UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITÈCNICA SUPERIOR DE GANDIA

Grau en Eng. Sist. Telecom., So i Imatge

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

ESCOLA POLITÈCNICA
SUPERIOR DE GANDIA

"Integració d'un sistema d'alertes en temps real en emissions de Twitch"

TREBALL FINAL DE GRAU

Autor/a:
Carles Fèlix Tur

Tutor/s:
Jordi Bataller Mascarell

GANDIA, 2017

Resum

En la televisió convencional s'ha posat de moda connectar amb les xarxes socials i mostrar en pantalla les interaccions dels usuaris. Un exemple prou conegut és quan un programa de televisió habilita una etiqueta (*hashtag)* en la xarxa social Twitter perquè els televidents, puguen comentar-lo publicant missatges (*tweets)*, per posteriorment, la pròpia cadena, mostrar-ne alguns a l'atzar en forma de rètols.

La televisió convencional va perdent cada dia protagonisme per culpa de la televisió per Internet, ja que aquesta pot arribar a tot el món i amb un baix pressupost. Un exemple és Twitch, el qual és una plataforma de *streaming* (retransmissió per Internet) en directe i a la carta de videojocs per Internet.

En aquest projecte, s'intentarà dissenyar per als usuaris que emeten en Twitch, un sistema d'alertes que es puga incrustar en el vídeo en directe, i que mostre en temps real les interaccions dels usuaris que ocorren en les xarxes socials, sense utilitzar aparells físics com fan les televisions convencionals.

Resumen

En la televisión convencional se ha puesto de moda conectar con las redes sociales y mostrar por pantalla las interacciones de los usuarios. Un ejemplo muy conocido es cuando un programa de televisión habilita una etiqueta (*hashtag)* en Twitter para que los televidentes puedan comentarlo publicando mensajes (*tweets)*, para posteriormente, la propia cadena, mostrar algunos al azar en forma de rótulos.

La televisión convencional cada día va perdiendo protagonismo por culpa de la televisión por Internet ya que puede llegar a todo el mundo y con un bajo presupuesto. Un ejemplo es Twitch, el cual es una plataforma de *streaming* (retransmisión por Internet) en directo y a la carta de videojuegos por Internet.

En este proyecto, se intentará diseñar para los usuarios que emiten en Twitch, un sistema de alertas que se pueda incrustar en el vídeo en directo, y que muestre en tiempo real, las interacciones de los usuarios que ocurren en las redes sociales, sin utilizar aparatos físicos como hacen las televisiones convencionales.

Abstract

In conventional television, it has become fashionable to connect to social networks and to display on the screen the users' interactions.

A well-known example is when a television program enables a label (hashtag) on twitter so that viewers comment on it by posting messages (tweets), to then show some of them randomly.

Day after day, conventional television is losing its popularity since the so-called internet television can be spread all over the world with a low budget. A clear example is Twitch.tv which is a live streaming platform (Internet broadcasting) and online videogames on demand.

In this project, we will attempt to design a warning system embedded in the live video, for users who broadcast in Twitch.tv

This system will show in real time the users' interactions in social networks without using physical devices like conventional televisions do.

Índex

Capítol 1. Introducció ... 3
1.1 Motivació ... 3
1.2 Objectius del projecte ... 3
1.3 Estructura del document ... 4
Capítol 2. Anàlisi dels requeriments ... 5
Capítol 3. Ferramentes ... 10
3.1 JavaScript ... 10
3.2 Node.js [5] ... 10
3.3 HTML ... 10
3.4 CSS (Cascading Stylesheets) ... 11
3.5 Postman [7] ... 12
3.6 Twitter ... 13
3.7 Strawpoll ... 13
3.8 OBS (Open Broadcaster Software) [8] ... 14
3.9 Sublime Text [9] ... 15
Capítol 4. Disseny de l'aplicació web ... 16
4.1 Llibreria twitch_controller.js ... 16
4.2 Llibreria Utilities.js ... 18
4.3 Llibreria Overlay.js ... 19
4.4 Arxiu postGenerator.js ... 22
4.5 Arxiu auth.js ... 24
4.6 Arxiu dashboard.js ... 25
4.7 Arxiu follower_alert.js ... 27
4.8 Llibreria Twitter_controller.js ... 28
4.9 Arxiu Twitter_alert_conf.js ... 29
4.10 Arxiu Twitter_alert.js ... 30
4.11 Arxiu Strawpoll_conf.js ... 31
4.12 Arxiu Strawpoll.js ... 31
4.13 Arxiu follower_list.js ... 33
Capítol 5. Implementació ... 34
5.1 Organització de fitxers ... 34
5.2 Problemes trobats ... 36
5.2.1 Renderització de rètols ... 36
5.2.2 Peticions a l'aplicació de Twitter ... 36

5.2.3 Peticions a l'aplicació de Strawpoll 36
5.2.4 Autentificació en programes emissors 36
Capítol 6. Guia d'ús 37
6.1 Guia d'instal·lació 37
6.2 Instruccions d'ús 37
6.2.1 Inici de sessió 37
6.2.2 Secció d'usuaris 38
6.2.3 Alertes 39
Capítol 7. Conclusions 42
7.1 Treball futur 42
Bibliografia 43

Capítol 1. Introducció

1.1 Motivació

Justin.tv va ser una xarxa social nascuda al 2007 on quelcom registrat podia visualitzar o emetre vídeo en directe per Internet.

Aquest lloc englobava canals de vídeojocs, política, cine i comèdia entre altes. Aquests canals estaven classificats per categories en funció del tipus del contingut. La categoria de vídeojocs va començar a fer-se molt popular, fins al punt que, al 2011, els propietaris de Justin.tv van crear una altra plataforma anomenada Twitch, centrada solament en aquesta categoria.

Actualment, Twitch [1] és una xarxa social que ofereix un servei de *streaming* en directe per a transmetre videojocs i un servei a la carta perquè es puguen veure les emissions ja finalitzades.

En aquesta plataforma, cada usuari registrat té el seu propi canal de televisió amb una sala de xat, no obstant això, sols l'utilitzen aquells usuaris que transmeten en directe, mentre la resta visualitza videojocs i xateja en canals d'altres usuaris que sí que estan emetent.

L'usuari que transmet, pot interactuar amb els que el visualitzen escrivint per la seua sala de xat o parlant pel micròfon en el directe, però de vegades, aquest no pot estar pendent de les seues xarxes socials (incloent Twitch) perquè també està centrat en jugar al vídeojoc. Una solució per a minimitzar aquest problema i connectar més amb els espectadors, seria que, el programa que usa l'usuari per a emetre, tinguera una funcionalitat que detectara interaccions importants en les seues xarxes socials, per posteriorment ser mostrades en el seu directe.

1.2 Objectius del projecte

Malauradament, cap programa que emet vídeo per Internet té la funcionalitat de mostrar missatges provinents de xarxes socials en la pantalla que s'està emetent en directe.

Aquest fet motiva aquest projecte, que tindrà com a objectiu principal implementar una mena de tituladora que mostre impressionat sobre el vídeo que s'està emetent, de manera automàtica, les interaccions més importants dels espectadors.

Els requeriments principals són implementar un sistema d'inici de sessió mitjançant un compte de Twitch, i també un menú per accedir a diferents alertes que mostraran rètols en funció de les interaccions que ocorren en algunes xarxes socials.

Aquest projecte també tindrà objectius formatius que seran imprescindibles per poder implementar l'aplicació web. Aquests objectius seran aprendre a programar en JavaScript, en Node.js, en CSS i en HTML

1.3 Estructura del document

Aquest document està estructurat per un primer capítol que consisteix en una breu introducció, un segon que planteja com serà i quines parts tindrà la futura aplicació web, un tercer que explica les ferramentes que s'usaran durant el desenvolupament del projecte, un quart que mostra i planteja l'estructura que tindrà cada arxiu programat en JavaScript [2], un cinqué que mostra l'estructura resultant del projecte i els problemes que hauran sorgit durant la seua implementació, un sisé que explica com usar i instal·lar la futura aplicació web i un seté que consisteix en una breu conclusió per resumir allò que s'ha fet i allò que podria implementar-se en un futur.

Capítol 2. Anàlisi dels requeriments

El sistema d'alertes es programarà en HTML [3] i JavaScript ja que, en l'actualitat, molts programes emissors poden mesclar en directe diverses fonts multimèdia, com per exemple, l'aspecte visual d'una web amb una webcam.

Aleshores, el projecte serà dissenyar i construir una web amb diferents seccions: pàgina principal, secció d'usuaris, alerta de seguidors, alerta de Twitter, llista de seguidors i alerta Strawpoll.

Pel que fa a la pàgina principal (figura 1), tindrà una zona de posts que explicaran i aportaran a l'usuari informació i tutorials per aprendre a usar la futura aplicació web. També tindrà un botó per a poder iniciar sessió amb els comptes de Twitch i així poder accedir posteriorment a la secció d'usuaris.

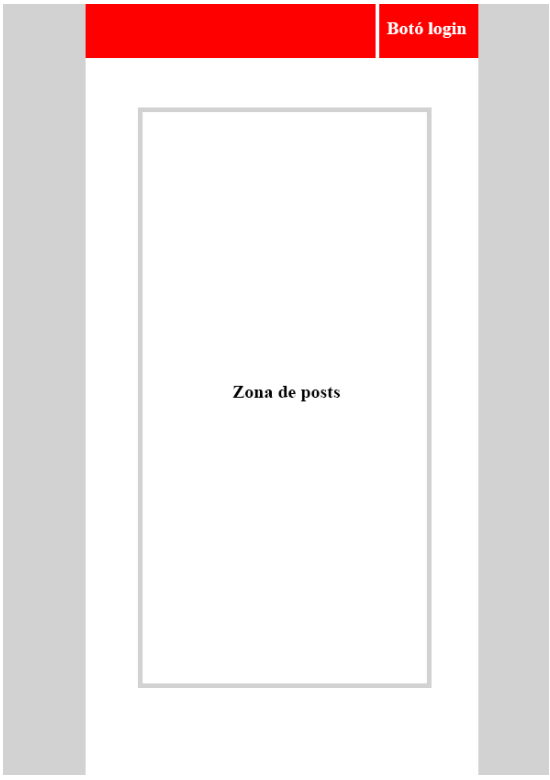

Figura 1. Esbós de la pàgina principal

Pel que fa a la secció d'usuaris, sols serà accessible a aquells que prèviament hagen iniciat sessió a través de la futura aplicació web amb el seu compte de Twitch.

Aquesta pàgina mostrarà la imatge de perfil i el nom d'usuari d'aquell que haja iniciat sessió en un determinat compte de Twitch, i també comptarà amb un menú per enllaçar a cada una de les pàgines, les quals mostraran les diferents alertes tal i com es pot observar en la figura 2.

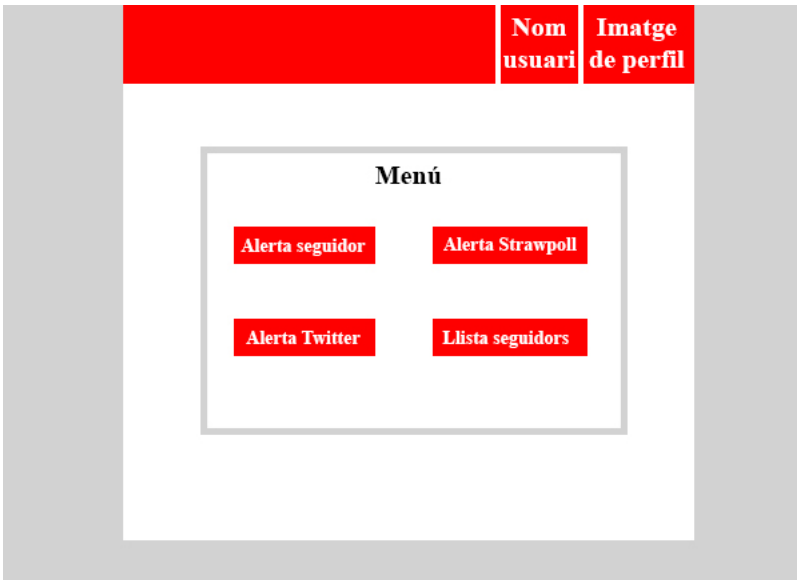

Figura 2. Esbós de la secció d'usuaris

Pel que fa a cada pàgina d'alertes, que sols són accessibles per als usuaris que han iniciat sessió en la pàgina principal, mostraran una determinada informació en temps real a partir d'altres webs o xarxes socials. Les pàgines d'alertes que van a implementar-se són les següents:

Alerta seguidor: l'usuari que visualitze aquesta pàgina, quan un espectador segueix el seu canal en Twitch, li apareixerà durant cinc segons un rètol amb el nom del nou seguidor donant-li les gràcies tal i com es pot veure en la figura 3.

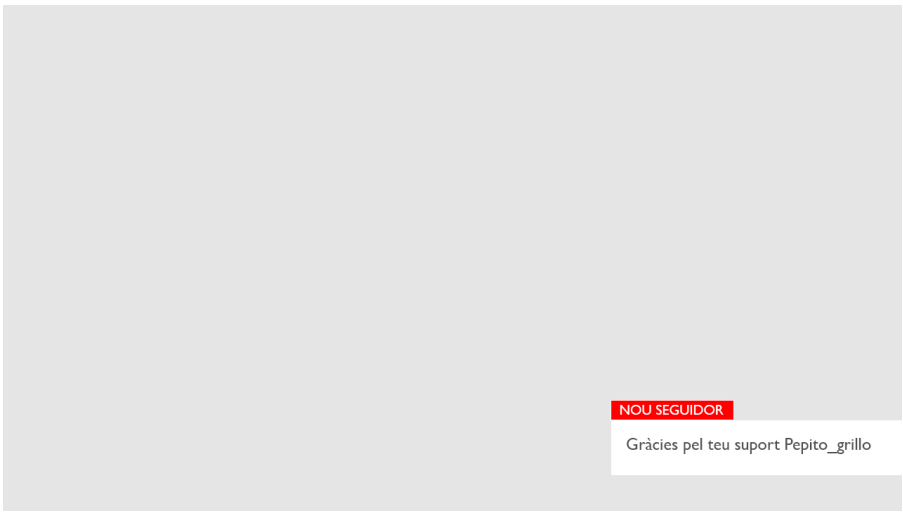

Figura 3. Esbós alerta seguidor

Llista seguidors: es mostrarà a l'usuari els últims set seguidors del seu canal de Twitch (figura 4).

Figura 4. Esbós de llista seguidors

Alerta Twitter: aquesta pàgina, cada quinze minuts, mostrarà en un rètol una ràfega dels quinze *tweets* més recents que continguen un *hashtag* que determina l'usuari. Aquest *hashtag* es configurarà abans en una pàgina que, posteriorment, llançarà aquesta alerta, és a dir, quan es clique el botó "Alerta Twitter" del menú de la pàgina "secció d'usuaris", en lloc d'enviar directament a "Alerta de Twitter" es redirigirà a una pàgina anomenada "Configuració de l'alerta Twitter", que conté un formulari amb un camp on s'ha d'introduir el *hashtag* desitjat i un botó d'enviament tal i com es pot veure en la figura 5. Aquest botó d'enviament redirigirà a la pàgina "Alerta de Twitter" amb el *hashtag* configurat (figura 6).

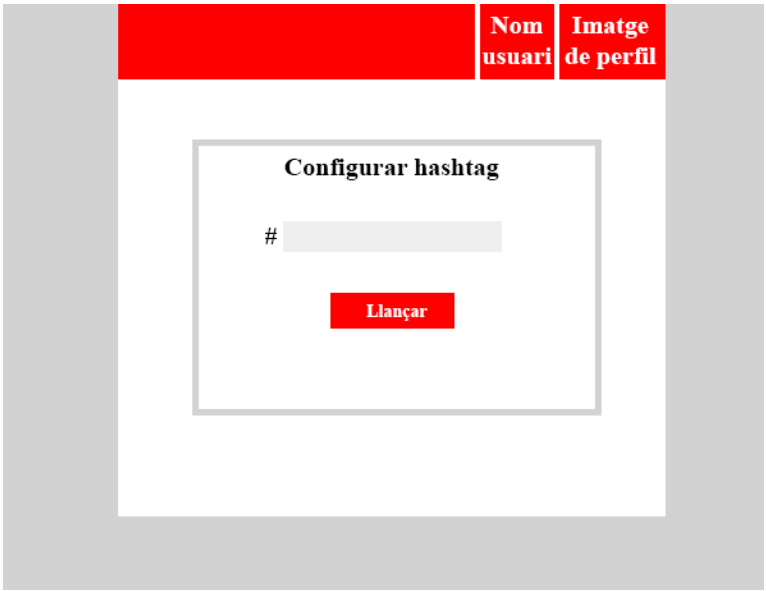

Figura 5. Esbós de configuració de l'alerta de Twitter

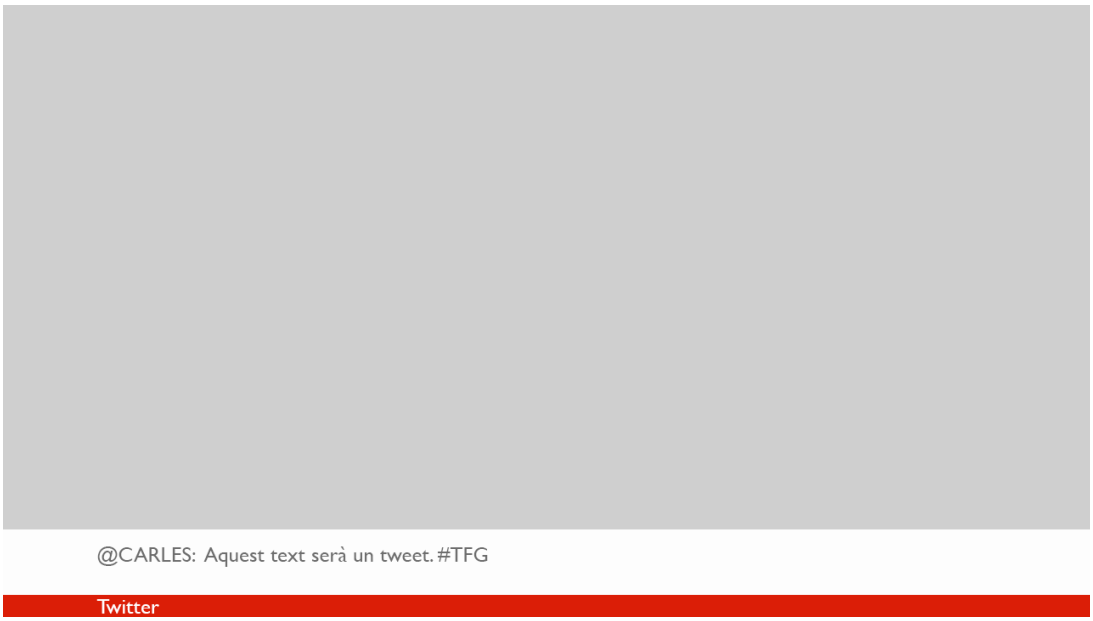

Figura 6. Esbós de l'alerta Twitter

Alerta Strawpoll: aquesta pàgina mostrarà en temps real i en forma de rètol, una determinada enquesta d'una web anomenada Strawpoll [4]. L'enquesta a mostrar, es determinarà abans en una pàgina que, posteriorment, llançarà aquesta alerta, és a dir, quan es clique el botó "Alerta Strawpoll" del menú de la pàgina "secció d'usuaris", en lloc d'enviar directament a "Alerta Strawpoll", es redirigirà a una pàgina anomenada "Configuració d'enquesta" (figura 7), que conté un formulari amb un camp on s'ha d'introduir l'enllaç de l'enquesta desitjada i un botó d'enviament. Aquest botó d'enviament redirigirà a la pàgina "Alerta de Strawpoll" amb l'enquesta configurada (figura 8).

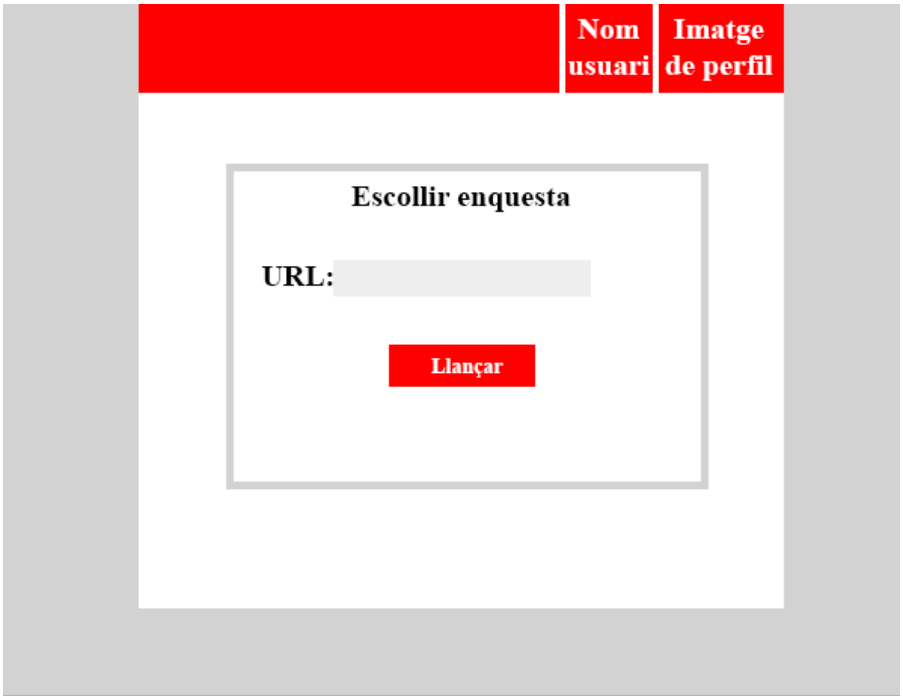

Figura 7. Esbós de configuració d'enquesta

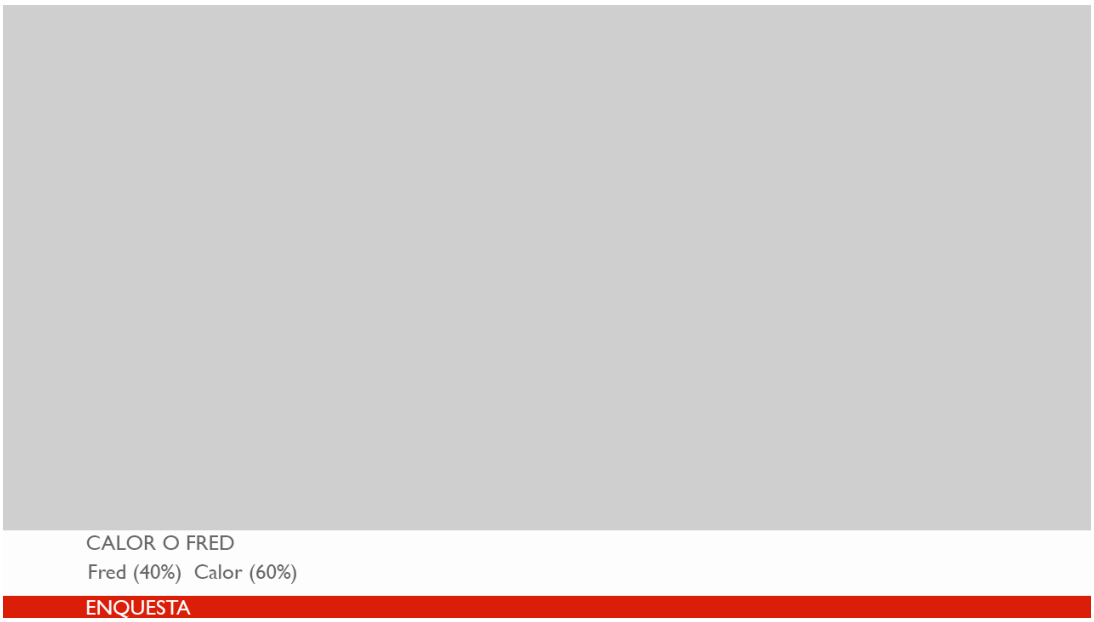

Figura 8. Esbós alerta Strawpoll

Capítol 3. Ferramentes

3.1 JavaScript

JavaScript és un llenguatge de programació que s'executa en la part del client amb l'objectiu de millorar l'aspecte visual de pàgines web en navegadors i atorgar a aquestes dinamisme.

Les sentències de JavaScript estan basades en les del llenguatge de programació Java i C. No obstant això, hi ha una diferència important: en JavaScript no es defineix el tipus de variable, és a dir, el tipus està associat al valor.

En aquest projecte es farà ús d'aquesta ferramenta per poder implementar funcionalitats imprescindibles a la futura web, com per exemple, canviar l'aspecte visual de la web en temps real o obtenir dades d'altres pàgines webs.

3.2 Node.js [5]

Els llenguatges de programació convencionals que s'usen en la part del servidor creen un nou fil per cada client que estableix una connexió. Com cada fil creat consumeix recursos, els servidors estan bastant limitats quan existeixen moltes connexions simultànies.

Aquesta limitació motiva a crear Node.js, el qual és un llenguatge de programació JavaScript en la part del servidor capaç d'executar-se sense crear fils, i que gràcies a la seua execució asíncrona permet una gran escalabilitat.

Aquest llenguatge innovador és emprat per grans empreses com Google, Yahoo i Mozilla entre moltes altres, ja que és ràpid, de codi obert, fàcil de programar i escalable.

En aquest projecte, s'emprarà aquesta ferramenta per a crear de manera senzilla i ràpida el servidor web, que contindrà la futura aplicació web.

3.3 HTML

HTML (HyperText Markup Language) és un llenguatge de programació per elaborar pàgines web.

El codi HTML és interpretat per qualsevol navegador i s'utilitza per definir contingut a les pàgines web que, mitjançant etiquetes, es determinen quin tipus d'arxius van a incrustar-se (per exemple: vídeo, àudio i música), per poder després ser interpretat per qualsevol navegador web.

Per tant, HTML és un estàndard que defineix solament el contingut, no obstant això, es pot integrar codi JavaScript i CSS [6] per millorar i agregar noves funcionalitats a les planes web.

En aquest projecte s'emprarà aquesta ferramenta per agregar i definir botons, formularis, imatges etc.

3.4 CSS (Cascading Stylesheets)

CSS (Cascading Stylesheets) és un llenguatge de disseny gràfic que va nàixer al 1995 per canviar i millorar l'aspecte visual de la plana web HTML.

En aquest projecte s'emprarà aquesta ferramenta per definir la posició de cada element, color de fons de la pàgina web, color dels botons, ombres, transparències etc.

Per a veure la millora visual que es pot aconseguir en una web gràcies a aquesta ferramenta, a continuació es mostra la plana web de Twitter sense CSS (figura 9) i amb CSS (figura 10). S'ha canviat el codi font de Twitter, utilitzant la consola de desenvolupadors de Chrome.

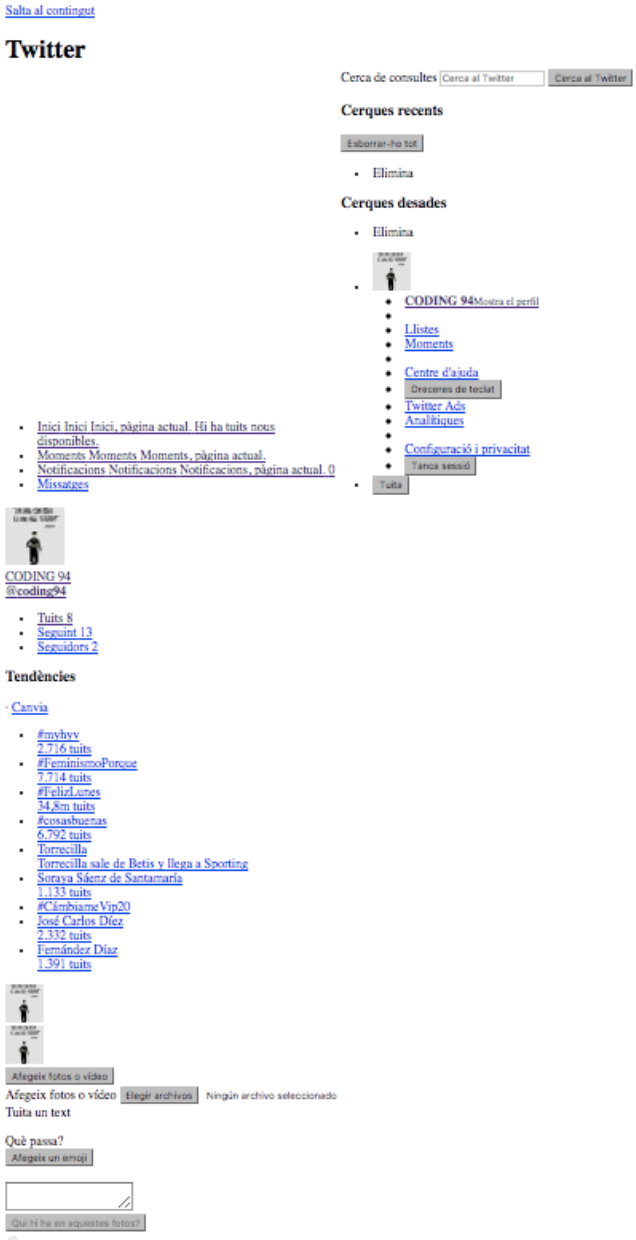

Figura 9. Twitter sense CSS

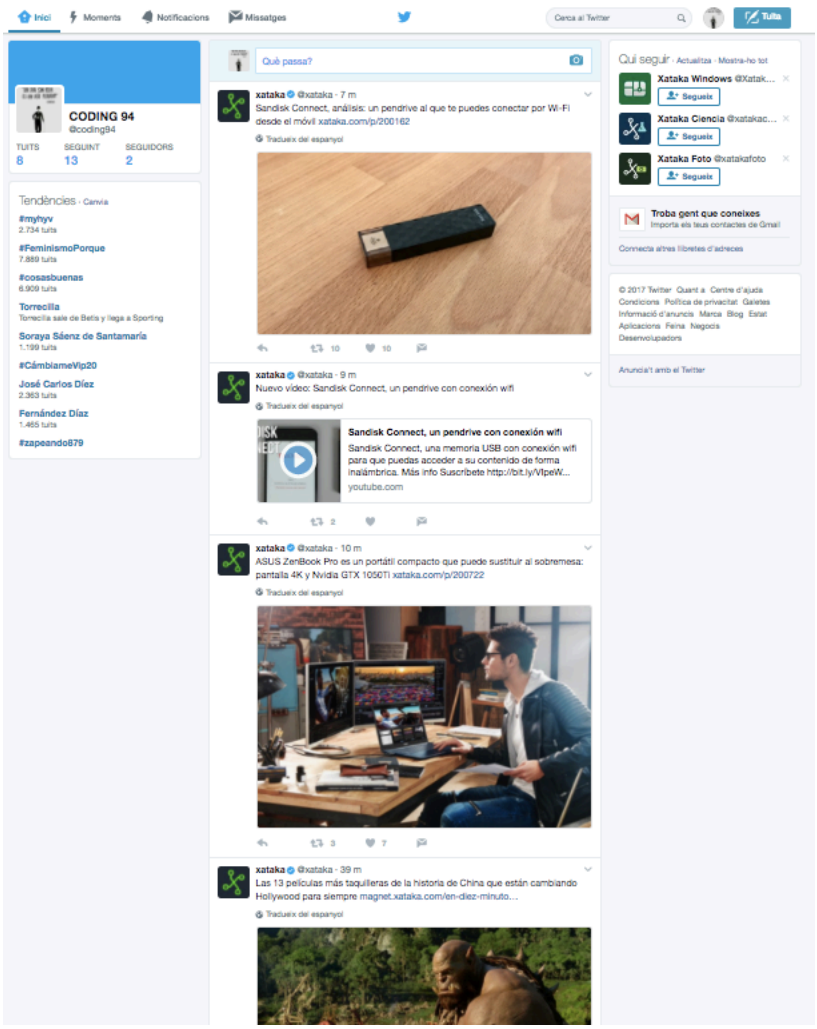

Figura 10. Twitter amb CSS

3.5 Postman [7]

El sistema d'alertes que va a implementar-se necessitarà llegir informació de les bases de dades de cada una de les xarxes socials, per posteriorment, poder detectar i mostrar en temps real les noves interaccions en cada una d'elles.

Les xarxes socials, almenys les que s'empraran en aquest projecte, tenen aplicacions web que, en altres paraules, són enllaços on es poden fer peticions per obtenir alguna determinada informació de les seues bases de dades.

Cada xarxa social proporciona unes instruccions per saber utilitzar la seua aplicació i així poder posteriorment, escriure el codi JavaScript que s'encarregarà de realitzar cada una de les peticions. No obstant això, de vegades, és difícil entendre com fer una petició i s'han de fer molts intents de prova i error, obligant al programador a reescriure el codi fins que finalment li funcione exitosament.

Per aquesta raó s'utilitzarà Postman (figura 11) que, sense escriure codi, es poden fer i guardar tot tipus de peticions indicant-li solament l'enllaç, les dades que s'han de configurar en la seua capçalera i les que s'han de configurar en el seu cos.

Per tant, en aquest projecte, Postman ajudarà a entendre cada una de les peticions, mitjançant proves d'èxit o error, per programar solament una volta aquell codi que obtinga exitosament les dades d'una determinada xarxa social.

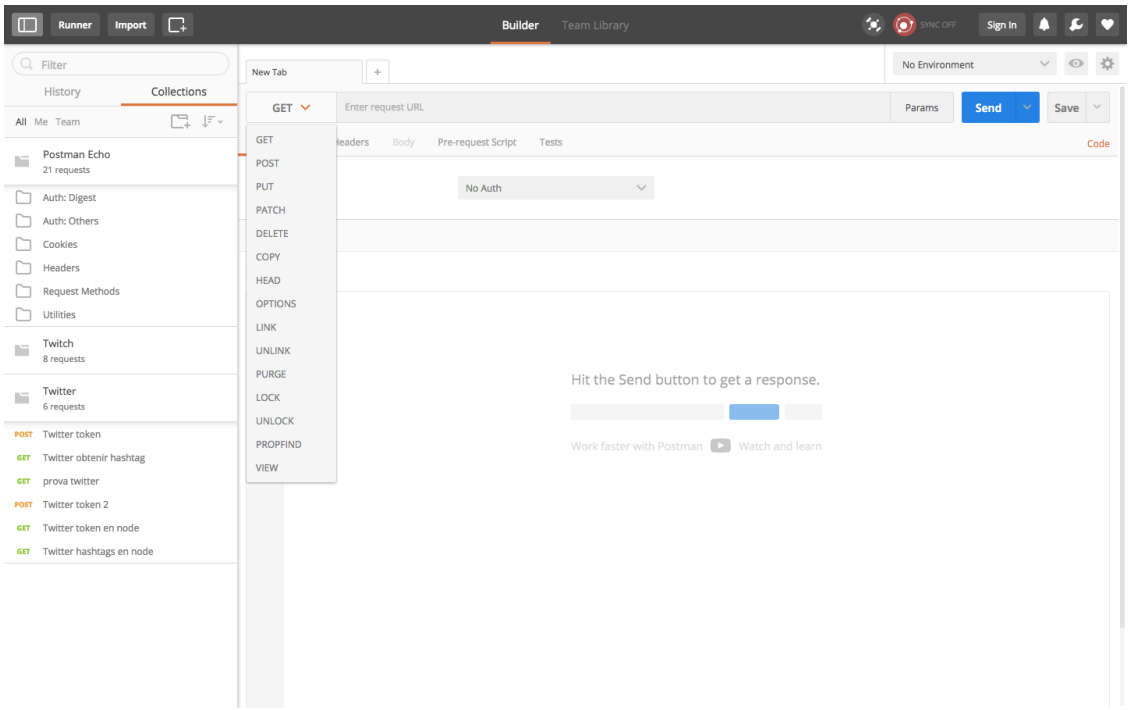

Figura 11. Postman

3.6 Twitter

Twitter és una xarxa social que permet publicar missatges de text d'una longitud màxima de 140 caràcters que, posteriorment, seran visualitzats per aquells usuaris que segueixen a l'usuari que els ha publicat.

En aquesta xarxa social (mostrada en la figura 10) s'utilitzen els *hashtags*, que en escriure el símbol "#" seguit d'una paraula, serveixen per indicar en el propi missatge de quin tema està parlant-se.

En aquest projecte es farà ús d'aquesta ferramenta per capturar i mostrar en forma d'alerta aquells missatges d'aquesta que continguen un determinat *hashtag*, el qual estarà configurat prèviament per l'usuari que vaja a usar la futura aplicació.

3.7 Strawpoll

Strawpoll (figura 12) és una plataforma web on quelcom, estiga registrat o no en aquesta, pot crear o participar en enquestes.

En aquest projecte es farà ús d'aquesta ferramenta per implementar l'alerta que mostrarà, en temps real, un rètol amb els resultats d'una determinada enquesta de Strawpoll, la qual serà elegida prèviament per l'usuari que vaja a usar la futura aplicació.

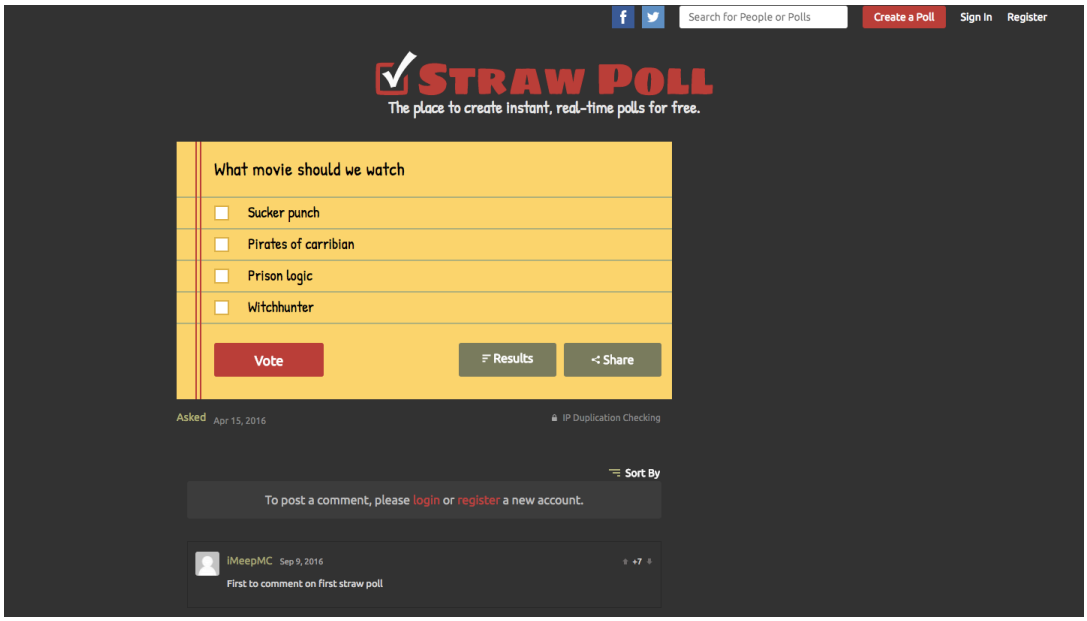

Figura 12. Strawpoll

3.8 OBS (Open Broadcaster Software) [8]

Open Broadcaster Software (figura 13) és una aplicació d'escriptori lliure i de codi obert que permet gravar i emetre per Internet en directe.

Aquest programa s'ha fet molt popular perquè, a part de ser lliure, permet incrustar i mesclar fàcilment en el vídeo arxius multimèdia, webcams, aspectes visuals de pàgines web i vídeojocs que estiguen jugant-se en el mateix ordinador. També permet instal·lar infinitat d'extensions que agreguen funcionalitats que no estan implementades de forma nativa (com per exemple mostrar en el vídeo la cançó que s'està escoltant en el reproductor multimèdia Winamp).

Aquest programa s'usarà per capturar les futures pàgines web que mostren les alertes i mesclar-les amb altres fonts, com per exemple, un vídeojoc i una webcam.

La mescla final de les diferents fonts multimèdia podrà ser emesa en directe per Internet o gravada.

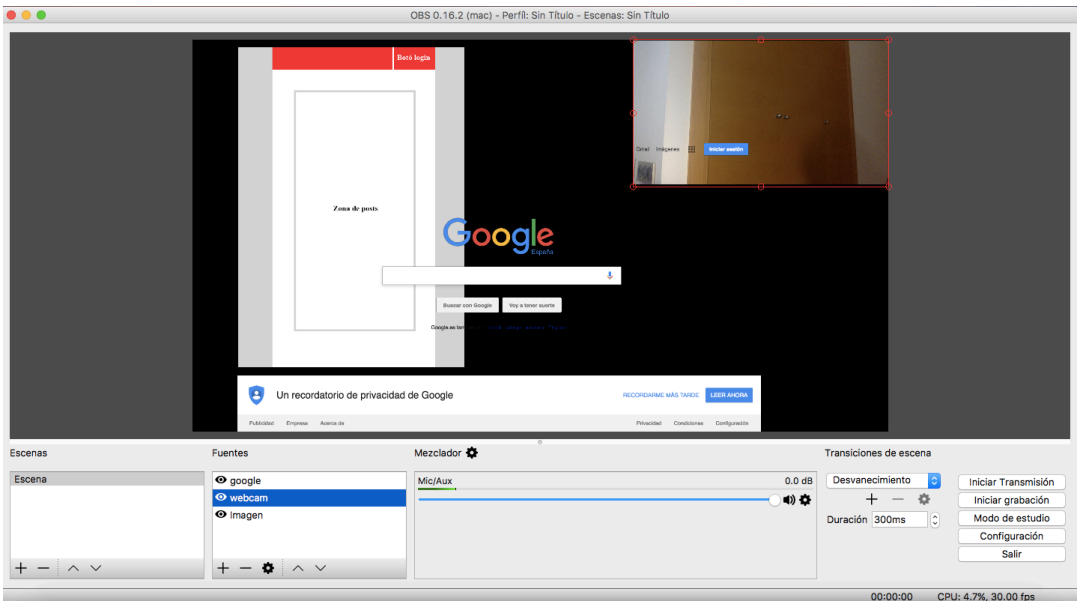

Figura 13. Aspecte visual del programa Open Broadcaster Software

3.9 Sublime Text [9]

En aquest projecte s'utilitzarà l'editor de text Sublime Text (figura 14) perquè està dissenyat per facilitar l'escriptura de quasi qualsevol codi de programació i que per tant, podrà estalviar temps a l'hora de programar la futura aplicació web.

Algunes de les característiques més importants d'aquest editor de text són les següents: autocompletar codi al prémer el tabulador, mostrar la jerarquia de tots els arxius del projecte, pintar el text segons les expressions pròpies de la sintaxi, poder plegar tota la sintaxi que està implícita en una etiqueta o funció, i possibilitat d'agregar extensions per afegir funcionalitats que no existeixen de forma nativa.

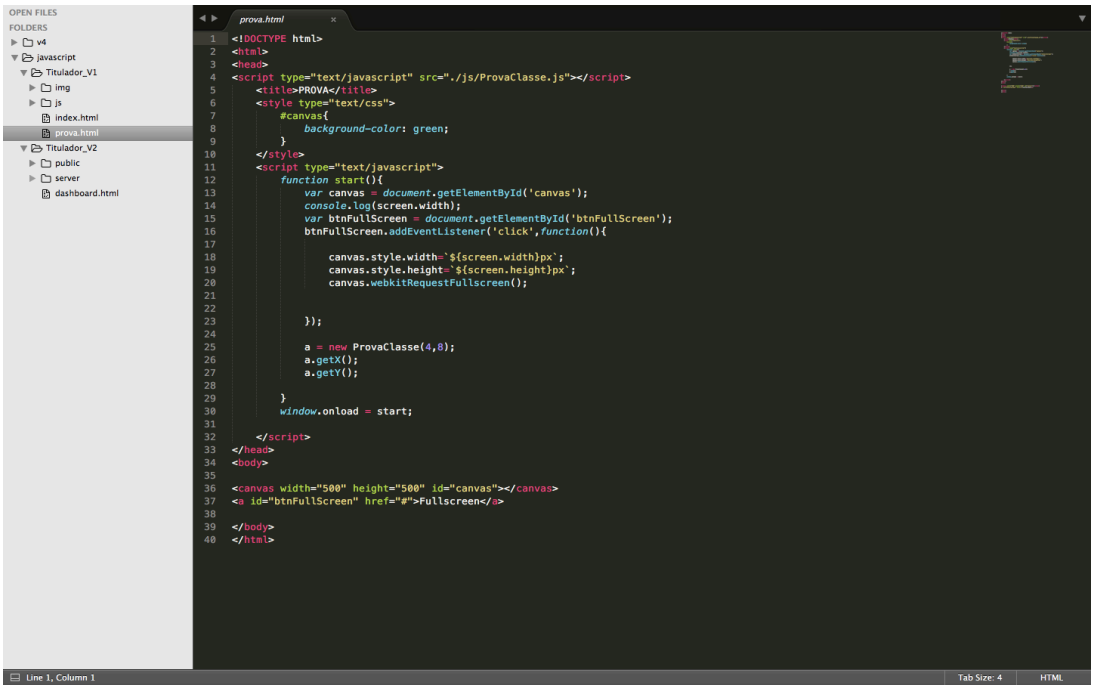

Figura 14. Aspecte visual de l'editor Sublime Text

Capítol 4. Disseny de l'aplicació web

4.1 Llibreria twitch_controller.js

Aquesta llibreria serà l'encarregada de connectar-se amb Twitch i estarà formada per les funcions que es mostren a continuació (figura 15a i 15b):

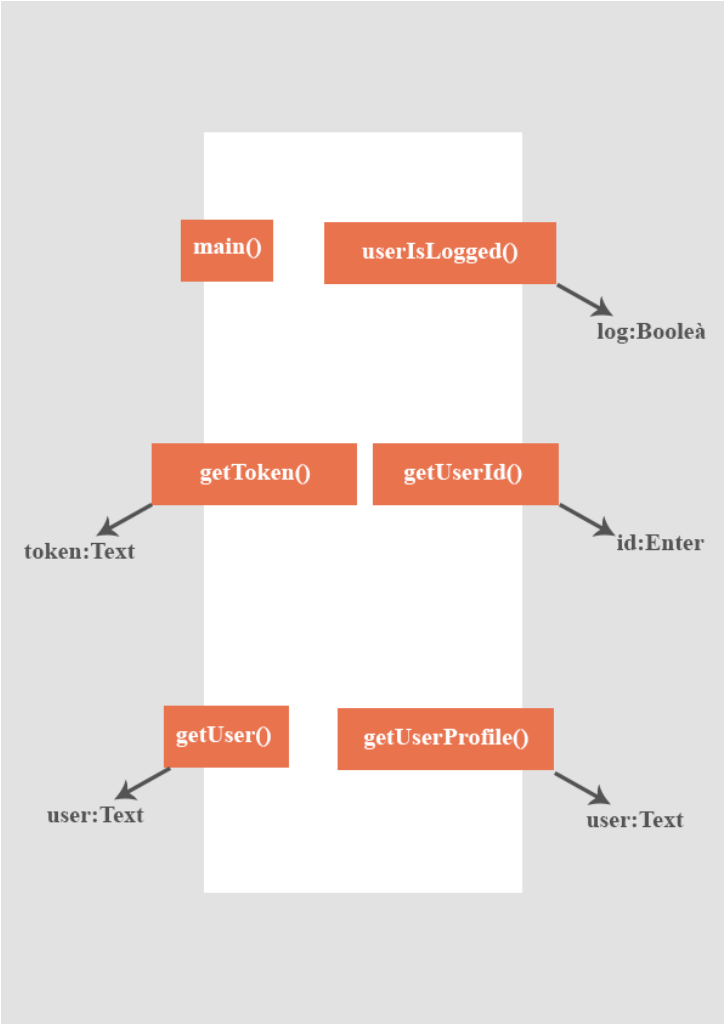

Figura 15a. Disseny de twitch_controller.js

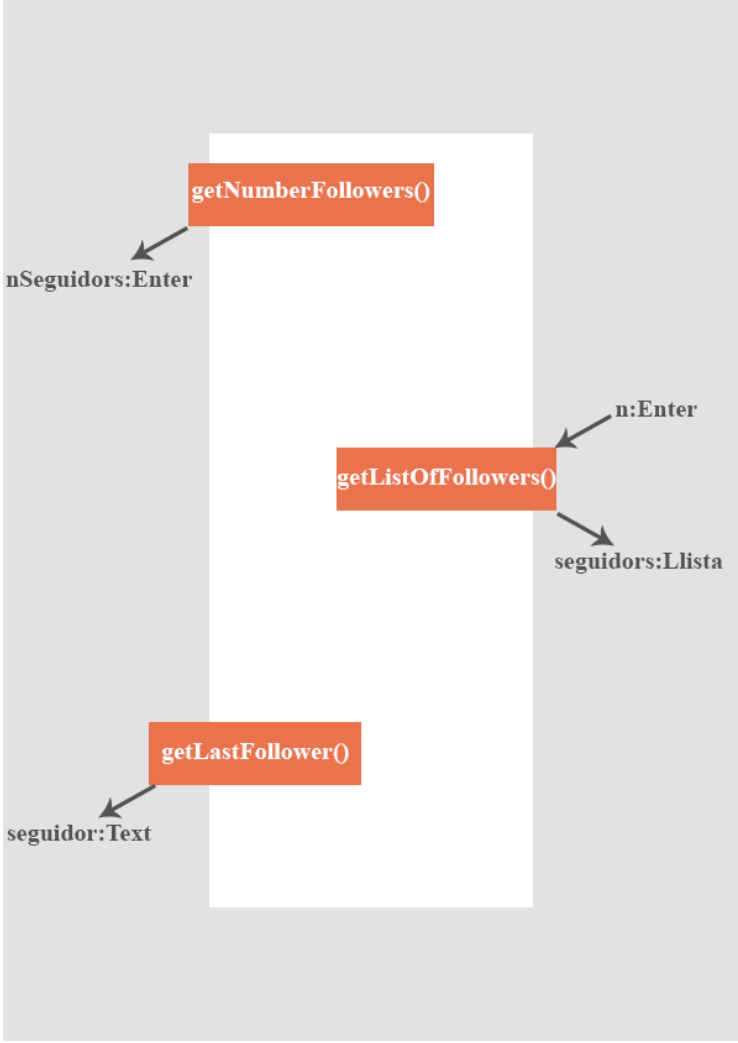

Figura 15b. Disseny de twitch_controller.js

La funció *main()* serà la primera en iniciar-se i s'encarregarà de detectar si el navegador té guardat el *token* que és facilitat prèviament per Twitch. La funció *main*() redirigirà o no a la plana web principal (zona de posts i d'iniciar sessió) segons el valor que retorne la funció *userIsLogged()*, explicada a continuació.

Per altra part, *userIsLogged()*, mitjançant la llibreria AJAX [10], serà l'encarregada d'enviar una petició a l'aplicació web de Twitch perquè aquesta conteste amb una variable de tipus *boolean*, sent vertader si el *token* de l'usuari que està guardat al navegador és vàlid, o fals si el *token* no és vàlid. Finalment, *userIsLogged()* retornarà fals si la contestació de l'aplicació web de Twitch és falsa i vertader si la contestació d'aquesta ha sigut vertadera.

Pel que fa a la funció *getToken()* llegirà el *token,* guardat en el navegador web, i el retornarà en format text.

La funció *getUserId()*, mitjançant la llibreria AJAX, serà l'encarregada d'enviar una petició a l'aplicació web de Twitch perquè aquesta conteste el número al qual està associat únicament a un determinat usuari.

Aquesta funció serà útil i imprescindible, ja que quan es vulga fer qualsevol petició a l'aplicació web de Twitch per obtenir unes determinades dades d'un compte concret, aquesta, sols identificarà l'usuari amb un número identificador, és a dir, quan és faça una petició sobre un usuari concret s'haurà d'indicar aquest amb un número que l'identifique i no amb el seu *nick*.

La funció *getUser()*, mitjançant la llibreria AJAX, farà una petició a l'aplicació web de Twitch, per saber de quin usuari és el *token* que està guardat en el navegador.

La funció *getUserProfile()* farà una petició a l'aplicació web de Twitch per saber l'enllaç de la imatge de perfil de l'usuari. Aquesta retornarà l'enllaç en format text.

La funció *getNumberFollowers()* s'encarregarà d'obtenir el nombre de seguidors que té l'usuari. Aquesta retornarà en format d'enter.

La funció anomenada *getLastFollower()*, comunicant-se amb l'aplicació web de Twitch, retornarà en format text el seguidor més recent del canal de l'usuari.

L'última funció que s'implementarà en aquest mòdul serà l'anomenada *getListOfFollowers()* que serà l'encarregada de retornar una llista dels seguidors més recents. Aquesta tindrà una entrada de valor de tipus enter per indicar quants seguidors incloure a la llista, i també una eixida de tipus llista amb format text per retornar aquests seguidors.

4.2 Llibreria Utilities.js

Aquest arxiu JavaScript s'ha creat per programar funcions útils i genèriques que s'utilitzaran moltes vegades en diferents pàgines web amb la finalitat d'evitar repetir i d'estalviar codi innecessàriament.

Com es pot veure a continuació (figura 16), almenys de moment, inclourà sols una funció, no obstant això, durant la implementació, se'n podran afegir més si en sorgeix la necessitat.

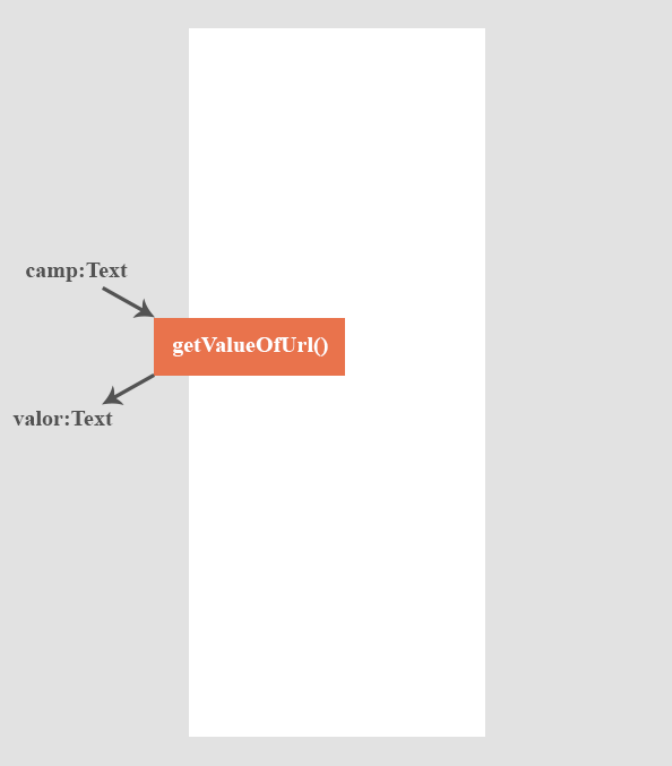

Figura 16. Disseny de la llibreria Utilities.js

Per entendre la funció *getValueOfURL()*, abans s'ha de saber la motivació d'aquesta.

Com ja s'ha comentat en punts anteriors, hi haurà alertes com la de Twitter o la de Strawpoll que s'hauran de configurar prèviament en una altra pàgina web mitjançant un senzill formulari, que una volta enviat, es redirigirà a l'alerta ja configurada.

Per poder passar la informació d'una pàgina (formulari), a una altra (l'alerta), s'ha plantejat una opció fàcil que consisteix a incloure-la en el propi enllaç de l'alerta.

Per tant, al enviar-se el formulari, es redirigirà a la direcció web de l'alerta però, aquesta estarà modificada, ja que s'afegiran cadenes de text addicionals que seran les dades de configuració que necessitaran algunes d'aquestes alertes. A continuació, es mostra el format de l'enllaç que s'usarà en aquest projecte per afegir i passar informació addicional:

http://localhost/Twitter_alert.html?hashtag=bondia

Com es pot veure s'han afegit les paraules 'hashtag' i 'bondia', sent la primera el camp i l'altra el valor de configuració.

Per aquest motiu, la llibreria "Utilities" inclourà la funció *getValueOfURL()* que, mitjançant l'enllaç, tindrà com a finalitat interpretar quins valors corresponen a cada un dels camps.

4.3 Llibreria Overlay.js

La llibreria Overlay serà utilitzada per totes les alertes i s'encarregarà de generar rètols mitjançant l'etiqueta *canvas* que proporciona de forma nativa JavaScript.

Aquesta llibreria es comportarà com una classe o objecte i tindrà un constructor que donarà valors a les dues variables internes *obj* i *ctx*.

A continuació es mostra el disseny (figura 17) del futur arxiu "Overlay" per poder posteriorment explicar totes i cada una de les seues funcions.

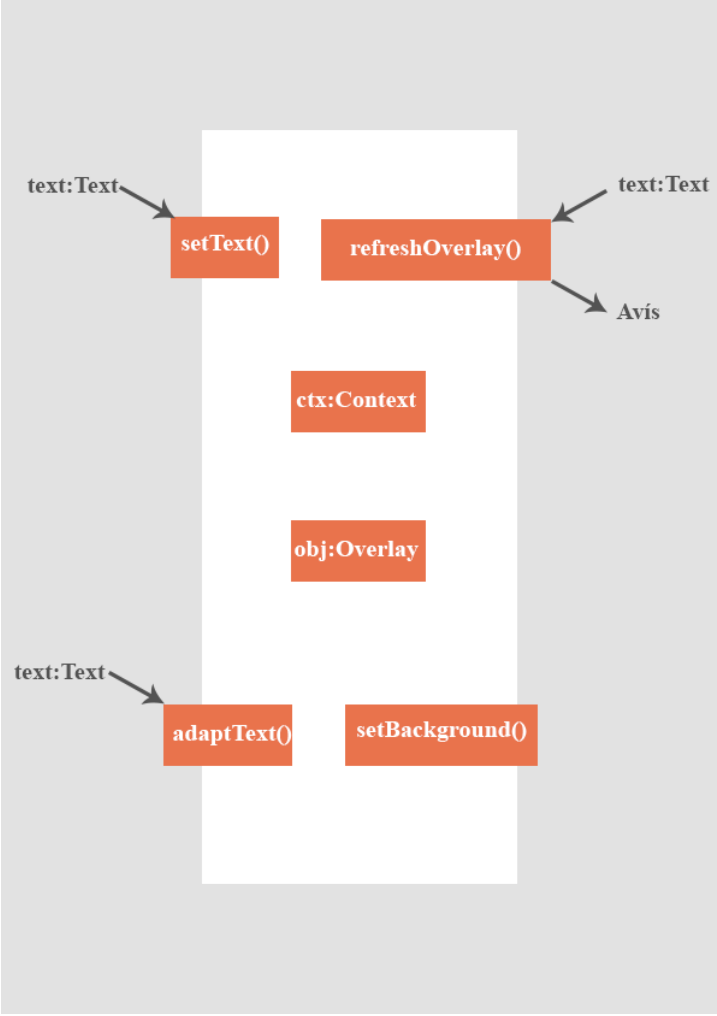

Figura 17. Disseny de la llibreria Overlay.

Abans de començar amb les funcions, cal explicar de manera il·lustrada de què estarà composta la variable de tipus Overlay (figura 18), ja que aquesta serà un objecte de tipus *JSON* que contindrà un format que no es pot explicar implícitament en la figura 17.

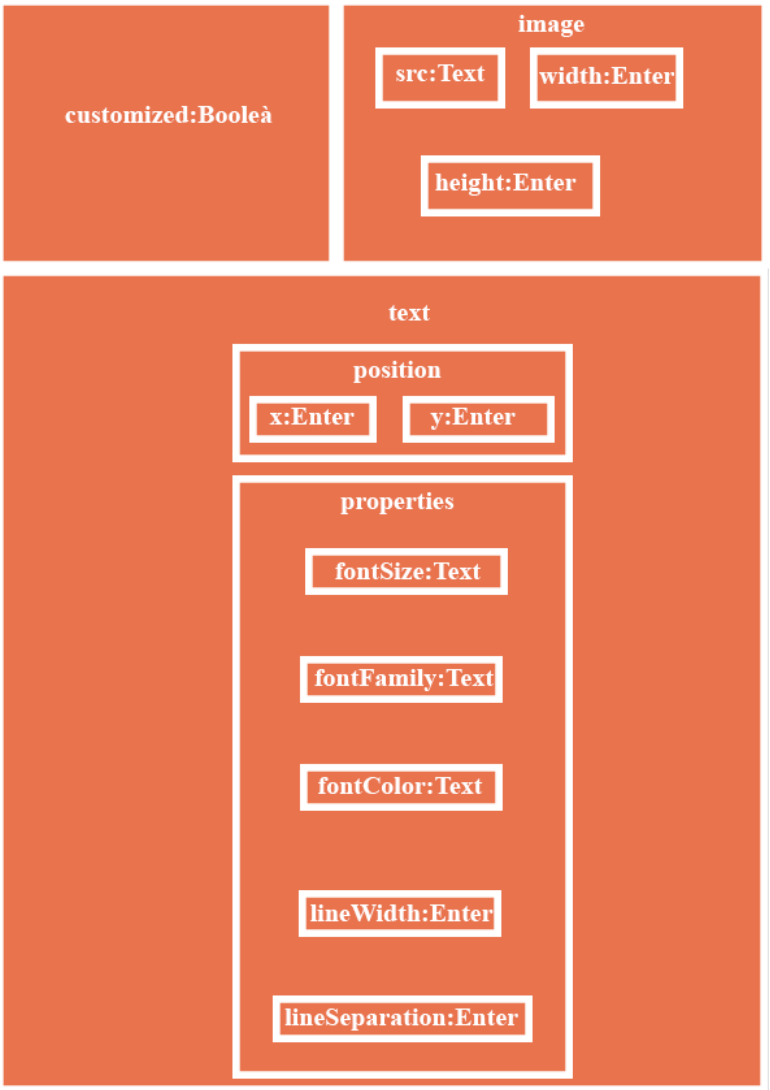

Figura 18. Estructura de la variable de tipus Overlay.

Tal i com es pot veure en la figura 18, la variable de tipus Overlay serà l'encarregada de donar-li format al rètol mitjançant la variable *customized* i els objectes imatge i text.

Per una part, la variable *customized* es vol afegir per si en un futur s'implementara un sistema que permetera a l'usuari personalitzar un determinat rètol. D'aquesta manera, el sistema d'alertes podria detectar si el rètol ha estat modificat o és l'original.

Per altra part, l'objecte anomenat *image* tindrà les variables *src*, *width* i *height*. La primera indicarà la ruta de la imatge del rètol i la segona i la tercera informaran sobre la grandària de la d'aquesta.

Per últim, l'objecte *text*, contindrà els objectes *position* i *properties*. El primer, que tindrà dues variables de tipus enter, serà l'encarregat de definir la posició del text en el rètol, i el segon serà l'encarregat de definir la grandària del text, el tipus de llera, el color, els marges i la separació que hi haurà entre línies.

Tornant a l'explicació de la llibreria Overlay, aquesta contindrà la variable interna *ctx* i les funcions *setText(), adaptText(), refreshOverlay()* i *setBackground()*.

Pel que fa a la variable interna *ctx*, inclosa en el constructor, serà de tipus Context. Aquest tipus de variable servirà per poder identificar l'element *canvas* que estarà incrustat al HTML, amb la finalitat de poder aplicar en aquest canvis gràfics mitjançant JavaScript.

La funció *setText()* de la llibreria Overlay simplement s'encarregarà de canviar o actualitzar el text que es mostre en el rètol.

Per altra part, la funció *adaptText()* implementarà l'opció d'adaptar el text en funció de la informació que estarà implícita en la variable *obj*.

Ja per finalitzar, la funció *setBackground()* serà l'encarregada de posar un fons al *canvas*.

4.4 Arxiu postGenerator.js

Aquest arxiu servirà per generar els posts a la pàgina principal. Aquest mòdul de JavaScript es podria no implementar, no obstant això, no seria escalable, ja que per cada post escrit s'hauria d'escriure més codi HTML.

Per aquesta raó s'afegirà aquest mòdul que generarà el codi HTML resultant de tots els posts a partir d'un objecte JSON de tipus posts, el qual es mostra en la figura 19.

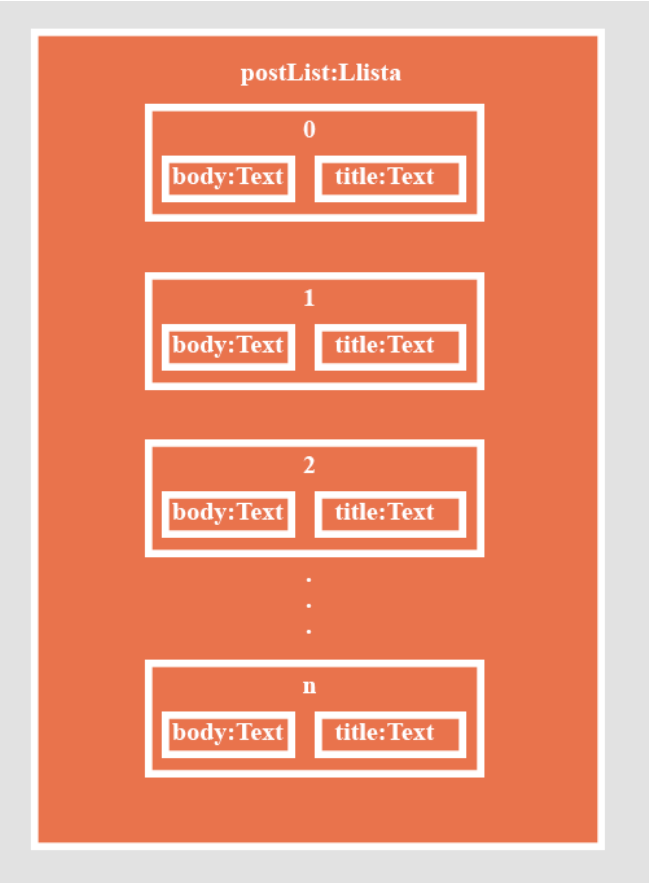

Figura 19. Estructura de la variable de tipus posts.

Com es pot observar, l'objecte de tipus posts contindrà una llista d'objectes que tindran les variables *body*, que emmagatzemaran els texts dels continguts dels posts, i *title*, que emmagatzemaran els títols dels posts.

A continuació, amb ajuda de la figura 20, s'expliquen les funcions que hi haurà implementades en aquest mòdul.

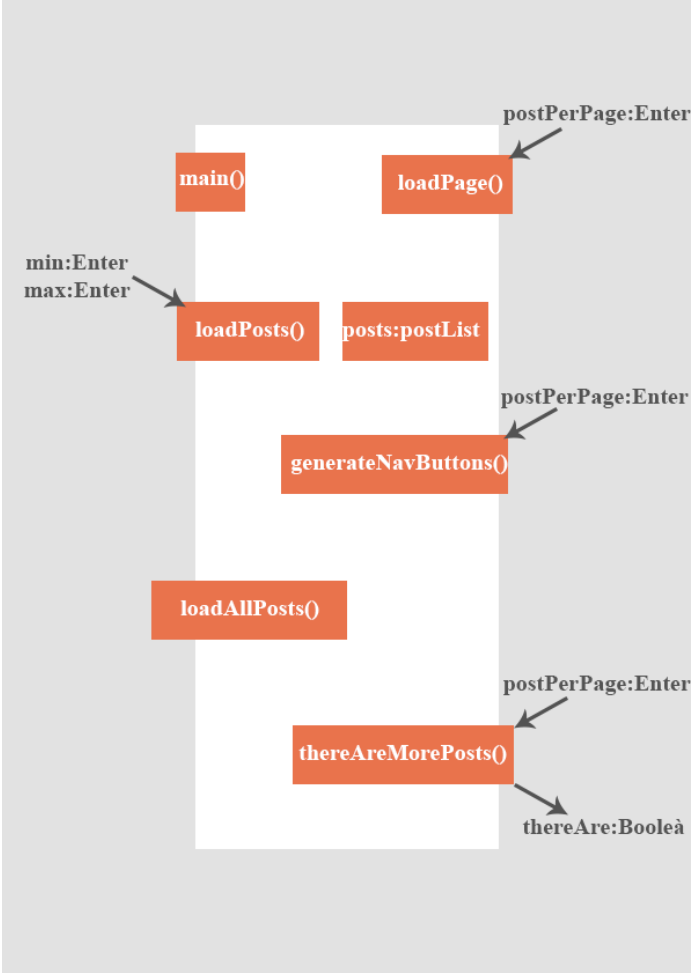

Figura 20. Disseny del mòdul postGenerator.js

La funció *main()*, serà l'encarregada d'inicialitzar les funcions que s'hagen d'executar quan es carregue la pàgina web.

Per altra part, la funció *loadPosts()* generarà un interval de posts a la plana web mitjançant codi HTML. Aquest interval es determinarà a les entrades de la funció, sent la variable *min* el número del primer post a generar i *max*, el número de l'últim.

Pel que fa a la funció *loadAllPosts()*, tindrà una funcionalitat pareguda a *loadPosts()* però amb la diferència que aquesta generarà codi HTML per mostrar tots els posts en una mateixa pàgina web.

En aquest mòdul, també s'implementarà la funció *loadPage()* que tindrà l'objectiu de determinar l'interval de posts a mostrar en funció del número de la pàgina que estiga mostrant-se al navegador.

Per mostrar els posts en vàries pàgines, serà necessari que es puga accedir a la pàgina següent i a l'anterior mitjançant dos botons. Aquesta funcionalitat s'implementarà en la funció *generateNavButtons()*, que mitjançant JavaScript, generarà HTML que mostren els botons anteriorment esmentats.

L'última funció que s'implementarà en aquest mòdul serà *thereAreMorePosts()*, que retornarà vertader si hi ha més posts a mostrar en la pàgina següent, i fals si ja no en queden més. Aquesta funció serà útil a l'hora de generar els botons de navegació que canvien de pàgina, ja que si es detecta que no se'n poden mostrar més, els botons de navegació es podran configurar perquè no

donen opció a l'usuari d'accedir a la pàgina següent, perquè canviar de plana web implicaria mostrar una pàgina sense contingut.

4.5 Arxiu auth.js

Com s'ha observat anteriorment, la plana web que s'implementarà donarà l'opció d'iniciar sessió mitjançant comptes de Twitch.

Com pot intuir-se, Twitch no donarà accés a la base de dades on estan emmagatzemades les contrasenyes de cada compte ja que és informació personal. És per això que Twitch utilitza un sistema de *tokens* en la seua aplicació web [11] [12] que identifiquen de manera única a un usuari durant un interval de temps, que en altres paraules, és com una clau d'accés temporal que identifica a un compte concret.

Perquè quede clar i es puga entendre la utilitat d'aquest mòdul, abans de continuar, es mostren els passos que es necessitaran per poder iniciar sessió en la futura web utilitzant comptes de Twitch (figura 21).

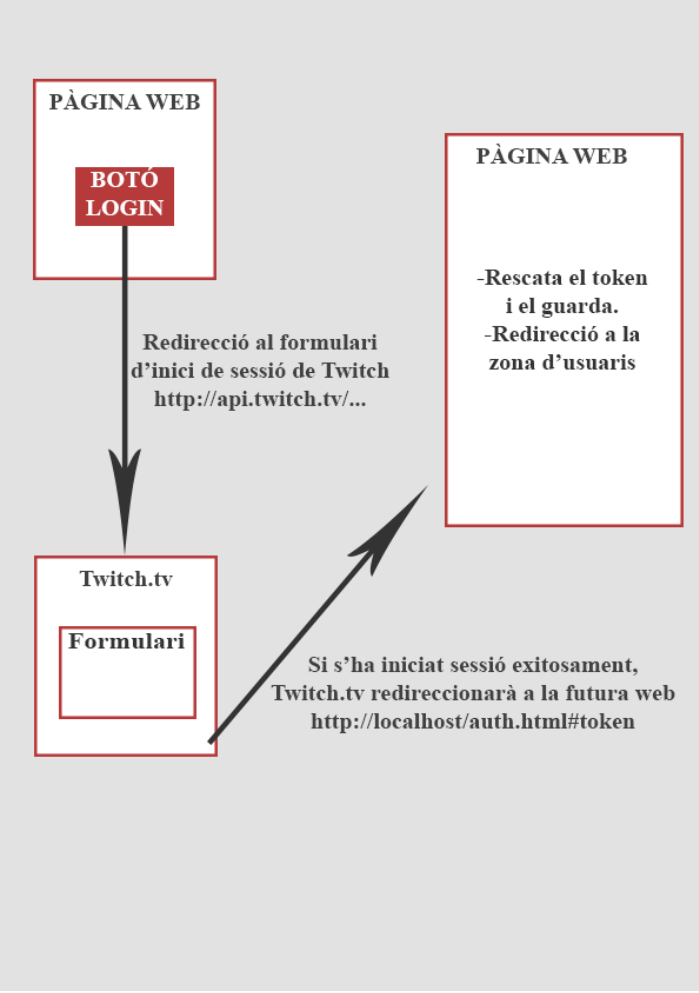

Figura 21. Passos per iniciar sessió a través de Twitch

Per tant, quan l'usuari inicie sessió exitosament, Twitch redirigirà a la futura aplicació web que va a implementar-se i incorporarà en l'enllaç de redirecció, una cadena de text que serà el *token*.

Aleshores, la funcionalitat d'aquest mòdul serà poder extraure el *token* que estarà implícit en l'enllaç de redirecció de Twitch per poder posteriorment guardar-lo de manera local en el navegador. D'aquesta forma, en qualsevol moment i encara que es canvie de plana web, es

podrà accedir sempre al *token* amb la finalitat d'identificar de manera constant de quin usuari es tracta. Una vegada s'hagen realitzat tots aquests passos, aquest mòdul redirigirà a la secció d'usuaris.

Ja per acabar, cal remarcar que aquest mòdul que implementarà el sistema d'inici de sessió serà imprescindible, ja que l'obtenció del *token* serà necessària a l'hora de mostrar algunes alertes de Twitch, perquè sense aquest, hi haurà informació a què no es podrà accedir, és a dir, aquesta clau també determinarà els privilegis d'accés d'informació que té cada usuari.

4.6 Arxiu dashboard.js

Aquest arxiu s'executarà en la secció d'usuaris i serà l'encarregat de generar els menús de les diferents alertes, i també la foto de perfil i el nom d'usuari d'aquell compte que en un futur inicie sessió.

Igual que en apartats anteriors, abans explicar totes i cadascuna de les funcions, és necessari mostrar l'enginyeria inversa (figura 22).

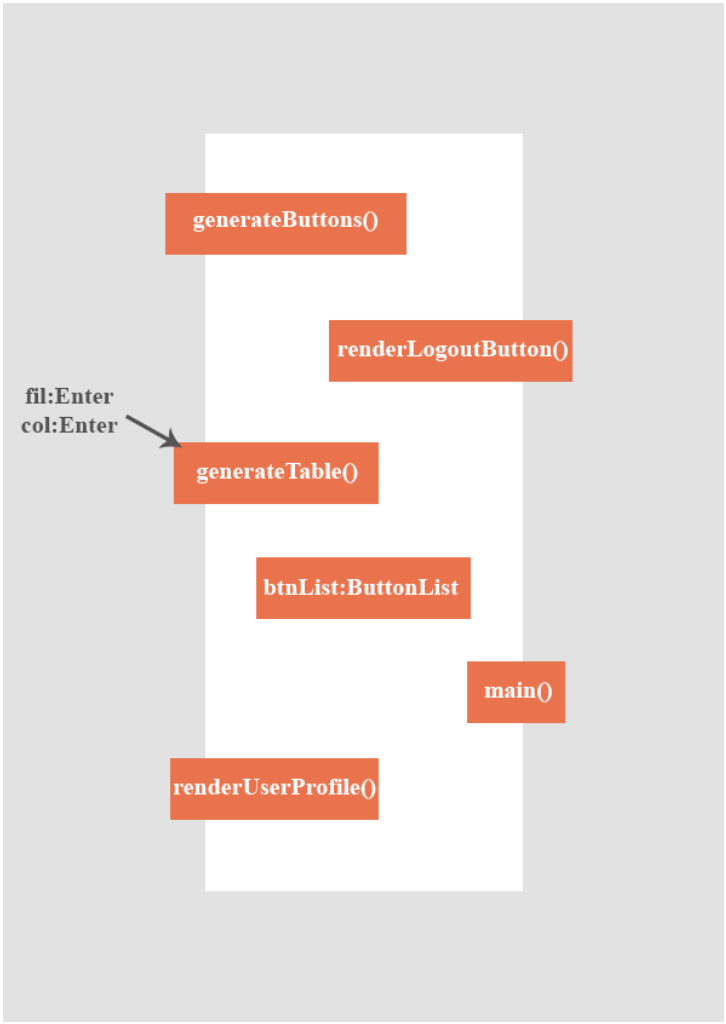

Figura 22. Disseny del mòdul dashboard.js

Pel que fa al component anomenat ButtonList que es mostra en la figura 22, és un objecte que conté una llista de botons que estan definits mitjançant les variables de tipus text *title*, *id* i *link*. La primera determina quin text es mostrarà en els botons, la segona defineix un identificador i la

tercera conté l'adreça web que redirigirà cada botó. A continuació es mostra gràficament l'estructura que té aquest objecte (figura 23).

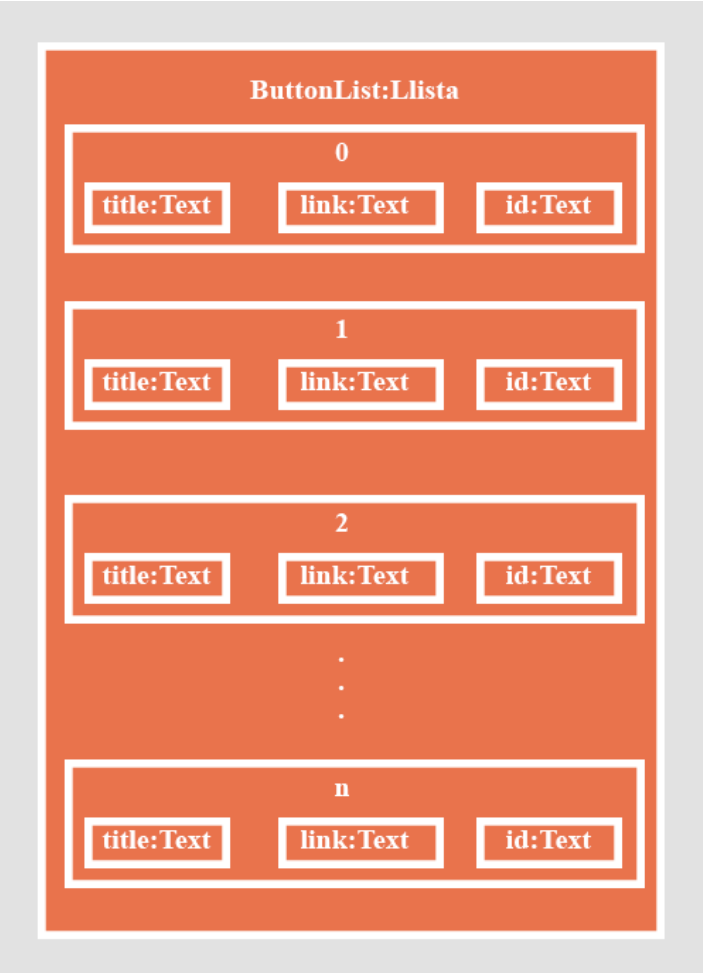

Figura 23. Estructura de la variable de tipus ButtonList.

Una vegada explicat l'objecte de tipus *buttonList* ja es pot entrar en detall d'allò que faran totes i cadascuna de les funcions d'aquest mòdul.

En primer lloc, la funció *generateTable()* tindrà com a objectiu generar una taula que posteriorment s'encarregarà de contenir tots i cadascun dels botons que redirigiran a cadascuna de les alertes, amb la finalitat que aquests apareguen alineats verticalment i horitzontalment. *GenerateTable()* tindrà dues entrades que definiran el nombre de files i columnes que es voldran generar, les quals es determinaran en funció de la quantitat de botons que vagen a generar-se en aquesta.

En segon lloc, la funció *generateButtons()* tindrà com a objectiu col·locar un botó en cada graella de la taula que prèviament s'haurà creat amb *generateTable()*.

Pel que fa a les funcions *renderUserProfile()* i *renderLogoutButton()*, la primera mostrarà en la pàgina web la imatge de perfil i el nom d'aquell usuari que haja iniciat sessió, i la segona generarà un botó per poder tancar-la.

Per últim, la funció *main()* inicialitzarà les funcions prèviament explicades.

4.7 Arxiu follower_alert.js

Aquest mòdul s'executarà en la plana web "Alerta Seguidor", i tindrà l'objectiu de mostrar un rètol quan hi haja un nou seguidor al canal Twitch de l'usuari que visualitze aquesta futura pàgina web.

A continuació es mostra (figura 24) i s'explica totes i cadascuna de les funcions que s'implementaran en aquest mòdul.

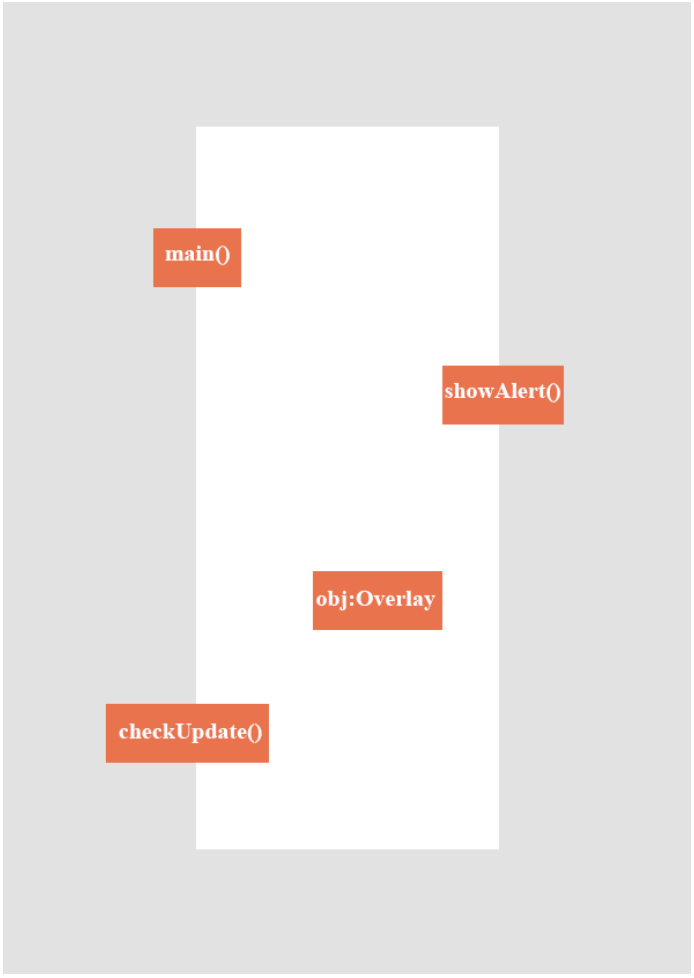

Figura 24. Disseny del mòdul follower_alert.js

En primer lloc, la funció *checkUpdate()* s'encarregarà de comprovar si el nombre de seguidors de Twitch ha canviat. En cas d'haver-hi un canvi en el nombre de seguidors aquesta funció cridarà a *showAlert()*.

En segon lloc, *showAlert()* s'encarregarà de mostrar durant uns segons el nom del nou seguidor en forma de rètol. Aquesta funció canviarà dinàmicament el CSS per ocultar (quan s'acaba l'alerta) o mostrar (quan es llança l'alerta) l'element on està situat el rètol.

En tercer lloc, la funció *main()* serà l'encarregada d'inicialitzar les funcions d'aquest mòdul.

Per finalitzar, és important saber que la classe Overlay serà utilitzada en una funció anomenada *showAlert()* en qualsevol mòdul que mostre alertes, amb l'objectiu de generar rètols que estaran formats per un text i una imatge.

4.8 Llibreria Twitter_controller.js

Aquest mòdul s'encarregarà de comunicar-se amb l'aplicació web de Twitter [13] [14] per poder accedir als seus *tweets*. Per tant, aquest arxiu serà important ja que permetrà al mòdul Twitter_alert.js establir una connexió per poder llegir i mostrar alertes de *tweets* d'un *hashtag* determinat.

Pel que fa al disseny, aquesta llibreria tindrà les següents funcions (figura 25):

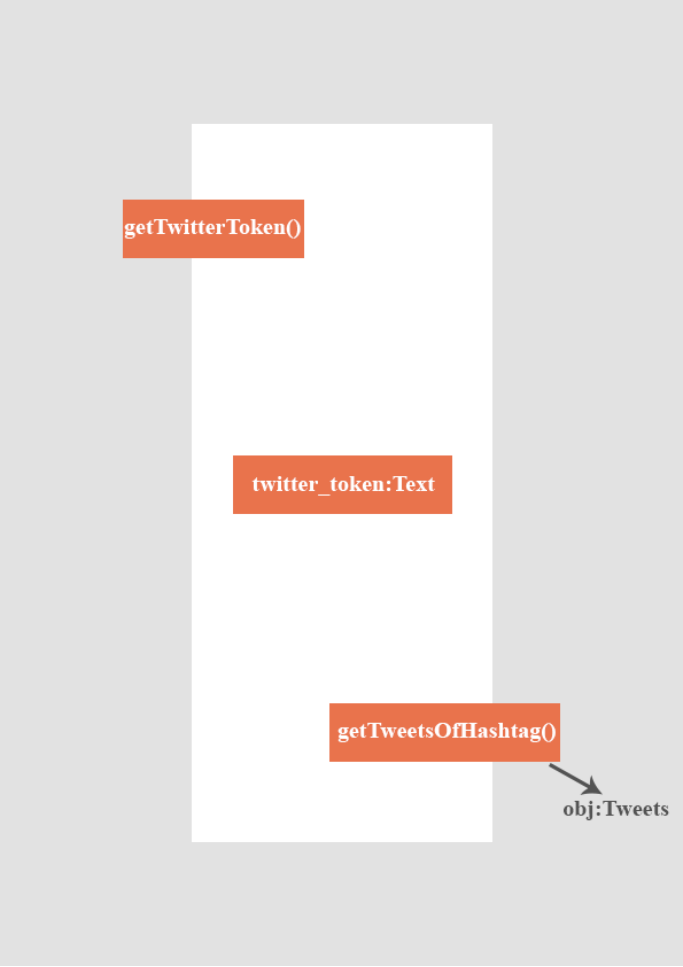

Figura 25. Disseny de la llibreria Twitter_controller.js

En primer lloc, la funció *getTwitterToken()*, serà l'encarregada d'obtenir el *token* de Twitter i guardar-lo en la variable *Twitter_token*.

Aquest *token* identificarà l'aplicació que anteriorment s'haurà registrat en la zona de desenvolupadors de Twitter, amb l'objectiu de poder tindre autorització per accedir a les dades d'aquesta xarxa social, és a dir, el *token* que obté aquesta funció servirà perquè l'aplicació web, a l'hora de fer-li peticions, identifique la futura aplicació web per controlar a quina informació es podrà tindre accés.

En segon lloc, la funció *getTweetsOfHashtag()* s'encarregarà de demanar a l'aplicació web de Twitter els *tweets* més recents d'un determinat *hashtag*. Aquest *hashtag* serà definit en la URL tal i com s'explica l'apartat 4.2

Per últim, l'eixida de la funció *getTweetsOfHashtag()* serà un objecte de tipus *tweets* format per una llista d'estats. Cada estat contindrà una variable anomenada *user* i altra anomenada *text*. La primera definirà el nom d'usuari de Twitter i la segona definirà el contingut del *tweet*.

A continuació, en la figura 26, es mostra el disseny de la variable de tipus *tweets* amb la finalitat de facilitar la seua comprensió.

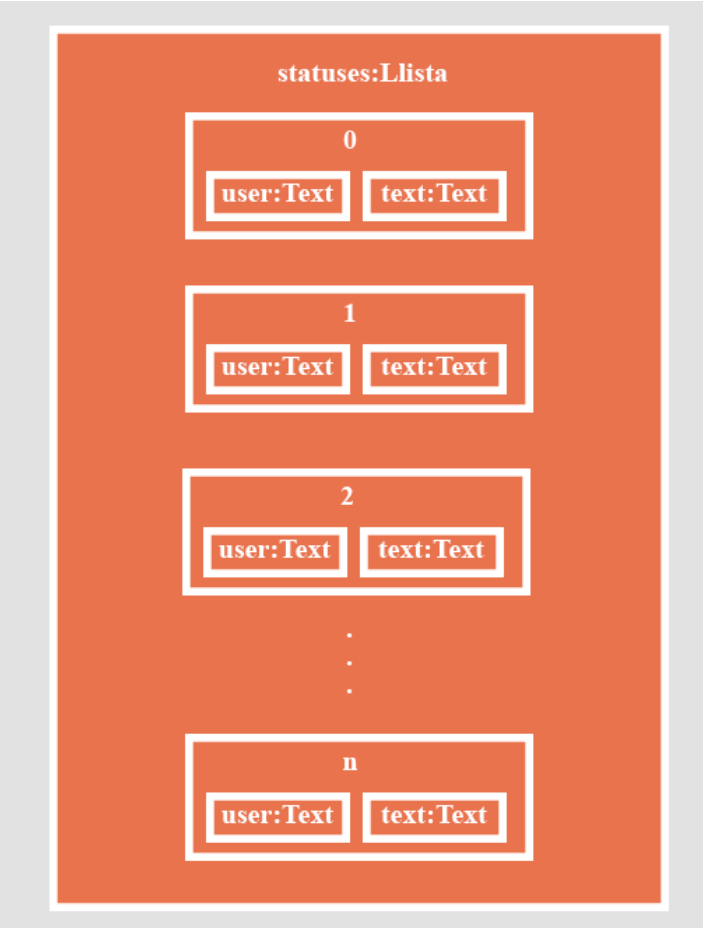

Figura 26. Estructura de la variable de tipus Tweets

4.9 Arxiu Twitter_alert_conf.js

Aquest mòdul (figura 27) s'executarà en la plana web de configuració del *hashtag* i tindrà com a objectiu afegir text a la URL que redirigirà a l'alerta de Twitter una vegada s'haja enviat el formulari. Aquest text afegit indicarà a l'alerta de Twitter el *hashtag* elegit.

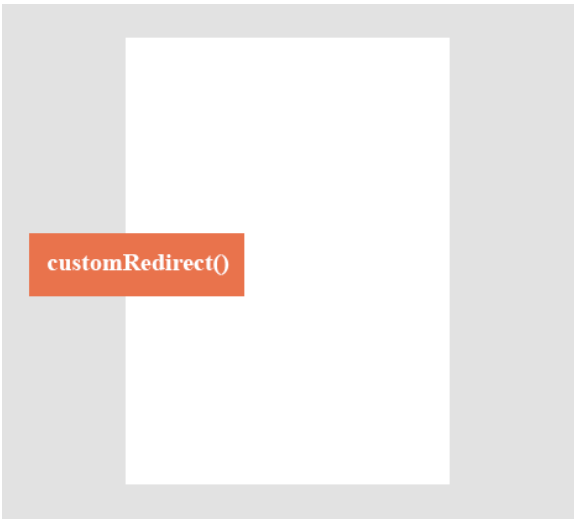

Figura 27. Disseny del mòdul Twitter_alert_conf.js

Per tant, la funció *customRedirect()* és llançarà quan s'envie el formulari i redirigirà a l'enllaç *http://localhost/Twitter_alert.html?hashtag=xxx*, sent *xxx* el *hashtag* configurat.

4.10 Arxiu Twitter_alert.js

Aquest arxiu s'executarà en la plana web que mostra els *tweets* d'un determinat *hashtag* (configurat prèviament en el formulari de la secció de configuració del hashtag) amb l'objectiu de mostrar i actualitzar els *tweets* més recents mitjançant un *canvas*.

A continuació s'expliquen les funcions que s'implementaran en aquest mòdul amb l'ajuda de la figura 28.

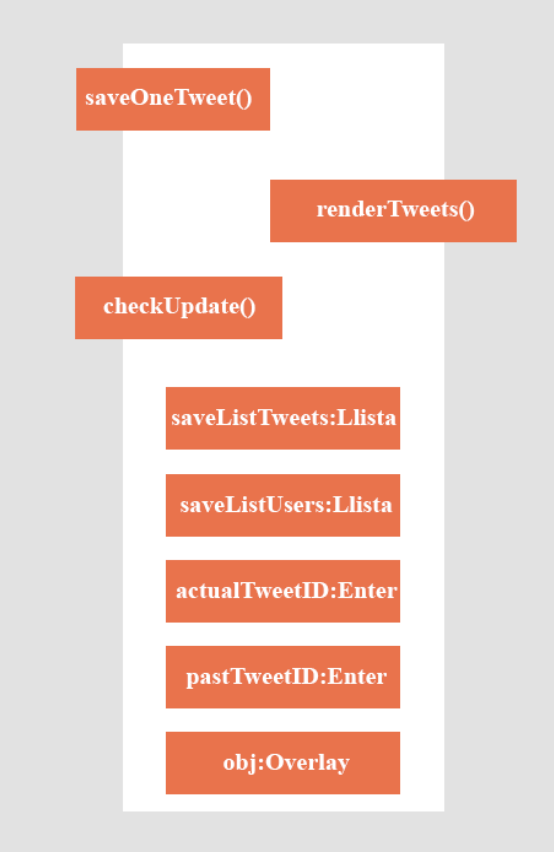

Figura 28. Disseny del mòdul Twitter_alert.js

En primer lloc, la funció *checkUpdate()* s'encarregarà de detectar si hi ha *tweets* nous comparant les variables *pastTweetID* i *actualTweetID*. Aquestes variables guardaran cadascuna un número que identificarà a un *tweet*, sent la primera l'identificador del *tweet* més recent de la comprovació anterior i la segona, l'identificador del *tweet* més recent de la comprovació actual.

En altres paraules, es comprova si el *tweet* més recent ha canviat respecte a la comprovació anterior.

Cada vegada que es detecte un nou *tweet*, es cridarà a la funció *saveOneTweet()* que s'encarregarà d'agregar aquest nou *tweet* a la variable *saveListTweets* i també, d'agregar a la variable *saveListUsers* l'usuari que l'haja escrit.

Quan la llista de la variable *saveListTweets* tinga 15 elements, es llançarà la funció *renderTweets()* que tindrà l'objectiu de recórrer la llista dels *tweets* i la llista dels autors de cadascun d'ells per a posteriorment mostrar-los en forma de ràfega en un rètol.

Una vegada s'haja llançat *renderTweets()*, les variables que emmagatzemen els *tweets* i els usuaris s'esborraran per poder tornar a repetir el procés.

4.11 Arxiu Strawpoll_conf.js

Aquest mòdul (figura 29) que s'executarà en la web anomenada "Configuració d'enquesta", tindrà la mateixa funcionalitat que Twitter_alert_conf.js (explicat en l'apartat 4.9). No obstant això, s'ha implementat perquè el missatge de configuració que anirà implícit en l'enllaç de redirecció no serà el mateix.

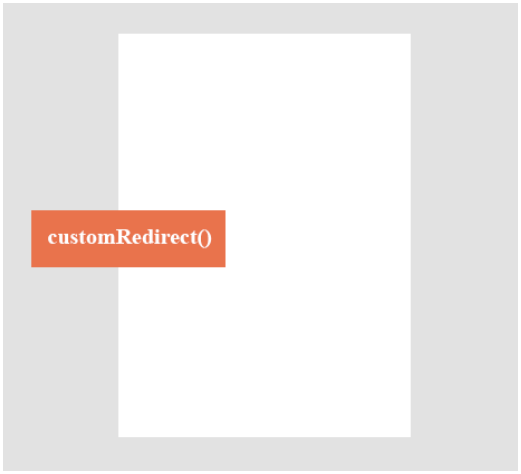

Figura 29. Disseny del mòdul Strawpoll_conf.js

Per això, la funció *customRedirect()* és llançarà quan s'envie el formulari i redirigirà a l'enllaç *http://localhost/Strawpoll.html?poll=yyy*, sent *yyy* l'identificador de l'enquesta que es mostrarà en la secció anomenada "Alerta Strawpoll".

4.12 Arxiu Strawpoll.js

Aquest arxiu s'executarà en la secció "Alerta Strawpoll" amb la finalitat de mostrar una determinada enquesta actualitzada en temps real.

A continuació és mostra i s'explica el disseny d'aquest mòdul (figura 30) que ajudarà a realitzar posteriorment la seua implementació.

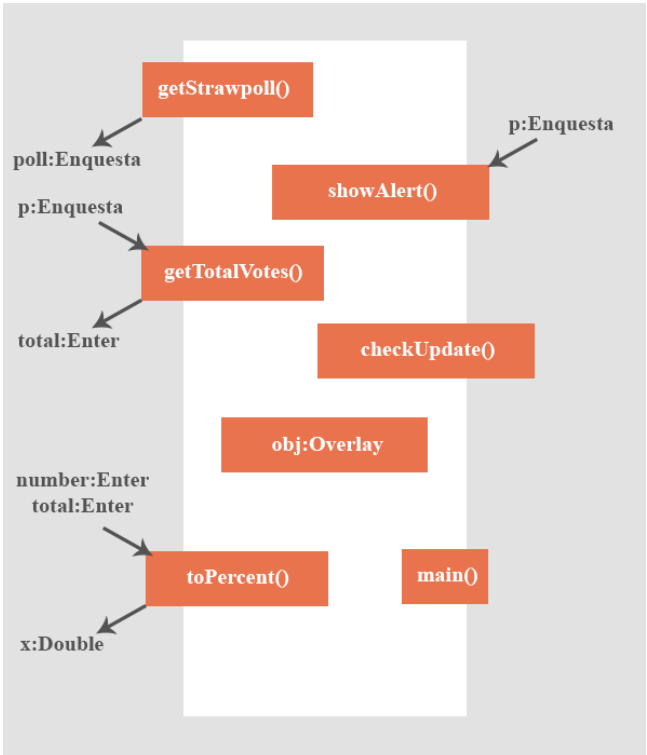

Figura 30. Disseny del mòdul Strawpoll.js

En primer lloc, la funció *getStrawpoll()* s'encarregarà de comunicar-se en l'aplicació web de Strawpoll [15] per obtenir una determinada enquesta que serà retornada mitjançant un objecte de tipus Enquesta (figura 31).

En segon lloc, la funció *getTotalVotes()* calcularà els vots totals que tindrà una determinada enquesta.

En tercer lloc, la funció *toPercent()* convertirà la quantitat de vots en percentatges.

Per altra part, *checkUpdate()* s'encarregarà de detectar nous resultats de l'enquesta que estiga mostrant-se. Quan es detecten nous vots, aquesta funció llançarà *showAlert()* que renderitzarà els nous resultats.

Per últim, la funció *main()* inicialitzarà les funcions anteriorment explicades.

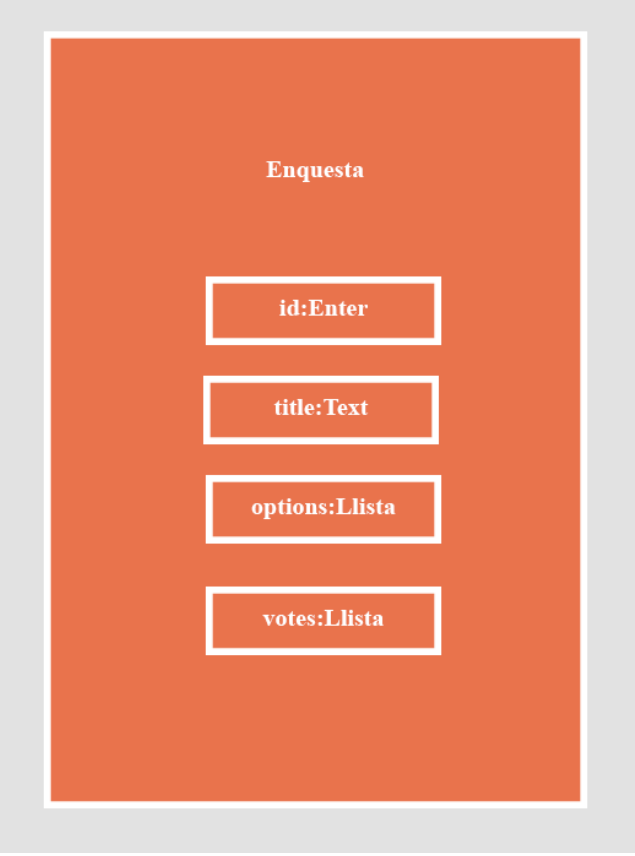

Figura 31. Estructura de la variable de tipus Enquesta

4.13 Arxiu follower_list.js

Aquest mòdul s'executarà en la secció web anomenada "Llista de seguidors" i s'encarregarà de renderitzar la llista de seguidors actualitzada en temps real i en forma de rètol.

A continuació, es mostra (figura 32) i s'explica el disseny d'aquest mòdul amb la finalitat de facilitar la seua comprensió.

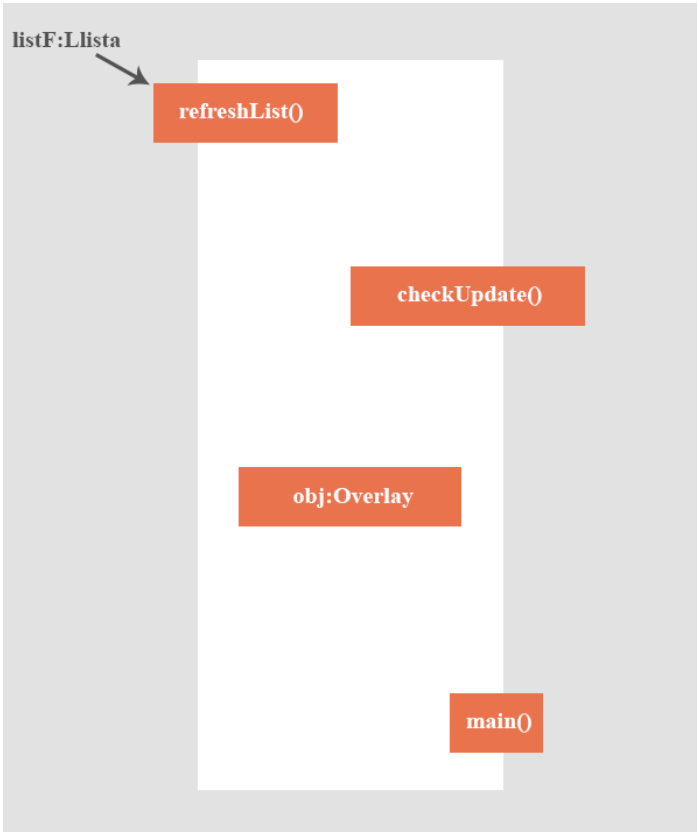

Figura 32. Disseny del mòdul follower_list.js

Per una part, la funció *checkUpdate()* comprovarà si els últims seguidors han canviat. En cas que hi haja nous seguidors, *checkUpdate()* llançarà la funció *refreshList()* que serà l'encarregada d'actualitzar i renderitzar de nou el rètol.

Per altra part, la funció *main()*, inicialitzarà *checkUpdate()* i *refreshList()*.

Capítol 5. Implementació

Aquest capítol és important perquè durant la implementació poden sorgir problemes que obliguen a no seguir completament el disseny anteriorment plantejat.

Per tant, quan es realitza el disseny no es tenen en compte els problemes que poden sorgir durant la implementació, ocasionant que aquests obliguen a modificar-lo puntualment amb l'objectiu de fer funcionar correctament l'aplicació.

En els següents apartats s'explica la jerarquia final dels fitxers que formen l'aplicació web, i també els problemes més importants que han sorgit durant la implementació amb la finalitat de mostrar el resultat final del projecte.

5.1 Organització de fitxers

El projecte resultant té les carpetes principals anomenades "api", "public" i "node_modules". La primera, programada en Node.js, conté el servidor web encarregat de servir els arxius HTML (continguts en la carpeta "public"), la segon conté tots els arxius que formen l'aplicació web i la tercera conté els arxius que necessita Node.js per poder funcionar.

La carpeta "public" conté les subcarpetes "CSS", "img", "js" i "logos". La primera emmagatzema els mòduls programats en CSS, la segona conté les imatges que s'usaran en els rètols, la tercera guarda els mòduls JavaScript i la carpeta "lib" que conté les llibreries, i la quarta, conté el *favicon* (imatge en extensió ".ico" que serveix per mostrar el logo de la web en la pestanya del navegador).

Ja per finalitzar, es mostra en la figura 33 una captura de totes les carpetes i arxius que formen el projecte amb l'objectiu de clarificar l'explicació d'aquest apartat.

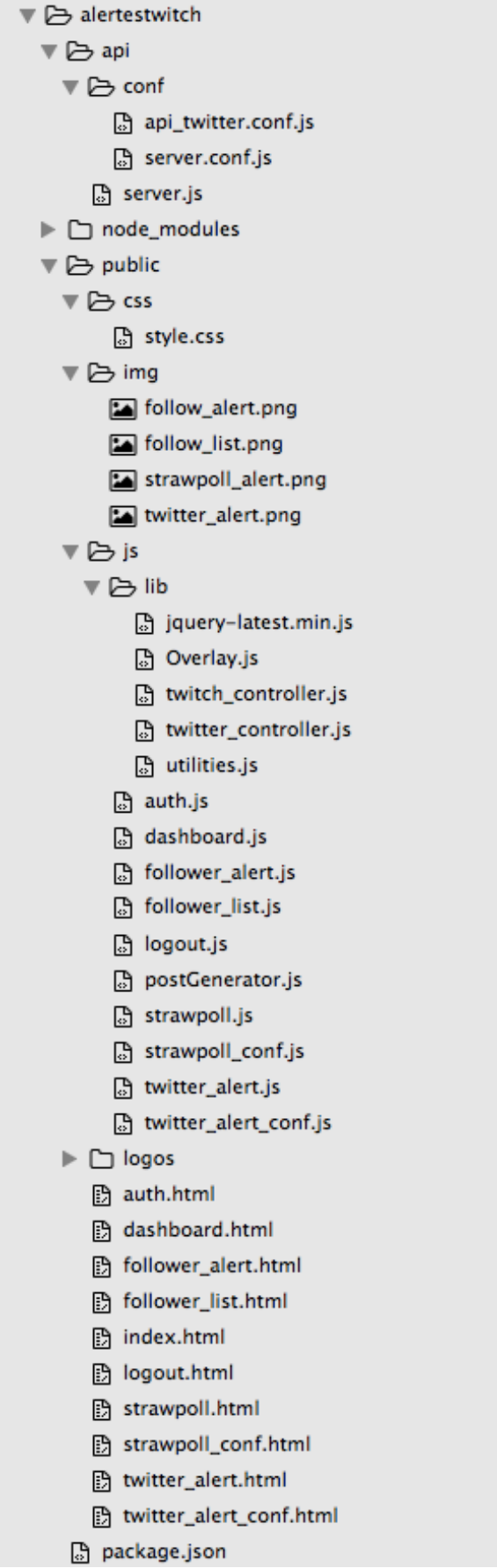

Figura 33. Organització dels fitxers

5.2 Problemes trobats

5.2.1 Renderització de rètols

Com ja s'ha comentat anteriorment en aquesta memòria, les alertes detecten la informació en temps real i quan hi ha novetats, aquestes actualitzen el text i la imatge que conformen el rètol.

El problema ocorria en aquesta actualització de text, ja que l'element *canvas* no podia reescriure el text provocant que es mostrara el text superposat al anterior.

Per a solucionar-ho, s'ha decidit que en cada actualització gràfica es renderitze primerament una capa de color verd amb la finalitat de fer desaparéixer el rètol anterior.

Pel que fa a l'elecció d'aquest color, s'ha utilitzat perquè la majoria dels emissors de vídeo tenen filtres de croma (fons de color verd), és a dir, que poden eliminar el fons verd.

5.2.2 Peticions a l'aplicació de Twitter

Quan es va dissenyar el mòdul que es comunicava en l'aplicació de Twitter no es va tindre en compte que aquesta no permetia peticions de clients web.

Durant la implementació es va comprovar que les pàgines estàtiques de la carpeta públic no podien fer peticions per problemes de seguretat.

Per tant, s'ha implementat la funció que feia les peticions en el servidor. Aquesta serà llançada quan un determinat client faça la petició a aquest, és a dir, el servidor reenviarà les peticions del client a l'aplicació de Twitter i posteriorment, el servidor enviarà la resposta de l'aplicació de Twitter al client.

5.2.3 Peticions a l'aplicació de Strawpoll

En un principi, les peticions a l'aplicació de Strawpoll, per obtenir una enquesta en format JSON, funcionaven en Postman però no en clients web.

Després d'algunes setmanes sense solucionar el problema, es va optar per posar-se en contacte amb els propietaris d'aquesta plataforma.

Finalment l'equip de desenvolupadors va trobar el problema el qual estava en la pròpia aplicació de Strawpoll i el van solucionar.

Una vegada l'equip de Strawpoll va actualitzar la seua aplicació web, les peticions ja funcionaven correctament.

5.2.4 Autentificació en programes emissors

Com ja s'ha comentat anteriorment, per mostrar les alertes es necessita indicar un *token* vàlid (obtingut prèviament en el sistema d'inici de sessió de Twitch) a l'aplicació de Twitch, ja que si no es redirecciona a la pàgina web "index.hml".

El problema és que l'emissor de vídeo no passa pel sistema d'inici de sessió (solament carrega la pàgina web de l'alerta) i per tant no pot guardar cap *token* per poder obtenir dades de l'aplicació de Twitch.

La solució que s'ha emprat és incloure el *token* que identifica l'usuari de Twitch en l'enllaç de l'alerta com a informació addicional amb la finalitat que l'emissor puga obtenir-lo i emmagatzemar-lo.

Capítol 6. Guia d'ús

6.1 Guia d'instal·lació

Per poder posar en funcionament l'aplicació web dissenyada, sols cal instal·lar un programa anomenat Node.js i escriure unes poques instruccions en la terminal de l'ordinador per iniciar-la.

En primer lloc, s'ha d'accedir a la web *https://nodejs.org/es/download/* i clicar en el botó de descàrrega per obtenir l'instal·lador. Una vegada descarregat, s'ha d'executar per instal·lar-ho.

En segon lloc, cal copiar la carpeta del projecte anomenada "alertestwitch" en qualsevol carpeta de l'ordinador (en cas de Windows, es recomana copiar-la al directori arrel, és a dir, *C:*).

Per últim, s'ha d'obrir la terminal i escriure el següent (cal substituir *[directori_del_projecte]* pel directori on s'ha copiat la carpeta del projecte):

cd [directori_del_projecte]/api

node server.js

Si tot ha anat bé, una vegada escrites les instruccions anteriors, apareixerà en la terminal el següent text:

servidor escoltant en el port 8080

Per accedir a l'aplicació web, entrar a localhost:8080 en un navegador

Una vegada realitzats els passos anteriors, per poder entrar a l'aplicació implementada, s'ha d'accedir a la direcció *localhost:8080* en un navegador web.

6.2 Instruccions d'ús

6.2.1 Inici de sessió

En aquesta secció (figura 34) l'usuari pot llegir publicacions per aprendre a usar aquesta aplicació web o també pot iniciar sessió polsant en el botó "Login".

Per poder usar i accedir a les alertes d'aquesta web es necessita introduir les credencials d'un compte de Twitch (en el botó "Login").

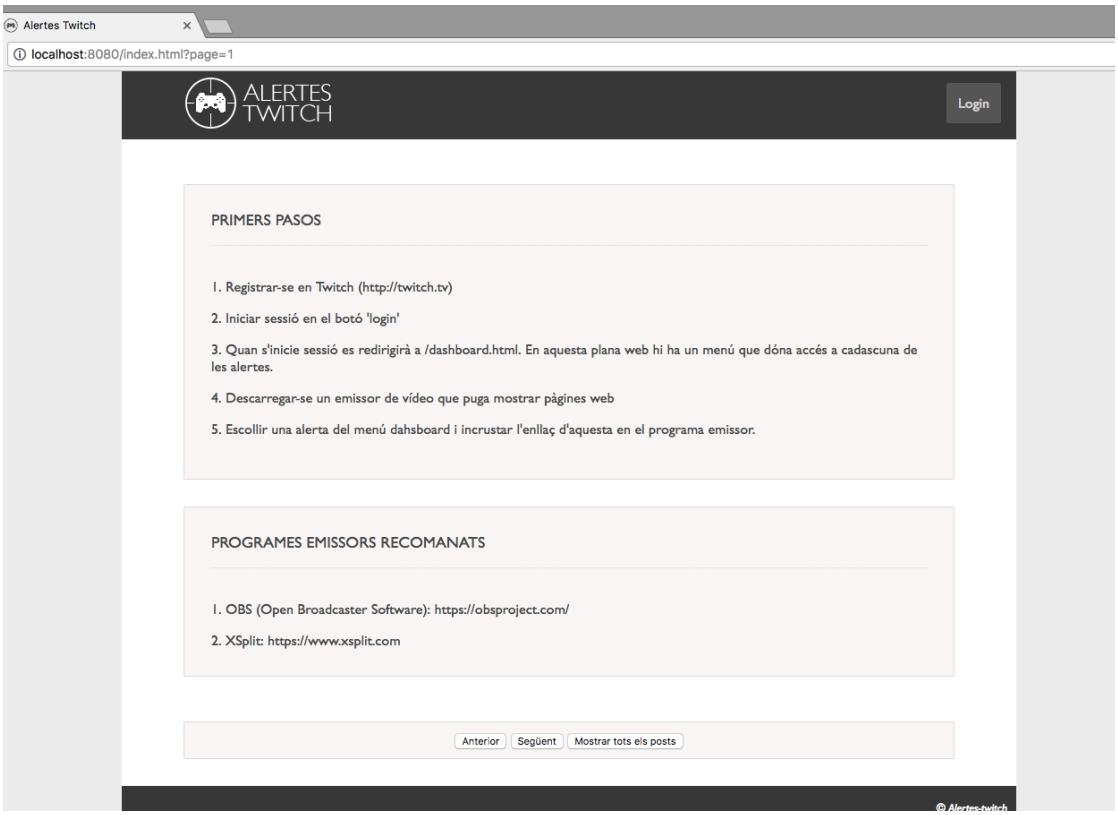

Figura 34. Plana web index.html

6.2.2 Secció d'usuaris

Si l'inici de sessió és exitós, es redirigeix a aquesta pàgina web (figura 35) que consisteix en un menú que dóna accés a cadascuna de les alertes.

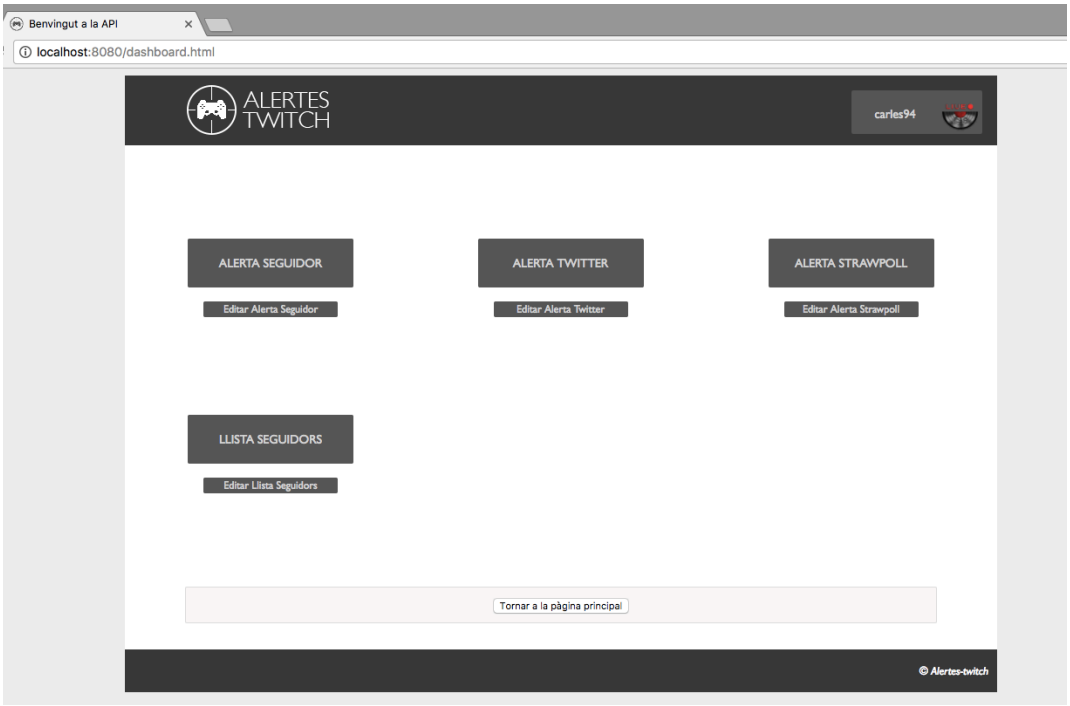

Figura 35. Pàgina web dashboard.html

6.2.3 Alertes

En primer lloc, quan se seleccione en el menú (figura 35) l'alerta de seguidors, es mostrarà un fons verd a l'espera que hi haja nous seguidors en el canal de Twitch. Quan hi haja nous seguidors es mostrarà un rètol com el que s'il·lustra en la figura 36.

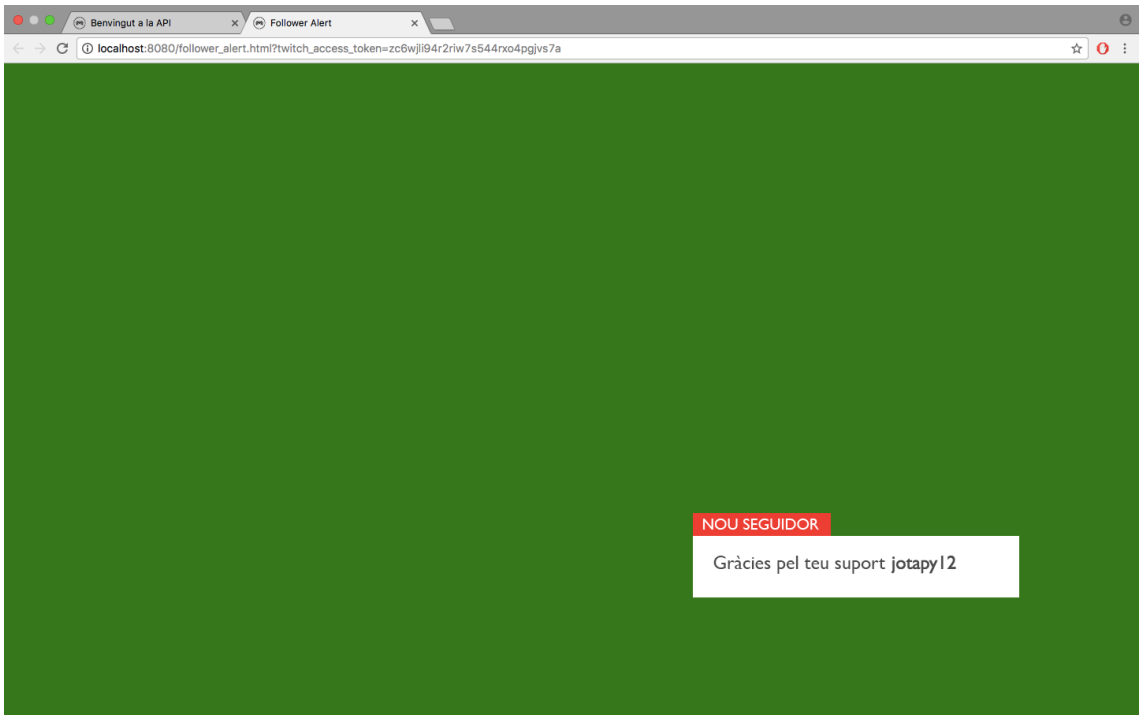

Figura 36. Pàgina web follower_alert.html

Per altra part, quan es trie l'alerta de Twitter en el menú de la pàgina web dashboard.html, es redirigirà a un formulari (figura 37) per indicar a l'aplicació el *hashtag* que han de tindre els tweets que es mostraran.

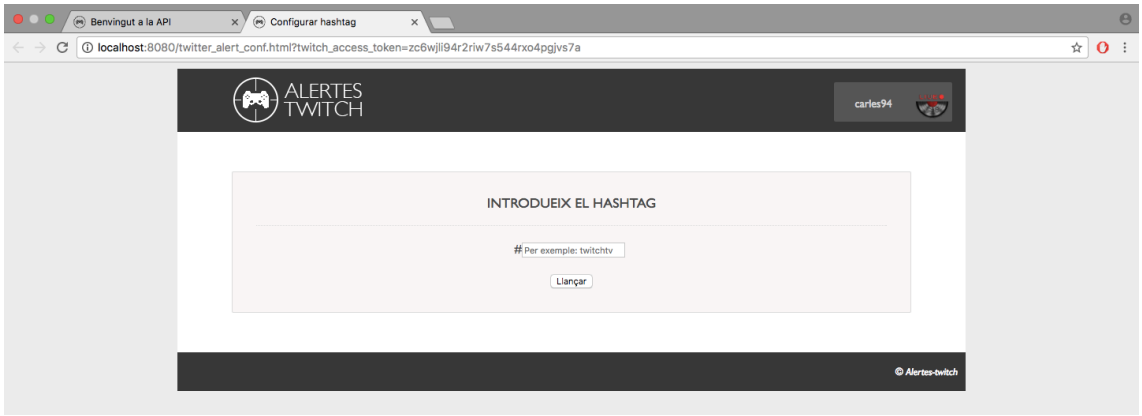

Figura 37. Plana web Twitter_conf.html

Una vegada enviat el formulari, es redirigirà a una pàgina web amb fons verd. Cada quinze minuts, si hi ha nous *tweets*, aquests es mostraran en forma de rètol, tal i com s'il·lustra en la figura 38.

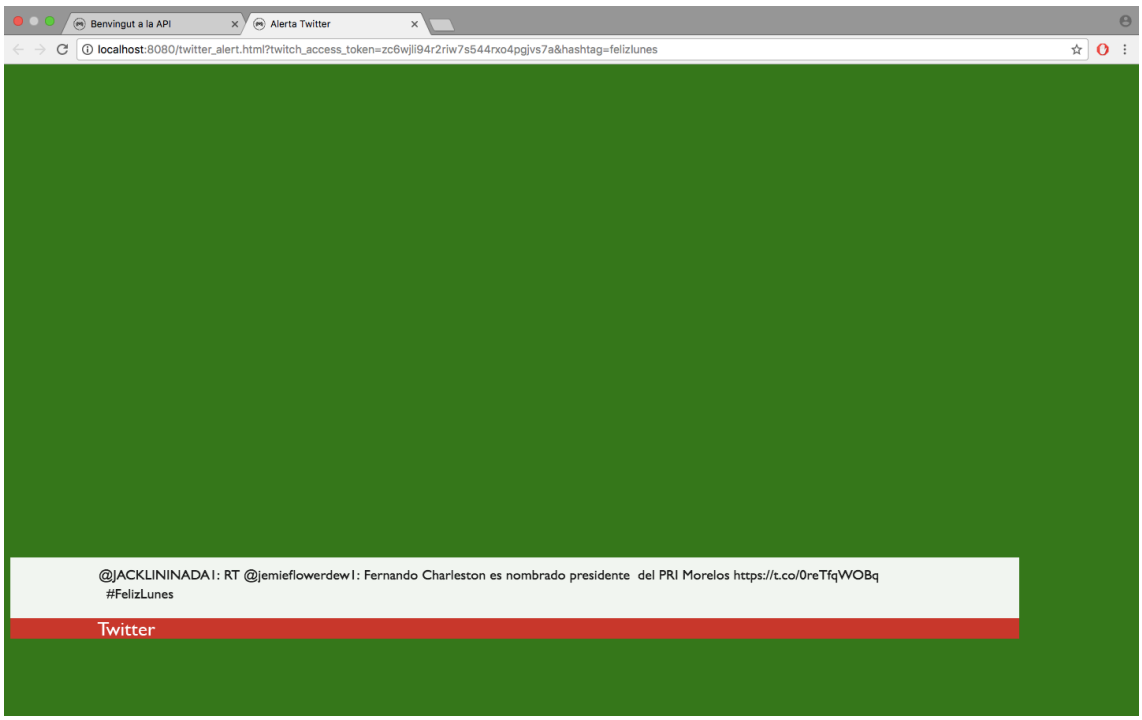

Figura 38. Plana web Twitter_alert.html

Per altra part, quan se seleccione en el menú l'alerta d'enquestes ("Alerta Strawpoll"), es redirigirà a un formulari on s'ha d'indicar l'enllaç de l'enquesta (de la plataforma Strawpoll) que es vol que es mostre, tal i com s'observa en la figura 39.

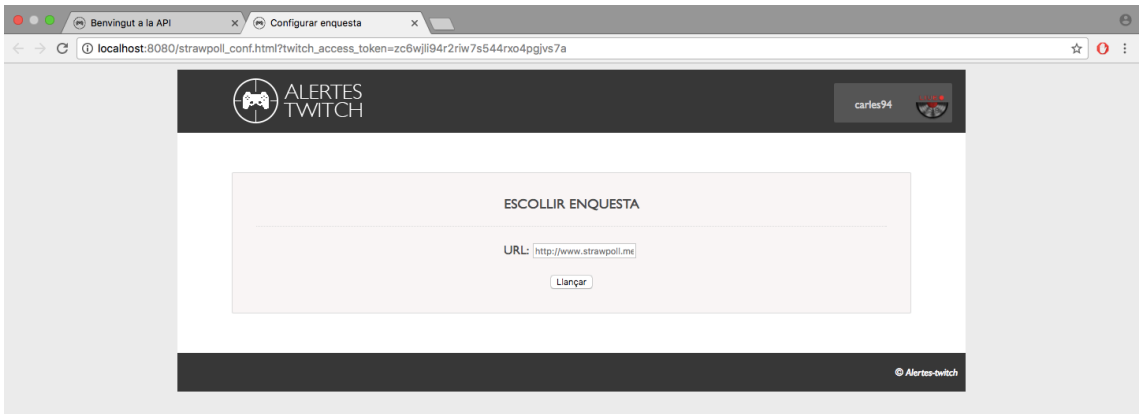

Figura 39. Plana web Strawpoll_conf.html

Una vegada enviat el formulari, es redirigirà a la plana web de l'alerta on apareixen els resultats de l'enquesta en temps real i en forma de rètol (figura 40).

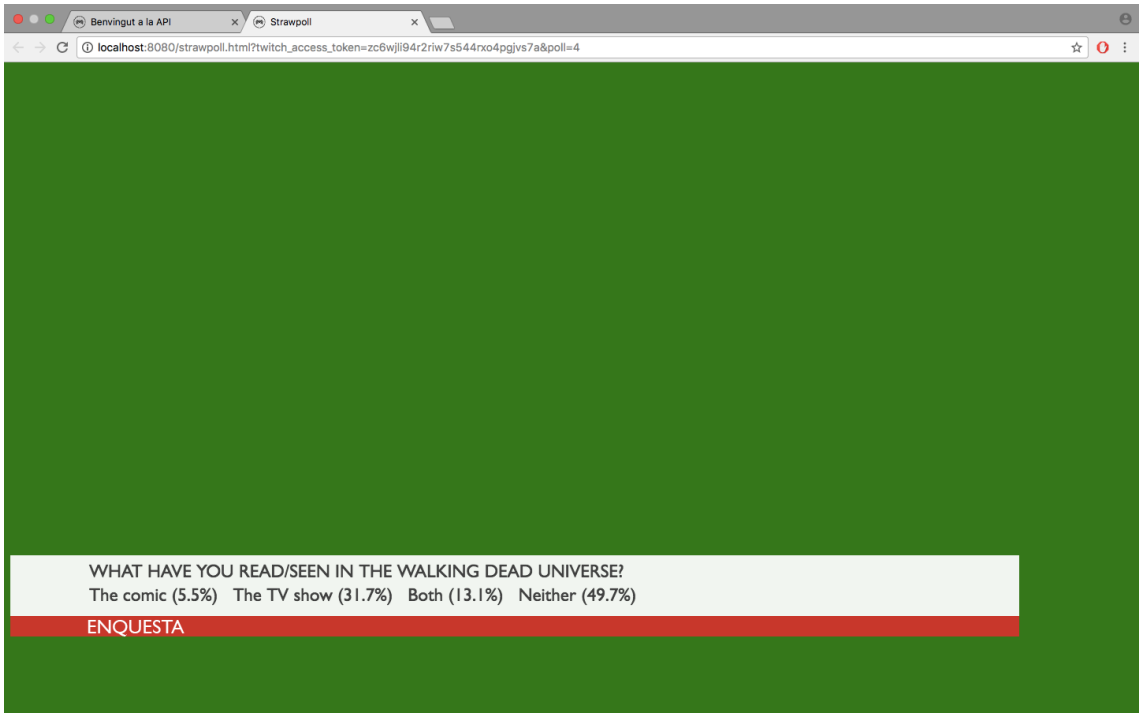

Figura 40. Pàgina web strawpol.html

Per últim, quan es trie l'alerta "llista seguidors" del menú de la secció d'usuaris, es redirigirà a una pàgina web amb fons verd on es mostren els últims set seguidors del canal de Twitch (figura 41).

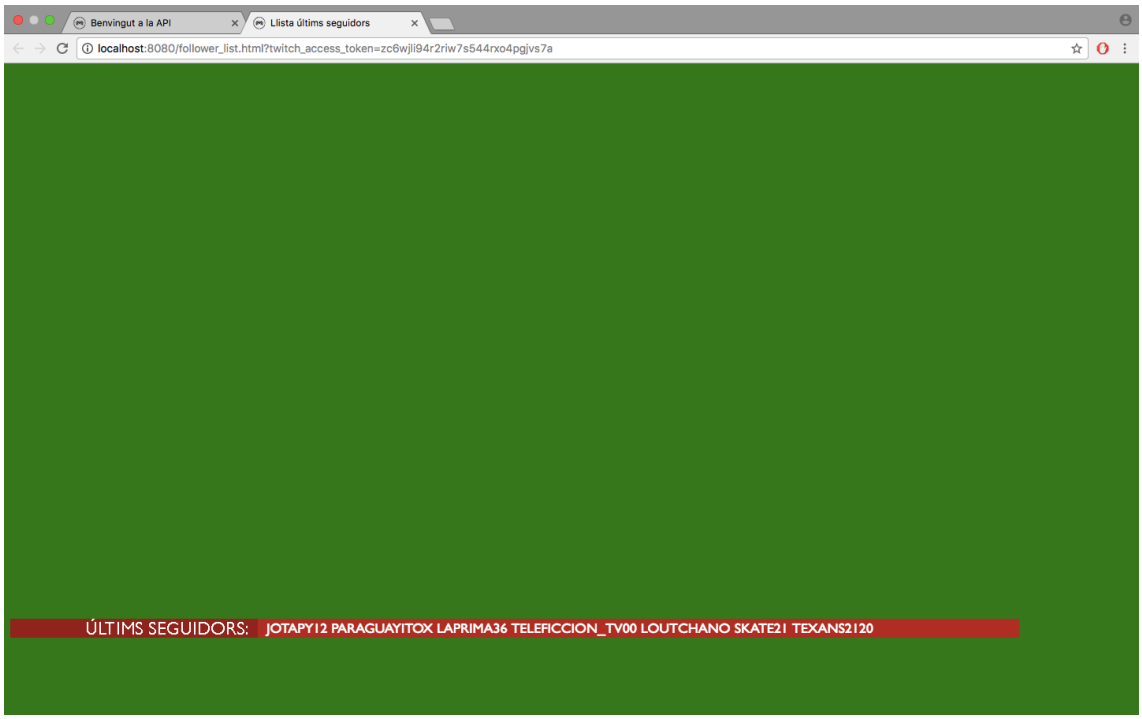

Figura 41. Pàgina web follower_list.html

Capítol 7. Conclusions

El principal objectiu d'aquest projecte, que era implementar una sistema que mostre missatges de xarxes socials a una emissió de vídeo en directe, s'ha acomplert totalment.

En concret, les funcionalitats que finalment s'han implementat són: avisar mitjançant rètols quan existeix un nou seguidor en Twitch, mostrar la llista dels últims seguidors, mostrar els *tweets* més recents que continguen un *hashtag* que determina l'usuari i mostrar en temps real els resultats d'una determinada enquesta de la plataforma Strawpoll que és elegida prèviament per l'usuari emissor.

En aquest projecte també s'ha creat una pàgina web senzilla i intuïtiva perquè l'usuari puga accedir fàcilment a cadascuna de les alertes i puga aprendre a usar-les (zona de publicacions).

Pel que fa als objectius formatius, cal destacar que també han sigut un èxit ja que s'han aprés en profunditat tecnologies de programació en la web, com ara Node.js, la nova versió de JavaScript (ES2015), CSS i HTML.

Per concloure, cal remarcar que el projecte ha complit tots els seus objectius ja que totes les implementacions que formen el sistema d'alertes han segut comprovades i han resultat ser un èxit.

7.1 Treball futur

Una ampliació que es podria dur a terme és implementar, mitjançant bases de dades en el servidor, una secció web on l'usuari puga editar el rètol canviant la imatge i modificant la posició del text (es pot observar que en la figura 34 del capítol de la implementació, s'han habilitat uns botons que més avant podrien servir per a aquest propòsit).

Pel que fa a les alertes, perquè l'aplicació web siga prou completa per a l'usuari emissor, faltaria implementar-se una altra que avise quan hi ha un nou subscriptor en el canal, no obstant això, no s'ha implementat perquè es necessita un compte de Twitch amb moltes visites diàries per poder provar si funciona o no aquesta (un compte de Twitch amb poques visites diàries, té les subscripcions desactivades).

A part de totes les alertes implementades i la que s'ha proposat en aquesta secció de la memòria, podrien implementar-se'n moltíssimes més, ja que hi ha més plataformes i xarxes socials (i les que es creen en un futur) que tenen una aplicació web de desenvolupadors que permet compartir dades amb la ferramenta que s'ha implementat en aquest projecte (sempre i quan es programen en un futur les peticions a aquesta), per poder posteriorment mostrar rètols amb informació actualitzada.

Bibliografia

[1] *Twitch.* [En línia] <https://www.twitch.tv> [Consulta: 13 de març de 2017]

[2] *JavaScript.* [En línia] <https://www.w3schools.com/js/default.asp> [Consulta: 28 de març de 2017]

[3] *HTML.* [En línia] <https://www.w3schools.com/html/default.asp> [Consulta: 28 de març de 2017]

[4] *Strawpoll.* [En línia] <http://www.strawpoll.me> [Consulta: 5 de juny de 2017]

[5] *Node.js.* [En línia] <https://nodejs.org> [Consulta: 1 de maig de 2017]

[6] *CSS.* [En línia] <https://www.w3schools.com/css/default.asp> [Consulta: 6 de juny de 2017]

[7] *Postman.* [En línia] <https://www.getpostman.com> [Consulta: 14 de març de 2017]

[8] *Open Broadcaster Software.* [En línia] <https://obsproject.com> [Consulta: 13 de març de 2017]

[9] *Sublime Text.* [En línia] <https://www.sublimetext.com> [Consulta: 26 de març de 2017]

[10] *AJAX.* [En línia] <http://api.jquery.com/jquery.ajax> [Consulta: 3 de maig de 2017]

[11] *Twitch: Documentació de l'aplicació web.* [En línia] <https://dev.twitch.tv/docs> [Consulta: 19 de març de 2017]

[12] *Twitch: Aplicació web.* [En línia] <https://api.twitch.tv> [Consulta: 19 de maig de 2017]

[13] *Twitter: Documentació de l'aplicació web.* [En línia] <https://dev.twitter.com> [Consulta: 20 de maig de 2017]

[14] *Twitter: Aplicació web (obtenir tweets).* [En línia] <https://api.twitter.com/1.1/search/tweets.json> [Consulta: 20 de maig de 2017]

[15] *Strawpoll: Documentació de l'aplicació web.* [En línia] <https://strawpoll.zendesk.com/hc/en-us/articles/218979828-Straw-Poll-API-Information> [Consulta: 6 de juny de 2017]

Bibliografia utilitzada per resoldre problemes d'implementació

CODE ACADEMY: Intro to JSON & API Forum (Twitter). [En línia] <https://www.codecademy.com/en/forum_questions/51d4a5a97c82ca640d002714> [Consulta: 23 de maig de 2017]

STACKOVERFLOW: Cross domain jQuery GET. [En línia] <https://stackoverflow.com/questions/15369577/cross-domain-jquery-get> [Consulta: 25 de maig de 2017]

STACKOVERFLOW: JQuery AJAX submit form. [En línia] <https://stackoverflow.com/questions/1960240/jquery-ajax-submit-form> [Consulta: 4 de maig de 2017]

STACKOVERFLOW: Pass request headers in a jQuery AJAX GET call. [En línia] <https://stackoverflow.com/questions/3258645/pass-request-headers-in-a-jquery-ajax-get-call> [Consulta: 25 de maig de 2017]

EXPRESSJS: Guide (routing). [En línia] <http://expressjs.com/en/guide/routing.html> [Consulta: 7 de maig de 2017]

EXPRESSJS: Serving static files in Express. [En línia] <https://expressjs.com/en/starter/static-files.html> [Consulta: 7 de maig de 2017]