Documentazione Web File Transfer

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

* Titolo: Web File Transfer con crittografia end-to-end
* Scuola/sezione: Scuola Arti e Mestieri Informatica
* Allievo: Armir Cetaj
* Docente: Mariano Fasano
* Periodo: 05.09.2025 – 19.12.2025

## Abstract

L’aumento del numero di servizi cloud per la condivisione di file ha portato con sé importanti rischi legati alla privacy e alla sicurezza. Molte di queste piattaforme, infatti, possono accedere ai contenuti caricati dagli utenti, compromettendo la riservatezza delle informazioni.

Questo progetto propone lo sviluppo di una piattaforma web per il trasferimento sicuro di file con crittografia end-to-end. Il file viene cifrato nel browser del mittente prima dell’upload, e decifrato solo nel browser del destinatario al momento del download. In questo modo il server funge solo da contenitore di file cifrati senza possibilità di accedere ai dati in chiaro.

Sono incluse funzionalità aggiuntive come link effimeri, limite di download e notifica opzionale al mittente in caso di avvenuto scaricamento. Il progetto mira a dimostrare come sia possibile realizzare una soluzione semplice e user-friendly, ma al tempo stesso sicura, per la protezione dei dati condivisi online.

## Scopo

Lo scopo del progetto è sviluppare un’applicazione web che garantisca:

* la condivisione sicura di file senza esposizione dei dati in chiaro,
* la generazione di link temporanei per il download,
* la possibilità di impostare scadenze o limiti di accesso,
* un’opzione di notifica per il mittente al momento del primo download,
* un’interfaccia semplice e intuitiva utilizzabile da utenti non esperti.

# Analisi

## Analisi del dominio

Il contesto è quello della condivisione di file tra utenti via internet. Attualmente il problema viene affrontato da servizi come WeTransfer e SwissTransfer, che offrono praticità ma non garantiscono totale riservatezza, poiché i file vengono salvati in chiaro sui loro server.

L’obiettivo del sistema è offrire un’alternativa privacy-first, in cui:

* Il server non può leggere né modificare i contenuti.
* Gli utenti (mittente e destinatario) sono gli unici a possedere le chiavi di decrittazione.
* I link generati hanno validità limitata per ridurre i rischi di accesso non autorizzato.

Gli utenti principali sono:

* Mittente, che carica i file e decide eventuali opzioni (scadenza, limite download, notifica).
* Destinatario, che riceve il link e decritta i file.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Upload e cifratura lato client |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Richiede l’utilizzo di librerie come libsodium |
| **Sotto requisiti** | |
| **001** | L’utente deve poter selezionare uno o più file dal browser |
| **002** | I metadati devono essere anch’essi cifrati (nome file, dimensione) |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Generazione link univoco |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il token deve essere impossibile da indovinare (es. 256-bit random) |
| **Sotto requisiti** | |
| **001** | Creazione URL di download contenente il token |
| **002** | Il token deve avere validità temporale |
| **003** | Il token deve essere a un massimo di download configurabile |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Download e decifratura del file |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | La chiave di cifratura deve essere ricavata dal link |
| **Sotto requisiti** | |
| **001** | Il file deve essere decifrato automaticamente nel browser |
| **002** | Il file deve essere scaricato in chiaro solo localmente |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Opzioni di scadenza e limiti |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il mittente decide durata del link o numero massimo di download |
| **Sotto requisiti** | |
| **001** | Possibilità di impostare un ora/data di scadenza |
| **002** | Possibilità di impostare |
| **003** | Eliminazione automatica dei file scaduti |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Notifica opzionale al mittente |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’email è facoltativa e serve solo per avvisi di download |
| **Sotto requisiti** | |
| **001** | Un mail server |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

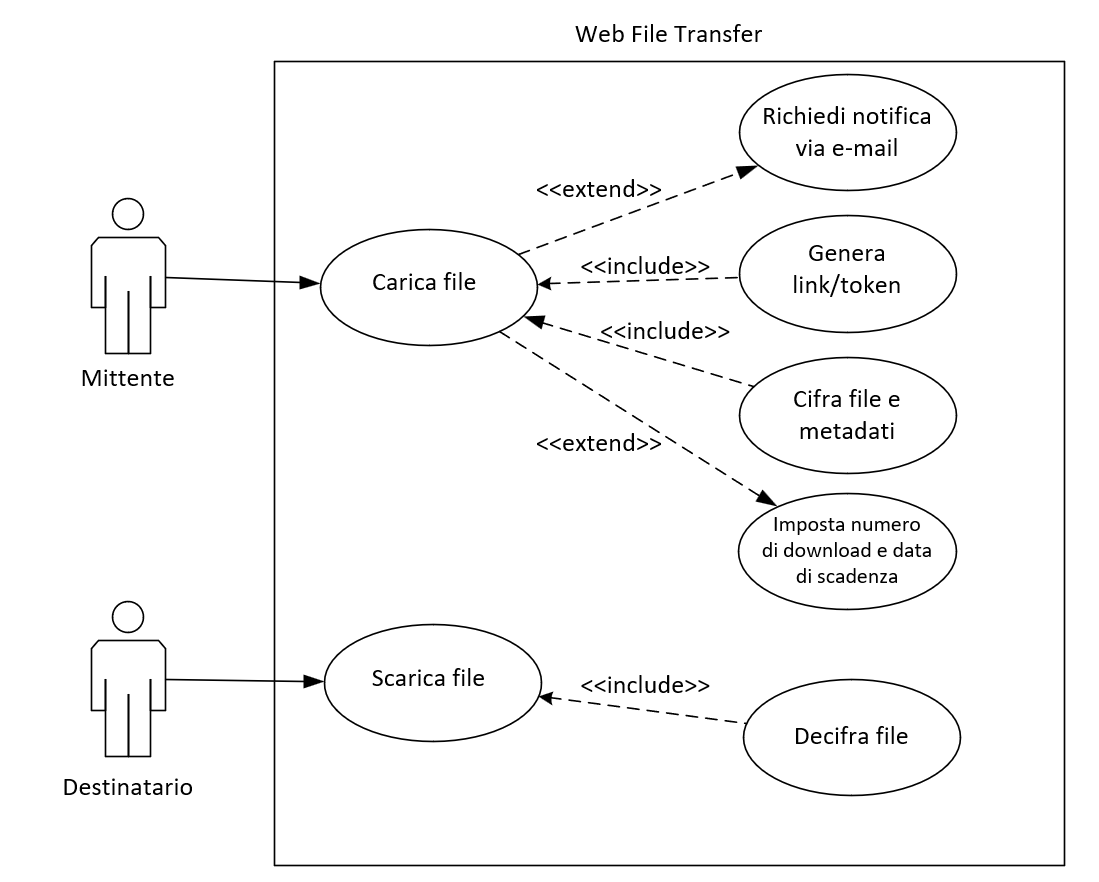


Figura 1: Diagramma Use case

## Pianificazione

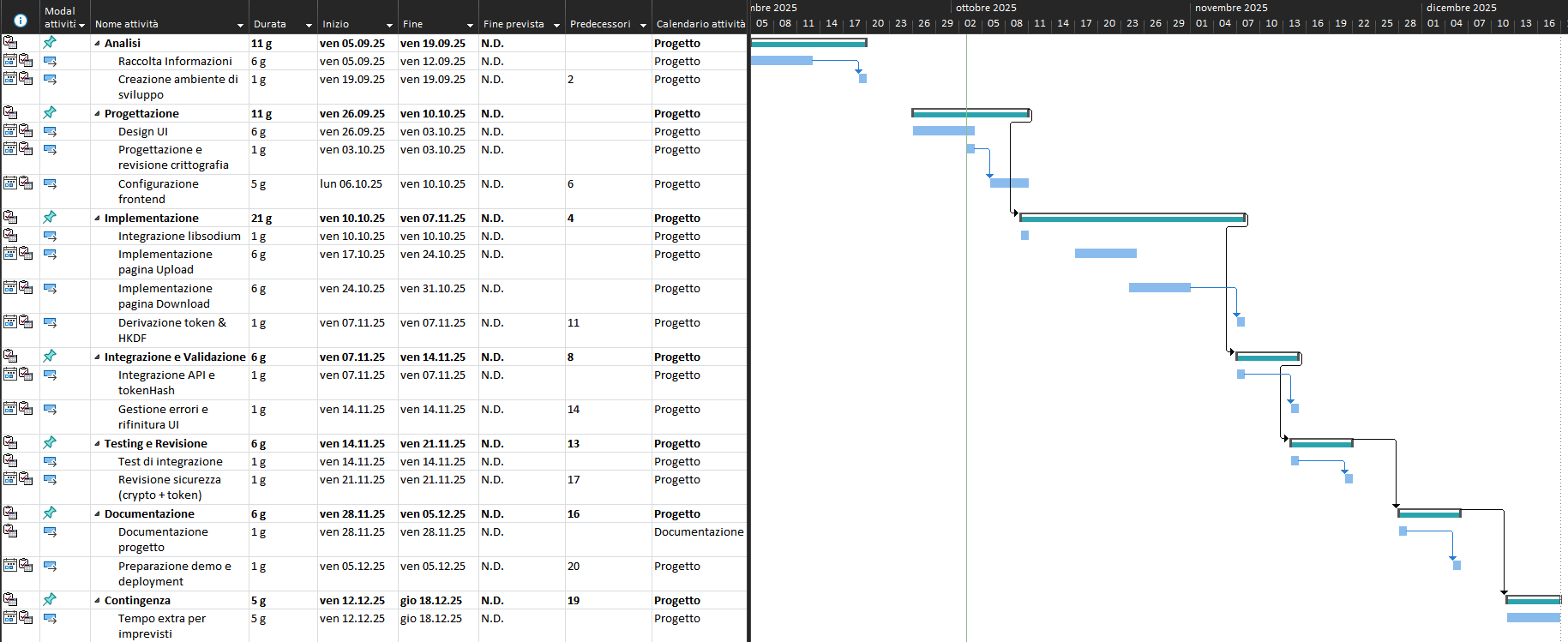


Figura 2: Diagramma di Gantt

## Analisi dei mezzi

### Software

**Linguaggi e runtime**

* JavaScript (ES6+): linguaggio principale, usato sia lato client che lato server.
* CSS 5.1: linguaggio utilizzato per definire lo stile del sito web.
* Node.js v22 LTS: runtime per eseguire codice JavaScript lato server.

**Framework e librerie backend**

* Express.js v5.1: framework leggero per la gestione delle API REST e del routing.
* Multer v2.2: middleware per la gestione degli upload di file (multipart/form-data).
* libsodium v1.0.20: libreria crittografica per la cifratura e decifratura sicura dei file lato server.
* pg (node-postgres) v8.16: driver ufficiale Node.js per interfacciarsi con PostgreSQL.

**Database**

* PostgreSQL v18: database relazionale utilizzato per memorizzare i metadati dei file condivisi: token temporanei, date di scadenza, indirizzi email opzionali del mittente (per notifiche).

**Mail server**

* Postfix v3.10 (configurato con emailwiz): gestisce direttamente l’invio delle email di notifica.
  + Configurato con SPF, DKIM, DMARC per garantire deliverability e ridurre lo spam.
  + Node.js non usa librerie intermedie: le notifiche vengono passate al mail server tramite SMTP locale.

**Strumenti di sviluppo**

* Visual Studio Code v1.93: IDE principale.
* Git v2.51 + GitHub: controllo versione del codice.
* curl / wget: test basilari delle API REST senza necessità di strumenti esterni.

**Browser (client)**

* Compatibilità con:
  + Google Chrome v141+
  + Mozilla Firefox v143+
  + Microsoft Edge v138+

### Hardware

* Per accedere al sito si ha bisogno di una piattaforma che garantisca una connessione stabile all’internet.
* Per replicare il server si ha bisogno di una macchina che consente la virtualizzazione (creare macchine virtuali Linux)

# Progettazione

## Design dell’architettura del sistema

### Architettura Generale

Il sistema è organizzato secondo un’architettura classica client-server:

Client:

* Cifra i file e i metadati localmente.
* Genera token e deriva la chiave per la cifratura.
* Effettua upload dei blob cifrati e avvia decrittazione al download.

Server (Node.js + Express):

* Espone API HTTPS per upload, download e stato.
* Memorizza solo blob cifrati e metadati cifrati; salva token\_hash e parametri (scadenza, limiti).
* Invoca il mail server locale per notifiche (se richiesto).

Database (PostgreSQL):

* Conserva record con token\_hash, path del blob, limiti, timestamp.

Mail server (Postfix/emailwiz):

* Gestisce la consegna delle email di notifica al mittente.

### Schema di rete

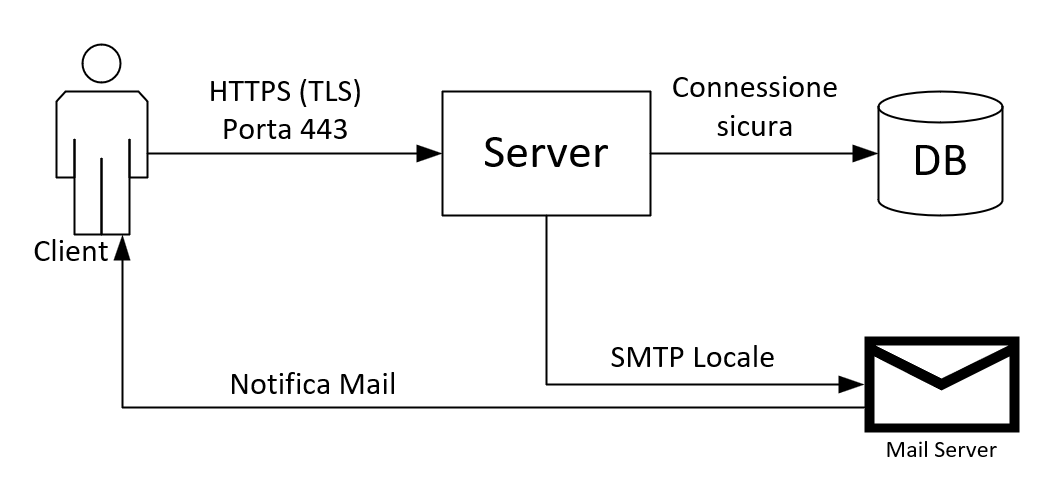


Figura 3: Schema di rete

Il client si connette al server stabilendo una connessione HTTPS sulla porta 443, il server comunica con il DB e il Mail Server, il Mail Server spedisce un’email al client.

## Design dei dati e database

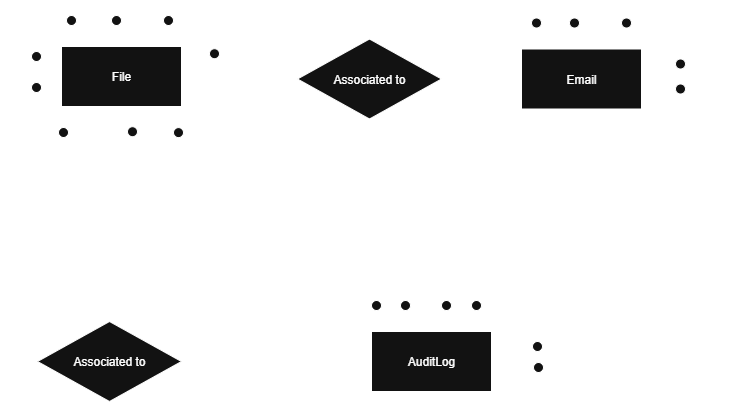


Figura 4: Schema ER

**files**

* id SERIAL PRIMARY KEY
* token\_hash BYTEA NOT NULL UNIQUE
* salt BYTEA NOT NULL
* metadata BYTEA NOT NULL
* path TEXT NOT NULL
* ciphertext\_length BIGINT NOT NULL
* max\_downloads INT NOT NULL DEFAULT 1
* download\_count INT NOT NULL DEFAULT 0
* expires\_at TIMESTAMP WITH TIME ZONE
* created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
* deleted\_at TIMESTAMP WITH TIME ZONE

**notifications**

* id SERIAL PRIMARY KEY
* file\_id INTEGER NOT NULL REFERENCES files(id) ON DELETE CASCADE
* email BYTEA
* notified\_at TIMESTAMP WITH TIME ZONE NULL
* created\_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()

**audit\_logs**

* id SERIAL PRIMARY KEY
* file\_id INTEGER NULL REFERENCES files(id) ON DELETE SET NULL
* type TEXT NOT NULL
* time TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
* actor\_ip INET
* details TEXT

## Design delle interfacce

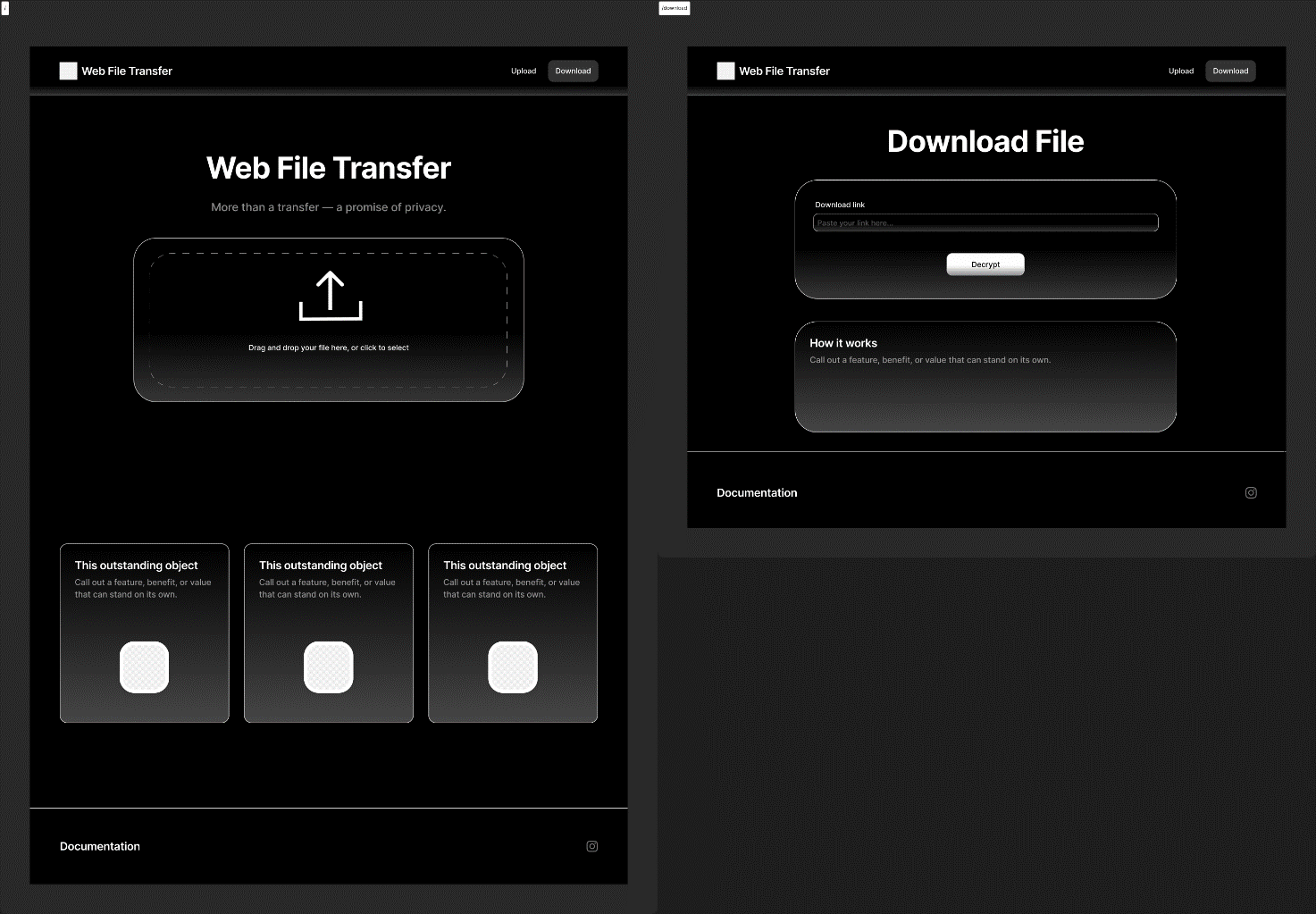


Figura 5: Mockup sito web

Pagina di Upload

* Titolo e sottotitolo: comunicano la funzione principale del servizio e l’enfasi sulla privacy.
* Area drag & drop: permette all’utente di trascinare o selezionare un file da caricare.
* Pulsanti di navigazione: accesso rapido alla sezione di download e alle altre funzionalità.
* Sezione informativa: tre riquadri sottostanti illustrano caratteristiche e vantaggi principali del servizio (es. sicurezza, velocità, anonimato).
* Footer: link a documentazione o informazioni aggiuntive.

Pagina di Download

* Campo link di download: l’utente incolla l’URL ricevuto.
* Pulsante “Decrypt”: avvia il processo di recupero e decifratura del file, tutto lato client per garantire la privacy.
* Sezione “How it works”: descrizione sintetica del funzionamento per aiutare l’utente a comprendere il processo.
* Footer: coerente con la pagina di upload.

## Design procedurale

### Diagramma Swimlane

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …