ISEL
INSTITUTO SUPERIOR DE
ENGENHARIA DE LISBOA

Tertúlias

António Manuel Peres Celorico Borba da Silva

Supervisors: Eng. Pedro Miguel Henriques Santos Félix

Progress report within the scope of Project and Seminar

Curso de licenciatura em Engenharia Informática e de Computadores

Summer semester 2015/2016

May 2016

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Tertúlias

22908 António Manuel Peres Celorico Borba da Silva

Supervisor: Eng. Pedro Miguel Henriques Santos Félix

Progress report within the scope of Project and Seminar

Curso de licenciatura em Engenharia Informática e de Computadores

Summer semester 2015/2016

May 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/Tertúlia>. Accessed on: March 21th, 2016)

The motivation for addressing this project is two fold:

- 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. 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 social service providers,
- 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 such as the Java's based Android development framework, Node.js's javascript and Microsof's TSQL, etc.

In the end we shall have a working app, available for Android, with documented API that will allow for development to other mobile or browser based platforms or for integration with third party applications.

Keywords: mobile; social; notifications.

Table of Contents

ABSTRACT **IV**

TABLE OF CONTENTS **III**

IMAGES LIST **IV**

TABLES LIST **IV**

1. INTRODUCTION **1**

2. CHAPTER 4. (STATUS **2**

3. PROBLEM SCOPE **3**

1.1. COMPETITIVE ANALYSIS 3

3.1. USER STORIES 5

4. STATUS **7**

5. SOLUTION BUILDING BLOCKS AND CORE DECISIONS **8**

5.1. SELECTION OF THE APPLICATION SERVER PLATFORM 8

5.2. CLIENT PLATFORM SELECTION 11

5.3. CLIENT AUTHENTICATION 12

5.4. CLIENT OFFLINE USAGE 12

5.5. PUSH NOTIFICATIONS 13

6. SOLUTION DESIGN **14**

6.1. APPLICATION SERVER 15

6.2. ANDROID CLIENT 15

6.3. DEVELOPMENT ENVIRONMENT 15

7. PROJECT PLAN AND STATUS **17**

7.1. PROJECT PLAN 17

7.2. DELIVERABLES 17

7.3. PROGRESS AND CHANGES TO THE INITIAL PLAN 18

7.3.1. Next steps 18

Images list

Figure 1: Service main building blocks 8
Figure 2: Solution building blocks 14

Tables List

Table 1: Applications analysis / strengths + weaknesses 4
Table 2: Applications analysis / opportunities + threads 5
Table 3: User stories 5
Table 4: Cloud services suppliers short-list 9
Table 5: Must comply requirements 9
Table 6: Scored requirements 10
Table 7: Short-list non-compliant candidates 10
Table 8: Short-list candidates scoring 11
Table 9: Android app vs. Mobile Web 11
Table 10: Project Schedule 17

1. Introduction

This document's goal is to introduce the Tertulias project and report its development progress.
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 structured is from the more general aspects of the development to the most particular ones, aiming to drive focus to each specific issue, preserving context and avoiding mixing subjects.
In chapter 3. (Problem Scope) the boundaries of the problem we try to solve is addressed, stating what shall be the outcome of development and what could possibly be the following evolution.

2. Chapter 4. (Status

Before getting into the application specifics, and within the context of a progress report, this chapter presents the current project development status.

- The scope of the project is agreed and the target architecture is defined.
- The application server infrastructure is configured and operational to support development.
- The source code version control system is also configured and in place.
- The development environment for all involved parts of the system is configured and operational.
- A mockup was built to prove the core end to end functionality.
    - An API endpoint was configured in the application server.
    - The end point accepts a GET request only from authenticated users, rejecting unauthenticated ones with an appropriate status code.
    - The application server on the end point described above queries the database to retrieve information in the context of the authenticated user.
    - An Android client application queries the application server getting the query result serialized in Json, parses it and display some fields of the received result in the application screen. The application has the ability to send the request for both an authenticated and unauthenticated user, and processes the result or the error accordingly.

Solution Building Blocks) explains the core decisions that define some of the determining aspect of the project, and details some specific actions taken – and the respective conclusions – leading to the final solution design.

Right after, chapter 6. (Solution Design) addresses the solution detailed design, by blocks and in terms of responsibility and functionality. The development environment setup required for each block is also addressed here.

Finally, chapter 7. (Project Plan and Status) presents the project schedule plan and states the progress so far.

3. 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” to lunch.
- Of course, in August they all go on holidays, so the meeting is on pause.
- Each time one of them needs to make a reservation in advance.
- Once in a while, someone suggest a new restaurant and they all decide to try it.
- … and so on …

Given this scenario, a couple of building blocks stand out as necessary:

- 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 service provider: It’s responsibilities is to ensure that the client application representing user is representing the user who he says he is representing.

With the above scenario in mind, there are two analyses that helps 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.

1.1. 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 table 1 below. Table 2 adds to the analysis the possible effects of the usage of both Tertulias app and the app in analysis by a user.

Table 1: 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 2: Applications analysis / opportunities + threads

Application	Opportunities[1]	Threads[2]
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

3.1. User Stories

User stories is a way of defining the project functional requirements from a user standpoint and, at the end of the day, satisfying customer's expectations is key for a successful project.

Table 3: 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

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

ID	As a …	I want …	so that …
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

4. Status

Before getting into the application specifics, and within the context of a progress report, this chapter presents the current project development status.

- The scope of the project is agreed and the target architecture is defined.
- The application server infrastructure is configured and operational to support development.
- The source code version control system is also configured and in place.
- The development environment for all involved parts of the system is configured and operational.
- A mockup was built to prove the core end to end functionality.
    - An API endpoint was configured in the application server.
    - The end point accepts a GET request only from authenticated users, rejecting unauthenticated ones with an appropriate status code.
    - The application server on the end point described above queries the database to retrieve information in the context of the authenticated user.
    - An Android client application queries the application server getting the query result serialized in Json, parses it and display some fields of the received result in the application screen. The application has the ability to send the request for both an authenticated and unauthenticated user, and processes the result or the error accordingly.

5. Solution Building Blocks and Core Decisions

Irrespective of the technology involved, it’s clear for us that the major blocks that will build the solution are the ones pictured in figure 1.

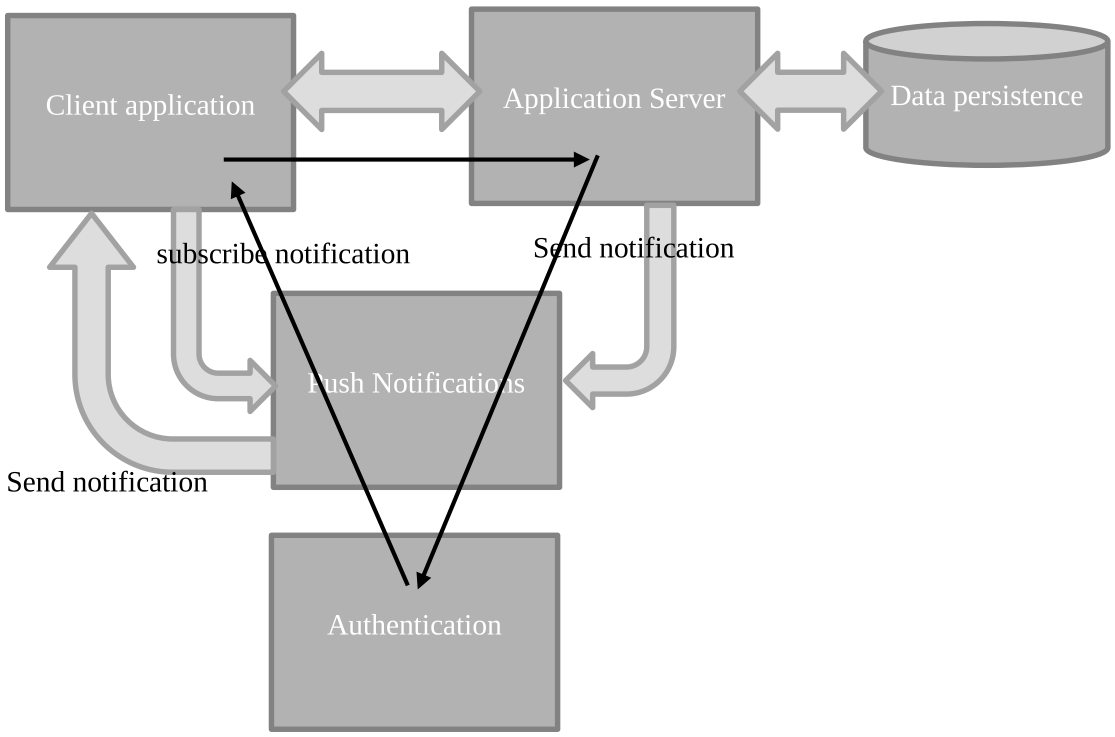

Figure 1: Service main building blocks

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.

5.1. 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[3]. After making a market survey, we came up with the short list of cloud services suppliers presented in table 4.

Table 4: 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 5.
- 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 6.

Table 5: 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.

[3] 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.

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 6: 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 7 were excluded on the basis of non-compliance with the indicated requirements.

Table 7: 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 6 to the remain candidates produced the results shown in table 8, leading us to select **Microsoft Azure Mobile Apps** as our backend cloud service.

Table 8: 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**

5.2. 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 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 9 is included keep track of what we took into account to support our decision.

Table 9: 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 solutions can leverage on the power of App stores.
Browser	+	Browser based solutions have a larger life span because the dependency of the device operating system version is reduced.
Browser	+	Browser based solutions are more likely to suffer security attacks (e.g. defacing).
App	+	Android based can take great advantage from push notifications.
App	-	App based solutions 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 solutions it easier to match specific features to specific markets.
Browser	+	Browser based solutions eliminate the problem of API evolution and aging.
Browser	-	Mobile browser based solutions don't support off-line mode.
App	+	App based solutions 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.

5.3. 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.

5.4. Client offline usage

In the Android it makes sense to keep replication of some server data to avoid dependency of network availability and latency and save on data communications and battery usage.
For that we shall use an Android’s content provider and use push notifications instead of server pooling for launching data synchronization.

5.5. 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 application development in a more advanced state.

6. Solution Design

The blocks referred above fit in the blocks of figure 1 as presented in figure 2.

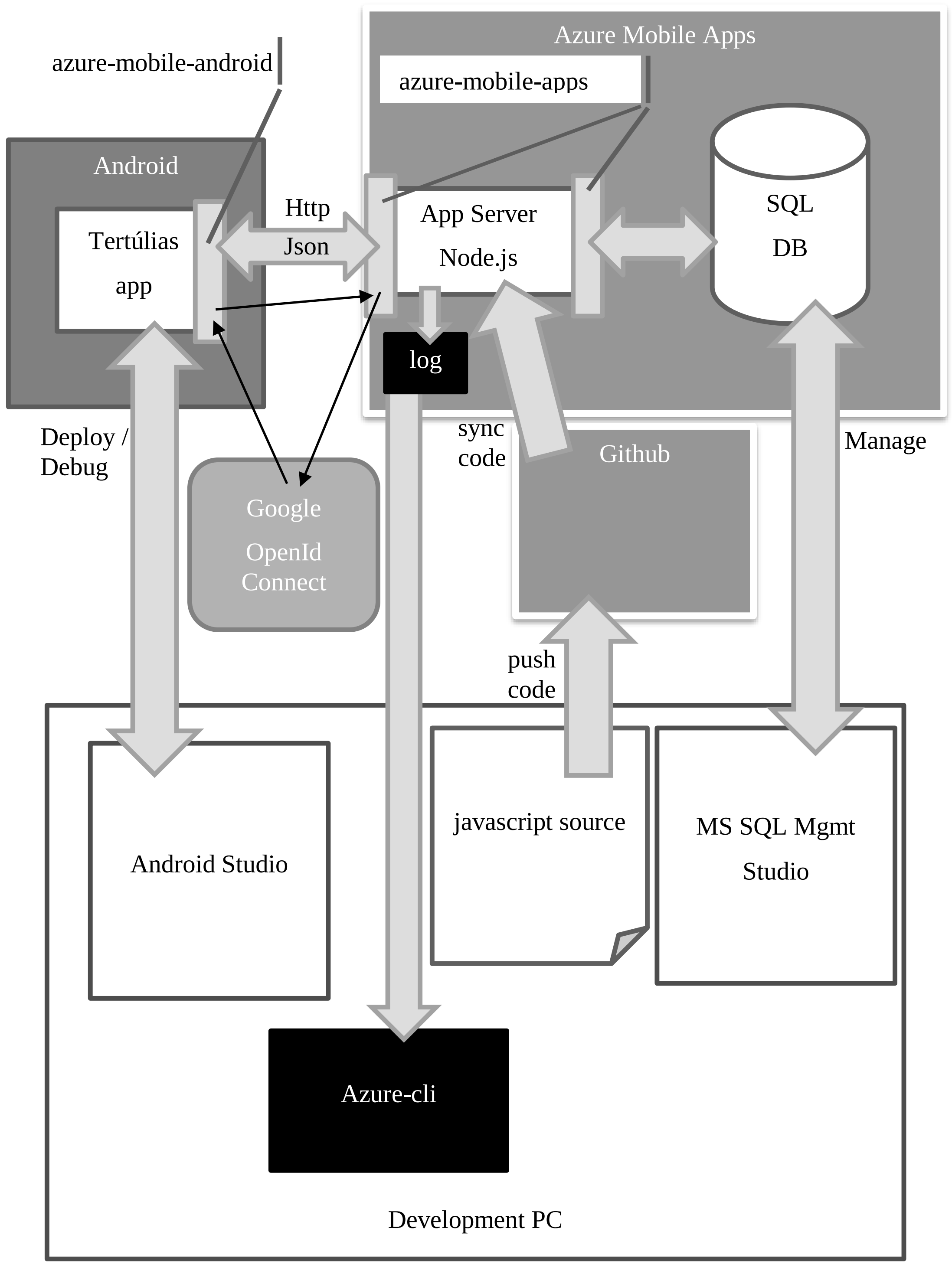

Figure 2: Solution building blocks

6.1. 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.

6.2. 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".

Additionaly, Google's gson library is used to Json conversion.

Client authentication is as presented in nr. 5.3 above.

6.3. 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 restarted.
  - 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[4] 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.

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

7. Project Plan and Status

7.1. Project Plan

The project schedule is presented in table 10. The columns meaning are:

- W – Project week number.
- Fri – Week Friday's date.
- Task – Task description.
- PS Delivery – Project and Seminar scheduled deliveries.

Table 10: Project Schedule

W	Fri	Tasks	PS Delivery
~~1~~	4 Mar	- Proposal draft	**Mar 28th:** Proposal delivery
~~2~~	11 Mar		
~~3~~	18 Mar		
~~4~~	25 Mar	- **Project proposal delivery**	
~~5~~	1 Apr	- Android dev. setup / users / tertúlia / gathering modeling	**May 2nd:** Progress report and Individual presentation delivery
~~6~~	8 Apr	- Android UI; **Server technology selection document delivery**	
~~7~~	15 Apr	- Server setup / Android server interface	
~~8~~	22 Apr	- Individual presentation preparation	
~~9~~	29 Apr	- Server push notifications / Android notifications - **Progress report finish and delivery**	
10	6 May	- Server functionality / console	**Jun 13th:** Poster and Beta version delivery
11	13 May	- Server component development	
12	20 May	- **Alpha version delivery**	
13	27 May	- User management	
14	3 Jun	- Server component UI - **Poster delivery**	
15	10 Jun	- Android / server final integration - **Beta version delivery**	
16	17 Jun	- Report finish	**Jul 23rd:** Final version
17	24 Jun	- Final tweaks and tests	
18	1 Jul	- **Report Delivery**	
19	8 Jul	- **Final version delivery**	
20	15 Jul		
21	22 Jul		

7.2. Deliverables

To fulfill the Project and Seminar lecture requirements and further documentation agreed with the project's supervisor, the following list presents the committed deliverables (these deliverable shows highlighted in table 10):

- Project proposal delivery

- Server technology selection document delivery: A document with the evaluation of different technologies to be used as the backend for the project.
- Individual presentation: A presentation on a theme in the scope of Project and Seminar lecture.
- Progress report: A document that reports the intermediate project status – This document.
- Alpha version: A version of the project aiming at features validation and supervisor's comments.
- Beta version: A full working version to start live testing with selected users.
- Final version: Final version deployed in the App store.

7.3. Progress and changes to the initial plan

Today we are starting week 10.

So far, the deliverables are in line with the agreed schedule in the previous numbers.

The development is one week delayed, but we think it is fully recoverable, so no rescheduling is necessary.

7.3.1. Next steps

Next steps are to negotiate incrementally the API endpoints structure (routes) necessary to support all the server side functionality, while implementing that functionality both on server side and client side, according to the agreed schedule.