directament les mesures a partir dels límits configurats.
      ii. **set_mesura(data: f'DD/MM/AAAA', hora: f'HH:MM:SS', valors_mesura: [float], unitats_mesura: [string]):** Afegeix una mesura. Justificació: la classe Dispositiu IoT li passa els paràmetres ja que no pot accedir a ella (per seguretat).

5. **Llindar:**
   a. **Funció:** Representa els límits d'una constant vital per determinar si una mesura és normal o anòmala.
   b. **Atributs:**

i. **id:** Identificador únic del límit. Justificació: Permet referenciar un límit específic en les relacions amb altres entitats, com ConstantsVitals.
ii. **nom:** Nom descriptiu del límit (p. ex., "Pressió arterial sistòlica"). Justificació: Facilita l'organització i comprensió del sistema.
iii. **nivell_min i nivell_max:** Defineixen el rang acceptable per a una mesura. Justificació: Aquests atributs són essencials per a la validació de mesures i la detecció de valors fora de rang.
iv. **unitats:** Les unitats (p. ex., "mmHg", "bpm") garanteixen la coherència en la interpretació dels valors. Justificació: Això és important en un sistema mèdic per evitar errors d'interpretació de mesures.

c. Mètodes:

i. **get_id():** Retorna l'identificador del límit. Justificació: Necessari per identificar el límit en altres operacions.
ii. **get_nivell_min() i get_nivell_max():** Permeten consultar els valors mínim i màxim. Justificació: Ajuda a verificar les mesures sense modificar-les.
iii. **verificar_valor(valor: float):** Comprova si una mesura concreta està dins del límit. Justificació: Permet validar automàticament dades recollides per dispositius.

6. Alertes:

a. Funció: Representa alertes generades per mesures fora dels límits. És una classe abstracta.

b. Atributs:

i. **id:** Identificador únic de l'alerta. Justificació: Necessari per gestionar múltiples alertes en el sistema.
ii. **nivell_risc:** Descriu el risc de l'alerta (p. ex., "alt", "mitjà"). Justificació: Permet prioritzar respostes.
iii. **telefon:** Contacte associat per gestionar l'alerta. Justificació: Facilita les notificacions a cuidadors o serveis mèdics.
iv. **nom_constant_vital:** Nom de la constant vital relacionada amb l'alerta. Justificació: Identifica la causa de l'alerta.
v. **id_user:** Identificador de l'usuari afectat. Justificació: Enllaça l'alerta amb una persona.

c. Mètodes:

i. **get_id():** Retorna l'ID de l'alerta. Justificació: Necessari per identificar alertes en sistemes amb múltiples notificacions.
ii. **get_nivell_risc():** Retorna el nivell de risc de l'alerta. Justificació: Permet prioritzar les alertes segons la gravetat.
iii. **get_telefon():** Retorna el número de telèfon associat. Justificació: Necessari per notificar els contactes corresponents.

iv. **generar_dades():** Genera dades necessàries per enviar l'alerta. Justificació: Proporciona informació bàsica per construir notificacions.
v. **enviar_alerta():** Envia l'alerta al destinatari. Justificació: Funció crítica per notificar situacions anòmales.

7. **Subclasses d'alerta:**
   a. **AlertaServeiEmergència**
      i. **Funció:** Gestiona alertes crítiques que requereixen una resposta immediata.
      ii. **Atributs:**
         1. **ambulancia:** Indica si s'ha activat el servei d'emergència.
         2. **codi_historial:** Proporciona dades addicionals per als serveis mèdics.
      iii. **Mètodes:**
         1. **get_ambulancia():** Retorna si l'ambulància ha estat activada. Justificació: Permet verificar l'estat de la resposta d'emergència.
         2. **get_codi_historial():** Retorna el codi d'historial associat. Justificació: Proporciona informació per una atenció mèdica efectiva.
   b. **AlertaPersonalMèdic**
      i. **Funció:** Gestiona alertes adreçades a professionals mèdics.
      ii. **Atributs:**
         1. **clau_acces_temporal:** Clau per accedir a dades privades.
         2. **missatge_descriptiu:** Explicació de la situació.
      iii. **Mètodes:**
         1. **generar_clau_temporal():** Genera una clau per accedir temporalment a dades. Justificació: Assegura un accés controlat.
         2. **generar_missatge_descriptiu():** Crea un resum de la situació mèdica. Justificació: Facilita la comprensió per part dels professionals.
   c. **AlertaCuidador**
      i. **Funció:** Gestiona alertes destinades a cuidadors.
      ii. **Atributs:**
         1. **missatge_instruccions:** Instruccions per al cuidador.
      iii. **Mètodes:**
         1. **enviar_instruccions():** Envia les instruccions al cuidador. Justificació: Proporciona una resposta ràpida i efectiva.

2.5.2. Relacions

El diagrama de classes mostra diverses relacions entre les classes. Aquestes relacions inclouen associacions, agregacions, composicions i generalitzacions/herències, cadascuna dissenyada per modelar adequadament les interaccions entre els elements del sistema. A continuació, s'expliquen les relacions i dependències amb la seva justificació:

1. **Relació Llindar - ConstantsVitals (1.. ↔ 1)***
    a. **Tipus de relació:** Agregació.
    b. **Descripció:** Cada constant vital (instància de la classe ConstantsVitals) pot estar associada amb un o més límits (Línidar), que defineixen els valors acceptables per a aquesta constant. A més, un límit pot ser reutilitzat per diferents constants vitals, per això el símbol * fora del parèntesis.
    c. **Justificació:** Aquesta agregació permet adaptar els límits segons les necessitats mèdiques (per exemple, els nivells acceptables de pressió arterial poden variar segons l'edat o l'historial mèdic de l'usuari). D'altra banda, el disseny evita la duplicació innecessària de dades perquè un mateix límit pot ser compartit.
2. **Relació ConstantsVitals - Línidar**
    a. **Tipus de relació:** Dependència.
    b. **Descripció:** La validació de les mesures d'una constant vital depèn dels límits definits a Línidar.
    c. **Justificació:** Aquesta dependència assegura que el sistema pugui validar automàticament les mesures, sense que ConstantsVitals tingui una relació rígida amb Línidar. Això fa que els límits puguin ser configurats o ajustats dinàmicament sense afectar les constants vitals.
    d. **Dependència afegida <<use>>:** Constants vitals fa servir els límits definits en Llindar per avaluar si els mesures estan dins dels valors permesos i generar alertes. Això és una dependència funcional.
3. **Relació Usuari - ConstantsVitals (1 ↔ 0..*)**
    a. **Tipus de relació:** Associació.
    b. **Descripció:** Cada usuari pot tenir associades zero o més constants vitals per monitoritzar. A més, cada constant vital està relacionada amb un sol usuari.
    c. **Justificació:** Aquesta associació reflecteix la personalització del sistema: diferents usuaris poden tenir diferents constants vitals a monitoritzar (per exemple, un usuari pot necessitar monitoritzar la pressió arterial i un altre només la temperatura). D'altra banda, el disseny assegura que cada constant vital estigui clarament associada al seu propietari.
4. **Relació Dispositiu IoT - ConstantsVitals (0.. ↔ 1)***
    a. **Tipus de relació:** Agregació.

   b. **Descripció:** Un dispositiu IoT pot monitoritzar zero o més constants vitals, mentre que cada constant vital ha d'estar monitoritzada per un dispositiu.
   c. **Justificació:** Aquesta relació permet la configuració flexible del sistema, on un dispositiu pot mesurar diverses constants vitals (per exemple, un dispositiu pot mesurar la temperatura i la freqüència cardíaca). D'altra banda, també permet escalar el sistema afegint nous dispositius IoT segons sigui necessari.

5. **Relació Dispositiu IoT - Mesures (1 ↔ 0..*)**
   a. **Tipus de relació:** Composició.
   b. **Descripció:** Cada dispositiu IoT genera zero o més mesures. Les mesures no poden existir sense el dispositiu que les ha creat.
   c. **Justificació:** Aquesta composició és essencial perquè les mesures només tenen sentit dins del context del dispositiu que les ha generat. A més, si un dispositiu es retira o es desactiva, totes les mesures associades poden deixar de ser vàlides o es poden esborrar del sistema
   d. **Dependència afegida <<access>>:** El dispositiu IoT accedeix a les dades de Mesures perquè s'encarrega d'enregistrar i obtenir les mesures realitzades. No fa servir directament la funcionalitat de Mesures, sinó que accedeix a El Dispositiu IoT accedeix a les dades de Mesures perquè s'encarrega de registrar i obtenir las mesures realitzades. No "utilitza" directament la funcionalitat de Mesures, sinó que accedeix a les dades generades per aquesta classe.
6. **Relació ConstantsVitals - Mesures (1 ↔ 0..*)**
   a. **Tipus de relació:** Associació.
   b. **Descripció:** Cada mesura està associada a una constant vital específica (per exemple, una mesura de "36.5" pot correspondre a la constant vital "Temperatura").
   c. **Justificació:** Aquesta associació assegura que cada mesura pugui ser interpretada correctament dins del seu context, associant-la a la constant vital correcta i validant-la contra els límits configurats.
7. **Relació Alerta amb subclasses (AlertaServeiEmergència, AlertaPersonalMèdic, AlertaCuidador)**
   a. **Tipus de relació:** Generalització/Herència.
   b. **Descripció:** La classe Alerta és la classe base per a les subclasses (abstracta), que representen diferents tipus d'alertes. Cada subclasse afegeix atributs i mètodes específics (p. ex., ambulancia a AlertaServeiEmergència o missatge_instruccions a AlertaCuidador).
   c. **Justificació:** Aquesta herència facilita l'estructura i l'extensibilitat del sistema: totes les alertes comparteixen atributs i comportaments comuns (p. ex., generar_dades() i enviar_alerta()), mentre que cada tipus d'alerta pot tenir funcionalitats úniques. Cal destacar que el disseny també promou la

reutilització de codi i simplifica la gestió de diferents tipus d'alertes en el sistema.

d. **Dependència afegida <<refine>>:** Les subclasses refinen el concepte general d'Alerta adaptant-lo als cassos específics (emergències, mèdics, cuidadors). Això mostra una relació d'especificació.

2.5.3. Observacions

A continuació, presentem certes observacions realitzades durant el disseny del diagrama:

1. En la classe Constants Vitals, havíem afegit el mètode **configurar_alerta(valor: float)** però l'hem eliminat perquè aquestes es generen automàticament pel sistema, quan els valors de les mesures no segueixen el llindar establert.
2. En la classe Dispositiu IoT, havíem creat el mètode **get_potencia(): float**, però l'hem eliminat perquè no té ús en el sistema
3. En la classe Dispositiu IoT no hem afegit un atribut que emmagatzemi totes les mesures preses perquè aquestes ja referencien des de quin dispositiu s'han mesurat.
4. Entre la classe Mesures i Constants vitals vam afegir una relació de dependència, la qual vam eliminar en adonar-nos que aquesta afegia redundància al sistema. Ja que aquesta ja està definida perquè Mesures depèn de Dispositius IoT i aquests de Constants Vitals. A més, que els dispositius ja contenen la constant que estan mesurant, per tant, no cal cap relació.
5. En la classe Llindar, tot i que es podrien establir els atributs com a protegits, els hem mantingut privats. El motiu és perquè preferim accedir a les classes mitjançant mètodes i no modificar els atributs de manera directa. Així, mantenim la integritat i seguretat del sistema.
6. En la classe Llindar, havíem dissenyat els mètodes **set_nivell_min(min: float)** i **set_nivell_max(max: float)**. Els hem eliminat perquè són redundants, ja que els límits no es canvien dinàmicament en el sistema, sinó que els valors es configuren una única vegada al crear-se l'objecte.
7. En un principi, vam establir la relació Llindar-Constants Vitals com dependent. Però ho vam canviar ja que és una relació persistent en el temps, no transitòria. A més, Llindar pot existir independentment de ConstantsVitals i pot ser reutilitzat. D'altra banda, reflecteix una configuració del sistema estable i de llarg termini. Finalment, una dependència seria massa feble i no representaria adequadament la naturalesa estructural d'aquesta relació.
8. Abans de revisar el disseny, vam establir la relació Constants Vitals - Dispositius IoT com associació. Però ho vam canviar a agregació ja que reflecteix la relació "part-tot" on cada part (les constants vitals) pot existir independentment del tot (el dispositiu). A més, ara manté la modularitat i escalabilitat del sistema.

Finalment, expressa millor que el dispositiu "agrega" constants vitals per monitoritzar-les, sense que aquestes constants depenguin exclusivament del dispositiu.

En conclusió, aquest disseny modular, amb classes i mètodes clarament definits, garanteix escalabilitat, mantenibilitat i flexibilitat en un sistema crític com aquest. A més, assegura que les relacions entre les entitats siguin coherents, escalables i fàcils de mantenir.

3. Implementació

Aquesta secció exposa el desenvolupament i implementació de l'aplicació. S'aborden aspectes com el codi inicial, patrons de disseny i el compliment de bones pràctiques en l'enginyeria del software.

3.1. Patrons de disseny

Per assegurar un desenvolupament adequat d'aquesta aplicació, s'ha seguit un conjunt de patrons de disseny que garanteixen una organització clara i eficient del sistema.

El primer patró de disseny aplicat en l'elaboració d'aquesta aplicació/pàgina web és el Model-Vista-Controlador (MVC).
Aquest patró assegura una separació adequada entre:

- Backend, responsable de la gestió de la informació i la lògica de negoci.
- Frontend, encarregat de mostrar la informació a l'usuari i recollir-ne dades.
- Base de dades, on es guarda la informació.

Aquesta separació facilita la comprensió i manteniment del codi.

Un altre patró utilitzat en el desenvolupament de l'aplicació és el Singleton. Aquest patró garanteix que només hi hagi una instància d'una classe al llarg de l'aplicació. Per exemple, quan definim la base de dades, ens assegurem que aquesta instància es comparteix per tota l'aplicació, evitant la creació de múltiples connexions innecessàries.

Finalment, també s'ha implementat el patró Factory. Aquest patró s'utilitza per crear objectes sense especificar explícitament la classe concreta que s'ha d'instanciar. Quan creem l'aplicació amb Flask, obtenim una instància ja configurada prèviament, que facilita la inicialització de l'aplicació.

3.2. Implementació inicial del sistema

L'aplicació s'ha desenvolupat utilitzant Flask com a framework principal, gràcies a la seva flexibilitat per gestionar aplicacions web i la seva integració senzilla amb Python, un llenguatge que tots els membres de l'equip dominen.

S'han definit tres classes diferents: User, Appointment i MedicalRecord. Aquestes tres classes corresponen a les taules que s'emmagatzemen a la base de dades i s'encarreguen de definir i gestionar els processos necessaris per afegir noves entrades a la base de dades. A més, garanteixen la realització de totes les comprovacions requerides sobre els valors crítics.

A banda de les classes, també s'ha implementat un conjunt de funcions. Aquestes funcions han estat nomenades segons la seva utilitat dins del sistema. Per exemple, la funció upload gestiona tots els processos relacionats amb el segment del sistema on es carreguen els historials mèdics.

Finalment, aquestes funcions actuen com a pont entre el frontend i la base de dades, gestionant les peticions generades al servidor i emmagatzemant totes les dades necessàries de forma eficient.

Per la part de la base de dades, s'ha integrat SQLite, un ORM que facilita la definició d'estructures de dades i la seva manipulació sense necessitat d'escriure consultes SQL manualment.

S'ha optat per una base de dades relacional, on cada taula representa una entitat clau del sistema. La taula User emmagatzema informació dels usuaris, com el nom, correu electrònic i edat, assegurant que cada registre sigui únic i vàlid. La taula Appointment s'encarrega de gestionar les cites mèdiques, amb la restricció de no permetre duplicats per a una mateixa data i hora per usuari. Finalment, la taula MedicalRecord guarda informació mèdica associada als usuaris, com el grup sanguini i possibles al·lèrgies.

Això es reflecteix al Diagrama Entitat-Relació, que defineix les entitats principals de l'aplicació (User, Appointment i MedicalRecord) i les seves relacions.

L'entitat User representa els usuaris, amb atributs com nom, cognom, correu electrònic únic, contrasenya i edat validada (18-120 anys). Aquesta entitat està vinculada amb les altres dues a través de relacions d'un a molts.

L'entitat Appointment gestiona les cites mèdiques dels usuaris. Cada cita està associada a un usuari mitjançant una clau forana, i es garanteix que les cites siguin úniques per usuari, data i hora, evitant conflictes.

L'entitat MedicalRecord emmagatzema informació mèdica, com el grup sanguini i les al·lèrgies, també vinculada a un únic usuari, permetent múltiples registres per usuari però evitant duplicats.

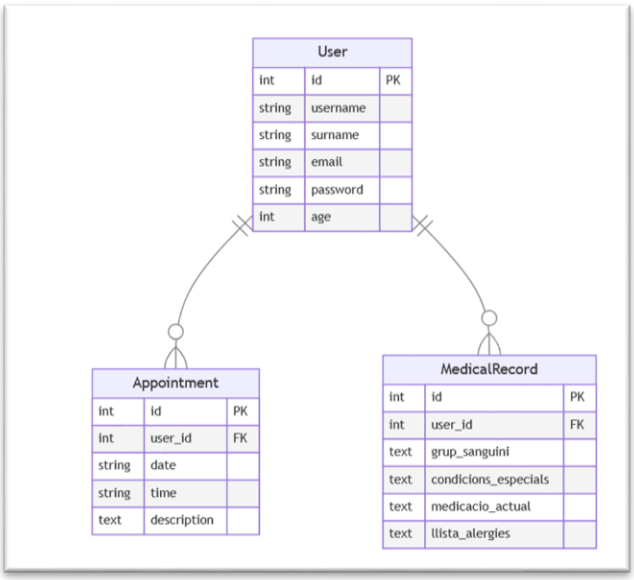

Diagrama ER de l’aplicació

La configuració inicial de l'aplicació ha permès generar automàticament aquestes taules amb les restriccions definides. Aquesta metodologia garanteix que la integritat de les dades sigui respectada des del primer moment, evitant conflictes i assegurant registres consistents.

Implementació del frontend de la web

El desenvolupament del frontend de SeniorLife ha estat centrat en oferir una experiència d'usuari clara i intuïtiva, amb un disseny adaptatiu gràcies a l'ús d'HTML, CSS i un estil responsiu.

1. *Estructura de la interfície:*
    - S'han creat diferents vistes per a les funcionalitats clau del sistema, com ara la pàgina de registre, inici de sessió, selecció d'opcions, gestió de cites, i visualització de registres mèdics.
    - Totes les pàgines inclouen un **header comú** amb el logo de SeniorLife, missatges personalitzats de benvinguda per a l'usuari, i en alguns casos icones associades al perfil.
2. *Pàgines clau i la seva funció:*
    - Home: Serveix com a porta d'entrada al sistema amb opcions per iniciar sessió o registrar-se, i un missatge benvingut per als nous usuaris.
    - Login i Registre: Implementades amb formularis clars que validen dades bàsiques com correu electrònic i contrasenya per assegurar la integritat dels comptes. Els estils garanteixen que es mantingui la coherència visual del sistema.
    - Gestió de cites: Un formulari interactiu permet seleccionar data i hora, i afegir una descripció de la cita, assegurant que els inputs de l'usuari siguin fàcils de comprendre i gestionar.
    - Registres mèdics: Una pàgina amb funcionalitats per afegir al·lèrgies, grup sanguini, condicions especials i medicació actual, amb opcions visuals per agregar elements amb icones temàtiques i llistes interactives.
    - Carrega de documents: Permet als usuaris carregar el seu historial mèdic en diversos formats (PDF, DOC, etc.), amb retroalimentació visual immediata sobre l'estat del fitxer seleccionat.
    - Benvinguda i alertes: Mostra notificacions personalitzades sobre cites programades i altres alertes amb una visualització neta i organitzada.

3. *Estilització i coherència:*
    - Per a cada pàgina, s'ha utilitzat una combinació de fitxers CSS globals i específics (per exemple, appointments.css per a la gestió de cites).
    - S'ha prioritzat una paleta de colors suau i icones visuals que transmeten confiança i professionalitat, especialment dissenyades per l'usuari final (persones grans i els seus familiars).

4. *Interactivitat:*
    - L'ús de JavaScript permet una millor experiència d'usuari en funcionalitats com l'afegit d'ítems en llistes o la gestió del formulari de càrrega de documents.

- Els missatges flash (get_flashed_messages) integrats ajuden als usuaris a comprendre els errors o confirmacions d'acció.

Aquest treball de frontend assegura una connexió clara i sòlida entre l'usuari i les funcionalitats del sistema desenvolupades al backend, oferint una experiència que combina estètica i funcionalitat.

3.3. Proves i validació

Per garantir la qualitat del sistema, s'han realitzat diverses proves utilitzant una base de dades temporal que permet validar les funcionalitats sense afectar les dades reals. Aquestes proves han inclòs situacions com el registre d'usuaris, la creació de cites i la gestió de registres mèdics.

Un dels aspectes clau ha estat validar que els correus electrònics siguin únics, evitant que dos usuaris comparteixin el mateix correu. També s'ha verificat que només es permetin registres per a usuaris amb edats compreses entre 18 i 120 anys, assegurant que les dades són coherents amb els requeriments establerts. A més, s'ha confirmat que les cites no es poden duplicar per a una mateixa data i hora, mantenint així la integritat del sistema.

Aquestes proves han demostrat que l'aplicació és robusta i que les funcionalitats implementades compleixen amb els requisits del projecte. Si volguéssim afegir més funcionalitats a l'aplicació, també podríem desenvolupar més el test per tal de testejar-les.

4. Aprovació del document

Aquest document de Software Requirements Specification (SRS) ha estat revisat i aprovat per les parts implicades en el desenvolupament del sistema. En aprovar aquest document, totes les parts interessades accepten els requisits especificats i es comprometen a seguir els detalls establerts en les següents fases del projecte.

4.1. Aprovacions

A continuació, es detallen les signatures dels responsables del projecte i altres stakeholders clau (Product Owner, CTO, experts en el domini i CFO) que han revisat i aprovat aquest document:

Nom	Càrrec	Data d'aprovació	Firma
Carla Qurban	Emprenedora i directora executiva	22/11/24	
Alejandra Popa	Enginyera en Informàtica i CTO	22/11/24	
Pascual Peña	Metge i Amic de Carla	22/11/24	
Eduardo Gasch	Cosí de Carla i CFO	22/11/24	

4.2. Comentaris addicionals

Aquest document serà revisat periòdicament a mesura que avancin les fases de disseny, desenvolupament i proves, i qualsevol canvi significatiu en els requisits serà documentat i aprovat en una versió actualitzada del mateix.