Gymky.me

Treball Final de Grau - Desenvolupament d'aplicacions interactives

Xavier Vilà i Albiol
UOC - Grau en Multimedia
Consultor: Kenneth Capseta Nieto
Professor: Carlos Casado Martinez
Gener de 2016

Abstract

Gymky.me és una aplicació web que permet la creació de tasques, les quals seran completades per part d'uns usuaris que competeixen en diferents grups en una mena de gimcana virtual.

L'aplicació està enfocada per a un target d'usuaris força ampli, ja siguin grups de treballadors que vulguin realitzar activitats conjuntes per enfortir dinàmiques de grup, com per a grups d'amics que vulguin realitzar una gimcana per passar l'estona.

L'aplicació s'estructura de forma en que l'administrador, que serà un usuari que haurà d'haver-se registrat en l'aplicació, genera les tasques a realitzar, a més a més d'introduir els usuaris que hi participaran. Un cop les tasques han estat generades, els usuaris participants rebran una notificació on se'ls informa de la data de l'activitat.

Tècnicament l'aplicació està desenvolupada íntegrament en javascript emprant la tecnologia MEAN, la qual permet desenvolupar tots els aspectes de l'aplicació en el mateix llenguatge a la vegada que facilita la migració d'aquesta cap a una aplicació mòbil.

Paraules clau: Gymky.me, aplicació web, tasques, treballadors, activitats, amics, dinàmiques de group, javascript, MEAN.

Abstract (english version)

Gymky.me is a web application that allows creating tasks, which are going to be completed by the users who compete with each other's, something like a virtual gymkhana.

The app is focussed to a huge type of users, which could be co-workers that want to make activities in group in order to team building, or friends who want to make a gymkhana to enjoy together their spare time.

The application is structured in a way where the admin, who has registered previously, creates the tasks that are going to be completed, as well as creates the future users. When the tasks would have been created, the users will receive a notifications with the activity's date.

Technically, the app is developed integrally in javascript using MEAN tech, which allows creates the whole project using the same programming language, besides it helps to migrate from a web app to a mobile app.

Keywords: Gymky.me, web app, tasks, co-workers, activities, friends, team building, javascript, MEAN.

Index

1. Introducció

Alhora d'afrontar el Treball final de Grau, desenvolupar una aplicació que tingués més recorregut que no solament l'entrega del propi treball era un dels principals objectius, és per això que el treball va enfocat en la implementació d'una aplicació real dirigida a usuaris reals i amb l'objectiu de llançar-la a un mercat real.

D'entrada la idea del projecte sorgeix de l'observació de certes dinàmiques de grup que es duen a terme en moltes companyies de desenvolupament de software on seguint una certa filosofia de treball "nord-americana", es duen a terme activitats que pretenen crear dinàmiques de grup, "*team building*", que persegueixen crear complicitats i inèrcies positives entre els treballadors.

Aquestes activitats que poden ser tant sortides en grup com activitats guiades, sempre tenen com a punt en comú la participació de tots els membres de l'equip, cercant que la confiança i la solidaritat entre ells es consolidi i que posteriorment generin una millora en el treball diari del grup.

Partint d'aquest concepte i alhora ampliant-ne el focus, el projecte pretén desenvolupar una aplicació web i posteriorment una de mòbil, que permeti la creació d'aquesta mena d'activitats, jocs o gimcanes, de forma dinàmica i senzilla, i que pugui ser emprada tant per empreses com per usuaris particulars.

2. Descripció

Gymky.me pretén ser una aplicació que permeti la creació d'activitats interactives, on els usuaris vagin superant una serie de "proves" plantejades des de l'aplicació amb l'objectiu de batre un equip rival amb el qual estan competint.

El flux de treball de l'aplicació s'inicia amb el registre de l'administrador del joc, aquest serà l'encarregat de la planificació de les activitats, l'organització dels equips així com de la seva inscripció. Per a la creació d'aquestes tasques l'administrador disposarà d'un planell d'administració que li oferirà totes les eines necessàries per a la seva implementació i des d'on organitzarà tot el "workflow" de l'aplicació.

De forma dinàmica podrà anar creant els diferents passos que hauran de seguir els usuaris per completar les tasques. Aquestes podran ser tant simples preguntes o endevinalles, com proves físiques que hauran de superar i justificar el seu compliment mitjançant l'enviament de fotografies o vídeos a l'aplicació.

L'administrador, també serà l'encarregat d'organitzar els diferents equips, registrant-ne els membres, introduint-ne el seu corresponent correu electrònic, el qual, quan el joc s'inici, permetrà als usuaris accedir als continguts de la seva activitat.

Un cop s'hagi completat la creació de les diferents proves o activitats, els usuaris que l'administrador haurà introduït rebran un correu electrònic amb tota la informació detallada per tal d'accedir a l'aplicació i poder iniciar l'activitat.

Alhora d'iniciar l'activitat els membres de cada grup podran accedir a les seves tasques via web o aplicació mòbil i hauran d'anar completant els diferents passos fins aconseguir l'objectiu final.

Finalment, i amb una visió més comercial, es podria encarar el projecte cap a una idea de negoci que anés enfocada a que alguna empresa s'encarregui, mitjançant l'aplicació, de la creació de les activitats i les ofereixi a empreses que vulguin realitzar aquest tipus d'activitats de forma planificada, sense tenir que encarregar-se de res més. Aquesta versió de l'aplicació, que seria de pagament, podria cobrir unes necessitats que possiblement aniran en augment en certs sectors laborals. Òbviament aquesta versió de l'aplicació queda fora del que es pretén amb el TFG, sense deixar d'ésser un dels objectius potencials del projecte.

Pel que fa a les tecnologies que s'empraran s'opta per desenvolupar tota l'aplicació en javascript, utilitzant la tecnologia MEAN. Bàsicament al desenvolupar una aplicació d'aquesta manera s'utilitza Node.js, Express, Mongodb i Mongoose per al backend i Angular com a framework MVC per a tota la lògica del frontend.

Aquesta decisió tècnica a més a més de possibilitar el desenvolupament de tota l'aplicació emprant el mateix llenguatge de programació, facilita força la posterior migració cap a una aplicació mòbil, que tot i que no està previst dins l'àmbit del TFG si que es considera com l'evolució natural del projecte.

3. Objectius

Principals

- Desenvolupar una aplicació web que permeti la creació de tasques.
- Generar una base de dades que permeti emmagatzemar els continguts de l'aplicació, tant les activitats, com els usuaris com els diversos estats de cada activitat.
- Generar una API que permeti obtenir els diferents continguts de l'aplicació.
- Implementar un registre d'usuaris (administradors) que permeti que aquests accedeixin al planell d'administració per tal de gestionar l'aplicació.
- Implementar un planell d'administració des d'on es pugui gestionar la creació de les tasques per part de l'administrador de cada activitat.
- Associar les tasques desenvolupades a un seguit d'usuaris i agrupar-los en equips.
- Generar notificacions, via correu electrònic, que permetin informar als usuaris del l'estat de l'activitat en que estan registrats així com accedir a l'aplicació quan l'activitat sigui activa.
- Generar un "look and feel", coherent i atractiu, per a una aplicació d'aquesta mena.

Secundaris

- Implementar l'aplicació mòbil, aprofitant tant el "backend" com el "frontend" de l'aplicació web, reduint així les necessitats de manteniment i de correcció de *"bugs"*.
- Implementar geolocalització en l'aplicació mòbil per tal d'augmentar les possibilitats alhora de crear proves o activitats.

4. Continguts

Com ja s'ha descrit en punts anteriors Gymky.me pretén ser una aplicació que permeti la creació de tasques o activitats, que els usuaris hauran d'anar completant, alhora que competeixen amb un altre equip per realitzar-les primer. Bàsicament l'usuari que realitza el rol de creador de les tasques serà l'encarregat de la creació de les activitats i dels diferents equips i de la creació dels perfils dels usuaris que hi participaran. Un cop les activitats estan creades els usuaris podran iniciar el joc que consistirà en anar completant les tasques, ja siguin qüestions o proves físiques, fins a completar-les totes.

Si s'analitza tot el flux de l'aplicació s'observa que aquesta s'inicia amb el registre per part de l'administrador en l'aplicació. Aquest registre amb el rol d'administrador, permet l'accés d'aquest al planell d'administració, des d'on es pot procedir a la creació de les diferents tasques que conformen l'activitat.

En aquest planell l'usuari podrà introduir el nom de l'activitat a crear, la data en que es realitzarà, una imatge o avatar de l'activitat i els equips que hi participaran. Seguidament és procedirà a la creació de les diferents tasques, escollint entre diferents estructures que ja estaran dissenyades per defecte, que podran ser del tipus, preguntes o endevinalles amb les possibles respostes, de localització on haurà d'introduir un posició en el mapa amb algunes pistes per poder-hi arribar, o bé pujar una imatge per demostrar que s'ha realitzat una activitat descrita en l'enunciat de la tasca.

Posteriorment a la creació de l'aplicació, els usuaris que hauran estat registrats per l'administrador, rebran una notificació en forma de correu electrònic on se'ls informarà de la data de l'activitat, així com de les dades per poder accedir-hi.

El dia marcat per a la realització de l'activitat els usuaris podran accedir-hi, i anar realitzant les diferents tasques fins a completar-les. Un cop finalitzada l'activitat aquesta quedarà registrada i podrà ser consultada des de la web, poden organitzar classificacions amb els diferents resultats dels diferents usuaris.

5. Metodologia

El desenvolupament d'un projecte d'aquesta mena i amb aquestes característiques requereix una organització força acurada i pautada, en que estigui planificat tot el procés de desenvolupament del projecte i així poder respectar els terminis d'entrega.

Per tal d'aconseguir complir amb aquests principis, es considera que una metodologia força adequada, i que serà la utilitzada per al TFG, és la metodologia *"Agile"*, la qual està indicada per als processos de implementació de software.

Aquesta metodologia, que es òptima per al treball en equip, es basa en el desenvolupament parcial de petites parts del projecte. Per tant inicialment es dividirà el projectes en petites tasques a realitzar que es desenvoluparan fins a que totes elles siguin completades.

Aquesta divisió del treball en petites tasques, permet fer una estimació més realista de la carrega de treball, alhora que també facilita el control de l'estat del projecte així com de les seves debilitats.

Existeixen diverses eines o aplicacions que permeten l'ús d'aquesta metodologia, però bàsicament totes es basen, en crear un llistat de tasques que es van movent pels diversos estats que conformen el cicle de vida de cada tasca. Inicialment hi ha un llistat amb totes les tasques a realitzar que anomenem *"backlog"*. Posteriorment es seleccionen un grup d'aquestes tasques a realitzar i l'espai temporal en que s'hauran de completar o també anomenat *"Sprint"*, el qual acostuma a ser de dues setmanes. Durant cada *"Sprint"* s'han de realitzar les tasques, testejar-les i finalment considerar-les com a completes, amb el que s'entén que ja poden ser emprades a producció, o sigui que ja estan llestes per a que els usuaris les utilitzin.

Per tal d'organitzar totes aquest procés i que gràficament es pugui identificar l'estat de cada tasca, aquestes s'acostumen a agrupar per columnes, on primerament hi apareixen les tasques a realitzar. Quan es vulgui realitzar una d'aquestes tasques s'ha de moure cap a una columna on hi ha les que s'estan desenvolupant en aquell moment. Un cop es considera que la tasca ja està realitzada, s'ha de moure a la columna on hi ha les tasques que han de ser testejables i finalment quan ja es considera que la tasca està completada i que funciona correctament, serà moguda a la columna de completades. En cas de que alguna de les tasques no superés el testeig i per tant no estigués adequadament realitzada s'hauria de moure de nou a la columna de tasques a realitzar i començaria de nou el seu cicle.

Bàsicament tot aquest procés requereix un treball extra, però garanteix un control sobre la feina diària a la vegada que també permet supervisar la qualitat del producte que s'està desenvolupant i per tant minimitzar els potencials errors.

6. Arquitectura de l'aplicació

L'arquitectura de l'aplicació està desenvolupada tota amb javascript emprant la tecnologia MEAN.

Aquesta Arquitectura basa la seva estructura en l'ús de les següents tecnologies:

Client

Pel que fa al client el contingut està estructurat emprant *HTML5* i *CCS3* pel *look and feel*, concretament *SASS* que permet generar els estils de forma més ràpida i dinàmica, emprant variables, funcions, iteracions, etc...

Pel que fa a la lògica del client, s'emprarà el patró MVC, que aporta *Angular*, per tal d'organitzar més eficientment el codi javascript, alhora que s'implementa un codi font, més clar, net i escalable.

Tota la lògica de l'aplicació, la comunicació amb la base de dades, l'actualització de les vistes, la interacció amb l'usuari estarà implementada del cantó del client, per tant la única interacció que es realitzarà amb el "*backend*" serà per fer peticions a l'API Rest i que aquesta consulti la BBDD i obtenir-ne les dades que després seran gestionades des del frontend per ser mostrades a l'usuari.

Servidor

El servidor sera desenvolupat mitjançant *Nnginx i Node.js,* els quals s'encarregaran de gestionar les peticions des del web. Aquesta tecnologia permet la creació d'un servidor mitjançant javascript des d'on correrà tota l'aplicació, i des d'on s'implementarà l'API Rest que rebrà les peticions des del frontend. Tot aquesta lògica també s'organitzarà mitjançant el patró Model, Vista i Controlador (MVC).

Aquesta API tindrà definides unes rutes a les que s'accedirà des del frontend i que seguint el model *CRUD,* realitzaran les consultes a la BBDD i obtindran els continguts en format json, que *Angular* des del client s'encarregarà de renderitzar de forma adequada.

Des del servidor també s'emprarà el generador de plantilles *ejs*, que permetrà crear la pàgina inicial de l'aplicació des d'on *Angular* s'encarregarà de la gestió de les diferents vistes, a més a més de poder gestionar variables globals i diversos valors per controlar el flux de treball de l'aplicació. Per qüestions de seguretat les pàgines de registre i login es desenvoluparan emprant la mateixa estratègia.

Base de Dades

Per a la base de dades es treballarà amb *Mongodb*, que permet emmagatzemar les dades en format json, cosa que facilitat tot el flux de treball amb javascript. Els models de les bases de

dades, es defineixen emprant, *Mongoose* que és una mena d'*ORM* que permet la creació dels models, així com la seva validació i tota una serie de processos amb les dades.

7. Plataforma de desenvolupament

Per al desenvolupament de l'aplicació es requereixen dels següents elements:

- Un Macbook Pro com a maquina de desenvolupament.
- L'IDE PHPStorm per a la escriptura del codi i el desenvolupament de l'aplicació.
- El navegador Chrome, per a la visualització de l'aplicació, i la seva consola de desenvolupament per a testeig.
- Bitbucket com a repositori en remot, que ens permet realizar el control de versions de l'aplicació mitjançant git.
- Accés al repositori de mòduls npm, per tal d'accedir i poder descarregar i instal·lar tots els mòduls externs que es requereixen (Node, Express, Angular, Mongoose i tot un seguit de plugins que faciliten certes tasques, que han estat implementades per altres desenvolupadors i que són d'accés lliure)
- El servidor de BBDD Mongodb, MongoLab per a la creació de BBDD per tal d'accedir a dades reals i per tant poder observar el comportament de l'aplicació en un escenari real abans de que l'aplicació sigui pujada a un servidor de producció en remot. (mongolab.com)
- Finalment per al servidor de producció es disposa d'un Droplet a Digital Ocean, on es correrà sobre el sistema operatiu Ubuntu, emprant Nnginx i Node.js

8. Planificació

La planificació del projecte es basa en la metodologia de treball *Agile*, descrita anteriorment i per tant es dividirà tota l'aplicació en petites tasques que permetran fer una estimació real de la carrega de treball que s'haurà d'assumir. Aquestes tasques seran agrupades en diversos *"Sprint"* de dues setmanes que que haurien de coincidir amb les diverses PAC's que seran encarades.

Per tant a més a més de l'entrega de les corresponents PAC's s'haurà d'anar informant al consultor de l'evolució que anirà seguint en forma paral·lela, el cicle dels *"Sprints",* així com del llistat de tasques a realitzar en cada cicle.

Un cop realitzat el llistat de tasques a realitzar, aquestes s'intentaran agrupar pels objectius que es pretenen assolir en cada PAC i que es detallen seguidament.

Per a la PAC2 (28/10/2015), a més a més de tota la documentació requerida en la memòria, s'haurà de tenir implementat un prototip bàsic de l'aplicació, que haurà d'incloure les funcions bàsiques d'aquesta que són:

- La implementació de la Base de dades (Mongodb, Mongoose)
- Creació de la API, que permetrà l'accés a les dades (Node, Express, Mongoose, javascript)
- Creació del registre d'usuaris (Angular, Node, Express, Moongose)
- Desenvolupament del planell d'administració amb funcions bàsiques (Angular)

Aquests punts, estaran dividits en petites tasques que facilitaran que es pugui completar cada objectiu. Es contempla l'ús d'alguna eina que permeti compartir tota l'organització de les tasques amb el consultor.

De cara a la PAC3(29/11/2015), s'implementarà la documentació requerida així com les següents tasques:

- Creació dels estils de l'aplicació per tal d'implementar el "look and feel" escollit. (SASS)
- Finalització del planell d'administració
- Implementació de les notificacions
- Implementació de la interfície gràfica

Finalment, per a l'entrega final(11/01/2016), es completarà la memòria i es finalitzarà el la primera versió de l'aplicació així com la seva implementació al servidor de producció.

9. Procés de treball/desenvolupament

El procés de treball de l'aplicació s'inicia amb l'anàlisi de tots els requeriment i dependències que hi ha en el projecte. En una aplicació *Node.js*, s'han de definir totes les dependències que que hi ha en un arxiu anomenat package.json, on es defineix l'aplicació. Un cop implementat aquest arxiu, s'hauran d'instal·lar totes les dependències que hauran estat definides, i així tenir accés a tots el mòduls de tercers que es necessiten per al desenvolupament de l'aplicació.

Si es donés el cas que durant el desenvolupament de l'aplicació es necessites afegir algun mòdul extern, només s'haurà d'afegir a l'arxiu package.json i actualizar-lo corrent a la consola amb l'*script* corresponent. L'ús d'aquesta metodologia permet organitzar de forma molt precisa i organitzada les aplicacions, així com gestionar molt eficientment la seva escalabilitat.

Un cop l'entorn està definit, s'inicia el desenvolupament de l'aplicació amb la creació del modul de registre de d'usuaris, aquesta tasca implica la creació de la col·lecció Usuaris a la base de dades *Mongodb*, que és on s'aniran emmagatzemant els usuaris.

Per tal de poder realitzar aquesta tasca, s'ha de crear anteriorment, un servidor emprant *Node.js i Express*, des d'on correrà l'aplicació. Aquesta tasca es realitza de forma bastant ràpida emprant Express, ja que compta amb una serie de mètodes que en faciliten força el desenvolupament.

Un cop el servidor està enllestit, s'ha de crear l'API Rest que permet les peticions a la BBDD. Aquesta API que realitzarà les funcions bàsiques CRUD, permet el poder crear, obtenir, actualitzar i eliminar els registres de la bases de dades.

Un cop l'API està generada i s'han definit els Models de la BBDD emprant *Mongoose*, s'haurà de crear el *frontend* de l'aplicació, des d'on el client, ja sigui l'ordinador o un mòbil, es comunicarà amb l'API per poder interactuar-hi i registrar-s'hi i posteriorment accedir-hi.

Aquesta part del desenvolupament es realitzarà mitjançant *Angular*, que serà l'encarregat de gestionar tota la lògica de l'aplicació, des d'estar pendent de les accions dels usuaris, fer les peticions a l'API Rest, gestionar les dades obtingudes, renderitzar les vistes que pertoquin, etc..

D'altra banda i de forma paral·lela, també s'haurà de crear tot *l'HTML* i els estils que formaran l'aplicació. Aquest *HTML* estarà estructurat en petites vistes o arxius independents que mostraran una part de l'aplicació. D'aquesta manera es poden reutilitzar vistes per a diferents requeriments i mantenir el codi més net, organitzat i estructurat.

Aquest mateix procés és repetirà per a la creació de les activitats i dels altres mòduls que s'aniran implementant en l'aplicació. Per tant per a la creació de les activitats es seguirà el mateix procés creant la col·lecció Activitats a la BBDD i posteriorment implementant els mòduls corresponents a l'API així com els frontend que s'hi comunicarà des del client.

10. APIs utilitzades

Com s'ha comentat en el punt anterior, en un projecte Node.js hi ha moltes dependències, que s'obtenen principalment del repositori npm i que es gestionen quasi de forma automàtica des de l'arxiu package.json.

Seguidament se'n detallen les principals:

Node i Express
Permeten la creació del servidor que s'empra per fer correr l'aplicació així com la gestió de sessions i tot un seguit de mètodes de seguretat.

Mongoose i Mongodb
Permeten l'emmagatzematge de dades així com tota un sèrie d'accions sobre aquestes dades com validacions.

Gulp
És un *Task runner* que permet automatitzar tota una sèrie de processos que es van repetint duran el desenvolupament de l'aplicació, com per exemple, minificar i unificar els arxius javascript que es van creant, convertir els arxius SASS en CSS per poder-los utilitzar al navegador...

Angular
Permet organitzar el codi javascript seguint el patró MVC, a més de facilitar en gran manera les peticions a l'API Rest, el *bindatge* amb els elements del DOM del document web...

Passport
Facilita la implementació dels registres d'usuaris a més a més de permetre emprar altres API's com les de facebook o twitter.

SASS
És un pre-processador de CSS que permet crear els full d'estil de forma més ràpida, alhora que es poden emprar certs elements no disponibles en els CSS normals, com mètodes, variables, funcions...

Mocha

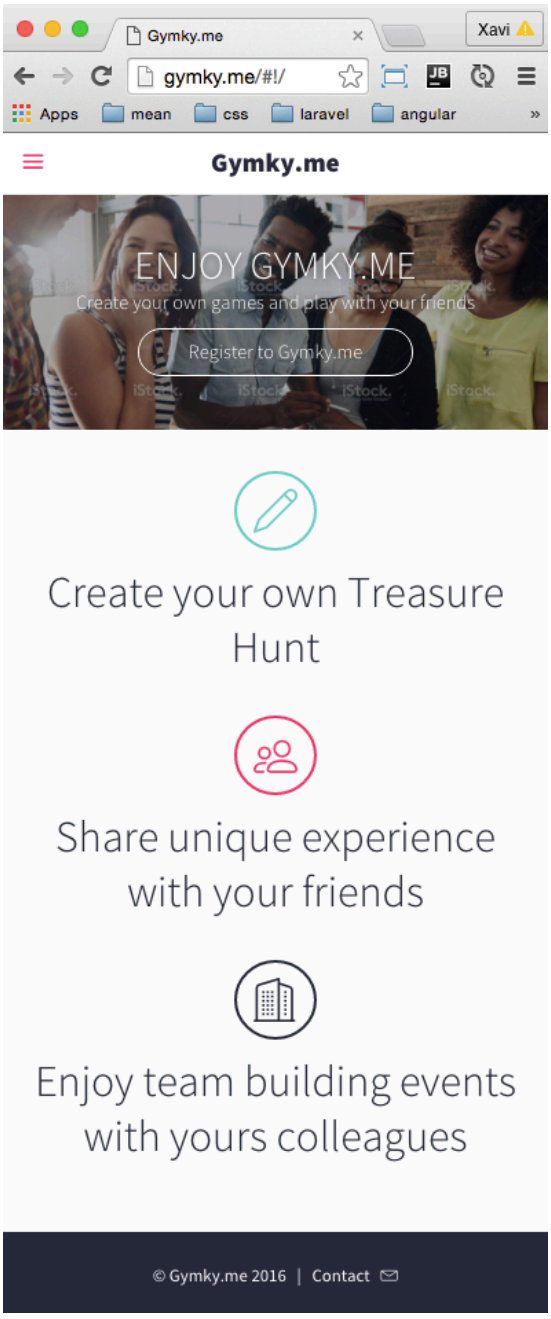
És un Framework de tests que permet testejar el codi javascript tant al frontend com a l'API Rest.

11. Prototips

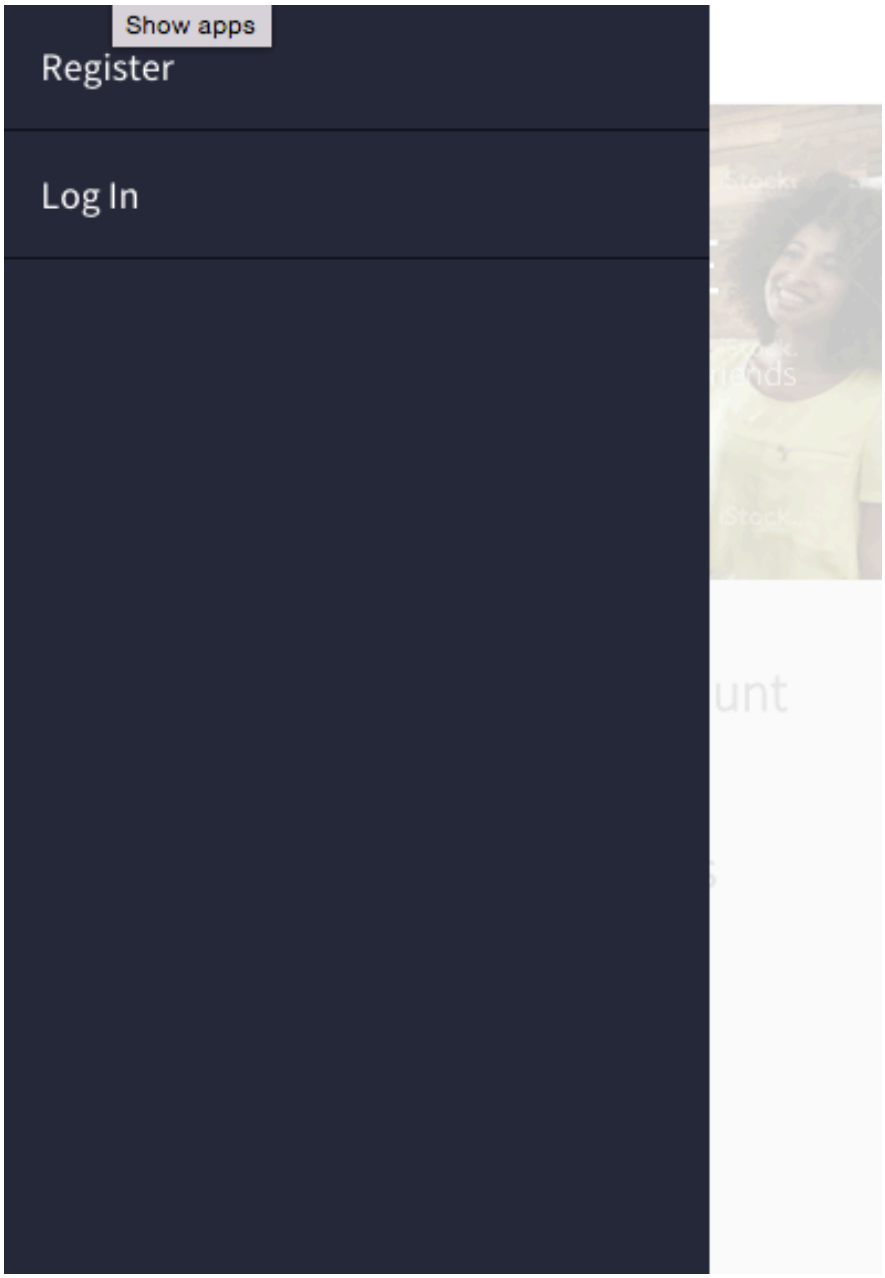
Pel que fa al prototipatge i com es veurà més desenvolupat en l'apartat d'Usabilitat, s'ha optat per la senzillesa i sobretot la intuïtivitat dels diferents flux de l'aplicació. Cal remarcar que el disseny segueix el process de desenvolupament gràfic, *Mobile First*, on s'inicia tot el disseny per les visualitzacions des de dispositius mòbils.

Seguidament podem observar alguna de les principals vistes de l'aplicació:

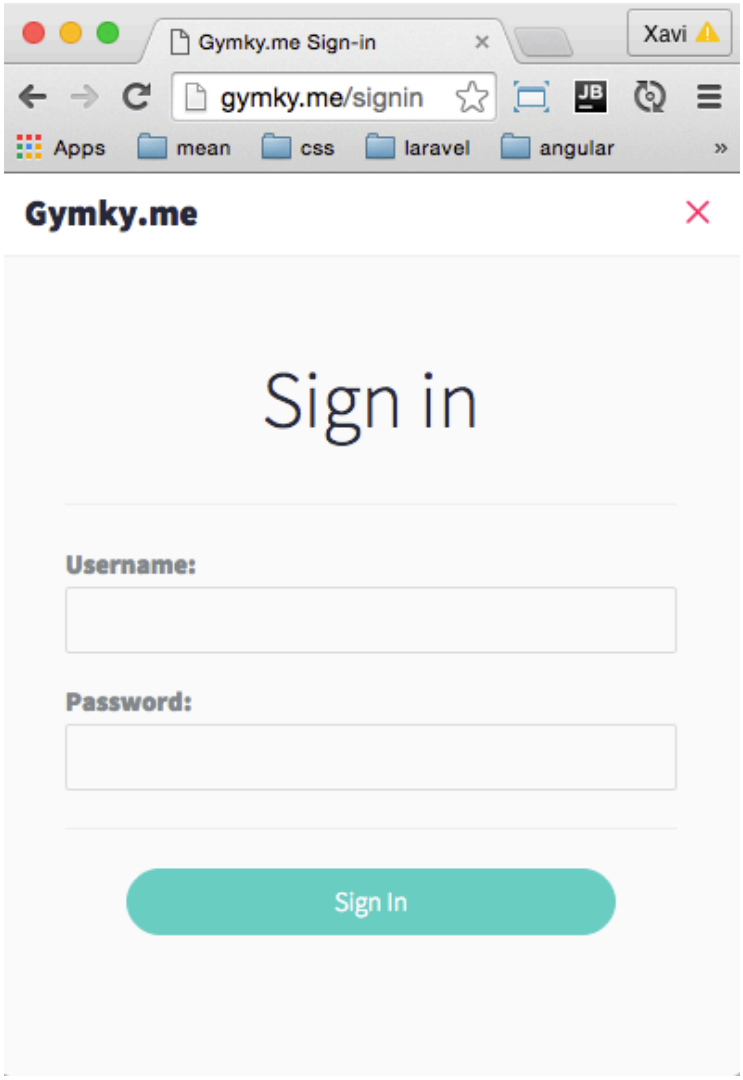
Landing Page



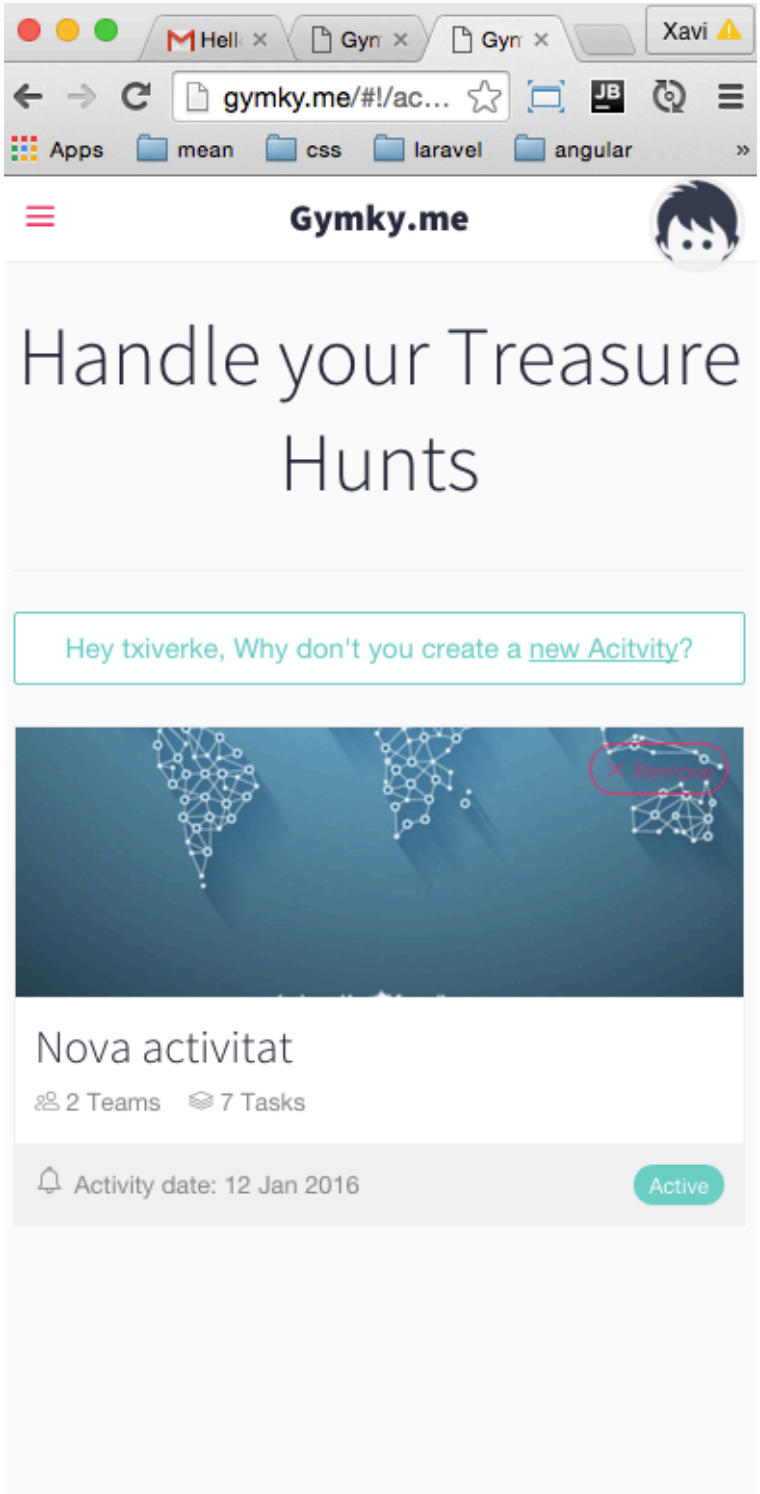
Menu Principal amb usuari desloguejat



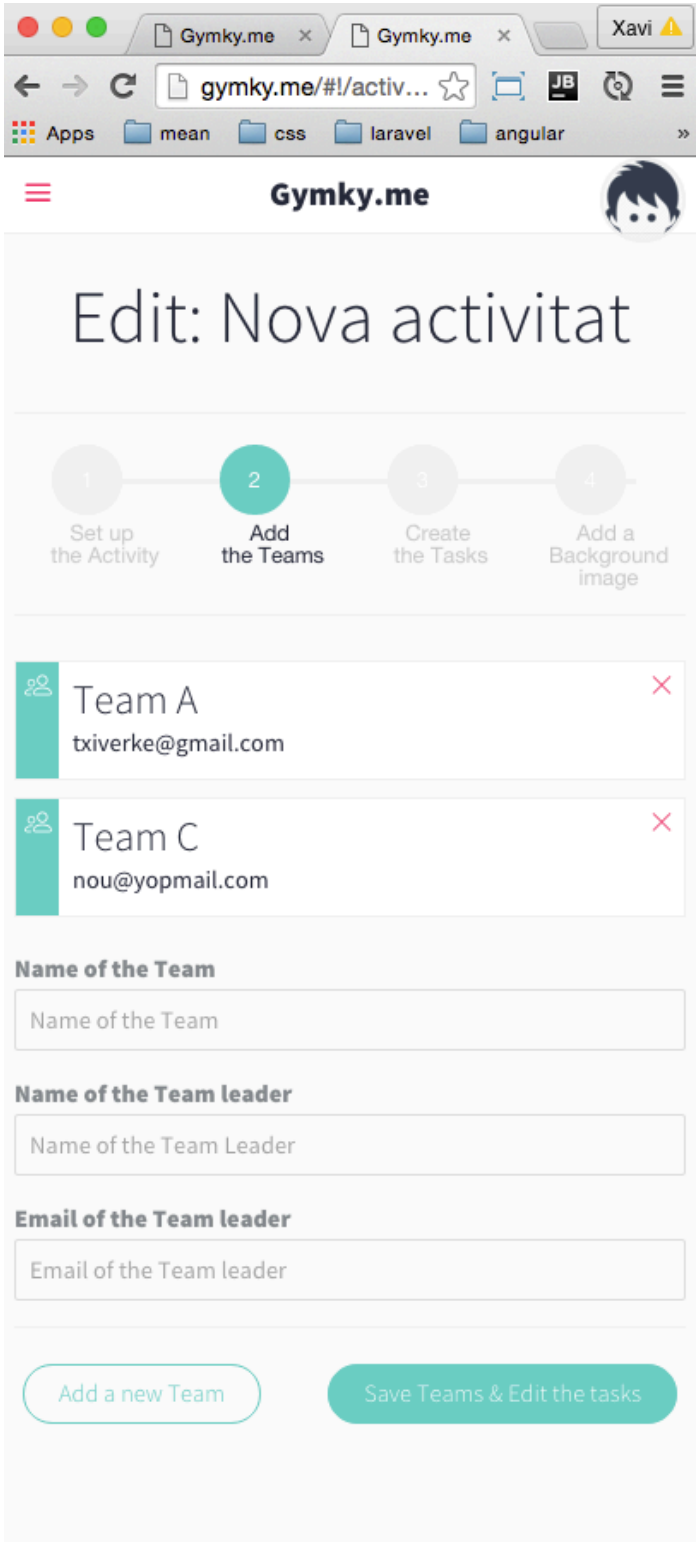
Pàgina de login



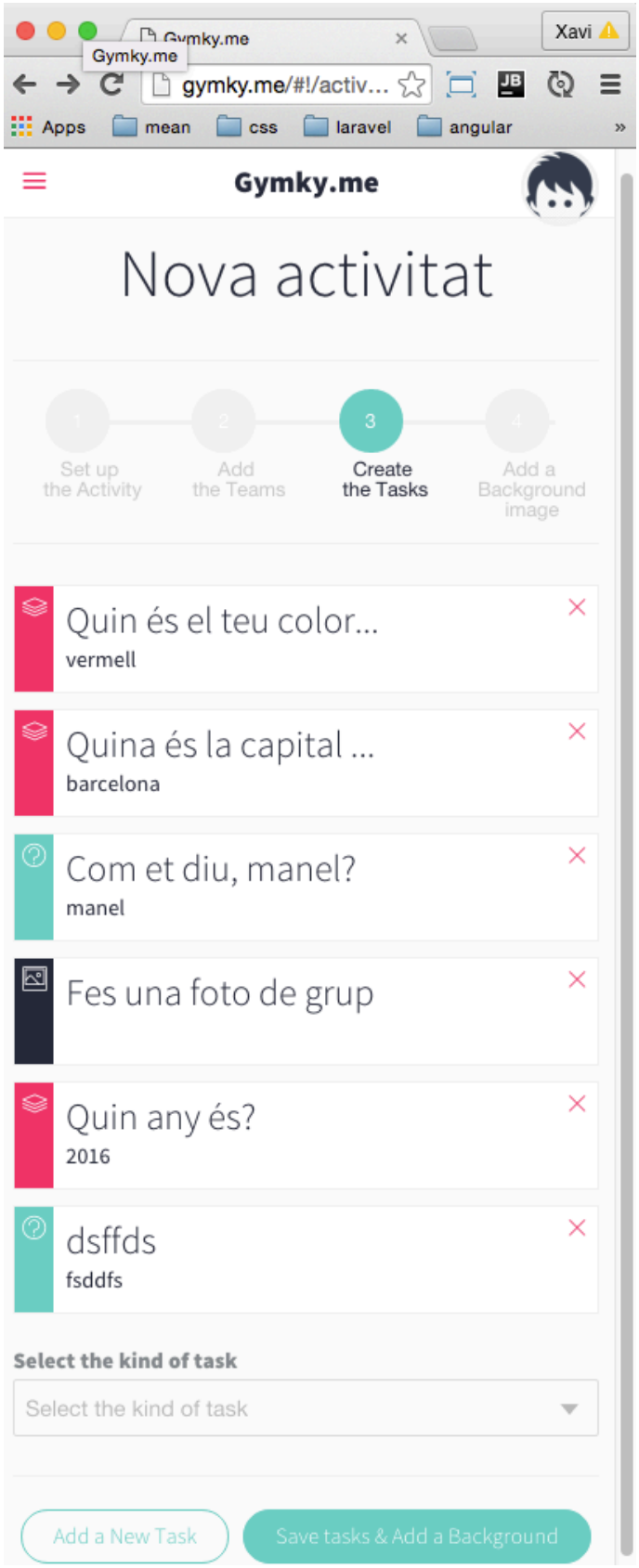
Llistat d'activitats



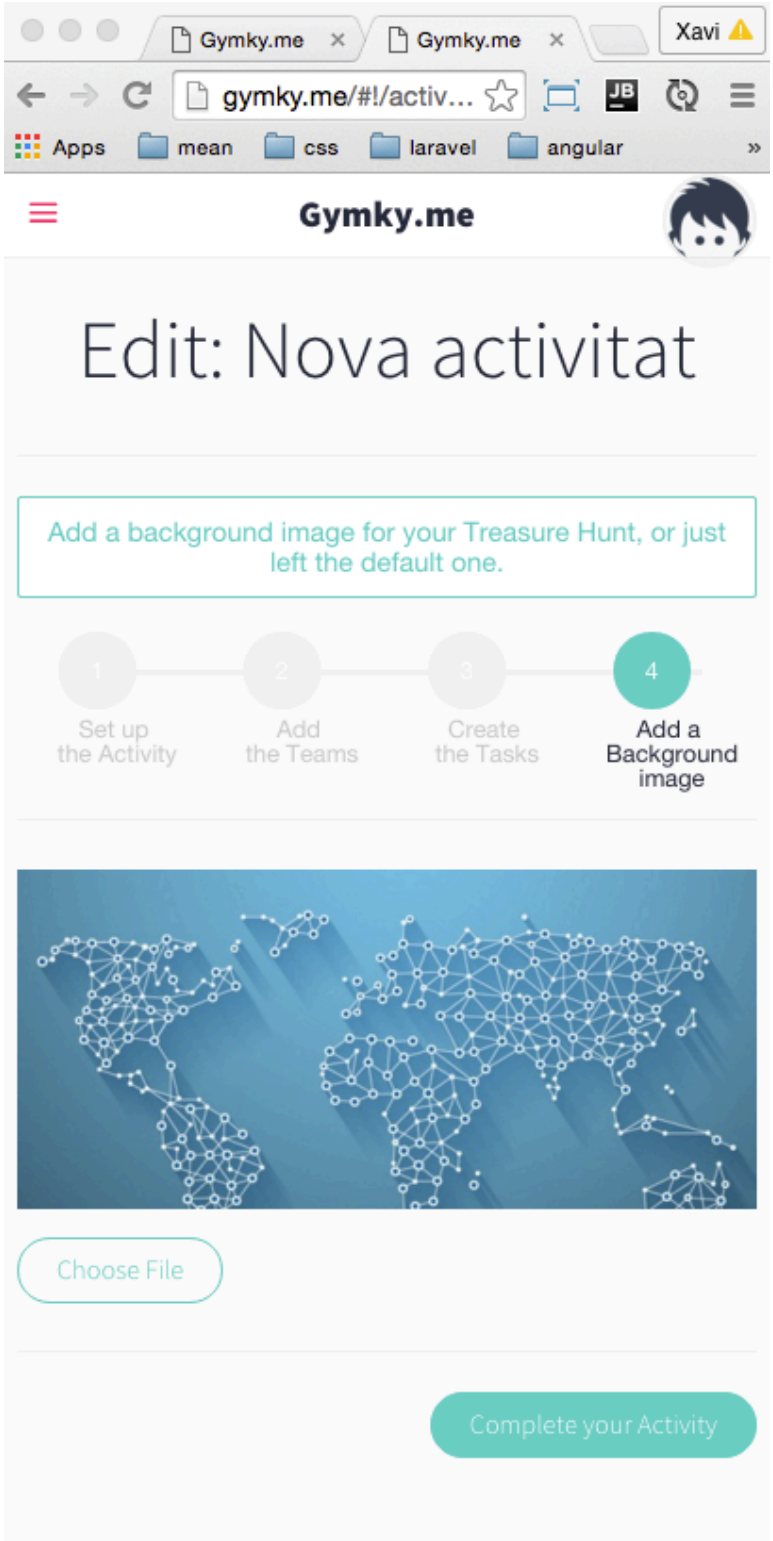
Pàgina de creació dels equips



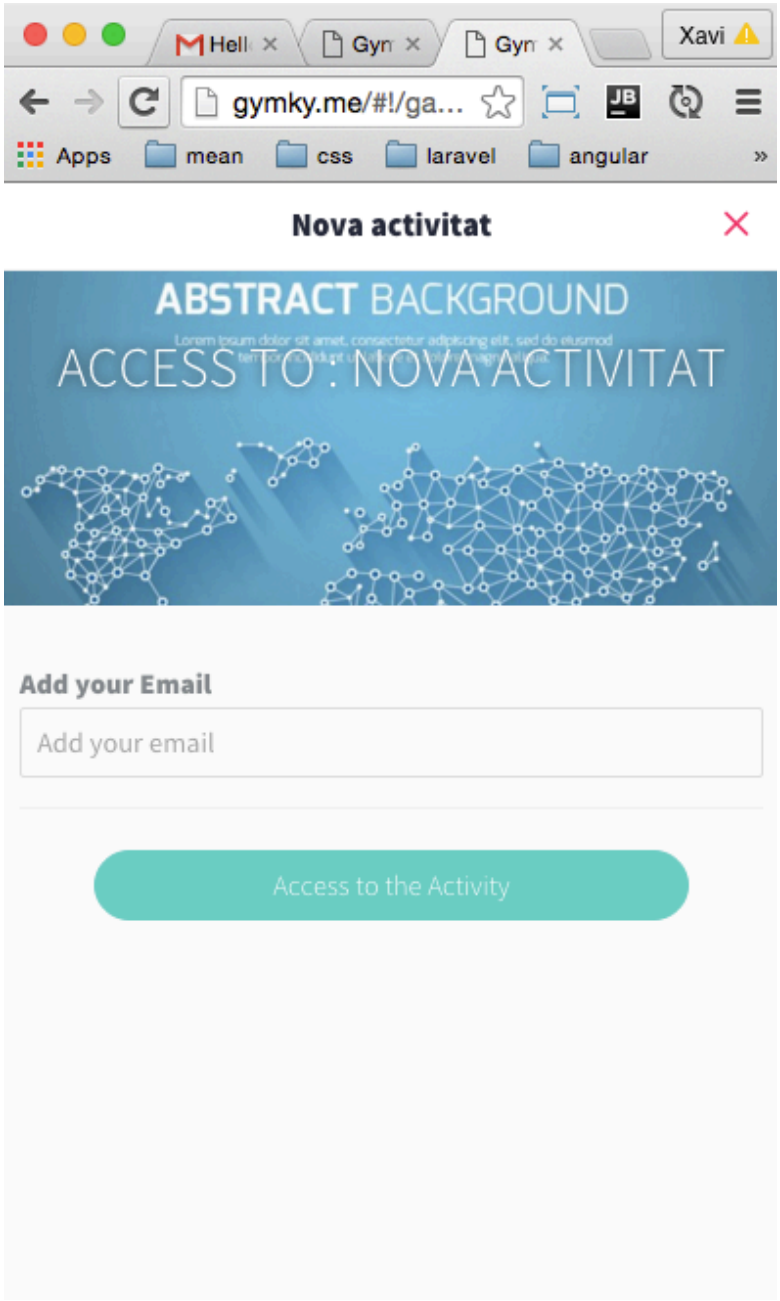
Pàgina de creació de tasques



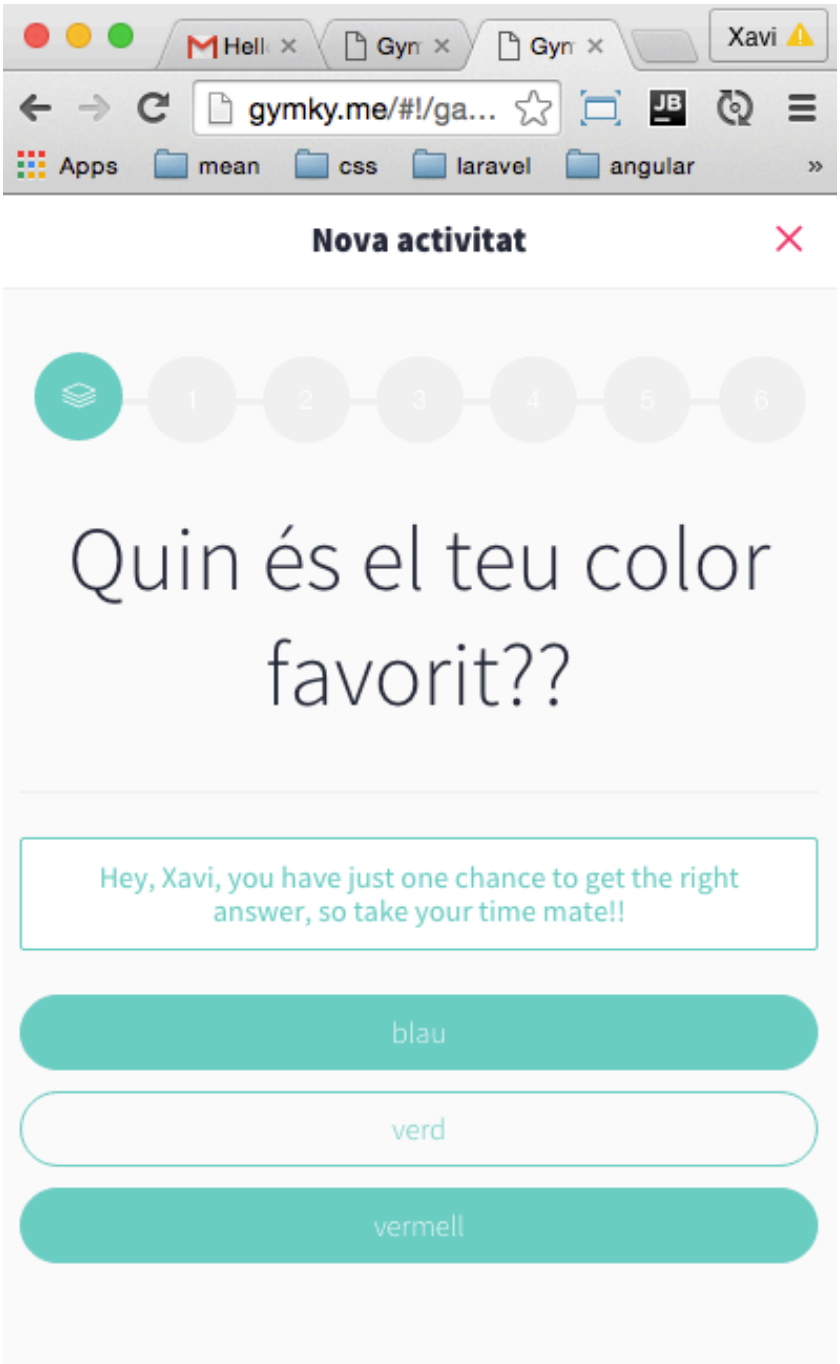
Pàgina per aferir la imatge de fons de l'activitat



Pàgina de login del joc



Pàgina de tasca del joc



12. Perfils d'usuari

Pel que fa als usuaris de l'aplicació bàsicament han estat estructurats en dos tipus:

Administrador

L'Administrador, el qual té accés a tots els continguts de l'aplicació alhora que n'és l'encarregat de la seva creació i edició. També és l'encarregat de gestionar als altres tipus d'usuari de l'aplicació com són els jugadors, d'afegir-los a l'aplicació i d'incloure'ls en els equips corresponents.

Aquest usuari requereix d'un registre complet en l'aplicació, i per tant haurà de completar el formulari de registre.

Jugador

Els jugadors, són els usuaris que només tenen accés als continguts creats per l'administrador, però sense poder editar-los, i només podran anar seguint les indicacions que obtindran de les activitats.

No requereixen de registre i poden accedir-hi directament introduint el correu que els farà arribar l'aplicació via notificació un cop l'administrador els hagi afegit algun dels equips.

13. Usabilitat/UX

La usabilitat és un punt clau per a una aplicació en que la interacció amb l'usuari es la clau de l'èxit i més si es té en compte que gran part d'aquesta interacció es durà a terme mitjançant dispositius mòbils i on per tant les dimensions del dispositiu obliguen a que els continguts es mostrin de forma clara i intuïtiva.

D'altra banda des de la perspectiva de l'administrador, també es clau que la experiència d'usuari sigui òptima, ja que en cas contrari, el fet de tenir que crear continguts complexos podria acabar sent un handicap difícilment assumible, per a usuaris amb coneixements tècnics estàndard.

Si s'entra més en detall en l'anàlisi de la usabilitat de l'aplicació, el principi que ha de regir-ne la implementació ha de ser la senzillesa, partint d'un disseny que seguirà les tendències de disseny *flat*, sense gaire concessions a l'excés d'elements gràfics que només busquin un component estètic. Ans al contrari tot element gràfic, a part de la landing page, anirà encaminat a cercar que el feedback amb l'usuari sigui el seu principal objectiu. Per aquesta raó es tendirà a unes interfícies gens recarregades, on la iconografia serà un element clau, amb un fons blanc i on els elements que hagin de ser destacats empraran colors força contrastats que clarament siguin identificables.

Pel que fa a la landing page, s'organitzarà l'estructura de la pàgina amb una capçalera on tindrem el logotip a l'esquerra els botons de registre i de login a la dreta, i amb un fons amb alguna imatge de gent rient o jugant, amb alguna frase promocional a sobre. S'ha de tenir en compte que el disseny de l'aplicació parteix de la tècnica *Mobile First*, on s'inicia el projecte des del disseny per a dispositius mòbils i on posteriorment es va generant les diferents interfícies per a altres dispositius basant-se en disseny responsive. Per tant, en tot moment es gestiona l'estructura de l'aplicació optimitzant la visibilitat dels continguts independentment del dispositiu des d'on es realitzi aquesta visualització. Seguidament sota la capçalera de la landing page, hi situarem activitats destacades, i posteriorment instruccions d'ús de l'aplicació.

Pel que fa al planell d'administració, l'usuari primerament accedirà a un formulari on introduirà les dades de l'activitat a realitzar i posteriorment accedira al planell de creació d'activitats, on per facilitar la interacció ja tindrà generades els tres tipus de activitat que s'implementaran d'un inici, preguntes, localització i pujar fotografies. L'usuari només haurà de posar el nom de la tasca i completar les dades requerides per la tasca. En tot moment es primarà els tooltips amb informació addicional per tal de guiar l'administrador en la creació de l'activitat.

Pel que fa als jugador, la interfície serà molt més simple i intuïtiva, bàsicament haurà d'introduir les seves dades que haurà rebut via notificació, i directament accedirà l'activitat que tingui activa. La interfície de la primera tasca ja li apareixerà, amb instruccions detallades de com completar-la, i en tot moment amb accés a un menú d'ajuda on se li oferirà informació de cada tipus de tasca.

14. Seguretat

Tot i de tractar-se d'una aplicació on no és gestionen dades econòmiques dels usuaris si que hi ha informació que pot ser confidencial, per tant s'ha decidit emprar una solució per a l'accés a l'aplicació força desenvolupat i provat en milers d'aplicacions com és el mòdul Passport.

Aquest mòdul, gestiona l'accés a l'aplicació, creant un token que es guarda en sessió, i que és eliminat quan l'usuari es desconnecta de l'aplicació o bé quan hi ha inactivitat durant un temps determinat. Les claus d'usuari també segueix un processat de xifratge i encriptació evitant així que en cas d'accés a la BBDD és puguin emprar les dades per reemplaçar la identitat de l'usuari.

Passport també ens permet l'accés a l'aplicació mitjançant facebook o twitter, emmagatzemant les dades que proporcionen les API d'aquestes aplicacions.

Un altre factor on la seguretat podria incidir, és en les dades emmagatzemades en la BBDD. Per aquest propòsit es desenvoluparà un mòdul el qual cíclicament va generant còpies de seguretat de l'aplicació alhora que les va desant, de forma que, en cas de caiguda del servidor o d'algun tipus d'atac, podem accedir-hi i tornar-les a bolcar.

També es tindrà en compte la seguretat en els formularis evitant que s'hi introdueixi cap tipus de codi i només acceptant text pla, per tal d'evitar els atacs d'injecció o XSS.

15. Tests

Un punt clau per al desenvolupament d'una aplicació moderna i que molts cops és obviat, és la creació de tests, que permetin comprovar que l'aplicació funciona correctament, sense la necessitat de que aquesta sigui emprada per usuaris reals.

Alhora de plantejar aquests tests, i tenint en compte que s'han d'implementar tant en l'API Rest com en el frontend de l'aplicació s'ha optat per desenvolupar-los emprant Mocha, que és un framework de tests, que aporta tot un seguit de mètodes per comprovar l'òptim funcionament de l'aplicació. Dins dels tests s'ha optat per desenvolupar el que es coneix com a BDD (Behavior-driven development).

Bàsicament, el que realitzem són simulacions de com hauria de funcionar l'aplicació, comprovant que les dades que s'obtenen són les adequades, tant en nombre com el tipus.

Primerament es descriu el funcionament, es criden les diferents funcions de l'aplicació i se n'esperen els resultats. Abans de cada test, s'han de crear o simular els requisits de cada funció que es vol testejar, per exemple si es comprova el login de l'aplicació s'ha de crear un usuari, posteriorment logejar-lo i posteriorment eliminar-lo.

16. Versions de l'aplicació/servei

Pel que fa al versionat de l'aplicació, encara es troba en procés de desenvolupament. Està previst que per a l'entrega de la PAC3 és disposarà d'una versió Beta on s'haurà implementat el registre d'usuari, la creació d'activitats i s'hauran desenvolupat les interfícies gràfiques.

Finalment la versió Final estarà enllestida per a l'entrega final, on s'haurà testejat l'aplicació en un servidor de producció i per tant s'encaminarà el projecte cap a una versió 1.0, preparada per ser emprada per usuaris reals.

17. Requisits d'instal·lació

Els requisits necessaris per instal·lar l'aplicació en una maquina local, bàsicament són tenir instal·lat Node.js a l'ordinador i disposar d'accés a Internet. Amb aquests dos requisits bàsics a disposició en el següent punt es detallen les instruccions d'instal·lació.

Per accedir a l'aplicació només cap accedir al domini www.gymky.me, des de qualsevol dispositiu emprant el navegador web.

18. Instruccions d'instal·lació

Per a la realització de l'instal.lació del projecte en una màquina local, només s'ha de descarregar el projecte en el directori que es desitgi, i posteriorment des del terminal de l'ordinador, instal.lar totes les dependències que ja apareixen en l'arxiu Package.json.

Per a realitzar aquesta tasca només s'ha de correr el comandament *"npm update"* al terminal i aquest descarregarà totes les dependències de l'aplicació en un directori anomenat *node_modules* on es desaran tots els moduls i llibreries necessaris.

Un cop totes les dependències estiguin instal·lades, s'haurà de correr a la consola el comandament *"gulp"* el qual, unificarà tot el javascript de l'aplicació en un sol arxiu anomenat *app.js* a més a més de generar tots els fulls d'estils necessaris.

Finalment, quan s'hagin realitzat tots els passos anterior, només caldrà iniciar el servidor Node.js, emprant el comandament *"node index.js"* a la consola i ja es podrà accedir al localhost del navegador que es desitgi apuntant al port :3000.

19. Bugs

Pel que fa referència a la gestió de bugs, es seguirà el procés descrit en l'apartat de metodologia, tenint-se que crear una tasca concreta per a solucionar cada un dels bugs, tenint que passar per les columnes de tasca a realitzar, tasca realitzant-se, tasca testejant-se i finalment tasca completada i per tant bug arreglat.

El cas dels bugs és un dels punts on la metodologia *Agile* emprada és més útil i facilita en gran manera que el projecte es mantingui estable, tot i la inevitable aparició d'aquests.

També cal remarcar que les aplicacions que compten amb un desenvolupament de tests profund, acostumen a tenir molts menys bugs en producció, ja que aquest són detectats durant el desenvolupament i per tant arreglats en aquell moment.

Pel que fa a l'estat actual de l'aplicació i a l'aparició de bugs, s'ha de tenir en compte que el testeig que s'ha realitzat a l'aplicació és únicament el realitzat per propi desenvolupador i per tant l'aparició de bugs, no es pot descartar en cap cas. El procediment per tal de solventar-los serà el descrit anteriorment amb la metodologia "*Agile*" i per tant durant tot el cicle de vida de l'aplicació s'haurà d'estar controlant la possible aparició d'errors.

Des d'una perspectiva de un producte acabat, s'hauria d'introduir dins el cicle de testeig la figura d'un QA, que seria l'encarregat de realitzar un testeig cíclic de les diferents part de l'aplicació, principlament de les més crítiques com serien la creació d'usuaris i activitats, l'accés a aquestes, la interacció amb l'usuari durant el joc…

20. Pressupost

Pel que fa al Pressupost, donat que es tracta d'un projecte personal i de la seva implementació durant el TFG, aquest és quasi inexistent. Bàsicament només s'ha realitzat la compra del domini gymky.me, així com el hosting a Digital Ocean.

Òbviament si l'aplicació, com s'espera, agafés volada, s'hauria de replantejar tota l'estratègia econòmica, però es considera que actualment aquest aspecte queda fora de l'àmbit del TFG, ja que no es persegueix un rendiment econòmic immediat sinó implementar els coneixements adquirits durant el Grau sumats a l'experiència laboral en un projecte tècnicament força ambiciós i il·lusionant.

21. Anàlisi de mercat

Donat que la idea, com s'ha explicat en alguns punts del treball, sorgeix de certes inèrcies viscudes en diferents entorns laborals, es considera que l'aplicació pot anar dirigida perfectament a certs entorns laborals, on les dinàmiques de grup requereixen d'aquests tipus d'activitats i el món de desenvolupament de software en podria ser un clar exemple.

Realitzant un breu anàlisi de la competència s'observa que existeixen força empreses, sobretot en el mercat anglosaxó que organitzen aquest tipus d'activitats de *Team building*, però en cap cas en forma d'aplicació que doni la llibertat als usuaris de crear les seves pròpies activitats. Més que res es tracta d'empreses que organitzen en forma presencial, activitats en grup, les quals són gestionades personalment pel propi personal de l'empresa.

Es per això que es considera que existeix un mercat molt gran per explorar i que bàsicament l'èxit de l'aplicació a part de la qualitat del mateix, depèn en força mesura de la capacitat d'entrar en aquests mercats.

Donat que aquest punt queda fora de l'àmbit del TFG, es considera que en primer terme els esforços han d'anar encaminats a crear un producte de qualitat, i posteriorment quan ja sigui un producte real analitzar detingudament les possibilitats i per tant la inversió necessària per tal de donar a conèixer el producte o fins i tot contactar directament amb les empreses que ofereixen aquesta mena d'activitats en forma presencial i oferir-los l'aplicació com una opció clara de reduir els seus costos.

22. Conclusió/-ns

Un cop finalitzat el projecte i amb la perspectiva de tot el procés de desenvolupament, caldria destacar la importància de la planificació. Amb unes dates d'entrega tancades i amb un sol desenvolupador per a la realització de la feina, la planificació és la clau de l'èxit o del fracàs del projecte, i tenint en compte l'experiència d'aquests mesos, cal remarcar que aquesta no sempre ha estat del tot realista. En molts casos s'ha hagut d'allargar el termini d'entrega donada la carrega de treball en molts casos inassumible. Per això s'hauria de fer constar que la planificació ha estat millorable, però alhora, constatar aquest fet serveix per entendre la importància d'aquest punt.

Pel que fa al procés de desenvolupament pròpiament dit, cal d'estacar tot l'aprenentatge obtingut. El fet d'haver d'assumir certes tasques que normalment no ens són les habituals, forcen a haver d'explorar, estudiar, cercar, provar, errar i tornar a provar molts processos del desenvolupament. Però és precisament aquesta dinàmica la que resulta realment enriquidora i que converteix la creació d'un projecte com aquest en tot un repte, el qual un cop finalitzat aporta moltes satisfaccions.

Tot i que s'entrega una versió final de Gymky.me, aquest continua sent un projecte en desenvolupament al qual s'aniran afegint més característiques per tal de fer-lo créixer i intentar convertir-lo en un projecte amb usuaris reals, que en facin un ús real. És precisament aquest fet el que converteix tot el procés de creació del TFG en una experiència del tot positiva i enriquidora.

Annex 1. Lliurables del projecte

Els següents arxius formen el backend de l'aplicació i es troben dins el directori app:

Controllers

Es defineixen les peticions a la base de dades. Les funcions declarades i que composen el CRUD de l'aplicació, son referenciades des de els arxius de rutes i també es requereixen els diferents models per poder realitzar les operacions corresponents.

- activities.server.controller.js
- index.server.controller.js
- users.server.controller.js

Models

Es on es defineixen els models de les Col·leccions de la BBDD i on a més a més es realitza la validació de les dades.

- activity.server.model.js
- user.server.model.js

Routes

Es declaren les rutes que es criden de d'angular per obtenir les dades de la BBDD, i es criden les funcions que hem declarat als Controllers.

- activities.server.routes.js
- index.server.routes.js
- users.server.routes.js

Views

Són les vistes que es generen des del servidor i no directament des d'Angular, per dir-ho d'alguna forma són l'esquelet d'HTML de l'aplicació. Estan creades emprant les plantilles ejs, però no són més que HTML al qual se li poden afegir variables i demés elements.

- index.ejs
- signin.js
- signup.js

Seguidament es detallen els arxius que formen el Front end de l'aplicació i que es troben dins del directori public:

A la carpeta app s'hi troba tota la lògica de l'aplicació, organitzats en els diferents mòduls que en formen part i seguint el mateix patró MVC emprat al Backend.

App

Es descriuen els principals arxius del modul Activities, donat que es el més complexe i alhora representatiu del Front end.

activities
- controllers :
  - activities.client.controller.js
    Aquest arxiu conté tota la lògica del mòdul i gestiona tant les crides a l'API, realitzades pels serveis com, com la conseqüent gestió de les dades obtingudes que es mostren a les vistes.
- directives
  - getStatus.client.directive.js
- Partials
  - getStatus.client.partial.html
- Services
  - activities.client.service.js
    Aquest arxiu realitza les crides a l'API i realitza el CRUD amb la BBDD.
- Views
  - activities.client.view.html
  - create-activity.client.view.html
  - edit-activity.client.view.html
  - list-activities.client.view.html
  - tasks-activity.client.view.html
  - teams-activity.client.view.html
  - view-activity.client.view.html

    Són les diferents vistes que mostren i recullen les dades que després són gestionades pel controlador.

config
Aquest directori emmagatzema les diferents rutes per accedir als continguts i que carreguen els mòduls adients.

login
Directori on es gestiona el login dels usuaris

menu
Mòdul on es gestiona el menu principal de l'aplicació

profile
Mòdul on es troba la gestió del profile d'usuari

users
Mòdul on es realitza la creació d'usuaris

app.js
Arxiu principal de l'aplicació i que inicializa com a dependències tota la resta.

La resta de directoris que trobem gestiones el css, les imatges, i les fonts emprades.

Annex 2. Codi font(extractes)

Index.js

Arxiu principal de l'aplicació on s'inicialitza el servidor, les dependències i es connecta amb la BBDD

```
/**
 * Initialize the environment
 * @type {string|*}
 */
process.env.NODE_ENV = process.env.NODE_ENV || 'development';

/**
 * Require dependencies
 */
var mongoose = require('./config/mongoose'),
   express = require('./config/express'),
   passport = require('./config/passport');

/**
 * Initialize the app
 */
var db = mongoose();
var app = express();
var passport = passport();
var port = 3000;

module.exports = app;

app.listen(port, function() {
   console.log('Listening http://localhost:%s/', port)
});
```

Mongoose.js

Arxiu on es carreguen els models que ens permeten accedir a la BBDD.

```
/**
 * Gymky.me
 * Created by xavi on 11/10/15.
 */

var config = require('./config'),
   mongoose = require('mongoose');


module.exports = function() {
```

```
  var db = mongoose.connect( config.db );

  require('../app/models/user.server.model');
  require('../app/models/activity.server.model');

  return db;
};
```

Index.ejs

Plantilla principal de l'aplicció on es carrega tot el css i javascript i on també es declara l'etiqueta main amb l'atribut ng-view d'Angular que ens permetrà renderitzar les diferents vistes de l'aplicació.

```
<!DOCTYPE html>
<html xmlns:ng="http://angular.js">
<head>
  <title><%= title %></title>
  <link rel="stylesheet" href="/css/fonts.css">
  <link rel="stylesheet" href="/css/external/normalize/normalize.min.css">
  <link rel="stylesheet" href="/css/app.css">
</head>
<body>

  <main ng-view></main>

  <script type="text/javascript">
    window.user = <%- user || 'null' %>;
  </script>

  <script type="text/javascript" src="/app.js"></script>
</body>
</html>
```

Gulpfile.js

Arxiu on és declaren les tasques per generar el css i el js final, després de minificar-lo i unificar-lo.

```
/**
 * Created by xavi on 10/10/15.
 */
/**
 * Gulp Dependencies
 * @type {Gulp|exports}
```

```
 */
var gulp    = require('gulp'),
   clean   = require('gulp-clean'),
   buffer  = require('vinyl-buffer'),
   source  = require('vinyl-source-stream'),
   concat  = require('gulp-concat'),
   uglify  = require('gulp-uglify'),
   sass    = require('gulp-sass'),
   minify  = require('gulp-minify-css'),
   assign  = require('lodash.assign');

gulp.task('default', ['generate-js', 'generate-css']);

gulp.task('delete-app',function(){
   return gulp.src(['./public/*.js'])
      .pipe(clean());
});

gulp.task('generate-css', function () {
   return gulp.src('./public/scss/app.scss')
      .pipe(sass().on('error', sass.logError))
      .pipe(concat('app.css'))
      .pipe(minify())
      .pipe(gulp.dest('./public/css'));
});

gulp.task('generate-js', ['delete-app'], function () {

   console.log('Compress-JS init');

   return gulp.src([
      './node_modules/angular/angular.js',
      './node_modules/angular-route/angular-route.js',
      './node_modules/angular-resource/angular-resource.js',
      './public/app/login/base.client.module.js',
      './public/app/login/**/*.js',
      './public/app/users/users.client.module.js',
      './public/app/users/**/*.js',
      './public/app/activities/activities.client.module.js',
      './public/app/activities/**/*.js',
      './public/app/config/*.js',
      './public/app/app.js'
   ])
   .pipe(concat('app.js'))
   //.pipe(uglify())
   .pipe(gulp.dest('./public/'));
});
```

Annex 3. Llibreries/Codi extern utilitzat

Node i Express
Ens permeten la creació del servidor que emprem per correr l'aplicació així com la gestió de sessions i tot un seguit de mètodes.

Mongoose i Mongodb
Ens permeten l'emmagatzematge de dades així com tota un sèrie d'accions sobre aquestes dades, com validacions.

Gulp
Es un *Task runner* que ens permet automatitzar tota una sèrie de processos que es van repetint duran el desenvolupament de l'aplicació, com per exemple, minificar i unificar els arxius javascript que anem creant, convertir els arxius SASS en CSS per poder-los al navegador...

Angular
Ens permet organitzar el codi javascript seguint el patró MVC, a més de facilitar-nos en gran manera les peticions a l'API Rest, el bindatge amb els elements del DOM del document web...

Passport
Ens facilita la implementació dels registres d'usuaris a més a més de permetre'ns emprar-ne d'altres com el de facebook o twitter.

SASS
És un pre-processador de CSS que ens permet crear els full d'estil de forma més ràpida, alhora que podem emprar certs elements no disponibles en els CSS normals, com certs mètodes, variables, funcions...

Mocha

És un Framework de tests que ens permet testejar el codi javascript tant al frontend com a l'API Rest.

Annex 4. Captures de pantalla

Les captures de pantalla han esta emprades en la secció de prototipatge

Annex 5. Bibliografia

Amos Q. Haviv (September 2014). MEAN Web Development . Packt Publishing Limited

Matt Frisbie (December 2014). Angular JS Web Application Development Cookbook. Packt Publishing Limited