Tertúlias

António Borba da Silva

Supervisors: Eng. Pedro Miguel Henriques Santos Félix

Project report – Beta version – within the scope of Project and Seminar

Computer Science and Computer Engineering

Summer semester 2015/2016

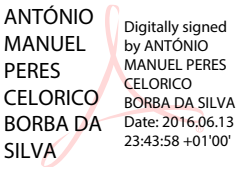
June 2016

Instituto Superior de Engenharia de Lisboa

Computer Science and Computer Engineering

Tertúlias

22908 António Manuel Peres Celorico Borba da Silva



Supervisor: Eng. Pedro Miguel Henriques Santos Félix

Project report – Beta version – within the scope of Project and Seminar

Computer Science and Computer Engineering

Summer semester 2015/2016

June 2016

Abstract

"A tertúlia is a social gathering with literary or artistic overtones, especially in Iberia or in Latin America. Tertúlia also means an informal meeting of people to talk about current affairs, arts, etc. The word is originally Spanish (borrowed by Catalan and Portuguese), but it has only moderate currency in English, used mainly in describing Latin cultural contexts."
(Wikipedia – the free encyclopedia. Available at: <https://en.wikipedia.org/wiki/Tertulia>. Accessed on: 2016/06/10)

This project is the implementation of a system for managing recurring events using an App developed for Android (KitKat) devices – with a twist for "Tertulias" events.

The motivation for addressing this project is twofold:

- On one side, and irrespective of the number of applications available in the market targeting activity scheduling, we haven't found one targeting this specific and important social and cultural habit of periodic gathering of groups of people, which has been cherished by communities for centuries. As such, we find that there must be a large number of citizens to whom this technology might be worth to use.
- On the other side, we have a great deal of interest in the domain of technology applications to enable social interaction and crowd mobilization, taking advantage of this recent context built around affordable mobile devices, data communications and cloud based persistence and notification services.

With the completion of this project we are taking for ourselves the experience of designing, implementing and making available a cloud based service aiming to scale according to usage demand. We achieve these goals by putting in practice the knowledge learned along the course, covering a substantial range of software areas, applied in a real world complex internet environment that includes:

- a mobile application made available in an app store,
- users being authenticated by independent a social service provider,
- synchronization with a cloud based server service via custom APIs,
- data persistence in a cloud hosted relational database and
- direct device notifications being pushed by a notification service.

For this we are making use of diverse technologies including the Java based Android development framework, NodeJS javascript, Microsof TSQL, OAuth 2.0 based authentication delegation, OpenId Connect user identification and push notifications service. The final result is a working

client app for the Android, with a documented API allowing development for other mobile or browser based platforms or for integration with third party applications.

Keywords: mobile; social; notifications.

Table of Contents

Images list

Tables List

1. Introduction

"A tertúlia is a social gathering with literary or artistic overtones, especially in Iberia or in Latin America. Tertúlia also means an informal meeting of people to talk about current affairs, arts, etc. The word is originally Spanish (borrowed by Catalan and Portuguese), but it has only moderate currency in English, used mainly in describing Latin cultural contexts."

(Wikipedia – the free encyclopedia. Available at: <https://en.wikipedia.org/wiki/Tertulia>. Accessed on: 2016/06/10)

This project is the implementation of a system for managing recurring events using an App developed for Android (KitKat) devices – with a twist for "Tertulias" events.

The motivation for addressing this project is twofold:

- On one side, and irrespective of the number of applications available in the market targeting activity scheduling, we haven't found one targeting this specific and important social and cultural habit of periodic gathering of groups of people, which has been cherished by communities for centuries. As such, we find that there must be a large number of citizens to whom this technology might be worth to use.
- On the other side, we have a great deal of interest in the domain of technology applications to enable social interaction and crowd mobilization, taking advantage of this recent context built around affordable mobile devices, data communications and cloud based persistence and notification services.

With the completion of this project we are taking for ourselves the experience of designing, implementing and making available a cloud based service aiming to scale according to usage demand.

We achieve these goals by putting in practice the knowledge learned along the course, covering a substantial range of software areas, applied in a real world complex internet environment that includes:

- a mobile application made available in an app store,
- users being authenticated by independent a social service provider,
- synchronization with a cloud based server service via custom APIs,
- data persistence in a cloud hosted relational database and
- direct device notifications being pushed by a notification service.

For this we are making use of diverse technologies including the Java based Android development framework, NodeJS javascript, Microsof TSQL, OAuth 2.0 based authentication delegation, OpenId Connect user identification and push notifications service.

The final result is a working client app for the Android, with a documented API allowing development for other mobile or browser based platforms or for integration with third party applications.

Apart from the fact that using a cloud based service for hosting the application server in a protected and scalable environment, it is worth to mention that the selected backend environment (Microsoft Azure) can make available a number of additional analytic services will be used to monitor application usage and resources consumption, in case app market adoption takes off.

1.1. System building blocks

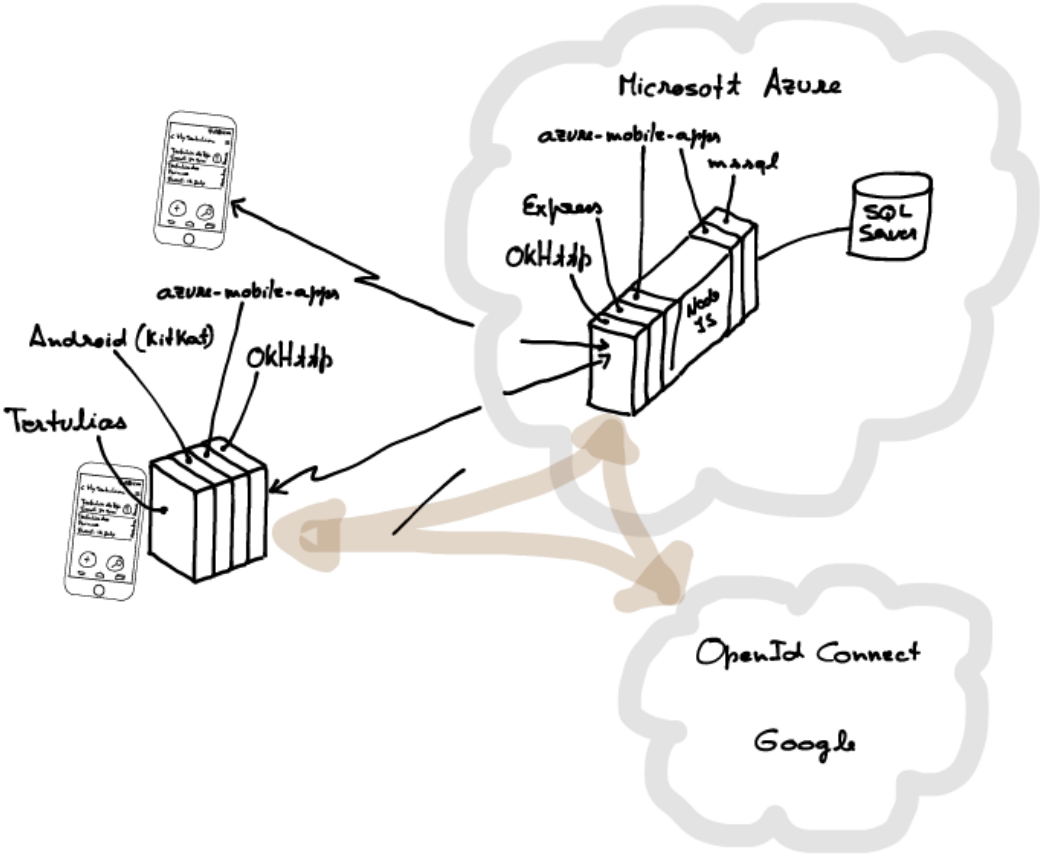


Figure 1: App Building blocks

The generic building blocks to grasp the system architecture are presented in figure 1 and it works as follows.

Once authenticated at Google's authentication provider (Facebook and Twitter are also supported, although not currently configured), the client devices running the Tertulias app accesses the shared data via an API provided by an application server running on a virtual machine at Microsoft Azure cloud platform. The Client App uses the "Azure Mobile Apps" SDK for Android enabling

a simplified application server interface. The application server, which is coded in NodeJS's Javascript, uses the "Azure Mobile Apps" module, which provides the API endpoint interface for Clients access, configured for accepting only authenticated requests. The "Azure Mobile Apps" module also provides user identification – each user is represented by a unique token. Towards the database, it provides all the communication protocol with the SQL Server managing the Tertulias database, allowing the use of SQL prepared statements with bound parameters.
Currently, the Beta version doesn't support local persistency in the client devices, but that functionality will be present in the final version, together with a synchronization mechanism which shall be triggered by push notifications received by the App running in the device, optimizing in this way the resources usage, both in the client and in the server sides.

1.2. Document structure

This version of the document represents work in progress aiming for the final version, so, the structure presented in target for that final version, although in the text of the chapter a lot of information is still missing.
The graphics presented are also "Beta", i.e., they might require some tweaks for the final version, and its resolution might require improvement in some cases.

This document's goal is to present the Tertulias project, scoping details of the agreed goals, functionality, implementation and support infrastructure.
The Tertulia project targets the deployment of a cloud based service where any person can organize and / or collaborate in the preparation of periodic meetings called Tertulias.
The way this document is organized is from the more general aspects to the most particular ones, aiming to drive focus to each specific issue, preserving context and trying to avoid mixing subjects.
In chapter 2. (Problem Scope) the boundaries of the problems we are trying to solve are addressed, stating what shall be the outcome of development and what could possibly be the following evolution.
In Chapter 3. (User Interaction) we present an overview of the user interaction with the app, including a graph representing the interaction flows and the wireframe screens the implement those flows.
In Chapter 4. (System Building Blocks and Core Decisions) the core decisions that shaped some of the determining aspect of the project, are stated. This chapter also includes the report of key actions that were taken, leading to the final system design.
Right after, chapter 5. (System Design) addresses the system detailed design, by blocks and in terms of responsibility and functionality. The development environment setup required for each block is also addressed here.

In chapter 6. (System development) we address all aspects related with the development of the system for the 3 major areas of concern: The Client app in Android, the server API in Node.JS and the database in SQL Server.

Finally, in chapter 7. (Conclusions) we present the conclusions we took from the development of this project, including the lessons learned and the identification of possible areas deserving further development.

2. Problem Scope

To give a general idea of the functionality the project needs to cover, picture the following scenario:

- A group of friends meets regularly – on the third Thursday of every month at "Restaurante Cave Real", in Lisbon, for lunch – This is a *private Tertulia* with a restricted set of *Members*, with a *default* meeting *location* and with an associated *schedule*.
- Of course, in August they all go on holidays, so they will skip this one.
- Every time one of them needs to make a reservation in advance – This is the *event* preparation.
- From time to time one of them celebrates something (e.g. birthday) and the restaurant allows them to bring in a cake and a couple of Champagne bottles; This requires a *contribution list* for that event that needs to be setup as part of the event preparation.
- The list of contributions might be built Ad Hoc or loaded from preset groups of goods; This groups are available as *templates* of *items* to be added to the tertúlia event.
- Once in a while, someone suggest a new restaurant and they all decide to try it. These suggestions are broadcasted to all members as *notifications*. As with every other update, once the new location is updated all members will be notified and these messages will be on the user's Inbox until the user checks it out.
- … and so on …

Given this scenario, a couple we figured a couple of building blocks that stand out:

- An application server: It's responsibilities are to accept and persist all application input and queries from the clients, preserve data privacy, and notify clients of status updates, thus implementing a convenient API which delivers the corresponding functionality.
- A data repository: Probably in the form of a *queryable* database that will be used by the application server to persist application data.
- A client application: It's main responsibilities are to provide an adequate user experience to the user and to dialog with the application server in order to maintain the user updated on all tertúlias he participates in, and to reflect the user updates in respect to each of these tertúlias.
- An authentication delegation service provider: It's responsibilities are to ensure that the person using the app is who he says he is. Additionally, this service must also provide user identification for the authorized user. This is a key feature for the application to be able to associate the user with he's data, irrespective of the device he is using. This means that one login in the authentication delegation service provider will always result in the same application token that represents the user. Not mandatory, but helpful, the provider

could supply some user information data such as the name, email and picture link to be used to provide defaults for the user profile edition.

With the above scenario in mind, there are two analyses that helps us scoping the problem:

- One is a review of what people use today to tackle this situation;
- The other is to align user stories to grasp what would be the key functionality a user would cherish most in our application.

2.1. Competitive Analysis

There are a number of different solutions available that can address the problem of managing these situations and in order to capture their pluses and the minuses is useful to make a competitive analysis – or a market survey – looking for functionality or characteristics among the software applications available today that we find people use for managing this scenario.

This analysis is presented in "Annex 1 – Competitive Analysis" and resulted in a SWOT matrix and a summary of the possible effects of the usage of both Tertulias app and the app in analysis by a user.

2.2. User Stories

User Stories are a way of defining the project functional requirements from a user standpoint – At the end of the day, satisfying customer's expectations is key for a successful project.

So, for the current project we collected a number of *User Stories* that seem consensual for the base functionality this project should provide. The sources for the User Stories were:

- Our own base ideas on the project functionality;
- Potential application users – friends and family with whom we meet regularly in a Tertulias way;
- The review of the project supervisor.

This list is in "Annex 2 - User Stories".

3. User Interaction

In order to support the design of the user interaction flow as wireframe screen representations, we have built a user interaction graph representing the core states and transitions. The graph is presented in figure 2.

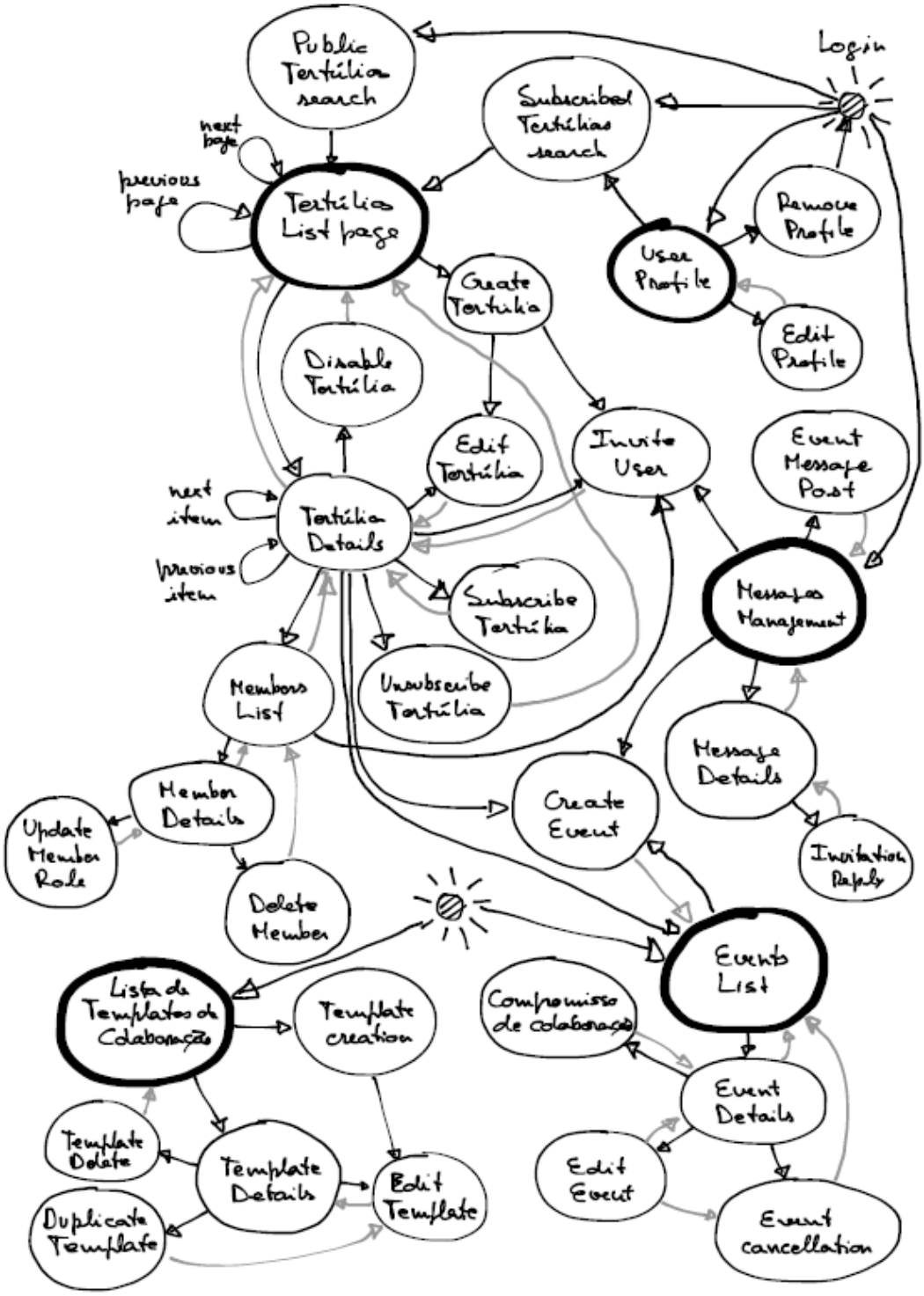


Figure 2: Interaction graph

In order to be able to use the application the user must be authenticated by a reliable authentication service provider – at this time only Google is supported – so, leaving that aside, the starting point for the app is the list of subscribed tertúlias ("Tertúlias List Page" in the graph), described in detail ahead. This will be empty in the beginning, but will reference the user subscriptions as it happens. There are three ways for a user to become a subscriber of a Tertulia:

- When a user creates a Tertulia, where he will be assigned a "Owner" role;
- When a user accepts an invitation to participate in a private Tertulia, where he becames a "Member" of that Tertulia;
- When a user subscribes a public Tertulia, where the assigned role in the Tertulia will be "Member".

For each Tertulia item in the list the user can access its details ("Tertúlia Details" in the graph). From the Tertulia details, the user can edit the Tertulia or setup a new Tertulia event – if he has adequate permissions (roles Owner and Manager), unsubscribe the Tertulia – in case he is not the owner, or manage the Members list, including inviting users to a private Tertulia.

The event setup interaction ("Create Event" in the graph) it is also worth mentioning for its importance. In the event setup, the user specifies the event data and location, and can specify a list of goods for the members to bring to the event. In order to ease the repetitive process of setting up events, the user can define templates, preset with a selection of items each with a default quantity, and then import some of these templates into the event and adjust the necessary quantities.

Another interaction worth mentioning is the Tertulia broadcast notifications; The users can post notifications to the Tertulia that will be readable by any member. Those notifications are tagged with a category and the users can mute the categories they are not interested in.

Back to the main point of the interaction – the list of subscribed tertúlias. This is the landing screen upon launching the application and authenticating. Additionally to the interaction referred above, this is where the user can create a new Tertulia or search for public Tertulias to subscribe. In the screen of the device the user will be presented with a summary of all the Tertulias he is subscribed, and for each Tertulia listed, information on the next scheduled event, unread messages and the user's role in that Tertulia. As an example, the wireframe layout of the screen of this entry point is presented in figure 3.

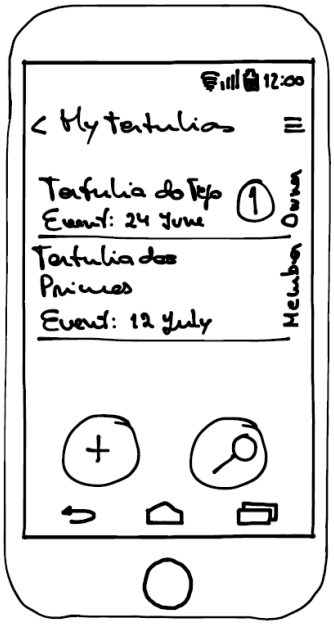


Figure 3: Layout of the App main screen

The complete set of wireframe screens for the implementation of the interaction graph is presented in Annex 3 - User Interaction, for the readers convenience.

4. System Building Blocks and Core Decisions

In this chapter we address the system building blocks and the most relevant decisions regarding the system's architecture and the technologies we in place to implement it.

The system major blocks are the ones pictured in figure 4.

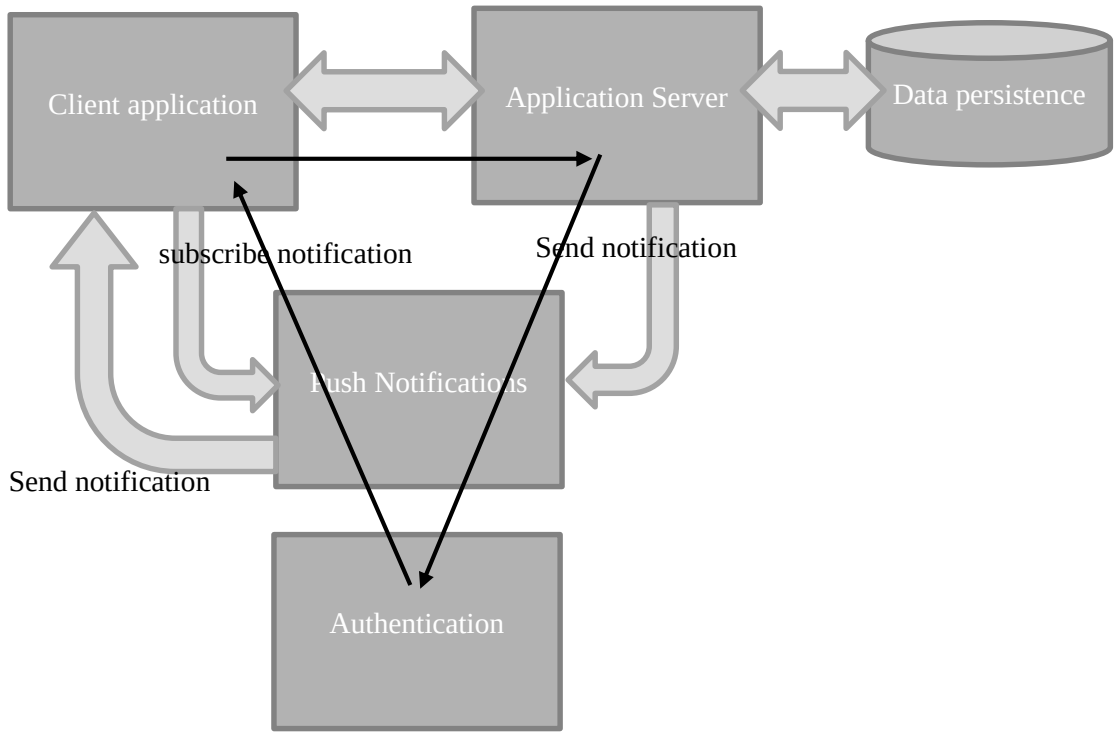


Figure 4: Service main building blocks

These are:

- A Client Application, providing a convenient user experience, maintaining offline status, synchronizing data and processing notifications from the application server.
- An Application Server, holding application status, ensuring data privacy and data persistence.
- A Data persistence Server, holding application server data.
- A Push Notifications service, that deliver Application Server notifications to Client Devices subscribing it.
- An Authentication service, that provides users authentication.

4.1. Selection of the application server platform

One of the first decisions taken was to define criteria to select a server platform, perform a market evaluation in order to build a short list of potential suppliers and apply the said criteria to select a platform.

The starting point is that it should be a full cloud base service platform in order to avoid time spending and risks associated with setting up and manage all required components of the platform[1]. The details of the evaluation can be consulted on "Annex 4 – Cloud platform selection". For what matters, we selected **Microsoft Azure** to support the application server. The evaluation summary is replicated in table 1.

Table 1: Short-list candidates scoring

Requirement	**W**	**Azure**	**StrongLoop**	**Appcelerator**	**AWS**	**FeedHenry**	**FireBase**
Low entry level costs	8	1	1	1	1	1	1
Cloud Free-tier	5	1	1	1	1	1	1
Server Side Code	8	1	1	1	1	1	0
GeoSpacial services	1	1	1	1	1	0	1
Management console	5	1	1	1	1	1	1
SDKs	4	1	1	1	1	1	1
Push notifications / sync services	8	1	1	1	1	1	0
Cloud & Hosted & OnPermises	3	2	2	1	2	3	1
Setup easiness	3	5	3	5	1	2	5
Overall personal impression	5	5	5	3	4	3	3
Score	**-**	**85**	**79**	**72**	**68**	**68**	**56**

4.2. Client platform selection

To take the obvious out of the way, it is easy to guess that mobile access is a must, and looking back to the technologies we have studied, the client side would either be an Android app or a Mobile Web site. The criteria we used for the decision is detailed in "Annex 5 – Client platform selection" and we opted to go for an Android client development for the current project scope, leaving for future development and iOS and a javascript based version.

As of the date of this report (June 2016), and according to Google Dashboards[2], versions starting from KitKat account for 77,1% of the devices accessing the Market / Play Store, as presented in table 2. This justified our decision to select Android API 19 as the target for our development.

Table 2: Android Versions share

API	**Codename**	**Versions**	**Share**
8	Froyo	2.2 - 2.2.3	0.1%
10	Gingerbread	2.3.3 - 2.3.7	2.0%
15	Ice Cream Sandwich	4.0.3 - 4.0.4	1.9%

[1] Virtual machine, OS, application server, database, software deployment, security management, DNS setup, push notifications or interface to a push notifications service, interface to authentication service providers, etc.

[2] https://developer.android.com/about/dashboards/

API	Codename	Versions	Share
16	Jelly Bean	4.1	6.8%
17	Jelly Bean	4.2	9.4%
18	Jelly Bean	4.3	2.7%
19	KitKat	4.4	31.6%
21	Lollipop	5.0	15.4%
22	Lollipop	5.1	20.0%
23	Marshmallow	6.0	10.1%

4.3. Client authentication

As stated above, only authenticated users shall be able to use the service. This authentication shall be via OAuth 2.0 authentication service providers, such as Google, Facebook or Twitter.

The Azure platform and SDK supports both the server authentication flow and client authentication flow.

Using the server authentication flow, the SDK takes care of all the authentication flow with the server and the authentication provider and it ends, either successfully with a valid token, or it fails with an Http 401 status error.

Using the client flow, the client obtains the token directly from the authentication provider and delivers it to an SDK class instance that takes care of validating it against the authentication provider and to grant client access.

At this stage, we have chosen to go with the first approach – server authentication flow – as the focus is to have the development progressing to deliver the required functionality.

4.4. Client offline usage

In the Android it makes sense to keep replication of some server data to avoid the dependency of network availability and latency and save on data communications and battery usage.

For that and given we are using Microsoft Azure SDK, we can either use the SDK support for offline usage or the Android's content provider - both use the internal Android's sqLite database to persist the data. We shall use an Android's content provider simply because we know what to expect from it, it's a tested and reliable environment with clear advantages for the UI update.

The risk here is on the usage of a syncAdapter because of the validity of the session token and the necessity to renew. Anyway we couldn't find any Microsoft documentation on how do they address this problem.

The general idea is to use push notifications – instead of server pooling – for launching data synchronization and this may be a way to overcame this constraint.

On this beta version we are not using local persistence to support offline usage.

4.5. Push notifications

Azure platform supports push notifications and we shall use it later to notify tertúlia member's devices of the need to synchronize data and alert device user of status changes.

We are leaving this aspect for later when we have the support for offline usage in place.

5. System Design

In this chapter we present the material produced to support the solution design.

5.1. Dependencies

An aspect of concern in the design of the application architecture was the dependencies between any two blocks of the system in respect to upgrades.

The identified blocks are:

- Application server
- Client App
- Database
- Authentication provider

5.1.1. Dependency of the Authentication provider

In respect to the authentication provider, as it is an external entity and the interaction with it is via a Microsoft library, there is not much that can be done. Anyway, as a service provider, Google, Facebook, Twitter or Microsoft have a too large customer base from such a large range of applications using authentication services that we can't identify a major or short term risk (as long as OAuth 2.0 is supported).

5.1.2. Database vs Application server

The database is only visible from the application server, therefore this is the only system that needs to be updated synchronously. As both systems are under our control we don't identify an unmanageable dependency risk here.

5.1.3. Client App vs Application server

There is a severe dependency between the client App and the application server. If the application server API needs to be changed, all installed client Apps need to change immediately to keep working.

In order to minimized this problem, we are using hypermedia to have the client driven by the server API.

5.2. Architecture

The blocks referred above fit in the blocks of figure 4 as presented in figure 5 which includes the development environment.

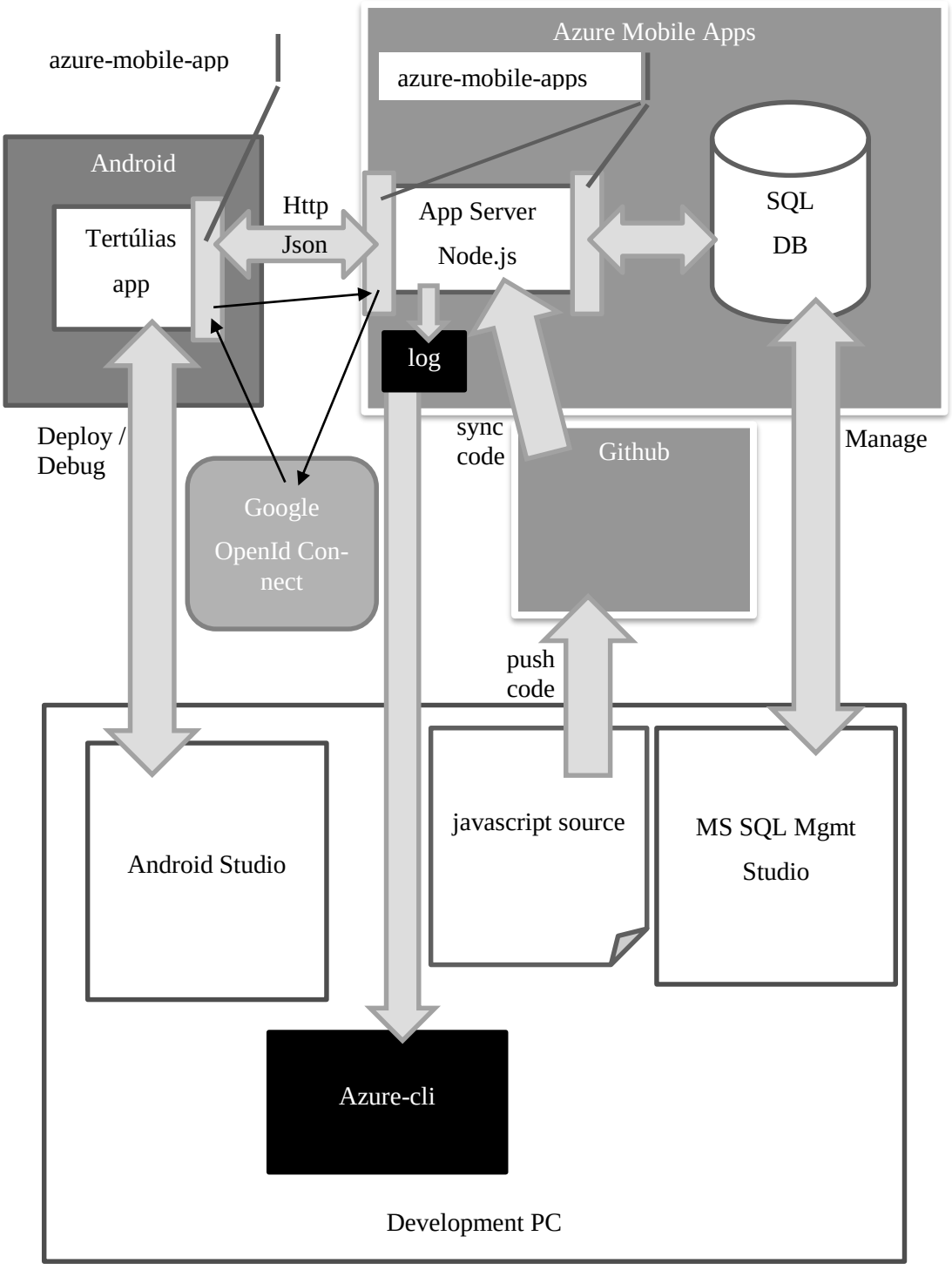


Figure 5: System building blocks

5.3. Application Server

The application server is developed in javascript for Node.js V8 engine, and uses Azure's "azure-mobile-apps" module to query the database, to handle the custom API routing behavior code and to maintain the user's authentication status with the authentication service provider.
The server application uses the "express" web framework and express.Router to create mountable route handlers to support the complex routing the application will be using.
We defined that the data exchanged with client applications shall be formatted as Json.

5.4. Android Client

The Android app uses the "azure-mobile-android" SDK library to interface with the custom API on the application server. The interface is done using the method "invokepi()" of "MobileServiceClient.class".
Additionally, Google's gson library is used for Json conversion.
Client authentication is as presented in point 4.3 above.

5.5. Development Environment

The development environment is as follows:

- Android development:
    - Android Studio is used for the Android app development.
    - Code versions are preserved in Github account (1).
    - While in development phase, code is uploaded directly to the device for testing.
    - Debugging is included in Android Studio, using Android's adb.
- Server API development:
    - Code development done locally using a text editor (Sublime Text).
    - A second Github account (2) is necessary due to lack of the privileges in the first Github account (1) required to setup the Continuous Development mode on the azure platform. In this way, every time new code version is pushed to the Github account, it is automatically uploaded by azure and the application server is re-started.
    - Application Server console log streaming is accessed by using azure-cli application on a local console.
- Server database:
    - Microsoft SQL Server 2014 Management Studio is used to manage remotely the Azure MS-SQL Database.

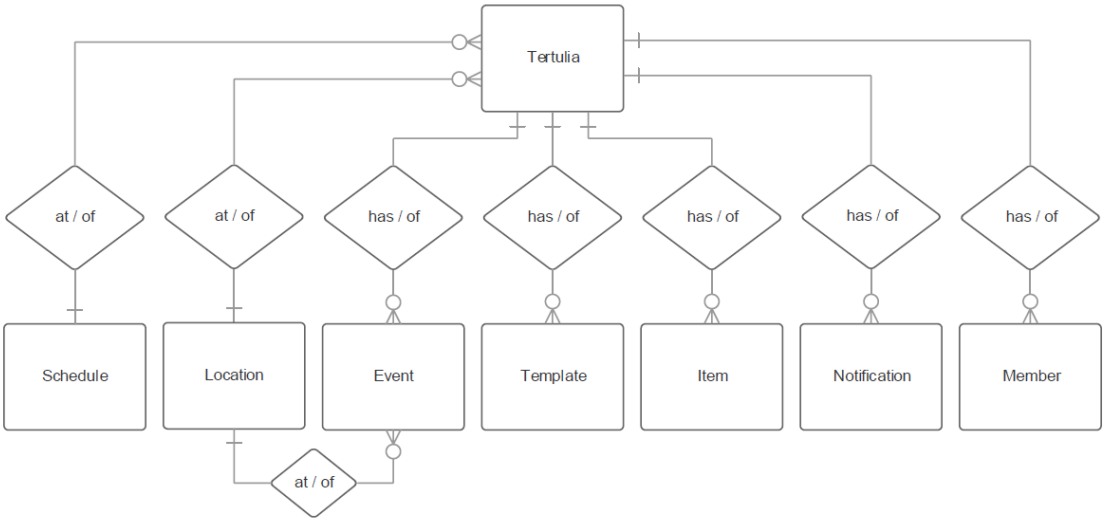
- Azure platform
    - Tertulias project was setup in a virtual machine in the West Europe datacenter in a small[3] instance size.
    - Application server implemented in Node.js v4.2.3.
    - Data persistence implemented in a MS-SQL database implementing Extended Transact-SQL compatibility / API version 12 and running in the same instance.

5.6. Database

5.6.1. ER Model

The full ER model that supports the application is presented in "Annex 5 – Data persistence: ER Model".

The Entity in the core is the Tertulia.



5.6.2. Physical data model

TODO

5.7. Application Server API

To be completed in the final version.

5.7.1. Endpoints structure

To be completed in the final version.

5.8. Client Domain model

To be completed in the final version.

[3] CPU: 1 core, RAM: 750Mb, HD: 20Gb, OS: Windows Server 2012 R2

6. System development

6.1. Tests

To be completed in the final version.

6.1.1. SQL Server

To be completed in the final version.

6.1.2. Application server

To be completed in the final version.

6.1.2.1. Mocha

To be completed in the final version.

6.1.3. Android

To be completed in the final version.

6.2. Database

To be completed in the final version.

6.3. Application Server

To be completed in the final version.

6.3.1. User authentication validation

To be completed in the final version.

6.3.2. Data serialization

To be completed in the final version.

6.3.3. Push Notifications

To be completed in the final version.

6.4. Android Client App

To be completed in the final version.

6.4.1. Android version

To be completed in the final version.

6.4.2. External libraries

To be completed in the final version.

6.4.3. Asynchronous programming

To be completed in the final version.

6.4.3.1. Microsoft SDK – Google Guava

To be completed in the final version.

6.4.3.2. AsyncTask

To be completed in the final version.

6.4.4. UI

6.4.5. Screen rotation

To be completed in the final version.

6.4.5.1. Fragments usage

To be completed in the final version.

6.4.6. Synchronization

To be completed in the final version.

6.4.6.1. Handlers usage

To be completed in the final version.

6.4.6.2. Local data persistence

To be completed in the final version.

6.4.6.3. Data Synchronization

To be completed in the final version.

6.4.6.4. Notifications and Broadcast Receivers

To be completed in the final version.

6.5. Services integration

To be completed in the final version.

6.5.1. Google Maps

To be completed in the final version.

7. Conclusions

To be completed in the final version.

7.1. Lessons Learned

To be completed in the final version.

7.2. Possible improvements

To be completed in the final version.

8. Annexes

8.1. Annex 1 – Competitive Analysis

Table 3: Applications analysis / strengths + weaknesses

Application	Strengths	Weaknesses
Email	- Available - Universal - Agnostic to technology - Mobility - Push notification	- Messages flood - No push-button reply - Hard to build enhanced feats - Hard to manage large user groups - Mail black lists - Hard to manage multiple tertúlias - No tertúlia management
WhatsUp	- Large user base - User awareness - User trust - Every platform - Ecosystem - Presence - Notifications - Groups setup - Well defined privacy policies - Easy to use	- No repetitive scheduling - No good for public tertúlias - No tertúlia management
Msft Link	- IT enabled on businesses - Presence - Flexibility - Hype - API	- Setup of private tertúlias - Specific context of tertúlias
Forums (Google Groups, etc.)	- Sophisticated user management	- Specific tuning for tertúlias (mainly UI)
Facebook	- Large user base - User awareness - User trust - Every platform - Ecosystem - Presence - Notifications - Well defined public/privacy policies - Strong event management - API	- Complex to tune for tertúlias - Complex management for multiple tertúlias - Generic app
Slack	- IT enabled on businesses - Every platform - Presence - Flexibility - Hype - API - Well defined public/privacy policies - Integration public/privacy	- Generic issues - Tech skills required to tune to tertúlias - Specific tuning for tertúlias (mainly UI) - Geek stuff

Table 4: Applications analysis / opportunities + threads

Application	**Opportunities**[4]	**Threads**[5]
Email	- Can be used for notification	- Users fall back to email in any case of dissatisfaction
WhatsUp	- Can be used for notification	- Users fall back to whatsup in any case of dissatisfaction
Msft Link	N/A	N/A
Forums (Google Groups, etc.)	- Post integration to capture users	N/A
Facebook	- Can be used for notification - Can be used for authentication delegation - Can be used for new users discovery	- Users fall back to fb in any case of dissatisfaction
Slack	- Can be used as extension	- Some users might prefer lock in

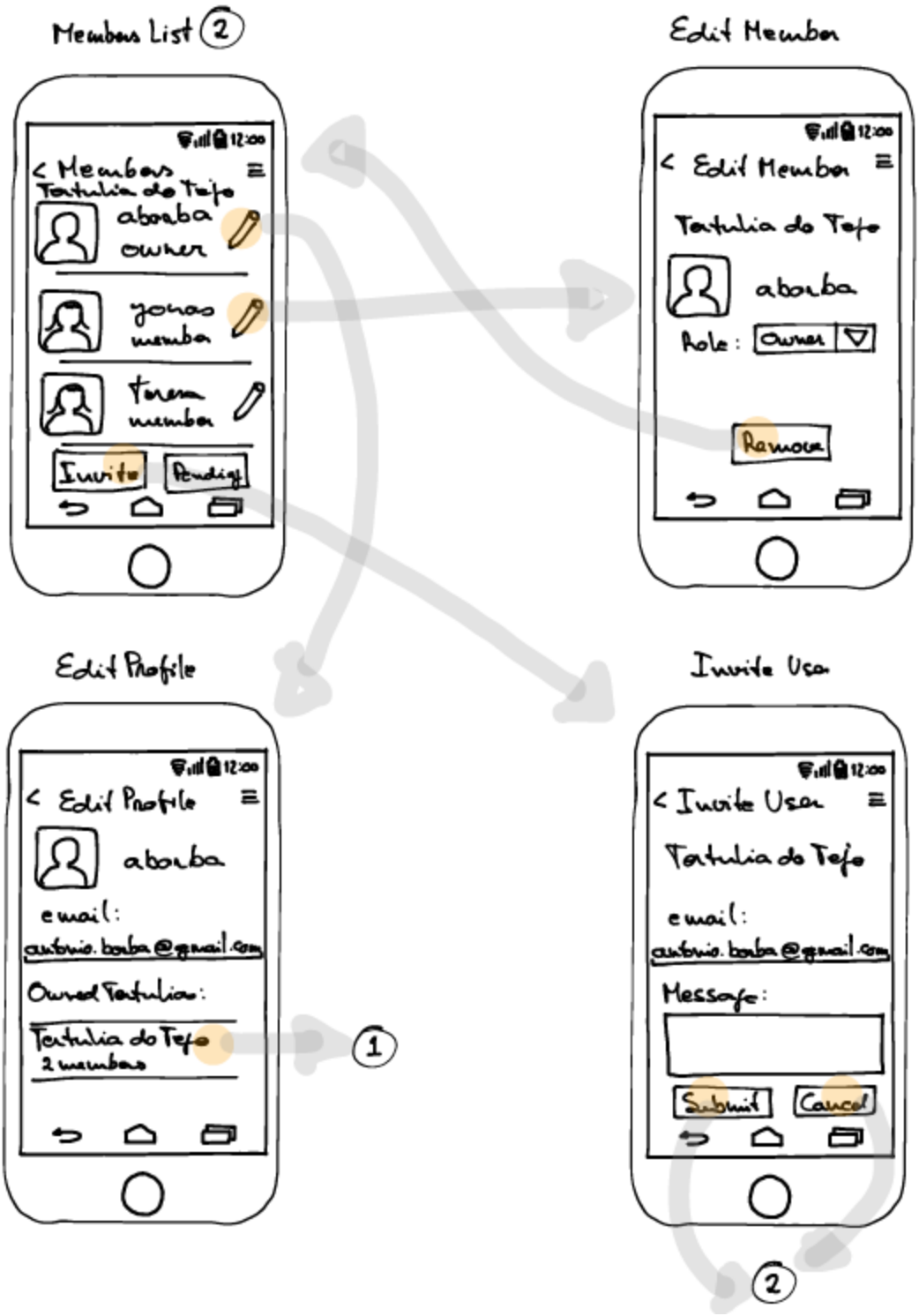
[4] Opportunities leveraged by the use of this app in the context of our own app.
[5] Threads originated in case we make use of this app and our users become unsatisfied with our own app.

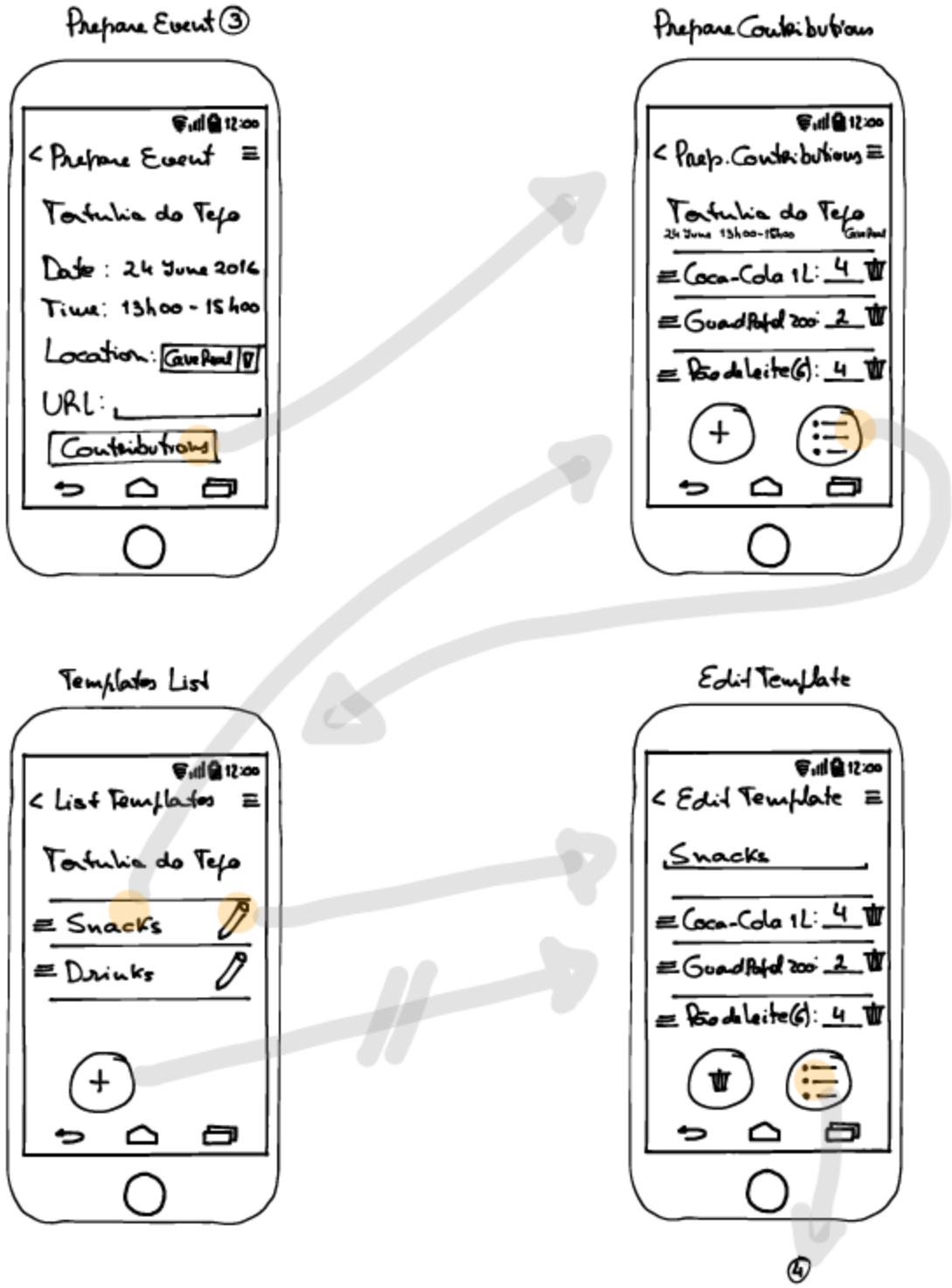
8.2. Annex 2 - User Stories

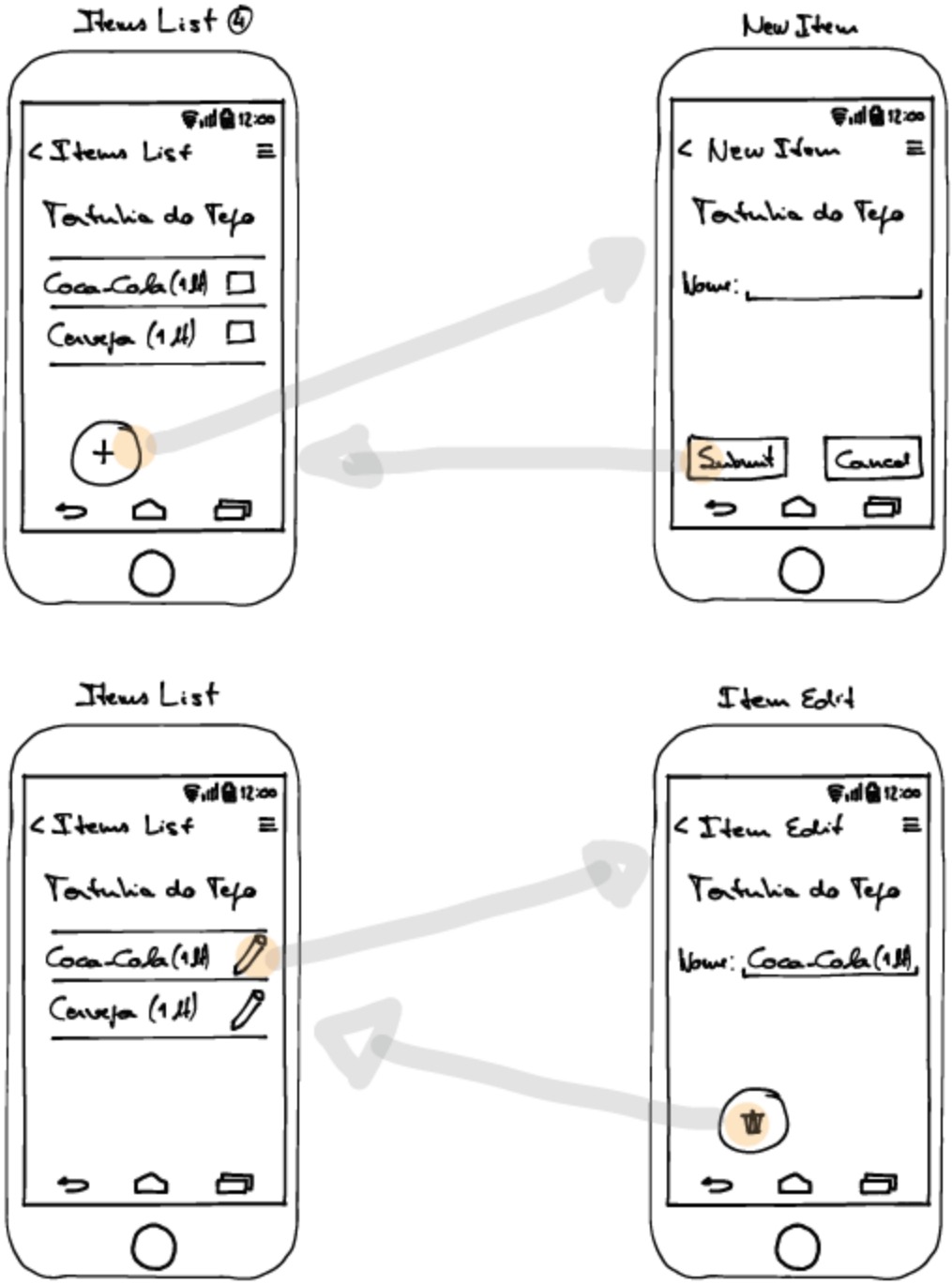
Table 5: User stories

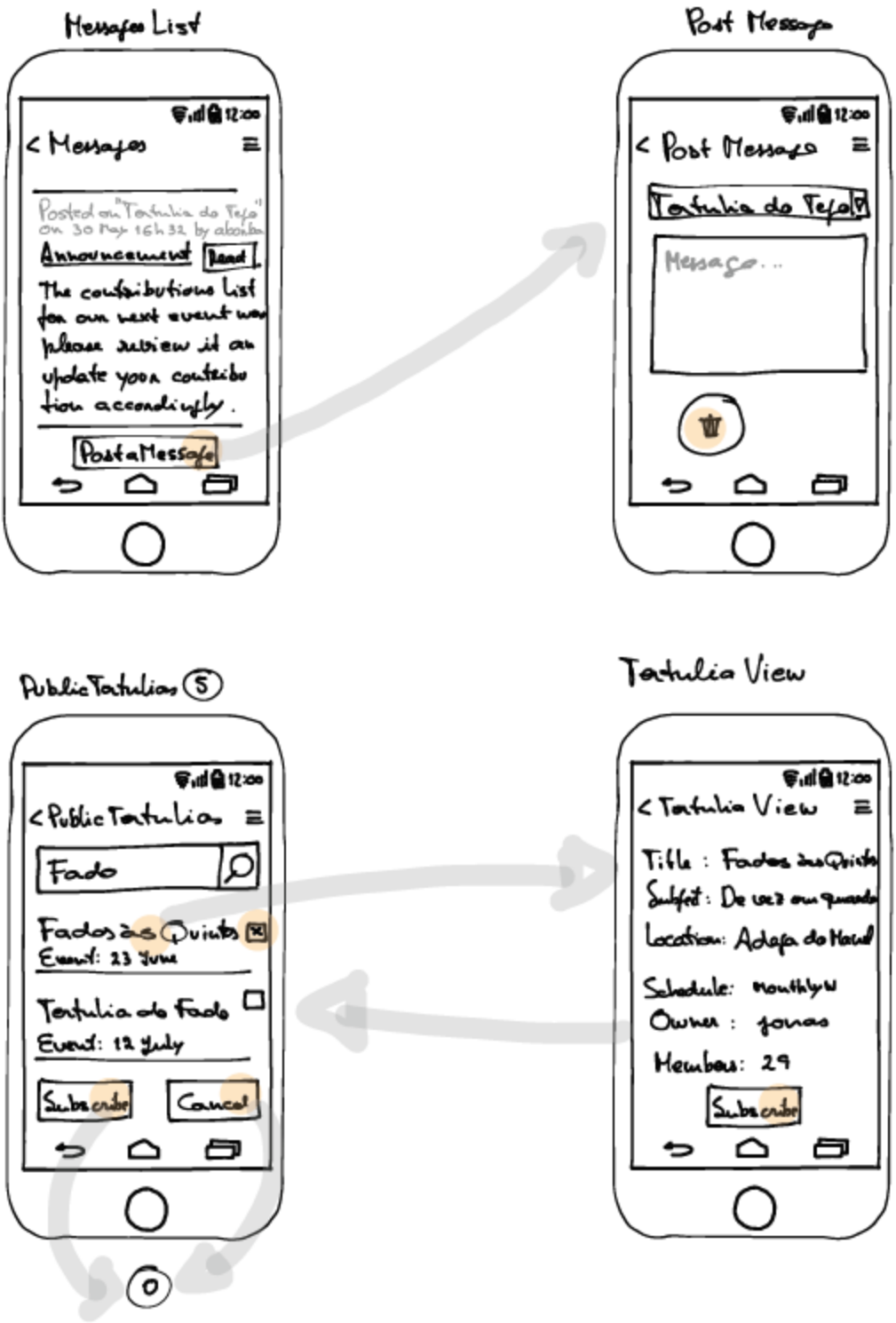
ID	As a …	I want …	so that …
1	user	to create a public or private tertúlia	I can try to build a community or a group around a subject
2	tertúlia member	to view tertúlia details (name, owner, description, rec. schedule, etc.)	I can check next gathering
3	tertúlia member	to view details of all tertúlias I am in (name, recurring schedule, etc.)	I can decide if I will participate or not
4	tertúlia owner	to update this tertúlia data (name, description, recurring schedule, etc.)	I can update tertúlia information
5	tertúlia owner	to setup a suspension period	all member get notified about it
6	tertúlia member	to invite a user to join in	tertúlia's interest grows
7	tertúlia member	to register for next tertúlia gathering	the organization counts me in
8	tertúlia member	to propose a change for next tertúlia (date, location)	it gets in line with my needs
9	tertúlia member	to vote on proposed changes for next tertúlia	it gets in line with my needs
10	tertúlia member	to mute/unmute tertúlia notifications	I can tune the level of awareness
11	tertúlia member	to see a map route to the tertúlia location	I can get hints on driving options
12	tertúlia member	to receive a tertúlia reminder in advance	I don’t forget to include it in my agenda
19	tertúlia manager	to publish a shopping list for a tertúlia	I can manage tertúlia logistics
20	tertúlia member	To choose tertúlia shopping list items	I can select my contribution

8.3. Annex 3 - User Interaction


Members List (2)
< Members
Tertulia de Tefo
aborba
owner
jonas
member
tarea
member
Invite
Pending
Edit Member
< Edit Member
Tertulia de Tefo
aborba
Role : Owner
Remove
Edit Profile
< Edit Profile
aborba
email:
aurbio.borba@gmail.com
Owned Tertulias:
Tertulia de Tefo
2 members
(1)
Invite User
< Invite User
Tertulia de Tefo
email:
aurbio.borba@gmail.com
Message:
Submit
Cancel
(2)


Prepare Event ③
< Prepare Event
Tertulia do Tefo
Date : 24 June 2016
Time: 13h00 - 15h00
Location: Cave Real
URL:
Contributions
Prepare Contributions
< Prep. Contributions
Tertulia do Tefo
24 June 13h00-15h00 Cave Real
Coca-Cola 1L: 4
Guarda Pateo 200: 2
Pão de leite(6): 4
Templates List
< List Templates
Tertulia do Tefo
Snacks
Drinks
Edit Template
< Edit Template
Snacks
Coca-Cola 1L: 4
Guarda Pateo 200: 2
Pão de leite(6): 4
④


Items List ④
< Items List
Tertulia do Tejo
Coca-Cola (1 lt)
Cerveja (1 lt)
New Item
< New Item
Tertulia do Tejo
Nome:
Submit
Cancel
Items List
< Items List
Tertulia do Tejo
Coca-Cola (1 lt)
Cerveja (1 lt)
Item Edit
< Item Edit
Tertulia do Tejo
Nome: Coca-Cola (1 lt)


Messages List
< Messages
Posted on "Tertulia do Tejo" on 30 May 16h32 by alcobia
Announcement
Read
The contributions list for our next event was released please review it an update your contribution accordingly.
Post a Message
Post Message
< Post Message
Tertulia do Tejo
Message...
Public Tertulias 5
< Public Tertulias
Fado
Fados às Quintas
Event: 23 June
Tertulia do Fado
Event: 12 July
Subscribe
Cancel
Tertulia View
< Tertulia View
Title : Fados às Quintas
Subject : De vez em quando
Location: Adega do Manel
Schedule: monthly
Owner : joana
Members: 29
Subscribe

8.4. Annex 4 – Cloud platform selection

For the server platform selection, after making a market survey, we came up with the short list of cloud services suppliers presented in table 6.

Table 6: Cloud services suppliers short-list

Subject	Subject Full Name	Owner
Azure	Microsoft Azure Mobile Services	Microsoft
StrongLoop	IBM Mobile Services	IBM
Appcelerator	The Appcelerator Platform	Axway
AWS	AWS Mobile Hub	Amazon
FeedHenry	RedHat Mobile Services	RedHat
FireBase	FireBase	Google
BaasBox	BaasBox	Apache
built.io	built.io flow	Built.io
MongoLab	mLab	MongoDb
Oracle	Oracle Mobile Cloud Service	Oracle

To compare the candidates in the short list for backend, we compiled an RFP like set of requirements and then split it into two groups according to its relevance to the project:

- The first group contains the requirements to which full compliance is required. Subjects non fully compliant to any of the stated requirements in this group do not qualify for evaluation and are discarded. Those requirements are listed in table 7.
- In the second group full compliance is not required and therefore the score awarded for each requirement is weighted prior to be added to the total score. The requirement's weight is a measure of the its relevance for the project. Those requirements are listed in table 8.

Table 7: Must comply requirements

Requirement description	What is appreciated
Offer is part of core business	The positioning of the subject in respect to the supplier's current core business.
Service maturity (> 5 yrs)	The subject in respect to its overall stability as perceived by market analysts and technology reviewers.
Customer base size	The market penetration of the subject as measure of the company commitment and its ability to deliver.
Relevant mobile Apps using it	If there are major mobile Apps using the subject as back end provider.
DB Backend (SQL/NoSQL)	If the subject includes and exposes either SQL or JSON repositories that can be interrogated.
IAM	If the subject provides authentication, authorization and privileges management within the system.

Requirement description	What is appreciated
REST API	If the subject provides in it's API, the HTTP methods GET, PUT, POST and DELETE, both for elements and collections, coded in either XML or JSON.

Table 8: Scored requirements

Requirement description	What is appreciated
Low entry level costs	If the subject related costs for a production system would be low enough for a small organization to support it for a 1 year period or until the business takes off.
Cloud Free-tier	If the subject provides a environment free for prototype development and tests.
Server Side Code	If the subject includes support for server side Javascript, Java or C# to build or enhance an API and/or serve HTML.
GeoSpacial services	If the subject includes the ability to query or filter based on positioning.
Management console	If the subject includes a management console to setup and update system configuration, manage users, view data and overall statistics.
SDKs	If the subject provides libraries for Android, IOs and client-side javascript.
Push notifications/sync services	If the subject provides push notification services and/or local persistency and client synchronization services.
Cloud & Hosted & OnPermises	The possible operating modes of the subject, among cloud based, hosted or on premises.
Setup easiness	Perceived easiness of the subject's environment setup for the current development scope.
Overall personal impression	Overall impression gained by the analyst from research on the subject, against the remaining short-listed subjects.

After applying the must comply criteria to the initial short-list, the candidates listed in table 9 were excluded on the basis of non-compliance with the indicated requirements.

Table 9: Short-list non-compliant candidates

Requirement	BaasBox	buit.io	MongoLab	Oracle
Offer is part of core business	Y	Y	Y	Y
Service maturity (> 5 yrs)	N	Y	N	N
Customer base size	N	N	N	N
Relevant mobile Apps using it	N	N	N	N
DB Backend (SQL/NoSQL)	Y	Y	Y	Y
IAM	Y	Y	Y	Y
REST API	Y	Y	Y	Y

The application of the scoring criteria from table 8 to the remain candidates produced the results shown in table 10, leading us to select **Microsoft Azure Mobile Apps** as our backend cloud service.

Table 10: Short-list candidates scoring

Requirement	**W**	**Azure**	**StrongLoop**	**Appcelerator**	**AWS**	**FeedHenry**	**FireBase**
Low entry level costs	8	1	1	1	1	1	1
Cloud Free-tier	5	1	1	1	1	1	1
Server Side Code	8	1	1	1	1	1	0
GeoSpacial services	1	1	1	1	1	0	1
Management console	5	1	1	1	1	1	1
SDKs	4	1	1	1	1	1	1
Push notifications / sync services	8	1	1	1	1	1	0
Cloud & Hosted & OnPermises	3	2	2	1	2	3	1
Setup easiness	3	5	3	5	1	2	5
Overall personal impression	5	5	5	3	4	3	3
Score	**-**	**85**	**79**	**72**	**68**	**68**	**56**

8.1. Annex 5 – Client platform selection

The problem we face is that here might not be enough time to develop both "tertulias" Android app and mobile web with the quality required for this project and within the available time frame and team dimension, so we had to make a choice and we chose to develop an Android app together with the required server API setup in a cloud based service.

Both the Android app and the Mobile web options have advantages (and drawbacks); Table 11 is included keep track of what we took into account to support our decision.

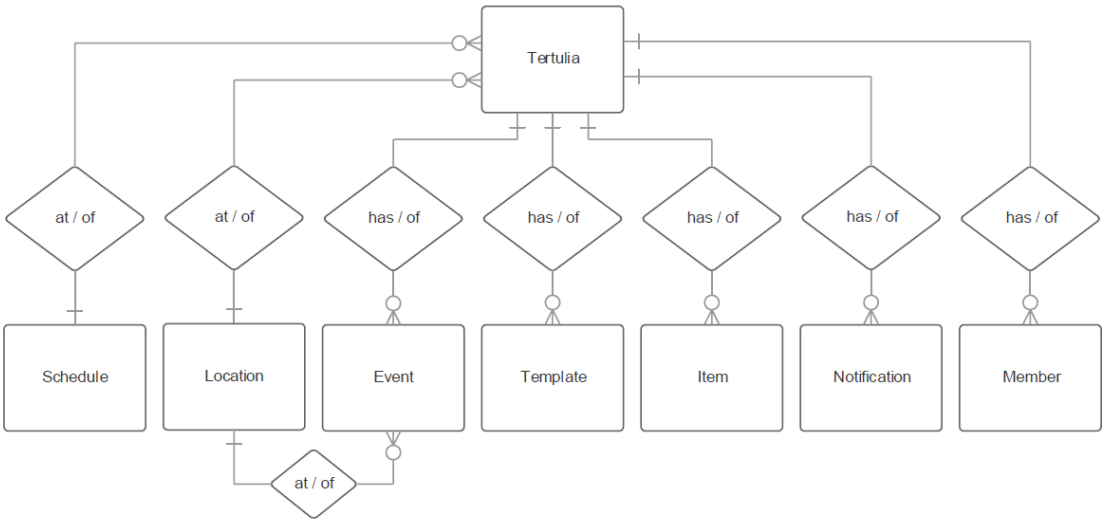
Table 11: Android app vs. Mobile Web

App vs Browser	**+ / -**	**Comment**
Browser	+	Browser development is more generic - it runs on every device provided it has a suitable web browser.
Browser	+	References to Web sites are easier to find on Internet searches than references to Apps.
App	+	App based systems can leverage on the power of App stores.
Browser	+	Browser based systems have a larger life span because the dependency of the device operating system version is reduced.
Browser	+	Browser based systems are more likely to suffer security attacks (e.g. defacing).
App	+	Android based can take great advantage from push notifications.
App	-	App based systems must be available to both Android and IOs in order to cope with tertulia's users diversity.
App	+	People know that if they want an App they find it in the store.
App	+	With App based systems it easier to match specific features to specific markets.
Browser	+	Browser based systems eliminate the problem of API evolution and aging.
Browser	-	Mobile browser based systems don't support off-line mode.
App	+	App based systems are more seamlessly integrated with the device and are likely to provide a better user experience.
Browser	-	Mobile web browsers provide very limited access to device sensors.
Hybrid	+	Hybrid apps - or Web wrappers - could be a third way to try to address some of the benefits of Apps, while retaining other benefits from the Mobile Web approach.

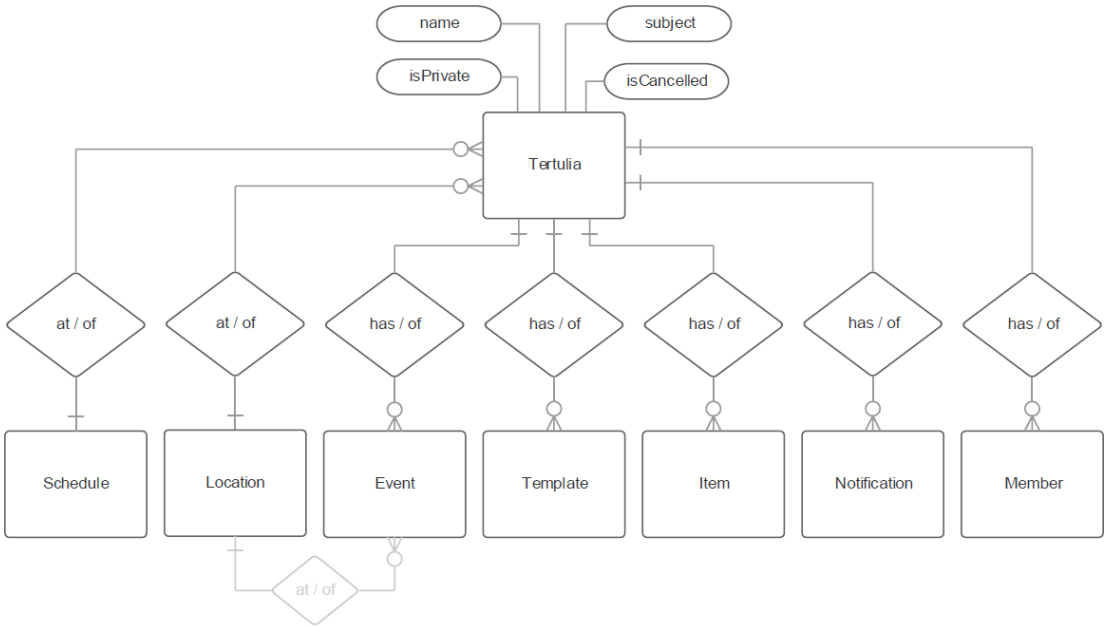
Weighted the pros and cons in each case we opted to go for an Android client development for the current project scope, leaving for future development and iOS and a javascript based version. As the server interface is done via authenticated Http requests, and as the selected server platform provides and SDK for all platforms, those future developments will be essentially client side development for application code porting because the app behavior and server interaction shall remain the same.

8.1. Annex 5 – Data persistence: ER Model

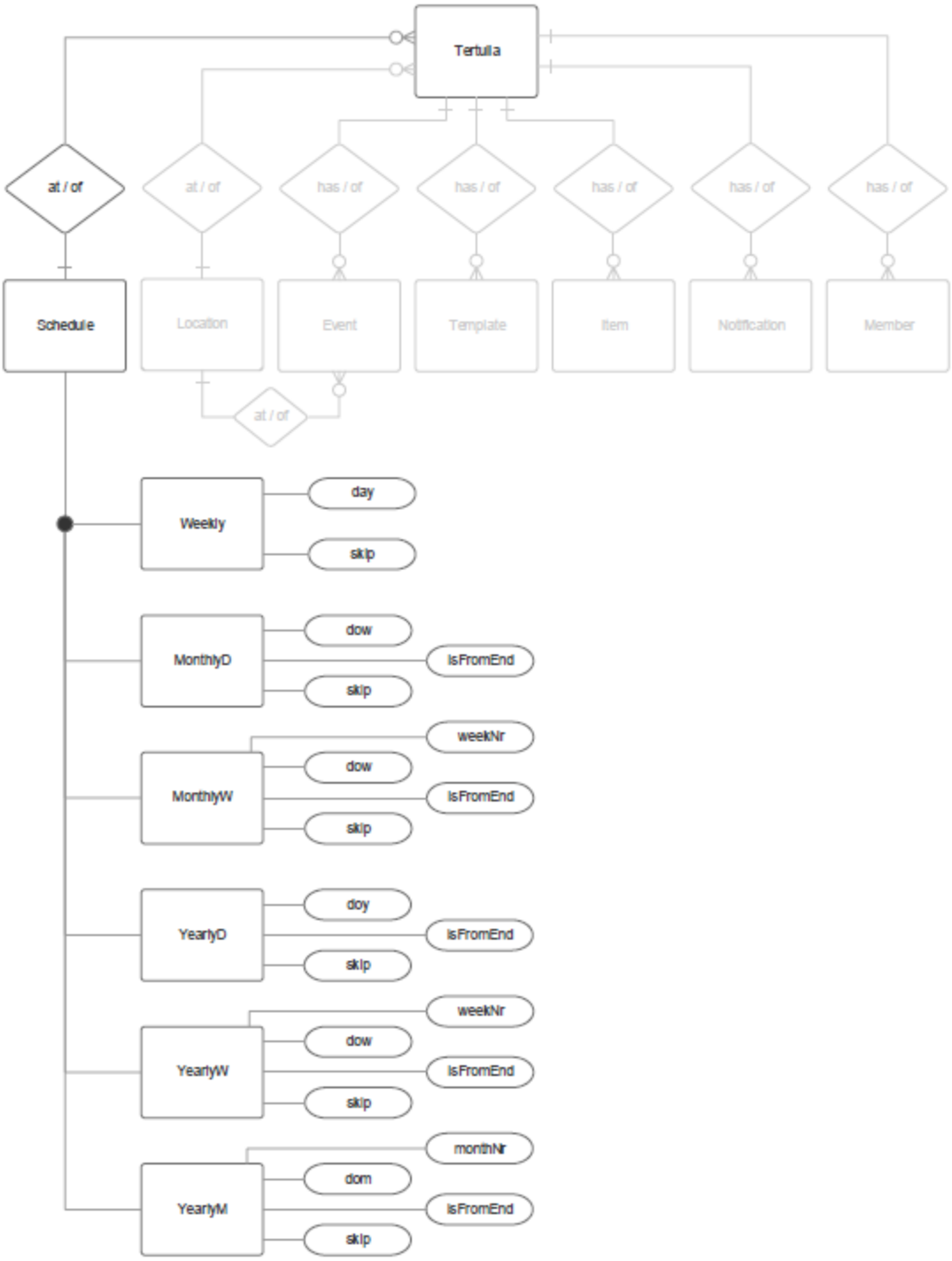
8.1.1. Main entities



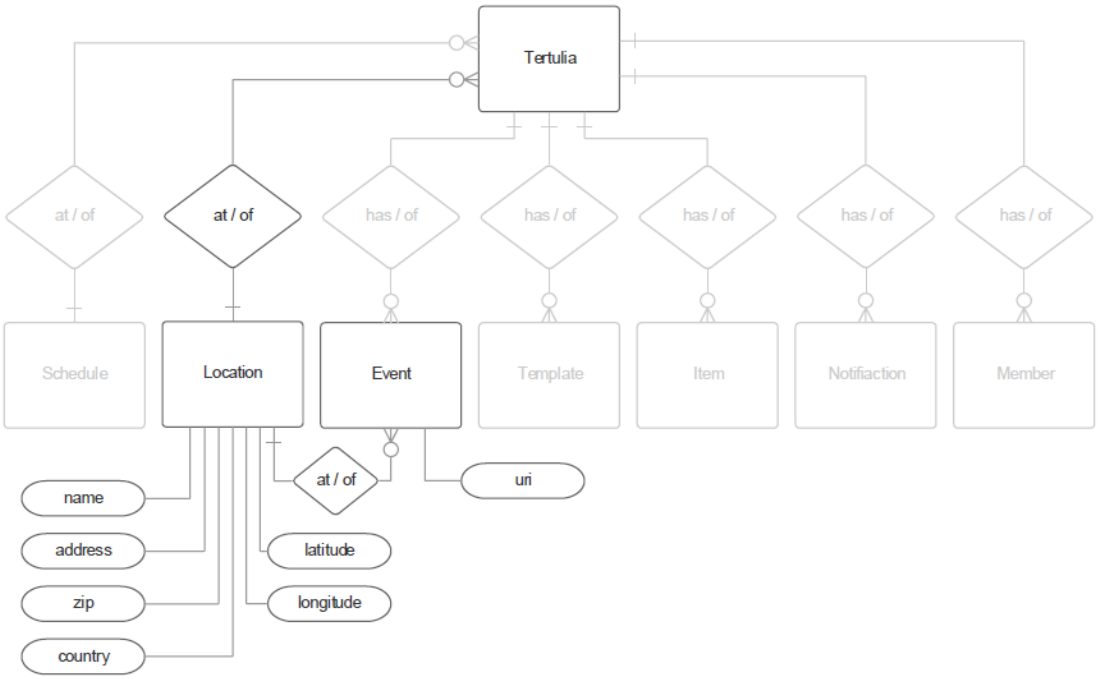
8.1.2. Relations of the Tertulia entity



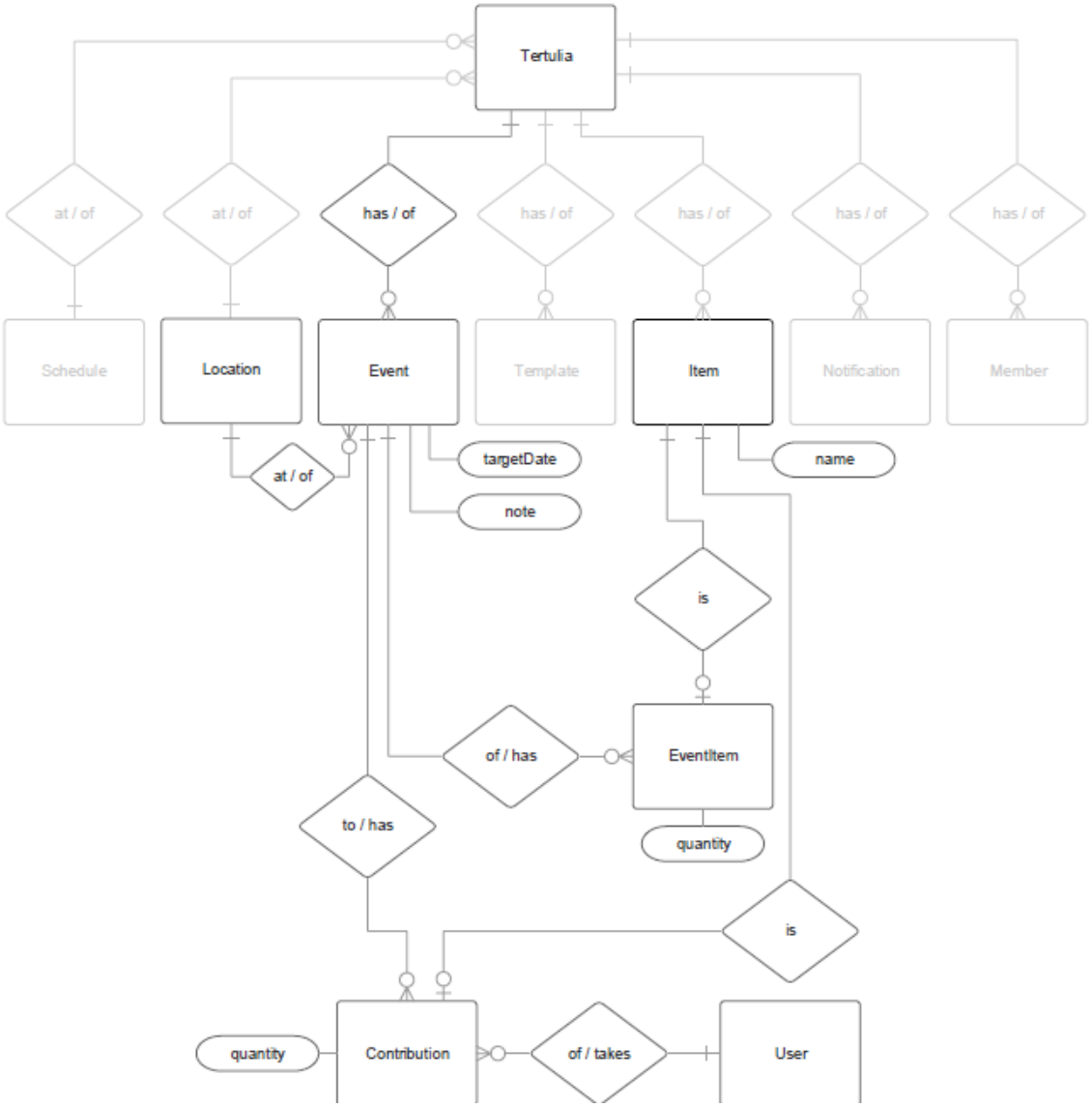
8.1.3. Relations of the Schedule entity



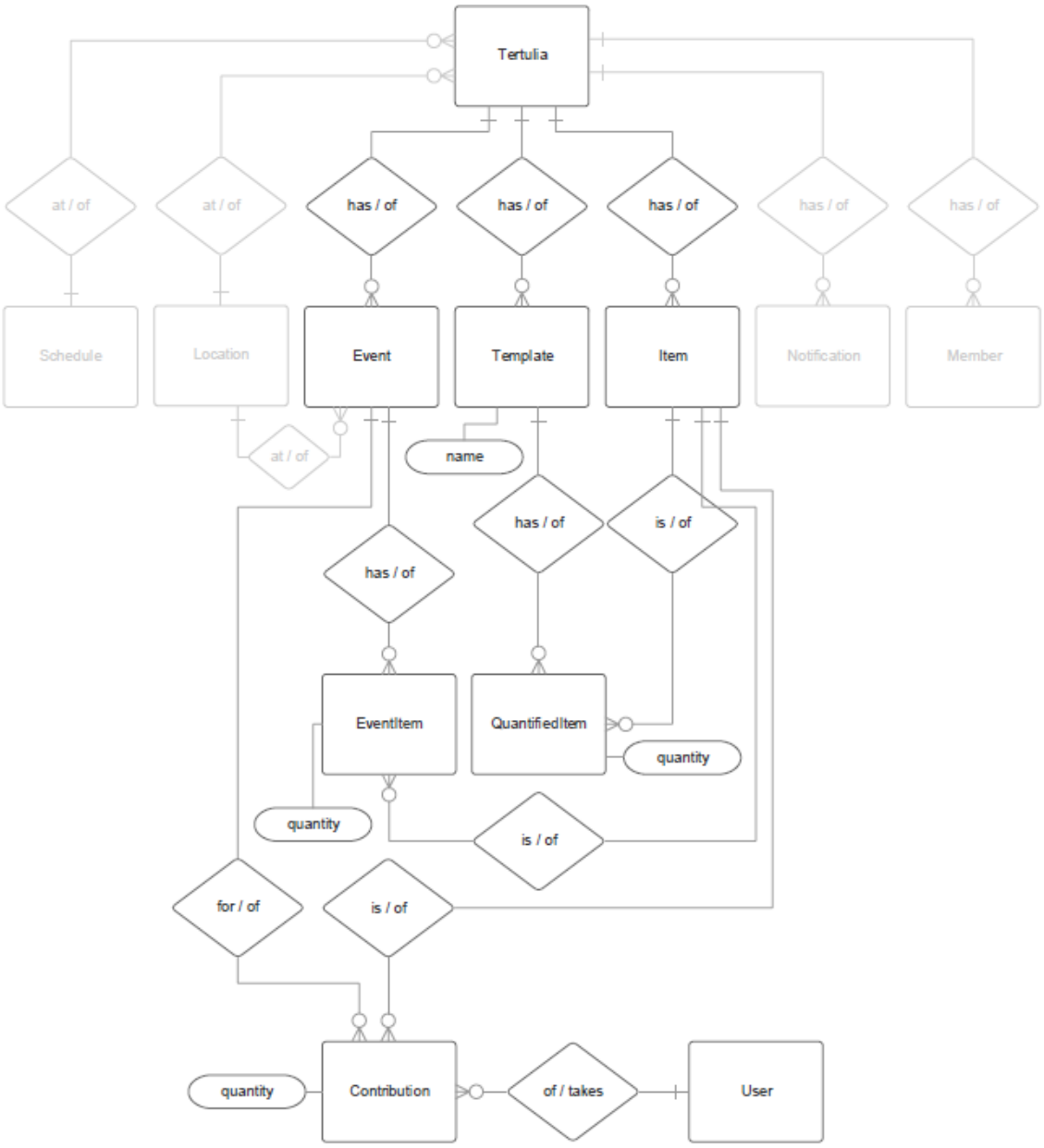
8.1.4. Relations of the Location entity



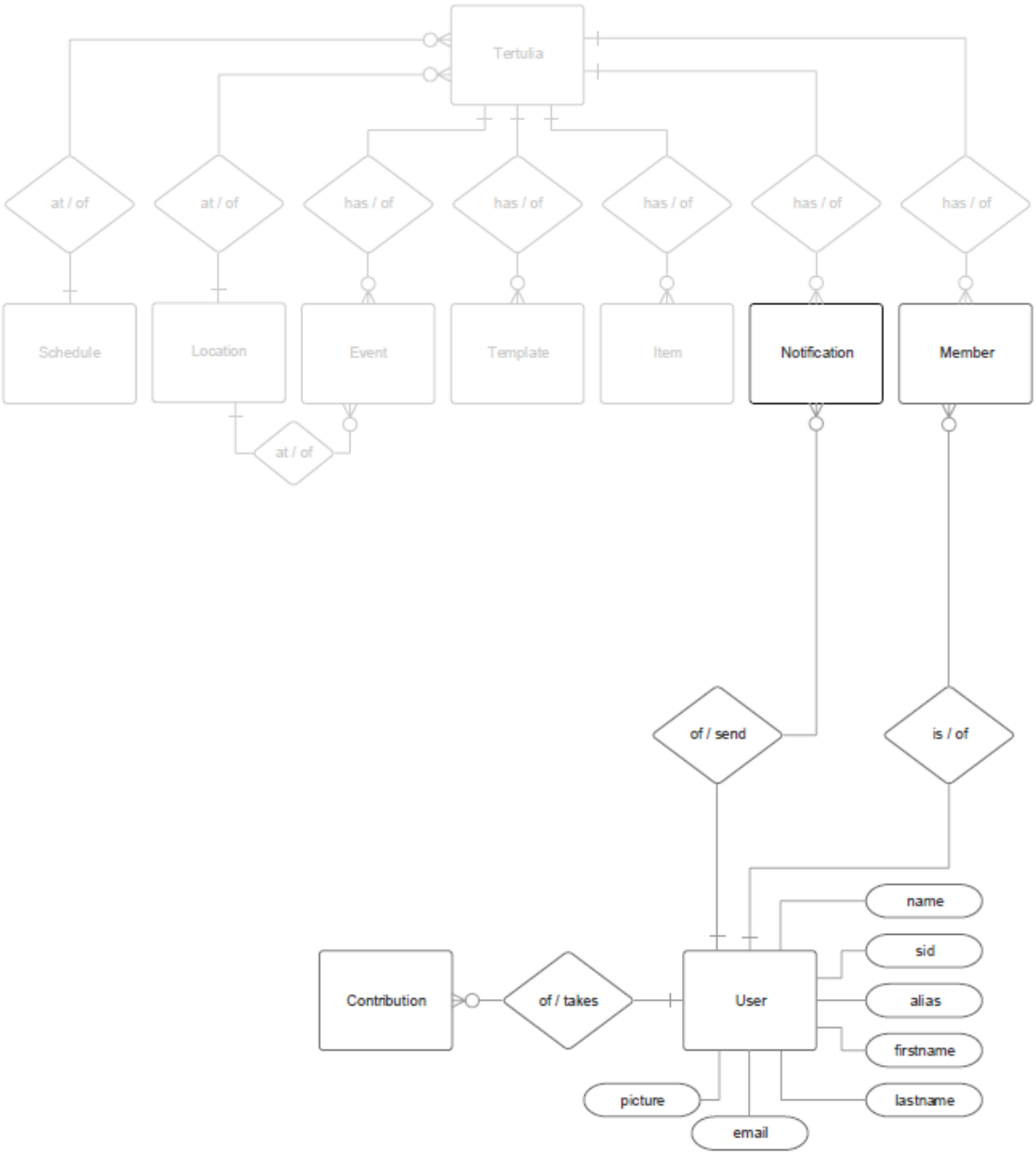
8.1.5. Relation of the Event entity



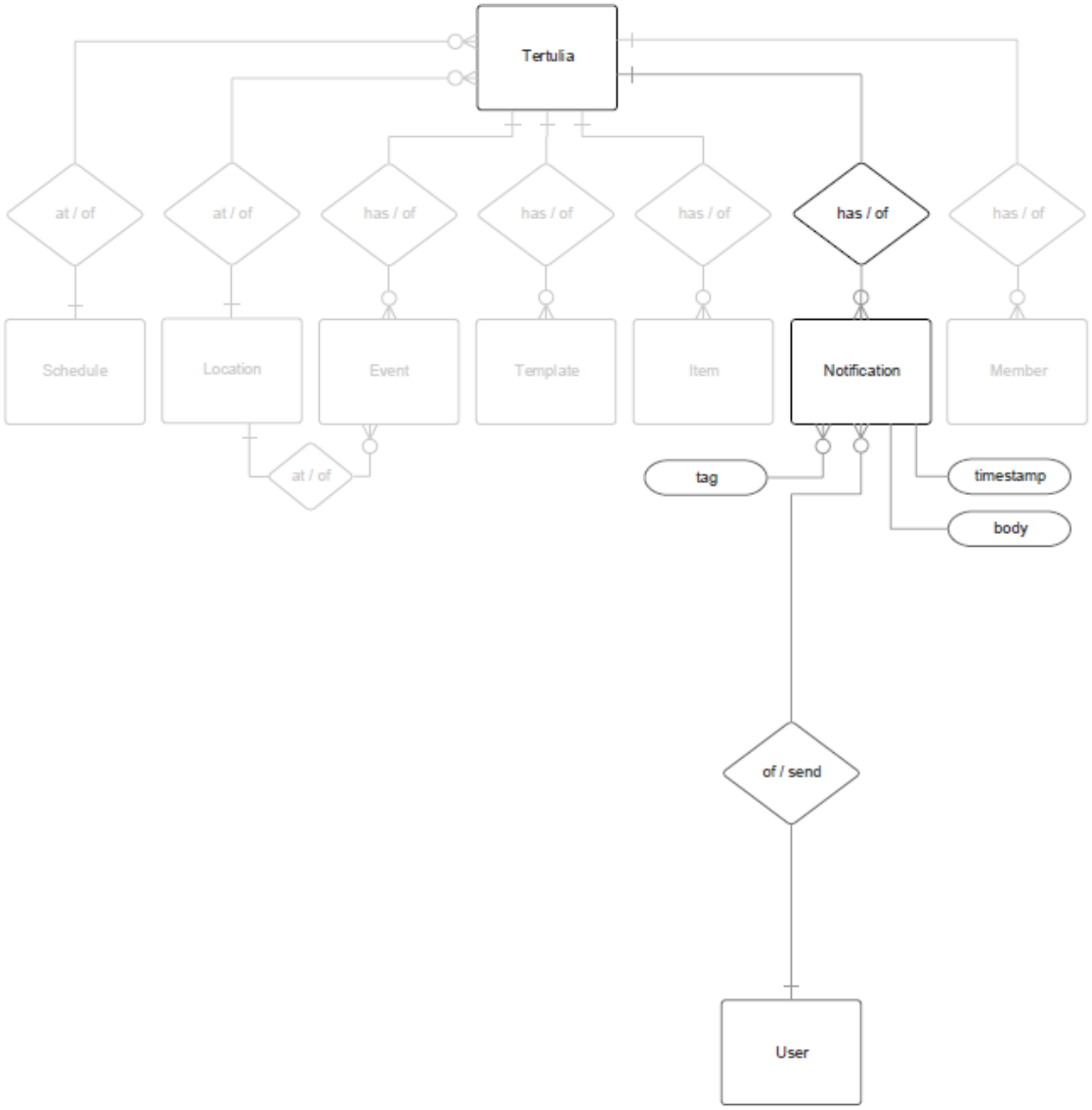
8.1.6. Relations of the Event, Template, Item, EventItem, QuantifiedItem and Contribution entities



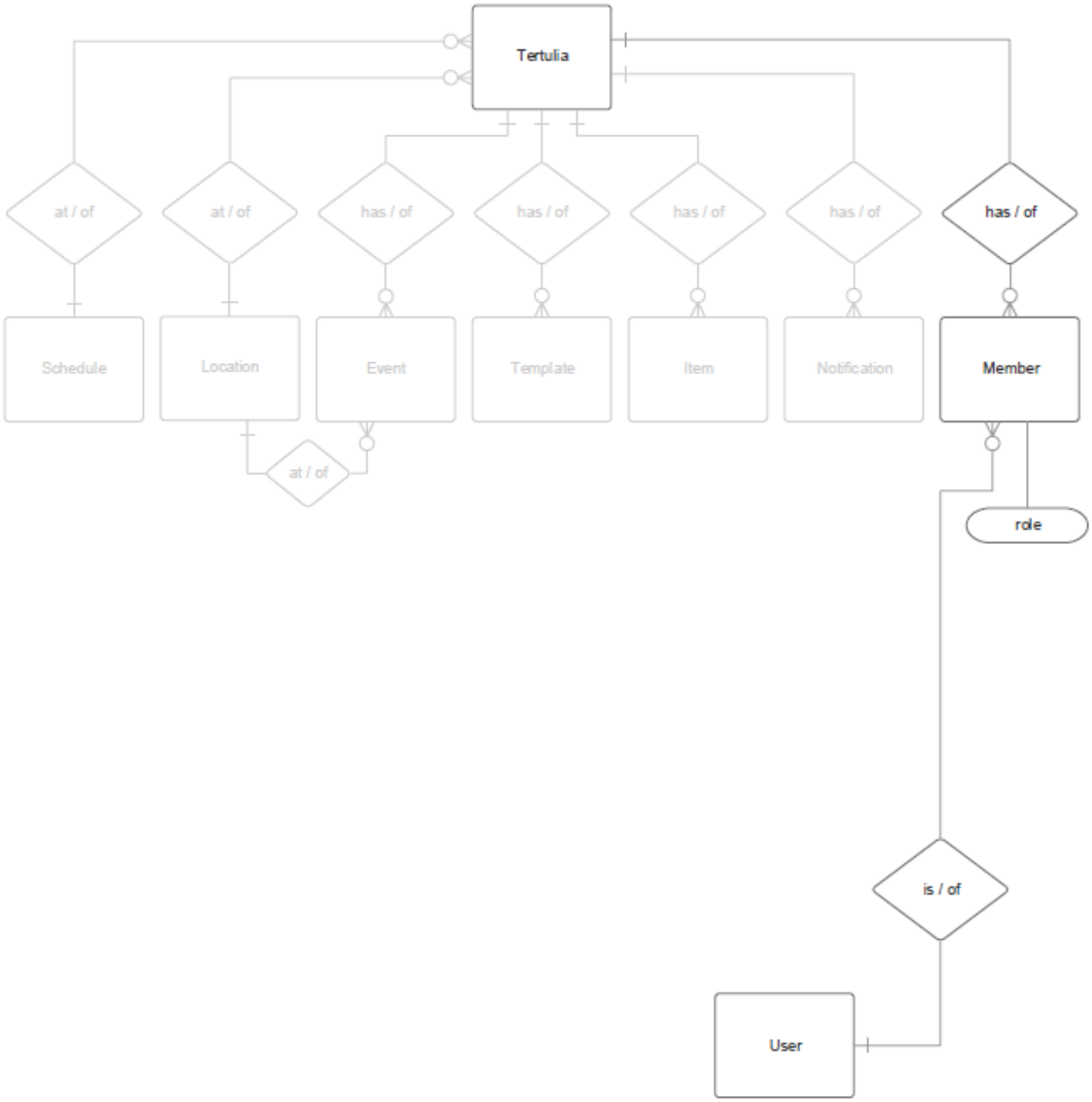
8.1.7. Relations of the User entity



8.1.8. Relations of the Notification entity



8.1.9. Relations of the Member entity



8.1.10. Relations of the Invitation entity