|  |  |  |
| --- | --- | --- |
| Facultatea de Automatica și Calculatoare |  | Sisteme Distribuite și Tehnologii Web |

Topichat

Aplicatie Mobilă crossplatform

|  |
| --- |
| Coordonator științific  Prof.dr.inf. Mitică CRAUS |
| **Absolvent**  Ciprian Ionut Holbia |

Cuprins

[Rezumatul lucrării 3](#_Toc485564488)

[Fundamentarea teoretică și documentarea bibliografică 4](#_Toc485564489)

[Aplicații Existente Pe Piața 4](#_Toc485564490)

[Facebook Messenger 5](#_Toc485564491)

[WhatsApp Messenger 6](#_Toc485564492)

[WeChat 7](#_Toc485564493)

[Alte Aplicații 7](#_Toc485564494)

[Specificațiile Aplicației 8](#_Toc485564495)

[Sisteme de operare suportate 8](#_Toc485564496)

[Device-uri suportate 8](#_Toc485564497)

[Protocolul de comunicare 8](#_Toc485564498)

[Idei pentru interfața cu utilizatorul 11](#_Toc485564499)

[Proiectarea aplicației 12](#_Toc485564500)

[Protocolul MQTT 12](#_Toc485564501)

[Istoria protocolului 12](#_Toc485564502)

[Cum functionează 12](#_Toc485564503)

[Metoda de comunicare publish/subscribe 13](#_Toc485564504)

[MQTT Client 14](#_Toc485564505)

[MQTT Broker 14](#_Toc485564506)

[Xamarin 15](#_Toc485564507)

[Utilizare 15](#_Toc485564508)

[Xamarin Forms 16](#_Toc485564509)

[Configurarea Proiectului 16](#_Toc485564510)

[Serverul MQTT - MQTT Broker 18](#_Toc485564511)

[Introducere 18](#_Toc485564512)

[Beneficii si avantaje 19](#_Toc485564513)

[Instalarea si configurarea 19](#_Toc485564514)

[Implementarea aplicației 20](#_Toc485564515)

[Comunicarea cu Broker-ul 20](#_Toc485564516)

[Protocolul de comunicare 21](#_Toc485564517)

[Comunicarea unul la unul 22](#_Toc485564518)

[Comunicarea într-un grup 22](#_Toc485564519)

[Structuri de date folosite 23](#_Toc485564520)

[Message 23](#_Toc485564521)

[Topic 23](#_Toc485564522)

[Conversation 23](#_Toc485564523)

[Interfața cu Utilizatorul 23](#_Toc485564524)

[Pagina conversațiilor 24](#_Toc485564525)

[Pagina subiectelor de discuție 25](#_Toc485564526)

[Pagina contactelor 25](#_Toc485564527)

[Pagina de comunicare 26](#_Toc485564528)

[Codul sursă 27](#_Toc485564529)

[Probleme și dificultați întâlnite 28](#_Toc485564530)

[Testarea aplicației 28](#_Toc485564531)

[Testarea comunicarii 28](#_Toc485564532)

[Testarea business logic 29](#_Toc485564533)

[Testarea interfeței 30](#_Toc485564534)

[Concluzii 31](#_Toc485564535)

[Bibliografie 32](#_Toc485564536)

# Rezumatul lucrării

Topichat este o aplicație de chat pentru telefoane mobile bazată pe subiecte de discuție. Toate conversațiile utilizatorului vor fi grupate pe subiecte în așa fel încât găsirea unei anumite discuții sa fie foarte simplă.

Am ales acesta tema deoarece am considerat că o astfel de aplicație mobilă este foarte folositoare utilizatorilor și deocamdată nu există ceva asemănător.

Aplicația va folosi un protocol de comunicare numit MQTT foarte performant si rapid, folosit foarte mult în Internet Of Things (IoT). Tehnologia folosita în implementarea aplicației va fi Xamarin (C#). Folosind Xamarin putem avea același cod C# pentru toate platformele.

Serverul de MQTT folosit se numește EMQ, o implementare open source. Cunoscut sub numele de MQTT Broker, serverul va fi instalat pe o instanța AWS (Amazon Web Services). Acea mașina virtuala aflata în AWS va folosi Ubuntu ca și sistem de operare.

Deocamdată un Backend nu va fi implementat deoarece se dorește ca utilizatorii să poată folosi aplicația fără a fi nevoiți sa își creeze un cont. Ca și identificator pentru utilizatori vom folosi numărul de telefon, similar cu abordarea aplicației Whatsapp.

Conversațiile utilizatorilor vor fi stocate local, pe dispozitiv și nu undeva in cloud. Acesta abordare este mult mai benefică utilizatorului deoarece nu apar probleme de securitate și protejarea datelor personale.

Comunicarea cu Broker-ul MQTT va fi securizata folosind SSL/TLS.

# Fundamentarea teoretică și documentarea bibliografică

Având în vedere că acum majoritatea personalelor folosesc device-urile mobile, în special telefoanele mobile smartphone ca si device-ul lor principal, am ales să implementez o aplicație pentru telefoane mobile.

Unul din principalele motive de folosi un astfel de device este comunicarea. În 2017 portofoliul de aplicații de comunicare este foarte divers. Pe piața putem găsi aplicații de la companiile gigant cum ar fi Google, Facebook până la aplicații micuțe folosite doar in unele regiuni ale lumii.

De asemeni în 2017 lumea a început sa pună accent pe securitate si protejarea datelor personale. Acestea Persoanele care sunt preocupate de protejarea datelor personale vor evita sa folosească aplicațiile marilor companii deoarece este bine-cunoscut faptul că deși aceste aplicații par a fi pe gratis, utilizatorii plătesc pentru ele cu datele lor personale. De obicei datele personale si activitatea utilizatorului online sunt folosite în scopul publicități si a ofertelor direcționate perfect pe tipul profilului utilizatorului.

O alta problemă a aplicațiilor care sunt acum existente pe piața este organizarea conversațiilor. De multe ori avem problema în a găsi o anumita conversație pe un anumit subiect care s-a desfășurat mult în trecut. De exemplu, o conversație de acum 3-4 ani când s-a planificat o petrecere de revelion si vrei să vrei sa afli numele restaurantului sau a hotelului pentru ca ti-a plăcut foarte mult.

Din cele spuse mai sus vine idea proiectului. O aplicație mobila care este organizata pe subiecte de discuție, cum ar fi „Petrecere de revelion” și care pune mult accent pe securitate si protecția datelor personale.

## Aplicații Existente Pe Piața

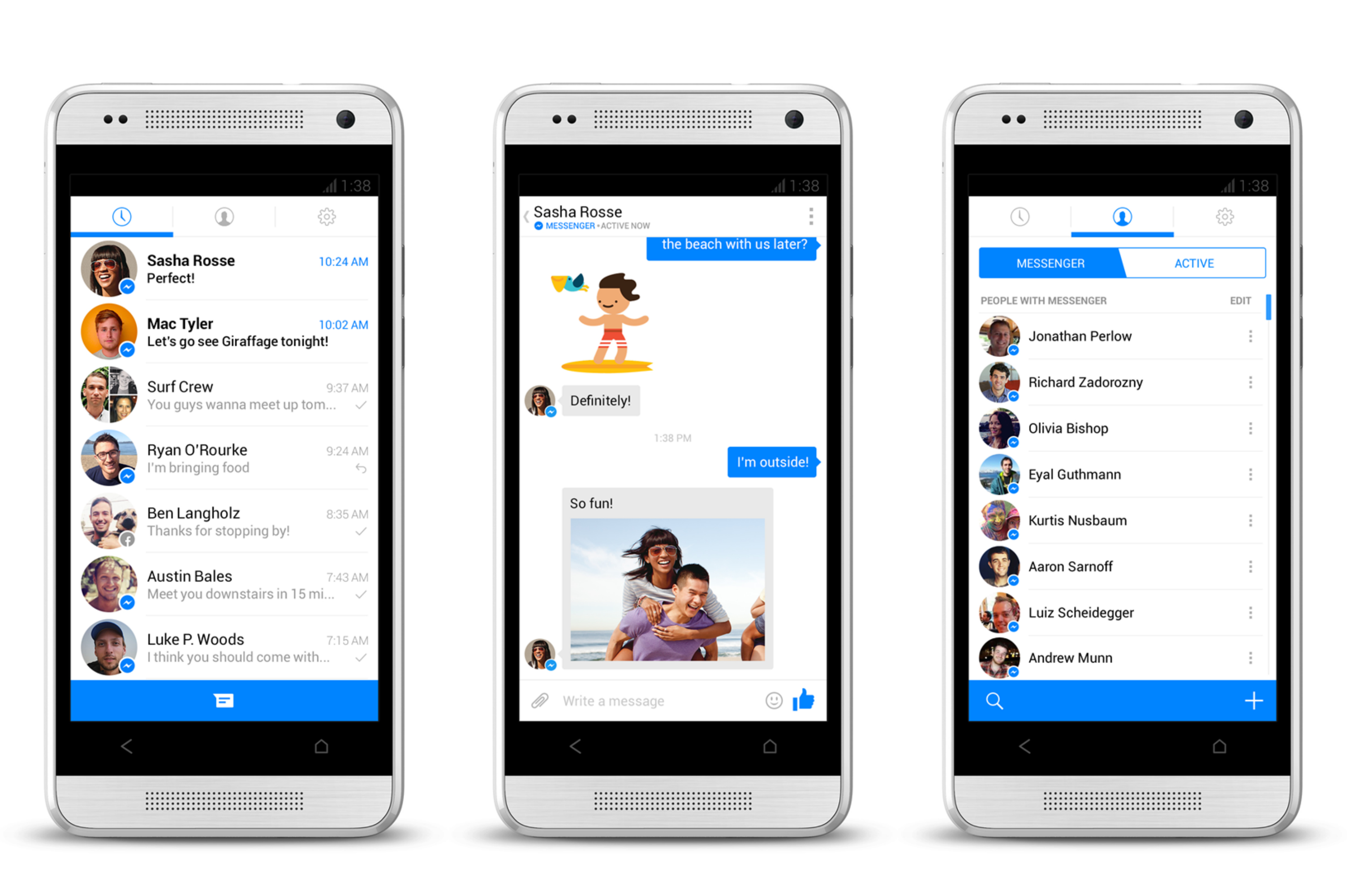
După cum am menționat in introducere acum pe piața este un număr foarte mare de aplicații destinate comunicării. Potrivit site-ului [Engadget.com](http://www.engadget.com) cele mai populare aplicații de comunicare sunt in ordinea următoare:

1. Facebook Messenger
2. WhatsApp Messenger
3. WeChat
4. Line
5. Viber
6. Blackberry Messenger (BBM)
7. Telegram Messenger
8. Kakaotalk
9. IMO
10. Skype
11. Snapchat
12. KIK

În următoarele pagini le von analiza pe cele mai cunoscute din aceste aplicații.

### Facebook Messenger

Messenger este cea mai cunoscuta si cea mai descărcata aplicație de comunicare. Ea are în jur de 100 de miliarde de utilizatori în toata lumea.

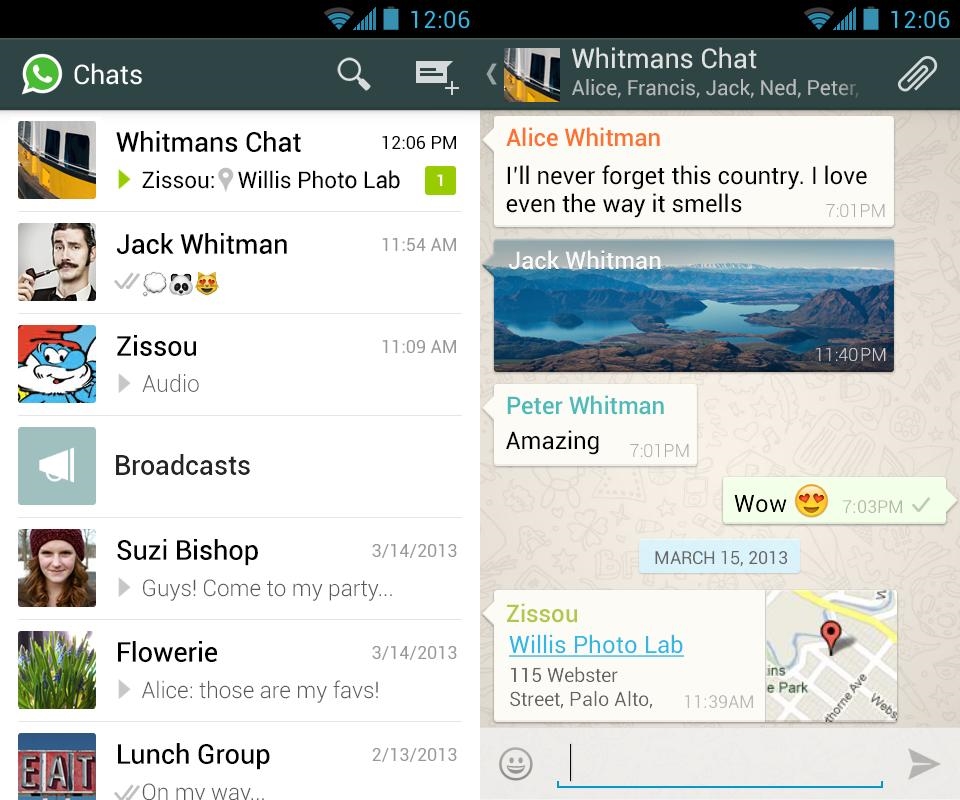


Fiind cea mai cunoscuta si utilizată aplicație de comunicare, utilizatorii care sunt mai preocupați de securitate si protejarea datelor personale au tendința de a evita aceasta aplicație.

Facebook este una din cele mai mari companii daca nu cea mai mare care oferă dezvoltatorilor de aplicații publicitate pentru fi integrata in aplicațiilor lor. Ei sunt foarte cunoscuți pentru ofertele foarte bine direcționate pe profilul utilizatorului. Având atât de mulți utilizatori si știind atât de multe despre utilizatorii lor le este foarte ușor sa facă acest lucru.

### WhatsApp Messenger

Similar cu Facebook Messenger, WhatsApp este instalat de peste un miliar de oameni. Acesta aplicație a fost foarte îndrăgita de utilizatori în trecut pentru simplitatea și anonimitatea ce o oferea.



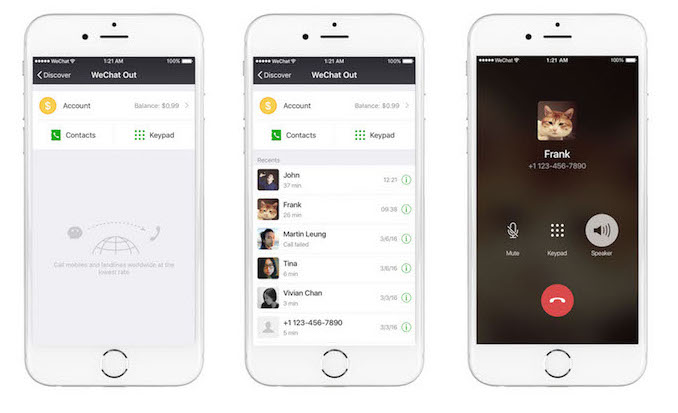
O mare problema a acestei aplicații în trecut era securitatea. Mesajele erau trimise în clar, lucru care nu o recomanda utilizatorilor preocupați de securitate.

În 2014 Facebook a anunțat că a achiziționat WhatsApp Messenger pentru suma de 1.5 miliarde de dolari. De atunci compania gigant a investit foarte multe in noi funcționalități și de aceea acum aplicația numără in jur de 1 miliard de utilizatori. De asemenea problemele de securitate au fost rezolvate, acum aplicația suporta criptare end to end.

Similar cu Facebook Messenger acesta aplicație ridică mari semne de întrebare în privința protejări datelor personale.

### WeChat

WeChat este soluția folosita de chinezi pentru comunicare. WeChat este mult mai mult decât o aplicație de comunicare. În principiu poți sa o folosește pentru aproape orice.



Aplicația poate fi folosita pentru ați plăti facturile, pentru comanda un taxi, pentru ați rezerva un bilet de avion, a investi la bursă si multe altele. E foarte greu sa catalogam acesta aplicație ca o aplicație destinata comunicării.

### Alte Aplicații

Alte aplicatii precum Skype sau Viber sunt mai mult concepute pentru convorbiri voce (VoIP).

## Specificațiile Aplicației

Similar aplicației WhatsApp, Topichat va fi strict o aplicației de comunicare. Utilizatorii vor comunica „unu la unu” cât si cu un grup de persoane. Diferența pe care o introduce Topichat este ca toate conversatiile utilizatorilor vor fi grupate pe topicuri.

Utilizatorii nu vor putea sa creeze o noua discuta vara îi a seta un topic acesteia. Fiecare conversație cu o persoana sau cu un grup de persoane va conține unul sau mai multe topicuri.

### Sisteme de operare suportate

Ne propunem sa suportam majoritatea sistemelor de operare acum existente pe piața de telefoane mobile.

Android si iOS sunt cele mai cunoscute si populare sisteme de operare, dar ar fi idea sa putem sa suportam si Windows Phone. O abordare ar fi sa dezvoltam 3 aplicații diferite pentru fiecare sistem de operare sau să folosim un Framework deja existent care ne-ar permite sa implementam doar o singura aplicație pentru toate cele 3 sisteme de operare.

### Device-uri suportate

Se dorește suportarea majorității dispozitivelor iOS începând cu iPhone 5 și cele mai populare dispozitive Android. Va fi mult mai ușor testarea pe sistemul de operare a celor de la Apple pentru că dispozitivele sunt puține si bine cunoscute dar după cum știm pentru Android sunt mii de dispozitive existente și de multe ori până și companiile mari de software au probleme în a suportat toate aceste dispozitive.

### Protocolul de comunicare

Aplicația va implementa un protocol de comunicare care va îndeplini următoarele criterii:

* Rapid si ușor de implementat
* Sigur si securizat
* Cunoscut si bine documentat
* Folosit deja în alte aplicații similare

Protocoalele care îndeplinesc deplin sau parțial criteriile de mai sus sunt XMPP si MQTT.

Pentru a alege unul din ele trebuie sa știm avantajele si dezavantajele fiecăruia.

#### XMPP

Avantaje:

* Este un protocol bazat pe XML
* Foarte popular. Exista deja multe librarii client pentru majoritatea limbajelor de programare.
* Necesita mai puțin efort pentru implementare ca alte protocoale
* Suporta trimiterea de Poze/Video

Dezavantaje:

* Structura mesajelor este foarte mare. Nu este optimizat pentru date mobile.
* Autentificare se face cu Jid (Jabber id). Deci necesita crearea de un cont înainte de a putea comunica.
* Este o tehnologie învechita
* Nu suporta criptate end to end. Am putea folosi un plugin pentru acest lucru dar chiar si acesta suporta doar comunicarea unu la unu.

#### MQTT

Avantaje:

* Este foarte eficient și rapid
* Este un protocol foarte popular mai ales în industria IoT din cauza performanței ridicate
* Cea mai mare aplicație de comunicare Facebook Messenger folosește acest protocol. Având in vedere numărul foarte ridicat de utilizatori pe care îl au si ei au avut pe primul loc performanta când au ales protocolul de comunicare.

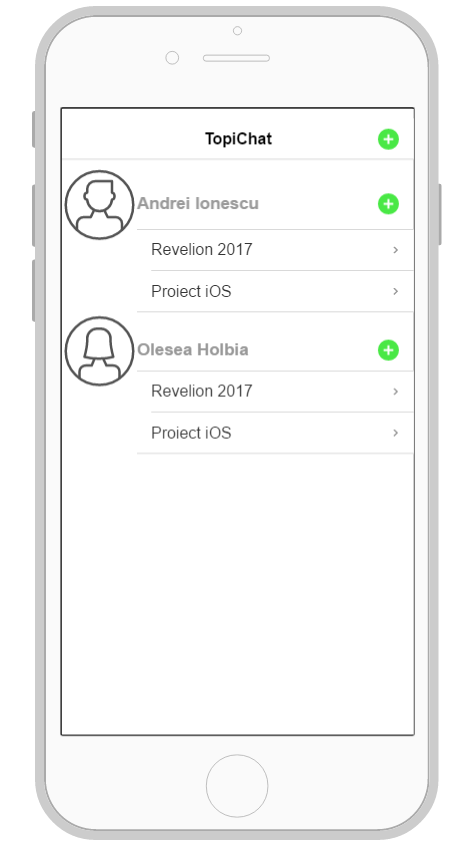
Dezavantaje:

* Deși Facebook Messenger folosește acest protocol, nu exista pe internet prea multe exemple sau idei despre cum ar fi ideal sa folosim acest protocol pentru aplicația noastră.
* Structura mesajelor trebuie definita si implementata de noi. Nu exista un standard.
* Nu exista prea multe librarii client pentru limbajul de programare C#.

Analizând avantajele si dezavantajele fiecărui protocol prezentat mai sus am ales sa mergem pe o tehnologie de viitor și anume MQTT. Chiar dacă deocamdată nu exista prea multa documentație despre acest protocol va face aceasta lucrare cu atât mai interesanta.

### Idei pentru interfața cu utilizatorul

Ne propunem sa grupam toate conversațiile utilizatorului pe subiecte de discuție. De aceea trebuie evidențiat acest lucru în interfața produsului.



După cum vedem în modelul de mai sus sunt doua contacte, unde fiecare au câte doua subiecte de discuție.

Pentru ca utilizatorul sa folosească aceasta abordare trebuie sa fim siguri ca este foarte ușor de creat o conversație cu un anumit topic.

# Proiectarea aplicației

După cum am prezentat în capitolul precedent ne propunem sa implementam o aplicație software care sa suporte un hardware deja existent. De aceea în acest capitol vor vorbi doar despre proiectare software.

Vom grupa aplicația in următoarele module:

* Protocolul de comunicare folosit
* Framework-ul folosit pentru implementarea clienților
* Instalarea si configurarea serverului

## Protocolul MQTT

MQTT este un protocol de comunicate client/server de tipul publish/subscribe. Acest protocol a fost conceput pentru a fi foarte eficient, rapid și ușor de implementat. Aceste caracteristici îl fac ideal pentru a fi folosit în IoT (Internet of things) datorita dimensiunii reduse a mesajului.

### Istoria protocolului

MQTT a fost inventat de Andy Stanford-Clark (IBM) și Arlen Nipper (Arcom, acum Cirrus Link) în 1999. Atunci, idea a fost de a crea un protocol de comunicare care folosește un minim de baterie și care are o dimensiune redusă pentru conectarea conductelor de petrol prin conexiune satelit. Specificațiile proiectului erau:

* Simplu de implementat
* Verificarea datelor trimise
* Eficient
* Protocol real-time

Aceste obiective sunt în continuare nucleul MQTT, în timp ce accentul sa schimbat de la sistemele embedded sper utilizarea în IoT.

### Cum functionează

MQTT folosește o structură de comunicare de tip publish/subscribe.

Serverul MQTT cunoscut sub numele de MQTT Broker este o aplicație server foarte eficienta ce poate fi instalată pe aproape orice sistem de operare (Mac, Linux, Windows). Fiind o aplicație mica este ideală pentru device-uri embedded sau device-uri cu resurse mai puține gen Raspberry Pi.

Clienții poate fi orice dispozitiv care implementează comunicarea de tipul publish/subscribe.

### Metoda de comunicare publish/subscribe

Metoda de comunicare folosita de protocolul MQTT este o alternativă a metodei clasice client-server unde un client comunica direct cu un endpoint. Folosind MQTT, clienții sunt decuplați unul de celălalt. Asta înseamnă că un client care trimite un mesaj (publish) nu va ști destinatarul mesajului care poate fi unul sau mai mulți clienți care fac subscribe la acel tip de mesaj.



În diagrama de mai sus putem vedea trei clienți conectați la Broker. Un senzor de temperatură, un telefon mobil și un laptop. Telefonul mobil și laptopul subscriu la mesaje de tip "temperatură". Atunci când intervine o schimbare de temperatură, senzorul de temperatură va publica un mesaj de tip "temperatură" care va fi redirecționat de Broker către toți clienții, în cazul nostru telefonul mobil și laptopul.

### MQTT Client

Un client MQTT este orice dispozitiv de la un microcontroler până la un server, care folosește o librărie MQTT și se conectează la un broker MQTT pe orice tip de rețea. Acesta ar putea fi un dispozitiv foarte mic și limitat în resurse, care este conectat printr-o rețea WiFi și care folosește o libărie MQTT optimizata la maxim sau un computer tipic care rulează un client grafic. Practic orice dispozitiv care suporta TCP/IP și implementează comunicarea MQTT.

Librării de clienți MQTT sunt disponibile pentru o mare varietate de limbaje de programare, de exemplu Android, Arduino, C, C ++, C #, Go, iOS, Java, JavaScript, .NET. O listă completă poate fi găsită pe wiki-ul MQTT.

### MQTT Broker

Contrar unui client MQTT este brokerul MQTT, care este inima oricărui protocol de comunicare publish/subscribe. În funcție de implementare, un broker poate gestiona până la mii de clienți MQTT conectați simultan.

Brokerul este în primul rând responsabil pentru primirea tuturor mesajelor, pentru filtrarea lor, pentru a decide cine este interesat de acesta și apoi trimiterea mesajului către toți clienții abonați. De asemenea, deține sesiunea tuturor clienților persistenți, inclusiv subscripțiile și mesajele pierdute. O altă responsabilitate a brokerului este autentificarea și autorizarea clienților. De cele mai multe ori, un broker este de asemenea extensibil, ceea ce permite integrarea cu ușurință a unei autentificări personalizate (multe aplicații folosesc autentificări personalizate).

Brokerul este hub central, prin care fiecare mesaj trebuie să treacă. Ca urmare, este important ca acesta să fie foarte scalabil, integrabil în sistemele Backend, ușor de monitorizat și, desigur, stabil si robust.

## Xamarin

Xamarin este o companie de software din San Francisco, California, înființată în 2011 de către inginerii care au creat Mono, Mono for Android și MonoTouch, care sunt implementări cross-platform ale Common Language Infrastructure (CLI) și Common Language Specifications (Microsoft .NET).

Folosind limbajul de programare C#, programatorii pot folosi instrumentele Xamarin pentru a scrie aplicații native Android, iOS și Windows cu interfețe de utilizator native și cod partajat pe mai multe platforme, inclusiv Windows și MacOS. Potrivit lui Xamarin, peste 1,4 milioane de dezvoltatori folosesc produsele Xamarin în 120 de țări din întreaga lume începând cu aprilie 2017.

Pe 24 februarie 2016, Microsoft a anunțat că a semnat un acord definitiv pentru a achiziționarea Xamarin. Acum Visual Studio vine la pachet cu framework-ul Xamarin. De asemenea din 2017 Microsoft a lansat „Visual Studio for Mac”, varianta pentru OSX a cunoscutului IDE.

### Utilizare

Vom folosi Visual Studio for Mac ca și IDE deoarece este singura opțiune pentru a dezvolta aplicații pe iOS. Apple nu permite dezvoltatorilor sa creeze aplicații pentru sistemele lor de operare fără a avea cel puțin un device Apple. De asemenea nu avem posibilitatea de a testa aplicația pe un device fizic deoarece pentru acest lucru avem nevoie de un cont de dezvoltator Apple autorizat (Apple D0eveloper account) care costa 100 de Euro pe an. Din cauza acestor impedimente vom dezvolta si rula aplicația pe un emulator.

Deoarece ne propunem sa dezvoltam o aplicație cross platform va trebui sa ne orientam si către sistemul de operare al celor de la Google si anume Android. Folosind Visual Studio for Mac și Xamarin acest lucru este foarte simplu deoarece IDE-ul instalează automat si Android Sdk împreuna cu emulatoarele de care avem nevoie.

### Xamarin Forms

Xamarin Forms este abordarea celor de la Xamarin si Microsoft de a putea implementa interfața cu utilizatorul într-un mod cross platform.

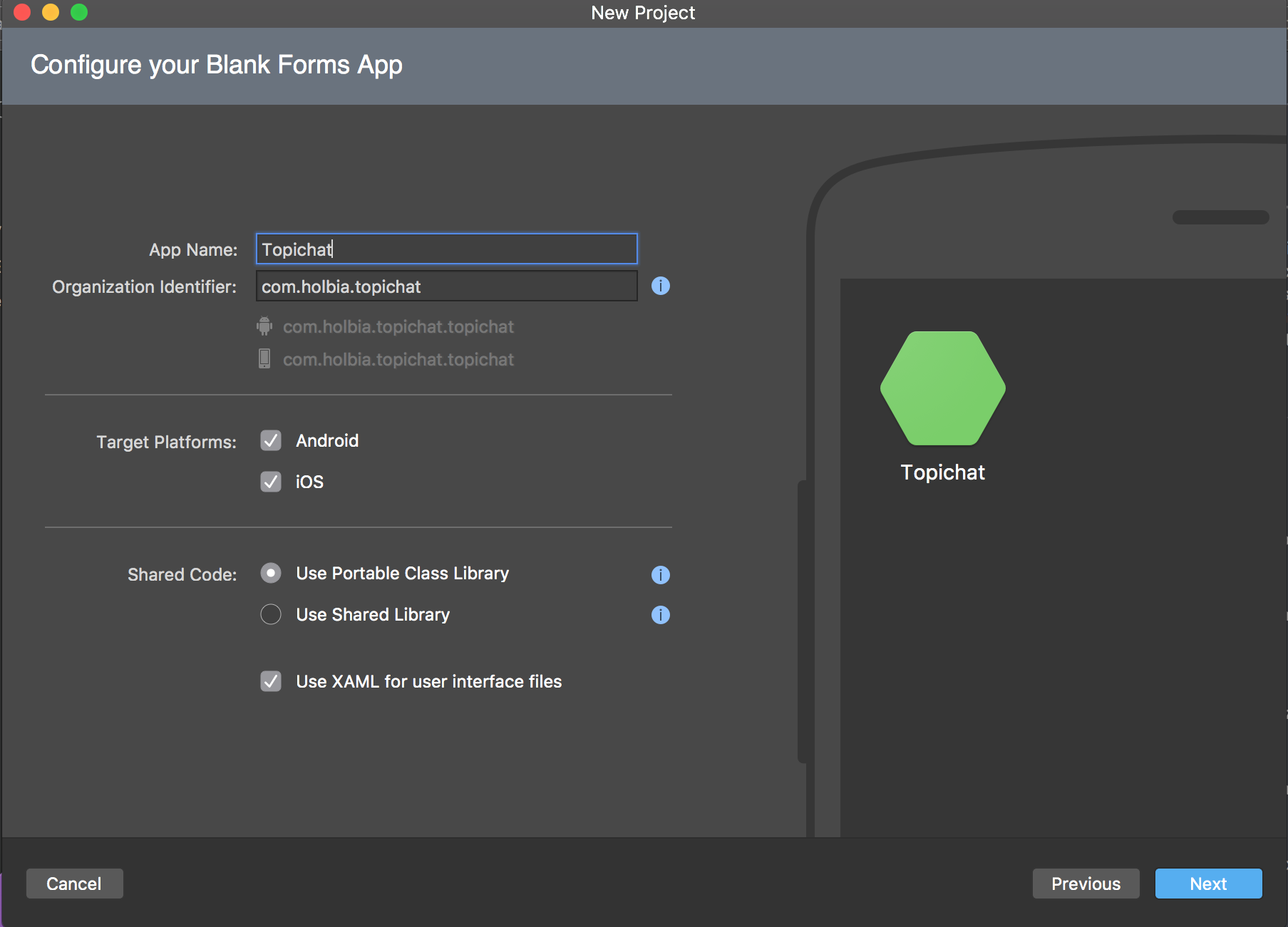
Xamarin Forms este o modalitate ideală de a implementa interfețe cross platform care nu necesită prea multe funcționalități specifice fiecărei platforme. Folosind aceasta abordare putem partaja in proporții de 90% din codul aplicației. De asemenea este mai confortabil de implementat interfața deoarece Xamarin Forms folosește XAML, unde programatorul își definește controalele in limbaj XML.

### Configurarea Proiectului

În următoarele pagini vom prezenta cum putem crea un proiect cross platform folosind Visual Studio for Mac.

#### Selectarea tipului de proiect

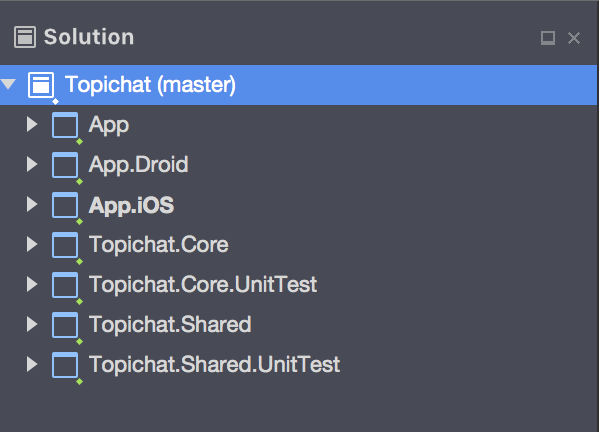
Pentru a crea un proiect cu o interfața cross platform vom selecta New Solution și de acolo vom alege Blank Forms App, adică o aplicație Xamarin Forms goală, fără nici o pagina. Paginile le vom crea noi mai târziu.



După cum vedem in imaginea de mai sus acest proiect este destinat pentru platformele iOS si Android.

De asemenea avem posibilitatea de a alege tipul de librărie unde vom avea codul partajat. Ca si default este ales Portable Class Library (PCL) si vom alege aceasta opțiune.

#### Structura proiectului



După cum vedem in imaginea de mai sus vom avea 5 proiecte principale plus alte două proiecte pentru testare (UnitTest).

**App**

Primul proiect conține interfața cross platform. Acest proiect este un proiect PCL care conține o referinta către Xamarin forms. El va conține toate paginile aplicației care vor fi definite in XAML și de asemenea tot codul partajat ce ține de interfața cu utilizatorul. Toate proiectele platform specific vor avea o referința către acest proiect.

**App.Droid**

Acest proiect este proiectul specific sistemului de operare Android. El va conține numai implementarea specificare acestui sistem de operare.

**App.iOS**

Similar proiectului anterior App.iOS este un proiect specific sistemului de operarea a celor de la Apple. De asemenea el va conține numai implementarea specificare sistemului de operare iOS.

**Topichat.Core**

Acest proiect este un proiect de tip PCL. El va conține tot codul de funcționalitate a aplicației care poate fi partajat. Folosind o librărie de tip PCL avem avantajul că putem sa testăm această librărie cu un proiect UnitTest si de asemenea acest tip de librărie poate fi folosita pe toate sistemele de operare. Dezavantajul proiectelor PCL este ca nu implementează tot .net Framework-ul ci doar o parte care nu necesită implementări specifice platformei.

**Topichat.Shared**

Ultimul proiect va fi cumva similar proiectului Core. Va conține implementări care vor fi partajate în proiectele platform specific. Diferența este ca acest tip de proiect nu este compilabil/link-abil. El grupează un set clase care sunt incluse in proiectele platform specifice si acolo se face compilarea si link-area codului. Avantajul acestui proiect este ca putem folosi toata funcționalitatea .net Framework deoarece fiecare proiect va folosi implementarea specifica din Mono pentru acea funcționalitate.

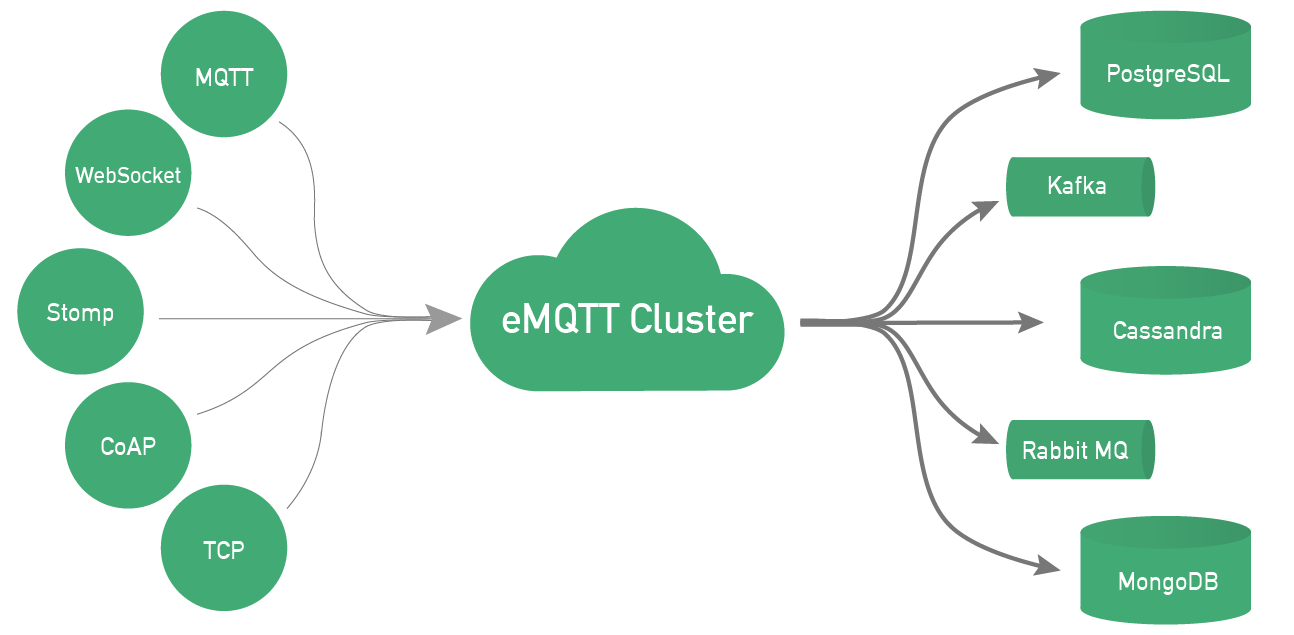
## Serverul MQTT - MQTT Broker

EMQ (Erlang MQTT Broker) este serverul de MQTT folosit pentru implementarea aplicației.

### Introducere

EMQ este un MQTT Broker distribuit, masiv scalabile si extrem de extensibil scris în limbajul de programare Erlang / OTP.

EMQ este un server MQTT complet open source avand licența Apache versiunea 2.0. EMQ implementează specificațiile MQTT versiunea 3.1 cât și v3.1.1 și suporta în același timp MQTT-SN, CoAP, WebSocket, STOMP și SockJS.



### Beneficii si avantaje

* Open source
* Foarte scalabil
* Suportă specificațiile ultimei versiuni MQTT (3.1.1)
* Foarte ușor de instalat
* Modular si extensibil prin plugin-uri

### Instalarea si configurarea

Pentru instalarea Broker-ului MQTT EMQ, vom folosi o mașina virtuala aflat in AWS (Amazon Web Services).

AWS oferă utilizatorilor posibilitatea de o folosi o instanță EC2 gratis. Acesta oferta este valabila doar un an de zile dar este o opțiune ideala pentru prima versiune a produsului.

Pe acesta mașina virtuala vom folosi Ubuntu x64.

Pentru instalarea serverului vom ultimul pachet oficial de pe site-ul <emqtt.io>.

Următorul pas este decomprimarea produsului si pornirea broker-ului.

1. unzip emqttd-centos64-v2.0.zip
2. cd emqttd && ./bin/emqttd console

După ce primi doi pași au fost executați fără probleme, avem posibilitatea de a porni broker-ul ca Daemon. Acest lucru este foarte recomandat deoarece în cazul unei probleme în care server-ul necesita repornire, broker-ul va porni automat.

Următorul pas este de a ne asigura că pe mașina noastră virtuala din cloud avem deschise cele doua porturi necesare Broker-ului EMQ. Predefinit cele două porturi sunt 1883 pentru comunicarea necriptata si 8883 pentru SSL/TLS.

# Implementarea aplicației

După cum am prezentat în capitolul precedent vom folosi Visual Studio for Mac împreuna cu Xamarin pentru implementarea proiectului. Proiectul va fi open source salvat pe Github. Inițial ne vom focusa pe platformele iOS si Android pentru că dețin majoritatea pieței de telefoane mobile.

Fazele implementării vor fi următoarele:

* Implementarea comunicării cu broker-ul.
* Definirea unui protocol de comunicare între clienți
* Implementarea structurilor de date necesare pentru protocolul definit
* Implementarea interfeței

## Comunicarea cu Broker-ul

Pentru comunicarea cu broker-ul MQTT avem la dispoziție câteva librarii deja existente. Din păcate pentru C# nu avem foarte multe opțiuni. Singura librărie care poate fi folosita în proiectul nostru cross platform este [M2Mqtt Dotnet Core](https://www.nuget.org/packages/M2MqttDotnetCore/).

#### Autentificarea clienților

Autentificarea clienților va fi bazată pe numărul de telefon. Mesajele vor fi trimise inițial pe portul 1883 necriptate, ca după într-o versiune următoare toți clienții sa folosească portul 8883 cu TLS/SSL activat. După ce versiune cu TLS/SSL activat va fi disponibila vom bloca pe server portul 1883.

#### Conectarea la Broker

Pentru conectare se folosește ca și Client ID numărul de telefon. Ne vom înregistra pentru mesaje primite folosind eventul MqttMsgPublishReceived.

this.mqttClient = new MqttClient(BrokerUrl);

this.mqttClient.MqttMsgPublishReceived += MqttMsgPublishReceived;

this.mqttClient.Connect(contactManager.Me.PhoneNumber);

if (this.mqttClient.IsConnected)

{

this.mqttClient.Subscribe(new string[] { $"{MqttTopicPrefix}/{contactManager.Me.PhoneNumber}/#" }, new byte[] { MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE });

}

După ce ne conectam la Broker, vom face subscribe la toate mesajele destinate numărului de telefon folosit ca si Client ID.

## Protocolul de comunicare

Pentru a ști cu cine comunicăm și câte persoane sunt implicate în conversație va trebuie sa definim un protocol de comunicare standard care va fi implementat de toți clienții care se vor conecta la broker.

**Message / [ReceiverId] / [TopicId] / [TopicName] / [MembersIDs] / [SenderID]**

ReceiverId – Numărul de telefon al receptorului.

TopicId – Id-ul subiectului. Acesta va fi un GUID unic.

TopicName – Subiectul de discuție

MembersIDs – Membri implicați în discuție

SenderID – Numărul de telefon al celui care trimite mesajul.

Folosind acest model fiec1are client va face subscribe la message/[MyId]. Acest lucru va garanta faptul ca el va primi toate mesajele adresate lui.

Atunci când un utilizator vrea sa nu mai primească mesajele care sunt trimise pentru un anumita discuție/subiect el va putea face unsubscribe folosind TopicId-ul.

Folosind aceasta structura un client va ști din structura mesajului MQTT toate detaliile despre discuți, de la subiect pana la membri discuției.

### Comunicarea unul la unul

În cazul unui utilizator cu numărul de telefon 0744360800, acesta va face subscribe la message/0040744360800/#. Cel care inițiază discuția va asigna automat un ID discuției (GUID unic).

Un expeditor cu numărul de telefon 0040740660810 va face publish la:

message/ 0040744360800 /a55e4be9-977d-4f2d-a060-7614e4ef5772/VacantaLaMunte/0040740660810/0040740660810

atunci când vrea sa comunice cu 0744360800 despre subiectul „Vacanta la munte”. Parametrul MembersIDs va conține numai numărul de telefon al celui care trimite deoarece este o conversație unu la unu.

### Comunicarea într-un grup

Similar comunicării unu la unu, într-un grup administratorul grupului va asigna automat un ID discuției.

Diferența fata de conversațiile unu la unu este parametrul MembersIDs. Într-un grup acest parametru va conține toți membri separați prin caracterul „-”.

De exemplu într-un grup de trei persoane, când un utilizator cu numărul de telefon 0744360800 vrea sa trimită un mesaj nou, el va genera trei mesaje pentru fiecare destinatar.

message/0040744555555/d428a487-9932-471c-b701-74aef7889b4e/Revelion/0040740000000-0040740111111-0040740222222/0040740000000

message/0040740111111/d428a487-9932-471c-b701-74aef7889b4e/Revelion/0040740000000-0040744555555-0040740222222/0040740000000

message/0040740222222/d428a487-9932-471c-b701-74aef7889b4e/Revelion/0040740000000-0040740111111-0040744555555/0040740000000

## Structuri de date folosite

Pentru a implementa protocolul descris mai sus vom folosi următoarele structuri de date:

### Message

Message este clasa care conține informațiile despre fiecare mesaj.

public class Message

{

public DateTime TimeStamp { get; set; }

public Contact Sender { get; set; }

public List<Contact> Receivers { get; set;}

public string Topic { get; set; }

public string Text { get; set; }

public string TopicId { get; set; }

}

### Topic

Clasa Topic va retine toate mesajele trimise pe un anumit subiect de discuție.

public class Topic

{

public string Id { get; private set; }

public string Name { get; set; }

public IEnumerable<Contact> Participants { get; private set; };

public ObservableCollection<Message> Messages { get; private set; }

}

### Conversation

Clasa Conversation va conține toate subiectele de discuție intre unu sau mai mulți interlocutori.

public class Topic

{

public ObservableCollection<Topic> Topics { get; private set; }

public IEnumerable<Contact> Participants { get; private set; }

}

## Interfața cu Utilizatorul

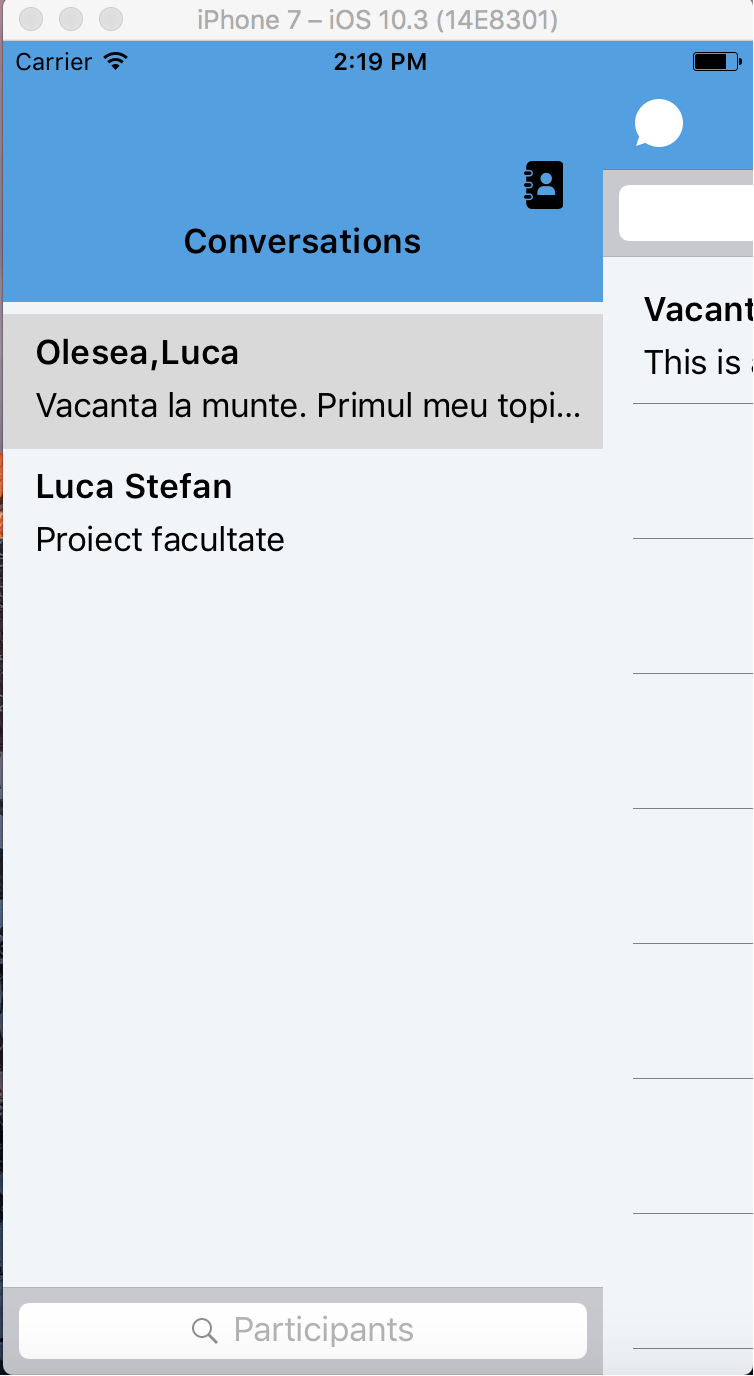
Interfața cu utilizatorul va fi implementata folosind Xamarin.Forms și va folosi controalele standard definite in acest framework.

Interfața va conține următoarele pagini:

* Pagina conversațiilor
* Pagina subiectelor de discuție
* Pagina contactelor
* Pagina de comunicare

### Pagina conversațiilor

Acesta pagina va conține toate conversațiile utilizatorilor grupate pe subiecte de discuție. Ea va fi implementata ca o pagina de tip drawer pentru simplitatea de accesare.



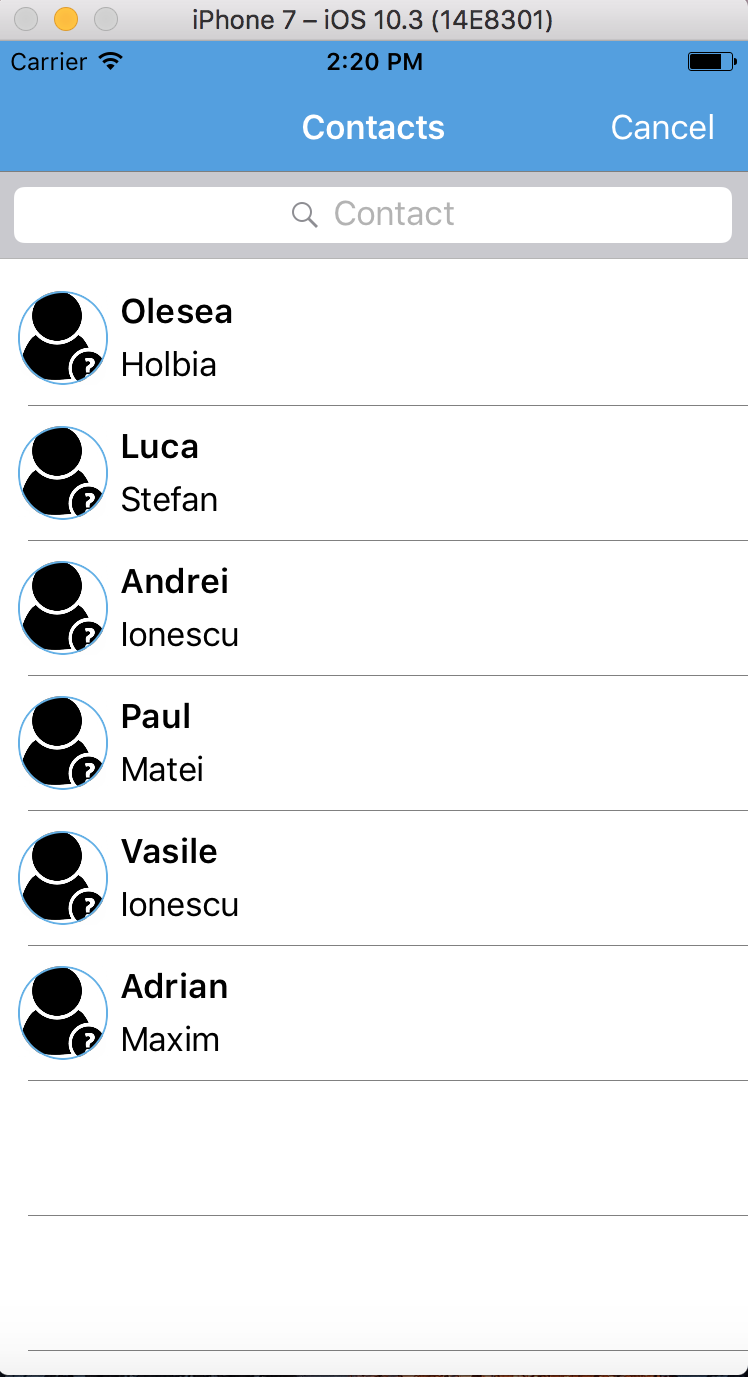
### Pagina subiectelor de discuție

Aceasta pagina va conține toate subiectele de discute dintre doua sau mai multe persoane. Folosind aceasta pagina putem șterge sau crea noi subiecte de discuție.



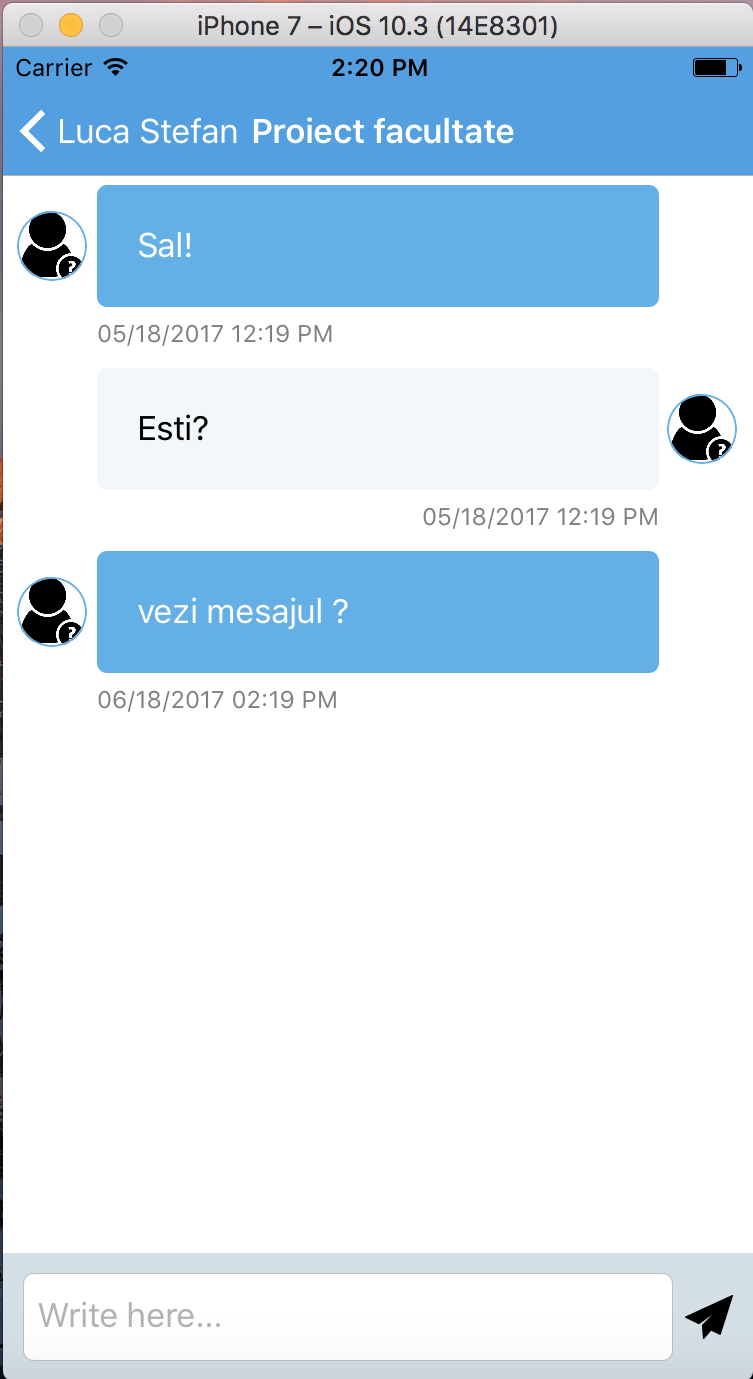
### Pagina contactelor

In pagina contactelor vom fi redirecționați atunci când vrem sa inițiam o noua conversație cu unul sau mai mulți interlocutori.



### Pagina de comunicare

Pagina de comunicare este efectiv pagina unde putem primi si trimite mesaje.



## Codul sursă

Codul sursa poate fi găsit pe Github

<https://github.com/ionutholbia/topichat>

## Probleme și dificultați întâlnite

Având in vedere că Xamarin și Xamarin.Forms este un framework relativ nou si încă în dezvoltare de multe ori exista probleme de compatibilitate între platforme. De exemplu un simplu buton poate avea marginile conturate pe o platforma si pe alta nu.

O alta problema sunt IDE-urile celor de la Microsoft. Exista multe probleme de compatibilitate intre Visual studio for Mac si tradiționalul Visual studio existent pe Windows. De aceea nu am reușit încă sa suport si platforma Windows. Am decis sa lucrez pe Mac si ca aplicația sa suporte la început doar iOS si Android.

Însă cea mai mare problema întâlnita a fost legata de librăria MQTT pentru client. Când am început proiectul nu exista o librărie care sa suporte proiectele de tip Android și iOS și a trebuit sa portez codul existent. După a apărut M2Mqtt Dotnet Core ca Nuget Package si am decis sa folosesc acest pachet.

# Testarea aplicației

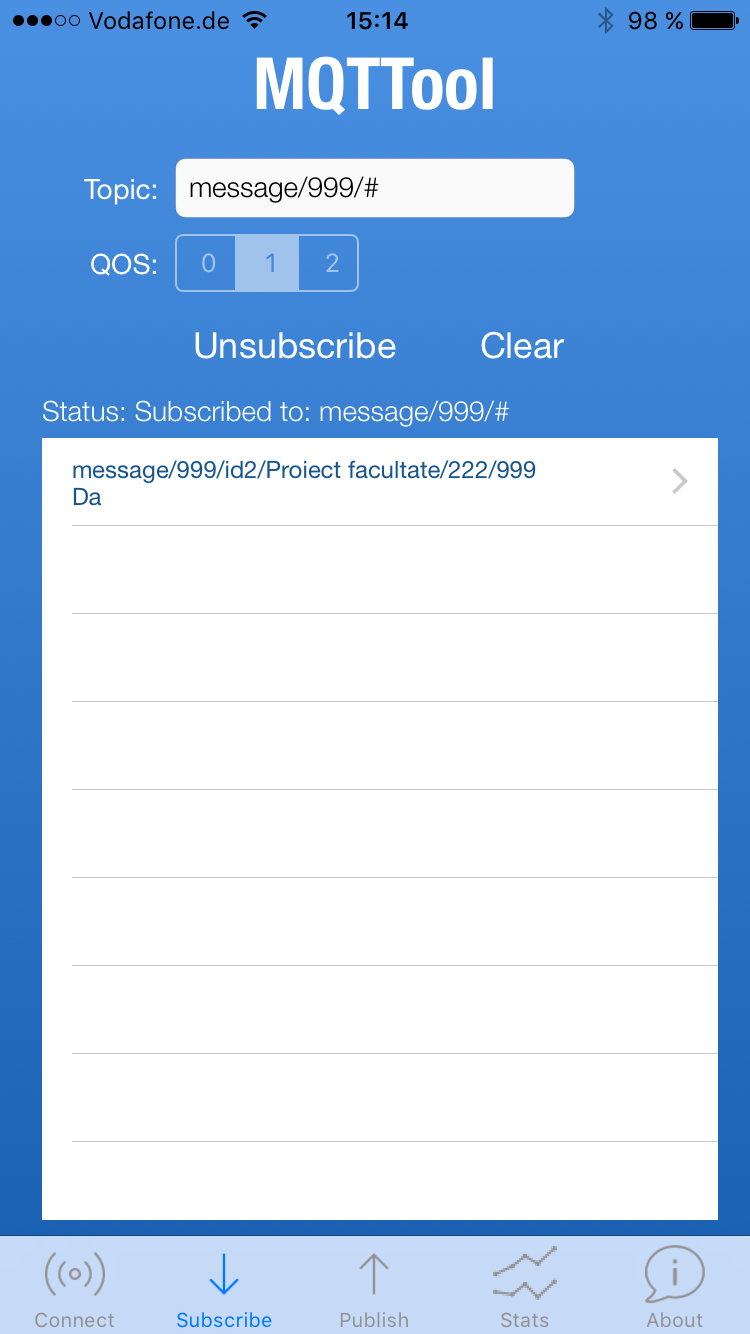
Testarea s-a realizat pe fiecare etapa a implementării. Vom defini următoarele etape ale testării:

* Testarea comunicării folosind broker-ul MQTT
* Testarea business logic-ului
* Testarea interfeței

## Testarea comunicarii

Acesta faza a fost executata inițial folosind o aplicație MQTT ajutătoare. Am luat în calcul scenariul in care trimit din codul meu un mesaj si este primit in aplicația ajutătoare și invers.

Aplicația ajutătoare se numește MQTTool si este disponibila pe Apple Store.



Folosind aceasta aplicație am reușit sa testez atât trimiterea de mesaje când si primirea unu mesaj.

## Testarea business logic

Acest lucru s-a realizat în mare prin implementarea de unit teste. În proiect gasim unit teste atât pentru testarea codării si decodării unui mesaj cat si pentru structurile de date care vor stoca aceste mesaje.

## Testarea interfeței

Am folosit emulatoarele care vin la pachet cu framework-ul Xamarin pentru testarea interfeței.

# Concluzii

Proiectul la prima vedere pare destul de simplist si ușor de implementat dar după o analiza mai aprofundată se dovedește a fi foarte complex.

Am întâlnit multe dificultăți în implementarea lui și sigur vor si fi altele daca voi reuși sa îl duc la stadiu de release. Acest proiect a venit ca o necesitatea de a mea în a-mi organiza mai bine conversațiile și probabil poate ajuta si alți utilizatori daca va fi lansat.

Pentru moment lipsesc câteva feature-uri pe care multe alte aplicații de comunicare le au. De asemenea se simte lipsa unui design modern și mai atractiv.

Un alt punct unde mai trebuie lucrat este utilizabilitatea. Cred ca la acest capitol mai trebuie lucrat pentru ca aplicația sa fie foarte intuitiva si ușor de folosit.

Planuri de viitor:

1. Implementarea unui backend care va manageriza Brokerii MQTT. Pentru moment exista un singur server MQTT, adresa lui fiind predefinită în produs.
2. Salvarea istoriei conversațiilor pe un serviciu cloud al utilizatorului. Pe motiv de protejare a datelor personale vrem ca toate conversatiile sa fie salvate criptat pe cloud-ul personal al utilizatorului (Google Drive, Dropbox, Microsoft OneDrive).

# Bibliografie

* Wikipedia
* MSDN
* Exemplele de aplicații găsite pe developer.xamarin.com
* [Prezentarea](https://www.youtube.com/watch?v=3w61d2hLnf0) lui Alex Corrado pentru evenimentul Xamarin Evolve 2016