

# Report Progettino di autenticazione

## 1 User multi-factor authentication

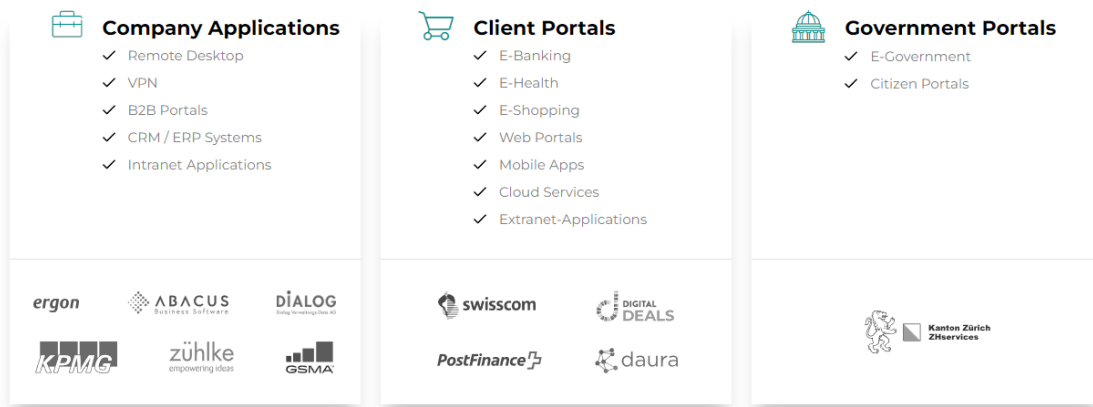
### 1.1 Architettura Mobile ID

Mobile ID permette all'utente di autenticarsi ai servizi online tramite il suo cellulare. Per utilizzarlo è necessario possedere una carta SIM compatibile con Mobile ID, i maggiori operatori offrono SIM compatibili.

Mobile ID può essere impiegato per il Login, per la firma digitale e per confermare una decisione. È una delle autenticazioni a due fattori più sicure al mondo.

#### Dove si può utilizzare Mobile ID?

Potete emettere firme digitali, effettuare il login o confermare la vostra volontà su moltissimi siti, app o con tool e servizi: il tutto sempre nella stessa modalità. La lista dei provider e dei portali cresce continuamente:



Fonte: <https://www.mobileid.ch/it/clienti-privati>

Naturalmente è possibile offrire questa autenticazione ai nostri clienti qualora fossimo noi un'azienda fornitrice di prodotti e servizi (B2B o B2C). (<https://www.mobileid.ch/it/clienti-commerciali>)

## Vantaggi

### Per i vostri clienti



#### Sicurezza

Grazie al PIN del Mobile ID o alla verifica biometrica e all'accesso al vostro smartphone, il vostro login è doppiamente protetto.



#### Comodità

Non più dispositivi aggiuntivi: Funzioni di Mobile ID via smartphone.



#### Un ID multipurpose

Sia nel contesto professionale che privato, con Mobile ID potete firmare, effettuare l'accesso e confermare la vostra volontà ovunque.



#### Semplicità

Con Mobile ID accedete in modo semplice e rapido, ovunque vi troviate.

### Per il tuo business



#### Facilità di utilizzo

Un login sicuro senza lunghi codici o verifiche SMS.



#### Correttezza

I costi sono trasparenti. Pagate per ogni utente attivo al mese.



#### Semplicità

I vostri clienti possono attivare autonomamente il Mobile ID e gestirlo tramite MyMobileID.



#### Flessibilità

Mobile ID non richiede altro hardware o software. Il nostro servizio può essere integrato perfettamente nel vostro sito web o applicazione tramite protocolli standard.

Fonte: <https://www.mobileid.ch/it/clienti-commerciali>

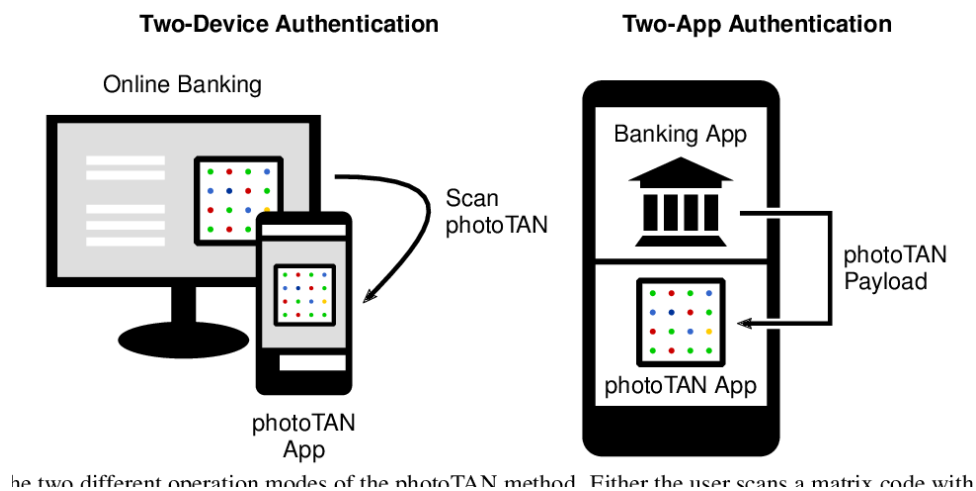
## 1.2 Architettura con autenticazione mTAN

In questo tipo di autenticazione l'accesso al servizio viene completato tramite l'invio di un codice di conferma via SMS. Questo codice è valido solo per un tempo limitato, solitamente nell'ordine dei minuti.

Inizialmente l'utente inserisce le proprie credenziali (e-mail e password per esempio) dopodiché viene richiesto l'inserimento del numero di cellulare cui inviare il codice per completare l'accesso.

## 1.3 Architettura Photo-TAN

In questo tipo di autenticazione l'accesso al servizio viene completato tramite l'invio di un mosaico contenente un codice allo schermo dell'utente. L'utente inquadra il mosaico con il proprio cellulare o un apposito lettore decodificando e visualizzando il codice di accesso occultato.



he two different operation modes of the photoTAN method. Either the user scans a matrix code with

Fonte: <https://www.semanticscholar.org/paper/On-App-based-Matrix-Code-Authentication-in-Online-Haupt-M%3BCller/5750f41cf39fd16f75a42b7a736ff2e95bfa5419>

## 1.4 Architettura biometrica per mobile banking

L'autenticazione biometrica sfrutta una "firma" fisica univoca dell'utente.

Questo può avvenire tramite

- Riconoscimento del viso
- Riconoscimento delle impronte digitali
- Riconoscimento degli occhi
- Riconoscimento vocale

Sebbene questi sistemi siano spesso molto sicuri, eventuali fattori esterni possono disturbare il riconoscimento: sole, disturbi sonori, condizioni delle dita, ecc... Inoltre quando il riconoscimento non è programmato correttamente esso può essere superato.

## 2 API sicura

OAuth	JWT	SAML
OAuth is a framework	WT is a token in the JSON format.	SAML is a Single Sign-On login standard.
A user accessing an API resource from any client software such as browser-based web apps, native mobile apps, or desktop apps, can use OAuth for identification and authorization purposes.	JWT is one of the tokens that the client can present while using the OAuth framework. It is a JSON object which contains encoded data structure with information about the token issuer, subject (claims), expiration time, and more.	SAML allows logged in users to access applications by transferring the identity details from other sessions.
OAuth centralizes the authorization server, which allows the clients or third-party users to seek permission before accessing the information on a particular server.	JWT is one of the many security mechanisms that can be implemented as part of the OAuth Framework.	SAML uses an XML document, which is digitally encrypted, to provide authentication and deliver the desired message to the client.
OAuth framework specifies different protocols such as HTTP/HTTPS that can be used for presenting the tokens.	JWT specifies the token format in the form of JSON object definition (JWT, JWS, and JWE).	SAML strongly follows HTTP communication protocols and various other protocols such as SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), and more.
OAuth defines security patterns in which the client can obtain an access token from the Authorization Server.	JWT, on the other hand, is independent of the security pattern that the client uses to obtain or present the token.	SAML allows only one computer in the network to perform the security check and share credentials seamlessly with all the connected computers.

## 3 Progetto JWT

### 3.1 Panoramica

Il progetto è composto da un sistema di registrazione all'interno del quale è possibile registrare un nuovo utente.

Email address

We'll never share your email with anyone else.

Password

Confirm password

Name

Last name

Select your birthday

Describe yourself

☐ Non sono un robot

Successivamente l'utente viene autenticato tramite un form di login. Se l'utente è presente nel database e

la password è corretta il backend genera e restituisce un token che viene memorizzato all'interno del browser.

Email address

Password

[Login](#) [New user? Register](#)

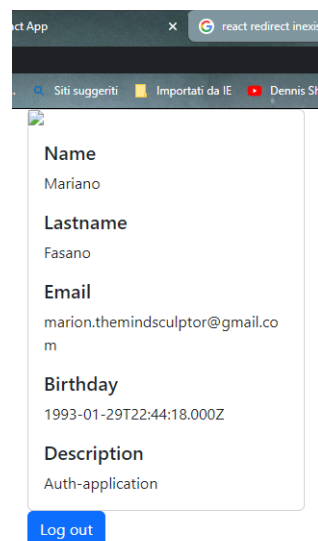
Per merito di questo token, jwt, l'utente viene autorizzato ad accedere ad alcune pagine e ad alcuni dati, nello specifico a dati memorizzati al momento della registrazione.

Dal momento che un utente è in possesso del token non potrà più ritornare alla pagina di login, a meno che non effettui un log out. Questo per evitare che ci si possa autenticare più volte. Esso viene indirizzato a una pagina di benvenuto.

**Welcome, you logged in successfully!**

[Go to your data](#)

Selezionando il pulsante sarà possibile leggere i dati dell'utente, come si può osservare nell'immagine sottostante.



The screenshot shows a web browser window with a single tab titled 'react redirect inexis'. The address bar shows 'Siti suggeriti', 'Importati da IE', and 'Dennis She'. The main content area displays a user profile with the following fields:

Field	Value
Name	Mariano
Lastname	Fasano
Email	marion.themindsculptor@gmail.com
Birthday	1993-01-29T22:44:18.000Z
Description	Auth-application

At the bottom of the profile card is a blue button labeled 'Log out'.

Nel momento in cui l'utente esegue il log out i dati memorizzati localmente sul browser, legati al token, vengono cancellati e si viene indirizzati alla pagina di login. Nel caso effettuasse il login un utente con ruolo di Admin sarebbe in grado anche di vedere la lista di utenti con ruolo user presenti nel database.

Il progetto di laboratorio è stato realizzato impiegando le seguenti tecnologie

- Visual Studio Code, come ambiente di sviluppo
- React, per il frontend
- Nodejs ExpressJs, per il backend
- MySQL, in qualità di database

Oltre al principale gruppo sopraccitato, sono state impiegate ulteriori librerie a supporto.

#### Backend

- **Bcrypt**, necessaria per criptare la password dell'utente
- Body-parser
- Cors
- Dotenv
- Express-validator, validazione dei campi lato server
- **Jsonwebtoken**, utile per creare il token, assegnarvi un payload e verificare i token estratti dall'header
- Mysql2
- Node-fetch, per effettuare la richiesta di verifica del recaptcha a Google
- Nodemailer
- Nodemon, riavvia automaticamente il backend in caso di modifica del codice
- **Sequelize**, ORM impiegato per l'interazione con il database, molto Js style

#### Frontend

- **Axios**, per effettuare le chiamate al backend
- Bootstrap
- React
- React-bootstrap
- React-datepicker
- React-dom
- **React-google-recaptcha**
- **React-hook-form**, per la validazione dei campi, tramite schema yup, lato client
- React-router, implementare il routing lato client (React)
- React-router-dom
- Yup, per gli schemi di validazione

### 3.2 Funzionamento del token

Ad autenticazione compiuta, il backend fornisce e invia un token che il frontend memorizza nel browser. Nelle successive richieste per le quali si necessita di un'autorizzazione si inserisce nel header, per esempio di axios, il token come Bearer token. Successivamente il backend estrarrà e verificherà il token. Da esso è possibile estrarre il payload nel quale è contenuto l'indirizzo e-mail dell'utente loggato.

Al log out si cancella il token lato browser.

#### Miglioramenti e sviluppi futuri

Escluso il token e gli accorgimenti per evitare autenticazioni doppie è stato aggiunto l'indirizzamento predefinito nel caso l'utente cerchi di raggiungere delle route lato client inesistenti.

Se per esempio si cercasse di raggiungere "/nome-di-route-inesistente" l'utente verrebbe indirizzato alla route root di login, la quale, se l'utente fosse in possesso del token, indirizzerebbe a sua volta alla pagina di benvenuto.

### 3.3 Controllo in fase di registrazione

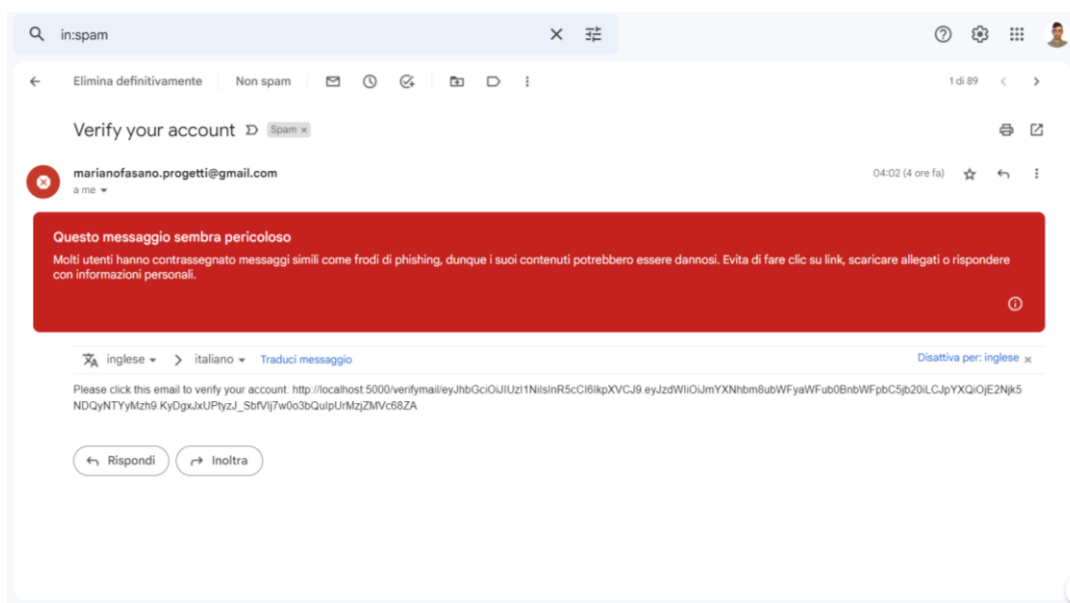
Si sono aggiunti due controlli aggiuntivi a livello di sicurezza, entrambi in fase di registrazione. Il primo è un recaptcha di Google con lo scopo di evitare registrazioni automatiche di utenti, mentre il secondo consiste

in una e-mail di verifica dell'account. In quest'ultima si invia un link contenente un token generato all'occasione e selezionando il link si invia una richiesta di verifica del token al server. Successivamente si viene indirizzati alla Homepage (il login). Se per qualche motivo la verifica di questo token fallisse l'utente dovrà ritentare la verifica. In ogni caso verrebbe indirizzato al login.

Entrambi i sistemi hanno incontrato dei problemi durante l'implementazione.

Il recaptcha sembra non raggiungere la verifica dell'API di google, dalla quale si dovrebbe ottenere, fra altre informazioni, un campo "success" di tipo booleano che ci indica se la verifica è andata a buon fine. Tuttavia secondo un `console.log()` del campo menzionato ciò che si ottiene è un "undefined". Rimaneva da indagare sul motivo di questo non funzionamento, poiché un primo test su un progetto esterno ha avuto esito positivo (tuttavia impiegava tecnologie differenti). Dopo una serie di prove inconcludenti siamo passati dalla libreria `axios` a quella di `node-fetch` per effettuare la richiesta a Google e il recaptcha ha infine funzionato.

L'e-mail di verifica invece ha incontrato qualche difficoltà poiché da maggio 2022 (<https://support.google.com/accounts/answer/6010255?hl=en>) Google non supporta più applicazioni di terze parti che permettono un livello di sicurezza inferiore e molti esempi di implementazione sono precedenti a quella data. L'invio è stato infine permesso grazie all'abilitazione della verifica in due passaggi e alla scelta di una password per delle app (ed è stata la password utilizzata per l'invio della mail); in caso della presenza di un antivirus potrebbe essere necessario disabilitarlo per inviare l'e-mail. L'e-mail potrebbe trovarsi, come nel caso di seguito illustrato, nella cartella di spam.



Cliccando sul link, ora differente rispetto all'immagine sopra riportata, si va a verificare il token inviato al suo interno. Dopodiché l'utente può effettuare il login correttamente e vedere i propri dati.

Un ulteriore controllo mancante è la scadenza (expiration time) del token, sia quello legato alla verifica sia quello legato all'autenticazione-autorizzazione.

Infine sarebbe estremamente comodo uno script o del codice che crei automaticamente il database e le tabelle in MySQL qualora non fossero già presenti, ma al momento questo non è implementato.

## 4 Framework OAUTH2

### 4.1 OAUTH2

OAuth 2.0 authorization framework è un protocollo standard open source per l'autorizzazione che si basa su SSL. Permette agli sviluppatori di implementare facilmente un sistema sicuro che fornisce flussi di autorizzazioni alle risorse protette, alle applicazioni web, applicazioni mobile e applicazioni desktop, come ad esempio la condivisione di risorse tra due siti web distinti. Queste autorizzazioni vengono concesse senza immettere le proprie credenziali e le comunicazioni avvengono tutte con il protocollo HTTP.

In pratica il sistema è molto semplice, OAuth introduce un livello di autorizzazione fornendo un determinato ruolo al proprietario di una risorsa e un altro ai clienti che richiedono tale risorsa. Quando un cliente richiede una risorsa, ospitata su un server e controllata da un proprietario, OAuth fornisce a tale richiesta un token di accesso.

Il token di accesso ricevuto da OAuth ha diversi attributi, tra cui ad esempio la sua scadenza. L'applicazione che ha creato il token- Il formato del token è JSON Web Token (JWT).

Il token inoltre ha anche dei permessi che vengono chiamati Scopes.

L'invio del token da parte del client verso il server può essere fatto in due modalità distinte:

- 1) Attraverso bear token, ovvero tramite GET o POST, esempio:  
GET / http/1.1  
Host: supsi.ch  
Authorization: Bear token
- 2) MAC (Message Authentication Code), calcolato tramite gli elementi della richiesta e inoltrato all'intestazione dell'autorizzazione.

I ruoli di OAuth sono essenzialmente 4:

- Proprietario della risorsa
- Server che ospita la risorsa
- Cliente (applicazione)
- Server di autorizzazione (OAuth)

### 4.2 Differenze tra OAuth e JWT

La principale differenza per iniziare è che JWT (Json Web Token) è solo un formato di token firmato digitalmente (il client non li può modificare), fornisce infatti un sistema compatto per trasmettere i dati, mentre OAuth definisce un protocollo di autenticazione e trasmissione del token.

I JWT vengono generati dal server e inviati al client, dopo di che il client lo archivia in modo che possa essere riutilizzato ad ogni richiesta che il client fa al server. Dunque non bisogna fare nessuna interrogazione al database.

I JWT non hanno la complessità di OAuth, infatti sono solo un token in formato Json, dunque una semplice stringa. Ciò rende JWT più veloce da verificare.

La sicurezza di OAuth è maggiore, infatti è testato praticamente in tutti i casi limite di sicurezza.

Si può utilizzare OAuth anche assieme a JWT per avere un livello maggiore di sicurezza!

## 5 Autenticazione JWT – pratica

Eventuali rischi legati all'impiego dei token jwt è che qualcuno possa decodificarli e scoprire al suo interno delle informazioni rilevanti. Nel nostro caso la decodifica porterebbe alla luce un indirizzo e-mail. Esso è sì un dato personale e privato dell'utente, però il token non contiene altro di rilevante e a partire dalle e-mail non è in grado di risalire alle altre, quindi un aggressore si ritroverebbe un indirizzo e-mail e niente di più.

## 6 Fonti

Capitolo 1:

Mtan <https://www.ebas.ch/it/mobile-tan-mtan-sms-tan/>

Photo tan <https://www.ebas.ch/it/procedura-photo-tan-procedura-con-tan-ottico/>

Mobile id

<https://www.mobileid.ch/it>

<https://www.postfinance.ch/it/privati/prodotti/digital-banking/mobile-id.html>

<https://www.swisscom.ch/it/business/enterprise/offerta/security/application-and-endpoint-security/mobile-id.html>

Biometrica

<https://www.onespan.com/it/topics/biometric-authentication>

Capitolo 2:

<https://www.kellton.com/kellton-tech-blog/api-security-design-patterns>