|  |
| --- |
| **Universitatea din CraiovaFacultatea de Automatică, Calculatoare și ElectronicăDepartamentul de Calculatoare și Tehnologia Informației** |
| **Monitorizare video folosind Raspberry Pi si un smartphone** |
| Nicolae Nicusor Bogdan |
|  |
|  |
| Coordonator S.l.dr.ing. Nicolae Enescu |

|  |
| --- |
| **Anul universitar 2016-2017** |

Cuprins

[Raspberry Pi 2](#_Toc483763000)

[Fundatia Raspberry Pi 4](#_Toc483763001)

[Specificatii Raspberry Pi 3 4](#_Toc483763002)

[WebRTC 7](#_Toc483763003)

[Functionalitate 8](#_Toc483763004)

[getUserMedia 8](#_Toc483763005)

[RTCPeerConnection 8](#_Toc483763006)

[RTCDataChannel 9](#_Toc483763007)

[SDP: Session Description Protocol 9](#_Toc483763008)

[STUN 10](#_Toc483763009)

[TURN 10](#_Toc483763010)

[Securitate bazata pe browser 11](#_Toc483763011)

[WebSockets 13](#_Toc483763012)

[Node.js 15](#_Toc483763013)

[Programare bazate pe evenimente 16](#_Toc483763014)

[(Engine) V8 de la Google Chrome 16](#_Toc483763015)

[Node packeage manager (npm) 17](#_Toc483763016)

[Socket.io 17](#_Toc483763017)

[Anexe 18](#_Toc483763018)

[Bibliografie 19](#_Toc483763019)

# Raspberry Pi

Mai multe generații de Raspberry Pi au fost lansate. Prima generație (Raspberry Pi 1 Model B) a fost lansată în februarie 2012. A fost urmată de un model A care era simplificat și mai ieftin. În 2014, fundația a lansat o placă cu un design îmbunătățit numita Raspberry Pi 1 Model B +. Aceste mini calculatoare sunt de aproximativ dimensiunea cărții de credit și reprezintă factorul standard de bază al modelului. Modelele îmbunătățite A + și B + au fost lansate un an mai târziu. Un "modul de calcul" a fost lansat în aprilie 2014 pentru aplicațiile încorporate, iar în noiembrie 2015 a fost lansat un nou model Raspberry Pi Zero cu dimensiuni mai mici și capabilități de intrare / ieșire reduse (I / O) ce costa $5. Raspberry Pi 2, care a adăugat mai mult RAM, a fost lansat în februarie 2015. Raspberry Pi 3 Model B lansat în februarie 2016, este asociat cu WiFi, Bluetooth și USB. Începând din ianuarie 2017, Raspberry Pi 3 Model B este cea mai nouă linie principală Raspberry Pi. Raspberry Pi se are preturi între 5 și 35 USD. Începând cu 28 februarie 2017, a fost lansat Raspberry Pi Zero W, care este identic cu Raspberry Pi Zero, dar are funcționalitatea Wi-Fi și Bluetooth a Raspberry Pi 3 tot cu un pret 10 USD

Toate modelele au un sistem Broadcom pe un chip (SoC), care include o unitate centrală de procesare compatibilă ARM (CPU) și o unitate de procesare grafică on-chip (GPU, VideoCore IV). Viteza procesorului variază de la 700 MHz la 1,2 GHz pentru modelul Pi 3 și memoria plăcii de la 256 MB la 1 GB RAM. Cardurile Secure Digital (SD) sunt utilizate pentru a stoca sistemul de operare și memoria programată în dimensiunile SDHC sau MicroSDHC. Cele mai multe placi au între unul și patru sloturi USB, ieșire video HDMI și compozit și o mufă de telefon de 3,5 mm pentru audio. Nivelul de ieșire inferior este furnizat de un număr de pinii GPIO care susțin protocoale comune precum I²C. Modelele B au un port Ethernet 8P8C, iar Pi 3 și Pi Zero W au la bord Wi-Fi 802.11n și Bluetooth.

Fundația furnizează Raspbian, o distribuție Linux bazată pe Debian pentru descărcare, precum și Ubuntu , Windows 10 IOT Core. Promovează Python și Scratch ca limbă principală de programare, cu suport pentru multe alte limbi. [15] Firmware-ul implicit este o sursă închisă, în timp ce o sursă deschisă neoficială este disponibilă [citare necesară]

## Fundatia Raspberry Pi

Fundația Raspberry Pi a fost înființată în 2008 ca o organizație de caritate cu sediul în Regatul Unit, cu scopul "de a continua promovarea educației adulților și a copiilor, în special în domeniul calculatoarelor, informaticii și informaticii si subiectelor anexe ".

Prin intermediul subsidiara de tranzacționare, (Raspberry Pi Trading Limited), inventeaza și vinde computerele low-cost, de înaltă performanță pe care oamenii le folosesc pentru a învăța, a rezolva probleme și a se distra. Între lansarea primului produs în Februarie 2012 și cea de-a patra aniversare din februarie 2016, a vândut peste opt milioane de computere și a ajutat la înființarea unei comunități globale de producători și educatori digitali.

În octombrie 2015, Fundația a fuzionat cu Code Club, o rețea de cluburi de codare post-școală conduse de voluntariat pentru copii de 9-11 ani.În noiembrie 2015 a lansat primul calculator din lume de 5 dolari, Raspberry Pi Zero.

Fundatia foloseste profiturile generate de activitățile comerciale pentru a atinge obiectivele educaționale.Ei primesc, de asemenea finanțarea și sprijinul în material oferit de parteneri generali și donatori care împărtășesc misiunea ei.

## Specificatii Raspberry Pi 3

Raspberry Pi 3 este a treia generație de Raspberry Pi. Acesta a înlocuit modelul Raspberry Pi 2 Model B în februarie 2016. În comparație cu Raspberry Pi 2 are:A 1.2GHz 64-bit quad-core ARMv8 CPU

* 802.11n Wireless LAN
* Bluetooth 4.1
* Bluetooth Low Energy (BLE)

Like the Pi 2, it also has:

* 4 USB ports
* 40 GPIO pins
* Full HDMI port
* Ethernet port
* Combined 3.5mm audio jack and composite video
* Camera interface (CSI)
* Display interface (DSI)
* Micro SD card slot (now push-pull rather than push-push)
* VideoCore IV 3D graphics core
* Raspberry Pi 3 are un factor de formă identic cu Pi 2 precedent (și Pi 1 Model B +) și are o compatibilitate completă cu Raspberry Pi 1 și 2.

# WebRTC

WebRTC (Web Real-Time Communications) este o tehnologie care permite aplicațiilor web și site-urilor să capteze și să transmită în mod fie medii audio sau video, precum și să facă schimb de date arbitrare între browsere fără a solicita un intermediar. Setul de standarde care compune WebRTC face posibilă partajarea datelor și efectuarea teleconferințe directe intre utilizatori, fără a cere utilizatorului să instaleze plug-in-uri sau orice alt software terță parte. Acesta tehnologie este integrata din ce in ce mai multe browser web prin care enumeram Google Chrome (varianta desktop dar si pentru telefoane mobile), Mozilla Firefox, Opera,Microsoft Edge (partial) desi in momentul de fata nu este in Apple Safari , s-a anuntat ca se lucreaza la implementarea .

WebRTC este alcătuit din mai multe interfețe API intercorelate și protocoale care lucrează împreună pentru a realiza acest lucru. Documentația pe care o veți găsi aici vă va ajuta să înțelegeți fundamentele WebRTC, cum să configurați și să utilizați conexiuni de date și media și multe altele.

WebRTC servește mai multor scopuri și se suprapune substanțial cu aplicațiile Media Capture și Streams API. Împreună, oferă capabilități multimedia puternice pe Web, inclusiv suport pentru conferințe audio și video, schimb de fișiere, gestionarea identității și interfață cu sistemele telefonice vechi, prin trimiterea semnalelor DTMF. Conexiunile între utilizatori pot fi făcute fără a fi nevoie de drivere sau plug-in-uri speciale și pot fi adesea făcute fără servere intermediare.

Conexiunile dintre doi utilizatori sunt create utilizând interfața RTCPeerConnection. Odată ce o conexiune a fost stabilită și deschisă, fluxurile media (MediaStreams) și / sau canalele de date (RTCDataChannels) pot fi adăugate la conexiune.

Fluxurile media pot consta din orice număr de piste de informații media; Care sunt reprezentate de obiecte bazate pe interfața MediaStreamTrack, pot conține unul dintre mai multe tipuri de date media, inclusiv audio, video și text (cum ar fi subtitrarea sau chiar numele de capitole). Cele mai multe fluxuri constau din cel puțin o piesă audio și probabil o piesă video și pot fi utilizate pentru a trimite și a primi atât materiale live, cât și informații media stocate (cum ar fi un film redat în flux).

De asemenea, puteți utiliza conexiunea dintre doi colegi pentru a schimba date binare arbitrare utilizând interfața RTCDataChannel. Acest lucru poate fi folosit pentru informații despre canalul din spate, schimb de metadate, pachete de stare a jocului, transferuri de fișiere sau chiar ca un canal principal pentru transferul de date.

## Functionalitate

WebRTC se bazează in cea mai pare parte pe trei API-uri, fiecare realizând o funcție specifică pentru a permite comunicarea în timp real într-o aplicație web. Aceste API vor fi numite și explicate pe scurt. Implementarea și detaliile tehnice ale fiecărui protocol și tehnologie sunt în afara domeniului de aplicare al acestei lucrari.

## getUserMedia

De mulți ani a fost necesar să se bazeze pe pluginuri de browser terță parte, cum ar fi Flash sau Silverlight, pentru a capta audio sau video de pe un computer. Cu toate acestea,in noua era HTML 5 a inaugurat accesul hardware direct la numeroase dispozitive și oferă JavaScript API-uri care interfață cu capabilitățile hardware de bază ale sistemului

GetUserMedia este un astfel de API, care permite unui browser să acceseze camera și microfonul utilizatorului. Deși este utilizat de WebRTC, acest API este oferit ca parte a HTML 5.

### RTCPeerConnection

RTCPeerConnection este primul dintre cele două API-uri care sunt oferite în mod specific ca parte a specificației WebRTC. O interfață RTCPeerConnection reprezintă conexiunea reală WebRTC și se bazează pe gestionarea fluxului eficient de date între doi colegi.

Atunci când un apelant dorește să inițieze o conexiune cu o parte la distanță, browserul pornește de la instanțierea unui obiect RTCPeerConnection. Aceasta include o descriere SDP generată de sine pentru a fi schimbată cu colegii lor. Destinatarul, la rândul său, răspunde cu propria descriere SDP. Descrierile SDP sunt utilizate ca parte a fluxului de lucru complet ICE pentru traversal NAT.

Odată cu stabilirea conexiunii, RTCPeerConnection permite trimiterea de date audio și video în timp real ca un bitstream între browsere.

În cele din urmă, RTCPeerConnection API este responsabil pentru gestionarea întregului ciclu de viață al fiecărei conexiuni de tip peer-to-peer și încapsulează toate configurarea conexiunilor, gestionarea și starea într-o singură interfață ușor de utilizat.

RTCPeerConnection are două trăsături specifice: - Comunicare directă peer-to-peer între două browsere - utilizand UDP / IP - nu există nici o garanție pentru sosirea pachetelor (ca în TCP / IP). - (Permițând pierderea unor date, ne putem concentra pe oferirea unei comunicări în timp real.)

### RTCDataChannel

RTCDataChannel este al doilea API principal oferit ca parte a WebRTC și reprezintă canalul principal de comunicare prin care schimbul de date despre aplicații arbitrare are loc între colegii. Cu alte cuvinte, este folosit pentru a transfera date direct de la un partener la altul.

Deși există o serie de opțiuni alternative pentru canalele de comunicații (de exemplu, WebSocket, Evenimente expediate pe server), aceste alternative au fost concepute pentru a comunica cu un server mai degrabă decât cu un peer conectat direct. RTCDataChannel seamănă cu popularul WebSocket, dar în schimb are un format peer-to-peer în timp ce oferă proprietăți personalizabile de livrare a transportului de bază.

Cele trei API-uri principale sunt aspectele de interes pentru dezvoltatori de aplicatii WebRTC, dar există o serie de tehnologii ce stau la baza acestora si care sunt utilizate pentru a furniza aceste protocoale (API-urile RTCPeerConnection și RTCDataChannel).

ICE, STUN și TURN sunt necesare pentru a stabili și a menține o conexiune peer-to-peer peste UDP. DTLS este folosit pentru a asigura toate transferurile de date între utilizatori, deoarece criptarea este o caracteristică obligatorie a WebRTC. În cele din urmă, SCTP și SRTP sunt protocoalele de aplicație utilizate pentru a multiplexa diferitele fluxuri, pentru a asigura congestionarea și controlul fluxului și pentru a oferi o livrare parțial sigură și alte servicii suplimentare pe lângă UDP.

## SDP: Session Description Protocol

Session Description Protocol (SDP) este un protocol descriptiv care este utilizat ca metodă standard de anunțare și gestionare a invitațiilor la sesiune, precum și de efectuarea altor sarcini de inițiere pentru sesiunile multimedia. SDP reprezintă capabilitățile și preferințele browserului într-un format text și pot include următoarele informații:

* Capabilități media (video, audio) și codecuri utilizate;
* Adresă IP și număr port;
* Protocol P2P de transmisie de date (WebRTC utilizează SecureRTP);
* Lățimea de bandă utilizabilă pentru comunicare –
* Atributele sesiunii (nume, identificator, timp activ, etc.) –
* Cu toate acestea, acestea nu sunt utilizate în WebRTC
* Alte metadate conexe ...

În ziua de zi, SDP este utilizat pe scară largă în contextul protocolului de inițiere a sesiunii (SIP), al Protocolului de transport în timp real (RTP) și al Protocolului de Streaming în timp real (RSP).In continuare se prezinta cateva utilitare ce au rolul de a descoperi adreasa IP a utilizatorilor implicatii in cazul in care se utilizeza NAT dar si de a initia conexiunea .

## STUN

Pentru a realiza comunicarea P2P, ambele părți necesită cel puțin cunoașterea adresei IP a utilizatorilor ce se conecteaza și a portului UDP atribuit. Ca urmare, este necesar un anumit volum de schimb de informații înainte ca comunicarea WebRTC să poată fi stabilită.

Un server STUN este utilizat de fiecare utilizator pentru a determina adresa lor IP publică și este referit de cadrul ICE în timpul stabilirii conexiunii. Serverele STUN sunt de obicei accesibile publicului și pot fi utilizate în mod liber de aplicațiile WebRTC. De exemplul Google pune la dispozitia publicului larg http://stun.l.google.com:19302.

## TURN

În eventualitatea în care stabilirea unei comunicații P2P nu reușește, o opțiune de rezervă poate fi furnizată prin intermediul unui server TURN. Prin relocarea traficului între colegi, comunicarea WebRTC poate fi asigurată, dar poate suferi degradări în ceea ce privește calitatea și latența media.

Serverele TURN pot asigura un succes mare în stabilirea apelurilor, indiferent de mediile utilizatorilor finali. Pe măsură ce datele sunt trimise printr-un server intermediar, se consumă și lățimea de bandă a serverului. În cazul în care multe apeluri sunt direcționate simultan prin server, lățimea de bandă a devenit de asemenea considerabilă.

În mod normal, serverul nu este accesibil în mod liber și trebuie furnizat (sau închiriat) de către furnizorul de aplicații. Exemple de astfel de server pot fi gasite in oferta Google Cloud Computing, Microsoft Azure dar si Amazon S3 dar fi create in functie de necesitatii.

## Securitate bazata pe browser

Există o serie de modalități prin care o aplicație de comunicație în timp real poate impune riscuri de securitate, atât pentru transportator, cât și pentru utilizatorii finali. Astfel de riscuri de securitate pot fi aplicabile oricărei aplicații care se ocupă cu transmiterea de date și mijloace de comunicare în timp real.

WebRTC diferă de alte aplicații RTC, oferind o infrastructură puternică și fiabilă pentru dezvoltatorii chiar incepatori, care să le utilizeze fără a compromite securitatea. Acum vom continua să discutăm despre modul în care WebRTC se ocupă de fiecare dintre aceste riscuri.

Arhitectura WebRTC presupune din perspectiva securității că resursele de rețea există într-o ierarhie de încredere. Din perspectiva utilizatorului, browserul (sau clientul client) este baza tuturor securității WebRTC și acționează ca baza de calcul de încredere (TCB).

Activitatea browserului este să permită accesul la internet, oferind utilizatorilor o protecție adecvată a securității. Cerințele de securitate ale WebRTC sunt construite direct pe baza acestei cerințe; Browserul este portalul prin care utilizatorul accesează toate aplicațiile și conținutul WebRTC.

Cu toate că HTML și JS furnizate de server pot determina browserul să execute o varietate de acțiuni, browserul segregă acele script-uri în sandbox-uri. Sandbox menționate permit izolarea scripturilor între ele și de la computerul utilizatorului. În general, scripturile au permisiunea de a interacționa numai cu resurse din același domeniu - sau, mai exact, cu aceeași "origine".

Browserul aplică toate politicile de securitate pe care utilizatorul dorește și este primul pas în verificarea tuturor părților terțe. Toate entitățile autentificate își verifică identitatea prin browser.

Dacă utilizatorul alege un browser adecvat despre care știe că poate avea încredere, atunci toată comunicarea WebRTC poate fi considerată "sigură" și să urmeze arhitectura de securitate acceptată standard a tehnologiei WebRTC. Cu toate acestea, în cazul în care există vreo îndoială că un browser este "de încredere" (de exemplu, dacă a fost descărcat de la un terț, mai degrabă decât dintr-o locație de încredere), atunci toate interacțiunile cu aplicațiile WebRTC sunt afectate și nu pot fi sigure.

Cu alte cuvinte, nivelul de încredere acordat utilizatorului de către WebRTC este direct influențat de încrederea utilizatorului în browser. Ele sunt in continua dezvoltare rapidă datorită frecvenței și gamei de riscuri la care sunt expuși utilizatorii, precum și caracterul lor omniprezent (și importanța informațiilor accesate prin browser). Dat fiind că componentele WebRTC sunt oferite ca parte a unui browser, ele sunt, de asemenea, actualizate ori de câte ori browserul este actualizat. Dacă o vulnerabilitate viitoare ar putea fi găsită în implementarea WebRTC a browserului, o remediere va fi probabil livrată rapid. Acest lucru se dovedește a fi adevărat în ciclurile de dezvoltare rapidă ale Chrome și Firefox. De fapt, în epoca actualizărilor automate, componentele WebRTC pot fi actualizate printr-o nouă versiune de browser imediat ce patch-ul este disponibil pe servere. Cele mai multe browsere moderne au un record bun de auto-actualizare în termen de 24 de ore de la descoperirea unei vulnerabilități grave sau amenințare.

În epoca modernă a telefoanelor inteligente și a dispozitivelor mobile, oamenii comunică mai mult decât oricând și chiar în moduri mai personale decât am știut anterior. În special, criptarea a devenit un subiect important în ultimii ani, ca urmare a creșterii nivelului de conștientizare a scandalurilor majore de hacking corporative și a interceptărilor de telecomunicații pe scară largă. Rezultatul a fost o creștere rapidă a neîncrederii utilizatorilor față de astfel de organizații și solicită armelor să pună în aplicare măsuri de securitate mult îmbunătățite. Toți utilizatorii finali doresc să știe că datele lor personale sunt păstrate sub control.

WebRTC are un mare avantaj față de majoritatea serviciilor VoIP din zona de securitate. Până în prezent, majoritatea serviciilor au tratat de obicei securitatea ca opțională, ceea ce înseamnă că majoritatea utilizatorilor finali utilizează apeluri VoIP fără criptare. Corporațiile mari, în special, sunt un vinovat pentru acest lucru, alegând să economisească bani pe implementări mai ieftine, decât să ia în considerare în mod corespunzător utilizatorii sau valoarea datelor pe care le manipulează. Dar, deoarece WebRTC interzice comunicarea necriptată, utilizatorii pot fi siguri că datele lor rămân în siguranță și private.

Având în vedere faptul că WebRTC a fost conceput în funcție de securitate, acesta impune sau încurajează concepte de securitate importante în toate domeniile principale. Ca atare, pe lângă faptul că este pur și simplu construit în siguranță, acesta încurajează dezvoltatorii WebRTC să-și ia și serios securitatea.

Ca urmare a acestui accent puternic pe comunicarea securizată, WebRTC este în prezent considerat de unii ca fiind una dintre cele mai sigure soluții VoIP aflate acolo. Premisa principală de a avea în mod implicit criptare este că un apel este privat în permanență. Securitatea și criptarea nu mai sunt considerate ca fiind opționale. Și pentru a indulci totul, WebRTC este disponibil gratuit pentru toată lumea, oferind un cadru tentant și fiabil dezvoltatorilor pentru a-și construi următoarea aplicație.

În viitorul apropiat ne putem aștepta să vedem din ce în ce mai multe servicii de comunicații care oferă utilizatorilor lor o securitate mult mai mare. Dar pentru moment, WebRTC este unul dintre cei care conduc acuzatia.

# WebSockets

WebSocket este un protocol de comunicații de calculatoare, oferind canale de comunicare full duplex printr-o singură conexiune TCP. Protocolul WebSocket a fost standardizat de IETF ca RFC 6455 în 2011, iar API-ul WebSocket în Web IDL este standardizat de W3C.

WebSocket este proiectat pentru a fi implementat în browsere web și servere web, dar poate fi utilizat de orice aplicație client sau server. Protocolul WebSocket este un protocol TCP independent. Singura ei relație cu HTTP este că strângerea de mana este interpretată de serverele HTTP ca o cerere de upgrade. [1] Protocolul WebSocket permite interacțiunea dintre un browser și un server web cu cheltuieli generale inferioare, facilitând transferul de date în timp real de la și către server. Acest lucru este posibil prin furnizarea unui mod standardizat serverului de a trimite conținut către browser fără a fi solicitat de către client și care să permită transmiterea mesajelor înainte și înapoi, menținând conexiunea deschisă. În acest fel, o conversație în două direcții (bidirecțională) poate avea loc între un browser și server. Comunicațiile se efectuează prin numărul de port TCP 80 (sau 443 în cazul conexiunilor criptate TLS), ceea ce este benefic pentru acele medii care blochează conexiunile de Internet fără internet folosind un firewall. O comunicație dublă față de browser-server a fost realizată în moduri ne-standardizate folosind tehnologii stopgap cum ar fi Comet.

Protocolul WebSocket este în prezent acceptat în majoritatea browserelor importante, inclusiv Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari și Opera. WebSocket necesită, de asemenea, aplicații web pe server pentru al susține.

Spre deosebire de HTTP, WebSocket oferă comunicații full-duplex. În plus, WebSocket permite fluxuri de mesaje în partea de sus a TCP. TCP singur se ocupă cu fluxuri de octeți fără niciun concept inerent de mesaj. Înainte de WebSocket, comunicarea cu portul 80 complet duplex a fost realizată folosind canalele Comet; Cu toate acestea, implementarea Comet este netriviala, iar datorita handshake-ului TCP si a antetului HTTP, este ineficient pentru mesajele mici. Protocolul WebSocket vizează rezolvarea acestor probleme fără a compromite ipotezele de securitate ale web-ului.

Specificația protocolului WebSocket definește ws și wss drept două noi scheme de identificare a resurselor (URI) , care sunt utilizate pentru conexiuni necriptate și criptate, respectiv. În afară de numele schemei și fragmentul (# nu este acceptat), restul componentelor URI sunt definite pentru a utiliza sintaxa generică URI.

# Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Node.js a fost scris inițial de Ryan Dahl în 2009, aproximativ zece treisprezece ani de la introducerea primului mediu de mediu al serverului, LiveSearch Web Netscape. Versiunea inițială a acceptat numai Linux și Mac OSX. Dezvoltarea și întreținerea acestuia au fost conduse de Dahl și ulterior sponsorizate de Joyent .

Dahl a fost inspirat să creeze Node.js după ce a văzut o bară de progres pentru încărcarea fișierelor pe Flickr. Browserul nu știa cât de mult a fost încărcat fișierul și a trebuit să interogheze serverul Web. Dahl a dorit o cale mai ușoară

Dahl a criticat posibilitățile limitate ale celui mai popular server web din 2009, Apache HTTP Server, de a gestiona o mulțime de conexiuni concurente (până la 10.000 și mai mult) și cel mai comun mod de a crea coduri (programare secvențială), atunci când codul a blocat Întregul proces sau stive implicite de execuție multiplă în cazul conexiunilor simultane.

Dahl a demonstrat proiectul la inaugurarea European JSConf pe 8 noiembrie 2009. Node.js a combinat motorul JavaScript V8 de la Google, o buclă de evenimente și un API I/O de nivel scăzut. Proiectul a primit ovații în picioare la eveniment.

În ianuarie 2010, a fost introdus un manager de pachete pentru mediul Node.js numit npm. Managerul de pachete facilitează programatorilor să publice și să partajeze codul sursă al bibliotecilor Node.js și este conceput pentru a simplifica instalarea, actualizarea și dezinstalarea bibliotecilor.

În iunie 2011, Microsoft și Joyent au implementat o versiune nativă Windows a Node.js. Primul Node.js care suportă Windows a fost lansat în iulie 2011.

În ianuarie 2012, Dahl a renunțat, promovând colaboratorul și creatorul lui Isaac Schlueter pentru a gestiona proiectul. În ianuarie 2014, Schlueter a anunțat că Timothy J. Fontaine va conduce proiectul .

În februarie 2015, intenția de a forma o fundație neutră Node.js a fost anunțată.

Aceasta a adus caracteristicile V8 ES6 în Node.js și un ciclu de eliberare pe termen lung.

## Programare bazate pe evenimente

Acest stil de programare - prin care, în loc să utilizați o valoare de retur, defneți funcțiile .Sunt numite de sistem atunci când apar evenimente interesante - se numește programare bazate pe eveniment sau programare asincron.

Aceasta este una dintre caracteristicile definitorii ale node.js. Acest stil de programare înseamnă ca procesul curent nu va bloca atunci când face I/O la un fisier sau baza de date. Prin urmare, pot apărea mai multe operații de I / O

Paralel și fiecare funcție inversă de apel va fi invocată atunci când operațiunea se potrivește. Stilul de programare condus de eveniment este însoțit de o buclă de evenimente. O buclă de evenimente se construiește în principal două funcții într-o buclă continuă - detectarea evenimentului și evenimentul

Handler declanșator. În orice derulare a buclă, trebuie să detecteze evenimentele care s-au întâmplat. Apoi când se întâmplă un eveniment, bucla evenimentului trebuie să determine apelul de functie al evenimentului și să îl invocă.

## (Engine) V8 de la Google Chrome

V8 este motorul de excutie JavaScript Google de înaltă performanță. Este scris în C ++ și este folosit în Google Chrome, browserul Google open source si node.js dar si alte proiecte.

V8 implementează ECMAScript așa cum este specificat în ECMA-262, ediția a 5-a și rulează pe Windows (XP sau mai nou), Mac OS X (10.5 sau mai nou) și sisteme Linux care utilizează procesoare IA-32, x64 sau ARM.

V8 compilează și execută codul sursă JavaScript, gestionează alocarea memoriei pentru obiecte, iar gunoi colectează obiectele pe care nu mai are nevoie. Întrerupătorul V8, generatorul și colectorul de gunoi, este unul dintre cheile performanței V8. Puteți afla despre acest aspect și alte aspecte legate de performanță în Elementele de design V8.

JavaScript este folosit cel mai frecvent pentru script-urile de pe partea clientului într-un browser, fiind folosit, de exemplu, pentru a manipula obiecte Document Object Model (DOM). Totuși, DOM nu este furnizat de motorul JavaScript, ci de un browser. Același lucru este valabil și pentru V8 - Google Chrome furnizează DOM. Cu toate acestea, V8 furnizează toate tipurile de date, operatorii, obiectele și funcțiile specificate în standardul ECMA.

V8 permite oricărei aplicații C ++ să-și expună propriile obiecte și funcții la codul JavaScript. Depinde de fiecare proiect să decida obiectele și funcțiile pe care doreste să le expuneți la JavaScript. Există multe exemple de aplicații care fac acest lucru, de exemplu: Adobe Flash și widgeturile tabloului de bord în Mac OS X de la Apple și Yahoo Widgets.

## Node packeage manager (npm)

Odată ce depindeți de acest cod de la alți dezvoltatori, npm permite foarte ușor să verificați dacă au făcut actualizări și să descărcați aceste actualizări atunci când sunt făcute.

Acești biți de cod reutilizabil sunt numiți pachete sau, uneori, module. Un pachet este doar un director cu unul sau mai multe fișiere în el, care are de asemenea un fișier numit "package.json" cu unele metadate despre acest pachet. O aplicație tipică, cum ar fi un site web, va depinde de zeci sau sute de pachete. Aceste pachete sunt adesea mici. Ideea generală este aceea că creați un mic bloc de clădiri care rezolvă o problemă și o rezolvă bine.

Se pot găsi pachete care să ajute să construiți aplicația prin navigarea pe site-ul Web npm. Când navigați pe site,se găsi diferite tipuri de pachete. Veți găsi o mulțime de module de noduri. Npm a început ca manager de pachete de nodo.js , astfel încât veți găsi o mulțime de module care pot fi utilizate pe partea de server. Există, de asemenea, o mulțime de pachete care adaugă comenzi pentru a le utiliza în linia de comandă. Și în acest moment puteți găsi un număr de pachete care pot fi utilizate în browser in pentru parte de client (browser).

# Socket.io

Socket.IO este un packet de librarie de tipul node.js ce se foloseste standardul WebSockets pentru a realiza o legatura din client si server de comunicarea bidirecțională pe bază de evenimente în timp real.Se compune din:

* Un server Node.js
* O librarie de clienți Javascript pentru browser (sau un client Node.js)

Sunt disponibile și câteva implementări în alte limbi: Java , C ++,Swift.

Principalele sale caracteristici sunt:

1. Rapiditarea deoarece transmisiunile sunt tcp in binar sunt foarte rapide
2. Fiabilitate
   1. Conexiunile se stabilesc chiar și în prezența:
      1. Proxy-uri și balansoare de sarcină.
      2. Firewall personal și software antivirus.
   2. În acest scop, se bazează pe Engine.IO, care stabilește mai întâi o conexiune de polling lung, apoi încearcă să facă upgrade la transporturi mai bune, care sunt "testate" pe partea laterală, cum ar fi WebSocket etc.
3. Suport auto-reconectare dacă nu este instruit altfel, un client deconectat va încerca să se reconecteze pentru totdeauna, până când serverul este disponibil din nou
4. Detectarea deconectării. Această funcționalitate este realizată cu cronometrele setate atât pe server, cât și pe client, cu valori limită de timp (parametrii pingInterval și pingTimeout) partajați în timpul strângerii legăturii. Aceste cronometre necesită ca orice apeluri client ulterioare să fie direcționate către același server, prin urmare, cerința sesiunii lipicioase atunci când se utilizează noduri multiple
5. Suport binar orice structuri de date serializabile pot fi emise, incluzând
   1. ArrayBuffer și Blob în browser
   2. ArrayBuffer și Buffer în Node.js

Instalare ruland comanda in terminal **npm install socket.io.**

Se foloseste folosit un API si anume functia emit(data) pentru a trasmite un eveniment si date si function on() pentru a receptiona un eveniment cu parametri corespunzatori.

# MongoDB

MongoDB este o baza de date de tipul NoSQL ce foloseste notatia json (intern bson) la codificarea datelor.

Inregistrarile

O înregistrare în MongoDB este un document, care este o structură de date compusă din perechi de câmpuri și valori. Documentele MongoDB sunt similare cu obiectele JSON. Valorile câmpurilor pot include alte documente, matrice și matrice de documente.

Avantajele folosirii documentelor sunt:

Documentele (adică obiectele) corespund tipurilor de date native din multe limbi de programare.

{

name:”bogdan”,

age:29

}

Documentele încorporate și matricele reduc nevoia de îmbinări costisitoare.

Schema dinamică sprijină polimorfismul fluent.

Inalta performanta

MongoDB oferă o persistență a datelor de înaltă performanță. În special,

Suportul pentru modelele de date încorporate reduce activitatea I/O în baza de date.

Indexurile acceptă interogări mai rapide și pot include chei din documente și matrice încorporate.

# Obiectivele proiectului

Un sistem format din Raspberry Pi poate se capteze imagini si sa filmeze folosind o camera web.

Aceste imagini sunt stocate in cardul din Raspberry si analizare datorita marimi cardului SD nu se pot stoca decat 2 GB deci trebuie sa se stearga anumite inregistrari dupa o perioada de timp.

O aplicatie Android prin care puse la dispozitie utilizatorului de telefon inteligent imaginile si fisiere video. Apoi sa trimit un mesaj prin Push Notification la utilizator a se inregistreaza sau ca nu mai are spatiu in cardul SD.

Sa poata inregistra anumite evenimente .Un eveniment poate fi o miscare in fata camerei dar si se poate o comanda din smartphone.

Componente hardware :

1. Raspberry Pi cu un card SD cu 8GB
2. O camera video prin USB
3. Un smartphone ce se poate conenecta la Internet

Dezvotaterea de aplicatie android prin care sa poate fi analizata continutul capturi foto /video.

Sistem format din Raspberry trebuie :

* Sa folosesca o camera web pentru a inregistra imaginii si video
* Sa poate sa le analize folosind diverse filtre
* Sa poata sa le pastreze local in cardul SD
* Sa poate sa le trimita pe un server
* Sa poata sa fie accesibil direct de smartphone

Aplicatia sa fie sigura:

* Sa aiba un sistem de utilizatori
* Sa fie capabila sa receptioneze mesaje trimise
* Sa poate sa primesca alerte si mesaj de tipul Push Notification
* Sa poate vede o istorie a inregistrarilor ce au trebut prin fata camerei.

# Smarth Cam –Aplicatie server pentru Raspberry Pi 3

Se foloseste de node.js pentru a rula serverul ce poate capta video inregistrari ce se pot vedem mai tarziu pe aplicatie mobila.

Se foloseste de WebRTC pentru a comunica direct cu clientul de smartphone ce este o aplicatie nativa de Android.

Se poate connecta folosit unction download() {

//stopRecording();

mediaRecorder.stop();

socket.emit('video-end');

console.log(mediaRecorder.state);

console.log("recorder stopped");

console.log(mediaRecorder.state);

console.log("recorder stopped");

console.log("recorder stopped"+chunks.length);

var blob = new Blob(chunks, {type: 'video/webm'});

var url = window.URL.createObjectURL(blob);

var a = document.createElement('a');

a.style.display = 'none';

a.href = url;

a.download = 'test.webm';

document.body.appendChild(a);

a.click();

setTimeout(function() {

document.body.removeChild(a);

window.URL.revokeObjectURL(url);

}, 100);

}

Se foloseste de Socket.io pentru a inregistra video cu API MediaRecorder din standardul MediaStream.

Aplicatie mobila smarth Cam- Proiectul Android

Aplicațiile Android sunt scrise în limbajul de programare Java. Instrumentele Android SDK compilează codul împreună cu orice fișiere de date și resurse într-un pachet APK, un pachet Android, care este un fișier de arhivă cu un sufix .apk. Un fișier APK conține întregul conținut al unei aplicații Android și este fișierul pe care dispozitivele Android il folosesc pentru a instala aplicația.

Fiecare aplicație Android exista în propria spatiu izolat de sistem, protejată de următoarele caracteristici de securitate Android:

1. Sistemul de operare Android este un sistem Linux multi-utilizator în care fiecare aplicație este un utilizator diferit.
2. În mod implicit, sistemul atribuie fiecărei aplicații un cod de utilizator unic Linux (ID-ul este utilizat numai de sistem și nu este cunoscut de aplicație). Sistemul stabilește permisiunile pentru toate fișierele dintr-o aplicație, astfel încât numai numele de utilizator atribuit acelei aplicații să le poată accesa.
3. Fiecare proces are o mașină virtuală proprie (VM), astfel încât codul aplicației rulează separat de alte aplicații.
4. În mod implicit, fiecare aplicație rulează în propriul proces Linux. Sistemul Android pornește procesul când trebuie executate oricare dintre componentele aplicației, apoi oprește procesul când nu mai este nevoie sau când sistemul trebuie să recupereze memoria pentru alte aplicații.
5. Sistemul Android implementează principiul celor mai puțin privilegiate. Adică, fiecare aplicație are, în mod implicit, acces doar la componentele pe care le cere pentru a-și desfășura activitatea și nu mai mult. Acest lucru creează un mediu foarte sigur în care o aplicație nu poate accesa părți ale sistemului pentru care nu i se acordă permisiunea. Cu toate acestea, există modalități pentru ca o aplicație să partajeze date cu alte aplicații și pentru ca o aplicație să acceseze servicii de sistem:

O aplicație poate solicita permisiunea de a accesa datele dispozitivului, cum ar fi contactele utilizatorului, mesajele SMS, spațiul de stocare montat (cardul SD), camera și Bluetooth. Utilizatorul trebuie să acorde în mod explicit aceste permisiuni. Pentru mai multe informații, consultați Lucrul cu permisiunile de sistem.

Restul acestui document introduce următoarele concepte:

Componentele cadru de bază care definesc aplicația de android:

1. Fișierul manifest în care declarați componentele și caracteristicile necesare ale dispozitivului pentru aplicația dvs.
2. Resurse care sunt separate de codul aplicației și care permit aplicației să-și optimizeze în mod grațios comportamentul pentru o varietate de configurații de dispozitive.

O activitate este punctul de intrare pentru interacțiunea cu utilizatorul. Acesta reprezintă un singur ecran cu o interfață de utilizator. De exemplu, o aplicație de e-mail poate avea o singură activitate care afișează o listă de e-mailuri noi, o altă activitate pentru a compune un e-mail și o altă activitate de citire a e-mailurilor.In cazul nostru lista de inregistrari sau sau setari. Deși activitățile lucrează împreună pentru a forma o experiență de utilizare coerentă în aplicația de e-mail, fiecare este independentă de celelalte. Ca atare, o altă aplicație poate iniția oricare dintre aceste activități dacă aplicația de e-mail o permite. De exemplu, o aplicație de cameră poate începe activitatea în aplicația de e-mail care compune poștă electronică nouă pentru a permite utilizatorului să partajeze o fotografie. O activitate facilitează următoarele interacțiuni cheie între sistem și aplicație:

Urmărirea a ceea ce îngrijorează în prezent utilizatorul (ceea ce se află pe ecran) pentru a vă asigura că sistemul continuă să desfășoare procesul care găzduiește activitatea.

Știind că procesele folosite anterior conțin lucruri pe care utilizatorul le poate reveni la activități (oprite) și, prin urmare, au un rol mai important în menținerea acestor procese în jur.

Ajutând aplicația să se ocupe de procesul său dezactivat, utilizatorul se poate întoarce la activități cu restabilirea stării sale anterioare.

Furnizarea unei modalități prin care aplicațiile să implementeze fluxurile de utilizatori între ele, iar sistemul să coordoneze aceste fluxuri. (Exemplul cel mai clasic aici este de a împărtăși.)

Implementați o activitate ca o subclasă a clasei de activitate. Pentru mai multe informații despre activitatea din Ghidul dezvoltatorilor de activități.Este devoltata in Android Studio si se pazeaza pe librariile oferte de Google in java.Este conceputa sa fie usor de folosit in controlul aplicatie serverul dar si comunitatie audio /video .

SmarthCam este impartia in cateva module de baza si anume

1. Api cu serverul
   * Autentificare user /parola
   * Inregisterare noi utizatori
2. Streaming live
   * Se produce prin componenta webRTC din web
3. Control pentru inregistrarea video prin webSockets
   * Se foloseste un de libraria **socket.io** pentru a creea un API super rapid de control al inregistrilor
4. Offline
   * Comunica prin push Notificatie adica serverul trimite notificari daca apar evenimente.Un eveniment poate fi o eroare a serverului spatiu insuficient sau pot fi create de utilizator.
5. Vizionare de inregistrari din server.
   * Se trimite direct din serverul HomeCam catre

SmarthCam

Este impartina in trei taburi pentru a putea sa naviga usor intre ele:

1. Activitatea Live in direct ce se vede la camera
2. Activitatea Arhive ce contine toate inregistrarile pe care server .
3. Activitatea Recente ne prezinta cele mai recente inregistrari sau activitati

# Activitea Live (direct)

Acest a

# Anexe

# Bibliografie

*http://webrtc-security.github.io*. (2017, May 10).

*https://developer.mozilla.org/en-US/docs/Web/API/WebRTC\_API*. (2017, May 20). Preluat de pe https://developer.mozilla.org/.

*https://docs.npmjs.com/getting-started/what-is-npm*. (fără an).

*https://en.wikipedia.org/wiki/WebSocket*. (fără an).

*https://en.wikipedia.org/wiki/WebSocket*. (fără an).

*https://github.com/socketio/socket.io/*. (fără an).

*https://github.com/v8/v8/wiki/Introduction*. (fără an).

*https://nodejs.org/en/*. (fără an).