|  |  |  |
| --- | --- | --- |
| 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 2](#_Toc485396762)

[Fundamentarea teoretică și documentarea bibliografică 3](#_Toc485396763)

[Aplicații Existente Pe Piața 3](#_Toc485396764)

[Facebook Messenger 4](#_Toc485396765)

[WhatsApp Messenger 5](#_Toc485396766)

[WeChat 6](#_Toc485396767)

[Alte Aplicați 6](#_Toc485396768)

[Specificațiile Aplicației 7](#_Toc485396769)

[Sisteme de operare suportate 7](#_Toc485396770)

[Device-uri suportate 7](#_Toc485396771)

[Protocolul de comunicare 7](#_Toc485396772)

[Idei pentru interfața cu utilizatorul 10](#_Toc485396773)

[Proiectarea aplicației 11](#_Toc485396774)

[Protocolul MQTT 11](#_Toc485396775)

[Istoria protocolului 11](#_Toc485396776)

[Cum functionează 11](#_Toc485396777)

[Metoda de comunicare publish/subscribe 12](#_Toc485396778)

[MQTT Client 13](#_Toc485396779)

[MQTT Broker 13](#_Toc485396780)

[Xamarin 14](#_Toc485396781)

[Utilizare 14](#_Toc485396782)

[Xamarin Forms 15](#_Toc485396783)

[Configurarea Proiectului 15](#_Toc485396784)

# Rezumatul lucrării

Topichat este o aplicație de chat pentru telefoane mobile bazată pe topicuri. Toate conversațiile utilizatorului vor fi grupate pe topicuri în aşa 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.

Pentru moment nu vom folosi un Backend pentru ca utilizatori vor putea sa folosească 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 device si nu undeva in Cloud. Acesta abordare este mult mai benefică utilizatorului deoarece nu apar probleme de privacy si securitate.

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. Persoanele care sunt preocupate de protezare 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 topic care s-a desfășurat mult in 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 topicuri, cum ar fi „Petrecere de revelion” si care pune mult accent pe securitate si protecția datelor personale.

## Aplicații Existente Pe Piața

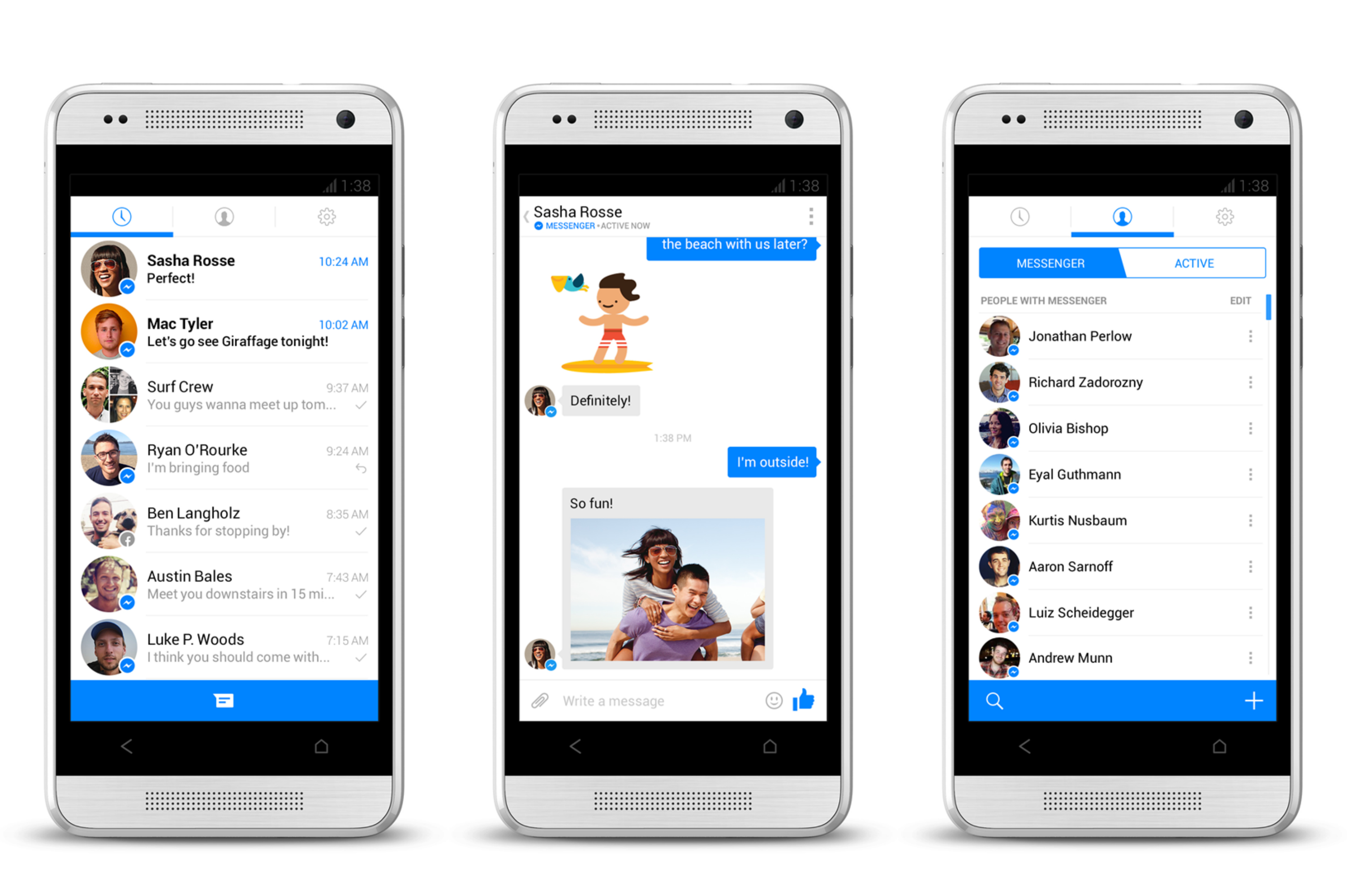
După cum am menționat in introducere acum pe piața este un număr foarte mare de aplicații destinate comunicării. Potrivit site-ului [Engadget.com](http://www.engadget.com) cele mai populare aplicații de comunicare sunt in ordinea următoare:

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.

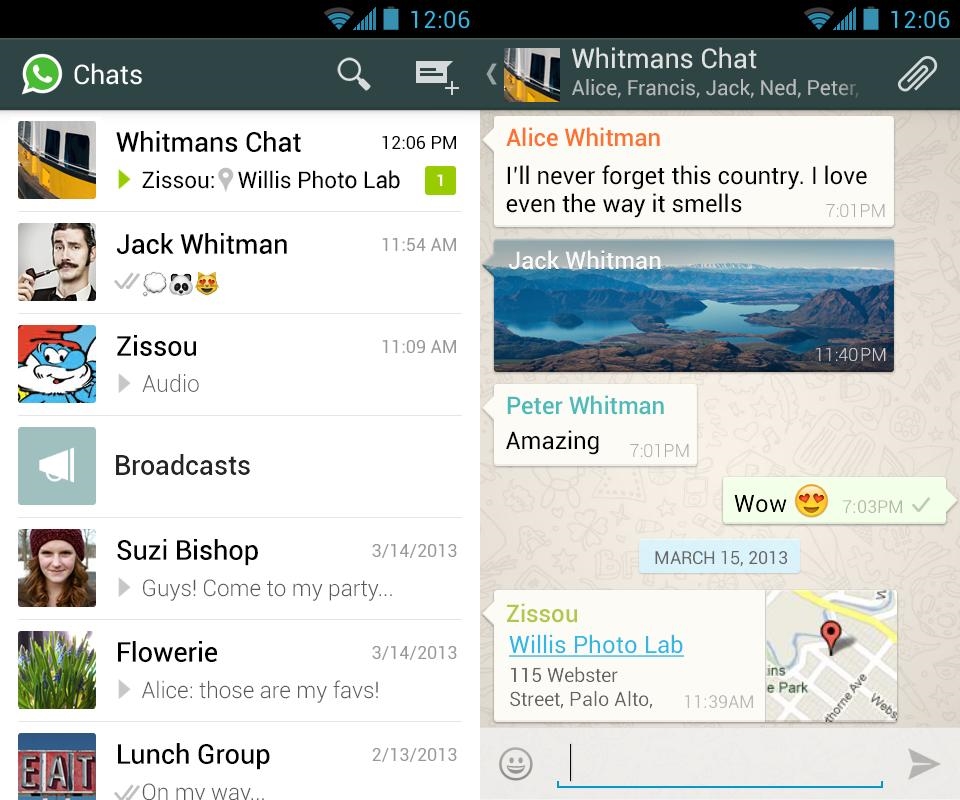


Fiind cea mai cunoscuta si utilizată aplicație de comunicare, utilizatorii care sunt mai preocupați de securitate si protejarea datelor personale au tendința de a evita aceasta aplicație.

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.



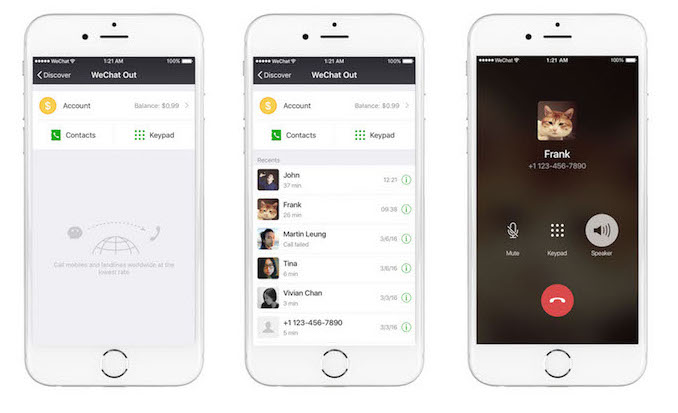
O mare problema a aceste aplicații era securitatea. Mesajele era trimise în clar, lucru care nu o recomanda utilizatorilor preocupați de securitate.

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 feature-uri și de aceea acum aplicația numără in jur de 1 miliard de utilizatori. De asemenea probleme 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, investi la bursă si multe altele. E foarte greu sa catalogam acesta aplicație ca o aplicație destinata comunicării.

### Alte Aplicați

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

## Specificațiile Aplicației

Similar aplicației WhatsApp, Topichat va fi numai o aplicației de comunicare. Utilizatorii vor putea sa comunice 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

Ne propunem sa suportam majoritatea device-urilor iOS începând cu iPhone 5 și cele mai populare device-uri Android. Pentru clienții Apple ne este mult mai ușor sa testam pentru ca device-urile sunt puține si bine cunoscute dar după cum știm pentru Android sunt mii de device-uri existente și de multe ori până și companiile mari de software au probleme în a suportat toate aceste device-uri.

### 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.

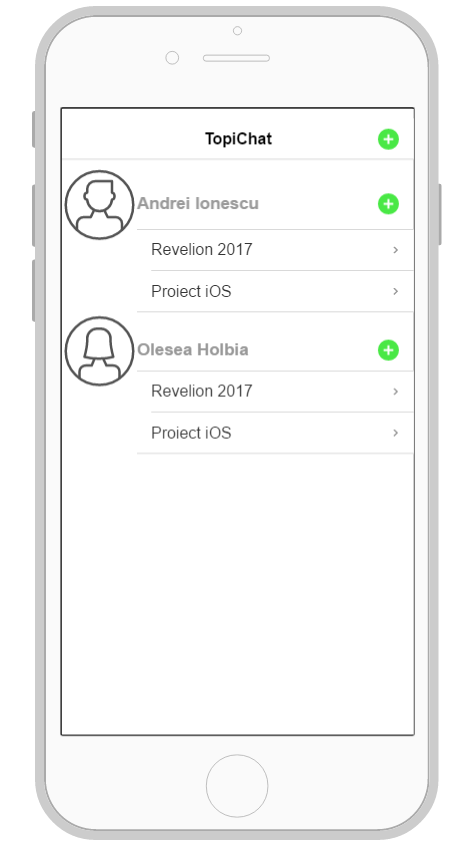
Dezavantaje:

* 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 topicuri. De aceea trebuie evidențiam acest lucru în interfața produsului.



După cum vedem in modelul de mai sus avem doua conversații, cu Olesea Holbia si Andrei Ionescu. Fiecare din aceste doua conversații are cate doua topicuri.

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.

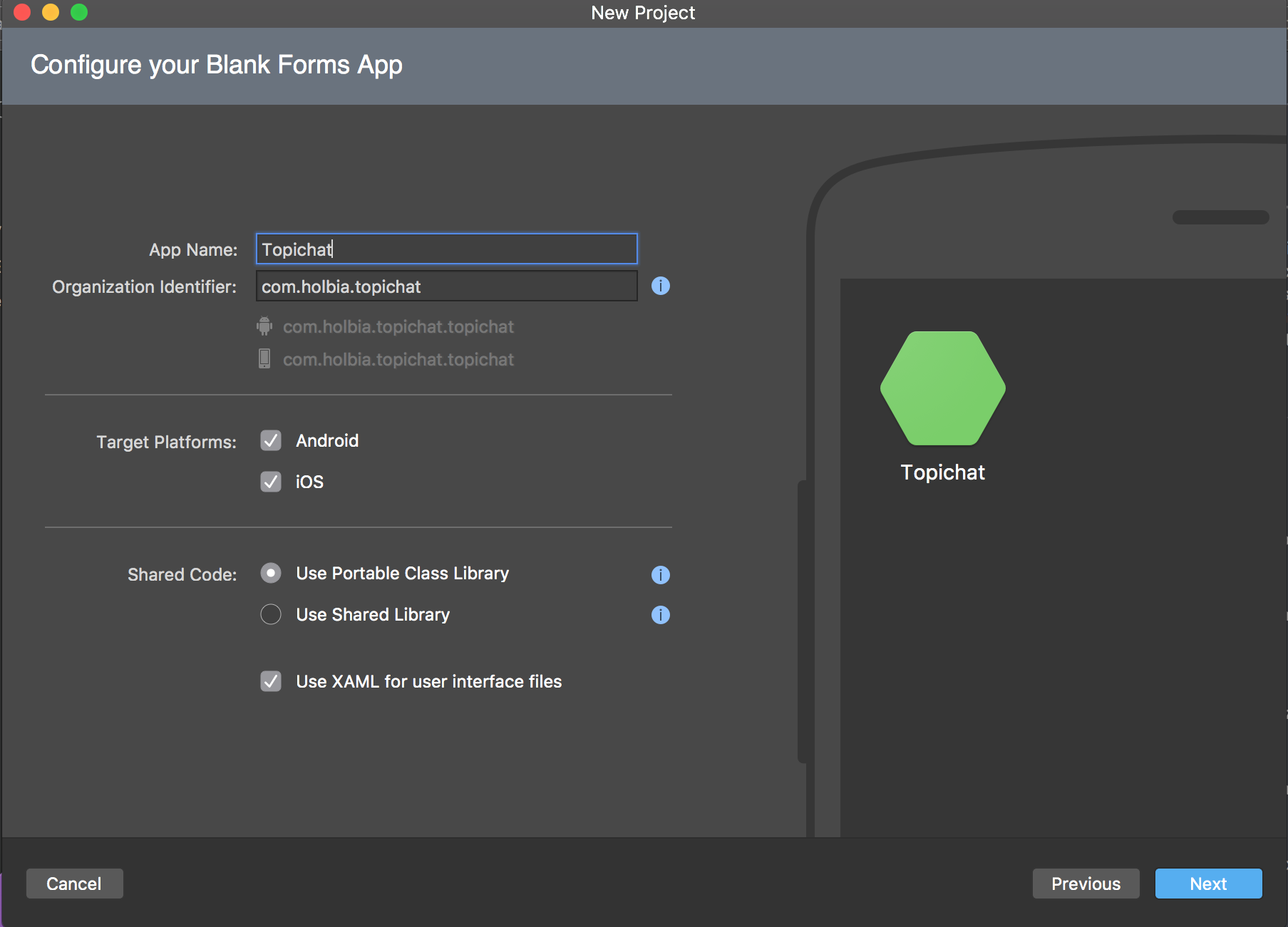
Xamarin Forms este o modalitate ideală de a implementa interfețe cross platform care nu necesită prea multe funcționalități specifice fiecărei platforme. Folosind aceasta abordare putem partaja in proporții de 90% din codul aplicației. De asemenea este mai confortabil de implementat interfața deoarece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

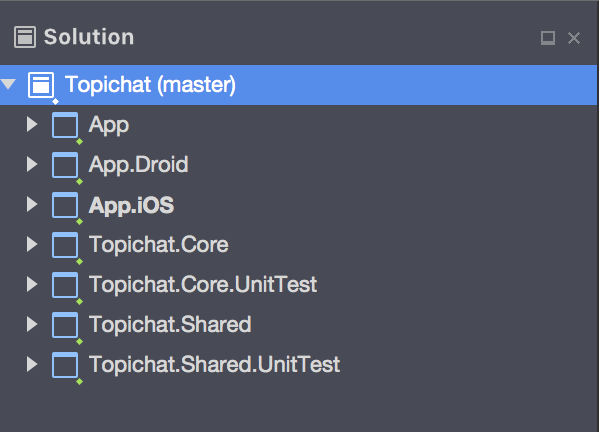
Pentru a crea un proiect cu o interfața cross platform vom selecta New Solution și de acolo vom alege Blank Forms App, adică o aplicație Xamarin Forms goală, fără nici o pagina. Paginile le vom crea noi mai târziu.



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.