Università degli Studi di Bergamo

Laurea Magistrale in Ingegneria Informatica

Modulo di Progettazione Algoritmi e Computabilità (9 CFU)

PishTrap

Luca Ghisleni

Novembre 2024

Contents

List of Figures

List of Tables

1 Iterazione 0

1.1 Descrizione dello Scopo

Lo scopo principale dell'applicativo **PishTrap** è fornire un sistema semplice ed efficace per il rilevamento di attacchi di phishing attraverso l'analisi automatica delle email.

L'applicativo permette di:

- Identificare URL e contenuti sospetti utilizzando API esterne e dati locali.
- Raggruppare email di phishing in categorie per migliorare la comprensione delle minacce.

Questo approccio non solo aumenta la sicurezza dell'utente, ma contribuisce anche alla sua formazione nella prevenzione di attacchi futuri.

1.2 Requisiti e Casi d'Uso

I requisiti funzionali dell'applicativo saranno documentati utilizzando una **matrice di tracciabilità dei requisiti**. Questa matrice consente di tracciare i requisiti rispetto ai casi d'uso correlati e alla loro priorità, fornendo una visione chiara delle funzionalità implementate.

ID Requisito	Descrizione	Priorità	Casi d'Uso Collegati
CD-1	Login dell'utente nell'applicazione	Alta	CD-2
CD-2	Logout	Alta	CD-1
CD-3	Autenticazione con account Google	Alta	CD-1
CD-4	Scansione email non lette	Alta	CD-1
CD-5	Scansione email per giorno	Media	CD-1

Table 1: Matrice di tracciabilità dei requisiti funzionali

Ogni requisito è identificato da un codice univoco (**ID Requisito**), classificato per priorità (**Alta**, **Media**, **Bassa**) e collegato ai casi d'uso che lo soddisfano.

La matrice sarà aggiornata iterativamente durante lo sviluppo per garantire che tutti i requisiti funzionali siano tracciati e implementati correttamente.

Oltre ai requisiti funzionali, l'applicativo **PishTrap** deve rispettare una serie di requisiti non funzionali per garantire prestazioni, sicurezza e affidabilità. Di seguito sono riportati i principali requisiti non funzionali:

- **Prestazioni**: L'applicativo deve garantire una risposta rapida durante l'interazione dell'utente, specialmente per operazioni critiche come il login e la scansione delle email.
- **Scalabilità**: Il sistema deve essere in grado di gestire un aumento del numero di utenti senza degradare significativamente le prestazioni.
- **Sicurezza**: Tutti i dati sensibili devono essere protetti tramite comunicazioni cifrate e conformità alle normative sulla privacy.
- **Affidabilità**: Il sistema deve essere stabile e garantire la disponibilità continua dei servizi principali, riducendo al minimo i tempi di inattività.

- **Usabilità**: L'applicativo deve essere intuitivo e facile da usare, con un'interfaccia chiara che permetta agli utenti di completare le operazioni senza difficoltà.
- **Manutenibilità**: Il codice e l'architettura del sistema devono essere progettati in modo da agevolare future modifiche e aggiornamenti.
- **Compatibilità**: L'applicazione deve essere compatibile con dispositivi Android che supportano versioni moderne del sistema operativo e garantire un funzionamento ottimale.
- **Conformità**: Il sistema deve rispettare le normative sulla protezione dei dati personali, come il GDPR, e fornire una chiara informativa sulla privacy agli utenti.

Questi requisiti saranno monitorati e verificati durante tutte le fasi di sviluppo, attraverso test specifici e simulazioni, per assicurare il rispetto degli standard di qualità definiti.

1.3 Diagramma dei Casi d'Uso

Il diagramma dei casi d'uso rappresentato nella Figura 1 illustra le principali funzionalità offerte dall'applicativo **PishTrap** e le interazioni tra l'utente e il sistema.

1.4 Diagramma di Deployment

Il **Diagramma di Deployment** riportato nella Figura 2 illustra l'architettura di distribuzione dell'applicativo **PishTrap**, mostrando come i diversi componenti del sistema sono distribuiti tra i dispositivi hardware e i nodi software.

Il diagramma evidenzia:

- Il dispositivo utente (Smartphone Android) con l'applicazione Flutter.
- Il WebServer AWS, che ospita il back-end e i suoi servizi modulari.
- Le connessioni con servizi esterni come Google Cloud (OAuth 2.0, Gmail API, Safe Browsing API) e PhishTank.
- La comunicazione con il database MongoDB Atlas per l'archiviazione dei dati.

Questo diagramma riflette diversi stili architetturali che caratterizzano il sistema:

- **MVC (Model-View-Controller)**: È chiaramente visibile nella separazione tra il front-end (View), il back-end (Controller) e il database (Model).
- **Three-Tier Architecture**: Mostra la stratificazione in tre livelli principali: Presentazione (Flutter), Logica (back-end su AWS) e Persistenza dei dati (MongoDB).
- **Client-Server**: Il modello base di comunicazione tra il client (app Flutter) e il server back-end.
- **Microservizi**: Alcuni componenti del back-end, come il modulo per l'autenticazione OAuth 2.0 o l'integrazione con API esterne, possono essere visti come microservizi che operano indipendentemente.

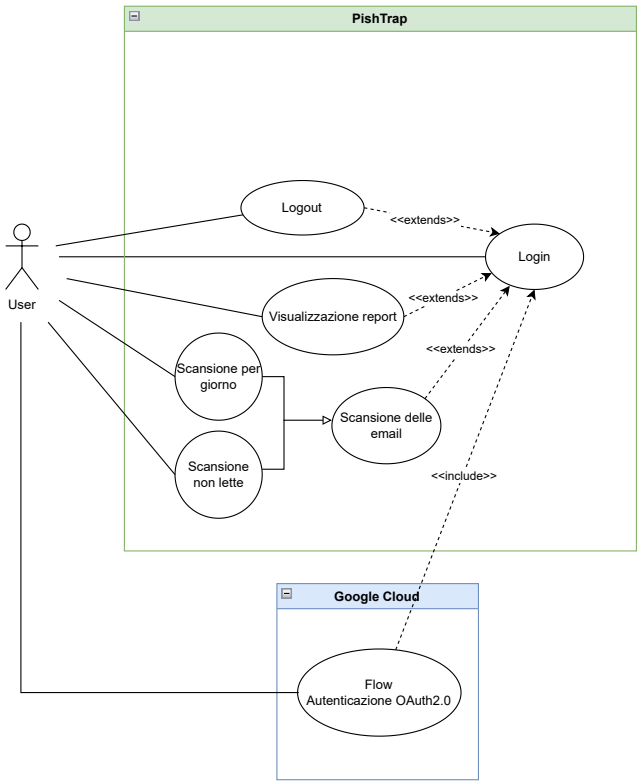

Figure 1: Diagramma dei Casi d'Uso

1.5 Toolchain Utilizzata

Per lo sviluppo e il deployment dell'applicativo **PishTrap**, è stata utilizzata una toolchain composta da tecnologie e strumenti moderni che garantiscono efficienza, scalabilità e flessibilità. Di seguito è riportata una lista completa degli strumenti utilizzati, con una breve descrizione delle loro funzionalità principali:

- **Flutter**: Framework open-source sviluppato da Google per creare applicazioni mobili, web e desktop con un'unica codebase. Utilizzato per lo sviluppo dell'interfaccia utente dell'applicativo su dispositivi Android.

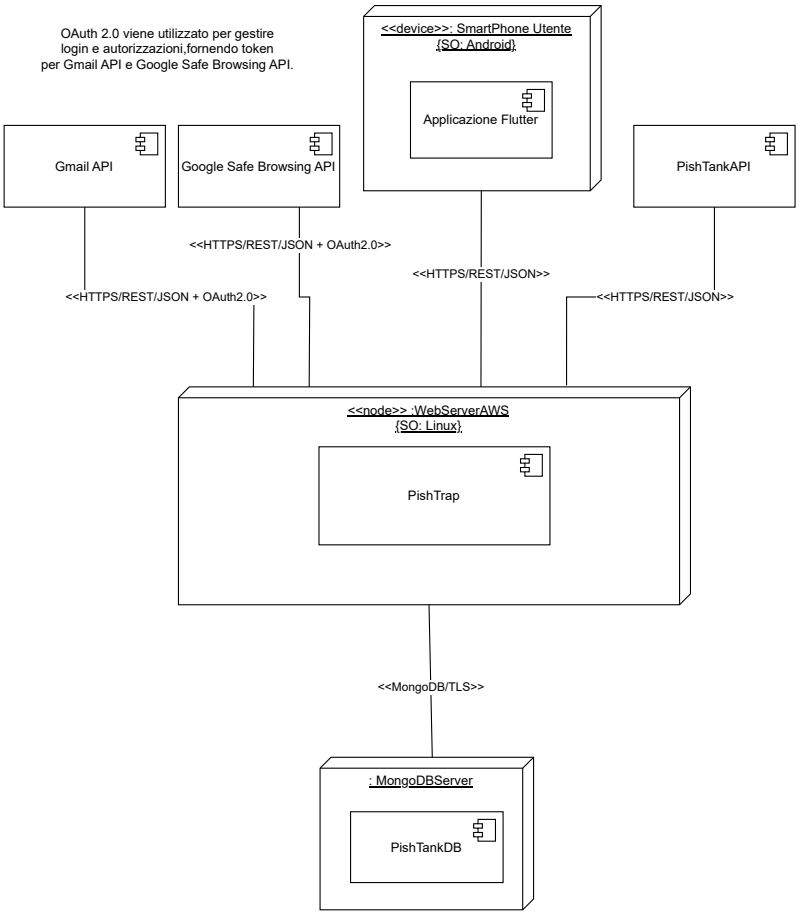

Figure 2: Diagramma di Deployment

- **GitHub**: Piattaforma per il versionamento del codice e la collaborazione tra sviluppatori. Utilizzato per il controllo delle versioni del progetto, la gestione dei branch delle iterazioni e la documentazione del codice.
- **PhishTank API**: Servizio online che fornisce un database costantemente aggiornato di URL noti per il phishing. Utilizzato per verificare l'affidabilità degli URL presenti nelle email.
- **MongoDB**: Database NoSQL cloud-based utilizzato per l'archiviazione dei dati analizzati, come email sospette e risultati delle scansioni.

- **AWS (Amazon Web Services)**: Piattaforma cloud utilizzata per il deployment del back-end, con particolare utilizzo di servizi come EC2 per l'esecuzione dei servizi server-side.
- **Python**: Linguaggio di programmazione utilizzato per implementare la logica del back-end, come l'integrazione con API esterne, la gestione dei dati e l'elaborazione degli algoritmi di clustering (K-means).
- **Google Cloud (Gmail API e Google Safe Browsing API)**: Servizi di Google Cloud utilizzati per l'integrazione con account Gmail (accesso e lettura delle email) e per il controllo della sicurezza degli URL tramite Google Safe Browsing.
- **Postman**: Strumento per testare e documentare le API REST. Utilizzato per verificare la correttezza delle API implementate nel back-end e per simulare le richieste provenienti dal client.
- **Draw.io**: Strumento per la creazione di diagrammi UML e schemi architetturali. Utilizzato per rappresentare graficamente i diagrammi di componenti, deployment e casi d'uso.
- **Overleaf**: Piattaforma online per l'editing collaborativo di documenti in LaTeX. Utilizzata per creare e gestire la documentazione del progetto, inclusi report e diagrammi integrati.

Questa toolchain è stata scelta per garantire uno sviluppo rapido ed efficace, assicurando al contempo la scalabilità, e una documentazione chiara e ben strutturata.