Web de seguiment de taques a la pell

Daniel Calvo Ramos

Resum— En aquest projecte s'ha realitzat la implementació d'una pàgina web que serveixi com a web de seguiment de taques a la pell. La pàgina web, permet a pacients pujar imatges de les seves taques a la plataforma. Posteriorment, aquestes imatges són evaluades per una API externa, la qual analitza les imatges i els hi dona un percentatge de malignitat. Acte seguit, aquestes imatges són avaluades per metges qualificats, els quals poden realitzar comentaris sobre les imatges que podran ser vistos pels propis pacients. D'aquesta forma, amb la realització d'aquest projecte s'espera poder ajudar a que aquests tipus de càncer puguin ser detectats de manera precoç i, d'aquesta forma, poder ajudar a que moltes persones puguin ser tractades a temps.

Paraules clau— Melanoma, taques, Software, pàgina web, Laravel, metge, pacient

Abstract— This project has implemented a website that serves as a skin stain tracking website. The website allows patients to upload images of their spots to the platform. These images are evaluated by an external API, which analyzes the images and gives a percentage of malignancy to the image. Then, these images are evaluated by qualified doctors, who are allowed to comment on the images. That comments can be viewed by the patients. In this way, with the realization of this project, it is hoped to be able to help to detect these cancers early and, in this way, to be able to help many people to be treated in time.

Index Terms— Melanoma, spots, Software, website, Larvael, doctor, patient

1 Introducció

Com succeeix amb tots els càncers, una detecció precoç del mateix augmenta en gran mesura la esperança de supervivència dels pacients. Per això, des de la empresa Deep Solutions, i amb col·laboració amb la Generalitat, es proposa realitzar una pàgina web que classifiqui imatges de pigues, taques i lesions a la pell segons la seva perillositat i que doni la probabilitat que té de que sigui un melanoma.

Per tant, l'objectiu del projecte és ajudar a que els càncers de pell de tipus melanoma puguin ser detectats de manera més precoç per tal de poder augmentar la taxa de supervivència dels pacients amb aquesta malaltia.

Amb la realització d'aquest treball de final de grau, s'espera poder aplicar tots els conceptes adquirits al llarg dels 4 anys de carrera i poder consolidar tots els coneixements i habilitats apreses, així com poder tenir una mica de contacte amb la forma de treballar del món laboral gràcies al fet de què el projecte es realitzi en col·laboració amb una empresa externa al grau d'Enginyeria.

- *E-mail de contacte: daniel.calvor@uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Helena Boltà Torrell (Computer Science)*
- *Curs 2021/2022*

La pàgina web resultant de la realització d'aquest treball, serà entregada a l'empresa DeepSolutions, què és el client que ha demanat el projecte. A part de ser clients, també realitzaran la part del projecte relacionada amb xarxes neuronals.

2 Objectius

En aquest apartat es comentaran les motivacions i objectius del treball de final de grau que s'ha realitzat.

Els objectius prinicapls associats a la realització d'aquest projecte són els següents:

- Reduïr la taxa de mortalitat del melanoma.
- Augmentar la precocitat en el diagnòstic del melanoma.
- Afavorir la interacció entre metge i pacient.
- Realitzar una pàgina web funcional que permeti, a partir d'una imatge d'una taca, identificar el percentatge de malignitat de la mateixa i serveixi com a portal d'intercanvi d'informació entre metges i pacients.
- Posar en pràctica totes les tècniques i coneixements adquirits durant aquests anys d'estudi.
- Poder desenvolupar un projecte de Software des de zero seguint el cicle de desenvolupament del Software.
- Poder realitzar una pàgina web de forma funcional i de forma autònoma.

- Millorar la tècnica de treball en equip (en aquest cas amb una empresa externa).
- Poder integrar codi extern al projecte.
- Perfeccionar la tècnica amb el framework web Laravel.

Millorar les habilitats de comunicació amb el client i saber plasmar correctament les seves necessitats .

3 Estat de l'art

La idea de detectar el càncer de pell de tipus melanoma mitjançant algun tipus de tecnologia de reconeixement d'imatges o intel·ligència artificial no és nova. De fet, en aquest apartat s'explicaran projectes o aplicacions que tenen una finalitat semblant:

- Projecte DALEM: El projecte DALEM és un projecte que va ser posat en marxa per la multinacional tecnològica GMV, juntament amb la Fundació de Investigació Biomèdica del Hospital Universitario La Paz de Madrid (FIBHULP). Aquest projecte consta d'una aplicació mòbil què permet enregistrar l'evolució de pigues cutànees, les quals posteriorment podran ser revisades pel metge en el moment en el què es faci una revisió mèdica. D'aquesta forma, el metge pot realitzar un diagnòstic més precís i fiable. [1].

- M-Skin Doctor: m-Skin Doctor és una aplicació mòbil de detecció de melanoma en temps real què utilitza tècniques de processament d'imatges i visió artificial. Aquesta aplicació, està basada en l'algorisme 'Grab cut', el qual és un algorisme basat en vectors. Aquest algorisme s'aplica com una tècnica de classificació basant-se en característiques de la imatge com: la textura, l'àrea o el perímetre. L'aplicació té una taxa de sensibilitat del 75% a 80% i triga uns 15 segons en retornar una resposta [2].

Amb la realització de la pàgina web de seguiment de taques, s'ha volgut integrar en una mateixa aplicació, el sistema d'emmagatzemar d'històric d'una taca, realitzat al projecte DALEM, i l'anàlisi i processament d'imatges amb intel·ligència artificial que proporciona l'aplicació m-Skin Doctor.

A més, la particularitat de la meva pàgina web, és permetre i fomentar la interacció entre metge i pacient, gràcies a que el metge pot veure les imatges pujades pel pacient i retornar *feedback* al segon.

4 Metodologia

En aquest apartat es parlarà de la metodologia que sha utilitzat per a realitzar el desenvolupament del projecte i de quina forma s'ha realitzat el control de versions del mateix.

4.1 Metodologia de desenvolupament del projecte

La metodologia que s'ha seguit per a la realització del projecte ha sigut una metodologia iterativa o incremental basada en petits Sprints de curta duració.

Per a dur a terme el projecte, s'ha dividit el mateix en petites parts o paquets, els quals són els *Sprints* del projecte. Cada *Sprint* ha tingut una duració d'una setmana. Cap al final de cada *Sprint*, s'ha realitzat un *Sprint Review* amb el client de l'empresa per a comprovar que el projecte anava en la direcció correcta i els objectius inicials s'anaven assolint [3] .

S'ha decidit realitzar *Sprints* de curta duració, ja que el que es buscava era desenvolupar el projecte d'una forma altament iterativa i incremental, per tal de facilitar el control del treball realitzat durant cada *Sprint*, així com poder fer més èmfasi en cadascuna de les parts realitzades i augmentar la capacitat de detecció d'errors.

El fet d'estar desenvolupant el projecte seguint aquesta metodologia, fa que es puguin detectar de forma ràpida desviacions en el projecte o requisits no complerts o no adequats. I al realitzar una reunió setmanal d'uns 20 minuts amb el client, també ajuda a que el client estigui implicat i assabentat dels canvis que es van realitzant en el projecte i poder seguir d'una forma més propera l'evolució del mateix.

4.2 Control de versions

El control de versions del projecte s'ha fet utilitzant el software de control de versions GIT i la plataforma GitHub, a on es va crear un repositori del projecte el qual va ser vinculat amb el IDE utilitzat i on s'han anant realitzant pujades per a poder tenir el projecte emmagatzemat al núvol i així poder tenir backups i per a què el codi sigui fàcilment accessible per al client de l'empresa o la tutora.

Referent als informes del projecte, també s'han anant pujats al mateix Github a on s'ha publicat el codi font de la web.

5 Planificació

El treball a realitzar ha tingut una duració de 19 setmanes de principi a fi: del 14 de febrer de 2022 fins al 26 de juny del 2022.

Per a la realització de la planificació, s'han dividit totes les activitats segons les fases del cicle del desenvolupament del Software a la que pertanyen.

Per a la fase d' implementació, què és la més llarga, s'ha dividit el treball a realitzar en paquets o petites unitats de treball, tal i com s'ha comentat anteriorment en l'apartat de metodologia.

A la (figura suplementària A1) de l'annex es pot veure un Diagrama de Gantt que mostra de forma molt visual la planificació del projecte al llarg de les 19 setmanes i la duració esperada de cada activitat del mateix.

6 Disseny

En aquest apartat s'expliquen més en profunditat els aspectes relacionats amb el model relacionat de la base de dades i la arquitectura de Software del projecte.

6.1 Llenguatge i arquitectura Software

El projecte a realitzar s'ha desenvolupat amb *Laravel*, què és un *framework* de codi obert per a desenvolupar aplicacions web amb PHP. El principal avantatge que ofereix aquest *framework* és la capacitat que té de permetre obtenir una gran eficiència en el desenvolupament del codi, evitant el que es coneix com a 'codi *spaghetti*'. Aquest concepte fa referència a aquells programes que tenen una estructura o control de flux massa complexe i difícil d'entendre. Entre d'altres avantatges que té, trobem que ofereix una documentació de qualitat i molt àmplia o que ofereix gran quantitat de llibreries gràcies a les quals es poden implementar multitut de funcionalitats.

El projecte s'ha fet seguint una arquitectura de Software MVC o Model-Vista-Controlador, què separa les dades de la web, la interfície d'usuari i la lògica de control en tres parts ben diferenciades.

Pel que fa a la base de dades, s'ha utilitzat una base de dades de tipus SQL mitjançant l'eina *PHPmyadmin* i com a servei per a aixecar la pàgina web, s'ha fet servir el propi servei que ofereix *Laravel* per a poder veure el desenvolupament del projecte de forma local.

6.2 Model relacional

El model relacional del sistema està format per 5 entitats: usuari, admin, metge, pacient i imatge, dues relacions d'associació: metge-pacient i pacient-imatge i 3 relacions d'herència: pacient,metge i admin són usuaris.

Les imatges estan associades als pacients, què són els que fan les fotos i les pugen. Un pacient pot tenir n imatges associades, ja que es guardarà un històric de les imatges del pacient i, al mateix temps, les imatges només estan associades a un pacient.

En referència a la relació d'associació metge-pacient, cada pacient està associat a un metge i aquesta associació pot canviar si el metge vol. Un metge pot tenir n pacients, però un pacient només pot estar associat a un metge.

A continuació, es pot veure de forma gràfica el model relacional en format taula i les relacions entre les taules [4]:

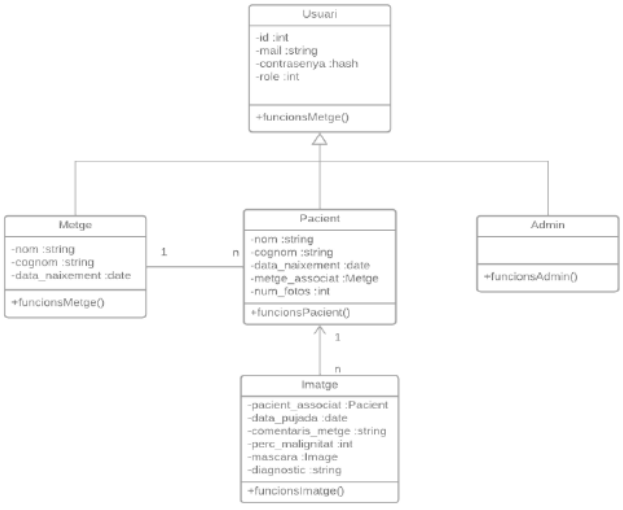

Figura 1. Model relacional del sistema

7 DESENVOLUPAMENT I RESULTATS

Tal i com es pot veure al diagrama de Gannt de la (figura suplementària A1), el projecte es va dividir en sis fases: Planificació, anàlisi, disseny, implementació, testing i entrega. Per tant, per a explicar aquest apartat es dividirà en les diferents fases: planificació, anàlisi, disseny, implementació, testing, entrega [5][6] .

7.1 Fase de planificació

Aquesta primera fase va servir principalment per a conèixer a la persona de l'empresa DeepSolutions amb la que havia de treballar i definir amb ella els abasts i objectius del projecte. També vaig realitzar una primera versió de la planificació del projecte i de com podria anar evolucionant el mateixal llarg de les setmanes. Finalment, es va acordar juntament amb el client, amb quina freqüència es farien les reunions de seguiment i que s'esperava obtenir de la realització d'aquestes reunions [7].

7.2 Fase d' anàlisi

En aquesta fase, es va definir de forma detallada que és el que el client vol que el *Software* a implmenetar faci i quines són les seves necessitats. Posteriormente a la reunió de captació de requeriments, es va fer una llista amb tots els requeriments necessaris per a cobrir les necessitats esmentades pel client. Aquest document va ser posteriorment validat, revisat i aprovat pel client A *la* (figura suplementària A2) es pot veure una taula amb el llistat final dels requeriments del sistema.

7.3 Fase de disseny

Durant aquesta fase, es va realitzar un primer disseny realista del projecte (quin llenguatge es farà servir, quin servidor web s'utilitzarà, com es farà el control de versions del projecte, de quina forma es podrà accedir al codi font del projecte etc...).

7.4 Fase d'implementació

Aquesta fase és la més llarga de tot el projecte i és en aquella on s'ha realitzat tot el desenvolupament del projecte i està dividida en 11 parts o subfases. Cada fase correspon a una divisió de treball o paquets en els quals s'ha dividit el treball.

7.4.1 Definició entorn de treball + control de versions

Primer de tot, s'ha hagut de fer una cerca sobre quin podria ser el IDE que millor es pogués adaptar al projecte. Finalment es va seleccionar Visual Studio Code, degut a la gran quantitat de extensions sobre *Laravel* que té i per la seva facilitat a l'hora d'utilitzar el control de versions. Posteriorment, es va crear el projecte base amb *Laravel* i es va enllaçar el projecte en local amb un repositori de Github, el qual també es va crear.

Finalment, es va definir un servidor *Apache* per a poder visualitzar el resultats de la programació realitzada i una base de dades mySql.

7.4.2 Estructura de dades, creació de models i controladors i establiment de les relacions BD

En aquesta segona subfase s'ha definit l'esquelet relacional de la web. La creació de la estructura de dades, s'ha fet mitjançant migracions, que es la forma que té el framework Laravel per a crear entitats dins una base de dades. Aquestes migracions, consisteixen en definir les taules i els atributs de les mateixes (juntament amb el seu tipus) de forma local, per a que posteriorment aquestes es puguin crear a la base de dades. Un cop definides les taules i els seus atributs, es va fer una migració, què és el procés d'exportar les taules definides localment a la base de dades de totes juntes per a tenir-les disponibles al servidor Apache.

Posteriorment, es van crear els models i els controladors necessaris per a l'esdevenir del projecte. Inicialment, en el projecte es van necessitar 4 models: *UserModel, PacientModel, MetgeModel i ImatgeModel* i 4 controladors, un per cada model. Tal i com es veurà en fases posteriors, el nombre de controladors anirà augmentant.

Referent a les relacions entre taules comentades anteriorment, aquestes no s'han fet a nivell de base de dades, sinó que s'han fet de forma interna mitjançant relacions '*Eloquent*'. Eloquent és la tècnica que utlitza Laravel per a manipular de forma ràpida i senzilla els processos corresponents a l'utilització de bases de dades. D'aquesta forma, establim la relació entre les diferents taules mitjançant una funció ja definida que varia en funció de la relació.

Tal i com podem veure a la figura 3 , la entitat pacient té relacions amb dues entitats. En el cas de la relació amb l'entitat metge, aquesta és 1-N. Per tant, pel que fa al pacient, un pacient només pot estar associat a un metge. Per aquest motiu, utilitzem la funció *belongsTo()* , a la qual li indiquem el model amb el que té la relació i mitjançant quin atribut fa de *foreign key*.

En el cas de la relació entre pacient i imatge, un pacient pot tenir moltes imatges associades, per tant, s'utilitza la funció *hasMany()* indicant també a quin Model fa referència la relació i la clau forànea.

```
public function metge(){
    return $this->belongsTo(Metge::class, 'id');
}
public function imatges(){
    return $this->hasMany(Imatge::class, 'ID_pacient_associat');
}
```

Figura 2. Relacions del model Pacient

7.4.3 definició rols+ login+ creació pacients

A la nostra pàgina web tenim 3 tipus d'usuaris: metges, pacients i admin. Per tant, necessitàvem que, segons el tipus d'usuari amb el que es fes l'inici de sessió, es mostrès una vista o una altra. La forma de fer això ha sigut agrupar les rutes i les vistes segons el rol de cada usuari. Com s'havia de definir el rol de cada usuari, es va crear una nova entitat usuari, la qual agrupava els diferents tipus d'usuari. Aleshores, aquesta entitat usuari tenia un atribut 'rol', el qual podia prendre valors entre 0 i 2 i indicava el rol que tenia l'usuari (rol=0 per als admin, rol=1 per als metges i rol=2 per als pacients).

Per a realitzar el login i la funcionalitat de creació de pacients, es va agafar com a base un *Log In* i un sistema de registre de *Bootstrap*, què és un *framework* de *front-end* que està dissenyat per a facilitar el procés de desenvolupament proporcionant plantilles i components visuals.

A partir d'aquesta base, els mòduls es van anar adaptant d'acord a les nostres necessitats.

El mòdul d'inici de sessió no va patir canvis en aquesta fase perquè a nivell de *front-end* la vista ja era correcta. A nivell de *back-end*, va ser necessari crear 3 *middleware,* un per cada tipus d'usuari, per a poder gestionar les peticions rebudes i indicar que cal fer amb les mateixes segons el tipus de rol de l'usuari que inicii sessió.

Al controlador del *login* es van fer diferents redireccions sobre quina vista mostrar depenent del tipus de rol tingui l'usuari que s'ha logegat. Per a accedir al rol de l'usuari logegat, s'havia d'accedir als atributs de l'usuari que s'havia logegat mitjançant la funció *Auth()* i a partir d'aqui seleccionar el camp rol de l'usuari.

Per a implementar el formulari de creació de pacient, es va utilitzar un formulari de registre base obtingut de *bootstrap* i es va modificar segons els camps que havia de tenir el pacient. Cal destacar que el propi metge era qui definia la contrasenya del pacient, però que aquest la podia canviar al iniciar sessió. S'ha d'afegir que la contrasenya és guardava hasheada a la base de dades mitjançant un *hash bcrypt*, què és el tipus de hash que utilitza *Laravel*.

L'únic camp que el metge no ha d'emplenar i que es realitza de forma automàtica és el camp ID_metge_associat. Aquest camp guarda l'id del metge al que està associat el pacient. Aquest valor serà l'id del metge que ha creat el pacient.

L'obtenció d'aquest id és una mica complexa, ja que no es podia accedir directament a les dades del metge. A l'únic que es podia accedir era a les dades de l'usuari que s'havia logegat. Per tant, per a poder obtenir aquest id, s'havia de una *query* que consistia en fer un *leftJoin* entre les taules users i metges igualant els camps email de les dues taules. Posteriorment, s'establia com a condició que és retornés l'id de les files on el mail de l'usuari que ha iniciat sessió coincidís amb el mail del metge.

D'aquesta forma, podíem obtenir l'id del metge que ha iniciat sessió i guardar-lo a l'atribut corresponent del pacient.

7.4.4 Disseny front-end i creació rutes web

En aquesta subfase es va realitzar un disseny de mitja fidelitat de tot el *front-end* que tindria la web. D'aquesta forma, es podia tenir una idea molt més clara i precisa de totes les finestres i transicions que tindria la web. Aquest disseny es va validar amb el client. Com a resultat de fer el disseny, es va poder definir que la pàgina web tindria 9 finestres o pestanyes.

Posteriorment, es van definir amb Laravel totes les rutes per a fer possible la connexió entre totes les pestanyes així

com afegir botons de 'anar enrere' per a que totes les finestres estiguessin lligades entre si i l'usuari pogués navegar de forma còmoda entre les finestres.

Per a crear una ruta i fer-la activa, primer de tot s'havia de definir la ruta al fitxer de gestió de rutes, posteriorment, el controlador al qual feia referència aquella ruta i finalment la funció que faria servir la ruta. Un cop definida la ruta, al controlador, s'havia de retornar la vista que s'havia de mostrar en aquella ruta.

A la figura següent es pot veure de forma visual com estaven agrupades les rutes i com estaven definides.

Figura 3. Definició rutes en Laravel

7.4.5 Pujar imatges + guardar l'històric del pacient

En aquesta subfase s'ha realitzat el mòdul/funcionalitat per a que un usuari pugui pujar imatges emmagatzemades en local. Per a fer la vista, es va agafar una plantilla base de *bootstrap* i s'ha modificat per a què segueixi el mateix estil que la resta de pàgines de la web.

Abans de pujar la imatge, primerament es comprovava el tipus de l'arxiu que es volia adjuntar, i en cas de no ser una imatge, es tornava a carregar la pàgina de pujar imatge.

Un cop es comprovava que l'arxiu pujat era una imatge, s'agafava el nom de la imatge i es movia la mateixa a la ruta públic/images.

L'últim pas era crear la nova imatge a la base de dades i associar-la al pacient que pujava la imatge. Per a associar-la al pacient, primerament es necessitava l'id d'aquest pacient. Com només es tenien les dades de l'usuari autenticat, s'havia de fer un *leftJoin* entre les taules users i pacients. La query consistia en comparar els camps email de les dues taules, i retornar el camp id de la taula pacient on el mail del pacient i el mail de l'usuari autenticat coincidissin.

Un cop es tenia l'id del pacient, calia fer una query per obtenir el pacient, ja que s'havia d'incrementar el valor del camp 'num_fotos' del pacient, ja que s'havia pujat una nova imatge. Per a fer aquest increment, es va fer una consulta sobre la taula imatges i es va fer un count d'aquelles imatges les quals els seu 'ID_pacient_associat' coincidís amb l'id del pacient. Posteriorment, s'incrementava el valor en una unitat i s'actualitzava el valor a la base de dades.

Per a crear la imatge, simplement es feia un *create* modificant els camps 'ID_pacient_associat', al qual se li posava l'id del pacient que es va trobar abans, el camp 'imatge_pujada', on se li posava el nom de la imatge, i la 'data pujada', la qual contenia la data actual de la pujada. Això es va fer mitjançant la funció *now()* de la llibreria *Carbon*, i posteriorment es va filtrar per a obtenir només el dia, mes i any.

Així les fotos que va pujant el pacient, van conformant el seu històric d'imatges.

7.4.6 Implementació frontend (part metge)

En aquesta sisena subfase s'ha realitzat tota implementació de les vistes principals de la pàgina web referents al rol metge.

Per a fer les vistes, es va utilitzar com a referència el disseny que es va realitzar a la subfase 4. Primer, es parlarà de com funciona l'estructura interna per a mostar les dades.

Inicialment, al fitxer de gestió de rutes, s'havia de crear una ruta, a la qual se li passava el controlador i la funció sobre la qual actuaria. Acte seguit, a la funció del controlador, es feien les operacions corresponents a la funció (fer *queries*, gestionar peticions ...). Un cop realitzada tota la manipulació de dades, es retornava la vista que s'havia de mostrar a la ruta juntament amb les variables que s'havien de mostrar a la vista (les vistes estaven organitzades en diferents carpetes segons a l'usuari al que perntanyien). Un cop explicat això, es procedeix a explicar cadascuna de les vistes de la pàgina web.

El metge, un cop inicia sessió, el primer que veu és una taula dels pacients que té assignats tal i com es pot veure a la figura 4.

Per a poder realitzar aquesta taula, es van agafar els pacients associats al metge que ha iniciat sessió. Per a poder trobar els pacients associats a un metge, es va utilitzar la mateixa consulta explicada a l'apartat 7.4.3. Un cop teniem el metge, mitjançant les relacions Eloquent que es van realitzar a la fase 7.4.2, es podien trobar tots els pacients associats a un metge concret. Per tant, l'últim pas va ser realitzar un for per a recórrer tots els pacients associats al metge que s'ha logegat. De cada pacient associat es mostrava l' id, nom, cognom, data naixement i num. fotos. Addicionlament, també hi havia un botó per a poder veure els detalls del pacient concret. A la figura 4 es pot veure de forma visual el resultat final de la vista.

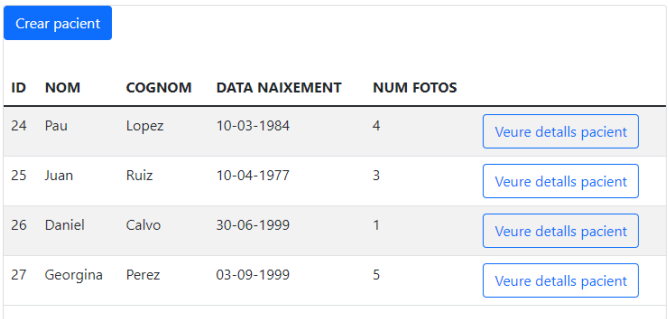

Figura 4. Vista inicial metge

Per a cada pacient, el metge pot veure l'històric d'imatges que té un pacient. (Figura 5).

Figura 5. Històric d'imatges d'un pacient

Per a realitzar aquesta vista, es va seguir una metodologia d'implementació semblant a l'anterior: s'obtenia el pacient gràcies a l'id, i amb les relacions *Eloquent* s'obtenien totes les imatges associades al pacient i es mostraven. Com la vista ha de ser un històric, les imatges es mostraven en ordre invers, és a dir, les imatges més recents es mostraven al principi de tot. De cada imatge, es mostrava la data, la imatge original, i la imatge amb màscara, què no deixa de ser la imatge original amb el contorn de la taca ressaltat.

Al veure el detall de la imatge d'un pacient, el metge podia veure les dues imatges àmpliades i en una mida més gran per a que les pogués comparar més fàcilment. Addicionalment, podia veure altres camps com la data de pujada, un camp de comentaris on el que el metge escrigui és el que veurà el pacient, un camp de percentatge de malignitat i un camp de diagnòstic. (figura 6).

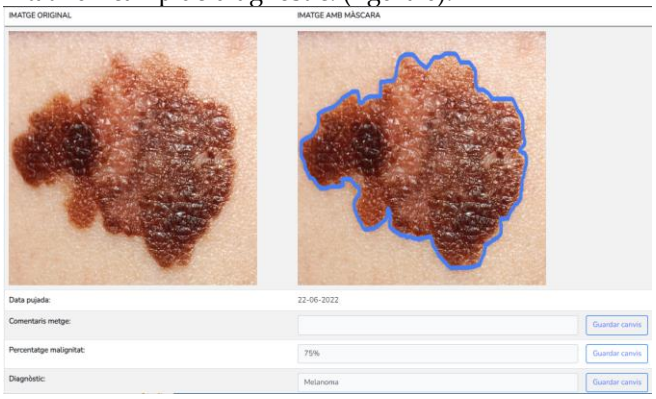

Figura 6. Detalls d'una imatge (vista metge)

7.4.7 Implementació frontend (part pacient)

En el cas del pacient, l'històric d'imatges que veu és el mateix que pot veure el metge, però la diferència està a la vista del detall d'una imatge. Mentre que el metge pot veure els valors dels camps que retorna l'API externa, el pacient només podrà veure les dues imatges, la data de pujada i el camp de comentaris del metge. Això està realitzat d'aquesta manera perquè si un pacient veu un percentatge o un diagnòstic de 'Melanoma' o 'alta probabilitat de melanoma' pot entrar en pànic. D'aquesta manera, el metge li pot comentar: 'passa't per la consulta' i d'aquesta forma la situació no serà tan alarmant pel pacient *(figura 7)*.

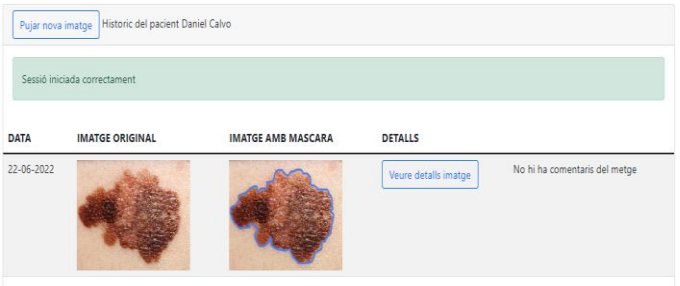

Figura 7. Vista inicial d'un pacient

Figura 8. Detalls d'una imatge (vista pacient)

7.4.8 Canvi contrasenya

Per a realitzar el mòdul de canvi de contrasenya, es va utilitzar un formulari de tres camps: contrasenya actual, nova contrasenya i nova contrasenya repetida.

Per a la realització de la part de *back-end*, primer de tot es va validar que els valors del tres camps tenien un mínim de 6 caràcters i un màxim de 100.

Posteriorment, es comprovava que el contingut del camp 'Repeteix nova contrasenya' era igual al contingut del camp 'Nova contrasenya'. En cas de coincidir, el següent pas era comprovar que la contrasenya actual coincidia amb la contrasenya original de l'usuari. Per a fer aquesta comprovació, s'havia de recuperar la contrasenya actual de l'usuari. Això es podia fer fàcilment recuperant l'usuari amb el que s'havia fet *login* i obtenir-ne la contrasenya. Com aquesta estava xifrada, el que es va fer és comprovar que els *hash* de la contrasenya de l'usuari logegat i del camp 'Contrasenya actual' coincidien. Si coiniciden, s'actualitzava la contrasenya a la base de dades. En cas contrari, es feia un redirect a la mateixa pàgina.

7.4.9 Canviar metge associat i modificació dades imatge

En aquesta subfase s'han realitzat les funcionalitats per a permetre què un metge pugui canviar el metge d'un pacient que ell tingui associat i s'ha fet la funcionalitat per a què un metge pugui modificar les dades d'una imatge que un pacient hagi pujat.

Per a implementar la funcionalitat de canvi de metge associat d'un determinat pacient, a nivell visual es mostrava un desplegable amb tots els metges registrats a la base de dades a la part superior esquerra de la pantalla un cop el metge es trobava a l'històric d'un determinat pacient.

A nivell intern, un cop el metge seleccionava un nou metge i confirmava els canvis, s'enviava una petició al controlador la qual contenia l'id del metge seleccionat.

A continuació, calia recuperar el pacient i modificar el seu metge associat. Per a simplificar el procés d'obtenció del pacient, també es passava l'id de l'usuari a la petició, d'aquesta forma, simplement accedint al valor pacient_id de la petició s'obtenia el pacient. El següent que s'havia de fer era recuperar el pacient en qüestió, actualitzar el valor del camp ID_metge_associat i guardar els canvis.

Referent a la funcionalitat de permetre a un metge modificar les dades d'una imatge d'un pacient, es va implementar un petit formulari a la pestanya de detalls d'una imatge. Aleshores, el metge podia modificar qualsevol dels valors associats a aquella imatge. Cal recordar que aquests valors venien definits inicialment per l'output de l'API externa. És a dir, l'API externa retornava uns valors (percentatge de malignitat i diagnòstic), però aquests valors podien (o no) ser correctes. Per això, un metge ha de poder modificar aquestes dades simplement actualitzant les dades de la imatge.

Internament, es recuperava la imatge mitjançant el seu id, el qual es podia obtenir gràcies a la petició, i un cop es tenia la imatge sobre la qual s'havien d'actualitzar els valors, s'igualaven els valors actuals al valors obtinguts per la petició, que són els valors introduïts pel metge al formulari.

7.4.10 Obtenir dades API externa

Inicialment, el projecte estava pensat per a que es connectés amb una API externa real que havia de crear l'empresa DeepSolutions. Aquesta API externa, havia de rebre una imatge, analitzar-la mitjançant l'ús de xarxes neuronals, i retornar un JSON que contenia una url per a poder descarregar la mateixa imatge amb el contorn de la imatge original pintat, un percentatge de malignitat, i un diagnòstic.

Com que en el moment que havia d'implementar aquesta part, l'empresa no tenia l'API finalitzada, em van muntar una API *'fake'* amb l'eina *nodeRed*, la qual servia per simular les crides i les peticions i per a que jo pogués seguir desenvolupant l'aplicació. Aquesta API *'fake'* la diferència que tenia respecte la final, era que no analitzava la imatge: simplement retornava una imatge amb cotorn aleatòria d'entre unes quantes que tenia emmagatzemades, i segons la imatge que sortia, definia un percentatge de malignitat i un diagnòstic aleatoris. La resta de funcionalitats, tant el paràmetre d'entrada (una imatge) com les peticions o el format de les respostes era el mateix. Per tant, el projecte ha estat desenvolupat sobre aquesta API.

L'obtenció de dades es va realitzar en dues fases: primer s'havia de fer una primera petició a l'API enviant la imatge que el pacient acabava de pujar, i després s'havia de rebre el JSON amb les dades proporcionades per l'API. Per a fer la petició, es va utilizar la llibreria Http, gràcies a la qual podem fer peticions d'una forma intuïtiva. Per a afegir la imatge com a cos de la petició, s'ha hagut de convertir la imatge a binari i posteriorment utilitzat una funció de la mateixa llibreria per a permetre enviar cos en la petició. El resultat de la petició el guardàvem en una variable, la qual contenia la resposta de la petició.

El seguent pas era obtenir la resposta de la petició que es va fer anteriorment. Per fer-ho es va accedir al JSON de la resposta, que era la part de la petició on es trobava la data. Per poder accedir a un camp concret del json s'havia de fer posant com a key el nom de l'atribut al que es volia accedir.

7.4.11 Processar dades API externa

El processament de dades, es va realitzar en dues fases: primer s'havia d' accedir a la url proporcionada i descarregar-se la imatge i emmagatzemar-la, i com a últim pas, mostrar totes les dades de la imatge (només les pot veure el metge). Cal afegir que totes aquests passos es van fer dins la funció utilitzada per a pujar imatges.

Un cop teniem accés a les dades que ens interessaven, el següent pas era actualitzar la imatge amb les dades rebudes. Per fer-ho, es va fer una consulta filtrant per l'id de la imatge (el qual teniem del moment en el que es va fer la pujada de la mateixa imatge),i es van igualar els valors del atributs 'percentatge_malignitat' i 'diagnòstic' als valors rebuts en la resposta de la petició.

Finalment, calia obtenir la nova imatge, guardar-la en local i a la base de dades. El primer que s'havia de fer era obtenir el link de descàrrega,el qual es podia obtenir de la mateixa resposta anterior. Posteriorment, ens havíem de guardar el contingut de l'enllaç, que es va fer mitjançant la funció *file_get_contents()*. D'aquesta forma, podíem obtenir el contingut de la url. A continuació, s'havia de guardar el nom de la màscara (la imatge amb el contorn la qual es descarregava de l'enllaç). La nova imatge s'havia de guardar amb el nom de la imatge i l'extensió. Per agafar només aquest nom s'agafava tota la url i s'utilitzaven les barres (/) per a poder eliminar la resta de la url. Un cop es tenia el nom, s'afegia al principi del nom l'id de la imatge. D'aquesta forma, el nom de la màscara era: id+nom. Això es feia per a que el nom de la imatge original i la imatge màscara no es repetissin. Un cop fet aquest últim pas, es guardava el nom de la imatge a l'atribut 'màscara' de la base de dades .

Per a emmagatzemar la imatge en local, primer de tot s'havia de modificar, a l'arxiu de configuració, la ruta on emmagatzemar les imatges guardades per a que aquesten es guardessin a la ruta public/images. Posteriorment, es guardava la imatge a disc a la ruta definida anteriorment

7.4.12 Revisió i neteja del codi

En aquesta última subase de la fase d'implementament, es va revisar el codi en busca de variables amb noms poc explicatius, així com corregir petits aspectes visuals de la pàgina.

Per altra banda, es van substituir els valors numèrics estàtics utilitzats per variables globals (definides al fitxer de configuració .env). D'aquesta forma, en cas de voler modificar algun valor, només cal modificar-lo un cop a l'arxiu de configuració. Les variables a les quals se'ls va realitzar

aquest canvi van ser: les variables que definien l'alçada i l'amplada de les imatges, i les variables associades als rols (0 pel rol admin, 1 pel rol metge, 2 pel rol pacient).

7.5 Testing

Per a la realització del testing, primer de tot es va definir un pla de test, identificant els mòduls que s'avaluaran al sistema i quines proves es realitzaran. Posteriorment es va realitzar un document de testing en format Excel, en el qual per a cada mòdul avaluat, es podia trobar el cas que s'estava comprovant, una petita descripció del test realitzat, el resultat esperat de cada test, el resultat obtingut i un apartat de comentaris per si calia fer alguna aclaració. A la (figura suplementària A3.1) es pot veure els tests del mòdul de rutes i a la (figura suplementària A3.2) es poden veure les proves de sistema.

Al projecte s'han realitzat tres tipus de proves de test: proves d'integració, proves de sistema i proves d'acceptació.

- Proves d'integració: En aquesta primera fase de testeig, s'han agafat aquells mòduls o components del projecte en els quals hi ha interacció amb l'usuari, ja que són els components més crítics del projecte degut a la mateixa interacció. Els mòduls triats van ser els següents: *login*, crear pacient, canvi contrasenya, canvi metge associat, pujar imatge i modificar dades imatge juntament amb les rutes i privilegis, que s'ha considerat un altre mòdul, i es va testejat el seu funcionament per separat. Per fer-ho, es van pensat diferents escenaris per a intentar que el sistema falli i donés error. Tots els escenaris i resultats es van documentar en un excel indicant l'escenari triat, una petita descripció, el resultat esperat, el resultat obtingut i algun comentari si cal. Aquestes proves són considerades d'integració perquè en totes les proves fa falta un altre mòdul, que és la base de dades.
- Proves de sistema: Amb aquest tipus de prova es buscava realitzar proves que integressin tots els mòduls del projecte i testejar el mateix com a un tot. Per a fer aquest *testing*¸ es va definir un flux d'execució que involucrés tots el mòduls de la web. D'aquesta forma s'avaluava com els diferents components de l'aplicació interactuaven entre ells en el sistema de forma integrada.
- Proves d'acceptació: Un cop realitzades les proves anteriors, es va enviar al client el document excel amb les proves realitzades, per a que pogués veure quins testos es van realitzar. L'objectiu d'aquesta fase era que el propi client pogués provar l'aplicació i detectar possibles errors o incongruències.

7.6 Entrega

En aquesta última fase, es va comprovar amb el client que s'havien assolit els objectius inicials dels projecte, se li va entregar tota la documentació referent al projecte i es va donar el mateix per finalitzat.

8 Conclusió

En aquest projecte s'ha realitzat la implementació d'una pàgina web per a realitzar un seguiment de les taques de pacients. Durant la realització d'aquest projecte, he pogut millorar les meves tècniques de programació, de treball en equip i les meves habilitats comunicatives. També he pogut aplicar moltes de les tècniques i coneixements adquirits durant els quatre anys de carrera. Pel que fa a les assignatures de la carrera, les que més utils m'han sigut per al desenvolupament del projecte han sigut les assignatures de bases de dades (per a poder dissenyar un model relacional i fer consultes a una base de dades), tecnologies de desenvolupament per a Internet i web (per a conèixer l'estructura interna de les pàgines web i com dissenyar-les i implementar-les de forma eficient, requisits del Software (per a comprendre totes les fases de desenvolupament d'un projecte de Software des de l'inici fins a l'entrega) i laboratori integrat del Software (per haver realitzat un projecte real des de zero i per haver adquirit habilitats comunicatives i de treball en equip).

Referent als objectius i requeriments inicials del sistema, tots ells s'han assolit dins del temps esperat i de la forma esperada pel client.

Agraïments

Agraeixo a la meva tutora, Helena Boltà Torrell, per la seva bona tutoria del TFG al llarg de tot aquest semestre, i per maximitzar el meu rendiment al llarg de la realització del treball i motivar-me a fer una bona feina.

Per altra banda, també m'agradara agrair a l'empresa Deep Solutions i en concret a David Pérez, per la seva tutorització a nivell d'empresa externa i pel seu tracte sempre cordial i molt amable amb mi.

Bibliografia

[1] TEDAE. 2022. La aplicación de tecnología espacial en la vigilancia de lesiones cutáneas pigmentadas facilitará la detección precoz de melanomes Recuperat de: https://www.tedae.org/es/noticias/la-aplicacion-de-tecnologia-espacial-en-la-vigilancia-de-lesiones-cutaneas-pigmentadas-facilitara-la-deteccion-precoz-de-melanomas [Accedit Juny 2022].

[2] Ieeexplore.ieee.org. 2020. An app to detect melanoma using deep learning: An approach to handle imbalanced data based on evolutionary algorithms. [online] Recuperat de : https://ieeexplore.ieee.org/abstract/document/9207552 [Accedit Juny 2022].

[3] Proyectos Ágiles. 2022. Ejecución de la iteración (Sprint). Recuperat de: https://proyectosagiles.org/ejecucion-iteracion-sprint [Accedit Febrer 2022].

[4] LucidChart. 2008. Software de Diagramas Online. [online] Recuperat de: https://lucid.app [Accedit Febrer 2022].

[5] Mercado Ramos, Victor Hugo. Revista de Investigación, Desarrollo e Innovación. Editorial UPTC,2015. Trobat a: https://revistas.uptc.edu.co/ [Accedit febrer 2022]

[6] Concepción, Pedro. Planificación de proyectos de Software. [online] Recuperat de: http://www.cs.umss.edu.bo/ [Accedit febrer 2022]

[7] Monreal, Carlos. (2018). Ciclos de vida iterativo e incremental, ¿Qué son? [online] Recuperat de: https://www.cursodirecciónproyectos.com/ [Accedit febrer 2022]

[8] Cicle de vida del Software 2020: Tot el que necessites saber. [online] Recuperat de: https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber [Accedit febrer 2022]

APÈNDIX

A1. DIAGRAMA DE GANTT

	Número setmana			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
FASE	**Activitat**	**Inici activitat**	**Final activitat**	14-feb	21-feb	28-feb	07-mar	14-mar	21-mar	28-mar	04-abr	11-abr	18-abr	25-abr	02-may	09-may	16-may	23-may	30-may	06-jun	13-jun	20-jun
Planificació	Planificació del projecte	14/02/2022	06/03/2022																			
Anàlisi	Captació requisits	14/02/2022	06/03/2022																			
Disseny	Disseny general	14/02/2022	06/03/2022																			
Implementació	Definició entorn de treball + control de versions	07/03/2022	13/03/2022																			
Implementació	Estructura de dades i establiment relacions bd	14/03/2022	20/03/2022																			
Implementació	Definició rols + Login + creació pacients	21/03/2022	27/03/2022																			
Implementació	Disseny frontend i creació rutes	28/03/2022	03/04/2022																			
Implementació	Pujar imatges + guardar històric pacient	04/04/2022	10/04/2022																			
Implementació	Implementació frontend (part metge)	18/04/2022	24/04/2022																			
Implementació	Implementació frontend (part pacient)	25/04/2022	01/05/2022																			
Implementació	Canvi contrasenya	02/05/2022	08/05/2022																			
Implementació	Canviar metge associat i modificació dades imatge	09/05/2022	15/05/2022																			
Implementació	Obtenir dades API externa	16/05/2022	22/05/2022																			
Implementació	Processar dades API externa	23/05/2022	29/05/2022																			
Implementació	Revisió i neteja del codi	06/06/2022	12/06/2022																			
Testing	Proves d'integració, de sistema i d'acceptació	13/06/2022	19/06/2022																			
Entrega	Entrega del projecte	20/06/2022	26/06/2022																			

A2. LLISTAT DE REQUERIMENTS DEL SISTEMA

A2.1 Requeriments funcionals del sistema

REQUERIMENTS FUNCIONALS DEL SISTEMA	
Identificador	**Descripció del requeriment**
RF-01	El sistema ha de permetre la generació d'un històric de la evolució d'una taca d'un pacient.
RF-02	El sistema ha de permetre al pacient pujar imatges de taques.
RF-03	El sistema ha de permetre que els metges puguin modificar els camps d'una imatge.
RF-04	El sistema ha de proveir als metges la capacitat de crear nous pacients.
RF-05	El sistema ha de poder mostrar outputs diferents de dades segons el tipus d'usuari.
RF-06	El sistema ha de permetre al metge poder comparar les fotos d'un mateix pacient de forma clara.
RF-07	El sistema ha de permetre poder connectar-se a la API externa.
RF-08	El sistema ha de permetre obtenir dades d'una API externa i processar-les.
RF-09	El sistema ha de poder enviar fotos a la API externa.
RF-10	El sistema ha de permetre als metges navegar entre els diferents pacients.
RF-11	El sistema haurà de permetre mostrar l'evolució de les taques.
RF-12	El sistema ha de proveir als metges la capacitat de canviar el metge associat a un dels seus pacients.
RF-13	El sistema ha de proveir als pacients la capacitat de canviar la seva contrasenya un cop han iniciat sessió.
RF-14	El sistema ha de tenir tres tipus d'usuari: pacient, metge i admin.
RF-15	El sistema ha de permetre l'accés al sistema només al usuaris que estiguin registrats.
RF-16	El sistema ha d'assegurar que les dades estan protegides de l'accés no autoritzat.

A2.2 Requeriments no funcionals del sistema

REQUERIMENTS NO FUNCIONALS DEL SISTEMA	
Identificador	**Descripció del requeriment**
RNF-01	El sistema ha d'estar realitzar amb el *framework* Laravel.
RNF-02	L'aplicació web ha de tenir un disseny *Responsive*.
RNF-03	Les contrasenyes dels usuaris han d'estar xifrades amb un Hash.
RNF-04	El sistema ha de ser desenvolupat utilitzant l'IDE Visual Studio Code
RNF-05	El testing del sistema ha de ser realitzar de forma manual.
RNF-06	L'aplicació web ha de ser compatible amb Windows 10
RNF-07	El sistema ha d'estar disponible en català
RNF-08	Els procediments del sistema han de complir amb les lleis i reglaments de protecció de dades mèdiques

A3. EXEMPLES DEL PLA DE TEST

A3.1 Test d'integració del mòdul 'crear pacient'

Casos	Descripció	Resultat esperat	Resultat obtingut
Dades correctes	Es crea un pacient de forma correcta	El pacient s'ha creat correctament i es mostra a la llista de pacients del metge	El pacient s'ha creat correctament i es mostra a la llista de pacients del metge
contrasenyes diferents	S'introdueix contrasenyes diferents als camps contrasenya i confi	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient
fields buits	s'intenta crear un pacient sense emplenar cap field	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient
email ja existent	s'introdueix un mail ja existent a la BD	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient	El pacient no s'ha creat i es rediregeix al formulari de creacio del pacient

A3.2 Proves de sistema

Flux d'execució	Estat esperat	Estat obtingut
1. Un metge inicia sessió	el metge accedeix al llistat de pacients que té associats	el metge accedeix al llistat de pacients que té associats
2. El metge crea un pacient	El nou pacient creat apareix a la llista de pacients del metge	El nou pacient creat apareix a la llista de pacients del metge
3. El pacient inicia sessió	el pacient accedeix a l'històric d'imatges (ha d'estar buit)	el pacient accedeix a l'històric d'imatges (ha d'estar buit)
4. El pacient modifica la seva contrasenya	La contrasenya del pacient s'ha modificat a la taula usuari	La contrasenya del pacient s'ha modificat a la taula usuari
5. El pacient puja una imatge	El pacient veu la imatge pujada al seu històric juntament amb la màscara	El pacient veu la imatge pujada al seu històric juntament amb la màscara
6. El metge modifica els camps de la imatge i afegeix un comentari	Els valors de la imatge s'han modificat segons els valors que ha introduït el metge	Els valors de la imatge s'han modificat segons els valors que ha introduït el metge
7. El pacient veu els comentaris que el metge ha fet a la imatge que	El pacient veu al històric que la imatge té un comentari del metge i al fer clic a la imatge veu	El pacient veu al històric que la imatge té un comentari del metge i al fer clic a la imatge veu el comentari
8. El metge canvia el metge associat del pacient	El pacient ja no apareix a la llista de pacients del metge (a la BD s'ha modificat el atribut ID_	El pacient ja no apareix a la llista de pacients del metge (a la BD s'ha modificat el atribut ID_pacient_associat