# Desenvolupament i adaptació d'un mòdul mèdic ERP

## basat en software lliure per entorns geriàtrics

**Autor:** David Badosa Gonzalez

**Data:** 20/04/2015

**Director:** Cristian Barrue- KEMLG

**Ponent:** Ulises Cortés García- Ciències de la Computació

**Titulació:** Grau en Enginyeria Informàtica

**Especialitat:** Sistemes d'informació.

**Centre:** Facultat d'informàtica de Barcelona (FIB)

**Universitat:** Universitat Politècnica de Catalunya (UPC)-BarcelonaTech

# INDEX

**1. Introducció i estat de l'art.....5**
1.1. Que és KEMLg?.....5
1.2. Que necessiten?.....6
1.3. ERP de software lliure.....9
1.3.1. Compiere.....9
1.3.2 WebERP.....10
1.3.3. OFBiz.....11
1.3.4. TinyERP.....12
1.4. Eleccio de l'Erp.....17
1.4.1.GNU Health.....18
**2. Abast del projecte.....19**
2.1. Obstacles del projecte.....19
2.2. Metodologia.....20
2.3. Eines de seguiment.....21
**3. Planificació temporal.....22**
3.1. Fases del projecte.....22
3.1.1. Fase inicial.....22
3.1.2. Fase de desenvolupament.....24
3.1.3. Proves i instal·lació.....26
3.2. Recursos associats a cada fase del projecte.....27
3.2.1. Fase inicial.....27
3.2.2. Fase de desenvolupament.....27
3.3. Diagrames de Gantt.....29
**4. Gestió Econòmica.....30**
4.1. Costos de recursos materials.....33
4.2.Costos directes.....34
4.2.1. Equip personal.....34
4.2.2. Eines de desenvolupament i de desplegament.....34
4.3. Costos indirectes.....35
4.3.1. Consum elèctric de l'equip personal.....35
4.3.2. Accés a Internet.....35
4.4. Cost total.....36
**5. Sostenibilitat i Compromís Social.....37**
5.1. Impacte econòmic.....37
5.2. Pla de control.....37

**6. Fase inicial.....38**
6.1 Anàlisis de Requisits.....38
6.1.1 Actors.....40
**7. Fase de desenvolupament.....42**
7.1. Aprenentatge de l'eina.....42
7.1.1 crear un nou registre.....44
7.1.2 Eliminar un prova.....45
7.1.2 Eliminar un Pacient.....45
7.1.3 Estructura interna.....47
7.2. Implementació.....49
7.2.1. Iteració I: Estructura del mòdul i «Functional Assessment».....49
7.2.2 Iteració II: Proves «Behavioral Assessment» i «Mobility».....62
7.2.3. Iteració III: Proves «Impact Of Technology Assessment».....67
7.2.4. Iteració IV: Proves «Neuropsychological».....69
7.3. Joc de proves.....77
**8. Conclusions.....78**
**9. Referències.....80**

# 1. INTRODUCCIÓ I ESTAT DE L'ART

## 1.1. QUE ÉS KEMLG?

*Imatge 1: Logotip KEMLg*

El **Knowledge Engineering and Machine Learning Group (KEMLG)**[1] és un grup d'investigació de l'enginyeria del coneixement i l'aprenentatge automàtic de la UPC[2].

Actualment està duent a terme diversos projectes però nosaltres ens fixarem en tots aquells que intenten millorar la qualitat de vida de les persones grans. Com per exemple:

-FATE (Fall Detector for the Elderly)

Projecte enfocat a la detecció ràpida i precisa que puguin patir les persones grans al seu domicili com a l'exterior.

**-**IDONTFALL

Aquest projecte té com a principal objectiu la detecció de caigudes i la seva prevenció de certs grups de persones grans amb unes característiques concretes.

**-**ASSAM

ASSAM és un projecte que també està enfocat en les persones grans a les quals se'ls intentarà millorar la seva qualitat de vida a causa d'impediments físics o cognitius mitjançant diferents plataformes de mobilitat.

## 1.2. QUE NECESSITEN?

En el moment de realitzar les proves dels seus projectes el KEMLG aconsegueix moltes dades dels resultats però no tens informació dels pacients ni de les seves patologies, ja que totes aquestes dades es troben emmagatzemades als sistemes d'informació dels seus respectius hospitals, als quals no s'hi pot tenir accés. Per aquest motiu el KEMLG necessita desenvolupar un ERP en el qual els metges puguin introduir tota la informació del pacient i el KEMLG pugui introduir totes les dades dels seus projectes. D'aquesta manera es podrà tenir emmagatzemada tota la informació necessària per poder extreure les conclusions oportunes i per poder millorar tots els projectes que s'estan duent a terme.

Els metges de tots aquests pacients hauran de respondre diversos qüestionaris sobre els seus pacients. Amb aquesta informació podrem definir el perfil de cada un dels pacients, que ajudarà a definir el perfil.

A continuació tenim detallat breument quins seran aquests qüestionaris:

1-Mobility

1.1- Tinnetti Performance Oriented Mobility (POMA) (Tinneti, 1986)

1.2 – Fall Efficacy Scale (FES) (Tinneti, 1990)

1.3- Six Minutes Walking Test (Balke, 1963)

1.4- Ten Meters Test (Van Hadel et al., 2005)

2-Neuropsychological

2.1- Global Cognition

2.1.1- Mini Mental State Examination (MMSE) (Folstein et al., 1975)

2.2- Frontal Abilities

2.2.1- Trial Making Test (TMT) (Reitan, 1958)

2.2.2- Clock Drawing Test (CDT) (Freedman et al., 1994)

2.2.3- Verbal Fluency (Borkowisky et al., 1967)

2.2.4- Color Progressive Matrices (CPM47) (Raven, 1947)

2.3- Memory

2.3.1- Rey Auditory Verbal Learning Test (RAVLT, 1958)

2.3.2- Rey Complex Figure Test – recall (Rey, 1941)

2.3.3- Digit Span forward and backward (Wechsler, 1981)

2.3.4- Corsi Span forward and backward (Milner, 1971)

2.4- Language

2.4.1- Boston Naming Test (Kaplan et al., 1983)

2.5- Praxis

2.5.1- Rey Complex Figure Test - copy (Rey, 1941)

3- Behavioral Assessment

3.1- Anxiety

3.1.1- State Trait Anxiety Inventory – Y (STAI-Y) (Spielberger et al., 1983)

3.2- Depression

3.2.1- Geriatric Depression Scale (Yesavege et al. 1982)

4- Functional Assessment

4.1- Instrumental Activities of Daily Living (IADL) (Lawton and Brody, 1969)

4.2- Barthel Index (Mahoney and Barthel, 1965)

5-Impact Of Technology Assessment

5.1- Quebec User Evaluation of Satisfaction with Assistive Tecnology (Demers et al., 2000)

5.2- Psychological Impact of Assistive Device Scale (PIADS)

5.3- The World Health Organization Quality of Life (S.M. Skevington et al., 2004)

## 1.3. ERP DE *SOFTWARE* LLIURE

Com a única condició per desenvolupar aquest projecte el **KEMLG** ens va demanar que utilitzéssim *software* lliure. Per tant, el primer que es va dur a terme va ser un petit estudi de mercat sobre quines eren les diferents possibilitats que teníem.

A grans trets, els ERP de *software* lliure es diferencien en quatre famílies: Compiere, TinyERP, WebERP, OFBIZ.

### 1.3.1. COMPIERE

Compiere és una eina desenvolupada amb **J2EE**. Per les bases de dades es pot treballar amb **PostgreSQL**, **MySQL** però **Oracle** continua sent la seva base de dades oficial. Tota aquesta eina es troba basada en la llicència pública de Mozilla.

Un dels ERPs més coneguts desenvolupats amb Compiere és **OpenBravo.**

#### OPENBRAVO

*Imatge 2: Logotip Openbravo*

Openbravo[4] és una aplicació web inicialment desenvolupada per dos professors de la Universitat de Navarra l'any 1990. Implementada amb Java i es basa en l'arquitectura client servidor. Inicialment va ser un *fork* de Compiere. Al estar desenvolupada amb Compiere per la seva base de dades suporta bases de dades en PostgreSQL i Oracle. Es troba traduït en 6 idiomes entre els quals trobem el castellà.

### 1.3.2 WEBERP

La segona família que trobem són la dels ERP desenvolupats amb WebERP. Aquests ERP es troben desenvolupats amb PHP i Javascript. La seva base de dades funciona gràcies a MySQL.

Com a principal representant d'aquesta categoria trobem FrontAccountingERP.

#### FRONTACCOUNTINGERP

*Imatge 3: Logotip FrontAccounting*

Tal com hem dit FrontAccountingERP[5] és un sistema de codi obert basat en WebERP (desenvolupat amb PHP i MySQL per a la base de dades) i enfocat per a PIMES.

A l'any 2012 ja tenia més de 100.000 descàrregues de SourceForge i al següent any ja va aparèixer a la llista dels *Bossie Awards* com a un dels millors ERP de codi obert.

### 1.3.3. OFBIZ

OFBiz[4] és un conjunt d'aplicacions empresarials desenvolupat per la fundació Apache. OFBiz utilitza diversos llenguatges de programació com són J2EE, Java, SOAP i XML. Com a màxim representant d'aquesta família de ERPs trobem Opentaps.

#### OPENTAPS

*Imatge 4: Logotip OpenTaps*

Opentaps[5] és un ERP de la procedent d'un fork de OFBiz desenvolupat amb java. Actualment es troba a la versió 1.5 i té llicencies AGPL. Les seves bases de dades funcionen gràcies a MySQL i PostgreSQL. És un ERP poc implantat i no té tanta notorietat com altres ERPs de Software lliure.

### 1.3.4. TINYERP

Per acabar tenim una de les famílies de ERP de *software* lliure més importants que hi ha actualment. Aquesta plataforma es troba realitzada amb Python i les seves bases de dades es troben administrades amb PostgreSQL. Trobem dos ERPs desenvolupats a partir de TinyERP: Tryton i Odoo.

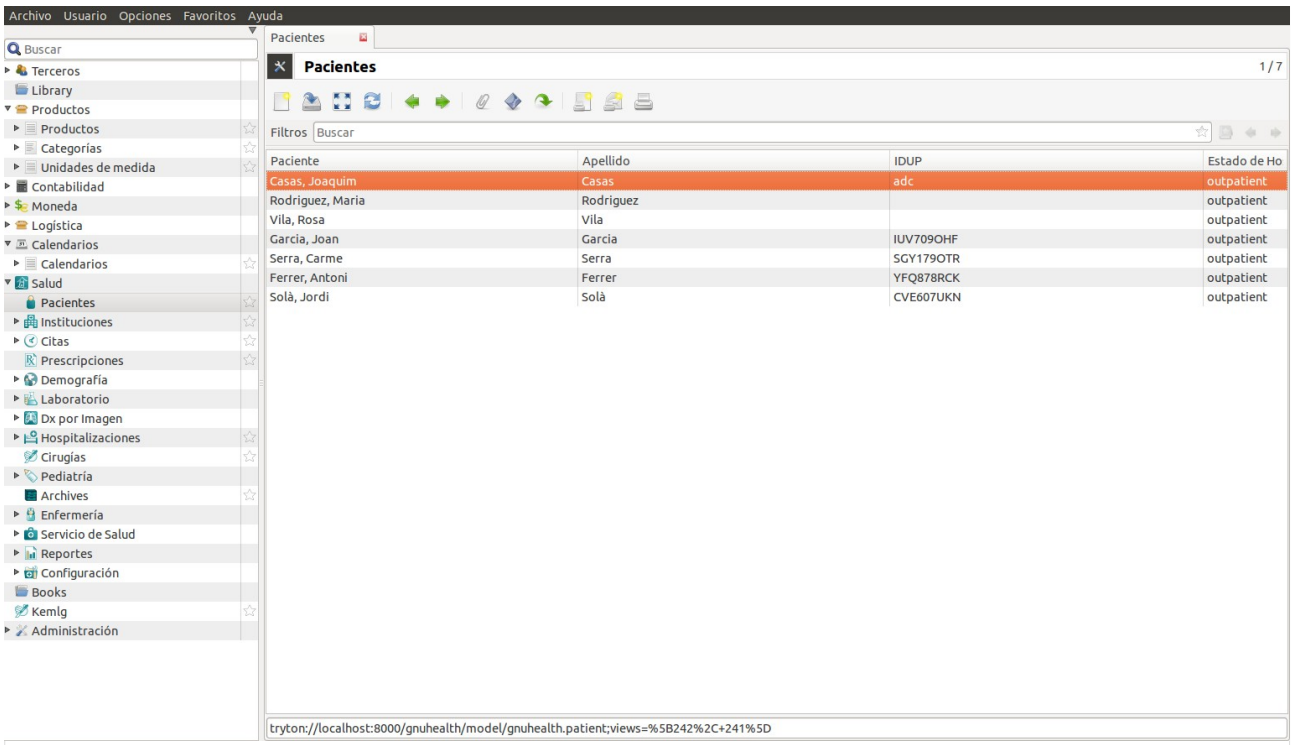
#### TRYTON

*Imatge 5: Logotip Tryton*

Tryton[6] es caracteritza per ser una plataforma de tres capes sobre la qual es desenvolupa una solució ERP. Tant Tryton com tots els seus mòduls oficials es troben sota la llicencia GPLv3.

Inicialment Tryton es va desenvolupar a partir de la versió 4.2 de TinyERP (actualment conegut com a Odoo) però a la primera versió publica presentada al novembre del 2008 ja s'havien reescrit diverses parts del nucli i dels mòduls principals.

Actualment Tryton es troba gestionat per una fundació, Tryton Foundation Privée amb seu a Bèlgica. Aquesta fundació privada s'encarrega d'ajudar en el desenvolupament de Tryton, donar suport mitjançant esdeveniments i activitats, gestionar la infraestructura de Tryton.org, organitzar la comunitat i la gestió i promoció de la marca Tryton.




*Imatge 6: Vista principal Tryton*

En el moment que accedim a Tryton se'ns obre la vista principal. Ens trobem amb un menú principal a l'esquerra de la pantalla en el que hi ha els accessos a tots els mòduls que tenim instal·lats en el nostre ERP Tryton. De tota manera, dintre del submenú d'administració tenim l'opció d'instal·lar diversos mòduls.

## ODOO

*Imatge 7: Logotip Odoo*

Com a últim ERP hem deixat el més important, Odoo[6] (anteriorment conegut com a OpenERP). Odoo és la continuació natural de TinyERP i es defineix com a la millor alternativa a SAP dintre del codi obert. També sent uns dels ERP amb un codi més senzill i amb més de dos milions d'usuaris actualment. Entre aquests usuaris podem trobar diverses empreses de renom com són: Danone, La Poste (Correus de França), Auchan (conegut a Espanya com Alcampo), WWF, Singer, Canonical, Exki, entre d'altres.

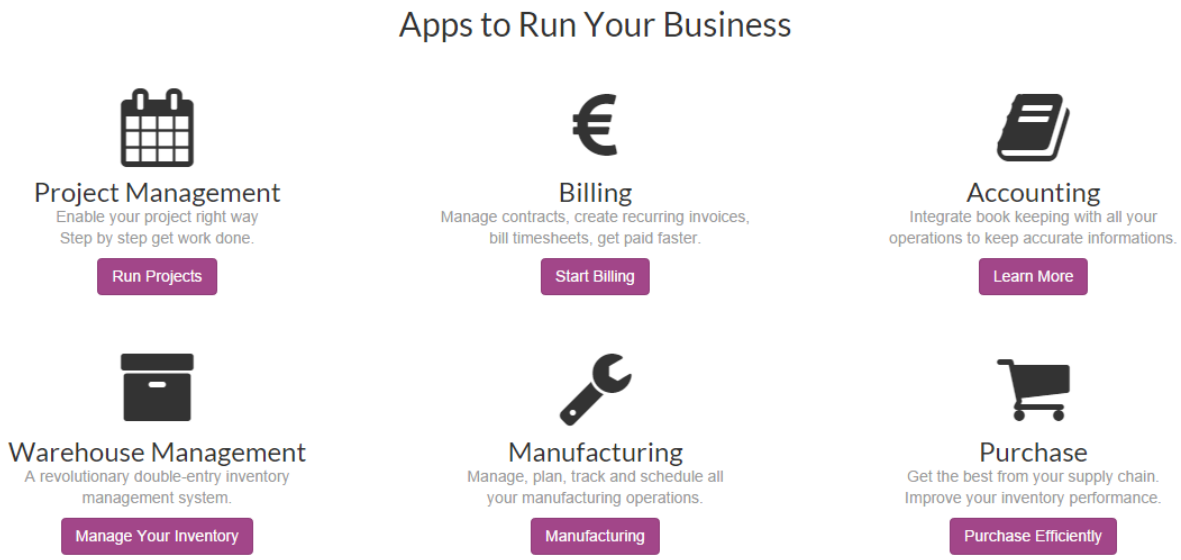
*Imatge 8: Logotip Danone*

*Imatge 9: Logotip La Poste*

*Imatge 10: Logotip Auchan*

Tot el codi de Odoo té llicència AGPL i per tant no es necessita pagar cap tipus de taxa per utilitzar-lo. Aquesta llicència també ens permet modificar el programa al nostre gust.

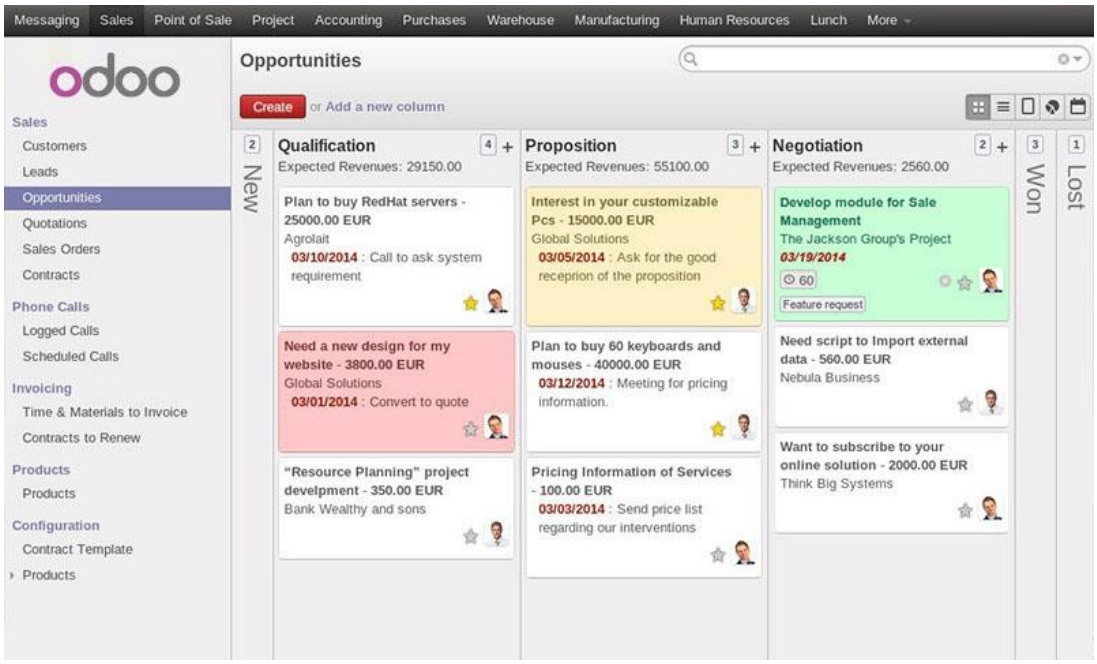
Com a novetat de l'última versió de Odoo ens trobem que hi ha una aplicació web que ens permet executar el programa des de qualsevol navegador. En canvi, l'aplicació d'escriptori ha quedat obsoleta des de la versió 7 (actualment es troben a la versió 8[7]).




*Imatge 11: Aplicacions Odoo*

Odoo es caracteritza per ser una de les eines desenvolupades amb codi obert amb més mòduls. A través de la seva web podem instal·lar una gran varietat de mòduls que afegiran totes les funcionalitats que necessitem al nostre ERP. En la següent imatge podem veure una breu representació d'alguns dels mòduls que poden instal·lar des de la seva pagina web.

El seu funcionament de cara l'usuari final és força intuïtiu, ja que l'aplicació es troba desenvolupada amb una interfície molt senzilla on disposem de dos menús una a la part superior i un altre a la part esquerra que ens permet accedir a totes les característiques i mòduls que tenim instal·lats al nostre sistema. En la següent imatge podem veure un exemple d'aquesta pantalla.




*Imatge 12: Menu pricipal Odoo*

## 1.4. ELECCIO DE L'ERP

Pel volum d’usuaris i la gran quantitat de mòduls disponibles ens hauríem decantat per Odoo per desenvolupar el nostre ERP. En primer moment **OEMedical**[8] també semblava un bon mòdul a partir del qual desenvolupar el nostre mòdul però després comprovar el funcionament de OEMedical ens vam adonar que no ens proporcionava una bona base per nosaltres, ja que només ens donava un esquelet de com hauria d'estar organitzat un mòdul enfocat a empreses de l'àmbit sanitari, però a part d'això no ens proporcionava cap tipus de base de dades en el qual hi hagués emmagatzemada tota la informació relacionada amb les malalties o com administrar els pacients.

Per tant, tot i tenir una comunitat d'usuaris menor a la de Odoo ens hem decantat per Tryton per desenvolupar el nostre mòdul, ja que en ell hi trobem un mòdul anomenat GNU Health. Aquest mòdul és molt més complet que OEMedical.

Pel que fa a la programació tant Tryton com Odoo provenen de TinyERP i encara que després de fer el fork Tryton va modificar el seu nucli encara conserven moltes parts en comú. Tots dos en trobem implementats amb Python i XML.

### 1.4.1.GNU HEALTH

*Imatge 13: Logotip GNU Health*

Tal com ens diuen a la seva web «Gnu Health és un sistema lliure de gestió hospitalària i d'informació de la salut». Aquest sistema es troba desenvolupat a Espanya, concretament per Luis Falcón. Aquest projecte es va iniciar a finals de l'any 2008 sent un mòdul de OpenERP però l'any 2011 va canviar OpenERP per Tryton. Actualment es troba en la versió 2.6.

Aquest projecte va néixer amb l'objectiu de proporcionar un ERP per àrees rurals però actualment GNU Health forma part del projecte GNU Solidari. El projecte GNU Solidari és un projecte que proporciona salut i educació mitjanant software lliure.

Aquest mòdul desenvolupat per Tryton és un dels mòduls més potents que podem trobar actualment de forma gratuïta. GNU Health no és únicament una estructura que hàgem d'omplir amb informació, sinó que en ell ja hi trobem una gran base de dades sobre medicaments, malalties, etc. Per tant aquest mòdul pot començar a ser utilitzat per qualsevol centre sanitari sense haver de fer gaires modificacions.

Per nosaltres aquest mòdul ens proporciona una bona base per desenvolupar el nostre mòdul, ja que en podrem aprofitar diverses parts com podria ser tota la fitxa clínica dels pacients.

L'únic inconvenient que podríem trobar i que ens limitarà és la personalització d'aquest mòdul. Tryton no permet un nivell de personalització molt elevat i per tant hem de seguir amb l'estructura que ja ens ve definida per no causar cap tipus de molèstia a l'usuari final.

# 2. ABAST DEL PROJECTE

Després d’analitzar les necessitats del grup d’investigació hem determinat que el nostre projecte es basarà en el desenvolupament d’un ERP gràcies al qual el KEMLG tindrà tota la informació necessària de tots els pacients que participen en els seus projectes. Aquesta informació serà introduïda al sistema pels metges d’aquests pacients. Gràcies a tota aquesta informació el grup d’investigació tindrà suficient informació per poder desenvolupar de manera correcta els seus projectes, ja que podrà veure quina és l'evolució d'aquests pacients després d'haver començat a formar part del mateix.

Gràcies a tota aquesta informació també es podrà fer un perfil mitjançant el qual es podrà definir el millor públic objectiu del projecte, aquell públic que aconseguirà millor sensiblement la seva qualitat de vida.

## 2.1. OBSTACLES DEL PROJECTE

Un dels obstacles més importants que podem tenir en el moment de desenvolupar el nostre sistema d'informació és la utilització d'un ERP que no es permet una gran personalització, sinó que haurem de mantenir l'estructura principal que ens imposa Tryton. Aquest fet el podem considerar un obstacle perquè hem de seguir les pautes que ja ens venen marcades però també podem considerar-ho com un avantatge de cara a l'usuari perquè no el forcem a aprendre el funcionament del nostre mòdul de zero sinó que podrà seguir utilitzant el mòdul tal com utilitzava GNU Health.

D'altre banda, tenim un obstacle de caràcter legal, perquè estarem treballant amb dades molt sensibles de cara a l'usuari. Per aquest motiu, haurem d'informar a l'usuari en tot moment que les seves dades estaran emparades sota la llei de protecció de dades que es troba vigent actualment. El nostre projecte necessita la informació de molts pacients per poder treure el màxim profit al nostre mòdul. Sense la informació dels pacients aquest mòdul no tindrà el correcte funcionament.

## 2.2. METODOLOGIA

El mètode de treball seleccionat per desenvolupar aquest projecte es basarà realitzar cicles curts on a cada cicle es vagin afegint funcionalitats addicionals al nostre mòdul ERP. Després de la finalització de cada cicle s’utilitzarà un petit joc de proves per comprovar que cada una de les diferents funcionalitats afegides funciona i no entra en conflicte en cap de les funcionalitats implementades anteriors.

En el moment que totes les funcionalitats estiguin acabades i funcionin correctament passarem a realitzar una comprovació amb una gran quantitat de dades per assegurar que el mòdul funciona correctament amb una càrrega de treball elevada.

Per realitzar aquests jocs de proves utilitzarem dades de pacients i anirem introduint resultats a les diferents proves que formen el nostre mòdul. Crearem, modificarem, eliminarem i gestionarem dades de diferents pacients per poder comprovar que totes les funcionalitats funcionen correctament i no hi ha cap tipus d’error.

## 2.3. EINES DE SEGUIMENT

Per poder  tenir ben estructurat en el temps el nostre projecte utilitzarem dues eines que ens permetran controlar si estem anant en bon camí per poder finalitzar el nostre projecte a temps.

L'eina bàsica que utilitzarem serà Google Calendar per marcar tots els terminis i reunions del nostre projecte. D'altra banda també disposem d'una eina que afegida al Google Drive ens permetrà crear diagrames de Gantt el que ens donarà d'una manera molt visual quin és el temps que hem de dedicar a cada una de les parts del nostre projecte per poder arribar a bon port.

# 3. PLANIFICACIÓ TEMPORAL

## 3.1. FASES DEL PROJECTE

Per intentar planificar correctament aquest projecte l'hem dividit en tres grans parts: Fase inicial, fase d'implementació i una fase de proves i instal·lació

### 3.1.1. FASE INICIAL

Aquesta fase es caracteritza per entrar en contacte amb el món relacionat en el qual treballa el departament **KEMLg** de la **UPC** per esbrinar totes les funcionalitats que esperen tenir un cop desenvolupat aquest sistema d'informació perquè els ajudi a desenvolupar correctament tots els seus projectes. Per tant, ens hauran d'informar sobre tots els paràmetres que necessiten saber sobre tots els pacients que formen part dels seus projectes i d'altra banda saber que és el que els metges necessiten veure dels seus pacients per saber si hi ha hagut una correcta evolució després d'haver començat a formar part del projecte.

En aquesta fase també tindrem en compte tots els costos que es despendran arran del desenvolupament del projecte, tant siguin materials o humans. S'analitzaran tots els riscos que pot impedir-nos realitzar el projecte dins dels terminis que establirem per poder tenir preparades alternatives per si es presenta algun imprevist.

Per conèixer la situació actual dels ERP de codi obert realitzarem un petit estudi de mercat en el qual comprovarem les funcionalitats dels ERP més importants i quines són les funcionalitats enfocades al món de la sanitat, quin és l'ERP més utilitzat i quin és el que té una major comunitat oferint suport perquè ens puguin ajudar en el cas de patir cap problema en el seu desenvolupament.

En aquesta part agafarem tota la informació que ens ha proporcionat el grup d'investigació i començarem a plantejar mitjançant, mapes conceptuals quina serà l'estructura del nostre programa.

També haurem de tenir en compte quines són les dades que els metges volen emmagatzemar dels seus pacients després d'haver realitzat els diferents tipus de proves als que han estat sotmesos.

Aquesta fase ens ajudarà en altres fases del projecte a treballar d'una manera organitzada i sense oblidar-nos de cap funcionalitat.

### 3.1.2. FASE DE DESENVOLUPAMENT

Aquesta és la més important del projecte, per aquest motiu la trobem subdividida en diversos apartats:

#### APRENENTATGE DE L'EINA

La primera tasca que haurem de realitzar un cop seleccionat l'ERP és estudiar en quin llenguatge està desenvolupat, quins mòduls podem adaptar pel nostre ERP i finalment, instal·lació de totes les eines necessàries per poder implementar el nostre projecte.

Una de les parts en els que hi haurem de dedicar més temps és en l'estudi de l'arquitectura perquè tot i utilitzar els mateixos llenguatges cada ERP utilitza una estructura interna clarament diferenciada. Per això abans de començar a desenvolupar el nostre mòdul hem de tenir clar quin tipus d'organització interna en fa falta per poder construir el nostre mòdul.

#### IMPLEMENTACIÓ

Després de la fase d'aprenentatge ja tindrem instal·lat l'ERP seleccionat amb totes les eines necessàries.

A partir d'aquests moments començarem a desenvolupar l'ERP en si. Amb tota la informació de la qual disposem fins al moment dividirem aquesta fase en quatre

subapartats, per tal de poder fer unes iteracions més curtes i comprovar si després de cada iteració el programa continua funcionant de manera correcta i assegurar-nos que cap funcionalitat nova entra en conflicte amb alguna funcionalitat implementada anteriorment.

## JOC DE PROVES

Per comprovar aquestes funcionalitats disposarem d'un breu joc de proves en el qual trobarem tots els casos que nosaltres considerem que poden ser conflictius per assegurar-nos que el nostre ERP funciona correctament. Aquests jocs de proves estaran formats per diferents pacients als quals els associarem amb cada una de les noves proves implementades en la nova iteració per assegurar-nos que no hi ha cap error de programació i que tot funciona correctament.

### 3.1.3. PROVES I INSTAL·LACIÓ

Un cop implementat tot el nostre projecte passarem a redactar tota la documentació necessària perquè qualsevol usuari pugui treballar amb l'ERP resultant. En aquesta documentació i vindrà detallada totes les funcionalitats i tots els procediments per utilitzar correctament el programa.

D'altra banda també es realitzarà una prova exhaustiva de totes les funcionalitats del sistema que  tot punciona correctament i no tenim cap mena d'error.

Per acabar aquesta fase, i tot el projecte, es procedirà a instal·lar tot el sistema ERP en els sistemes corresponent i la seva posada en marxa.

## 3.2. RECURSOS ASSOCIATS A CADA FASE DEL PROJECTE

### 3.2.1. FASE INICIAL

La fase inicial es basa en aconseguir la major quantitat possible d'informació per poder plantejar correctament el projecte. A primer cop d'ull podem veure com els recursos que necessitarem per aquesta fase són els diversos qüestionaris que utilitza el grup d'investigació per classificar cada un dels pacients que té a la base de dades. Aquests tests ens mostren diferents característiques d'aquestes persones incloent-hi dades sobre el seu estat de salut. També hem de saber quina és la visualització que prefereixen tant els metges com el KEMLg.

### 3.2.2. FASE DE DESENVOLUPAMENT

Per poder desenvolupar el projecte necessitem a tres recursos:

#### TRYTON

El més important és tenir el codi de l'ERP seleccionat, en aquest cas Tryton, així com tots els programes que Tryton pugui necessitar pel seu correcte funcionament, com poden ser els programes necessaris perquè el nostre sistema reconegui el llenguatge de Python o de SQL per poder guardar les dades.

#### GNU HEALTH

També hem de tenir el codi del mòdul en el qual basarem el nostre sistema d'informació per poder començar a treballar a partir d'ell i utilitzar moltes de les funcionalitats de les quals disposa.

### SERVIDORS DE UPC

En aquesta fase necessitem tenir accés al servidor de la UPC on s’emmagatzemarà el programa per poder provar el seu correcte funcionament.

## 3.3. DIAGRAMES DE GANTT

Per portar un control sobre el temps que hem de dedicar a desenvolupar cada part del projecte hem realitzat un diagrama de Gantt. En aquest diagrama hi hem detallat les parts més importants que creiem que hi hem de tenir especial atenció.

Com podem veure hi ha diferents fases del projecte que se'ns superposen però no suposa cap problema, ja que no entren en conflicte entre elles.

| | | Nombre | Duración | Inicio | Fin |
|---|---|---|---|---|---|
| 1 | | Fase inicial | 20d | 19/09/2014 | 16/10/2014 |
| 2 | | Fase de Desenvolupament: Aprenentatge de l'eina | 10d | 17/10/2014 | 30/10/2014 |
| 3 | | Fase de Desenvolupament: Analisis de requisits | 20d | 31/10/2014 | 27/11/2014 |
| 4 | | Fase de Desenvolupament: Implementació (1a iteració) | 20d | 28/11/2014 | 25/12/2014 |
| 5 | | Fase de Desenvolupament: Implementació (2a iteració) | 20d | 26/12/2014 | 22/01/2015 |
| 6 | | Fase de Desenvolupament: Implementació (3a iteració) | 20d | 23/01/2015 | 19/02/2015 |
| 7 | | Fase de Desenvolupament: Implementació (4a iteració) | 25d | 20/02/2015 | 26/03/2015 |
| 8 | | Fase de Documentació | 9d | 27/03/2015 | 08/04/2015 |
| 9 | | Presentació | 8d | 09/04/2015 | 20/04/2015 |

*Imatge 14: Diagrama de Gantt*

En el moment que la gestió del projecte està arribant al seu final ens trobem que ja tenim tota la informació necessària per començar l'especificació del projecte.

De la mateixa manera ens trobem que en el moment que estem realitzant la tercera iteració de la implementació ja podem començar a fer diversos jocs de proves de la primera i segona iteració d'aquesta manera aconseguim reduir el temps total a desenvolupar el projecte.

# 4. GESTIÓ ECONÒMICA

El nostre projecte es caracteritza perquè no necessitem una quantitat molt elevada de recursos materials addicionals, ja que la majoria del software que utilitzarem és *Open Source* i per tant no haurem de pagar cap tipus de llicència pel seu ús.

Per tant, les despeses més importants del nostre projecte se centraran en els recursos humans. A continuació tenim una taula que ens mostra el salari mitjà de cada un dels professionals que han de formar part del projecte. Aquests preus són els preus de facturació que se li cobrarien al client final.

| Rol | Salari |
|---|---|
| Cap de Projecte | 50€/h |
| Analista i Dissenyador | 50€/h |
| Programador i Tester | 30€/h |

**Taula 1: Salari treballadors**

A partir d'aquesta taula realitzarem un pressupost en el qual hi hauran detallades totes les fases del diagrama de Gantt amb tots els professionals que es necessiten per a cada una d'elles amb les seves corresponents hores i dies. Cal tenir en compte que tots els preus fan referència a l'import que se li cobraria al client final al qual se li està desenvolupant el projecte.

| Nom de la tasca | Dies | Hores diàries | Total hores | Recurs | Cost |
|---|---|---|---|---|---|
| **Front-End** | | | | | |
| GEP | 33 | 2.5 | 82.5 | C | 4.125€ |
| Reunions inicials amb el client | 4 | 2 | 8 | C,A | 400€ |
| Reunions de control | 4 | 2 | 8 | C,A | 400€ |
| Anàlisi de requisits | 3 | 2 | 6 | A | 300€ |
| Redacció de la memòria | 10 | 5 | 50 | C | 2.500€ |
| Especificació | | | | | |
| Definició d'actors | 3 | 3 | 9 | A | 450€ |
| Disseny | | | | | |
| Disseny de l'arquitectura lògica | 2 | 4 | 8 | D | 400€ |
| Disseny de la base de dades | 2 | 4 | 8 | D | 400€ |
| Fases de disseny | 5 | 4 | 20 | D | 1.000€ |
| Disseny de l'arquitectura física | 2 | 4 | 8 | D | 400€ |
| **Implementació** | | | | | |
| Fases d'implementació | | | | | |
| 1a iteració | 20 | 4 | 80 | P | 2.400€ |
| 2a iteració | 20 | 4 | 80 | P | 2.400€ |

| | | | | | |
|---|---|---|---|---|---|
| 3a iteració | 20 | 4 | 80 | P | 2.400€ |
| 4ª iteració | 20 | 4 | 80 | P | 2.400€ |
| Adaptació als servidors UPC | 5 | 4 | 20 | P | 600€ |
| **Proves i instal·lació** | | | | | |
| Fases de proves | 5 | 5 | 25 | T | 750€ |
| Desplegament | 3 | 5 | 15 | P | 450€ |
| **Total** | | | | | **21.775€** |

**Taula 2: Cost humà del projecte**

## 4.1. COSTOS DE RECURSOS MATERIALS

Tal com hem comentat anteriorment per aquest projecte necessitem diversos recursos però com que tots ells són de codi obert no ens suposaran un increment de pressupost.

Tant Tryton com Gnu Health com PostgreSQL són programes dels quals podem aconseguir sense cap mena de problema una llicència gratuïta. A través d'internet per poder treballar, modificar i adaptar aquests programes a les nostres necessitats perquè ens ajudin a poder desenvolupar el nostre sistema d'informació.

D'altra banda, per poder instal·lar el nostre programa necessitem uns servidors però en aquest cas utilitzarem els servidors de la UPC que per tant tampoc suposaran un increment del pressupost, ja que el nostre client ja té accés a aquests servidors i podrem instal·lar el nostre programa sense cap mena de problema.

## 4.2.COSTOS DIRECTES

### 4.2.1. EQUIP PERSONAL

El nostre cost directe més important es troba relacionat amb l'equip personal que necessitem per poder dur a terme el nostre projecte. Hem adquirit un equip personal que ens ha suposat una despesa de 1.000 € i per tant calculant la seva amortització en aquest projecte ens suposarà un increment de 42,80 €.

### 4.2.2. EINES DE DESENVOLUPAMENT I DE DESPLEGAMENT

Com portem anomenant durant tot el projecte tot el *software* utilitzar serà Open *Source* i per tant no tindrem cap problema de llicències per treballar i modificar aquest software. Però també tindrem en compte que si per poder programar necessitarem algun editor especial, sempre ens decantarem per un que sigui lliure por no augmentar el nostre pressupost de forma innecessària.

## 4.3. COSTOS INDIRECTES

Com a costos indirectes tindrem els costos bàsics que tindria qualsevol oficina per poder desenvolupar aquest projecte, és a dir, necessitarem un accés a internet i poder connectar el nostre ordinador a corrent.

### 4.3.1. CONSUM ELÈCTRIC DE L'EQUIP PERSONAL

Podem considerar que un equip personal bàsic per realitzar les tasques necessàries pel correcte desenvolupament del projecte consumeix uns 286W, per tant, considerant la jornada de treball aproximada rondarà al voltant de les 5 hores diàries en un any ens suposaria una despesa de 70,60€. Tenint en compte el nombre d'hores que ens costarà desenvolupar aquest projecte poder concloure que tindrem una despesa aproximada de 12,09€

### 4.3.2. ACCÉS A INTERNET

Com a última despesa indirecta tenim l'accés a Internet indispensable per poder treballar amb tots els nostres programes. Agafant el preu mig d'accés a Internet al nostre país podem veure que se situa al voltant dels 45 €/mes. Calculant el total d'hores que dedicarem al projecte obtenim que la despesa en Internet per aquest nombre d'hores és de 92,47 €.

## 4.4. COST TOTAL

Després de calcular els costos humans i materials podem veure un pressupost dels que ens costarà econòmicament desenvolupar un projecte d'aquestes característiques.

| Recurs | Cost |
|---|---|
| Recursos humans | 21775€ |
| Recursos materials | 104.56€ |
| Total: | 21.879,56€ |

**Taula 3: Cost total del projecte**

A la taula (taula 3) podem veure com el cost serà d'uns 22.000 € aproximadament. En els apartats anteriors ja hem vist que en aquest projecte la majoria de les despeses en venien provinents de recursos humans mentre que els recursos materials no arriben a un 10% del preu final del projecte.

# 5. SOSTENIBILITAT I COMPROMÍS SOCIAL

## 5.1. IMPACTE ECONÒMIC

El nostre projecte tindrà un impacte econòmic en el desenvolupament de projectes del KEMLG, ja que els ajudarà a obtenir molta més informació sobre tots els pacients que formen part dels projectes. Amb tota aquesta informació podran escollir millor quin és el camí a seguir per poder els millors resultats possibles.

Un cop finalitzat el nostre projecte el seu manteniment serà mínim, ja que únicament s'haurà d'anar actualitzant, en el cas que sigui necessari, afegint noves funcionalitats o modificant funcionalitats ja existents.

## 5.2. PLA DE CONTROL

Amb tots els càlculs realitzats podem dir que aquest projecte es podrà realitzar dins dels terminis establerts de totes maneres s'ha deixat un marge d'una setmana entre la finalització del projecte i la seva presentació per tenir temps més que suficient per poder solucionar qualsevol imprevist que pugui aparèixer. D'altra banda també s'ha organitzat el projecte de manera que més d'una tasca es poden desenvolupar simultàniament sense que entrin en conflicte per intentar agilitzar al màxim la seva programació.

Per poder portar un bon control sobre el desenvolupament del projecte controlarem cada fase amb els diagrames de Gantt mostrats anteriorment.

# 6. FASE INICIAL

## 6.1 ANÀLISIS DE REQUISITS

El més important a l'hora de començar a desenvolupar el nostre sistema d'informació és analitzar a fons quines són les necessitats que tindran, tant els metges que utilitzaran el nostre sistema com el grup d'investigació de la UPC, KEMLg, per poder aprofitar al màxim aquest mòdul.

El nostre mòdul només tindrem informació relacionada amb els resultats dels testos als quals s'han sotmès els pacients, per aquest motiu necessitarem relacionar el nostre mòdul amb GNU Health perquè d'aquesta manera no només podrem introduir el resultat dels testos sinó que també podrem introduir a GNU Health més informació sobre quin és l'historial mèdic del pacient, a quines cirurgies s'ha sotmès, veure les seves radiografies, etc. Sempre tenien el permís del pacient per accedir a aquesta informació i respectant les lleis de protecció de dades que estan vigents actualment.

Pels dos sectors pels quals estem desenvolupant aquest projecte tenen en comú que volen veure quina és l'evolució dels pacients, per tant, no els interessa només quins han set els resultats dels seus pacients en les últimes proves que se'ls han dut a terme sinó que volen veure quina ha estat l'evolució d'aquests pacients al llarg del temps per poder veure si els tractaments als quals han sotmès als pacients han ajudat a millorar la seva qualitat de vida. Per tant, no podem realitzar un mòdul que només ens mostri el resultat de l'última prova realitzada sinó que hem d'emmagatzemar tot l'historial del pacient i que es pugui consultar en qualsevol moment perquè d'aquesta manera en podem seguir la seva evolució.

La majoria dels testos que formen part del nostre sistema es poden incloure al mòdul que desenvolupem amb un formulari en el qual hi hagi únicament *checkboxs* o llistes en les quals hem de seleccionar l'opció corresponent. Però, d'altra banda també tenim testos en els quals necessitem veure que és el que ha contestat el pacient. Per

aquest motiu també hem d'incloure en el nostre mòdul l'opció de poder adjuntar imatges a algun d'aquests testos per poder veure correctament la resposta del pacient i d'aquesta manera proporciona un diagnòstic encertat.

Per poder veure el resultat de la prova d'una manera rapida i intuïtiva la gran part d'aquestes proves ens proporcionen un resultat numèric en funció de les respostes del pacient a les seves preguntes. Per aquest motiu també inclourem al nostre mòdul una variable que ens calcularà en funció dels paràmetres predeterminats per cada una de les proves quin és el seu resultat.

Per intentar que la línia d'aprenentatge dels usuaris finals al nostre mòdul hem decidit mantenir l'organització i l'estructura que té tan Tryton com GNU Health. Per aquesta raó, utilitzarem les mateixes icones i les mateixes funcions. D'aquesta manera evitarem que l'usuari hagi d'aprendre a utilitzar en el nostre sistema, així li serà més fàcil introduir les dades al sistema i veure'n els resultats.

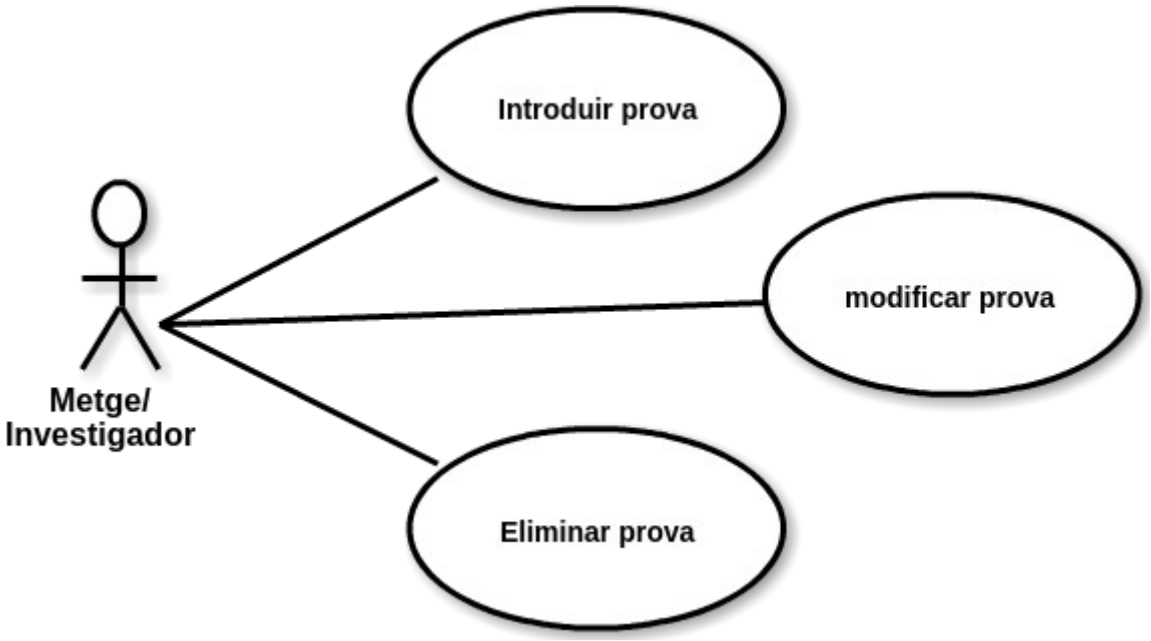
El nostre sistema d'informació ha de ser accessible ràpidament des de Tryton per aquest motiu hem decidit incloure el nostre mòdul al menú principal situat a l'esquerra de l'aplicació. Tot i que utilitzarem en gran part recursos dels proporcionats per GNU Health hem posicionat el nostre mòdul fora del seu menú principal.

### 6.1.1 ACTORS

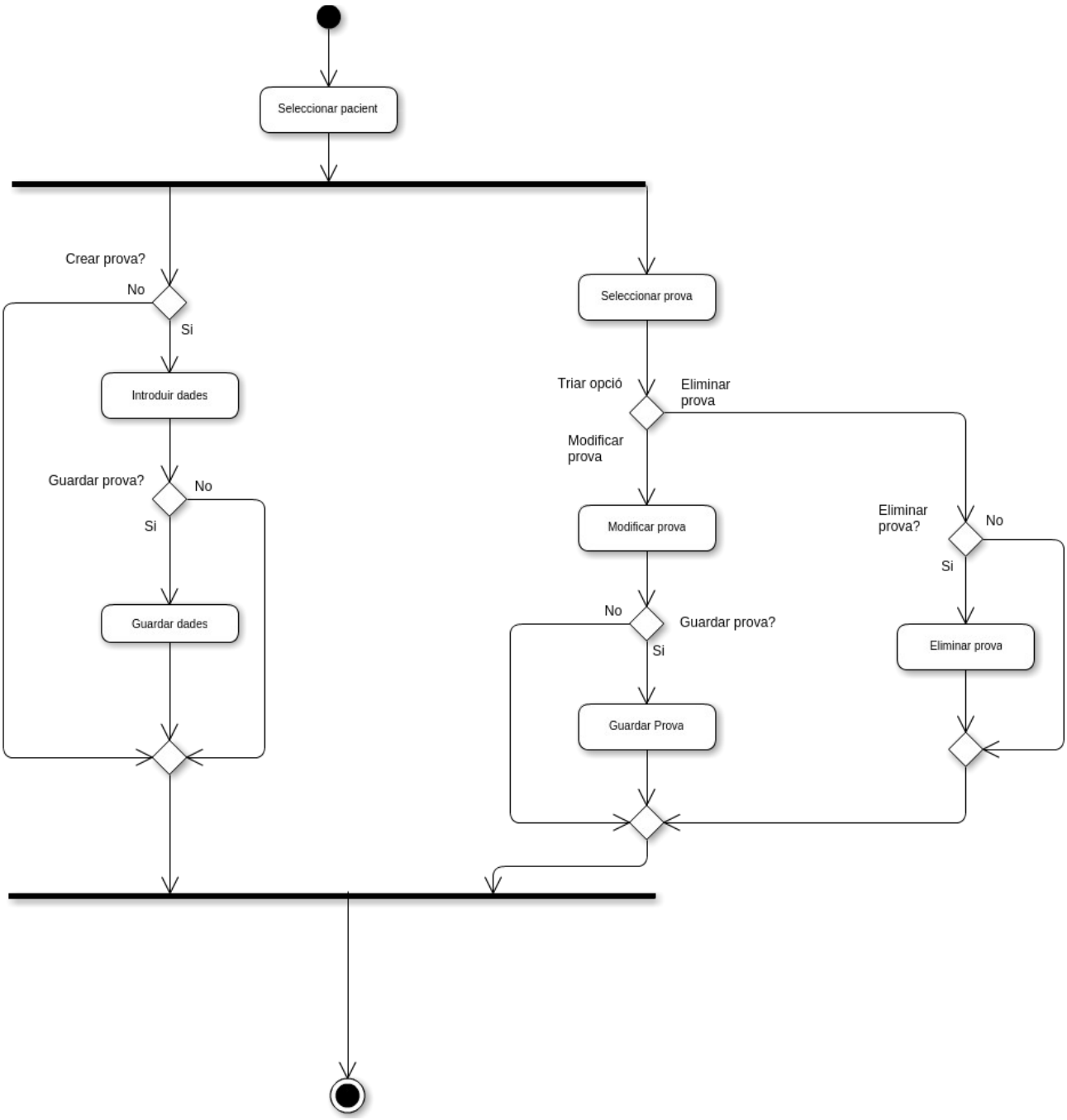
Aquest mòdul disposarà de dos actors principals. En primer lloc tindrem els metges que hauran d'introduir les dades de tots els pacients als quals els realitzin algunes de les proves de les quals tindrem al nostre mòdul. Els metges tindran l'opció de crear noves proves relacionant-les amb els diferents pacients que tindrem a la base de dades però també tindran l'opció d'afegir un nou pacient si aquest no es troba a la base de dades.

D'altra banda, també disposem del KEMLg, aquest grup d'investigació tindrà accés a veure totes les proves que han introduït els metges però també tindran la possibilitat de crear-ne de noves encara que la seva funció principal és observar l'evolució dels pacients.

En la imatge 15 podem veure els tres casos d'ús que hi haurà en el nostre sistema. Els dos actors principals podran realitzar les mateixes funcions.




*Imatge 15: Actor del sistema*




*Imatge 16: Diagrama d'activitats*

En la imatge 16 veiem un diagrama d'activitats on es veu breument com es desenvoluparien les tres accions principals que poden realitzar els actors al nostre sistema.

# 7. FASE DE DESENVOLUPAMENT

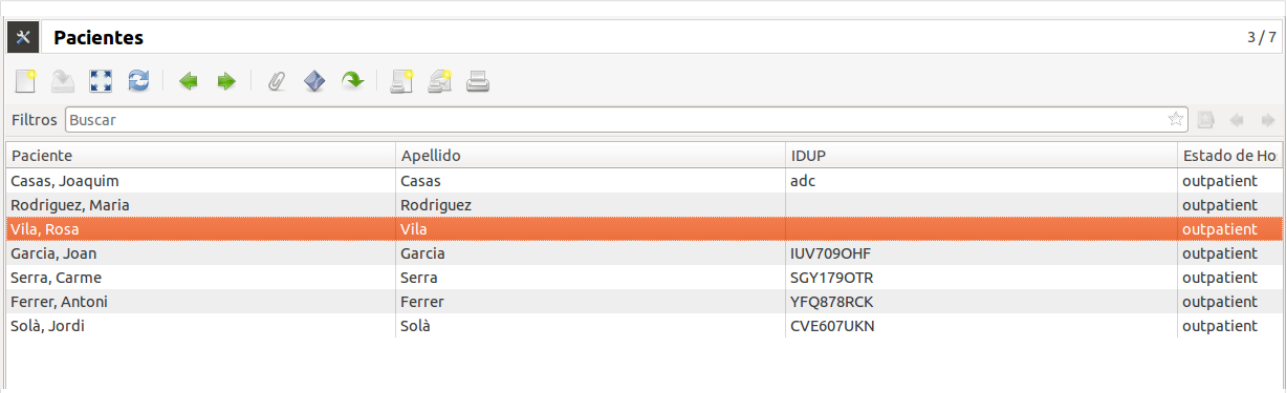
## 7.1. APRENENTATGE DE L'EINA

Per poder començar a desenvolupar el nostre projecte necessitem instal·lar al nostre sistema l'ERP Tryton. Aquest ERP es troba enfocat per la seva utilització en entorns on el sistema operatiu instal·lat sigui Linux. Tot i que també podrem instal·lar Tryton a Windows però ens apareixeran diversos problemes quan hi vulguem instal·lar GNU Health. El mòdul de GNU Health també el necessitem per poder començar a desenvolupar el nostre mòdul.

Tryton és un sistema desenvolupat amb Python, per tant, també necessitem les llibreries necessàries perquè el nostre sistema ens permeti compilar aquest llenguatge. Per les vistes Tryton utilitza XML.

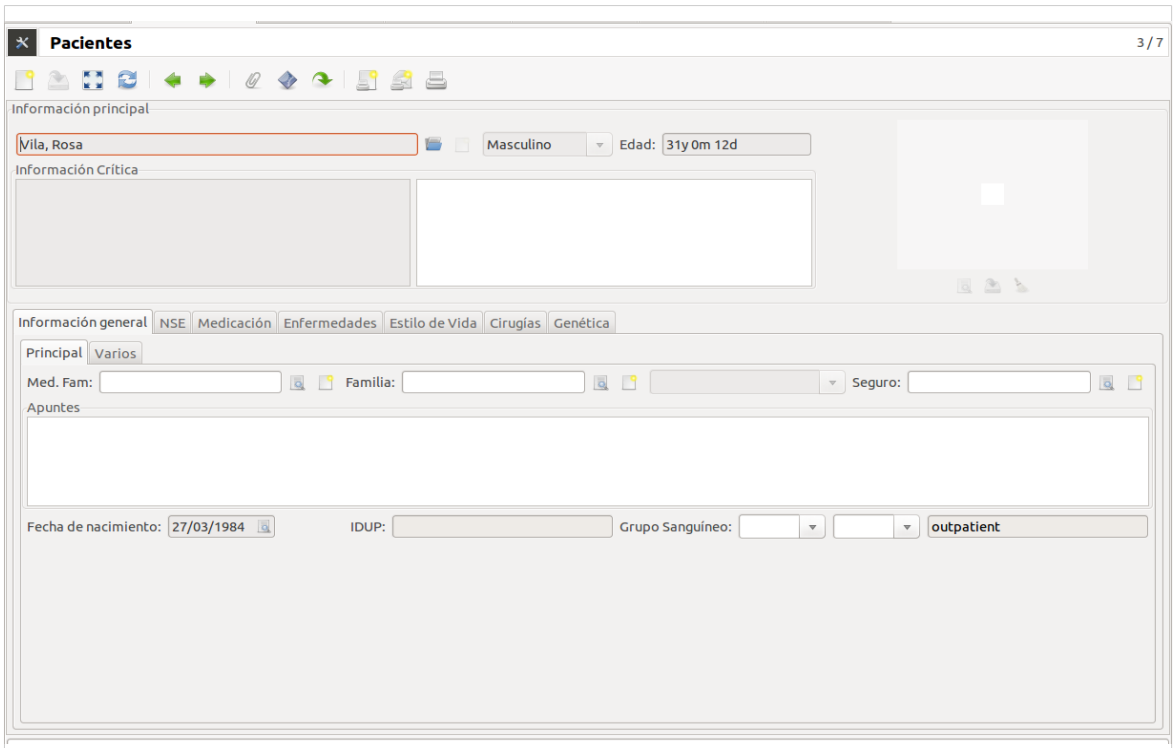
Les vistes que utilitza Tryton ens permeten poder triar si volem visualitzar les nostres dades com si es tractessin d'un formulari o d'altra banda triar l'opció de mostrar les nostres dades organitzades com si es tractés d'una llista de registres.

De manera predeterminada Tryton sempre ens mostra primer per defecte la llista. En aquesta llista podem veure un resum dels paràmetres més importants, que nosaltres prèviament hem seleccionat. Aquestes llistes es poden ordenar en funció de cada un dels paràmetres de manera creixent o decreixent.



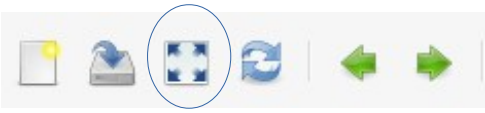

*Imatge 17: Vista llista de Tryton*

En canvi la vista tipus formulari en mostra la totalitat dels atributs que té aquesta classe i tots els paràmetres que hem d'omplir per poder guardar aquest registre. En el moment de visualització podem triar si volem veure la llista o el formulari però en el moment de crear un nou pacient, per exemple, el sistema sempre ens mostrarà el formulari.




*Imatge 18: Vista Formulari de Tryton*

Un cop hem creat el registre podrem tornar a la vista tipus llista o mantenir-nos al formulari utilitzant el botó  corresponent que trobarem al menú. D'altra banda sempre que estiguem a la llista i realitzem un doble clic sobre un dels registres se'ns obrirà una finestra emergent que  mostrarà el formulari.



*Imatge 19: Canvi de Vista*

### 7.1.1 CREAR UN NOU REGISTRE

Per poder crear un nou registre únicament hem de seleccionar la icona corresponen i seguidament aquest ens obrirà una nova finestra amb un formulari en blanc si ens trobàvem a la llista o ens habilitarà un nou formulari en blanc si ens trobàvem a la vista tipus formulari.

*Imatge 20: Crear nou registre*

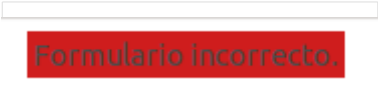
Després d'haver introduït els resultats que tenim de la prova podem guardar la prova al sistema seleccionant l'opció corresponent que trobarem al menú principal.

*Imatge 21: Guardar registre*

Si el registre s'ha guardat correctament, el sistema ens mostrarà un missatge a la part superior dreta de la pantalla indicant-nos que el registre s'ha guardat correctament.

*Imatge 22: Registre guardat correctament*

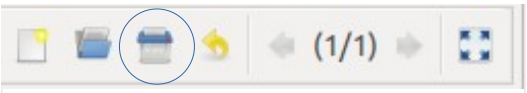
En cas contrari a la part superior dreta veurem un missatge de color vermell que ens recordarà que aquell registre no ha pogut ser guardat.



*Imatge 23: Formulari Incorrecte*

## 7.1.2 ELIMINAR UN PROVA

En el nostre cas un pacient pot tenir diverses proves i per algun motiu podem tenir la necessitat d'eliminar una d'aquestes proves. Per aquesta raó, Tryton ens proporciona una icona que ens permet eliminar cada una de les proves de manera individual. Per assegurar-nos que no eliminem cap prova de manera accidental, després de seleccionar el botó corresponent per eliminar la prova ens apareixerà un *pop-up* que ens demanarà si estem segurs que volem eliminar aquella prova.

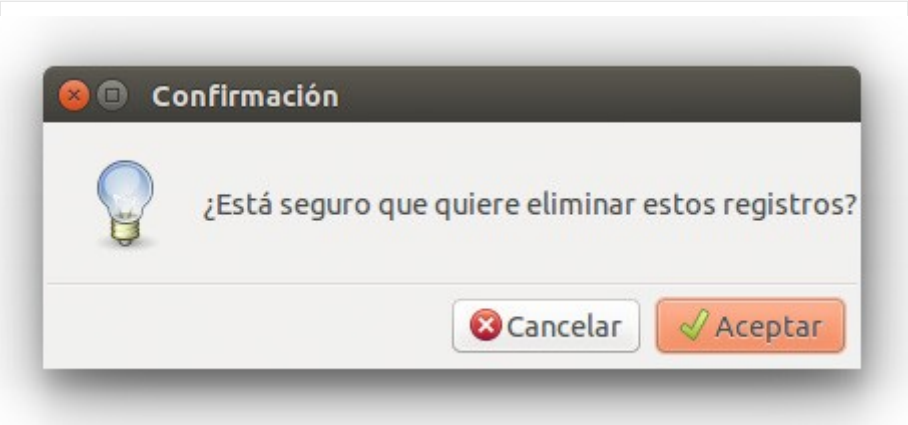


*Imatge 24: Eliminar registre*

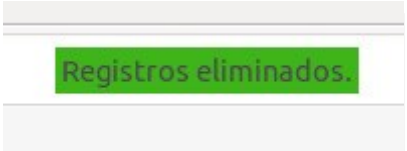
## 7.1.2 ELIMINAR UN PACIENT

En el nostre mòdul, a la pàgina principal hi apareixerà una llista amb tots els pacients que han participat en alguna de les nostres proves. Per poder eliminar algun d'aquests registres no trobarem cap botó que ens ho permeti sinó que haurem de seleccionar el registre desitjat i prémer «Control + D» al nostre teclat. Per confirmar que volem eliminar el registre ens demanarà una confirmació.

Després d'eliminar el registre seleccionat a la part superior dreta de la pantalla ens apareixerà un missatge que indicant-nos que el registre s'ha eliminat correctament, de la mateixa manera que passa en el moment quan volem eliminar per exemple una cirurgia. Hem de tenir en compte que en eliminar una pacient també s'eliminaran tota la informació que aquest pacient tenia relacionada, com poden ser radiografies, medicaments o hospitalitzacions.




*Imatge 25: Confirmació eliminar registre*




*Imatge 26: Registre eliminar correctament*

### 7.1.3 ESTRUCTURA INTERNA

A nivell intern ens trobem en què, contràriament al que podríem pensar, tots els fitxers necessaris per al funcionament d'un mòdul no es troben a dins de la mateixa carpeta, sinó que a dintre de la carpeta de mòduls hi trobem diverses carpetes en les quals es reparteix les diferents parts del mòdul de GNU Health. Cada una d'aquestes carpetes formem una petita part d'aquest mòdul, per exemple tenim carpetes en les quals hi han els fitxers necessaris per fer funcionar la part de cirurgies de GNU Health, altres carpetes amb tota la informació per la gestió de radiografies o una carpeta per gestionar tota la informació sobre l'estil de vida dels pacients.

El que tenen en comú totes aquestes carpetes és el tipus de documents que podem trobar al seu interior. En elles hi trobem els quatre fitxers principals en els quals hi ha descrit tot el codi del mòdul. Aquests fitxers són:

1. __init__.py
2. Tryton.cfg
3. un fitxer .py
4. un fitxer .xml

#### __INIT__.PY

El fitxer ***__init__.py*** és obligatori per poder inicialitzar qualsevol mòdul de Tryton. A l'interior d'aquest document hi trobem declarades totes les classes incloses dintre del fitxer de codi Python. D'altra banda també hi trobarem declarada la ubicació d'on es troben els arxius necessaris per poder iniciar el mòdul.

### TRYTON.CFG

En aquest fitxer hi trobem declarades les herències del mòdul i quins són els arxius XML que conten tota la informació relacionada amb les vistes del mòdul on es troba ubicat.

### FITXER .PY

L'arxiu ***.py*** és l'encarregat d'emmagatzemar tot el codi que permet el correcte funcionament del mòdul. En ell hi trobarem declarades totes les classes que anteriorment apareixent al fitxer __init__.py. En aquestes classes hi trobem totes les variables necessàries tant per realitzar els càlculs per obtenir els resultats desitjats, com per poder crear les vistes del mòdul.

### FITXER .XML

Arxiu ***.xml*** s'encarrega de crear tots els menús necessaris dintre de Tryton i de fer les crides a tots els arxius que tenim on hi han declarats els diferents formularis i llistes que formen les vistes d'aquell mòdul. Per realitzar correctament les crides a les vistes hem de tenir en compte que els formularis i les llistes es troben en fitxers XML diferents, per tant, hem d'especificar correctament quin és el nom de l'arxiu en el qual es troben cada una d'aquestes vistes.

A part d'aquests quatre fitxers a l'interior de cada carpeta hi trobem diverses subcarpetes que inclouen els arxius xml relacionats amb els formularis i les llistes, com hem comentat anteriorment, però també hi trobem arxius per modificar la icona del mòdul o informació referent a la instal·lació del sistema.

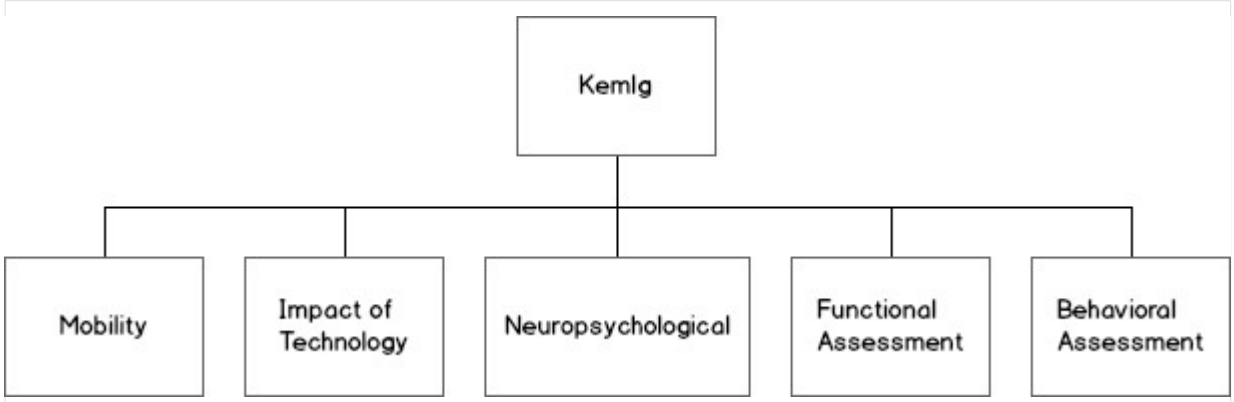
## 7.2. IMPLEMENTACIÓ

Tal com hem indicat en la nostra planificació hem decidit dividir la implementació en quatre iteracions diferents on en cada una d'elles anirem afegint funcionalitats o més testos al nostre sistema.

### 7.2.1. ITERACIÓ I: ESTRUCTURA DEL MÒDUL I «FUNCTIONAL ASSESSMENT»

En la primera iteració ens vàrem centrar en la unió del nostre futur mòdul amb GNU Health per poder relacionar els pacients ja introduïts al sistema de GNU Health amb els nostres pacients.

També es va crear l'estructura que tindria el nostre mòdul per poder començar a implementar les diferents relacions que hi haurien.




*Imatge 27: Mapa de navegació*

L'objectiu principal va ser organitzar el nostre mòdul seguint l'organització interna que tenen els mòduls més important de Tryton. Vàrem organitzar el nostre mòdul de la següent manera:

*Imatge 28: Estructura vista amb directoris*

Aquesta estructura es tradueix en la creació de sis carpetes amb els següents noms: *«Kemlg»* que és l'encarregada de crear el mòdul, menús i relacions amb el mòdul GNU Health. A partir d'aquí cinc carpetes, *«Kemlg_Behavioral_Assessment», «Kemlg_Functional_Assessment», «Kemlg_Impact_Technology_Assessment», «Kemlg_Mobility» i «Kemlg_Neuropsychological»* que heretaven de les funcionalitats de la carpeta principal.

*Imatge 29: Documents a l'interior d'una carpeta*

A l'interior de cada una d'aquestes carpetes hi trobem els quatre arxius descrits anteriorment ( __init__.py, Tryton.cfg, l'arxiu que conté el codi de Python i l'arxiu XML que conte tota la informació relacionada amb les vistes). A part d'aquests documents també hi haurà una carpeta en la qual trobarem totes les vistes necessàries.

En la majoria dels casos el document «*__init__.py*» és un document breu en el qual només hi indiquem al compilador on es troba el nostre fitxer de Python i quines classes hi trobarà al seu interior. En la següent imatge podem veure un exemple de com quedaria aquest document a l'interior de la carpeta «*Kemlg_Neuropsychological*»:

```
from trytond.pool import Pool
from .kemlg_IDF_Neuropsycolagical import *

def register():
    Pool.register(
        Mmse,
        Tmt,
        Cdt,
        Verbal_Fluency,
        Cpm,
        Ravlt,
        Rcft,
        Dsfb,
        Csfb,
        Bnt,
        Rcftc,
        Neuro,
        module='Kemlg_IDF_Neuropsychological', type_='model')
```

*Imatge 30 :fragment codi __init__.py*

Si alguna de les classes no apareix a l'arxiu amb el codi Python el sistema ens retornarà un error i no ens deixarà inicialitzar el sistema.

A l'interior de l'arxiu amb el codi Python i trobem tot el codi que farà funcionar el nostre mòdul. A les primeres línies d'aquest document inclouen informació sobre totes les classes que hi trobarem en ell i a continuació ens començarem a trobar totes les classes declarades al document *«__init__.py».*

```
class Mmse(ModelSQL, ModelView):
    'Mini Mental State Examination'
    __name__ = 'Mmse'
```

*Imatge 31: Declaració d'una classe*

Cada classe té un nom mitjançant el qual podem relacionar amb la vista on volem mostrar tota la informació obtinguda en aquesta classe. Per tant, hem de tenir en compte que la variable ***«__name__»*** és l'encarregada de realitzar aquesta unió.

En aquest fitxer també hi hem d'introduir totes les associacions que hi ha entre els diferents mòduls. Tryton té l'opció de declarar variables definint 3 tipus d'associacions diferents:

**Many2One:** Molts registres cap a un. En el nostre cas ho podem veure com que moltes proves poden pertànyer a un únic pacient.

**One2Many:** És el cas contrari que el Many2One, és a dir, un registre cap a molts. Ho utilitzarem per relacionar un pacient amb les seves diverses proves. Tota classe amb un One2Many ha de tenir declarat a la classe «destí» una variable Many2One per poder tenir una relació completa.

**Many2Many:** És una relació molts-molts. En el nostre mòdul no hem utilitzat cap relació d'aquest tipus. Per la seva implementació s'ha de utilitzar una taula intermèdia que ens emmagatzemarà l' «id» de les dues classes

Una vegada acabat d'implementar el codi Python passem al document .xml que ens permetrà crear les vistes. El primer pas serà afegir el nostre mòdul al menú principal de Tryton perquè els usuaris hi puguin accedir amb facilitat. Per afegir-lo al menú hem d'indicar quina vista s'ha de cridar en el moment que l'usuari seleccioni el nostre mòdul.

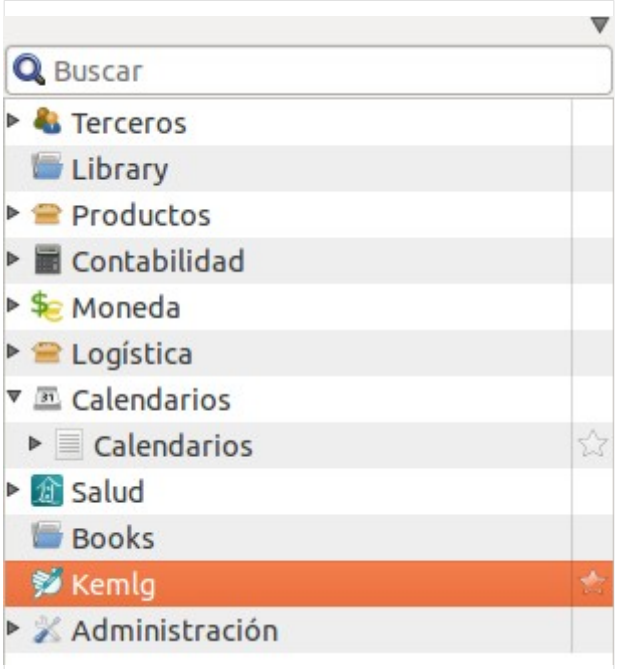
```
<record model="ir.action.act_window" id="action_kemlg_view">
    <field name="name">Kemlgs</field>
    <field name="res_model">kemlgkemlg</field>
</record>

<record model="ir.action.act_window.view" id="act_kemlg_view_list">
    <field name="sequence" eval="10"/>
    <field name="view" ref="kemlg_tree"/>
    <field name="act_window" ref="action_kemlg_view"/>
</record>
<record model="ir.action.act_window.view" id="act_kemlg_form">
    <field name="sequence" eval="20"/>
    <field name="view" ref="kemlg_view"/>
    <field name="act_window" ref="action_kemlg_view"/>
</record>

<menuitem name="Kemlg"
    id="IDF_menu" sequence="50"
    action="action_kemlg_view"/>
```

*Imatge 32: Fragment codi creació de menús*

Una vegada afegit el nostre mòdul al menú hem d'indicar quins arxius contenen la informació per crear les dues vistes que tenim, llista i formulari, i quina és la classe del codi Python que conté totes les variables necessàries per enllaçar els dos codis.




*Imatge 33: Menú principal Tryton*

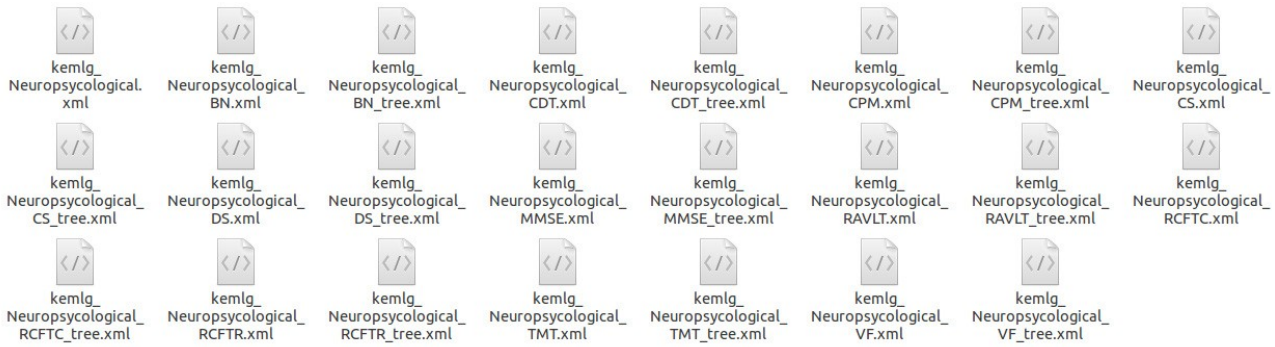
En el fragment de codi mostrat a la imatge 32 podem veure com relacionem el formulari amb nom *«Kemlg»* amb la classe Python amb nom «*kemlgkemlg*». De la mateixa manera també relacionem aquesta classe amb la llista «*kemlg_tree*». Aquesta sèrie d'instruccions ens deixen inicialitzades les dues vistes.

```
<record model="ir.ui.view" id="kemlg_view">
    <field name="model">kemlgkemlg</field>
    <field name="type">form</field>
    <field name="name">kemlg</field>
</record>

<record model="ir.ui.view" id="kemlg_tree">
    <field name="model">kemlgkemlg</field>
    <field name="type">tree</field>
    <field name="name">kemlg_tree</field>
</record>
```

*Imatge 34: Creació formulari i llista*

Després d'haver implementat el nostre codi amb xml encarregat d'unir el codi Python amb les vistes anem a crear la llista i el formulari corresponent. A l'interior de cada un dels directoris hi trobem una carpeta anomenada «*view*». A l'interior d'aquest directori hi ha els dos documents xml corresponents a cada una de les classes declarades al fitxer de codi Python.




*Imatge 35: Vistes de la categoria Neuropsychological*

El pas següent és crear un formulari que ens permeti introduir totes les dades necessàries. Per les nostres proves, que tots els camps estiguin ordenats d'una manera agradable i que faciliti la usabilitat de cara l'usuari final. Aquest formulari també es troba implementat «xml».

Per cada un dels camps que volem incloure dintre del formulari hem de realitzar dues crides. En primer lloc, realitzem una crida anomenada «label» per incloure el nom que ha de tenir al formulari. Aquest nom es troba declarat en el moment que inicialitzem el camp al fitxer de Python. En segon lloc, s'ha de fer la segona crida al mateix camp per poder introduir al formulari el valor que té aquella variable.

En ser un document XML podem utilitzar la majoria de les comandes d'aquest llenguatge per poder organitzar el nostre formulari. Hem de tenir en compte que totes les variables declarades com a «*Boolean*» al document Python es transforma en un «*checkbox*» en el document de XML. De la mateixa manera, tots els camps declarats com a «*Selection*» es transformen en llistes desplegables al XML.

```
<data>
    <xpath expr="/form/notebook/page[@id=&quot;kemlg_Information&quot;]" position="after">
        <page string="Mobility" id="kemlg_mobility2">
            <newline/>
            <notebook>
                <page string="POMA" id="kemlg_mobility_poma">
                        <field name="PomaResults"/>
                </page>
                <page string="FES" id="kemlg_mobility_fes">
                        <field name="FesResults"/>
                </page>
                <page string="6min Test" id="kemlg_mobility_SixMin">
                        <field name="SixMinutes"/>
                </page>
                <page string="10m Test" id="kemlg_mobility_TenM">
                        <field name="TenMeters"/>
                </page>
            </notebook>

        </page>
    </xpath>
</data>
```

*Imatge 36: Fragment de codi d'un formulari*

Una vegada finalitzat el formulari creem la llista. En aquest cas cada un dels camps que afegim a la llista serà una columna nova. Això ens permetrà triar per quina de les columnes volem ordenar la llista.

A diferència dels formularis en la llista únicament hem de fer la crida dels seus camps una vegada. Només hem de fer una crida amb el nom de «*field*».

```xml
<?xml version="1.0"?>
<tree string="tenMetersTest">
    <field name="date"/>
    <field name="tenM"/>
</tree>
```

*Imatge 37: Fragment codi per crear una llista*

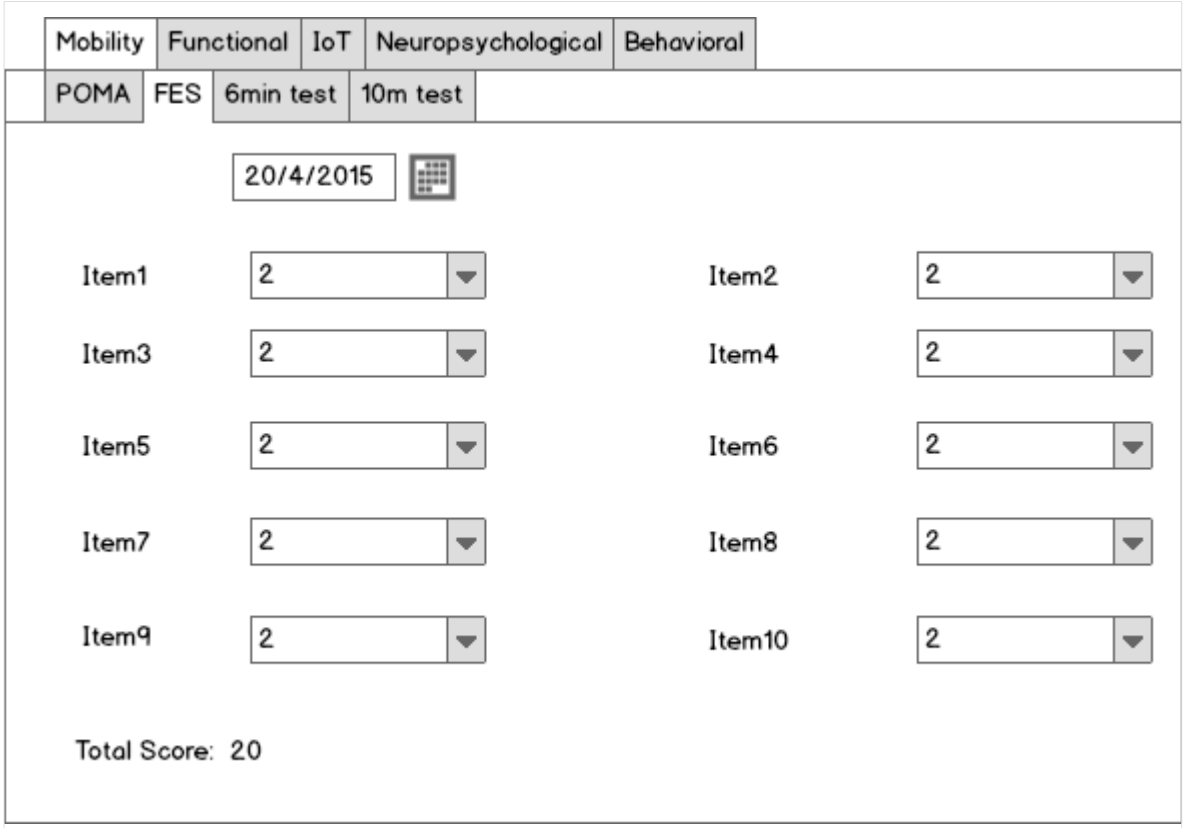
Finalment només ens queda implementar el fitxer «*Tryton.cfg*». En aquest document únicament hi posarem quines dependències té aquest mòdul i on podem trobar el fitxer XML que conté  les crides a les vistes. En la següent imatge tenim un exemple de com seria un d'aquest fitxer:

```
[tryton]
version=2.6.4
depends:
    Kemlg_IDF
xml:
    kemlg_IDF_Neuropsychological_view.xml
```

*Imatge 38: Fragment codi del fitxer tryton.cfg*

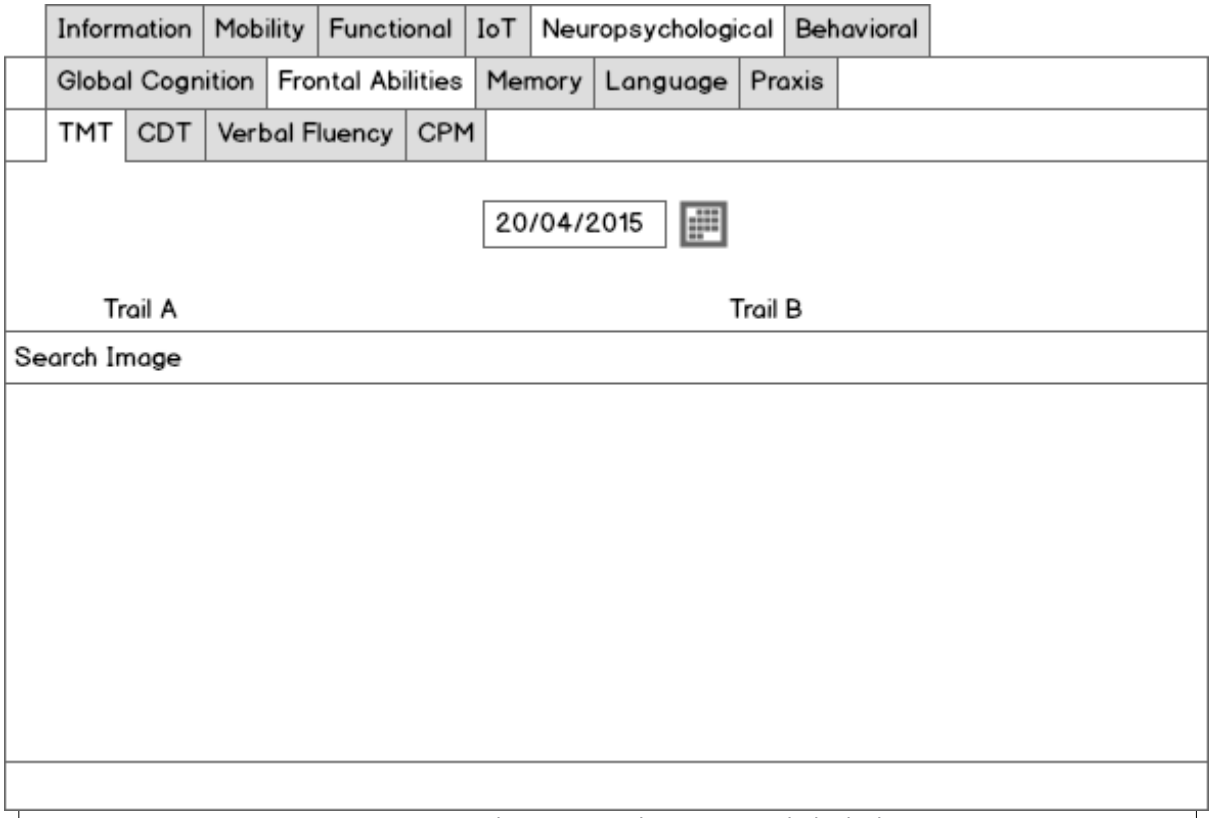
Una de les parts més important perquè aquesta aplicació arribi a bon port és que l'usuari final no trobi problemes en el moment d'utilitzar el sistema. Per aquest motiu vam decidir utilitzar la mateixa estructura que ens proporcionava GNU Health i per tant havíem de treballar amb diferents pestanyes.

En la majoria de les pestanyes no tenim cap problema d'usabilitat perquè únicament treballem amb dos subnivells de pestanyes on en el nivell principal hi ha els noms dels diferents grups de proves que formen el nostre mòdul i en el segon nivell trobem el nom de les diferents proves que hi ha a dintre d'aquell grup.




*Imatge 39: Mockups de les proves de "Mobility"*

El major problema de la part d'usabilitat el troben el moment que entrem a tractar amb les proves de l'interior de l'apartat «*Neuropsychological*». En aquest apartat trobem que ens apareixen tres subnivells de pestanyes, el limit que en termes d'usabilitat es considera correcta. Per aquest motiu hem mantingut aquesta estructura, tot i veure's una mica carregada d'informació és la que ens mostra de la millor manera totes les dades que el metge està esperant trobar en el nostre sistema.




*Imatge 40: Mockups proves de "Neuropsychological"*

En aquesta iteració a part de crear l'estructura bàsica del que seria el nostre futur mòdul també es van implementar les dues primeres proves que pertanyien a la part de *«Funcional Assessment».* Aquestes dues proves van ser *«Instrumental Activities of Daily Living (IDAL)»* i *«Barthel Index»*. D'aquestes proves en vam implementar tant la part de càlcul com la part de visualització.

## INSTRUMENTAL ACTIVITIES OF DAILY LIVING (IDAL)

Aquesta prova determina quin és el grau d'autonomia que tenen les persones d'avançada edat per realitzar tasques de la vida quotidiana. Aquesta prova es basa en vuit preguntes sobre aspectes quotidians com són cuinar, utilitzar el telèfon o prendre la medicació. Cada una de les vuit preguntes té quatre respostes amb diferent puntuació cada una d'elles.

Analitzant quina seria la millor manera de representar tota aquesta informació ens vam decantar per fer cada pregunta amb quatre «*checkboxs*» amb les quatre possibles respostes que ens donaven per cada una de les preguntes. D'altra banda també vam crear una variable que ens calculava en funció de les respostes seleccionades quina era la puntuació total d'aquell pacient en aquella prova.

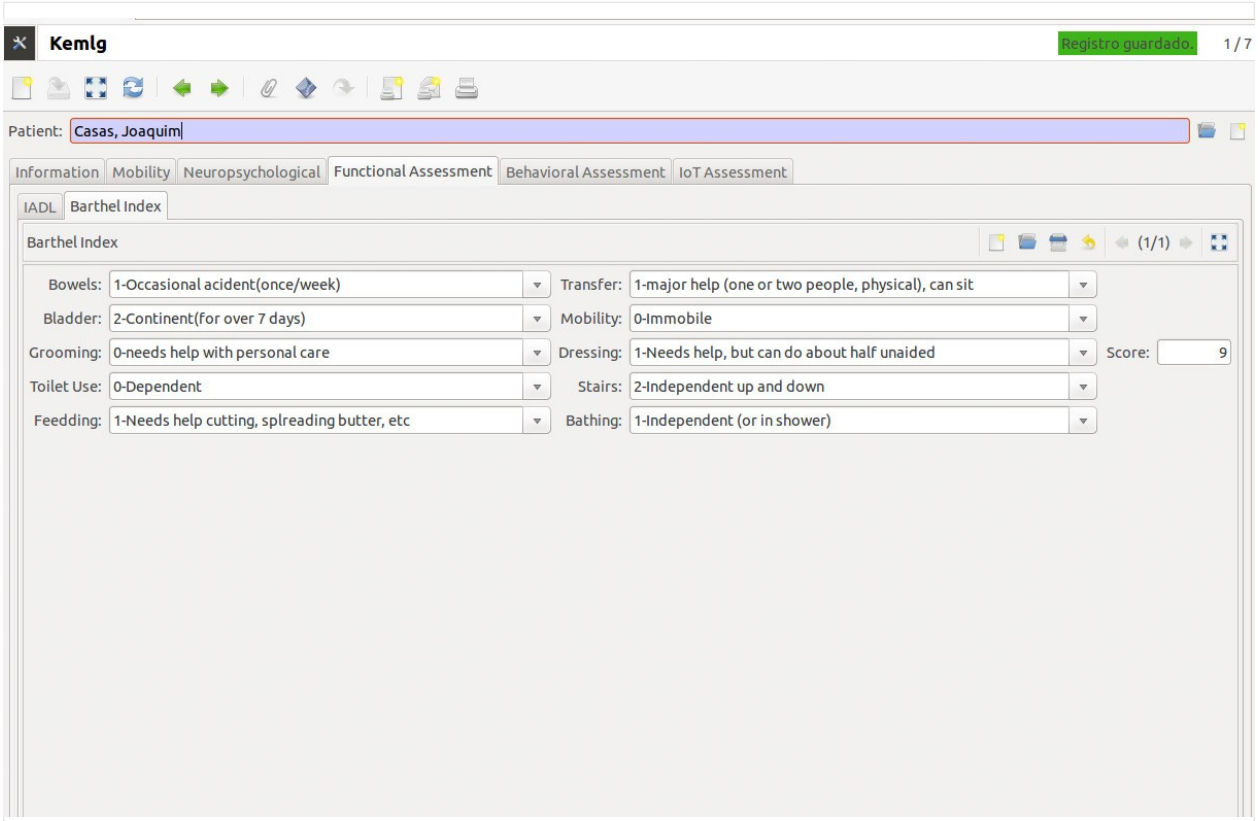
En la llista d'aquesta prova únicament hi tindrem dos valors. El primer farà relació a la data en la qual es va realitza la prova i el segon serà el resultat que el pacient ha obtingut en ella.

## BARTHEL INDEX

«*Barthel Index*» és una prova molt semblant a l'anterior. Es tracta de deu preguntes sobre activitats que es realitzen a la vida diària. Cada un d'aquestes preguntes té una puntuació diferent. La puntuació màxima és vint. Si el nostre resultat a la prova és pròxim a vint, ens indica que som una persona totalment independent, en cas contrari, seríem una persona dependent.

Per aquesta prova hem utilitzat deu llistes desplegables on en cadascuna d'elles el nostre pacient ha de triar quina és la resposta que s'adequa a la seva vida diària. En funció de les respostes la variable corresponent calcula quin és el grau de dependència del pacient.

De la mateixa manera que en el cas anterior la llista d'aquesta prova només disposarà de dues columnes. La primera serà la data en la qual s'ha realitzat la prova i la segona quina és la puntuació obtinguda en ella.




*Imatge 41: Formulari del Barthel Index*

### 7.2.2 ITERACIÓ II: PROVES *«BEHAVIORAL ASSESSMENT» I «MOBILITY»*

Després de comprovar que totes les funcionalitats de la primera iteració funcionaven correctament vam passar a implementar diverses proves més. En aquest cas vam decidir que en aquesta iteració seria convenient implementar totes les proves que formaven part del «*Behavioral Assessment*» i de «*Mobility*».

#### BEHAVIORAL ASSESSMENT

##### STATE TRAIT ANXIETY INVENTORY – Y (STAI-Y)

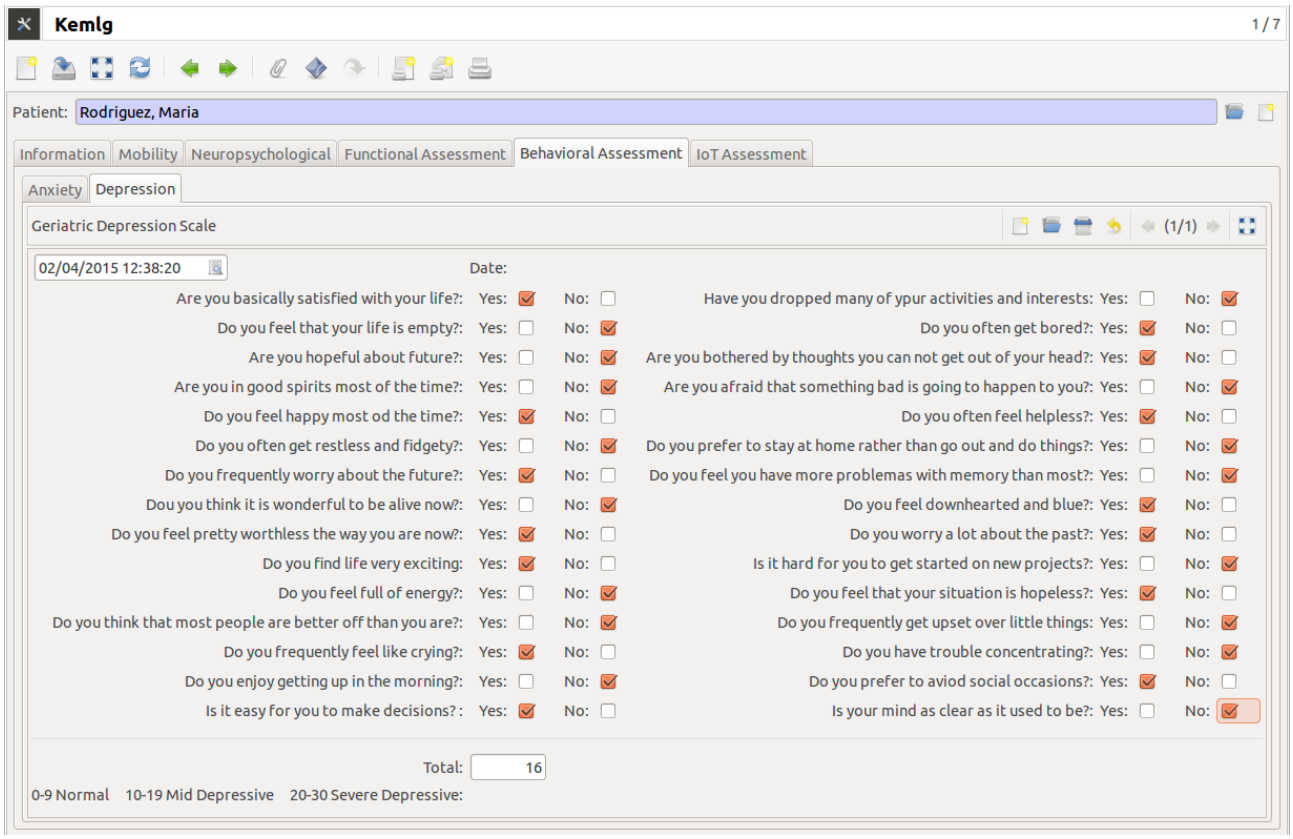
Aquesta prova s'encarrega de determinar els possibles símptomes de depressió o ansietat que poden patir els adults. Mitjançant vint preguntes on cada un d'elles té quatre respostes on cada resposta té un pes diferent per determinar l'estat del pacient.

En el formulari hem volgut recrear el clàssic formulari de paper en el qual tenim una pregunta i seguidament les quatre preguntes. També hi hem afegit un camp per obtenir el resultat d'aquesta prova i un altre perquè ens indiquin la data en la qual s'ha dut a terme la prova. En la llista només tenim la data i el resultat de la prova.

##### GERIATRIC DEPRESSION SCALE (GDS)

Tal com indica el seu nom, aquesta prova es troba destinada per detectar la depressió en les persones d'edat avançada. En aquesta prova es realitzen al pacient trenta preguntes en les quals ha de contestar si o no. Depenent de la resposta augmenta el grau de tenir depressió.

Igual que en el cas anterior per voler mantenir la màxima similitud amb el paper de la prova cada pregunta tindrà dos «*checkboxs*» corresponent a «Si» i «No» on cada pacient haurà de seleccionar la resposta correcta a les preguntes.




*Imatge 42: Exemple prova Behavioral Assessment*

## TINNETTI PERFORMANCE ORIENTED MOBILITY (POMA)

Aquesta prova, destinada a persones d'edat avançada, té com a principal funció avaluar l'equilibri I la mobilitat d'aquestes persones mitjançant un seguit de proves que se separen en dues categories, "*gait*" i *"balance"*, amb deu proves cada una d'aquestes categories.

Hem decidit que aquest formulari estigui compost per dos grans grups amb deu ítems desplegables cada un d'ells. Cada una de les opcions diferents opcions d'aquests ítems tenen un pes diferent que mitjançant una variable se sumen els deu resultats obtinguts a cada una de les categories i mostraran el resultat final de la prova.

## FALL EFFICACY SCALE (FES)

Aquesta prova ens indica la possibilitat que té el pacient, una persona d'edat avançada, de caure o la por d'ell mateix a caure. A través de deu preguntes en les quals han d'introduir un valor entre 0-10, com més gran és el valor més gran és la por a caure.

De la mateixa manera que en el cas anterior hem volgut crear una llista amb les preguntes on cada una d'elles té associada una llista desplegable on es pot seleccionar la resposta corresponent introduint un valor comprés entre 0-10. Al tractar-se d'una llista on únicament hi han números també tenim l'opció d'introduir el número corresponent en comptes de desplegar la llista per cercar el resultat.

## SIX MINUTES WALKING TEST

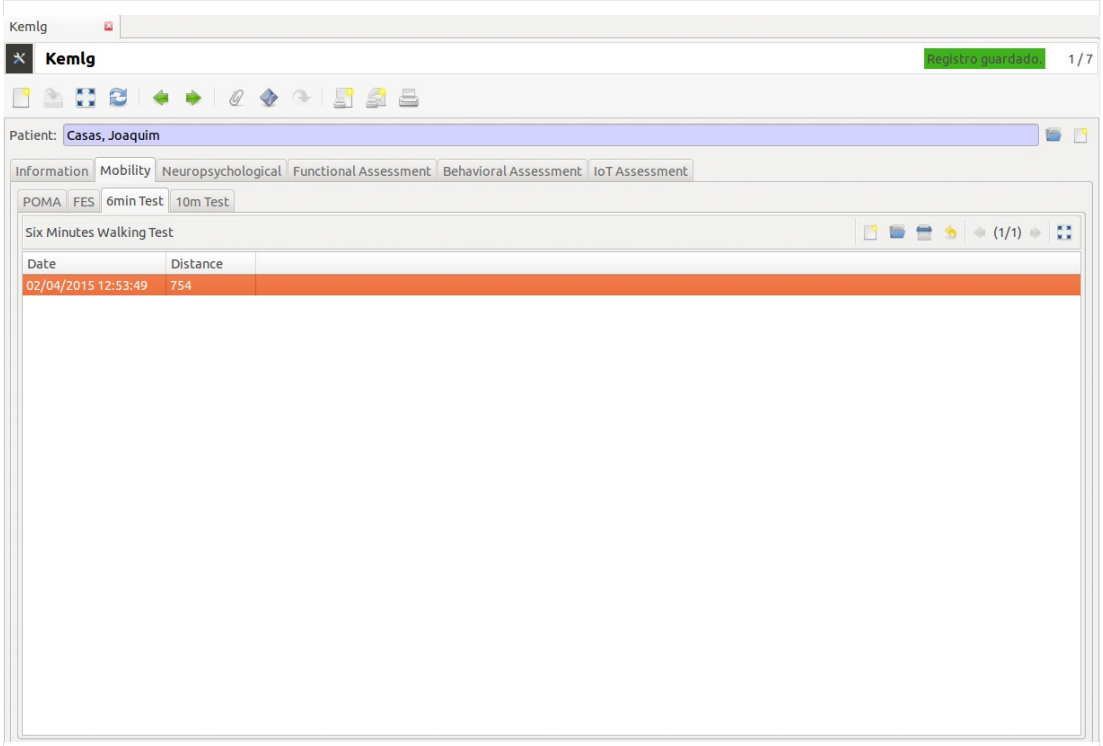
Tal com el seu nom indica, en aquesta prova es mesura la distància que pot recórrer una persona en sis minuts sobre una superfície dura i plana. En aquest temps també es permetrà al pacient seure per descansar. El resultat de la prova serà únicament la distància total que el pacient ha pogut recórrer en sis minuts.

En el formulari hem creat dos grups. El primer grup correspon a les dades que té el pacient abans de començar la prova (pressió arterial, ritme cardíac, saturació d'oxigen i dificultat de respiració). En el segon grup s'introduiran tots els paràmetres del pacient després de realitzar la prova (total de distància recorreguda, saturació d'oxigen, dificultat de respirar i notes). En aquest segon subgrup hem introduït una variable amb el nom de «*Notes*» on el metge pot introduir qualsevol dels incidents que han ocorregut durant el transcurs de la prova. Per la vista del tipus llista únicament necessitem la data i la distància total recorreguda.

## TEN METERS TEST

Aquesta prova està orientada a tots aquells pacients que pateixen «*Parkinson*» o algun desordre neurològic en el moviment. El pacient camina deu metres en el menor temps possible però només es mesuren els 6 metres intermedis. Aquesta prova es realitza sis vegades. Les tres primeres vegades s'indica al pacient que recorri els deu metres a una velocitat que ell consideri còmode i les pròximes tres vegades s'indica al pacient que recorri els deu metres a la màxima velocitat possible. De cada una dels sis intents s'anoten els temps i posteriorment s'analitzen per separat els dos tipus de proves realitzades. Per cada un dels dos grups s'indica quin és el millor temps i quin és el temps mig que s'ha trigat.

Per aconseguir una rapida visualització dels resultats en el formulari només necessitem que l'usuari del sistema introdueixi el temps que ha realitzat el pacient. Seguidament el sistema s'encarrega de calcular tots els resultats necessaris. A la llista mostrarem la data en la qual es va dur a terme la prova i la distància mitja recorreguda en cada un dels dos tipus de prova.




*Imatge 43: Exemple llista de la categoria "Mobility"*

Després d'implementar totes aquestes proves es varen realitzar els jocs de proves pertinents per assegurar-nos que totes les proves es guardaven correctament a la base de dades i que no teníem cap tipus d'error o interferència amb proves anteriors.

### 7.2.3. ITERACIÓ III: PROVES *«IMPACT OF TECHNOLOGY ASSESSMENT»*

En aquesta iteració vam desenvolupar únicament les proves que pertanyien a la secció de «Impact *of Technology Assessment*». Aquestes proves es diferenciaven de totes les anteriors desenvolupades fins al moment perquè algunes d'elles estan formades de qüestionaris de prop de 30 preguntes i per tant s'havia de buscar quina és la millor manera de mostrar tota aquesta informació sense fer que l'usuari trobi que l'aplicació és difícil d'utilitzar.

#### QUEBEC USER EVALUATION OF SATISFACTION WITH ASSISTIVE TECNOLOGY (QUEST)

Amb aquest qüestionari es valora la satisfacció que té un pacient en la utilització d'un producte tecnològic que l'ajuda en la seva vida quotidiana. En aquesta prova hi ha dos grups de preguntes. En el primer anomenat «Device Satisfaction» on hi trobem vuit preguntes i el segon anomenat «Service Satisfaction» on únicament hi ha quatre preguntes. Finalment el pacient ens ha de descriure el producte amb tres adjectius.

En el nostre programa hem decidit implementar aquest qüestionari col·locant una pregunta sota l'altre tenint a la part superior dos comptadors, a part de la data en què es realitza el qüestionari, què ens indiquen quina és la puntuació que té el producte segons el pacient en cada una de les dues categories.

## PSYCHOLOGICAL IMPACT OF ASSISTIVE DEVICE SCALE (PIADS)

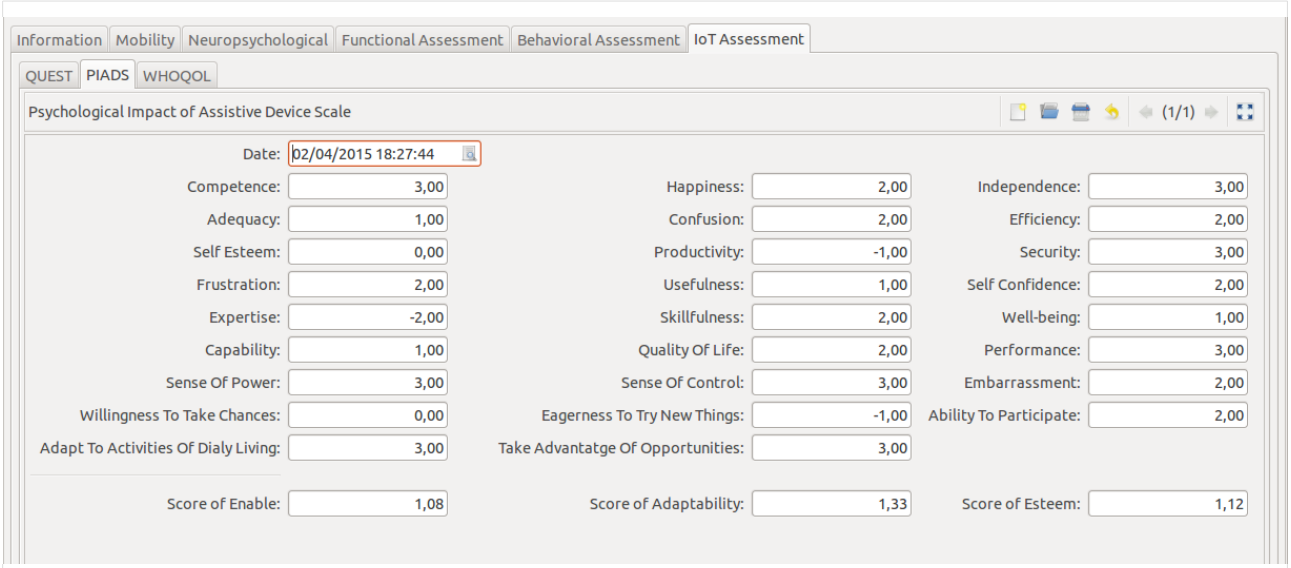
Aquest qüestionari es troba dissenyat per saber l'impacte que ha tingut la utilització d'un cert aparell tecnològic en la vida del pacient. Per mesurar aquest impacte es realitza un qüestionari on el pacient ha de puntuar l'impacte que ha tingut aquest aparell amb un valor entre -3 i 3. En aquesta prova hi han 26 preguntes dividides en tres subgrups: l'habilitat, format per 12 preguntes, adaptabilitat, en el qual hi ha 6 preguntes  i finalment l'autoestima, format per 8 preguntes.

En aquest cas hem creat una llista amb les 26 preguntes on cada participant ha d'introduir el resultat que creu apropiat entre -3 i 3. A la part inferior del formulari hi trobem els tres totals i depenent del tipus de pregunta que contestem aquesta incrementa un resultat o un altre. En la vista del tipus llista hi veiem una columna amb la data en la qual s'ha realitzat la prova seguida de tres variables amb cada un dels diferents resultats aconseguits per cada un dels grups.

## THE WORLD HEALTH ORGANIZATION QUALITY OF LIFE

Per acabar, en l'última prova d'aquesta iteració, tenim un extens qüestionari que mesura els sentiments de les persones a través de 26 preguntes en les quals cada una d'elles s'ha de contestar amb una de les cinc respostes que se'ns proporciona.

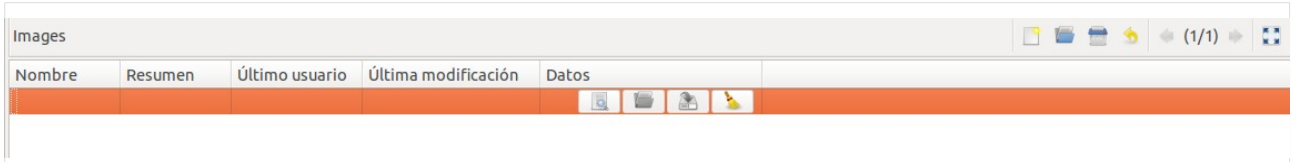
De la mateixa manera que en la prova anterior hem cregut que la millor manera de representar aquest qüestionari és situant una pregunta sota de l'altre simulant que es tracta d'un qüestionari que ens podrien donar per omplir amb paper.




*Imatge 44: Exemple de prova de Impact of Technology*

## 7.2.4. ITERACIÓ IV: PROVES *«NEUROPSYCHOLOGICAL»*

Després poder implementar correctament totes les proves anteriors ens vam adonar que les següents proves no es podien implementar només creen un formulari perquè moltes d'elles necessiten un document addicional per poder veure quin és el resultat de la prova. En la majoria de casos es demana al pacient que identifiqui una imatge, dibuixi una figura predeterminada, etc. Per això en aquesta part del mòdul vam haver d'incloure l'opció que es poguessin guardar imatges en aquestes proves.




*Imatge 45: Opció incorporar imatge a una prova*

Amb aquesta opció de carregar imatges aconseguim una funcionalitat addicional que permet poder veure totes les imatges necessàries per avaluar la prova, descarregar aquestes imatges o incloure imatges de com hauria set una resposta correcta.

## MINI MENTAL STATE EXAMINATION (MMSE)

Aquesta prova està composta de trenta preguntes que ens proporcionaran informació sobre el sentit de l'orientació, la memòria, el càlcul o el llenguatge, entre altres aspectes. Aquestes trenta preguntes són un conjunt d'ordres que es dóna al pacient i hem d'anotar quina és la resposta que proporciona el pacient. Cada una de les respostes que ens proporciona el pacient té una puntuació que hem d'apuntar al nostre formulari.

En el nostre sistema hem d'anotar la puntuació que ha obtingut el pacient en cada una de les respostes per finalment poder concloure quin és el millor diagnòstic. Totes les preguntes les trobem agrupades en cinc subgrups per tant en el nostre formulari hem decidit separar aquests cinc subgrups per fer més fàcil la interpretació a l'usuari final.

## TRIAL MAKING TEST (TMT)

Un cop arribat a aquest temps ens adonem que és completament diferent a tots els anteriors, ja que en ell no ens trobem amb cap tipus de formulari que hàgim d'omplir, ni preguntes per contestar. En aquesta prova es proporciona al pacient dos fulls de paper. En el primer full hi han una sèrie de números desordenats, repartits per tot el full però sense cap número repetit. En aquest cas el pacient ha de recórrer tots els números en ordre en el menor temps possible. En el segon cas en el full hi han número i lletres barrejades. En aquest segon full el pacient ha d'anar intercalant números i lletres per ordre és a dir 1-A-2-B. En els dos casos només hem de corregir el pacient si s'equivoca i anotar el temps final.

El nostre sistema realitzarem una presentació molt simple on únicament tindrem l'opció d'apuntar el temps que ha trigat el pacient a completar cada una de les proves però també hi afegirem una opció perquè la persona encarregada de supervisar la prova pugui adjunta les dues imatges corresponents als dos fulls en els quals el pacient ha realitzat la prova. Amb aquestes dues imatges podrem veure si el pacient ha comès algun error durant l'execució de la mateixa.

## CLOCK DRAWING TEST (CDT)

«*Clock Drawing Test*» és una prova que consisteix a proporcionar-li al pacient un full en blanc únicament amb un cercle al mig i demanar-li que ens dibuixi un rellotge amb una hora en concret.

En aquest cas també ens interessaria emmagatzemar la imatge que ha creat el pacient per poder avaluar-la. Per aquest motiu la nostra vista únicament contindrà uns «*checkboxs*» que ens mostraran el resultat obtingut i l'opció de poder incloure la imatge.

## VERBAL FLUENCY

Per avaluar la capacitat que té una persona per crear estratègies a partir d'una situació inicial. Aquest test es divideix en dues proves. La primera d'elles consisteix a proporcionar al pacient una categoria en concret, per exemple animals (categoria més indicada segons el test estàndard), a partir d'aquest moment el pacient ens ha de dir el màxim nombre de paraules que pertanyen a aquesta categoria. En la segona prova es diu al pacient una lletra i ens ha de dir el màxim nombre de paraules que comencin amb aquella lletra.

En aquesta ocasió en el nostre sistema únicament necessitem guardar a la base de dades el nombre de paraules de cada categoria que el pacient ha aconseguit dir.

## COLOR PROGRESSIVE MATRICES (CPM)

En aquesta prova es mesura la intel·ligència del pacient com també la seva capacitat deductiva. La prova està formada per 36 taules. En cada una d'aquestes taules hi ha una imatge en la qual falta una part i tenim sis possibles opcions sobre quina és la part que falta. El pacient ens ha de contestar les 36 taules indicant quina creu que és la resposta correcta.

En el nostre sistema crearem 36 caselles en les quals se'ns haurà d'indicar si el pacient ha encertat o no cada una de les taules. D'altra banda també hi inclourem l'opció d'incloure una imatge al registre per poder veure quina ha set la resposta del pacient a cada una de les taules.

## REY AUDITORY VERBAL LEARNING TEST

Aquesta prova es pot dividir en tres apartat. En el primer d'ells es diu al pacient una llista de 15 paraules. Després d'haver dit les 15 paraules es demana al pacient que repeteixi el màxim nombre de paraules. Aquesta seqüència es repetirà un total de 5 vegades. En el següent apartat es dirà al pacient una llista de 15 paraules totalment diferent de la inicial i es demanarà al pacient que repeteixi el nombre màxim de paraules de la primera llista. Per acabar, després de trenta minuts de descans es demanarà al pacient que repeteixi totes les paraules que recordi de la primera llista i finalment amb una llista escrita amb diverses paraules, es demanarà al pacient que reconegui quines d'elles formen part de la primera llista.

Tot i que en aquesta prova té molts apartats a nosaltres només ens interessa els resultats de cada un d'ells per tant crearem tres subgrups en els quals hi hauran els resultats de les tres parts de la prova. Remarcarem quins han set els resultats de cada una d'elles per una interpretació més rapida.

## REY COMPLEX FIGURE TEST – RECALL

Per avaluar la memòria de reproducció que pot tenir un pacient utilitzem el «*Rey Complex Figure Test*». Es proporciona al pacient una figura geomètrica complexa i en primer moment es demana que la copiï. Passats 20 minuts i sense referència visual, es demana al pacient que torni a dibuixar la mateixa figura geomètrica intentant recordar la major quantitat de destalls. Finalment s'avaluarà el nombre de parts de la figura que ha aconseguit recordar i d'aquesta manera obtenir la seva puntuació.

En el nostre sistema hi haurà una llista amb tots els detalls que el pacient hauria hagut de recordar però únicament marcarem els que ha recordat. D'aquesta manera es calcularà el resultat que ha obtingut en pacient en aquesta prova.

## DIGIT SPAN FORWARD AND BACKWARD

Consisteix a lletrejar xifres d'una a una començant amb 2 xifres I acabant amb 9 xifres. Després de cada una de les xifres el pacient ens ha de repetir quines són les xifres que hem dit. De la mateixa manera es demanarà al pacient que en compte de repetir les xifres que hem dit en el mateix ordre, les repeteix en ordre invers. El resultat de la prova és el número de nombres correctes que el pacient ha sigut capaç de dir.

Nosaltres dividirem el nostre formulari en dues parts corresponent a la part que s'ha de dir seguin el mateix ordre  o la segona part amb les xifres que s'han de dir a l'inrevés. Per tant el nostre formulari tindrà dues parts amb els mateixos caps (total xifres correctes, puntuació i notes). La puntuació de la prova es calcularà en funció del nombre de respostes correctes. Al camp de «*Notes*» el metge pot introduir qualsevol tipus d'informació relacionada amb la prova.

## CORSI SPAN FORWARD AND BACKWARD

En aquest cas mesurarem la memòria visual que tenen els pacients. La persona encarregada de realitzar aquesta prova tocarà uns cubs seguint una seqüència. El pacient haurà de repetir aquesta seqüència en el mateix ordre en què l'han realitzat o a l'inrevés, depenen del que es demani. La puntuació d'aquesta prova fa referència a la seqüència més llarga que hem pogut repetir.

En aquesta prova aprofitarem el formulari realitzat per la prova «*Digit Span Forward and Backward*», ja que es necessiten els mateixos camps. En la llista únicament hi inclourem la data en la qual s'ha realitzat la prova i quina ha estat la seva puntuació.

## BOSTON NAMING TEST

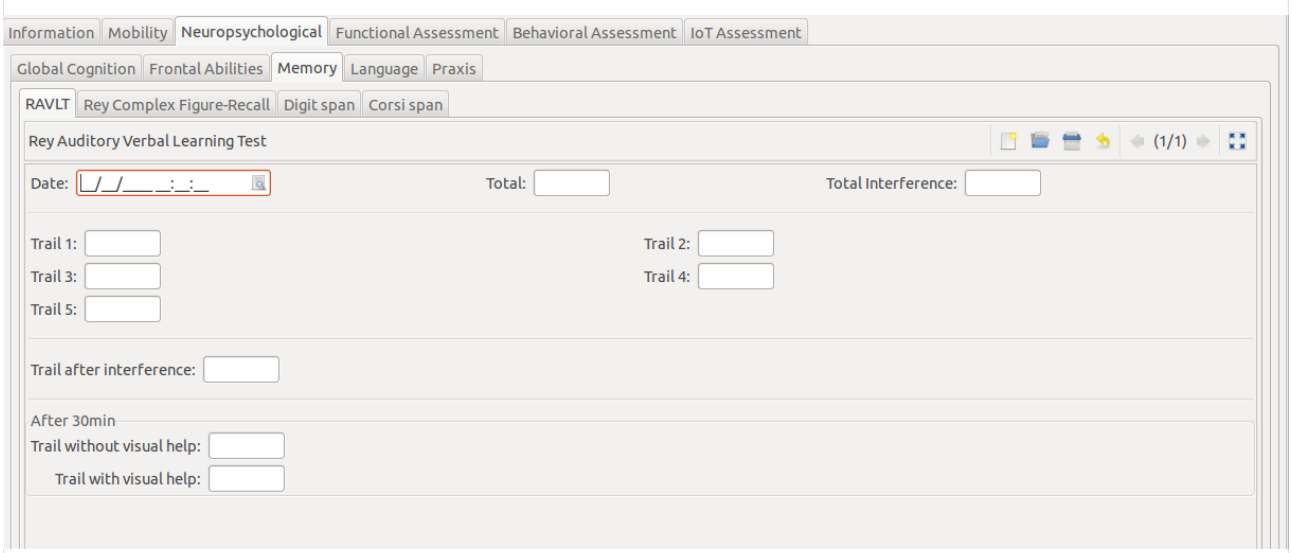
«Boston Namimg Test» es basa a mostrar al pacient una imatge i ens ha de dir de què es tracta. Amb aquesta prova mesurem l'habilitat del pacient per reconèixer imatges. En la prova es mostren un total de 60 imatges on cada una d'elles té una puntuació de 0 si la resposta és incorrecta o 1 en cas contrari.

En el nostre sistema hem creat una taula on hem d'introduir el resultat per cada una de les imatges. A la part inferior del formulari hi hem habilitat l'opció de poder adjuntar a la prova els documents que es creguin necessaris perquè es pugui diagnosticar el resultat de la mateixa.

## REY COMPLEX FIGURE TEST – COPY

En aquesta prova s'utilitza el mateix dibuix que utilitzem en el «Rey Complex Figure Test – Recall » però en aquest cas al pacient se li demanarà que copii amb tot luxe de detalls la imatge geomètrica mostrada. En el moment en què el pacient ha acabat de realitzar el dibuix la persona encarregada de supervisar la prova compararà els dos dibuixos per poder dir quin és el resultat. Per calcular aquest resultat es disposa d'una taula de puntuació en la qual cada element que el pacient no ha copiat es resta un punt.

En el nostre formulari copiarem aquesta taula de puntuació perquè s'hi pugui introduir el resultat. També tindrem l'opció de poder incloure al registre de la base de dades les imatges corresponents de la prova.



*Imatge 46: Exemple prova de "Neuropsychological"*

Després d'haver inclòs aquest últim grup de proves al nostre projecte comprovarem que tot funciona correctament per aquest motiu abans de realitzar joc de proves final realitzarem unes petites proves amb aquests últims formularis introduïts al sistema.

## 7.3. JOC DE PROVES

Un cap ja hem acabat la implementació de totes les proves que havien de formar part del sistema que ens havien encarregat passarem a realitzar un joc de proves per comprovar que tot funciona de la manera esperada i totes les proves es guarden correctament a la base de dades.

Per dur a terme totes aquestes comprovacions crearem un grup de pacient a GNH Health. Seguidament agregarem aquest grup de pacient al nostre sistema. A cada un d'aquests pacients li introduirem un resultat per totes les proves que formen part del sistema. En alguns casos forcarem al sistema perquè ens retorni un error en el cas que realitzem alguna acció de manera incorrecta.

Per acabar el joc de proves eliminarem proves per veure si aquestes s'eliminen de manera correcta de la base de dades i també eliminarem el pacient del nostre sistema per poder veure que en el moment que s'elimina el pacient també s'eliminen totes les proves que hi havien relacionades amb ell.

# 8. CONCLUSIONS

Un cop acabat de desenvolupar tot el sistema podrem dir que hem creat el sistema que el KEMLg necessita pels seus projectes. Gràcies a aquest sistema podran reunir molta més informació relacionada amb tots els pacients que en formen part i poder veure'n quina és la seva evolució.

Una de les principals dificultats que ens vam trobar en el moment d'iniciar el projecte va ser el d'entendre com està organitzat Tryton, quina és la seva estructura interna i aprendre com es comuniquen els dos llenguatges de programació que formen part de Tryton. Després de dedicar-hi moltes hores comences a veure com està organitzat i és en aquell moment quan et pots posar a programar d'una manera molt més àgil i efectiva que la que havies dut a terme fins al moment.

D'altra banda, Tryton i GNU Health no tenen una comunitat d'usuaris tan extensa com la que pot tenir Odoo per aquest motiu és molt més complicat poder solucionar un error si no tens cap tipus de referència sobre que és el que pot passar. Per intentar resoldre aquest problema ens vam posar en contacte amb Luis Faclón, persona encarregada en el desenvolupament de GNU Health, per demanar-li ajuda per saber quina era l'estructura interna de GNU Health i si disposava d'algun tipus de manal per poder aprendre d'una manera rapida tota l'arquitectura del sistema en el qual ens estàvem introduint.

Molta amablement Luis Falcón ens va proporcionar un breu diagrama de classes que ens mostrava quina era l'organització interna que tenia GNU Health. Però no ens va poder proporcionar cap manual i ens va recomanar que la millor manera d'aprendre el funcionament de Tryton era estudiar els mòduls que ja hi havien implementats i anar descobrint com es connectaven entre ells els diferents fitxers de diferents llenguatges de programació.

Un dels problemes més importants que hem trobat en aquesta arquitectura és la seva limitació en el moment de personalitzar la interfície. Utilitzar les opcions que et proporciona Tryton acaba sent molt fàcil de cara al programador però ofereix un catàleg d'opcions de personalització molt limitat. Aquest fet ens ha produït que no puguem crear uns formularis molt vistosos.

Actualment aquest projecte ja es podria posar en funcionalment però personalment crec que la millor opció seria proporcionar-li aquest sistema a un petit grup d'estudi perquè avaluïn totes les funcionalitats que té el sistema i d'aquesta manera aconseguir un «*feedback*» per poder modificar totes aquelles parts que l'usuari final troba que estan mal dissenyades o afegir funcionalitats que ells creuen imprescindibles pel correcte funcionament de tot el sistema.

# 9. REFERÈNCIES


1-Cortés, U. (19 de 11 de 2010). *Knowleage Engineering and Machine Learning Group (KEMLG)* . Obtingut de http://www.slideshare.net/ucortes/kemlg-descriptionnovember2010

2-KEMLG. (2014). *Grup d'Enginyeria del Coneixement i Aprenentatge Automàtic. KEMLG*. Obtingut de https://kemlg.upc.edu/

3-World Health Organization. (2001). Obtingut de http://www.who.int/classifications/icf/icf_more/en/

4-Dittmann, H. T. (15 de 9 de 2009). *Open-Source-ERP-Systeme im Vergleich*. Obtingut de http://www.heise.de/open/artikel/Quelloffene-Kuer-Open-Source-ERP-Systeme-im-Vergleich-763963.html

5-FrontAccounting. (06 de 10 de 2014). *FrontAcounting Wiki*. Obtingut de http://frontaccounting.com/fawiki/

6-OpenERP Spain. (2014). *Características técnicas OpenERP*. Obtingut de http://openerpspain.com/openerp/caracteristicas-tecnicas/

7-Odoo. (05 de 06 de 2014). *Odoo Strategy and Roadmap*. Obtingut de http://www.slideshare.net/openobject/keynote02

8- Gnu Health. (30 de 03 de 2015). Gnu Health. Obtingut de http://en.wikibooks.org/wiki/GNU_Health

9- Gnu Health. (2015). Gnu Health The free health and hospital information system. Obtingut de http://health.gnu.org/es/index.html

10- Manual de usuario de Tryton (2014). Obtingut de http://doc.tryton-erp.es/

11- Rehabilitation Measures Database (2014) Ten meters walking test form. Obtingut de www.rehabmeasures.org