

*Relatore*



DedicaDedica

SommarioSommario

\*Sommario

Ringraziamenti

\*Ringraziamenti

,

tableofcontents

lof

lot

Lista degli elementi di codice  
lof  
lot

## Contesto aziendale

Il presente capitolo introduce l'azienda presso cui ho svolto l'attività di *stage*, **SyncLab S.r.l.**, fornendo la base necessaria per comprendere il suo contesto aziendale. Presentazione azienda L'azienda ospitante, *SyncLab S.r.l.*, è stata fondata nel 2002 a Napoli e si è affermata fin da subito come una delle aziende leader nel settore della soluzioni software innovative. Inizialmente nata come *software house*, l'azienda si è dedicata allo sviluppo di soluzioni *software* innovative, progettate per soddisfare le esigenze di diversi settori.

Nel tempo, *SyncLab S.r.l.* ha ampliato il proprio raggio d'azione, assumendo un ruolo rilevante come *system integrator*. La duplice identità coniuga la creatività e la proattività di una *software house* con l'approccio orientato all'efficienza e alla qualità tipico di un'azienda di servizi. *SyncLab S.r.l.* promuove attivamente la collaborazione interna, incoraggiando l'interazione non solo tra i membri della stessa azienda, ma anche con i clienti e i partner.

Rami aziendali e progetti *SyncLab S.r.l.* collabora con un ampio numero di clienti appartenenti a molteplici settori industriali. Alcuni dei *software* che l'azienda ha prodotto sono:

**Sobereye** (ambito *web*): un'applicazione innovativa progettata per monitorare il rischio di deterioramento neuro-cognitivo.

**SynClinic** (ambito sanitario): un sistema integrato che supporta la gestione completa dei processi clinici e amministrativi.

**DPS 4.0** (ambito *web* e *privacy*): una piattaforma *web* che supporta i titolari, responsabili, *data protection officer* (DPO).

**Fast Ride** (ambito trasporti): una soluzione per la gestione di servizi di trasporto pubblico a chiamata, in contesti urbani e extraurbani.

**Way of working** In questa sezione tratterò di alcune tecnologie di cui ho avuto esperienza diretta per lo sviluppo dei progetti.

**Dart**: linguaggio di programmazione orientato agli oggetti sviluppato da Google, noto per la sua versatilità. Offre numerosi strumenti per lo sviluppo di un applicativo *cross-platform* che consente di creare applicazioni native per diversi sistemi da un'unica base di codice. La compilazione *Ahead-Of-Time (AOT)* che accelerano lo sviluppo.

**Flutter**: un *framework* basato su Dart per lo sviluppo di applicazioni multi-piattaforma, che offre vantaggi come:

**Hot-reload**: permette di visualizzare immediatamente le modifiche al codice durante lo sviluppo senza dover aspettare la compilazione.

**Compatibilità con material**: mette a disposizione un ricco arsenale di *widget* per creare interfacce utente moderne e intuitive.

**Firebase**: un *database* sviluppato da Google e ben integrato con Flutter che offre una serie di funzionalità di *backend* per applicazioni mobile.

**Authentication**: servizio che offre i servizi di *backend* e librerie pronte all'uso per autenticare gli utenti nell'applicazione.

**Firestore**: *database NoSQL* orientato ai documenti, che permette di archiviare i documenti in raccolte, le quali fungono da collezioni.

Per garantire continuità operativa e una efficace gestione dei processi collaborativi, è necessario disporre di strumenti di supporto.

**Git**: un sistema di controllo versione distribuito che permette di tracciare in modo efficiente le modifiche ai file e di coordinare le attività di sviluppo.

**Android Studio**: è un ambiente di sviluppo integrato (*IDE*) gratuito, progettato per lo sviluppo di applicazioni **Android**.

**Xcode**: è un ambiente di sviluppo integrato (*IDE*), sviluppato e mantenuto da Apple che contiene una *suite* di strumenti per lo sviluppo di applicazioni **iOS**.

**UMLet**: è uno strumento gratuito e *open source* che permette di creare diagrammi UML<sup>1</sup>. In particolare diagrammi di:

*Smart working* e strumenti di comunicazione L'azienda adotta un modello di lavoro prevalentemente da remoto: la  
Nel mio caso, trattandosi di un progetto di *scouting* tecnologico, gli incontri con il responsabile designato sono stati

Per garantire una gestione efficace delle attività e tracciare correttamente l'evoluzione del progetto, ho utilizzato diversi

**Google Calendar:** un calendario condiviso che permette la creazione e la modifica di eventi, specificandone durata e lu-

**GitHub Projects:** una sezione di GitHub dedicata alla gestione dei *ticket*, preferita in alternativa a **Trello** normalmen-

Per la comunicazione interna durante il lavoro da remoto:

**Discord:** utilizzato dai dipendenti per lo scambio di informazioni tramite *chat* testuali e vocali. La piattaforma funge a

[!ht] [width=0.8]..../assets/sviluppoComunicazione.png Strumenti di comunicazione

Spirito di innovazione aziendale

L'azienda *SyncLab* si caratterizza per una forte propensione all'innovazione, elemento che guida in modo significativo

**Miglioramento delle dinamiche interne:** l'azienda investe costantemente nell'ottimizzazione dei processi di gestione

**Proposizione di soluzioni d'avanguardia:** L'impegno innovativo non si limita all'organizzazione interna, ma si estende

In questo contesto, orientato alla ricerca dell'avanguardia si è collocata la mia esperienza di *stage*. La vocazione aperta a

[!ht] [width=0.75]..../assets/progettoStage.png Progetto di *stage* come intersezione tra le aree di competenze consolidate

Il progetto si è inserito coerentemente nella strategia aziendale, contribuendo su due fronti principali:  
**Scouting tecnologico:** L'utilizzo di Flutter per la realizzazione di un'applicazione, caratterizzata da requisiti in ambito tecnologico.  
**Adattamento di competenze consolidate:** sebbene *SyncLab* possieda già un panorama della sicurezza e dello sviluppo di applicazioni.  
L'approccio fondato su posizioni all'avanguardia di *SyncLab* ha rappresentato quindi un motore concreto che ha reso possibile l'adattamento delle competenze consolidate.  
Visione aziendale Offerte aziendali Per *SyncLab*, i programmi di *stage* rappresentano uno strumento strategico per il recrutamento di talenti.

[!ht] [width=]..../assets/collab.png Collaborazioni dell'azienda - Fonte: synclab.it

I programmi di *stage* si articolano principalmente in tre aree:

**Integrazione:** gli stagisti contribuiscono al perfezionamento di *software* già in uso, intervenendo su funzionalità specifiche.

**Analisi e ottimizzazione:** questa area prevede una valutazione approfondita delle soluzioni *software* esistenti, con l'obiettivo di ottimizzarle.

**Innovazione:** in questo ambito gli stagisti svolgono analisi teoriche e sperimentazioni su tecnologie emergenti, con l'obiettivo di contribuire alla ricerca e allo sviluppo.

[!ht] [width=0.8]..../assets/stage.png Contributo dei programmi di *stage* in SyncLab.

*Stage* in azienda Gli *stage* presso *SyncLab* rappresentano un investimento strategico bidirezionale, che riflette la duplice natura dell'esperienza di stage.

**Formazione e inserimento di risorse qualificate:** Il programma di *stage* funge da canale primario per identificare e formare nuovi talenti.

**Acquisizione di nuove competenze:** Il programma di *stage* funge da banco di prova per l'innovazione continua, riducendo i rischi di investimento.

[!ht] [width=0.8]..../assets/cap2Stage.png Investimento strategico degli *stage* per l'azienda

Ruolo del *tutor* aziendale Durante il percorso ogni stagista è affidato a un *tutor*. Il *tutor* è una figura professionale che guida il *stage*.

**Guida:** il *tutor* propone linee guida su come organizzare il lavoro e fornisce allo stagista materiali utili che possono aiutare a superare le difficoltà.

**Supporto:** il *tutor* offre una serie di suggerimenti o soluzioni possibili ai problemi incontrati, questo per scongiurare l'errore.

**Supervisore:** il *tutor* offre *feedback* allo stagista nella valutazione del lavoro svolto per verificarne la qualità e supportarne lo sviluppo.

Nel mio caso i dialoghi con il *tutor* assegnato sono stati molto importanti in quanto senza il suo confronto, e i suoi feedback, non avrei potuto ragionare su come procedere.

Motivazione dello *stage* L'obiettivo dello *stage* è stato la valutazione di fattibilità e il potenziale impiego di nuove tecnologie.

[!ht] [width=0.8]..../assets/codebaseUnica.png Confronto tra lo sviluppo nativo e l'approccio *cross-platform* con *Flutter*.

Progetto proposto Nel contesto attuale, i dispositivi mobili rappresentano uno strumento di comunicazione ampiamente utilizzato.

Per tale motivo, l'azienda ha proposto la progettazione e l'implementazione di un'applicazione in grado di generare e gestire dati.

La realizzazione del progetto ha comportato l'identificazione e la risoluzione di alcune sfide progettuali e tecnologiche quali:

\*Gestione delle chiavi crittografiche Il recupero della chiave privata usata per decifrare un messaggio rappresenta un problema critico.

[!ht] [width=0.9]..../assets/cap2crittografia.png Esempio di comunicazione critptata

\*Integrazione con il lettore NFC L'applicazione deve supportare l'utilizzo del lettore NFC presente sui dispositivi mobili.

Obiettivi e vincoli Obiettivi Il progetto proposto è stato guidato da una serie di obiettivi volti a garantirne il successo.

**Studio di fattibilità:** effettuare uno studio dettagliato e un'analisi approfondita sulla fattibilità tecnica dell'utilizzo di tecnologie esistenti.

l'integrazione con il *server Firebase*;

il funzionamento del lettore NFC;

la generazione di una coppia di chiavi (pubblica e privata) crittografiche;

Questo per permettere di valutare le potenzialità e i limiti degli strumenti, in modo da individuare i migliori da utilizzare.

**Sviluppo di una applicazione:** per verificare che i componenti applicativi funzionino bene tra di loro, e che permettano di raggiungere gli obiettivi.

**Vincoli tecnologici:**

**Utilizzo obbligatorio del framework Flutter:**

L'intera applicazione deve essere sviluppata interamente tramite *Flutter*, con il conseguente utilizzo del linguaggio *Dart*.

**Utilizzo del piano gratuito di Firebase:**

L'uso di *Firebase* presenta alcune limitazioni derivanti dal piano gratuito, che esclude funzionalità specifiche presenti solo nel piano premium.

**Vincoli architettura di sicurezza:**

**Protezione della chiave privata:**

L'architettura deve impedire la trasmissione *online* della chiave privata, garantendo però un meccanismo sicuro che consente la sua utilizzazione.

**Associazione delle chiavi ai documenti:**

Ogni coppia di chiavi deve essere associata a un documento, e non a *tag NFC* generici.

**Unicità del wallet per documento e account:**

Non deve essere consentita la creazione di più *wallet* associati allo stesso documento per un medesimo *account*. Preservando l'unicità del *wallet*.

[!ht] [width=0.8]..../assets/dualitàChiavi.png Schema che rappresenta l'architettura di sistema, la chiave blu corrisponde alla chiave privata.

Scelta dello *stage* Motivazioni della scelta Ho conosciuto *SyncLab* nel mese di aprile attraverso una comunicazione telematica.

Parallelamente, durante il periodo estivo avevo ricevuto ulteriori offerte di stage da altre aziende e stavo valutando le loro proposte.

Alla fine ho scelto di intraprendere il mio percorso in *SyncLab* per i seguenti motivi:

**Possibilità di lavorare da remoto**, condizione per me fondamentale poiché, dovendomi spostare frequentemente e non avere un ufficio fisico.

**Elevato livello di autonomia offerto dal progetto**, trattandosi di uno *scouting* tecnologico finalizzato alla valutazione delle competenze.

Obiettivi personali prefissati Nell'attuale contesto tecnologico, l'uso dei dispositivi mobili è divenuto parte integrante della vita quotidiana.

L'introduzione della proposta di legge europea relativa al controllo delle *chat* e alla possibilità di analizzare i dati dei propri contatti.

Da qui è nata la domanda che ha guidato parte della mia ricerca: esiste un modo per conservare le chiavi private in modo sicuro?

Nello sviluppo dell'applicazione mi sono quindi posto i seguenti obiettivi:

Progettare un'applicazione capace di gestire i dati privati degli utenti in modo distinto rispetto a quelli pubblici, incrementando la sicurezza.

Apprendere lo sviluppo di applicazioni per dispositivi mobili attraverso l'utilizzo del framework *Flutter*.

Acquisire i principi di crittografia necessari a consentire una comunicazione sicura tra due dispositivi mobili.

Per quanto qui permesso, lo *stage* presso *SyncLab* costituiva per me un'importante occasione per approfondire i concetti di base.

Attività	Settimane Ore						
	1	2	3	4	5	6	7
Ripasso costrutti di Java	X						5
Studio di Dart	XX						30
Studio di Flutter		XX					40
Studio algoritmi di criptazione		X	X	X			30
Analisi del problema	X						10
[H] Progettazione della piattaforma		X				25	Pianificazione del lavoro durante le 8 settimane
Sviluppo maschera di <i>login</i>	X					5	
Sviluppo di un prototipo che genera chiavi		X				30	
Sviluppo applicazione finale		XXX	100				
Stesura finale della specifica tecnica		XX	20				
<i>Live demo</i> e presentazione finale			X	5			
<b>totale ore</b>					300		

Durante le prime settimane di *stage*, è stata applicata una metodologia a cascata, essenziale per stabilire le fondamenta del progetto.

Una volta completata la prototipazione, il progetto è transitato verso una metodologia agile, in particolare durante la realizzazione del prodotto.

L'adozione quindi dell'approccio ibrido ha garantito la copertura formativa e analitica iniziale, consentendo al contemporaneo di acquisire conoscenze specifiche.

Analisi dei requisiti La fase di analisi svolta durante le prime settimane del percorso di *stage*, sono state cruciali per la definizione delle specifiche del prodotto.

Requisiti funzionali I requisiti funzionali definiscono le funzionalità e i servizi specifici che l'applicazione deve fornire.

**CodiceDescrizione**

- RF1** Generazione di una coppia di chiavi pubbliche e private
- RF2** Generazione ed eliminazione di *wallet* contenenti le coppie di chiavi
- RF3** Implementazione della parte di accesso e registrazione tramite appoggio di *database*
- RF4** Implementazione di un meccanismo di storage sicuro che permetta agli utenti di salvare le chiavi private sul *Keystore/Keychain* del dispositivo
- [H] **RF5** Riuscire a criptare un messaggio inserito nell'applicazione tramite la chiave pubblica di un altro utente
- RF6** Riuscire a decifrare un messaggio tramite la propria chiave privata personale
- RF7** Implementazione di un meccanismo di recupero *wallet*
- RF8** Implementazione di un meccanismo di lettura di *tag NFC* per associare un documento a un *wallet*

Requisiti funzionali

Requisiti di qualità I requisiti di qualità stabiliscono degli *standard* secondo cui tali funzionalità del prodotto devono essere implementate.

**CodiceDescrizione**

- RQ1** Assicurare un'accuratezza nella generazione e crittografia delle chiavi superiore al 95%
- [H] **RQ2** Mantenere il tempo di risposta per la generazione del *wallet* al di sotto dei 2 secondi
- RQ3** Raggiungere una copertura di test automatici del 70% per le principali funzionalità

Ognuno di questi requisiti è stato verificato attraverso lo sviluppo di test automatici.

Requisiti di vincolo I requisiti di vincolo definiscono le limitazioni operative e tecnologiche cui il progetto ha dovuto adattarsi.

**CodiceDescrizione**

- RV1** L'applicazione finale deve essere sviluppata tramite il *framework* Flutter e il linguaggio Dart
- [H] **RV2** L'applicazione finale deve appoggiarsi a Firebase come appoggio per una base di dati
- RV3** Le chiavi private generate non devono essere trasmesse *online* ma devono essere custodite in spazi sicuri del dispositivo come il *Keychain* e *Keystore*
- RV4** Ogni documento deve essere associato a unico *wallet*

Requisiti di vincolo

Implementazione Il linguaggio Dart Lo studio di questo linguaggio di programmazione ha rappresentato la prima sfida.

Dart è un linguaggio di programmazione orientato agli oggetti sviluppato da Google, è stato concepito come alternativa

[H] Esempio di *type-safe* dart void main() int temp = 10; temp = 20; // OK //temp = "ciao"; // ILLEGALE: stringa

print(temp); // stampa 20

dynamic temp1 = 30; print(temp1); // stampa 30 temp1 = "ciao"; print(temp1); // stampa ciao

\**null-safe* Il linguaggio integra un sistema di *null safety*: il valore *null* può essere associato a una variabile solo se

[H] Esempio di *null safety* dart void main() String nome = "Mario"; print(nome); // stampa: Mario //nome = null

String? cognome = "Rossi"; cognome = null; print(cognome); //Nessun problema stampa null

\*Interfacce e classi astratte In Dart sia le interfacce esplicite che le classi astratte sono dichiarate usando la *keyword* *implements*.

Un'interfaccia definisce un contratto: qualsiasi classe che la deriva deve utilizzare la *keyword* *implements*, impegnandosi a

[H] Esempio di interfaccia e classe astratta dart abstract class ContrattoDiConnessione void connetti(); void disconnect()

class B void bMethod()

abstract class BaseScreen void logicaComune()

class GestoreServizio implements ContrattoDiConnessione, B @override void connetti() /\* implementazione \*/ @

class HomePage extends BaseScreen void init() super.logicaComune(); In Dart, ogni classe definita genera implementazioni

## Il *framework* Flutter

La seconda sfida è stata lo studio del *framework* Flutter. Flutter non utilizza componenti di interfaccia nativi del sistema.

Per lavorare bene in Flutter è fondamentale comprendere che ogni elemento dell'interfaccia è rappresentato da un *widget*.  
[H] *Stateless Widget* dart class Titolo extends StatelessWidget final String testo;  
const Titolo(super.key, required this.testo);  
@override Widget build(BuildContext context) return Text(testo);  
\**Stateful widget* Uno *stateful widget* è un *widget* mutabile, in grado di mantenere e aggiornare un suo stato interno.  
[H] *StatefulWidget* dart class Contatore extends StatefulWidget const Contatore(super.key);  
@override State<Contatore> createState() => ContatoreState();  
class ContatoreState extends State<Contatore> int valore = 0;  
@override Widget build(BuildContext context) return Column( children: [ Text('Valore: \${valore}'), ElevatedButton(

Architettura a tre livelli L'applicazione sviluppata adotta il modello di architettura a tre livelli, una struttura che si basa su tre diversi strati:

Il primo livello di presentazione rappresenta un punto di interazione tra l'utente e il sistema. È composto da tutto ciò che l'utente vede e interagisce con.

Il secondo livello rappresenta la logica di *business*. Coordina le operazioni tra il livello di presentazione e il livello dei dati.

Il terzo livello è responsabile della persistenza e recupero dei dati. Definisce come queste vengono archiviate e rese disponibili. Permette la separazione delle responsabilità (SoC<sup>4</sup>) eliminando la necessità di propagare le informazioni tramite la gerarchia di controllo. Aumenta le *performance* consentendoci di costruire i *widget* strettamente necessari quando lo stato cambia, evitando i refresh inutili. Fornisce un meccanismo per accedere e aggiornare lo stato rendendo il codice più facile da mantenere, facilitando quindi la manutenzione.

la tecnologia NFC La tecnologia NFC permette a due dispositivi di connettersi, scambiarsi informazioni o attivare funzioni. Lettura dati nei *chip* RFID Uno dei requisiti era l'implementazione di un meccanismo di riconoscimento dei documenti.

Per realizzare questo sono state analizzate due librerie Flutter che permettono la scansione e la scrittura attivando il sensore NFC.

Grazie alla documentazione fornita dagli sviluppatori eseguire un prototipo che permetta la scansione dei documenti è stato semplice.

[H] [width=0.5]..../assets/PrototipoNFC.png Applicazione prototipale per dimostrare la fattibilità della scansione NFC.

La libreria ci permette di recuperare i campi di ID, lo standard ISO del *chip* RFID (ad esempio la carta d'identità).

Testando l'applicazione con una varietà di documenti e *tag* NFC, ho riscontrato differenze significative nella struttura dei dati.

Nel caso della CIE ho rilevato un comportamento peculiare: il campo identificativo (*id*) letto tramite NFC non rappresentava sempre il valore corretto.

Il *chip* della Carta d'Identità Digitale (CIE) è infatti progettato per memorizzare dati personali sensibili, proteggendole con una crittografia forte. L'identificatore temporaneo e non persistente rientra nelle misure previste per ridurre i rischi legati al tracciamento e all'identificazione.

Questo rappresenta per il progetto un bel problema in quanto senza un identificativo del documento recuperabile tramite NFC non è possibile.

Una prima soluzione valutata è stata quella di consultare la piattaforma dedicata agli sviluppatori che realizzano i software.

A seguito di riunioni interne con il mio responsabile, si è scelto di non procedere su questa strada, sia per ragioni di tempo che di costi.

I documenti quali la CIE, la tessera sanitaria o le carte di credito adottano lo standard ISO/IEC 14443-4, che prevedono specifiche per la lettura dei dati.

[H] Funzione che permette la scansione del tag nfc [font-size=14px]dart Future<NFCTag>? \_fetchNfcData() async try if (Future<void> \_scanNfcTag() async if (isScanning) return; setState(() => isScanning = true); tryFinalNfcService =



Bibliografia  
[heading=subbibliography,title=Siti web consultati,type=online]