Universitat Oberta de Catalunya

uoc.edu

Avaluació de projectes Snap! via web mitjançant SnapTools

Carlos-Cristian Arnau Ros
Grau Enginyeria Informàtica
Desenvolupament web

Gregorio Robles Martínez
Santi Caballe Llobet

12/06/2020

Aquesta obra està subjecta a una llicència de
Reconeixement-NoComercial-CompartirIgual
3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL

Títol del treball:	*Avaluació de projectes Snap! via web mitjançant SnapTools*
Nom de l'autor:	*Carlos-Cristian Arnau Ros*
Nom del consultor/a:	*Gregorio Robles Martínez*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*06/2020*
Titulació o programa:	*Grau Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament web*
Idioma del treball:	*Català*
Paraules clau	*Aprenentatge computacional, Snap!, desenvolupament web*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Tot i que existeixen plataformes online que permeten millorar les habilitats relacionades amb aprenentatge computacional, cap d'elles ofereix de manera nativa una retroalimentació respecte a la qualitat dels projectes dels seus usuaris.

Aquest fet que ha desencadenat en l'aparició de plataformes externes capaces d'importar i processar aquests tipus projectes per avaluar-los en diferents categories. El benefici per els usuaris és que aporten una manera de poder comprovar la correctesa de la programació, faciliten l'aprenentatge dels error propis i milloren el pensament computacional.

Aquest treball final de grau ha estat enfocat a la creació d'un portal web on els usuaris puguin avaluar projectes de la plataforma Snap!, omplint el buit que existeix actualment atès que no existeix cap utilitat en línia que ofereixi aquest servei.

Aquest plataforma escaneja el projecte i obté una puntuació per a cada una de les sis àrees avaluades (lògica, representació de dades, paral·lelisme, sincronització, interactivitat de l'usuari i control de flux), mostrant una pàgina a l'usuari amb el resultat obtingut a totes les proves, en la que pot veure també com ha de millorar per obtenir una puntuació major.

Per a la seva elaboració es va seguir una metodologia en cascada, on es van aplicar i posar en pràctica molts dels coneixements apresos durant la realització del grau, posant una menció especial als continguts del itinerari de programari.

El resultat ha estat la publicació d'una plataforma online disponible a través d'Internet, on els usuaris poden registrar-se i avaluar qualsevol projecte de la plataforma Snap!, obtenint una retroalimentació en diferents àrees del aprenentatge computacional.

Abstract (in English, 250 words or less):

Although there are online platforms that allow for the improvement of computer learning skills, none of them offer native feedback on the quality of their users' projects.

This fact has triggered the emergence of external platforms capable of importing and processing these types of projects to evaluate them in different categories. The benefit for the users is that they provide a way to check the validity of the programming, facilitate the learning of their own errors and improve the computational thinking.

This final degree work has been focused on the creation of a web portal where users can evaluate Snap! platform projects, filling the gap that exists at present since there is no online utility that offers this service.

This platform scans the project and obtains a score for each of the six areas evaluated (logic, data representation, parallelism, synchronization, user interactivity and flow control), showing a page to the user with the result obtained in the tests, where he can also see how he has to improve to obtain a higher score.

For its elaboration, a cascade methodology was followed, where many of the knowledge learned during the course of the degree was applied and put into practice, with a special mention to the contents of the software itinerary.

The result has been the publication of an online platform available through the Internet, where users can register and evaluate any project of the Snap! platform, getting feedback in different areas of computer learning.

Índex

1. Introducció 1
1.1 Context i justificació del Treball 1
1.2 Objectius del Treball 1
1.2.1 Principals 1
1.2.2 Secundaris 2
1.3 Enfocament i mètode seguit 2
1.4 Planificació del Treball 4
1.5 Recursos 5
1.6 Breu sumari de productes obtinguts 5
1.7 Breu descripció dels altres capítols de la memòria 6
2. Resta de capítols 7
2.1 Infraestructura 7
2.1.1 Circuit CI / CD 7
2.1.2 GitHub 8
2.1.3 GitHub Actions 9
2.1.4 Entorns 10
2.1.4.1 Desenvolupament 10
2.1.4.2 Producció 12
2.1.4.2.1 Contractació 12
2.1.4.2.2 Instal·lació 12
2.1.4.2.3 Ubuntu Firewall 13
2.1.4.2.4 Certbot 13
2.1.4.2.5 Nginx 14
2.1.4.2.6 Base de dades 15
2.1.4.2.7 Aplicació 15
2.1.4.2.7.1 Frontend 15
2.1.4.2.7.2 Backend 15
2.2 Frontend 16
2.2.1 Introducció 16
2.2.2 Desenvolupament 16
2.2.2.1 Estructura 17
2.2.2.2 Interacció amb backend 18
2.2.2.3 JavaScript 18
2.2.2.3 Emmagatzemat de dades 19
2.2.2.4 JWT 20
2.2.2.5 Validació client 20
2.2.2.6.Interfície web 20
2.2.2.6.1 Portada 20
2.2.2.6.2 Autentificació 21
2.2.2.6.3 Perfil 21
2.2.2.6.4 Projectes 21
2.2.2.6.5 Dashboard 22
2.3 Backend 23
2.3.1 Arquitectura d'alt nivell 23
2.3.2 Arquitectura de sistema 24

2.3.3 Arquitectura interna 24
2.3.4 Estructura dels fitxers del projecte 25
2.3.5 Estructura del codi font 26
2.3.5.1 Contexts 26
2.3.5.1.1 Authentication 26
2.3.5.1.1.1 Interfície HTTP 26
2.3.5.1.2 Dashboard 26
2.3.5.1.2.1 Interfície *HTTP* 26
2.3.5.1.3 Mail 27
2.3.5.1.4 Profile 27
2.3.5.1.4.1 Interfície *HTTP* 27
2.3.5.1.5.1 Interfície *HTTP* 27
2.3.5.1.6 Shared 27
2.3.5.1.7 Snap 27
2.3.6 Seguretat 27
2.3.6.1 JWT 27
2.3.6.2 Tallafocs 29
2.3.6.3 CORS 29
2.3.6.4 Rols 30
2.3.7 Components de *Symfony* 30
2.3.7.1 Missatger 30
2.3.7.2 Serialitzador 31
2.3.7.3 Swift Mailer 32
2.3.7.4 Expedidor d'esdeveniments 33
2.3.8 Processador projectes 34
2.3.9 Interfície de consola 35
2.3.9.1 Afegir administrador 35
2.3.9.2 Treure administrador 35
2.3.10 Proves unitàries 36
2.3.11 Base de dades 37
2.3.11.1 Model conceptual 37
2.3.11.2 Entitat Relació 37
2.3.11.3 Doctrine Migrations 38
2.4 Aplicació 39
2.4.1 Casos d´ús 39
2.4.2 Mètriques Snap! 39
2.4.2.1 Lògica 39
2.4.2.2 Representació de dades 40
2.4.2.3 Paral·lelisme 40
2.4.2.4 Sincronització 40
2.4.2.5 Interactivitat de l'usuari 40
2.4.2.6 Control de flux 41
3. Conclusions 42
4. Glossari 44
5. Bibliografia 45

1. Introducció

1.1 Context i justificació del Treball

Davant la necessitat de preparar als estudiants en les habilitats que seran de profit en el seu futur professional, els sistemes educatius d'arreu del món han començat a incloure programes que esperen millorar la capacitat de pensament computacional dels seus alumnes.

L'aprenentatge computacional ens permet la resolució de problemes a través d'un pensament analític i metòdic, que mitjançant l'execució de diferents tasques en un format semblant a un algoritme, s'apropa més a com solucionen les màquines els problemes que no a com ho fan les persones.

Existeixen diverses plataformes online que permeten millorar les habilitats esmentades mitjançant la creació de programes interactius, a través de llenguatges de programació basats en blocs que només requereixen d'un navegador web.

Malauradament aquestes plataformes no ofereixen la possibilitat d'avaluar els projectes creats per els seus usuaris, tot i que existeixen serveis externs que compleixen aquesta funcionalitat per algunes plataformes.

Actualment no existeix cap plataforma que ofereixi l'avaluació de projectes per a la plataforma *Snap!* [19], i per tant existeix una demanda de l'esmentat servei.

Aquest treball de grau es centra en una plataforma d'avaluació de projectes, per a que estudiants i professors disposin d'una eina amb la que poder valorar la correctesa dels seus projectes.

1.2 Objectius del Treball

L'objectiu general d'aquest treball és el desenvolupament d'una aplicació web que permeti avaluar projectes creats amb la eina *Snap!* [19].

1.2.1 Principals

- Crear un sistema d'autentificació que permeti als usuaris de la plataforma registrar-se, entrar al portal mitjançant usuari i contrasenya i recordar les credencials per correu en cas d'oblit.

- Disposar d'una pàgina privada d'usuari on pugui llistar i revisar els projectes avaluats fins al moment.

- Confeccionar una pàgina privada d'usuari on pugui ser capaç de canviar les seves dades personals i contrasenya.

- Desenvolupar un sistema capaç d'avaluar numèricament diferents apartats dels projectes que proporcionen els usuaris.

- Oferir un àrea privada a l'administrador que mostri diferents mètriques de la plataforma.

- Publicar una pàgina web allotjada en un servidor accessible des de Internet capaç de servir la plataforma de manera segura.

1.2.2 Secundaris

- Posar en pràctica els coneixements adquirits en el transcurs del grau.

- Utilitzar eines desconegudes per realitzar el desenvolupament del projecte amb l'objectiu de sortir de la zona de confort.

- Adquirir coneixement sobre l'aprenentatge computacional i les eines disponibles actualment que ajuden a la comunitat

- Aprofundir en el coneixement de diferents àrees del desenvolupament web i que fan possible la posada en marxa i manteniment del mateix.

- Aprendre conceptes d'arquitectura avançats així com de l'enfocament del disseny basat en el domini.

- Creació d'un portal web on els usuaris puguin obtenir retroalimentació sobre els seus projectes.

1.3 Enfocament i mètode seguit

Durant el procés de desenvolupament d'aquest treball final de grau, els estudiants hem hagut de seguir una avaluació contínua que ha comportat diverses entregues prèvies del treball realitzat a cada etapa.

Per aquest motiu s'ha cregut convenient realitzar aquest projecte seguint el model en cascada, atès que la confecció del programari està conformada per un conjunt d'etapes que s'executen una darrera l'altre de manera seqüencial, fet que es solapa amb el requerit per l'avaluació.

En un futur un cop finalitzat el treball final de grau pot ser convenient tornar a avaluar quina metodologia pot ser més adient, tenint en compte que és força habitual avui en dia en projectes de llarga durada desenvolupar sota alguna metodologia àgil com SCRUM.

Passem per tant a descriure els aspectes claus de la metodologia que s'ha fet servir per a desenvolupar aquest projecte.

Es realitza un anàlisis de les necessitats del projecte a desenvolupar i s'especifica tot el que ha de fer el sistema sense entrar en detalls tècnics. També és l'etapa on es descriuen els requisits del programari a desenvolupar.

Es descriu l'estructura interna del programari i les relacions entre les entitats que el componen, així com l'especificació del que deu fer cadascuna de les seves parts.

En aquesta fase es programen els requisits especificats fent ús de les estructures de dades dissenyades en la fase anterior, obtenint com a resultat un programa que executa els passos necessaris per a resoldre el problema dissenyat.

Un cop s'obté el codi font s'ha de verificar que compleix els requisits inicials, que els components funcionen de la forma esperada i que no contenen cap tipus d'error. Aquesta etapa es pot automatitzar amb proves unitàries i funcionals.

Es tracta de posar l'aplicació a funcionar en el sistema final i comprovar que tot funciona com s'espera.

Aquesta etapa consisteix en mantenir el valor del programari a través del temps, mitjançant nous requisits, resolent errors o modificant l'aspecte visual.

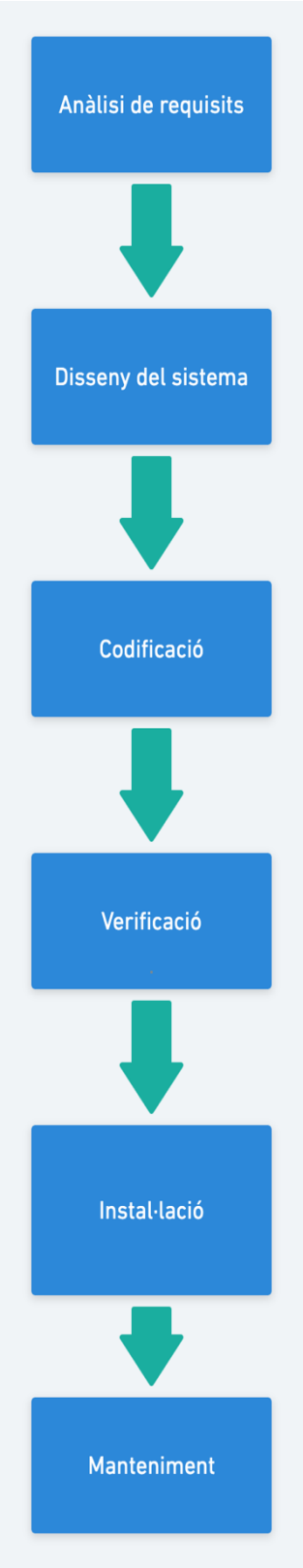
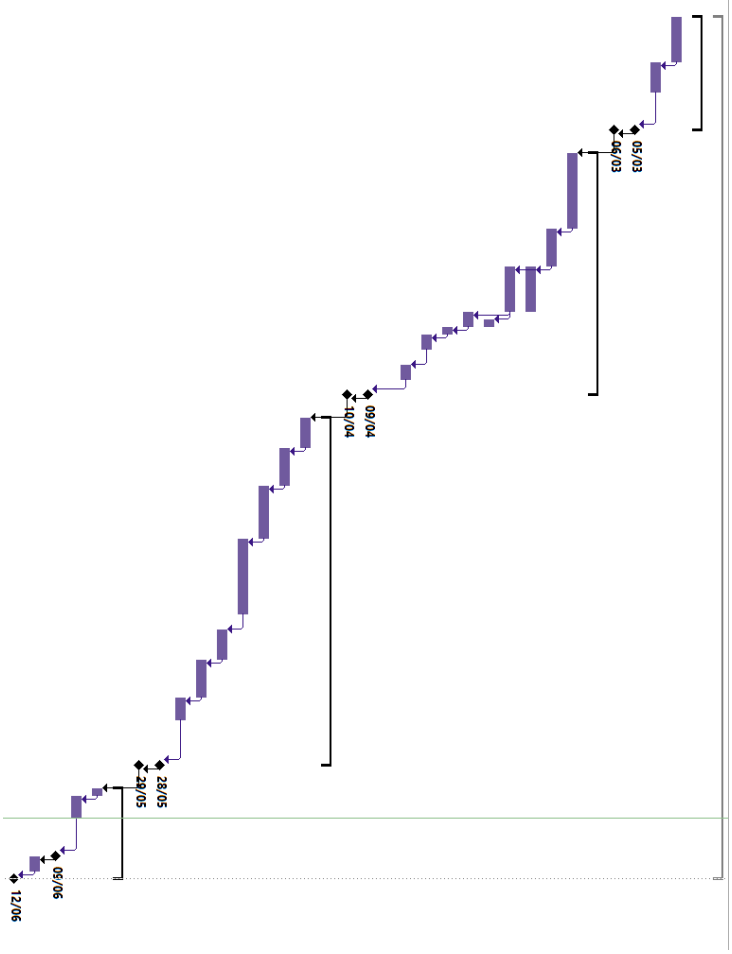


Figura metodología

1.4 Planificació del Treball

Snaptools Gantt	82 dias	jue 20/02/20	vie 12/06/20
1 Anàlisi de requisits	11 dias	jue 20/02/20	vie 06/03/20
1.1 Valoració diferents propostes	4 dias	jue 20/02/20	mar 25/02/20
1.2 Recerca d'informació del domini	4 dias	mié 26/02/20	sàb 29/02/20
1.3 Elaboració del document de la PAC 1	4 dias	lun 02/03/20	jue 05/03/20
1.4 Entrega PAC 1	0 dias	vie 06/03/20	vie 06/03/20
2 Disseny del sistema	24 dias	lun 09/03/20	vie 10/04/20
2.1 Creació de les pàgines estàtiques en HTML	8 dias	lun 09/03/20	mié 18/03/20
2.2 Creació dels diagrames UML	3 dias	jue 19/03/20	lun 23/03/20
2.3 Disseny arquitectura del sistema	5 dias	mar 24/03/20	dom 29/03/20
2.4 Disseny entitats	5 dias	mar 24/03/20	dom 29/03/20
2.5 Creació fitxers docker	1 dia	lun 30/03/20	mar 31/03/20
2.6 Instal·lació entorn local	2 dias	lun 30/03/20	mar 31/03/20
2.7 Preparació repositoris de versions de codi	1 dia	mié 01/04/20	mié 01/04/20
2.8 Preparació circuit CI/CD	2 dias	jue 02/04/20	vie 03/04/20
2.9 Recerca de rúbriques a implementar i dissenyar implementació	2 dias	lun 06/04/20	mar 07/04/20
2.10 Elaboració del document de la PAC 2	2 dias	mié 08/04/20	jue 09/04/20
2.11 Entrega PAC 2	0 dias	vie 10/04/20	vie 10/04/20
3 Codificació	34 dias	lun 13/04/20	vie 29/05/20
3.1 Implementació autentificació client/api	4 dias	lun 13/04/20	jue 16/04/20
3.2. Codificació sistema autentificació usuaris	3 dias	vie 17/04/20	mar 21/04/20
3.3 Codificació pàgina privada d'usuari	5 dias	mié 22/04/20	mar 28/04/20
3.4 Codificació de rúbriques	8 dias	mié 29/04/20	vie 08/05/20
3.5 Codificació sistema de processament de projec	4 dias	lun 11/05/20	jue 14/05/20
3.6 Integració crides client-api	3 dias	vie 15/05/20	mar 19/05/20
3.7 Proves de la plataforma	3 dias	mié 20/05/20	vie 22/05/20
3.8 Elaboració document de la PAC 3	4 dias	lun 25/05/20	jue 28/05/20
3.9 Entrega PAC 3	0 dias	vie 29/05/20	vie 29/05/20
4 Implementació	10 dias	lun 01/06/20	vie 12/06/20
4.1 Contractació hosting VPS	1 dia	lun 01/06/20	lun 01/06/20
4.2 Posada en marxa i configuració del servidor	3 dias	mar 02/06/20	jue 04/06/20
4.3 Elaboració dels documents per la memòria	3 dias	vie 05/06/20	mar 09/06/20
4.4 Elaboració memòria final	2 dias	mié 10/06/20	jue 11/06/20
4.5 Entrega memòria	1 dia	vie 12/06/20	vie 12/06/20



1.5 Recursos

A continuació es detallen els recursos tecnològics necessaris per al desenvolupament del projecte:

Programari	
Programa	**Ús**
JetBrains PHPStorm	Desenvolupament backend
JetBranins WebStorm	Desenvolupament frontend
Docker [1]	Aprovisionament de serveis
Git	Control de versions
Paw	Prova de crides HTTP

Maquinari		
Entorn	**Component**	**Descripció**
Desenvolupament	MacBook Pro i5 16GB RAM 256GB	macOS [17] Catalina 10.15.5
Producció	VPS 1GB RAM 25GB SSD	Ubuntu [4] 18.04.3 x64

1.6 Breu sumari de productes obtinguts

- Interfície d'usuari desenvolupada en *HTML*/*CSS*/*JavaScript*

  https://github.com/snaptools-uoc/snaptools-frontend

- *API* en *PHP*

  https://github.com/snaptools-uoc/snaptools-backend

- Memòria final del projecte

  PAC4_FINAL_MEM_ArnauRos_Carlos-Cristian.pdf

- Presentació del projecte

  PAC4_FINAL_PRS_ArnauRos_Carlos-Cristian.pdf

- Vídeo presentació del projecte

  Penjat a l'eina Present@.

Es pot consultar la versió en línia del producte final en la següent plana web.

https://www.snaptools.xyz

1.7 Breu descripció dels altres capítols de la memòria

La resta de capítols d'aquesta memòria es centren en la explicació dels diferents àmbits en els que es projecta aquest treball o que expliquen tant el desenvolupament d'aquest projecte com els patrons i eines fetes servir.

El primer apartat està dedicat a la infraestructura del projecte, on es fa un repàs als diferents entorns del projecte que han estat necessaris per la posada en marxa del projecte, la metodologia de treball i altres aspectes relacionats amb la posada en producció del codi generat.

A continuació seguim amb l'apartat destinat al client web, on s'explica l'organització del codi així com diferents apartats tècnics rellevants. Per finalitzar hi ha una mostra del disseny web amb la captura de totes les pantalles que son accessibles als usuaris.

Més endavant tenim l'apartat dedicat al *backend*, que és el motor que fa possible tota la funcionalitat. Aquesta secció és probablement la més extensa ja que cobreix la estructura del projecte, l'arquitectura de l'aplicació i l'explicació de múltiples decisions que s'han anat donant durant la seva creació.

A l'apartat d'aplicació tenim diferents aspectes que son més propers a la capa de negoci que al propi codi, com poden ser els casos d´ús i la lògica de domini.

Per últim tenim l'apartat de conclusions on es dona una visió global del que ha estat la realització d'aquest projecte des del punt de vista de l'estudiant.

2. Resta de capítols

2.1 Infraestructura

2.1.1 Circuit CI / CD

La integració contínua és la pràctica de la consolidació dels canvis de diferents desenvolupadors en un espai de codi comú. Aquesta tècnica és força utilitzada en equips amb diversos treballadors, que integren els seus canvis diversos cops al dia assegurant que el codi individual funciona amb la resta de canvis.

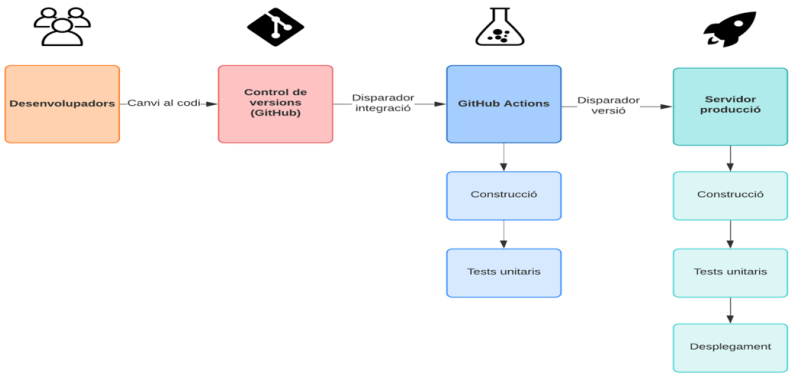
Vegem quins son els avantatges principals quan es segueix aquesta metodologia:

- Afavoreix la creació de codi que no interfereix amb altres parts del sistema ja codificades.
- Millora la qualitat del codi base atès que en la majoria dels casos el codi en desenvolupament pot ser enviat a l'entorn de producció amb garanties de que el projecte seguirà funcionant.
- Redueix el temps que passa entre que s'introdueix una nova falla i es detecta.
- Degut al apartat anterior, es redueix el cost de solucionar fallades de codi.

El desplegament continu és la pràctica de mantenir una aplicació desplegable en qualsevol punt o inclús automàticament desprès de validar la seva correctesa. Això dona molta llibertat als equips de desenvolupament perquè poden decidir quan volen fer noves versions del seu codi i no depenen de cap altre departament.

Tot i que en aquest projecte només hi va participar una persona, vaig considerar important establir una metodologia de treball inicial que em permetés certificar la correctesa dels meus desenvolupaments i automatitzés les tasques repetitives de desplegament.

Per els motius citats anteriorment vaig decidir crear i implementar un circuit de integració/desplegament continu amb GitHub Actions [9].



2.1.2 GitHub

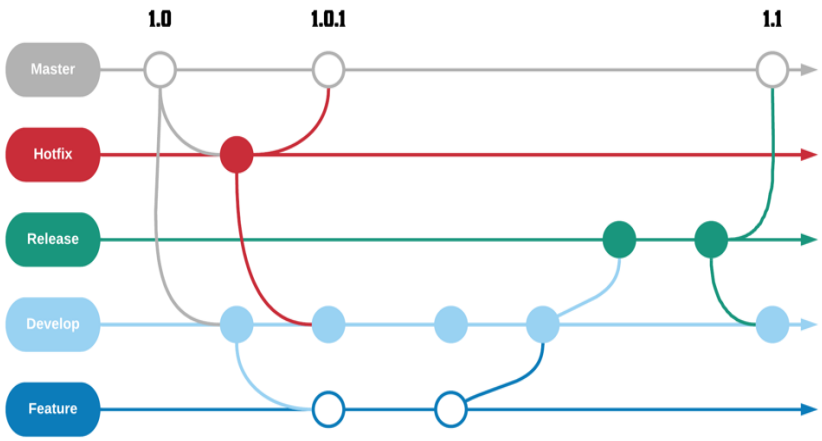
Una plataforma de control de versions és un sistema que ens permet guardar un registre de les modificacions que realitzem al llarg del temps als fitxers del nostre projecte. Hi ha diversos avantatges que han fet que sigui una eina indispensable avui en dia a l'hora de desenvolupar programari:

- Accés al registre de modificacions sobre un fitxer.
- Possibilitat de tornar a un punt de control anterior en el temps.
- Creació de branques per a separar el treball de diversos equips / desenvolupadors.
- Possibilitat d'etiquetar modificacions concretes, el que facilita la creació de versions.

D'entre les diverses plataformes que ofereixen control de versions basats en git [14] s'ha optat per fer servir GitHub [8], atès que l'experiència prèvia amb la seva interfície serà d'ajuda a l'hora de configurar el circuit d'integració/desplegament continu.

El flux de treball a seguir en el repositoris durant el desenvolupament serà Gitflow, que organitza el codi en dues branques principals i tres secundàries.

- Master: qualsevol commit que estigui en aquesta branca ha de estar preparat per a ser desplegat a producció.
- Develop: el codi d'aquesta branca és el que anirà en la pròxima versió planificada del projecte.
- Feature: originada a partir de la branca develop serveix per afegir noves funcionalitats. Un cop desenvolupat el codi s'ha de combinar de nou amb la branca develop.
- Hotfix: originada a partir de la branca master serveix per solucionar errors de manera ràpida a l'entorn de producció.
- Release: originada a partir de la branca develop serveix per preparar el pròxim codi a producció.



Tal i com veurem en els apartats següents, aquest projecte està format per un client al qual s'accedeix amb un navegador web i una api que li dona suport. Atès que son dos productes que tenen entitat pròpia i que estan desenvolupats en tecnologies diferents, cadascun d'ells estarà allotjat en el seu propi repositori.

Aquesta decisió s'ha pres també amb la intenció de que cadascun d'aquets subproductes pugi tenir el seu propi cicle de vida de desenvolupament i per tant no hi hagi una dependència directa entre ells.

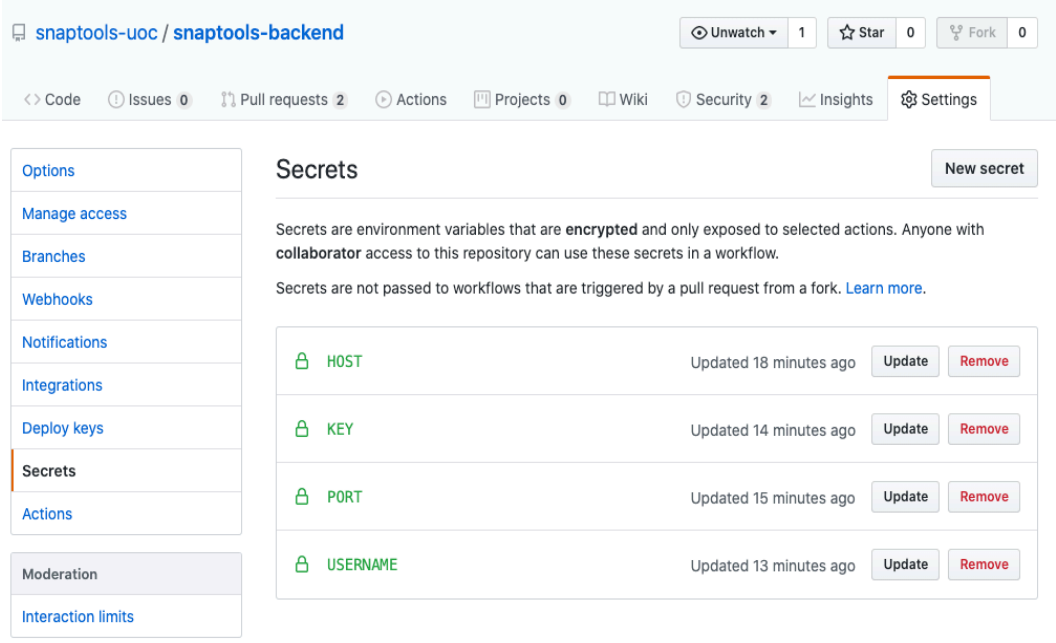
2.1.3 GitHub Actions

Aquesta eina ens permet automatitzar fluxos de treball mitjançant fitxers de text que han de residir al interior del repositori. Aquests fitxers de configuració contenen les passes que s'han d'executar quan hi ha certs canvis en el control de versions.

Per aquest projecte s'han configurat dos fluxos amb la següent funcionalitat:

- Comprovar que el codi de la branca de desenvolupament passa tots els tests unitaris.(*Continuous Integration*)
- Comprovar que el codi de la branca mestre passa tots els tests unitaris i desplega el codi al servidor de producció. (*Continuous Deployment*)

Com es pot apreciar en la configuració adjunta, per a la configuració de desplegament ha estat necessari crear diverses variables d'entorn mitjançant la pròpia interfície de *Github* [8]. Aquests paràmetres, també anomenats secrets, son els necessaris per a connectar mitjançant *ssh* al servidor on es desplegarà finalment el projecte web.



```
name: Continuous integration
on:
  push:
    branches: [ develop,  'release/**']
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: Install dependencies
        run: composer install --prefer-dist --no-progress --no-suggest

      - name: PHPUnit
        uses: chindit/actions-phpunit-symfony@master
```

```
name: Continuous deployment
on:
  push:
    branches: [ master ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Install dependencies
        run: composer install --prefer-dist --no-progress --no-suggest
      - name: Pass PHPUnit tests
        uses: chindit/actions-phpunit-symfony@master
      - name: Deploy to production server @ DigitalOcean
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.HOST }}
          username: ${{ secrets.USERNAME }}
          key: ${{ secrets.KEY }}
          port: ${{ secrets.PORT }}
          script: |
                cd /srv/host/github/snaptools-uoc/snaptools-backend &&
                git pull origin master && composer install && bin/console c:c &&
                bin/console doctrine:migrations:migrate --no-interaction
```

2.1.4 Entorns

2.1.4.1 Desenvolupament

Per tal d'agilitzar el desenvolupament de forma local en la meva màquina he fer servir *Docker* [1], un programari que permet crear contenidors amb

tot el necessari per fer funcionar una aplicació independentment del sistema operatiu en el que s'estigui executant.

Destaquem a continuació els diversos avantatges a l'hora de fer servir aquest programari que han portat al la decisió de fer-lo servir en comptes d'altres solucions com pot ser una màquina virtual o instal·lar el programari de forma local.

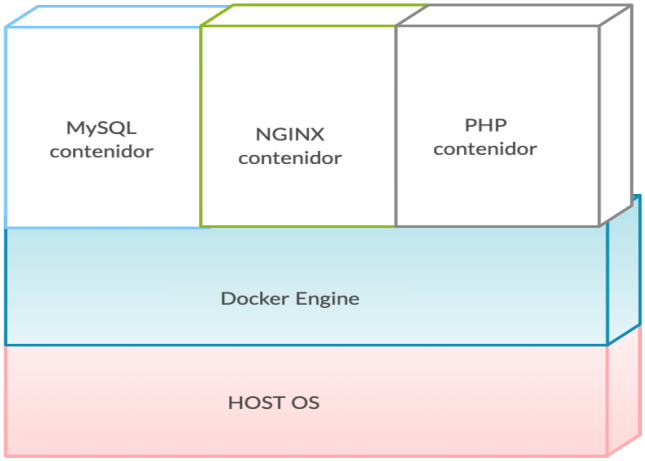
- Transformem la infraestructura de l'aplicació en codi, perquè la definició de tots els serveis necessaris està especificada en fitxers de text.

- L'anterior punt fa que el desplegament en qualsevol màquina garanteixi que es fan servir exactament les mateixes versions de programari.

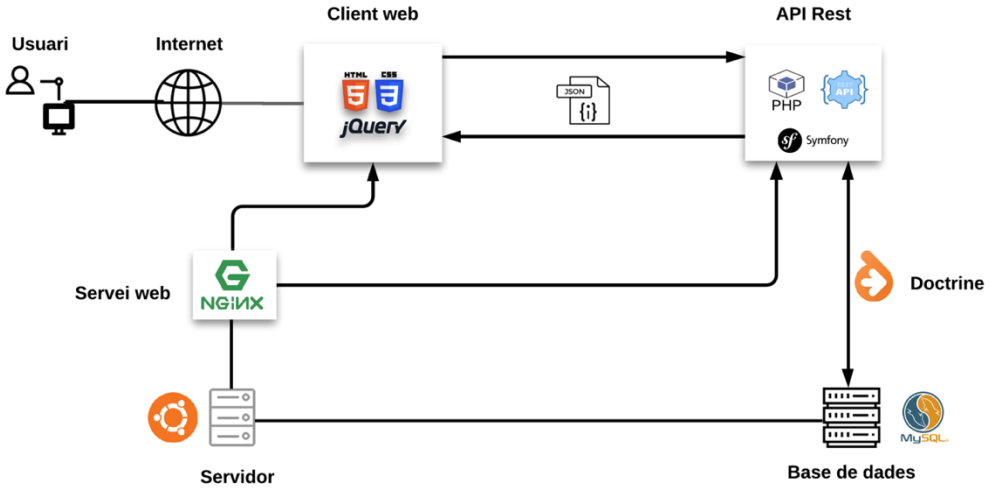
- Afegir nous serveis o modificar la versió dels existents es pot fer de forma molt senzilla. Facilita el manteniment del programari i la prova de l'aplicació amb diferents configuracions.

- La màquina on s'executa l'aplicació no ha de contenir cap programari a excepció de Docker [1].

- No és necessari un gran consum de recursos per executar els contenidors, cosa que si passa amb màquines virtual perquè és necessari virtualitzar un sistema operatiu sencer amb la quantitat de recursos que això suposa.



2.1.4.2 Producció

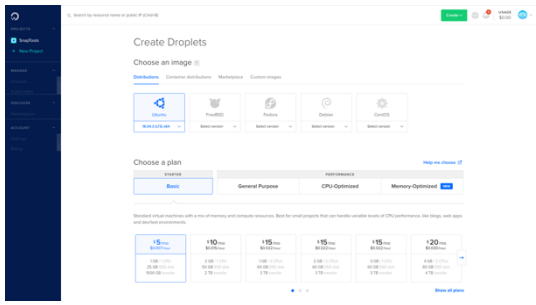


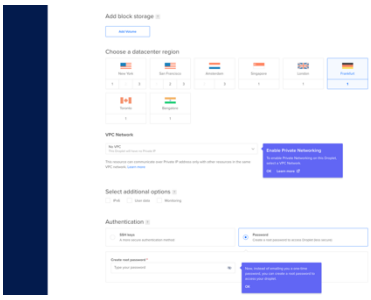
2.1.4.2.1 Contractació

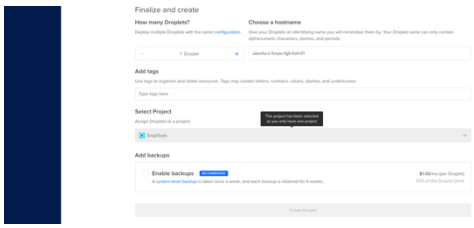
2.1.4.2.2 Instal·lació

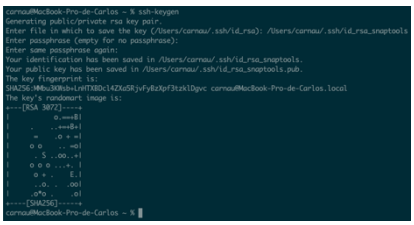
Tots els serveis necessaris són publicats per un servidor virtual privat amb el sistema operatiu Ubuntu [4] 18.04.3 x64 allotjat a DigitalOcean [23].

La creació del mateix es va dur a terme seguint els passos de la interfície web de la plataforma, generant una clau pública privada en la màquina local per connectar remotament per consola.









2.1.4.2.3 Ubuntu Firewall

Aquesta eina[5] permet configurar el tallafocs del sistema mitjançant una interfície intuïtiva, que fa servir internament *iptables* [2] per aplicar la configuració.

La configuració aplicada permet la connexió externa als ports necessaris per accedir a l'aplicació web i també per accedir al servidor de forma remota.

```
snaptools@snaptools:~# ufw allow 80/tcp
snaptools@snaptools~# ufw allow 443/tcp
snaptools@snaptools~# ufw allow 8081/tcp
snaptools@snaptools~# ufw allow 22/tcp
```

Un cop aplicades les regles va ser necessari habilitar el servei per aplicar la configuració i habilitar-la de forma permanent al sistema.

```
snaptools@snaptools:~# ufw enable
```

2.1.4.2.4 Certbot

Aquesta aplicació ens permet instal·lar de forma senzilla certificats *https* signats per una autoritat aprovada per els navegadors majoritaris.

```
snaptools@snaptools:~# apt-get update
snaptools@snaptools:~# apt-get install software-properties-common
snaptools@snaptools:~# add-apt-repository universe
snaptools@snaptools:~# add-apt-repository ppa:certbot/certbot
snaptools@snaptools:~# apt-get update

snaptools@snaptools:~# apt-get install certbot python3-certbot-nginx
```

Per generar els certificats i actualitzar la configuració del servei web podem fer servir la comanda següent i seguir les passes indicades.

```
snaptools@snaptools:~# certbot –nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx

Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: api.snaptools.xyz
2: www.snaptools.xyz
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave
input
blank to select all options shown (Enter 'c' to cancel):
```

2.1.4.2.5 Nginx

Primer va ser necessari instal·lar el servidor web [3], assegurant-nos primer que no estigués per defecte instal·lat *Apache* [16], que podria donar conflicte perquè fan servir el mateix port per defecte.

```
snaptools@snaptools:~# apt-get remove apache2
snaptools@snaptools:~# apt-get install nginx
```

Després aplicar les configuracions per a cada entorn als fitxers corresponents, que son disponibles a l'annex 6.1.

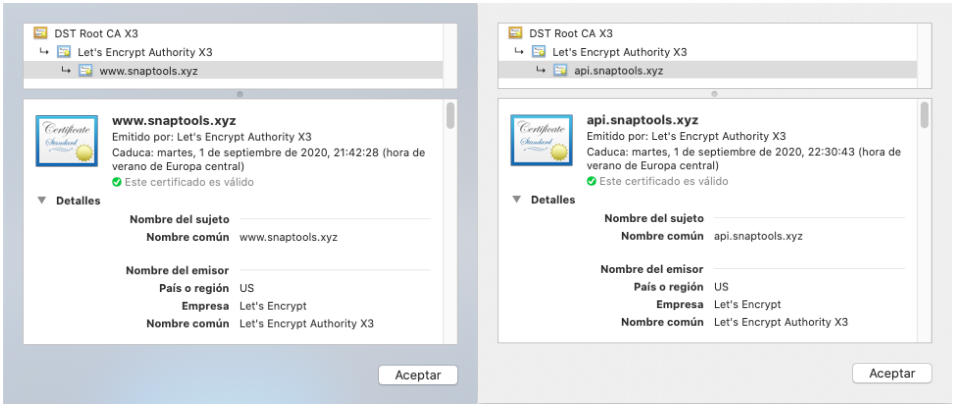
```
/etc/nginx/sites-available/api_snaptools.conf
```

```
/etc/nginx/sites-available/www_snaptools.conf
```

Per finalitzar activar les configuracions i reiniciar el servei.

```
snaptools@snaptools:~# cd /et/nginx/sites-enabled
snaptools@snaptools:~# ln -s ../sites-available/api_snaptools.conf
api_snaptools.conf
snaptools@snaptools:~# ln -s ../sites-available/www_snaptools.conf
www_snaptools.conf
snaptools@snaptools:~# service nginx reload
```

Ara ja podem comprovar en el nostre navegador que ambdós llocs tenen un certificat vàlid i serveixen el contingut sota una connexió segura.



2.1.4.2.5 PHP

PHP-FPM és una implementació alternativa de *PHP FastCGI* encarregada d'interpretar codi *PHP* [10] que s'executa com a servei independent del servidor web.

Les següents comandes van ser les executades per instal·lar aquest component essencial en el que es basa el bastiment *Symfony* [20].

```
snaptools@snaptools:~# apt-get install php7.4-fpm
snaptools@snaptools:~# apt-get install php7.4-xml
snaptools@snaptools:~# apt-get install php7.4-json
snaptools@snaptools:~# apt-get install php7.4-mbstring
snaptools@snaptools:~# apt-get install php7.4-zip
snaptools@snaptools:~# apt-get install php7.4-mysql
```

2.1.4.2.6 Base de dades

Les dades de l'aplicació quedaran emmagatzemades en una base de dades relacional proporcionada per el servei *MySQL* [7], que només escoltarà connexions entrants des de la pròpia màquina.

L'accés a aquest recurs per poder obtenir i registrar informació des de l'aplicació es va configurar mitjançant un usuari amb els permisos necessaris a la base de dades esmentada.

La generació del esquema inicial correspon a la pròpia aplicació ja que està registrat amb versions.

```
snaptools@snaptools:~# apt-get install mysql-server mysql-client
snaptools@snaptools:~# mysql_secure_installation
mysql> use snaptools_backend;
mysql> CREATE USER snaptools@localhost IDENTIFIED BY 'contrasenya';
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY TABLES ON snaptools_backend.* TO 'snaptools'@'localhost' IDENTIFIED BY 'contrasenya';
mysql> flush privileges;
```

2.1.4.2.7 Aplicació

2.1.4.2.7.1 Frontend

El client no té requeriments, només és necessari clonar el projecte des de el repositori.

```
snaptools@snaptools:~# cd /srv/host/github/snaptools-uoc
snaptools@snaptools:~# git clone git@github.com:snaptools-uoc/snaptools-frontend.git
```

2.1.4.2.7.2 Backend

El producte d'api requereix de diverses operacions un cop tenim descarregat el codi del repositori. A més d'aquest pas inicial és necessari instal·lar totes les llibreries externes amb el gestor de paquets composar, crear la base de dades i generar el esquema actual.

```
snaptools@snaptools:~# cd /srv/host/github/snaptools-uoc
snaptools@snaptools:~# git clone git@github.com:snaptools-uoc/snaptools-backend.git
snaptools@snaptools:~# cd snaptools-backend
snaptools@snaptools:~# composer install
snaptools@snaptools:~# bin/console doctrine:database:create
snaptools@snaptools:~# bin/console doctrine:migrations:migrate
```

Un cop fetes aquestes passes es pot comprovar que l'aplicació és operativa des de el navegador.

2.2 Frontend

2.2.1 Introducció

Per agilitzar la creació del projecte i ates que crear un entorn visual des de zero amb els requeriments actuals(bon disseny, responsiu, adaptat a diversos navegadors), es va fer servir com a base per la part visual una plantilla prefabricada.

Es van valorar diferents alternatives i es va optar per *Metronic* [24] com a primera opció, perquè oferia diferents versions sobre un mateix disseny en diferents bastiments a l'hora que també comptava amb una extensa documentació.

Després de treballar-hi amb més detall i com a resultat de trobar diverses dificultats a l'hora de desenvolupar que feien més costós el treball i endarrerien els temps previstos, es va decidir canviar de plantilla per una de les altres que havien estat valorades.

L'alternativa que havia quedat descartada en un primer moment, *AdminThemes* [22], va passar finalment a ser en la que es va realitzar el treball.

La llicencia d'ús permet crear una aplicació sempre que els usuaris finals no hagin de pagar per accedir-hi tant única com mensualment, cosa que no està prevista com a part del projecte.

2.2.2 Desenvolupament

La plantilla escollida venia amb diferents pàgines ja implementades que es van modificar per a que complissin les necessitats del projecte.

Entre les tasques a destacar que va comportar aquest procés podem destacar les següents:

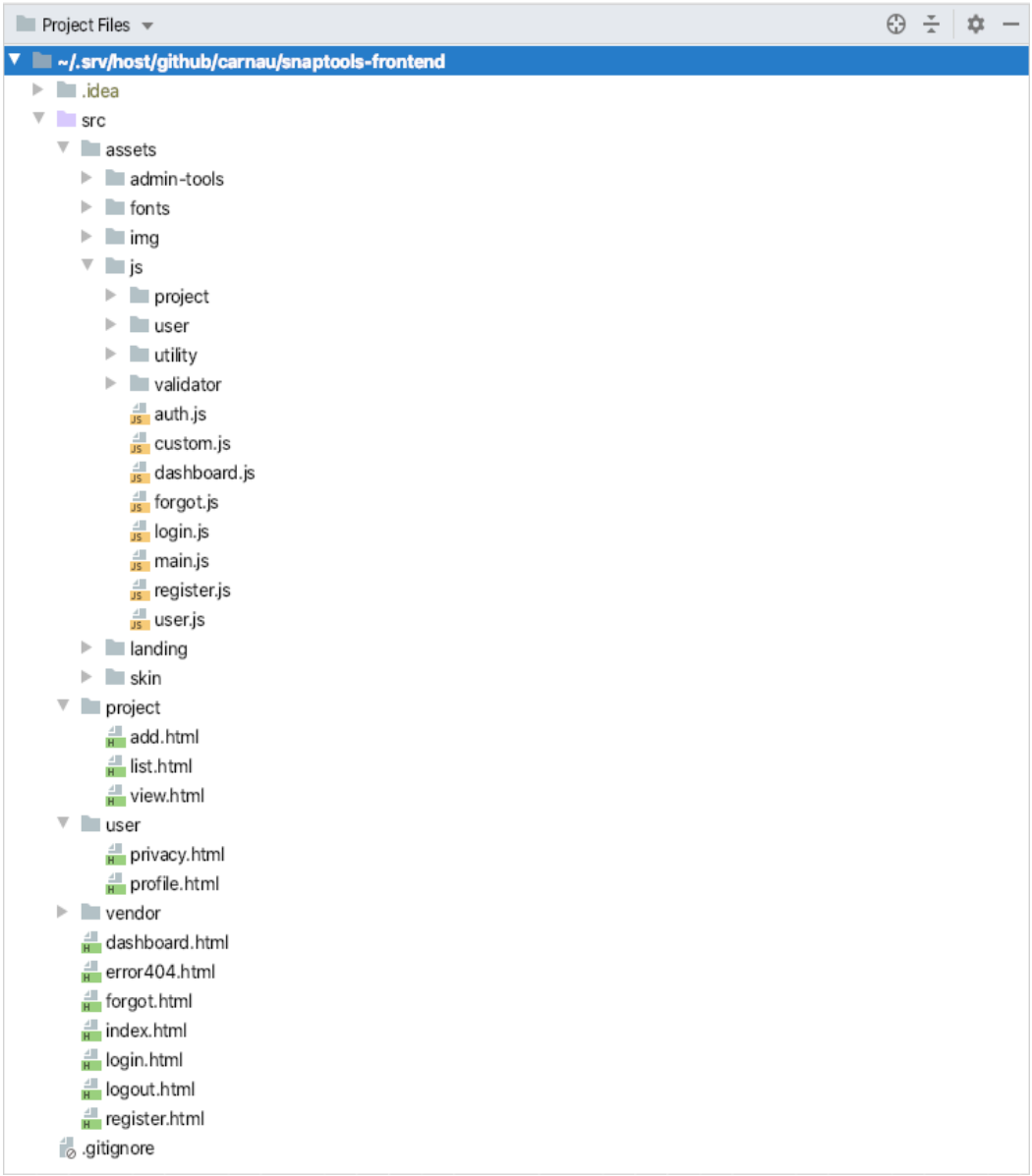
- Eliminar codi que no era necessari per el projecte.
- Modificar i reutilitzar el *HTML* original per a tenir el aspecte desitjat.

- Desenvolupar la interacció de cada pàgina fent servir la llibreria *JQuery* [25].

2.2.2.1 Estructura

A continuació podem veure com queda l'estructura de directoris del projecte, on cada pàgina està allotjada en la estructura de carpetes que correspon a la ruta en el navegador.

Cadascun dels fitxers codificats en HTML carreguen els diversos mòduls JavaScript [12] que proporcionen la interactivitat amb el servidor.



2.2.2.2 Interacció amb backend

Els diferents components desenvolupats per establir una comunicació amb el servidor backend fan servir la tècnica *AJAX*(*Asynchronous JavaScript* [12] and *XML*) que permet actualitzar el arbre *DOM* sense necessitat de recarregar la pàgina actual.

La llibreria utilitzada disposa d'un mètode per fer peticions de forma asíncrona, anomenat *AJAX*, que és el que es va fer servir per a obtenir les diferents dades necessàries per a cada pàgina.

En els casos que la petició havia d'adjuntar informació d'un formulari, aquestes dades es van transformar prèviament a la notació *JSON* suportada per el servidor *backend*.

En el cas de que el token emmagatzemat hagi expirat i el servidor retorni un codi d'estat *HTTP* 401, l'aplicació retornarà a l'usuari a la pàgina per autentificar-se de nou.

2.2.2.3 JavaScript

A continuació es detalla la funcionalitat i responsabilitat de cadascun dels fitxers JavaScript [12] desenvolupats.

Fitxer	Descripció
auth.js	Interactua amb localStorage per emmagatzemar les dades necessàries per l'aplicació i és capaç de validar si l'usuari té un token vàlid i per tant una sessió activa.
dashboard.js	Obté i presenta les dades estadístiques que es mostren en la pantalla d'administrador.
forgot.js	Envia una petició al servidor backend per a re establir la contrasenya de l'usuari.
login.js	Determina si les credencials proporcionades son correctes interactuant amb el servidor, i en cas afirmatiu emmagatzema el token, les dades d'usuari i redirigeix a la pàgina principal de l'aplicació. Proporciona validació de client per al formulari de la pàgina.
register.js	Registra un usuari al sistema amb les dades proporcionades. Proporciona validació de client per al formulari de la pàgina.
user.js	Interactua amb la base de dades del navegador per guardar i obtenir la informació de l'usuari autenticat.

	Proporciona validació de client per al formulari de la pàgina.
project/add.js	Envia les dades d'un nou projecte per a ser creat, un cop això succeeix redirigeix a l'usuari a la pàgina del projecte. Proporciona validació de client per al formulari de la pàgina. Addicionalment es va tenir que crear un nou mètode de validació personalitzat per a que només acceptés pàgines de projectes Snap! [19]
project/list.js	Obté el llistat de projectes creat per l'usuari en format de pàgines en funció de la quantitat de projectes creats per l'usuari.
project/view.js	Obté les dades del projecte que s'obté per la url.
user/profile.js	Permet actualitzar les dades de l'usuari mostrades al formulari. Proporciona validació de client per al formulari de la pàgina.
user/privacy.js	Permet actualitzar la contrasenya de l'usuari. Proporciona validació de client per al formulari de la pàgina.

2.2.2.3 Emmagatzemat de dades

Els navegadors actuals ens permeten emmagatzemar dades de manera local sense realitzar cap connexió a una base de dades mitjançant localStorage i sessionStorage.

És important escollir una o un altra en funció de les dades que volem guardar perquè metre localStorage emmagatzema les dades de manera indefinida, sessionStorage ho fa mentre la pestanya on s'estigui utilitzant estigui oberta.

La nostra aplicació necessita emmagatzemar el token per a poder fer crides al servidor i les dades de l'usuari actual, a més no volem que l'usuari hagi d'autentificar-se cada cop que tanca la pestanya del navegador sinó que pugui ser recordat durant un cert temps.

Per aquest motius es va decidir que era millor guardar les dades al localStorage i que en cas de que es necessites emmagatzemar més dades es valoraria on seria millor tenir-les.

2.2.2.4 JWT

El procés d'autentificació de l'usuari mitjançant usuari i contrasenya es du a terme a la pàgina de login.

Un cop s'ha validat que les credencials proporcionades son correctes, el client guarda localment el token d'accés proporcionat per el servidor.

Sempre que l'usuari vulgui accedir a un recurs protegit ha d'adjuntar una capçalera HTTP amb el següent format:

```
Authentication: Bearer {token}
```

En cas de que el servidor retorni que el token ha expirat, l'aplicació redirigirà a l'usuari de nou a la pàgina de login.

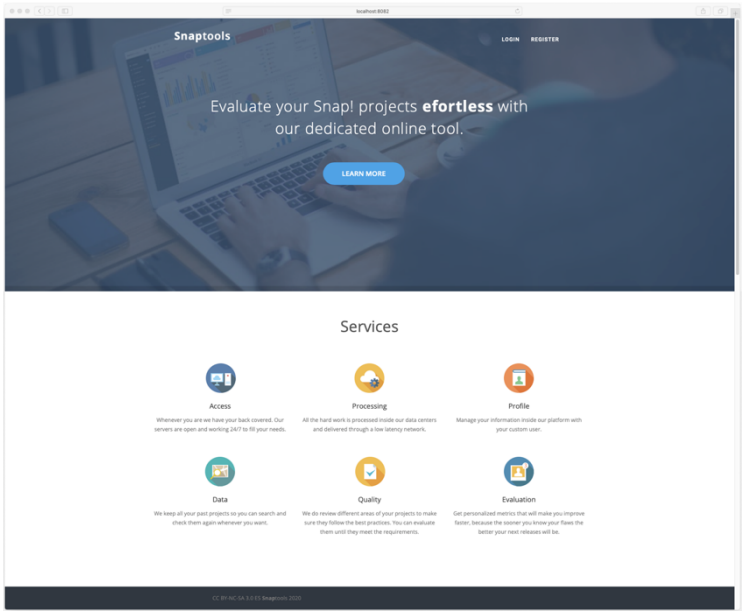
2.2.2.5 Validació client

Abans d'enviar dades al servidor és important assegurar-se de que tots els controls de formulari estan omplerts amb dades que tenen el format correcte. Aquesta validació proporciona una bona experiència d'usuari perquè elimina el temps que suposa enviar les dades i notificar a l'usuari que son incorrectes.

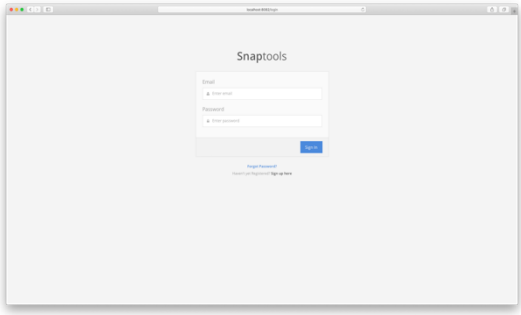
En aquest projecte implementa la validació de client fent servir un plugin de la llibreria JQuery [25], que carrega la nostra configuració personalitzada un cop ha carregat el DOM. Per tant, en totes les pantalles on hi ha un formulari la llibreria es preocupa d'afegir els manegadors necessaris als controls per validar les dades que contenen.

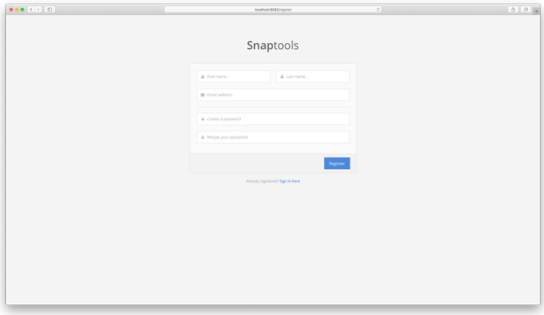
2.2.2.6.Interfície web

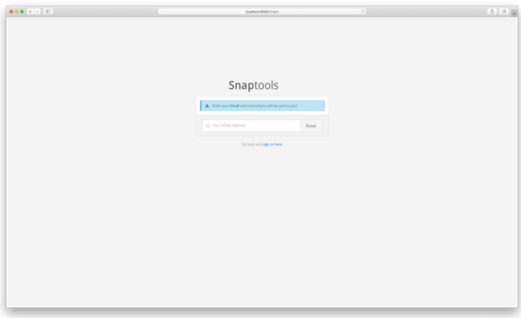
2.2.2.6.1 Portada

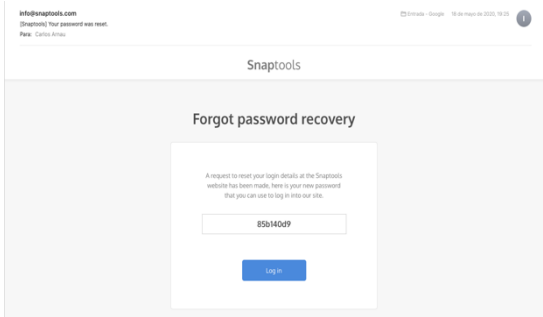


2.2.2.6.2 Autentificació

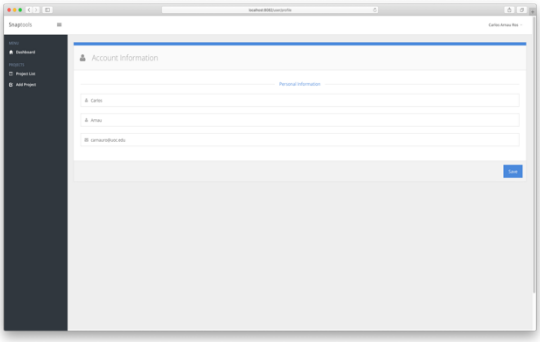


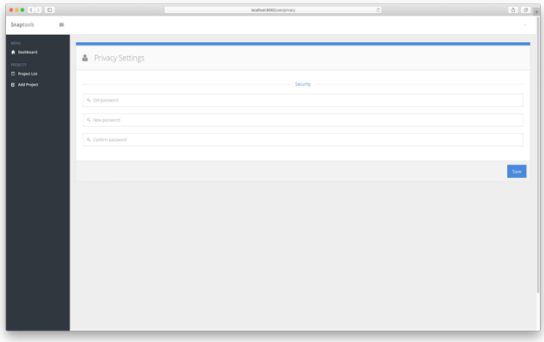




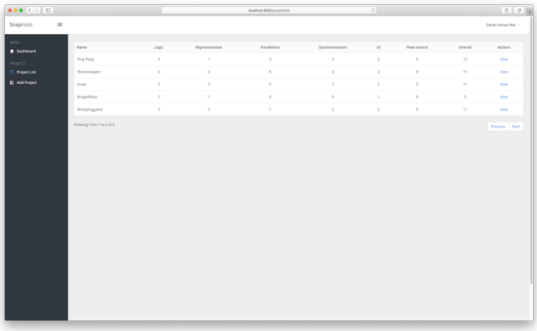


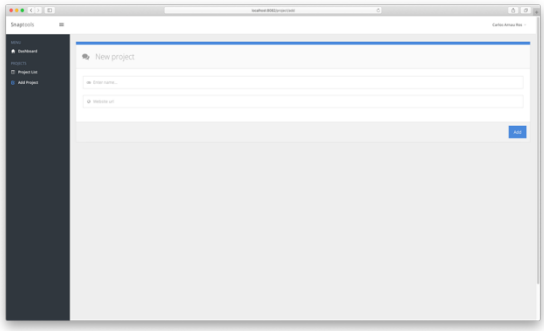
2.2.2.6.3 Perfil

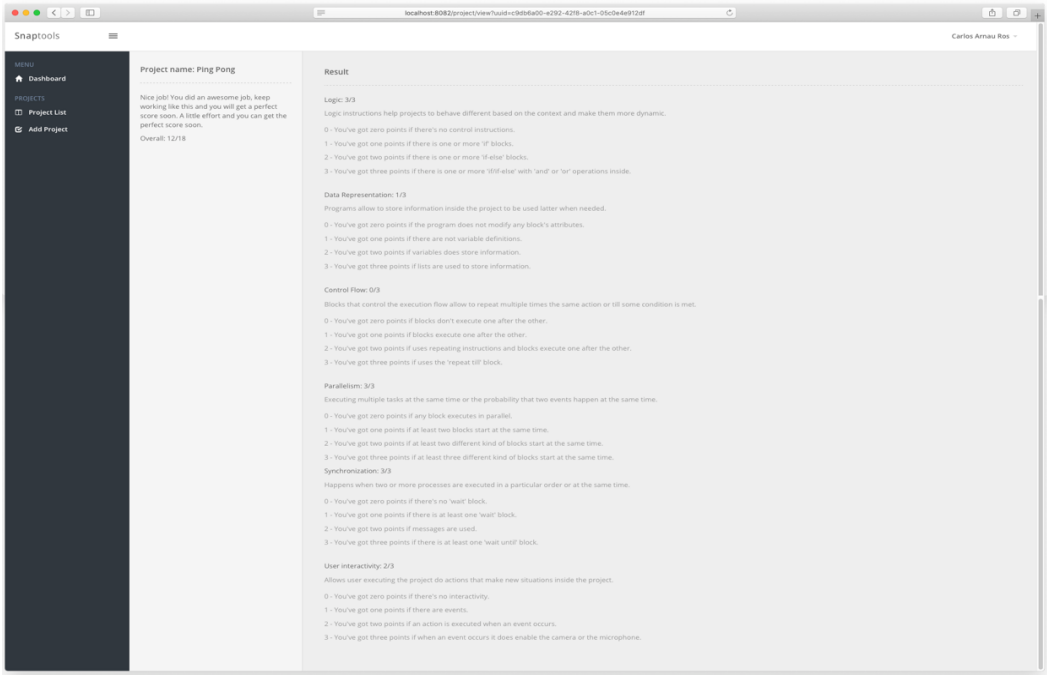




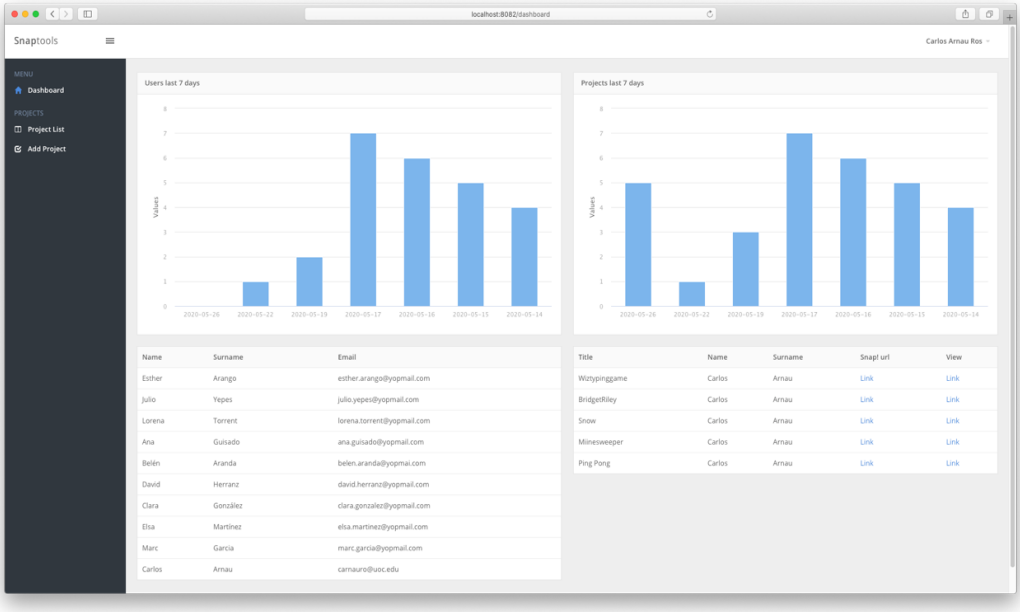
2.2.2.6.4 Projectes







2.2.2.6.5 Dashboard



2.3 Backend

Avui en dia existeixen multitud de solucions en diferents llenguatges per a dur a terme aquesta tasca, moltes de elles vàlides per aconseguir els objectius d'aquest treball. Aquest motiu ha portat a establir uns criteris de selecció que han estat escollits segons les necessitats i requeriments del projecte.

- Llenguatge enfocat al desenvolupament web.
- Ecosistema d'aplicacions en estat madur.
- Integració amb sistemes de persistència de dades.
- Possibilitat d'aplicar tècniques de test.

Una gran part dels estudis del grau han comportat el desenvolupament en el llenguatge Java, que de ben segur compleix tots els requisits exposats, però finalment el projecte serà desenvolupat en *PHP*.

Aquesta decisió es va basar en una preferència personal de l'autor del treball final de grau, sense deixar de banda que aquesta solució també ofereix documentació de qualitat, estabilitat entre versions, extensibilitat a través de components propis i flexibilitat il·limitada.

La part interna de l'aplicació es va desenvolupar en *Symfony* [20], un bastiment en llenguatge *PHP* [10] de programari lliure que permet crear aplicacions i llocs web de forma rapida, segura i professional.

2.3.1 Arquitectura d'alt nivell

REST és un estil d'arquitectura de programari àmpliament utilitzat avui dia per a tot tipus de serveis. El seu creixement ha augmentat notablement en els últims anys principalment perquè és una alternativa molt menys complexa que *SOAP*.

Existeixen diversos motius que expliquen el perquè d'aquest creixement i també perquè ha estat escollit per dur a terme aquest projecte.

La separació entre client/servidor ofereix un baix acoblament entre la interfície gràfica i la gestió de dades, que permet que els diferents components puguin evolucionar de forma independent.

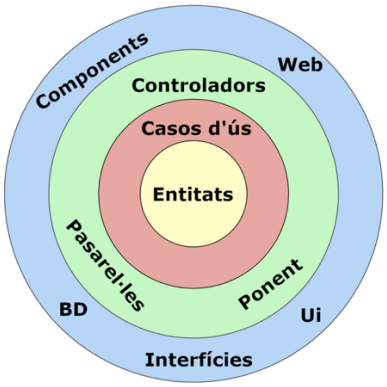
En projectes d'alt rendiment es pot escalar cada component en funció de les necessitats puntuals.

És independent de la plataforma o llenguatge, l'únic requisit és que segueixi els contractes de les interfícies que ofereix públicament.

2.3.2 Arquitectura de sistema

El projecte està organitzat segons una arquitectura hexagonal, que té com a objectiu crear aplicacions amb poc acoblament, fer que els components siguin intercanviables a qualsevol nivell i que es puguin testejar de forma isolada respecte a qualsevol element extern.

Els motius van portar tat a seleccionar aquest patró han estat els diferents beneficis que aporta al desenvolupament del programari com per exemple l'alta mantenibilitat, el baix deute tècnic, la facilitat per afegir nous components i una correcta separació de responsabilitat.



2.3.3 Arquitectura interna

Com hem vist anteriorment una *API Rest* s'encarrega principalment de gestionar recursos, el que implica crear, modificar, esborrar i consultar entitats de forma continuada.

Hi ha aplicacions que no tenen aquest tipus d'operacions balancejades, és a dir, tenen moltes més consultes que no modificacions o a l'inrevés.

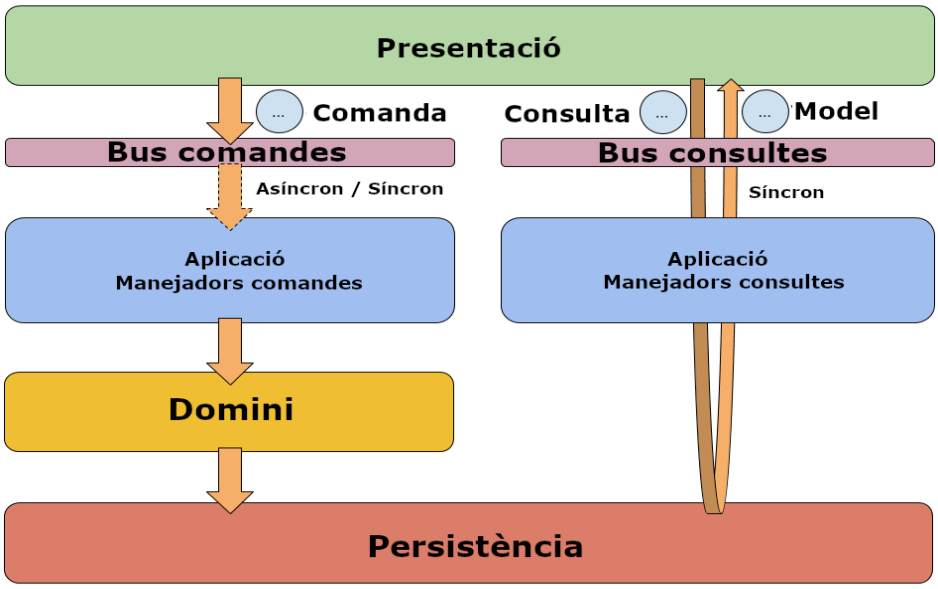
En aquests casos el més habitual és separar els models de consulta i modificació perquè llavors es poden escalar de forma independent.

Tot i que aquest projecte no tindrà tràfic i això no serà un problema s'ha trobat interessant aplicar una arquitectura que permeti l'escalabilitat i la possibilitat de afegir asincronia en alguns casos d'ús que podrien necessitar-ho.

CQRS és un patró d'arquitectura que permet separar les operacions d'un sistema en dues categories:

Consultes: Retornen un resultat sense canviar l'estat del sistema.
Comandes: Canvien l'estat del sistema.

A continuació es mostra el diagrama de com quedarà el sistema internament.



2.3.4 Estructura dels fitxers del projecte

Directori / Fitxer	Descripció
.docker	Fitxers de configuració Docker [1] per llançar l'aplicació.
bin	Els diferents binaris
config	Configuracions dels diferents mòduls.
src	Codi font de l'aplicació separat per contexts.
tests	Emmagatzema totes les proves unitàries.
var	Directoris que emmagatzemen els logs de sistema i la caché.
vendor	Llibreries externes que son necessàries per fer funcionar l'aplicació.
.env	Variables d'entorn per defecte del projecte.
.env.local	Variables d'entorn per el projecte en local.
.env.test	Variables d'entorn per el projecte en mode proves.
composer.json	Fitxer de configuració del gestor de paquets.

2.3.5 Estructura del codi font

El directori que conté els fitxers de l'aplicació es va distribuir en diversos límits lògics que anomenarem contextos. Aquesta és una manera de categoritzar va permetre agrupar codi en petits sub grups amb la mateixa intenció on cada context té la seva carpeta pròpia, dividida internament en les capes d'aplicació, domini i infraestructura.

Aplicació: Orquestra i coordina els casos d'ús per a que resolguin problemes, delegant aquestes tasques a la capa de domini.

Domini: Representa els conceptes de negoci i interactua amb la capa d'infraestructura mitjançant interfícies per no tenir una dependència directa.

Infraestructura: Interactua amb els serveis externs com poden ser la persistència de dades o enviament de correus.

2.3.5.1 Contexts

En aquest apartat es els diferents contexts que es van crear per sub dividir el codi font en unitats més petites.

2.3.5.1.1 Authentication

Gestiona les accions que realitza l'usuari contra el sistema per identificar-se o registrar-se.

2.3.5.1.1.1 Interfície HTTP

HTTP	Punt d'entrada	
GET	/api/v1/login	Identificar un usuari.
POST	/api/v1/register	Registrar un usuari.
POST	/api/v1/forgot	Posar nou password aleatori i enviar-lo per correu a l'usuari
GET	/api/v1/me	Retornar dades usuari.

2.3.5.1.2 Dashboard

Dona suport per mostrar la pàgina d'estadístiques a l'administrador del sistema.

2.3.5.1.2.1 Interfície *HTTP*

HTTP	Punt d'entrada	Funcionalitat
GET	/api/v1/get/dashboard/statistics	Retornar dades dels gràfics de projectes i usuaris.
GET	/api/v1/get/dashboard/users/last	Retornar últims usuaris.
GET	/api/v1/get/dashboard/projects/last	Retornar últims projectes.

2.3.5.1.3 Mail

Conté el servei que és capaç d'enviar correus a l'exterior. No disposa de interfícies amb l'exterior.

2.3.5.1.4 Profile

2.3.5.1.4.1 Interfície *HTTP*

HTTP	Punt d'entrada	Funcionalitat
POST	/api/v1/update/profile	Guardar dades d'usuari.
POST	/api/v1/privacy/password	Canviar contrasenya usuari.

2.3.5.1.5 Project

Agrupa tota la funcionalitat relacionada amb els projectes per la seva adició o llistat.

2.3.5.1.5.1 Interfície *HTTP*

HTTP	Punt d'entrada	
GET	/api/v1/get/projects	Retornar projectes d'usuari.
POST	/api/v1/add/project	Afegir i processar nou projecte.
POST	/api/v1/get/project	Retornar dades de projecte.

2.3.5.1.6 Shared

Disposa de múltiples classes reutilitzables en un o més contexts.

2.3.5.1.7 Snap

Proporciona el processador de projectes i tots els serveis encarregats de processar el *xml* provinent de ells. Inclou també el servei de descàrrega del projecte *Snap!* [19].

2.3.6 Seguretat

En el següent apartat parlarem de les mesures que es van dur a terme per garantir un accés controlat als serveis exposats per el servidor backend, que van ser configurats fent ús de llibreries externes que donen suport a *Symfony* [20].

2.3.6.1 JWT

JSON Web Tokens [18] és un estàndard per compartir secrets de forma segura, pensat per cobrir diversos tipus d'aplicacions entre les que es troben els components web.

Per instal·lar el mòdul que ens permet automatitzar l'autentificació mitjançant *tokens JWT* [18] podem fer servir el gestor de paquets *composer* [21].

```
composer require lexik/jwt-authentication-bundle
```

La llibreria fa servir criptografia de clau pública/privada tant per generar els tokens com per verificar-los. És important generar diferents claus per a cada entorn i que estiguin en un lloc segur que no sigui accessible per el servidor web.

En el nostre cas guardarem els fitxers generats dintre del directori de configuració del bastiment.

```
mkdir -p /srv/host/github/carnau/snaptools-backend/config/jwt
openssl genpkey -out config/jwt/private.pem -aes256 -algorithm rsa -pkeyopt
rsa_keygen_bits:4096
openssl pkey -in config/jwt/private.pem -out config/jwt/public.pem -pubout
```

Crear l'arxiu de configuració següent per indicar on hem guardat les claus generades.

```
config/packages/lexik_jwt_authentication.yml
```

```
lexik_jwt_authentication:
   secret_key:      '%kernel.project_dir%/config/jwt/private.pem'
   public_key:      '%kernel.project_dir%/config/jwt/public.pem'
   pass_phrase:     'your_secret_passphrase'
   token_ttl:       3600
```

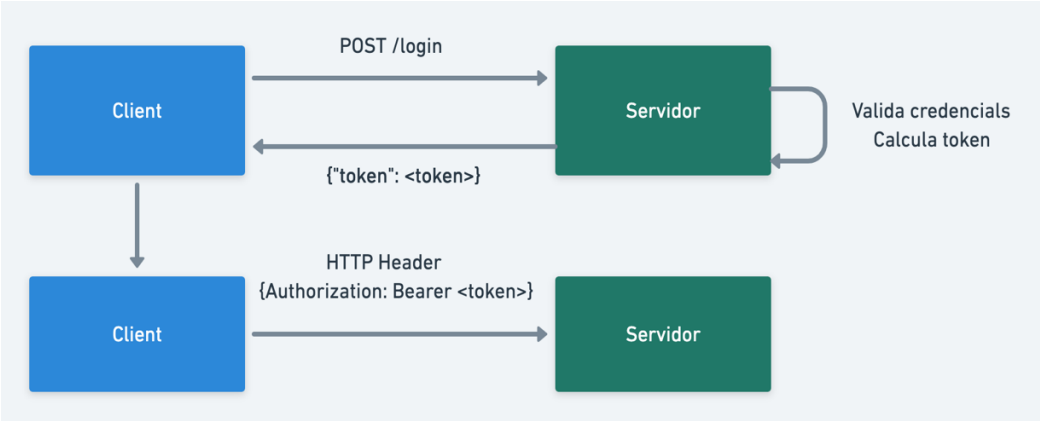
També hem d'afegir el *bundle* per a que sigui carregat a l'inici.

```
config/bundles.php
```

```
return [

Lexik\Bundle\JWTAuthenticationBundle\LexikJWTAuthenticationBundle::class
=> ['all' => true],
];
```

Finalment queda configurar el tallafocs de l'aplicació per a que faci servir com a mètode d'autentificació els tokens *JWT* [18] per a totes les crides que comencin per /api/v1

El client envia una petició de login al servidor, que valida si les credencials proporcionades son correctes. En cas afirmatiu genera un *token* que envia al client per a que el guardi i utilitzi per autentificar-se en les següents crides.



2.3.6.2 Tallafocs

A més de proporcionar un mecanisme d'autentificació per a clients externs, també va se necessari configurar el funcionament del tallafocs de l'aplicació.

Com que l'aplicació de les regles del tallafocs s'apliquen de forma seqüencial, va ser necessari definir quatre entrades.

- dev: no hi ha cap tipus d'autentificació per fitxers *css*, *js* i dels que formen part del *profiler*.
- login: permet la crida anònima perquè encara l'usuari no s'ha identificat. En cas de fer login genera el token *JWT* [18] corresponent.
- register: permet la crida anònima perquè l'usuari encara no s'ha identificat.
- forgot: permet la crida anònima perquè l'usuari encara no s'ha identificat.
- *api*: requereix un token *JWT* [18] vàlid per processar la petició.

Per finalitzar, es va configurar el control d'accés per a que totes les rutes requerissin que l'usuari estigués identificat.

2.3.6.3 CORS

Per raons de seguretat, els navegadors actuals no permeten l'intercanvi de dades entre servidors en dominis o ports diferents quan les peticions han estat iniciades des de un script.

Un exemple d'aquest tipus de sol·licitud seria un codi JavaScript [12] d'una aplicació web localitzada a https://www.snaptools.com:80 utilitzant una crida *AJAX* per obtenir dades de https://api.snaptools.com:8081.

Per establir aquesta comunicació de manera consensuada existeix una especificació anomenada *CORS* (Cross-Origin Resource Sharing), que mitjançant l'ús de diferents paràmetres a les capçaleres dels missatges HTTP validen l'intercanvi de dades.

Es per aquest motiu que va ser necessari instal·lar un component de programari extra al bastiment i configurar-lo per a que es pogués establir la comunicació.

A continuació podem veure com va ser aquest procés i el resultat final.

```
composer require nelmio/cors-bundle
```

```
vim config/packages/nelmio_cors.yml
```

```
nelmio_cors:
  defaults:
    origin_regex: true
    allow_origin: ['%env(CORS_ALLOW_ORIGIN)%']
    allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH', 'DELETE']
    allow_headers: ['Content-Type', 'Authorization']
    expose_headers: ['Link']
    max_age: 3600
  paths:
    '^/': null
```

```
vim .env.local
```

```
###> nelmio/cors-bundle ###
CORS_ALLOW_ORIGIN=^https?://(www.snaptools.xyz|localhost|127\.0\.0\.1)(:
[0-9]+)?$
###< nelmio/cors-bundle ###
```

2.3.6.4 Rols

A banda dels rols que ja venien creats per defecte, es va crear un nou rol 'ROLE_ADMIN' per accedir a les crides que el client fa servir per mostrar la informació a la pàgina de dashboard.

2.3.7 Components de *Symfony*

2.3.7.1 Missatger

La implementació del patró *CQRS* es va realitzar mitjançant el component de missatgeria de *Symfony* [20], definint dos busos independents per gestionar comandes i consultes.

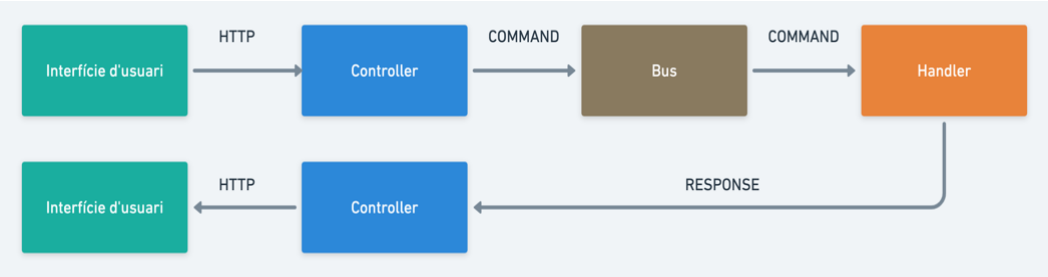
La seva instal·lació es va dur a terme de la següent manera fent servir el gestor de paquets.

```
composer require symfony/messenger
```

En el següent gràfic podem veure el cicle complert d'una petició, que comença a la interfície d'usuari (en el nostre cas el client web) amb la creació del missatge *HTTP* corresponent. Un cop el controlador rep la informació genera una comanda que es enviada al bus corresponent on és entregada al manegador.

En aquest projecte totes les cues treballen de forma síncrona, és a dir, el missatge es consumeix en el mateix moment que s'envia al bus.

Acabat el processament de la comanda, el manegador s'encarrega de retornar les dades al controlador que genera la resposta per al client.



2.3.7.2 Serializador

Aquest component automatitza la serialització d'entitats en formats específics i va ser àmpliament utilitzat en els controladors per transformar les entitats a format JSON per a ser enviades al client com a resposta a les peticions.

Cada cop que es converteix una entitat a un format es pot especificar quin grup de serialització volem fer servir, és a dir, cada atribut pot pertànyer a un o més grups de serialització que permet tenir diferents versions d'un mateix objecte.

Aquest comportament es pot fer servir per exemple per fer diferents versions del mateix projecte, on els clients han d'especificar quina versió volen fer servir. També evidentment per limitar la informació que s'envia al client per motius de seguretat o per estalviar transferència de dades.

En el nostre cas va ser necessari configurar dos grups, un principal anomenat '*main*' que conté tots els atributs a excepció dels considerats sensibles com per exemple les contrasenyes d'usuari, i un altre '*dashboard*' que té informació limitada per mostrar les estadístiques a l'administrador.

En una futura ampliació pot tenir sentit afegir més grups en funció de les versions del programari.

La instal·lació del component es va fer mitjançant el gestor de paquets composar tal i com es mostra a continuació.

```
composer require symfony/serializer
```

Cada vegada que el serializador ha de convertir una entitat carregarà els fitxers de configuració proporcionats a la carpeta config/serializer. Es pot veure una mostra en la figura següent de la classe *User*.

```
<?xml version="1.0" ?>
<serializer xmlns="http://symfony.com/schema/dic/serializer-mapping"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://symfony.com/schema/dic/serializer-
mapping
      https://symfony.com/schema/dic/serializer-mapping/serializer-mapping-
1.0.xsd"
>
   <class name="App\Snaptools\Authentication\Domain\Model\User">
      <attribute name="name">
         <group>main</group>
         <group>dashboard</group>
      </attribute>

      <attribute name="surname">
         <group>main</group>
         <group>dashboard</group>
      </attribute>

      <attribute name="uuid">
         <group>main</group>
      </attribute>

      <attribute name="email">
         <group>main</group>
         <group>dashboard</group>
      </attribute>

      <attribute name="roles">
         <group>main</group>
      </attribute>

      <attribute name="projects">
         <group>relations</group>
      </attribute>
   </class>
</serializer>
```

2.3.7.3 Swift Mailer

Durant el desenvolupament de la funcionalitat de recordar contrasenya es va decidir que el procés n'establis una nova si l'usuari no la recordava i se la notifiqués per correu.

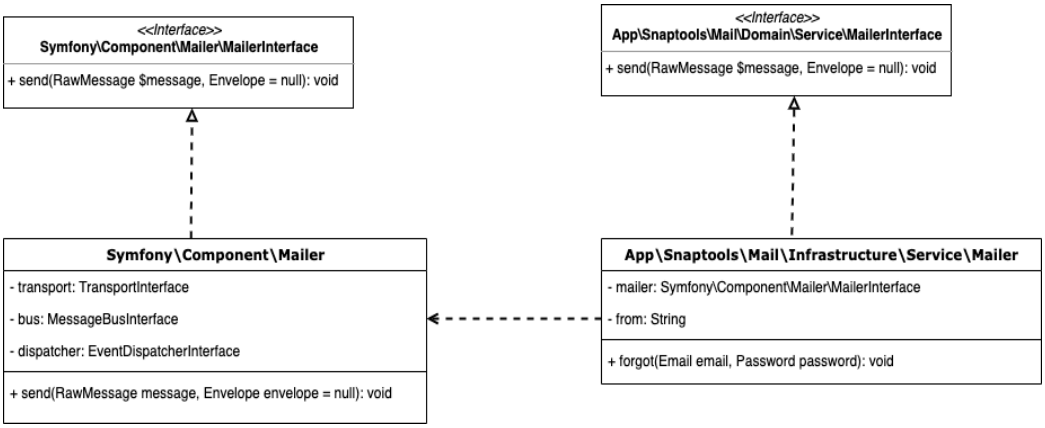
Per evitar tenir que instal·lar un servidor de correu local i aprofitant- que hi ha diversos proveïdors sota demanda que ofereixen el servei d'enviament de correus, es va decidir utilitzar *Sendgrid* [13] per la seva senzillesa.

La instal·lació del component es va fer a través del manegador de paquets *composer* [21].

```
composer require symfony/sendgrid-mailer
```

La pròpia plataforma ens va proporcionar una clau per utilitzar a l'hora de fer crides, que es va adjuntar a les variables d'entorn emmagatzemades al fitxer .env.local.

La integració es va dissenyar fet servir el patró adaptador, injectant la dependència externa del bastiment a un servei d'infraestructura de la nostra aplicació.

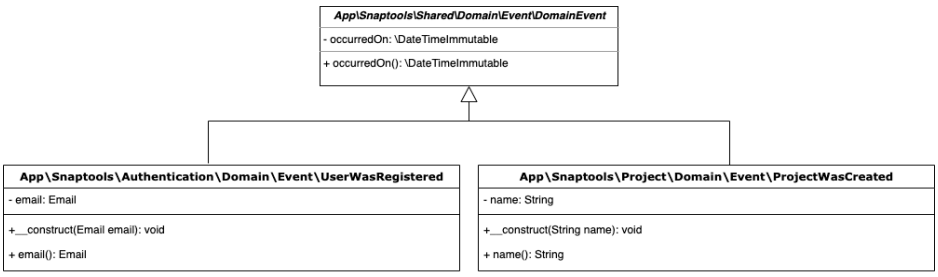


2.3.7.4 Expedidor d'esdeveniments

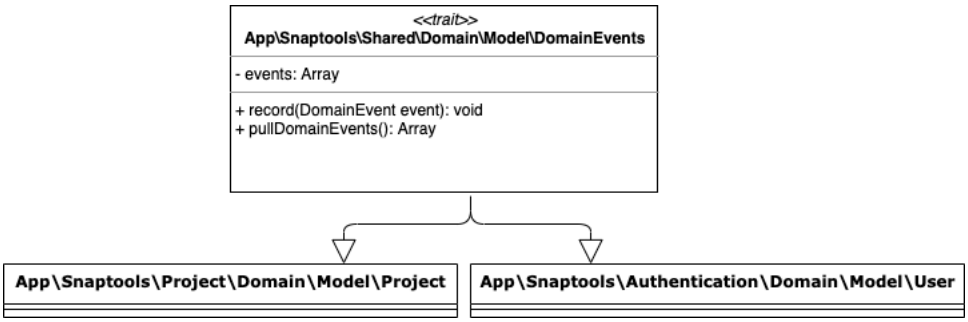
Un esdeveniment és un fet que ha succeït en el passat del que tenim una certa informació. Quan parlem en el context del programari, ens referim als fets que succeeixen dintre del domini i que altres parts del mateix tenen en compte perquè hi reaccionen d'alguna manera.

Elaborar una pantalla amb estadístiques de la plataforma era força costós computacionalment si cada cop que hi accedíem havia de calcular totes les dades a mostrar de nou, per aquest motiu es va decidir crear un model separat d'estadístiques que fos actualitzat a mesura que s'anessin generant nous esdeveniments.

A continuació podem veure el disseny en *UML* de les classes d'esdeveniments creades. Podem apreciar que estenen d'una classe abstracta que conté les propietats i mètodes comuns.

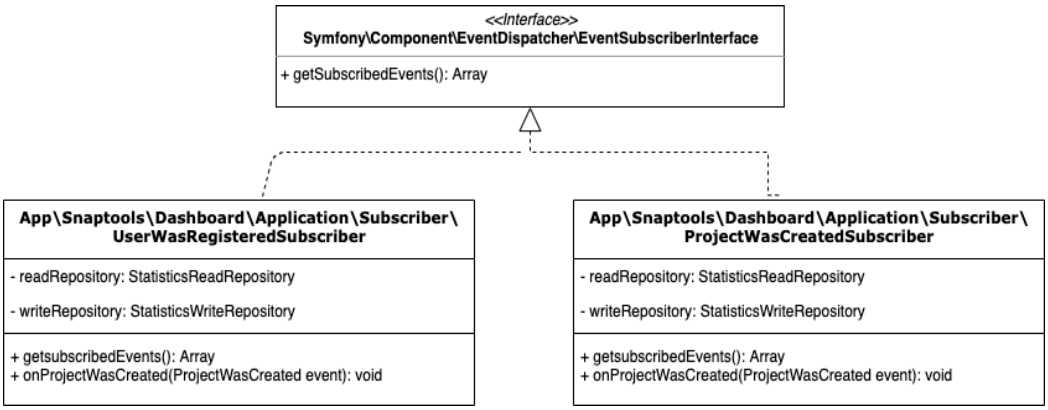


Cada entitat disposa d'una matriu d'elements on pot anar guardant els esdeveniments que van succeint, per tant té la responsabilitat d'emmagatzemar-los.



Els esdeveniments de domini son llençats per el manegador que llegeix de cada entitat els elements que conté i els expedeix fent servir el component corresponent.

El expedidor d'esdeveniments proporcionat per *Symfony* [20] està dissenyat fent servir el patró observador, per tant les tasques que contenen la lògica de domini que s'executa son classes subscrites als esdeveniments de domini corresponents.



2.3.8 Processador projectes

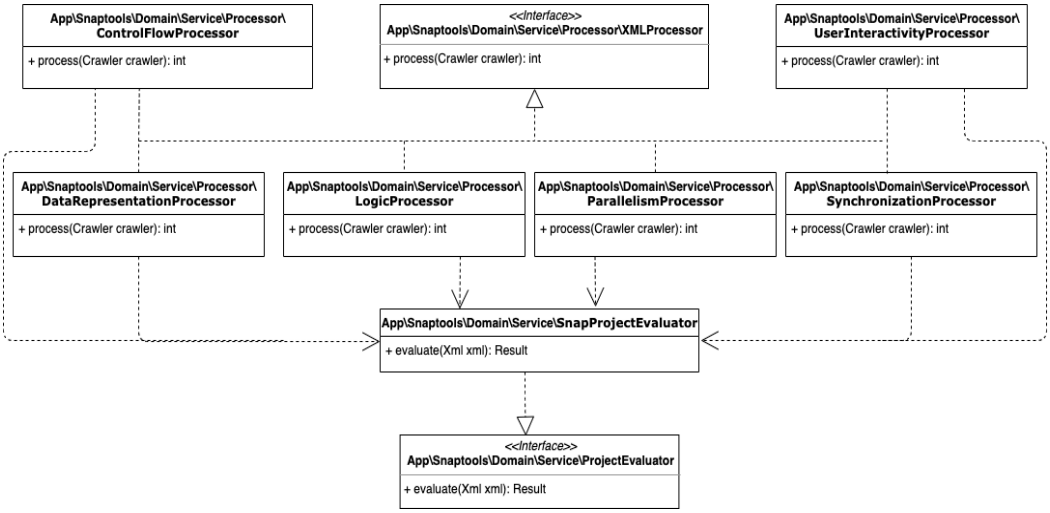
El servei principal té la responsabilitat de generar un objecte *DomCrawler* amb el *xml* del projecte i cridar a tots els processadors enviant una copia del mateix. Un cop ha rebut totes les puntuacions retorna un objecte del tipus *Result* que pot ser afegit al projecte.

Cada processador implementa una interfície comú *XMLProcessor* que l'obliga a retornar un valor enter que representa la puntuació de cada apartat.

El mètode per obtenir les puntuacions es basa en inspeccionar el *xml* amb diferents filtres, perquè cadascun dels blocs que es generen en la interfície web d'*Snap!* [19] té una representació en aquest fitxer.

Aquest component es va instal·lar mitjançant el gestor de paquets *composer* [21] com es pot veure a continuació.

```
composer require symfony/dom-crawler
composer require symfony/css-selector
```



2.3.9 Interfície de consola

Davant la necessitat de poder gestionar quins usuaris de la plataforma poden accedir a la pantalla d'estadístiques reservada als administradors, es va optar per afegir dues comandes de consola per realitzar aquestes operacions.

2.3.9.1 Afegir administrador

Per afegir un usuari com administrador només cal cridar la comanda de consola passant com a primer paràmetre el correu de l'usuari.

En cas de que el correu especificat no existeixi a la base de dades la comanda ho indicarà en la seva sortida. Si l'usuari existeix llistarà els permisos que té un cop afegit el d'administrador.

```
www-data@ubuntu-s-1vcpu-1gb-fra1-01:/srv/host/github/snaptools-uoc/snaptools-backend$ bin/console app:authentication:admin:grant invalid@email.com
User does not exist.
www-data@ubuntu-s-1vcpu-1gb-fra1-01:/srv/host/github/snaptools-uoc/snaptools-backend$ bin/console app:authentication:admin:grant carnauro@uoc.edu
User: carnauro@uoc.edu Roles: ROLE_USER, ROLE_ADMIN
```

2.3.9.2 Treure administrador

Per treure un usuari com administrador només cal cridar la comanda de consola passant com a primer paràmetre el correu de l'usuari.

En cas de que el correu especificat no existeixi a la base de dades la comanda ho indicarà en la seva sortida. Si l'usuari existeix llistarà els permisos que té un cop eliminat el d'administrador.

```
www-data@ubuntu-s-1vcpu-1gb-fra1-01:/srv/host/github/snaptools-uoc/snaptools-backend$ bin/console app:authentication:admin:revoke invalid@email.com
User does not exist.
www-data@ubuntu-s-1vcpu-1gb-fra1-01:/srv/host/github/snaptools-uoc/snaptools-backend$ bin/console app:authentication:admin:revoke carnauro@uoc.edu
User: carnauro@uoc.edu Roles: ROLE_USER
```

2.3.10 Proves unitàries

Una part important del desenvolupament de programari és la validació de que el codi fa el que s'espera d'ell en cada situació, doncs això es una garantia de qualitat i ens dona la seguretat de que el nombre de fallades en el nostre programari serà menor.

Per una banda, l'objectiu principal en aquest apartat va ser cobrir el major nombre de casos d'ús, certificant que la lògica de negoci del projecte era la correcte. Per l'altra, va ser escriure proves eficients que no recaiguessin excessivament en la tècnica de *mocking*.

En aquest sentit es pot veure per exemple que es va optar per simular els repositoris de dades amb implementacions *fake* dels mateixos, àmpliament utilitzats en els manegadors de diversos casos d'ús.

Aquest tipus d'estratègia ens va portar a tenir proves que depenen menys del codi a provar, que son més fàcils d'entendre i que tenen un cost computacional menor.

Existeixen diferents llibreries externes que proporcionen utilitats per a poder isolar i provar el nostre codi de manera eficient d'entre les quals destaca el bastiment de proves *PHPUnit* [11].

Aquesta és la solució per defecte en el ecosistema *PHP* [10] i es la que es va fer servir en aquest projecte.

Es va procedir a la instal·lació fent servir el gestor de paquets *composer* [21] com és habitual. En aquest cas però es va afegir que només fos requerit en l'entorn de desenvolupament ja que a producció no es passen les proves unitàries.

```
composer require --dev symfony/phpunit-bridge
```

Totes les proves automatitzades estan ubicades a la carpeta tests i s'encarreguen de validar unitàriament els casos d'ús de cada context i la lògica que segueix cadascun dels processadors del *xml* que representa el projecte *Snap!* [19].

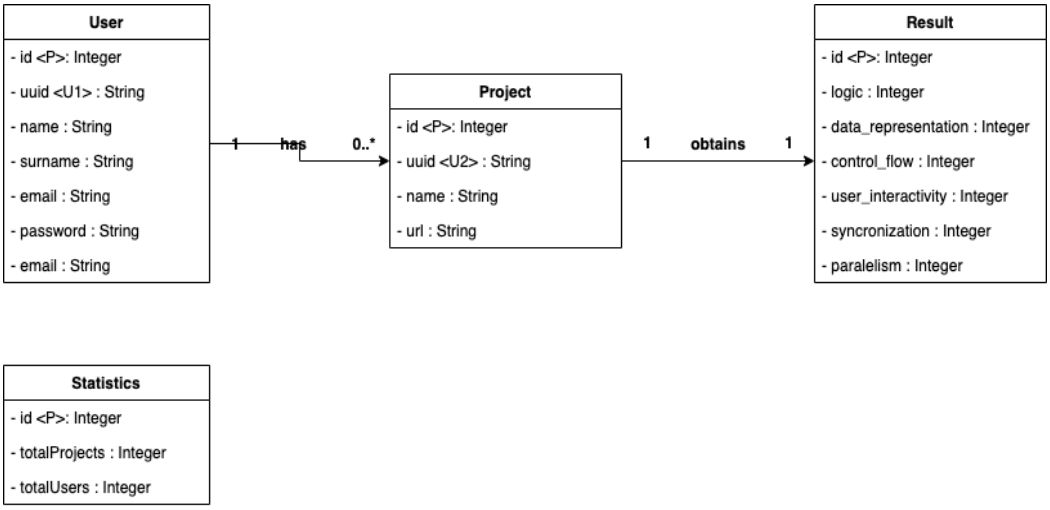
En la següent imatge es pot veure l'execució de totes les proves de la plataforma.

```
bin/phpunit tests --configuration phpunit.xml
```

```
Testing started at 18:04 ...
[docker://snaptools-backend_php:latest/]:php /srv/host/github/snaptools-uoc/snaptools-backend/bin/phpunit --configuration
PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Testing /srv/host/github/snaptools-uoc/snaptools-backend/tests


Time: 3.06 seconds, Memory: 10.00 MB

OK (52 tests, 92 assertions)
Process finished with exit code 0
```

2.3.11 Base de dades

2.3.11.1 Model conceptual



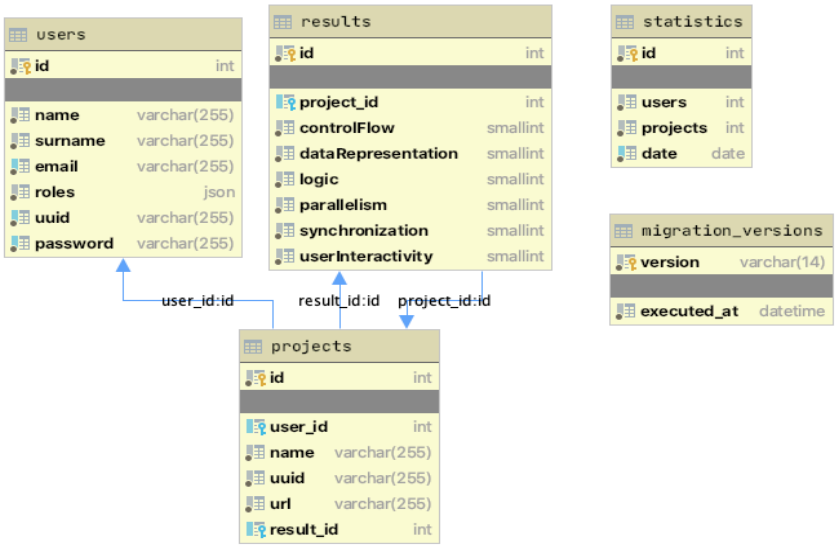
2.3.11.2 Entitat Relació

Els bastiments més actuals ofereixen la possibilitat de treballar amb llibreries *ORM*(*Object Relational Mapping*) que son capaces de associar classes escrites en codi amb objectes de base de dades de manera transparent.

Aquest tipus programari simplifica les tasques associades a la persistència de dades i és per això que és habitual treballar-hi en projectes de tota mida.

En aquest projecte s'ha optat per utilitzar *Doctrine* [15] i per aquest motiu no és necessari generar cap tipus de model entitat-relació.

A continuació es mostra el model generat automàticament i que es pot mostrar mitjançant el entorn de desenvolupament.



2.3.11.3 Doctrine Migrations

Durant el cicle de vida del projecte serà necessari dur a terme cada versió de l'esquema de la base de dades, atès que anirà canviant a mesura que es desenvolupen les noves funcionalitats.

La gestió dels canvis serà realitzada per un component extern que és capaç de sincronitzar les definicions existents al codi amb l'estructura a base de dades.

Tant la detecció dels canvis com la generació de noves versions és un procés no automàtic que ha de ser llançat per el programador durant el desenvolupament.

Per a portar el control de quines versions han estat aplicades hi ha una taula que conté el seu nom i el moment en el que van ser executades. Això és de força utilitat perquè automatitza el desplegament dels canvis en els entorns en el que s'ha d'executar l'aplicació.

A continuació podem veure la comanda que proporciona el component per detectar els canvis que existeixen entre les classes i l'estat actual de la base de dades.

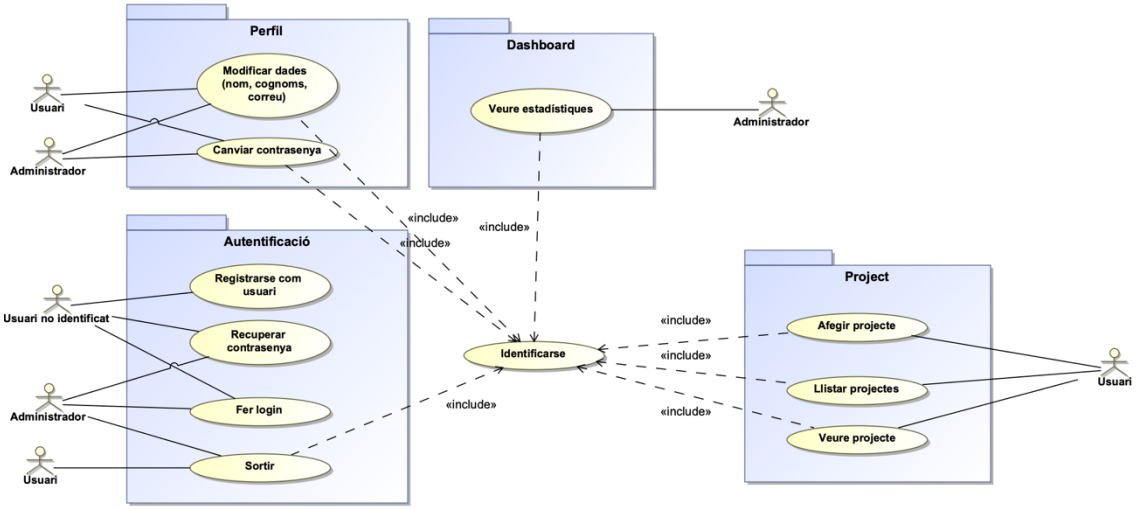
```
bin/console doctrine:migrations:diff
```

En cas de voler aplicar aquest canvis a la base de dades podem executar la següent comanda.

```
bin/console doctrine:migrations:diff
```

2.4 Aplicació

2.4.1 Casos d´ús



2.4.2 Mètriques Snap!

És un llenguatge de programació visual que permet crear programes fent servir una interfície *drag-and-drop* en comptes d'escriure amb el teclat. Un cop finalitzat el nostre projecte ens dona la possibilitat d'exportar-lo en format *xml*.

En els següents apartats veurem les diferents mètriques que avaluarà el nostre projecte i la puntuació que rebrà el programa en cada cas. S'ha decidit establir quatre nivells jeràrquics en funció de la correctesa del programa en cada apartat, en un intent de simplificar la comprensió dels resultats.

Tot i que hi ha forces més mètriques que poden ser avaluades, he considerat que aquestes son les més importants tant per l'usuari com per a un possible avaluador.

2.4.2.1 Lògica

Les instruccions lògiques ajuden a que els projectes es comportin de forma diferent en funció del context, ajudant a que siguin més dinàmics.

Nivell	*Descripció*
Nul	No hi ha instruccions de control.
Baix	Hi ha un o diversos blocs 'si'.
Mig	Hi ha un o diversos blocs 'si no'.
Alt	Hi ha un o diversos blocs si/si no amb operacions 'and' o 'or'.

2.4.2.2 Representació de dades

Existeixen diversos mecanismes per emmagatzemar informació al nostre projecte, que permeten desar diferents tipus de dades que podem necessitar durant l'execució del nostre programa.

Nivell	*Descripció*
Nul	No modifica atributs de blocs.
Baix	No defineix variables.
Mig	Fa servir variables per emmagatzemar dades.
Alt	Fa servir llistes per emmagatzemar dades.

2.4.2.3 Paral·lelisme

El paral·lelisme és la realització de múltiples accions en al mateix temps, o la possibilitat de que diversos esdeveniments succeeixin alhora.

Nivell	*Descripció*
Nul	No hi ha programes que s'executen en paral·lel.
Baix	Hi ha blocs que comencen a l'hora.
Mig	Succeeixen diverses circumstàncies quan l'usuari prem una tecla.
Alt	El mateix que l'anterior però ho fa amb diferents blocs diversos cops.

2.4.2.4 Sincronització

Parlem de sincronització quan dos o més processos succeeixen en un ordre predefinit o a l'hora.

Nivell	*Descripció*
Nul	No fa servir el bloc espera.
Baix	Fa servir el bloc 'espera'.
Mig	Fa servir missatges.
Alt	Fa servir 'espera'.

2.4.2.5 Interactivitat de l'usuari

La interactivitat permet a l'usuari que està executant el projecte realitzar accions que provoquin noves situacions en el projecte.

Nivell	*Descripció*
Nul	No hi ha interactivitat.
Baix	El codi conté esdeveniments.
Mig	Quan succeeix un esdeveniment s'executa alguna acció.
Alt	Fa servir la càmera o el micròfon quan succeeix un esdeveniment.

2.4.2.6 Control de flux

Els blocs que controlen el flux permeten modificar el flux d'execució fent que repeteixin diverses accions un nombre de vegades concret o fins que es compleixi una determinada condició.

Nivell	*Descripció*
Nul	Els blocs no s'executen un darrera l'altre.
Baix	Els blocs s'executen un darrera l'altre.
Mig	L'anterior i a més fa servir instruccions de repetició.
Alt	Fa servir el bloc 'repetir fins que'.

3. Conclusions

La realització d'aquest treball final de grau ha comportat nombroses hores de planificació, disseny i codificació per arribar al resultat final.

Tenint en compte que posseeixo experiència en el desenvolupament web, un dels meus objectius personals a l'hora de realitzar aquest projecte era el poder aprofundir els meus coneixements en totes les fases que comporta la creació d'un producte des de zero, perquè amb l'especialització que hi ha al món laboral cada cop és més difícil compaginar diverses àrees a l'hora.

Aquesta decisió, juntament amb la dedicació del meu treball a un domini desconegut fins al moment per a mi, m'ha portat a sortir fora de la meva zona de confort i trobo que ha estat molt encertada.

Durant la seva realització he tingut oportunitat de ampliar els meus coneixements d'arquitectura de programari, dels diferents processos externs que faciliten el desplegament d'aplicacions i d'altres eines que envolten el desenvolupament que em semblen força interesants.

Com comento anteriorment, el fet d'escollir un domini que no coneixia ha fet també que una part del temps l'hagi dedicat a aprendre el funcionament d'una aplicació nova per a mi, així com també ha estat necessari entendre el format de sortida dels seus fitxers.

Per el que respecta als objectius personals puc dir que l'experiència ha estat força enriquidora, sense deixar de banda que gran part dels objectius del treball també han estat assolits.

L'objectiu principal ha estat sempre el desenvolupament i posada en marxa d'una aplicació a Internet capaç de processar projectes en diferents àrees del aprenentatge computacional, objectiu que s'ha complert de forma satisfactòria.

N'hi ha de molts altres que també s'han assolit de forma satisfactòria però m'agradaria centrar-me en aquells que per diversos motius no s'han arribat a aconseguir.

La idea inicial va ser desenvolupar la part del client web amb una plantilla que havia de modificar per que complís amb els requisits del projecte, però la combinació de poca experiència en entorns *frontend* juntament amb tenir-la que fer funcionar amb *Docker* va portar-me molts problemes fins que vaig decidir canviar a una altra per no comprometre el projecte.

Aquesta situació va fer que portés una mica de retràs perquè vaig haver de codificar una bona part de nou, que va comprometre altes tasques que tenia interès a fer però no vaig tenir temps, com la posada en marxa del portal a producció amb *Docker*.

Un cop arribats a aquest punt m'agradaria ressaltar que tot i que la metodologia en cascada que s'ha fet servir ha estat adient per assolir el projecte, crec que una metodologia àgil com SCRUM m'hauria ajudat més en aquest o qualsevol altre contratemps.

De totes formes he complert amb totes les fites com estava marcat a la planificació temporal així que puc dir que estic satisfet amb la decisió.

Respecte a les línies de treball futur m'agradaria poder treballar en la interfície d'usuari, probablement amb alguna llibreria *frontend* més moderna que permeti una reutilització millor que la que hi ha actualment.

En la banda de servidor seria convenient poder oferir més funcionalitats, com per exemple la possibilitat de tenir versions d'un mateix projecte o inclús ampliar i millorar les mètriques associades.

Com a nota final m'agradaria recalcar que la realització d'aquest treball ha estat una experiència molt enriquidora per a mi, sobre tot perquè es tractava d'un projecte que un cop acabat tindria vida pròpia.

Tot i que aquest fet pot semblar poc important, obre la porta a un nou cicle de desenvolupament centrat en la observació, recerca, millora i manteniment de programari que pot ser força interessant per a mi.

4. Glossari

token: cadena de caràcters codificada en base-64.

git flow: flux de treball per treballar amb git.

https: Protocol de connexió segura

repositori: lloc central on les dades son guardades i gestionades.

HTML: Acrònim de HyperText Markdown Language, és el llenguatge de marques per a pàgines web.

DOM: Acrònim de Domain Object Model, és la representació en arbre dels objectes que representen una pàgina web.

AJAX: Acrònim de Asynchronous JavaScript and XML, és una tècnica que permet recarregar selectivament certes dades d'una pàgina.

JSON: Acrònim de JavaScript NOtation, és un format estàndard per representar objectes d'aquest llenguatge.

Plugin: Afegit de programari.

SOAP: Acrònim de simple Object Access Protocol, és un protocol que defineix com dos processos poden comunicarse mitjançant XML.

CORS: mecanisme que permet sol·licitar recursos restringits en una pàgina web desde un domini fora del domini que va servir el primer recurs.

5. Bibliografia

[1] Docker - https://www.docker.com

[2] Iptables - https://es.wikipedia.org/wiki/Iptables

[3] Nginx - https://nginx.com

[4] Ubuntu - https://www.ubuntu.org

[5] UFW - https://www.digitalocean.com/community/tutorials/como-configurar-un-firewall-con-ufw-en-ubuntu-18-04-es

[6] Certbot - https://certbot.eff.org

[7] MySQL - https://www.mysql.com

[8] Github - https://github.com

[9] Github Actions - https://github.com/features/actions

[10] PHP - https://www.php.net

[11] PHPUnit - https://phpunit.de

[12] Javascript - https://developer.mozilla.org/es/docs/Web/JavaScript

[13] Sendgrid - https://sendgrid.com/

[14] Git - https://git-scm.com/

[15] Doctrine - https://doctrine-project.org

[16] Apache - https://httpd.apache.org

[17] macOS - https://es.wikipedia.org/wiki/Mac_OS

[18] JWT - https://jwt.io

[19] Snap! - https://snap.berkeley.edu

[20] Symfony - https://symfony.com

[21] Composer - https://getcomposer.org/

[22] AdminThemes - https://shorturl.at/koIUW

[23] DigitalOcean - https://www.digitalocean.com/

[24] Metronic - https://shorturl.at/bgtJL

[25] JQuery - https://jquery.com/