SDD
System Design Document

English Validation Porting Android

Riferimento	
Versione	1.5
Data	
Destinatario	
Presentato da	
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
29/11/2019	1.0	Aggiunta capitoli 1,2,3,4,5. Revisione del Sommario e controllo ortografico.	Carmine Brancaccio, Antonio Calabrese, Fiorenzo Carino, Michele Costabile, Marco Fresolone, Paolo Schiavo, Stefano Tulino, Simone Veneruso.
29/11/2019	1.1	Correzioni paragrafo 3.2	Michele Costabile
29/11/2019	1.2	Correzioni generali	Marco Fresolone
30/11/2019	1.3	Correzione indentatura	Marco Fresolone
02/11/2019	1.4	Correzione paragrafo 3.3	Paolo Schiavo, Marco Fresolone
02/11/2019	1.5	Correzioni generali	Marco Fresolone

Contents

1. Introduzione

1.1 Obiettivi del sistema

Il sistema che il team intende creare ha come scopo il porting della già presente Web App di EV English Validation, cioè creare un'applicazione per dispositivi Android che abbia le stesse funzionalità dell'applicazione web e che, allo stesso tempo, sfrutti le potenzialità del sistema operativo Android.

L'obiettivo del sistema è dunque realizzare un'app che sfrutti il più possibile le feature di sistema e che consenta a qualunque utente in possesso di un dispositivo Android che necessiti dei servizi di EV English Validation di poterne usufruire semplicemente accedendo all'applicazione qualsiasi sia il suo ruolo, dal responsabile di dipartimento (amministratore di sistema) allo studente che richiede la validazione di un certificato passando per gli impiegati dell'ufficio di segreteria di competenza.

Come già detto, essendo l'applicazione proposta un porting di una Web App già esistente, offrirà le stesse funzionalità di base della stessa:

- Login e logout al sistema (per ogni tipologia di utente utilizzatore)
- Registrazione al sistema
- Possibilità di effettuare l'upload di documenti affinché possano essere validati (Utente studente)
- Visualizzazione dello stato della richiesta (Utente studente)
- Aggiunta dei CFU (Utente segreteria)
- Approvazione o rifiuto delle richieste degli studenti (Utente Admin)

Il sistema sarà dunque in grado di gestire i vari documenti (PDF e Excel) e manipolare i dati presenti su di essi, inviare notifiche push in tempo reale all'utente per avvisarlo, ad esempio, del cambio di stato della sua richiesta e sarà il più possibile user-friendly, semplice e intuitivo in modo che ogni utente possa agevolmente utilizzare l'app e fruire delle funzioni.

1.2 Design Goals

I Design Goals sono organizzati in cinque categorie: Performance, Affidabilità, Costo, Manutenzione, End User Criteria. I Design Goals identificati dal team sono:

ID	Descrizione	Categoria
DG1_TempoDiRisposta	Ogni utente deve poter utilizzare rapidamente e fluidamente l'applicazione per usufruire dei servizi offerti, dunque sono necessari tempi di risposta più brevi possibile	Performance
DG2_Memoria	L'app deve essere il più possibile compatta, in modo da non intaccare eccessivamente lo spazio di archiviazione dei dispositivi degli utenti	Performance
DG3_Throughput	Il sistema sarà capace di gestire contemporaneamente diversi	Performance

	utenti. In caso di elevato numero di accessi concorrenti, il sistema mostrerà, all'utente che tenta di autenticarsi, un messaggio che li inviti ad accedere in un altro momento.	
DG4_Robustezza	Il sistema informerà l'utente di eventuali errori nel caso di immissione di input non validi attraverso degli appositi messaggi.	Affidabilità
DG5_Sicurezza	Il sistema deve garantire la sicurezza dei servizi proposti. Il prodotto software sarà sviluppato in modo tale da controllare accuratamente le informazioni inserite in input dagli utenti che inseriranno le rispettive email e password per accedere.	Affidabilità
DG6_Disponibilià	Il sistema sarà disponibile all'utente in qualsiasi momento, in qualunque momento potranno accedere all'applicazione e usufruire dei servizi offerti	Affidabilità
DG7_CostiDiSviluppo	È stimato un costo complessivo di 350 ore per la progettazione e lo sviluppo del sistema (50 ore per ogni componente del team)	Costi
DG8_Estensibilità	È possibile aggiungere nuove funzionalità al sistema, dettate dalle esigenze del cliente o dall'avvento di nuove tecnologie.	Manutenzione
DG9_Adattabilità	Il sistema funziona solo per il dipartimento di informatica e per l'Università degli studi di Salerno, ma è adattabile a più dipartimenti o a più università modificando i relativi controlli.	Manutenzione
DG10_Portabilità	L'interazione con il sistema avviene attraverso un'app Android che funziona su ogni versione del sistema operativo quindi possiamo definirlo portabile poiché il sistema è accessbile da qualunque dispositivo che abbia incorporato una versione di Android.	Manutenzione
DG11_Tracciabilità	Utilizzo di una matrice di tracciabilità.	Manutenzione
DG12_Usabilità	Il sistema sarà di facile comprensione e utilizzo, permettendo di effettuare in modo semplice e immediato le varie operazioni grazie a un'interfaccia user-friendly.	End-User

1.2.1 Design Trade-Off

Memoria vs Estensibilità: Il sistema deve permettere l'estensibilità a discapito della memoria utilizzata. Tale preferenza permette al cliente di richiedere agli sviluppatori nuove funzionalità, dando meno importanza alla memoria utilizzata.

Tempo di risposta vs Affidabilità: Il sistema sarà implementato in modo tale da preferire l'affidabilità al tempo di risposta, in modo tale da garantire un controllo più accurato dei dati in input a discapito del tempo di risposta del sistema.
Disponibilità vs Tolleranza ai guasti: Il sistema deve sempre essere disponibile all'utente in caso di errore in una funzionalità, anche al costo di rendere non disponibile quest'ultima per un lasso di tempo.
Criteri di manutenzione vs Criteri di performance: Il sistema sarà implementato preferendo la manutenibilità alla performance in modo da facilitare gli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

1.3 Definizioni, acronimi e abbreviazioni

- **EV:** English Validation.
- **RAD:** Requirements Analysis Document.
- **SDD:** System Design Document.
- **USER-FRIENDLY:** Letteralmente "amichevole per l'utente", di facile utilizzo anche per chi non è esperto.
- **DB:** DataBase.
- **NO-SQL:** che sta per "non solo SQL" è un'alternativa ai tradizionali database relazionali in cui i dati sono collocati in tabelle e lo schema dei dati è progettato attentamente prima della creazione del database.
- **APP:** Applicazione sviluppata per dispositivi mobili.

1.4 Riferimenti

- Libri:
  • Kathy Schwalbe, "Information Technology Project Management", International Edition 7E, Cengage Learning, 2014

  • Bernd Bruegge, Allen H. Dutoit, "Object-Oriented Software Engineering Using UML, Patterns and Java", Third Ed., Pearson, 2010;

- Guide:
  Tutorial per l'utilizzo di FireBase.
  Documentazione degli strumenti sviluppatore per Android
  (https://developer.android.com/docs)

- Slide:
  - Slide delle lezioni della Prof.ssa F. Ferrucci e del Prof. C. Gravino
  - Slide delle lezioni del corso di Android.

1.5 Panoramica

Il documento si compone di una prima parte in cui vengono introdotti gli obiettivi di design. Di seguito l'architettura del sistema corrente e l'architettura del sistema proposto. Composizione del documento:

- Nel capitolo 2 viene mostrata l'architettura del sistema corrente;
- Nel capitolo 3 viene mostrata l'architettura del sistema proposto, vengono inoltre mostrati in dettaglio:
  - Divisione del sistema in sottosistemi di funzionalità e operazioni correlate da assegnare a vari membri del team in fase di sviluppo;

- Mapping Hardware/Software, riguardante la scelta della configurazione hardware del sistema, le componenti off-the-shelf e incapsulamento dei servizi di un sottosistema;
- Gestione dei dati memorizzati dal sistema e l'infrastruttura di gestione richiesta per memorizzarli;
- Controllo del flusso globale descrive quali operazioni eseguire ed in che ordine, per garantire il corretto flusso di controllo del sistema;
- Condizioni boundary per la gestione dei fallimenti dovuti alla chiusura forzata, o anche a errori di progettazione.

- Nel capitolo 4 vengono mostrati i servizi forniti da ogni sottosistema ed individua i loro limiti.
- Nel capitolo 5 viene fornito il glossario dei termini utilizzati nel documento con le relative definizioni.

2. Architettura del Sistema corrente

Il prodotto software a cui stiamo facendo riferimento è la Web Application di EV English Validation, software con la finalità di convalidare i CFU dell'esame di lingua inglese con la richiesta diretta dello studente verso l'Università degli Studi di Salerno. L'utente nel sistema corrente deve collegarsi mediante browser alla pagina dedicata, loggarsi o registrarsi all'interno del sistema, scaricare il modulo per la richiesta, compilarlo scannerizzarlo e inviare la richiesta di convalida che sarà inoltrata in segreteria che, una volta accertatasi della correttezza del documento tramite consultazione con l'ente esterno, lo passerà all'admin per approvarlo o, in caso di incongruenze, rifiutarlo. Il sistema che vogliamo realizzare dalla base di questo software permetterà di semplificare e velocizzare tutto l'iter aggiungendo inoltre nuove funzionalità, tutto sotto forma di applicazione per piattaforma Android.

3. Architettura del Sistema proposto

3.1 Panoramica

L'architettura scelta per questo sistema è il Model-View-Controller (MVC). Essa si struttura in tre componenti, ognuno con un compito diverso all'interno del sistema:

• il model fornisce i metodi per accedere ai dati utili all'applicazione;
• il view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
• il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

3.1.1 Vantaggi del Modello MVC

•MVC supporta lo sviluppo rapido e parallelo. Se un modello MVC viene utilizzato per sviluppare una particolare applicazione, è possibile che un programmatore possa lavorare su una view mentre un altro su un controller per creare la logica di business dell'applicazione. Quindi, in questo modo, l'applicazione sviluppata utilizzando il modello MVC può essere completata tre volte più velocemente rispetto alle applicazioni sviluppate utilizzando altri modelli di sviluppo;

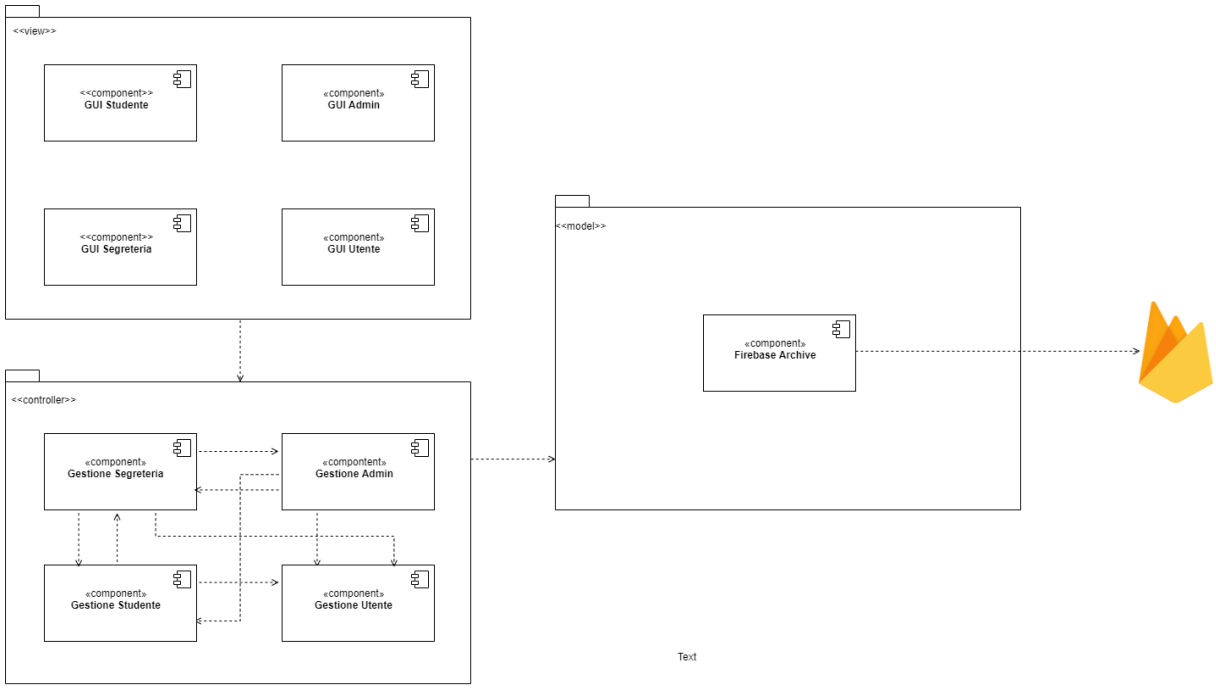
• Nel modello MVC, è possibile creare più view per un model. Inoltre, grazie a questo modello, la duplicazione del codice è molto limitata perché vengono separati i dati e la logica di business dalle interfacce grafiche;

• Per qualsiasi applicazione, l'interfaccia utente tende a cambiare molto velocemente. Vengono apportati di continuo modifiche al proprio sistema, come cambiamenti nei colori, caratteri, layout dello schermo. L'aggiunta di un nuovo tipo di view è molto semplice con il modello MVC poiché la parte Model non dipende dalla parte View. Pertanto, eventuali modifiche nel Model non influenzeranno l'intera architettura;

3.1.2 Svantaggi del Modello MVC

• Modello di design complesso. Ha bisogno di una buona conoscenza del flusso di controllo tra view, logica di business e controller. Altrimenti sarà difficile eseguire il debug;
• Se diversi team lavorano su classi UI e Model, allora devono avere lo stesso ritmo di sviluppo;
• Il processo di sviluppo isolato per la UI, la logica di business e i controller può portare a ritardi nello sviluppo dei rispettivi moduli.

3.2 Decomposizione in sottosistemi



Il livello View prevede la gestione di quattro sottosistemi e possiamo identificarli come oggetti boundary individuati nel rad :
•GUI Studente
•GUI Segreteria
•GUI Admin
•GUI Utente

Il livello Control prevede la gestione di quattro sottosistemi :

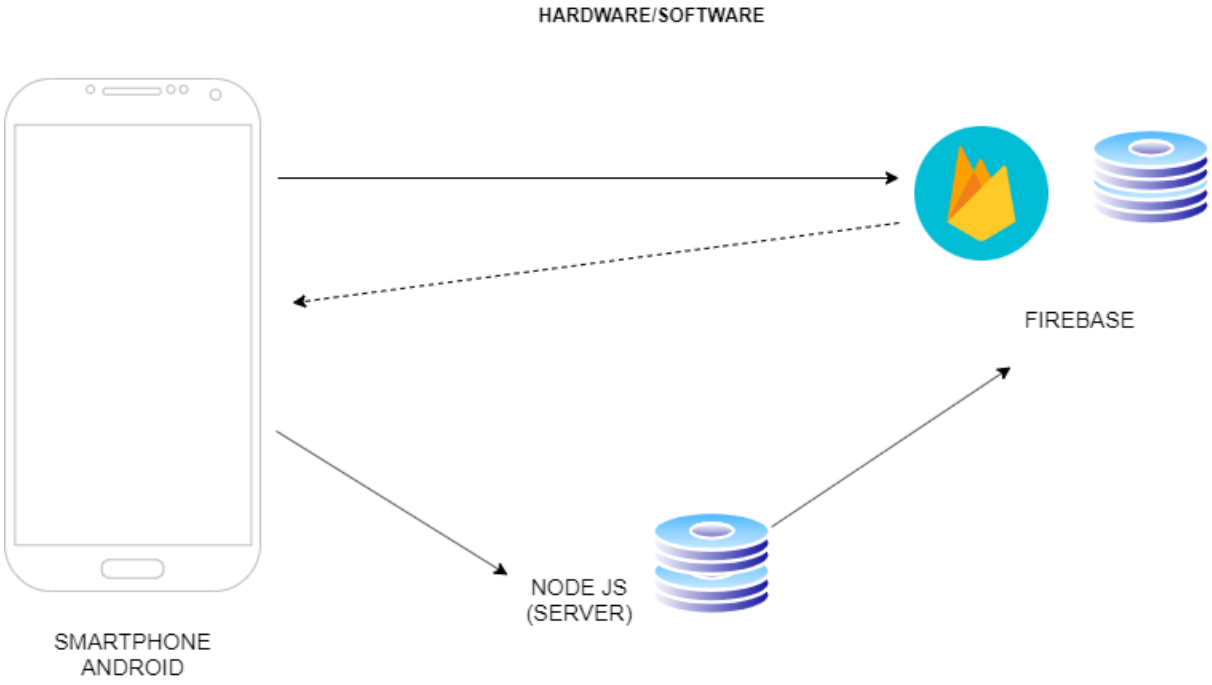
•Gestione Admin
La Gestione Admin si occupa della visualizzazione dei certificati e relative caratteristiche come : approvati, rifiutati, in attesa di revisione e di richiesta di esonero.
Inoltre permette all'attore Admin di : Controllare errori presenti nel certificato, Inviare email di correttezza, ricercare un certificato, relativo ad uno studente, per nome e controllare il codice di un certificato per assicurarsi la sua validità.
•Gestione Utente
La Gestione Utente si occupa del supporto multilingue, del login tramite impronte digitali della

visualizzazione e della modifica del profilo.
•Gestione Segreteria
La Gestione Segreteria si occupa della visualizzazione lista richieste, dell'aggiunta dei CFU, dell'invio esito della richiesta, della correzione errori, della lettura QR Code della lista richieste e del ricevere notifiche.
•Gestione Studente
La Gestione Studente si occupa dell' autenticazione dello studente ed interagisce con gli altri sistemi, offrendo vari servizi al suo interno, quali Compilazione Form, varie notifiche inviate allo studente e servizi inerenti ai vari documenti generati/inviati.

Il livello Model prevede la gestione di un sottosistema
•Firebase Archive

3.3 Mapping Hardware/Software

Il sistema che si desidera sviluppare sarà costituito dal Server che risponderà ai servizi richiesti dai client. Il client è un qualsiasi dispositivo mobile con Android attraverso il quale un utente (Studente, Segretario/a, Admin) può collegarsi utilizzando una connessione ad Internet.
Il server Firebase dove sono presenti i dati persistenti effettua uno scambio HTTP rest costante, per avere tutte le informazioni necessarie tra client Android e Firebase.
Node js viene utilizzato solo ed esclusivamente per generare file PDF ed Excel, poiché non lo si può mediante Android, verrà utilizzato quindi per generare i file che verranno comunicati ed inviati a Firebase.
Il client e il server saranno connessi tramite il protocollo HTTP con risposta in json :
il client inoltra delle richieste del server e quest'ultimo provvede a fornire i servizi richiesti.



3.3.1 Componenti legacy and off the shelf

Dato che il sistema nasce ex novo non presenta componenti legacy.

Per la gestione dei dati utente e delle richieste ci si affiderà a un componente off the shelf, Firebase, un'applicazione mobile e web sviluppata dall'omonima azienda, acquistata successivamente da Google; quest'ultima presenta varie funzionalità di cui noi utilizzeremo Firebase Authentication, Cloud Firestore e Firebase Realtime Database.

3.4 Gestione dati persistenti

3.4.1 Struttura delle collezioni

Di seguito sono riportate tutte le tabelle che vanno a formare il nostro database per la gestione di tutte le informazioni del sistema. In ogni tabella è indicato: il nome, il compito e tutti gli attributi ad essa associati, con relativi vincoli e tipo.

USER		
Campo	Vincoli	Tipo
email	Lunghezza massima: 50 caratteri;	String
name	Lunghezza massima: 50 caratteri;	String
surname	Lunghezza massima: 50 caratteri;	String
sex	Lunghezza massima: 1 carattere;	String
password	Lunghezza massima: 50 caratteri;	String
user_type	Lunghezza massima: 1 caratteri;	Number
profilePicture	Lunghezza massima: 400 caratteri;	String

ENTE		
Campo	Vincoli	Tipo
email	Lunghezza massima: 50 caratteri;	String
name	Lunghezza massima: 100 caratteri;	String
site	Lunghezza massima: 50 carattere;	String
state	Lunghezza massima: 4 caratteri;	Number

REQUEST		
	Vincoli	Tipo
level	Lunghezza massima: 7 caratteri;	String
release_date		Timestamp
expiry_date		Timestamp

year	Lunghezza massima: 4 caratteri;	String
requested_cfu	Lunghezza massima: 2 caratteri;	Number
serial	Lunghezza massima: 10 caratteri;	Number
validated_cfu	Lunghezza massima: 2 caratteri;	Number
filename	Lunghezza massima: 50 carattere;	String
qr_code	Lunghezza massima: 4296 carattere;	String

3.5 Controllo degli accessi e sicurezza

L' applicazione Android di English Validation permette l' accesso a 3 tipi di utenti differenti: admin, segreteria e studenti.
Per poter effettuare l'accesso all'applicazione qualunque tipo di utente precedentemente indicato dovrà inserire le proprie credenziali utilizzate nella registrazione (email e password) , in modo tale da poter svolgere le azioni descritte per ogni singolo attore.

Attori	Gestione Utente	Gestione Admin	Gestione Studente	Gestione Segreteria
Admin	Supporto multilingue Autenticazione tramite impronta digitale Visualizzazione profilo Modifica profilo	Visualizzazione lista richieste Verifica certificato tramite codice Approvare o rifiutare certificazioni e organizzarle in due liste separate Correzione errori certificato Invio email di Verifica	Invio email valutazione richiesta Ricerca certificato studente per nome	

		certificazione Visualizza numero di documenti non ancora revisionati		
Studente	Registrazione Supporto multilingue Autenticazione tramite impronta digitale Visualizzazione profilo Modifica profilo		Compilazione form Caricamento allegati Accesso area utente Notifica stato della pratica Visualizza QR code della pratica	Invio notifica allegati

Segreteria	Supporto multilingue Autenticazione tramite impronta digitale Visualizzazione profilo Modifica profilo	Inviare esito richiesta	Inviare esito richiesta	Visualizza lista richieste Aggiungi CFU Inviare esito richiesta Correzione errori Lettura QR Code Segreteria Notifica Segreteria

3.6 Controllo flusso globale del sistema

Il flusso del sistema Porting Android fornisce una serie di funzionalità che richiedono una continua interazione da parte dell'utente, rendendo il controllo del flusso globale del sistema di tipo eventdriven, ovvero guidato dagli eventi.

3.7 Condizione limite

- **Start-Up**: Per il primo start-up del sistema EV English Validation è necessario l'avvio di un server in cloud di Google che fornisca il servizio di un Database Firebase per la gestione dei dati persistenti. Inoltre, c'è bisogno di avviare il server Node.js per la gestione degli allegati. In seguito, tramite l'interfaccia di Login, sarà possibile autenticarsi tramite opportune credenziali (e-mail e password) oppure tramite impronta digitale. Una volta effettuato l'accesso, EV English Validation presenterà all'utente la propria Home, dalla quale sarà possibile usufruire di tutte le operazioni che la piattaforma offre.

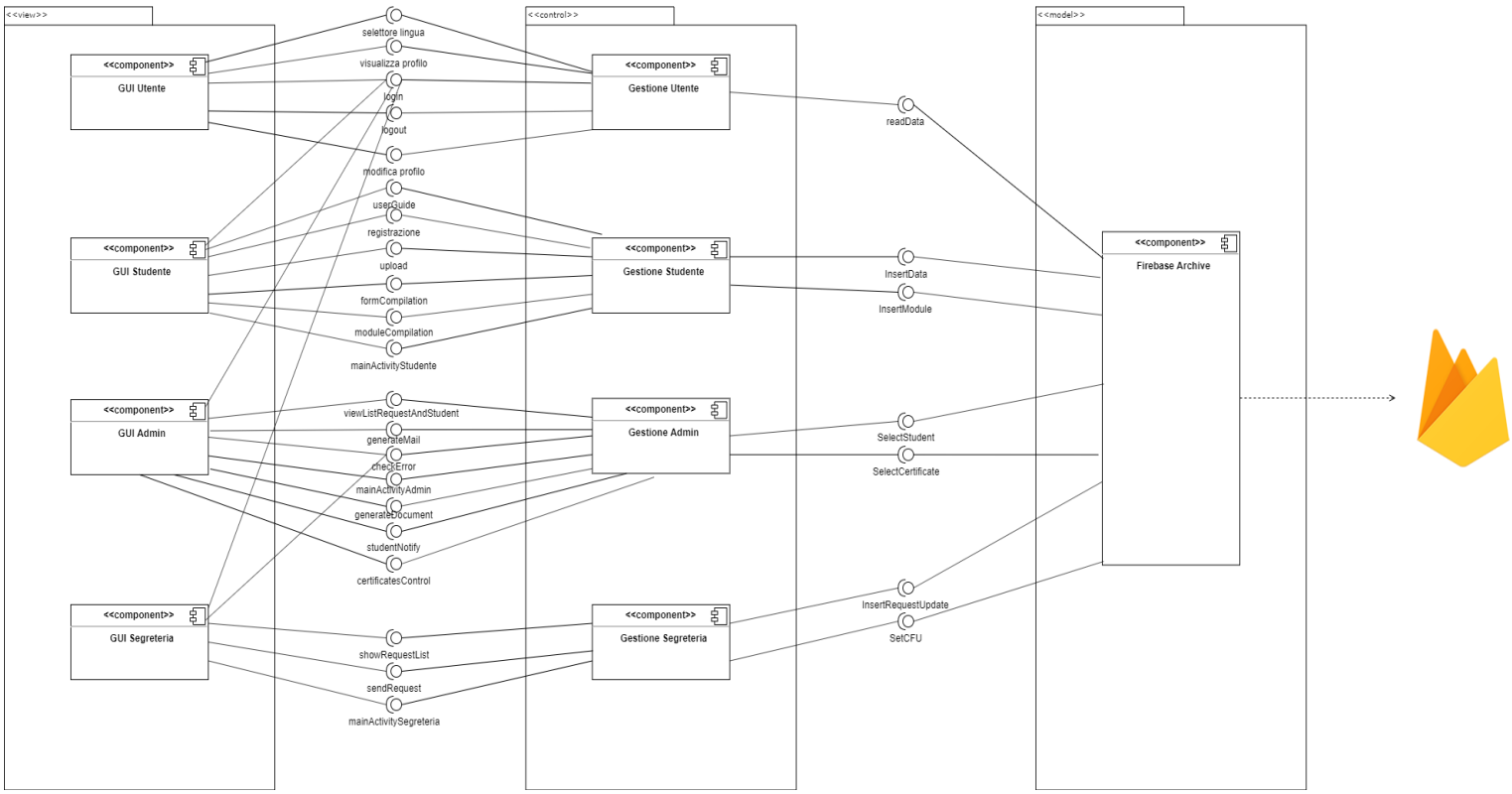
- **Shutdown**: Al momento della corretta chiusura dell'applicazione, si ha la terminazione del sistema con un regolare Log-out. Per consentire la corretta terminazione dei server Node.js e Firebase, l'amministratore del sistema dovrà effettuare la procedura di terminazione, dopo la quale nessun client potrà connettersi al sistema.

- **Fallimento**:

    1. Nel caso di guasti dovuti al sovraccarico del database Firebase con successivo fallimento dello stesso è prevista come procedura preventiva il salvataggio periodico dei dati sotto forma di codice Json per la successiva rigenerazione del DB.
    2. Nel caso in cui si verifichi un'interruzione inaspettata del dispositivo non sono previsti metodi che ripristino lo stato del Sistema precedente allo spegnimento non voluto.

3. Un altro caso di fallimento potrebbe derivare dal software stesso che causa una chiusura inaspettata dovuta ad errori commessi durante la fase di implementazione. Non essendo previste politiche correttive, l'unica operazione consentita in questa particolare situazione è la chiusura del sistema e il suo successivo riavvio.
4. Un altro caso di fallimento potrebbe essere dovuto ad un errore critico nell'hardware, contro il quale non è prevista alcuna contromisura.

4. Servizi dei sottosistemi



4.1 View

- GUI STUDENTE
  - •Registrazione
  - •Login
  - •Compilazione form
  - •Visualizzazione Profilo

- GUI SEGRETERIA
  - •Login
  - •Visualizzazione lista richieste

- GUI ADMIN
  - •Login
  - •Visualizzazione lista richieste
  - •Genera la mail
  - •Pagina correzione errori

- •Pagina principale dell'admin
- •Notifica lo studente in caso di errori

- GUI UTENTE
  - •Login
  - •Selezione della lingua del sistema

4.2 Control

- GESTIONE STUDENTE
  - •Generazione PDF
  - •Caricamento PDF
  - •Autenticazione con impronta
  - •Modifica profilo
  - •Visualizzazione della pratica

- GESTIONE SEGRETERIA
  - •Autenticazione con impronta
  - •Aggiunta CFU
  - •Lettura QRCode
  - •Manda una nuova richiesta all'admin

- GESTIONE ADMIN
  - •Autenticazione con impronta
  - •Verifica pratica con codice
  - •Generazione Excel
  - •Controlla i certificati con l'ente associato
  - •Ricerca certificati

- GESTIONE STUDENTE
  - •Modifica Profilo
  - •Autenticazione con impronta digitale
  - •Logout
  - •Visualizzazione guida utente

4.3 FireBase Archive

Gestione dei dati

5. Glossario

- **Utente:** rappresenta l'utilizzatore del sistema.
- **Studente:** rappresenta un utente registrato e autenticato, che può interagire con il sistema ed effettuare diverse operazioni di convalida del certificato.
- **Admin:** rappresenta un utente autenticato, amministratore, che effettua operazioni nel sistema, interagendo in alcuni casi anche con lo studente e la segreteria per effettuare operazioni di certificato.

- **Segreteria:** rappresenta un utente autenticato, che effettua operazioni di gestione dei certificati ed in alcuni casi può interagire con lo studente e l'admin, per il completamento delle operazioni.
- **Node.js:** piattaforma Open source event-driven che consente l'esecuzione di codice JavaScript server side.
- **HTTP:** protocollo di trasferimento di ipertesti che consente a due macchine, client e server, di interagire attraverso un meccanismo di richiesta/risposta.
- **Client:** componente che richiede e accede a servizi e risorse offerte dal server.
- **Server:** componente che offre servizi e risorse al client.
- **Model:** componente del modello MVC che si occupato dall'interazione tra l'applicazione e il database.
- **View:** componente del modello MVC dell'utente che mostra all'utente le possibili interazioni col sistema.
- **Controller:** componente del modello MVC che si occupata di elaborare le richieste di un utente e di comunicare con il model.
- **MVC:** modello architetturale che si basa sull'utilizzo di 3 componenti fondamentali (model,view e controller) per lo sviluppo di applicazioni android.
- **Android:** sistema operativo per sviluppo di applicazioni cellulari, che permette all'utente di interagire con il sistema per lo svolgimento di diverse operazioni.
- **JSON:** JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati che permette all'utente di interagire con il sistema per lo svolgimento di operazioni.

- **FireBase:** si tratta di un backend, un servizio offerto da Google per semplificare la gestione dei dati per le applicazioni Android.