Università degli Studi di Padova

Laurea: Informatica

Corso: Ingegneria del Software

Anno Accademico: 2025/26

Gruppo 17

Nome: BitByBit

Email: swe.bitbybit@gmail.com

Specifica Tecnica

Stato:	In lavorazione
Versione:	0.1.1
Responsabile:	Gabriele Scaggiante
Redattori:	Gabriele Scaggiante
Verificatori:	Riccardo Manisi
Destinatari:	Miriade srl
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo BitByBit

Registro delle modifiche

Versione	Data	Autore	Descrizione	Verificatore
0.1.1	2026-03-31	Gabriele Scaggiante	Aggiunta di EventBridge, SAM, CloudFormation e Docker nella sezione tecnologie.	Dennis Parolin
0.1.0	2026-03-30	Gabriele Scaggiante	Creazione del documento. Scrittura dell'introduzione e delle tecnologie utilizzate.	Riccardo Manisi

Indice

Elenco delle tabelle

Elenco delle figure

1. Introduzione

1.1. Scopo del documento

Il presente documento ha lo scopo di fornire una descrizione completa, formale e strutturata del prodotto software, delineandone in modo rigoroso le caratteristiche tecniche e architetturali.
La Specifica Tecnica rappresenta il riferimento principale per tutte le attività di progettazione, sviluppo, **Verifica**[G] e manutenzione del sistema, assicurando coerenza rispetto ai requisiti definiti nel documento di **Analisi dei requisiti**[G] e alle scelte progettuali adottate.
In particolare, il documento si propone di:

- Definire l'**Architettura**[G] complessiva del sistema, descrivendo le componenti software, le loro responsabilità e le modalità di interazione attraverso appositi diagrammi
- Specificare i moduli funzionali e le interfacce esposte, includendo i contratti di comunicazione e i formati dei dati scambiati
- Descrivere l'**Architettura**[G] di **Deployment**[G], evidenziando la distribuzione delle componenti nei diversi ambienti di esecuzione e le relative dipendenze infrastrutturali
- Documentare le tecnologie, i linguaggi di programmazione, i framework e gli strumenti adottati, motivando le scelte effettuate in relazione ai requisiti di progetto e al **Capitolato**[G]
- Illustrare i principali design pattern e le soluzioni architetturali utilizzate, con particolare attenzione agli aspetti di scalabilità, manutenibilità e sicurezza
- Fornire indicazioni sui metodi di testing e di **Validazione**[G]
- Supportare le attività di evoluzione e manutenzione del software, garantendo una base documentale chiara e coerente.

1.2. Riferimenti

Al fine di evitare ambiguità relative al linguaggio e ai termini utilizzati all'interno dei documenti formali, viene allegato il *Glossario*. I termini tecnici, gli acronimi e le parole con significato specifico all'interno del progetto sono contrassegnati da una lettera 'G' in apice (es. **Agile**[G]) e sono descritti in dettaglio nel documento apposito.

1.2.1 Riferimenti normativi

- **Capitolato[G] d'appalto C4: *L'app che Protegge e Trasforma***
  Parti consultate: documento completo
  Versione: 1.0
  Ultimo accesso: *2026-01-14*
  https://www.math.unipd.it/~tullio/IS-1/2025/Progetto/C4.pdf

- **Norme di Progetto**
  Parti consultate: documento completo
  Versione: v.1.0.0
  Ultimo accesso: *2026-02-09*
  https://swe-bitbybit.github.io/SWE-docs/docs/RTB/documenti_interni/norme_di_progetto.pdf

1.2.2 Riferimenti informativi

- **Slide sulla Progettazione Software del Prof. Vardanega**
  Parti consultate: documento completo
  Versione: *A.A.2025/2026*
  Ultimo accesso: *2026-03-30*
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/T06.pdf

- **Materiale relativo alla parte pratica del corso di Ingegneria del Software**
  Parti consultate: documento completo
  Versione: *A.A.2022/2023*
  Ultimo accesso: *2026-03-30*
  https://www.math.unipd.it/~rcardin/swea/2022/

- ***Repository*[G] *Github* del Prof. Cardin**
  Ultimo accesso: *2026-03-30*
  https://github.com/rcardin/swe-imdb

- **Guida Flutter sull'Architettura[G] di un'applicazione**
  Ultimo accesso: *2026-03-30*
  https://docs.flutter.dev/app-architecture/guide

- **Glossario**
  Parti consultate: documento completo
  Versione: *v.1.0.0*
  Ultimo accesso: *2026-02-27*
  https://swe-bitbybit.github.io/SWE-docs/docs/RTB/documenti_interni/Glossario.pdf

2. Tecnologie

Le tecnologie utilizzate sono state scelte trovando un punto d'incontro tra la necessità di garantire la sicurezza e la tutela dei dati personali degli utenti e il bisogno di realizzare un'**Architettura**[G] solida, modulare e facilmente estendibile. Di seguito sono riportate tutte le tecnologie adottate

2.1. Framework

2.1.1 Flutter

Flutter è un framework open-source sviluppato da Google per la creazione di applicazioni multi-piattaforma a partire da un unico codice sorgente. Consente di sviluppare interfacce per dispositivi mobili, web e desktop utilizzando il linguaggio Dart. Flutter si basa su un sistema di widget componibili, che possono rappresentare qualsiasi elemento dell'interfaccia **Utente**[G]. Flutter include un proprio motore grafico (come Impeller o storicamente Skia) che disegna direttamente i componenti a schermo, garantendo elevata coerenza visiva e prestazioni indipendenti dalla piattaforma sottostante. È il framework alla base dell'App Che Protegge e Trasforma, il che consente di poter generare build per iOS e Android da un unico codice sorgente, a fronte di un maggiore utilizzo di risorse rispetto alle soluzioni native e di una minore possibilità di controllo diretto e personalizzazione delle funzionalità native della piattaforma.

- **Versione utilizzata**: 3.41.6
- **Documentazione**: https://docs.flutter.dev/

2.2. Linguaggi di programmazione

2.2.1 Dart

Dart è un linguaggio di programmazione sviluppato da Google, progettato per la realizzazione di applicazioni client moderne. È un linguaggio tipizzato (con supporto sia statico che dinamico) e orientato agli oggetti. Viene utilizzato principalmente in combinazione con Flutter, dove il codice Dart viene compilato ahead-of-time in codice nativo per migliorare le prestazioni, oppure eseguito tramite compilazione just-in-time durante lo sviluppo. L'utilizzo di Flutter rende di fatto quasi obbligatoria l'adozione di Dart come linguaggio.

- **Versione utilizzata**: 3.11.4
- **Documentazione**: https://dart.dev/docs

2.2.2 Python

Python è un linguaggio di programmazione ad alto livello, interpretato e orientato agli oggetti, noto per la sua sintassi semplice e leggibile. Il codice Python viene eseguito da un interprete che traduce le istruzioni in bytecode, poi eseguito dalla macchina virtuale Python. Nel progetto è utilizzato lato backend per la scrittura delle funzioni AWS Lambda e per i test tramite Pytest.

- **Versione utilizzata**: 3.14.3
- **Documentazione**: https://docs.python.org/3/

2.3. Servizi AWS

2.3.1 Lambda

AWS Lambda è un servizio serverless che consente di eseguire codice senza gestire direttamente server o infrastruttura. Le funzioni vengono attivate in risposta a eventi e scalano automaticamente in base al carico. Nel progetto è utilizzato per implementare la logica backend, eseguendo funzioni scritte in Python invocate tramite API Gateway.

- **Versione utilizzata**: Runtime Python 3.14
- **Documentazione**: https://docs.aws.amazon.com/lambda/

2.3.2 API Gateway

AWS API Gateway è un servizio che permette di creare, pubblicare e gestire API REST e HTTP, fungendo da punto di ingresso per le richieste client. Gestisce autenticazione, routing e integrazione con altri servizi AWS. Nel progetto è utilizzato per esporre endpoint HTTP che invocano le funzioni Lambda.

- **Versione utilizzata**: HTTP API (v2)
- **Documentazione**: https://docs.aws.amazon.com/apigateway/

2.3.3 DynamoDB

Amazon DynamoDB è un database NoSQL completamente gestito, progettato per offrire alte prestazioni e scalabilità automatica. I dati sono memorizzati in tabelle con chiavi primarie e accessibili con bassa latenza. Nel progetto è utilizzato per memorizzare dati strutturati come le note testuali del diario criptato e le chat con il chatbot.

- **Documentazione**: https://docs.aws.amazon.com/dynamodb/

2.3.4 S3

Amazon S3 (Simple Storage Service) è un servizio di storage a oggetti che consente di archiviare e recuperare dati in modo scalabile e sicuro. I file sono organizzati in bucket e accessibili tramite API. Nel progetto è utilizzato per memorizzare contenuti multimediali come immagini e file audio.

- **Documentazione**: https://docs.aws.amazon.com/s3/

2.3.5 Cognito

Amazon Cognito è un servizio di gestione delle identità che consente autenticazione, autorizzazione e gestione degli utenti. Supporta integrazione con provider esterni come Google. Nel progetto è utilizzato per gestire l'autenticazione degli utenti tramite account Google e per il controllo degli accessi.

- **Documentazione**: https://docs.aws.amazon.com/cognito/

2.3.6 Bedrock

Amazon Bedrock è un servizio che consente di utilizzare modelli di intelligenza artificiale generativa tramite API, senza gestire direttamente l'infrastruttura sottostante. Permette l'accesso a diversi modelli foundation per task come generazione di testo. Nel progetto è utilizzato per implementare la funzionalità di chatbot.

- **Documentazione**: https://docs.aws.amazon.com/bedrock/

2.3.7 SES

Amazon SES (Simple Email Service) è un servizio per l'invio e la ricezione di email in modo scalabile e affidabile. Supporta integrazione via API e SMTP. Nel progetto è utilizzato per inviare email ai contatti fidati degli utenti.

- **Documentazione**: https://docs.aws.amazon.com/ses/

2.3.8 CloudWatch

Amazon CloudWatch è un servizio di monitoraggio e osservabilità che raccoglie metriche, log ed eventi dai servizi AWS. Consente analisi, visualizzazione e configurazione di allarmi. Nel progetto è utilizzato per la raccolta e l'analisi dei log generati dalle funzioni Lambda.

- **Documentazione**: https://docs.aws.amazon.com/cloudwatch/

2.3.9 EventBridge

Amazon EventBridge è un servizio di event bus serverless che consente di gestire e instradare eventi tra diversi servizi AWS. Esso permette di definire regole per reagire automaticamente a eventi generati da servizi AWS o da fonti esterne. Nel progetto è utilizzato per orchestrare i flussi di controllo di utilizzo dell'app da parte degli utenti, e l'invio delle email secondo le tempistiche stabilite dal Dead man's switch.

- **Documentazione**: https://docs.aws.amazon.com/eventbridge/

2.3.10 SAM

AWS Serverless Application Model (SAM) è un framework open-source che semplifica la definizione e il **Deployment**[G] di applicazioni serverless su AWS. Estende AWS CloudFormation introducendo una sintassi più compatta per risorse come Lambda, API Gateway e DynamoDB e include strumenti per build, testing locale e deploy delle applicazioni. Nel progetto è utilizzato per definire e distribuire l'infrastruttura serverless in modo semplificato.

- **Documentazione**: https://docs.aws.amazon.com/serverless-application-model/

2.3.11 CloudFormation

AWS CloudFormation è un servizio di Infrastructure as Code che consente di modellare e gestire risorse AWS tramite template dichiarativi in formato YAML o JSON. Nel progetto è utilizzato come base per il provisioning dell'infrastruttura, anche tramite template generati da SAM.

- **Documentazione**: https://docs.aws.amazon.com/cloudformation/
- **Versioni**: CloudFormation non prevede versioni di servizio esplicite; eventuali aggiornamenti sono gestiti direttamente da AWS. I template possono però utilizzare versioni di specifiche risorse e trasformazioni (es. AWS::Serverless).

2.4. Librerie

2.4.1 http.dart

http è una libreria per il linguaggio Dart che consente di effettuare richieste HTTP in modo semplice e asincrono. Fornisce metodi per eseguire operazioni come GET, POST, PUT e DELETE, gestendo automaticamente la serializzazione delle richieste e la ricezione delle risposte. È progettata per essere leggera e facilmente integrabile in applicazioni client.

- **Versione utilizzata**: 1.6.0
- **Documentazione**: https://pub.dev/packages/http

2.4.2 image_picker.dart

`image_picker` è una libreria Flutter che permette di accedere alla fotocamera e alla galleria del dispositivo per selezionare o acquisire immagini e video. Fornisce un'interfaccia unificata tra piattaforme diverse, gestendo le differenze tra iOS e Android.

- **Versione utilizzata**: 1.2.1
- **Documentazione**: https://pub.dev/packages/image_picker

2.4.3 path_provider.dart

`path_provider` è una libreria Flutter che consente di ottenere i percorsi delle directory del file system del dispositivo, come directory temporanee o documenti applicativi. Facilita l'accesso a percorsi corretti e compatibili tra piattaforme per la gestione dei file.

- **Versione utilizzata**: 2.1.5
- **Documentazione**: https://pub.dev/packages/path_provider

2.5. Tecnologie di analisi statica

2.5.1 SonarQube

SonarQube è una piattaforma open-source per l'analisi continua della qualità del codice. Esegue **Analisi Statica**[G] su diversi linguaggi di programmazione per individuare bug, vulnerabilità, code smells e problemi di copertura dei test. Il funzionamento si basa su scanner che analizzano il codice sorgente e inviano i risultati a un server centrale, dove vengono aggregati e visualizzati tramite una dashboard dedicata.

- **Versione utilizzata**: 26.3.0
- **Documentazione**: https://docs.sonarsource.com/sonarqube/

2.6. Tecnologie di analisi dinamica

2.6.1 Pytest

Pytest è un framework di testing per il linguaggio Python che consente di scrivere test in modo semplice e scalabile. Supporta test funzionali, unitari e di integrazione, offrendo funzionalità come fixture, parametrizzazione dei test e scoperta automatica dei test.

- **Versione utilizzata**: 9.0.2
- **Documentazione**: https://docs.pytest.org/

2.6.2 Moto

Moto è una libreria Python che consente di mockare i servizi AWS durante i test. Simula il comportamento delle API AWS, permettendo di testare codice che interagisce con servizi cloud senza effettuare chiamate reali. Funziona intercettando le richieste e restituendo risposte simulate coerenti con quelle dei servizi AWS.

- **Versione utilizzata**: 5.1.23
- **Documentazione**: https://docs.getmoto.org/

2.7. Altro

2.7.1 Ollama

Ollama è una piattaforma che consente di eseguire e gestire LLM in locale tramite una semplice interfaccia. Permette il download, l'esecuzione e l'interazione con modelli di intelligenza artificiale direttamente in ambiente locale, senza necessità di servizi cloud esterni. Il funzionamento si basa su un runtime che gestisce i modelli e espone endpoint per inferenza.

- **Versione utilizzata**: 0.18.3
- **Documentazione**: https://ollama.com/docs

2.7.2 Docker

Docker è una piattaforma di containerizzazione che consente di creare, distribuire ed eseguire applicazioni all'interno di ambienti isolati (container). I container includono tutto il necessario per eseguire un'applicazione (codice, runtime, librerie e dipendenze), garantendo portabilità e consistenza tra ambienti diversi. Nel progetto è utilizzato a supporto delle pipeline di **CI/CD**[G] e per testare in locale servizi come DynamoDB e S3.

- **Documentazione**: https://docs.docker.com/

3. Architettura logica

4. Architettura di deployment