

Sommario

[INTRODUCTION 3](#_Toc13407186)

[PURPOSE 3](#_Toc13407187)

[SCOPE 3](#_Toc13407188)

[GLOSSARY 3](#_Toc13407189)

[Definition 3](#_Toc13407190)

[Acronyms 3](#_Toc13407191)

[Abbreviation 3](#_Toc13407192)

[REFERENCE DOCUMENT 3](#_Toc13407193)

[DOCUMENT STRUCTURE 3](#_Toc13407194)

[ARCHITECTURAL DESIGN 3](#_Toc13407195)

[OVERVIEW 3](#_Toc13407196)

[Server 4](#_Toc13407197)

[Client 4](#_Toc13407198)

[COMPONENT VIEWS 4](#_Toc13407199)

[ALGORITHMS DESIGN 4](#_Toc13407200)

[USER INTERFACE DESIGN 4](#_Toc13407201)

[REQUIREMENTS TRACEABILITY 5](#_Toc13407202)

[IMPLEMENTATION, INTEGRATION AND TEST PLAN 5](#_Toc13407203)

# INTRODUCTION

## PURPOSE

## SCOPE

## GLOSSARY

### Definition

### Acronyms

### Abbreviation

## REFERENCE DOCUMENT

## DOCUMENT STRUCTURE

# ARCHITECTURAL DESIGN

## OVERVIEW

The goal of this chapter is to analyse and describe the architecture implemented in the **QDocs** application. This chapter is structured as follow:

* SELECTED ARCHITECTURAL STYLE AND PATTERNS: this first paragraph introduces the general architecture used for developing the application, analysing all parts involved and their connection/communication.
* COMPONENT VIEWS: this paragraph provide a more detailed analysis on the class structure of the mobile application, providing all necessary class diagrams and highlighting all the classes’ interactions.
* RUNTIME VIEWS: this paragraph provides a more detailed analysis on the interactions-flows for the more important action that can be performed in the application, such as login, registration, scanning, etc.
* ALGORITHMS DESIGN: this paragraph provides a description of the more significant algorithms implemented in the developing of the application, such as the update list algorithm, etc.
* USER INTERFACE DESIGN: the goal here is to provide the whole lifecycle of the application’s screens through apposite diagrams
* REQUIREMENTS TRACEABILITY: this paragraph provides all **QDocs’s** requirements analysing in detail which are the activities that are in charge to guarantee them.
* IMPLEMENTATION, INTEGRATION AND TEST PLAN: here all information about how the testing is performed are provided.

## SELECTED ARCHITECTURAL STYLE AND PATTERNS

[[ describe the main advantages of client-server arch. ]]

[[ describe patterns used such as MVC, firebase event listener ]]

[[ callback android paradigm, provide the methods lifecycle of each class ]]

[[ abstract pattern (storage adapter) ]]

The **QDocs** application was developed using the 3-tier client-server architecture [Figure 1].

“3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.”

* PRESENTATION TIER: Occupies the top level and displays information related to services that the system can provide. This tier only communicates with the application tier through internet request/response. The Mobile application **QDocs** is in this tier.
* APPLICATION TIER: Also called the middle tier, logic tier, business logic or logic tier. This tier is pulled from the presentation tier. It controls application functionality by performing detailed processing, such as calculations, logical decisions, data and model manipulation. This is basically the server application that provides APIs used by the presentation tier software. It communicates with both presentation tier and data tier
* DATA TIER: Houses storage servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic. For this tier we used the services offered by Firebase: Realtime Database and Cloud Storage.

Immagine che contiene screenshot

Descrizione generata automaticamente

Figure 1: General 3-tier architecture

In the specific case of **QDocs** application the Presentation Tier corresponds to the Android mobile application, the Application Tier to the Firebase server and the Data Tier to the Realtime Database and Storage.

[[ 3-tier for QDOCS image ]]

The following sub-paragraphs will provide more details about how each tier was implemented for the **QDocs** application.

### Server

The server represents the main backend logic, it allows multiple users to interact with it simultaneously and keeping them separated. From the technical point of view, the software was developed using the Firebase development platform providing a set of APIs that hides to the client all the backend logic.

The following list provides all the APIs included in the server:

* AUTHENTICATION API: the server can directly handle the authentication of the users or forward the authentication mechanism to other external services like Facebook and Google, in both cases provides a set of APIs that allows users to make request for registering, logging-in, etc.

It stores all user’s information such as usernames, passwords, personal image and so on.

* CLOUD STORAGE API: since in the server a Cloud Storage service is included, it provides a set of APIs that allows users to interact with their own cloud filed such as upload, download, create new folders and so on.
* REALTIME DATABASE API: since making request directly to the storage may require too much time, the idea was to provide also a Realtime Database service that is queried by the client whenever it want to retrieve static information about the stored files in the cloud, so without updating anything. For this reason the server provide a set of APIs used for listening on database information.
* CLOUD FUNCTIONS API: backend logic used to react on files upload/delete operation and keeping the Realtime DB updated.
* LOGGING: It logs all the interaction between users and the server.

### Storage

The storage layer is provided by the Firebase infrastructure (Google cloud) and in our case is basically composed by two main parts:

* CLOUD STORAGE: this is basically the cloud location where all the users’ files are stored, this is provided by the Google cloud.

[[ storage structure image ]]

* REALTIME DATABASE: not relational database provided by Firebase that is directly associated to the cloud storage and kept congruent with it, it allows a faster access to the stored data for read-only operation.

[[ database structure image, tree ]]

### Client

The client is represented by the Mobile Application itself (**QDocs**), it is a Native mobile app implemented in Java for the Android platform.

From the architectural point of view, it was implemented following the MVC pattern (COLL\_FIGURA)



Figure 2: Model-View-Controller diagram

[[ MVC applied to QDocs ]]

## COMPONENT VIEWS

[[ Class diagrams in details (methods) and their interactions ]]

## RUNTIME VIEWS

[[ Sequence diagrams (more important) showing interaction among activities (and server)]]

# ALGORITHMS DESIGN

[[ provide examples of algorithm implemented ]]

# USER INTERFACE DESIGN

[[ provide whole mobile app lifecycle ]]

# REQUIREMENTS TRACEABILITY

[[ use case diagram ]]

[[ describe which activity is associated to each requirement ]]

# IMPLEMENTATION, INTEGRATION AND TEST PLAN

[[ describe how tests are performed ]]