.7

# Specifica Tecnica

Versione 1.0.0


Arena Ivan Antonino
Baesso Nicola
Bousapnameme Ruth Genevieve
Calabrese Luca
Garon Martina
Liva Noemi
Marchiante Marco


## Progetto Ingegneria del Software


Dipartimento di Matematica
**Università degli Studi di Padova**




13 giugno 2023


**Contatti:** dotseventeam@gmail.com

## Registro delle versioni

| Versione | Data | Autore | Ruolo | Motivazione |
|---|---|---|---|---|
| 1.0.0 | 13/06/23 | Ruth Genevieve | Responsabile | Validazione |
| 0.0.6 | 19/05/23 | Ivan Antonino Arena | Progettista | Stesura §4.3, verificato da Marco Marchiante |
| 0.0.5 | 17/05/23 | Martina Garon | Progettista | Stesura §3 e relative sottosezioni (CAP-195), verificato da Ivan Antonino Arena |
| 0.0.4 | 12/05/23 | Nicola Baesso | Progettista | Stesura §4.2 e relative sottosezioni (CAP-166), verificato da Ivan Antonino Arena |
| 0.0.3 | 27/04/23 | Martina Garon | Amministratore | Stesura §2.2 (CAP-165), verificato da Noemi Liva |
| 0.0.2 | 20/04/23 | Martina Garon | Amministratore | Stesura §2 (CAP-153), verificato da Nicola Baesso |
| 0.0.1 | 10/04/23 | Ivan Antonino Arena | Amministratore | Stesura §1, verificato da Martina Garon |

Tabella 1: Registro di versionamento del documento

# Indice

# 1 Introduzione

## 1.1 Scopo del documento

Lo scopo del seguente documento è quello di descrivere nel dettaglio le scelte architetturali adottate durante la fase di progettazione del prodotto **CAPTCHA: Umano o Sovraumano?** a cura del gruppo **.7** e verificare che tali scelte soddisfino adeguatamente i requisiti previsti.

## 1.2 Scopo del prodotto

Lo scopo del prodotto è di fornire un sistema CAPTCHA per distinguere un utente umano da uno non-umano limitandone le possibili interazioni malevole di quest'ultimo con tale sistema ed integrarlo in un'applicazione web completa di funzionalità di registrazione e accesso.

## 1.3 Glossario

Onde evitare ambiguità lessicali si allega il *Glossario* redatto dal gruppo, contenente la terminologia tecnica utilizzata assieme alle relative defizioni.

## 1.4 Riferimenti

### 1.4.1 Riferimenti normativi

- Analisi dei Requisiti (versione 2.0.0);
- Capitolato d'appalto C1, Corso di Ingegneria del Software: `www.math.unipd.it/~tullio/IS-1/2022/Progetto/C1.pdf` (data ultimo accesso al link: 13 giugno 2023).

### 1.4.2 Riferimenti informativi

- OOP Principles Revisited, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2023/Object-Oriented%20Progamming%20Principles%20Revised.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Diagrammi delle Classi, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Diagrammi delle Classi, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Model-View Patterns, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/sweb/2022/L02.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Software Architecture Patterns, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Design Pattern Creazionali, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Creazionali.pdf` (data ultimo accesso al link: 13 giugno 2023);

- Design Pattern Strutturali, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Strutturali.pdf` (data ultimo accesso al link: 13 giugno 2023);
- Design Pattern Comportamentali, Corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2021/Design%20Pattern%20Comportamentali_4x4.pdf` (data ultimo accesso al link: 13 giugno 2023);
- T07 - Progettazione Software, Corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T07.pdf` (data ultimo accesso al link: 13 giugno 2023);

# 2 Struttura logica

Nel progetto, sono coinvolti più "componenti logici". Per chiarezza, si possono suddividere in 2 macro aree:

- Servizio CAPTCHA;
- Applicativo *web*.

Spieghiamo nel dettaglio a cosa di riferiscono.

## 2.1 Servizio CAPTCHA

Il "servizio CAPTCHA" è un insieme di funzionalità che si occupa delle comunicazioni che avvengono tra il *client* ed il *server*, riguardanti il CAPTCHA.
Il servizio è offerto come un API REST.
Il "servizio CAPTCHA" è intesto come "servizio di terze parti" ed ha le seguenti responsabilità:

- Creazione del CAPTCHA;
- Comunicazione del CAPTCHA, suddivisa in:
  - Ricezione di richiesta CAPTCHA, emessa dal *client*;
  - Invio del CAPTCHA al *client*;
  - Verifica delle risposte inviate dal *client*.

Si approfondiscono le singole responsabilità.

### 2.1.1 Creazione del CAPTCHA

I CAPTCHA sono preparati anticipatamente, rispetto alla richiesta di utilizzo.
Per questo motivo, un CAPTCHA attraversa due stati nel corso della sua vita:

- Stato 1;
- Stato 2.

Di seguito i dettagli.

### 2.1.2 Stato 1

I CAPTCHA nascono da una procedura, che li genera e li inserisce in un *pool*: al termine della stessa, i CAPTCHA ottengono lo "stato 1".
La procedura che genera un CAPTCHA in "stato 1" si suddivide in *step*:

- Creazione dell'immagine: date due immagini rappresentanti due soggetti diversi, viene creata una terza immagine, che rappresenta le due precedenti sovrapposte;
- Creazione delle risposte corrette: generazione delle parole che indicano i soggetti rappresentati nelle due immagini iniziali.

. Quindi, ciascun CAPTCHA in "stato 1" è composto da:

- Immagine, che rappresenta due soggetti sovrapposti in trasparenza;
- Due stringhe, che descrivono i due soggetti sovrapposti in trasparenza e che corrispondono alle risposte corrette del CAPTCHA.

Un CAPTCHA in "stato 1" viene inserito in un *pool*.

### 2.1.3 Stato 2

Un CAPTCHA in "stato 2" è pronto per essere inviato al *client*, allo scopo di essere risolto.
La procedura che fa avanzare in "stato 2" opera come descritto: dato un CAPTCHA dal *pool*, viene integrato di altri testi, che indicano dei soggetti non rappresentati dall'immagine.
I testi aggiunti in questa fase rappresentano le risposte errate al CAPTCHA.
Il CAPTCHA in "stato 2" è pronto per essere inviato al *client*.

### 2.1.4 Ricezione di richiesta CAPTCHA

Il "servizio CAPTCHA" è in ascolto di eventuali richieste da parte del *client*.
Quando riceverà una richiesta, si attiverà per inviare la risposta.

### 2.1.5 Invio del CAPTCHA al *client*

Il "servizio CAPTCHA" invia un CAPTCHA al *client*.
Il CAPTCHA verrà estratto da un *pool* di CAPTCHA preparati precedentemente.
Al *client* verrà inviato un CAPTCHA, composto da:

- Immagine, che rappresenta due soggetti sovrapposti in trasparenza;
- Due stringhe, che descrivono i due soggetti sovrapposti in trasparenza;
- Altre stringhe, che descrivono oggetti non presenti nell'immagine;
- Codice *secret*, per identificare in modo univoco la transazione;
- *Token* crittografato, contenente il *timestamp* dell'invio.

Per poter superare con successo il CAPTCHA, l'utente dovrà selezionare le due stringhe corrette (provenienti dal *pool*), ignorando quelle errate.

### 2.1.6 Verifica delle risposte utente al CAPTCHA

L'utente ha ricevuto il CAPTCHA e deve tentare di risolverlo, scegliendo le risposte corrette. Dal punto di vista tecnico, la risposta che l'utente invia al "servizio CAPTCHA" è composta da:

- *App ID*;
- Codice *secret*;
- Due risposte;
- *Token* che il *client* ha ricevuto precedentemente.

Il "servizio CAPTCHA" dovrà verificare la correttezza e la validità dei dati ricevuti.
Se tutti i controlli avranno esito positivo, il CAPTCHA si intende superato. Diversamente, se i controlli avranno esito negativo, il CAPTCHA si intende non superato.
Pertanto, il "servizio CAPTCHA" invierà l'esito della verifica al *client*.

## 2.2 Applicazione *web*

L'applicazione *web* è il sito *internet* che viene presentato all'utente.
L'applicazione *web* dimostra le funzionalità del "servizio CAPTCHA". Il sito *web* è composto dalle seguenti pagine:

- *Homepage*;

- Accedi: conterrà il *form* per il *login*. All'interno di questa pagina, sarà visibile il CAPTCHA, in una sezione dedicata;
- Pagina riservata, visibile dopo aver fatto il *login* con successo;
- Pagina di *logout*.

Le funzioni previste, ma non ancora implementate, sono segnalate con relativo avviso.

### 2.3 Accortezze per la sicurezza informatica

La sicurezza informatica è un tema fondamentale. Per questo motivo, alcune scelte architetturali hanno prestato particolare attenzione alla sicurezza ed a bloccare tentativi di elusione.
Ad esempio, il Servizio CAPTCHA non memorizza il *token* emesso: in questo modo, si evita la saturazione delle risorse dopo numerose richieste, potenzialmente malevoli.
Diversamente, il *token* ricevuto dal *client* viene memorizzato, solo per il suo periodo di validità. Questa accortezza esclude risposte multiple ad un CAPTCHA (quindi con un unico *token*) e limita le risorse occupate dalla memorizzazione. Quando il *token* è scaduto, viene cancellato.

# 3 Tecnologie coinvolte

## 3.1 Tecnologie per la codifica

### 3.1.1 Linguaggi

| Tecnologia | Versione | Descrizione |
|---|---|---|
| Java | 17 LTS | Linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica |
| HTML | HTML5 | Linguaggio di *markup* usato nel *frontend* |
| CSS | CSS3 | Linguaggio usato per definire la formattazione di documenti HTML |
| Javascript | 1.8.5 | Linguaggio di *scripting* |

Tabella 2: Linguaggi per la codifica

### 3.1.2 Strumenti

| Tecnologia | Versione | Descrizione |
|---|---|---|
| Docker | 23.0.4 | Strumento di containerizzazione |
| PostgreSQL | 14.6 | Database relazionale *open source* |
| Maven | 3.9.0 | Strumento di gestione di progetti software basati su Java e *build automation* |

Tabella 3: Strumenti per la codifica

### 3.1.3 Librerie e *framework*

| Tecnologia | Versione | Descrizione |
|---|---|---|
| THINGS 4 All | Non specificato | *Dataset* di immagini |
| Spring Boot | 2.7.11 | *Framework* per lo sviluppo web |
| Java 2D | Non specificato | API che la libreria standard di Java mette a disposizione per l'interazione con le immagini |
| ReactJS | 18.2.0 | Libreria JavaScript *front-end* gratuita e *open source* |

Tabella 4: Librerie e *framework* per la codifica

## 3.2 Strumenti per l'analisi del codice

### 3.2.1 Analisi statica

| Tecnologia | Versione | Descrizione |
|---|---|---|
| Compilatore Java | 8.0.660.18 | Traduce il codice sorgente (un altro programma) scritto in un linguaggio di programmazione di alto livello in codice oggetto o target (un terzo programma) scritto in un linguaggio di più basso livello |
| JaCoCo | 0.8.10 | Strumento per CI per Java (Spring Boot), per i test statici |

Tabella 5: Strumenti per l'analisi statica

### 3.2.2 Analisi dinamica

| Tecnologia | Versione | Descrizione |
|---|---|---|
| GitHub Actions | - | Piattaforma di integrazione continua e distribuzione continua (CI/CD) che consente di automatizzare la *pipeline* di compilazione, test e distribuzione. |
| React Testing Library | 14.0.0 | Libreria per testare i componenti sviluppati in ReactJS |
| Codecov | 5.0 | Servizio online che integra vari sistemi di CI/CD e visualizza i risultati dei test |
| JaCoCo | 0.8.10 | Strumento per CI per Java (Spring Boot), per i test dinamici |

Tabella 6: Strumenti per l'analisi dinamica

# 4 Architettura del prodotto

## 4.1 Architettura generale

### 4.1.1 Schema

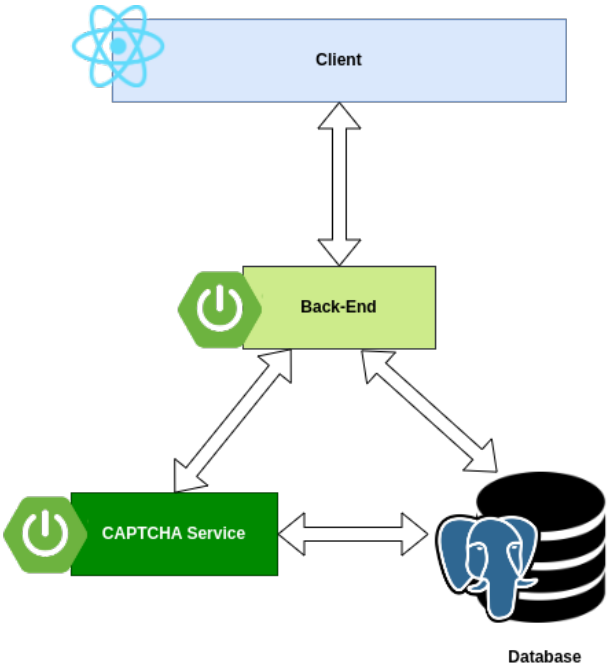


Figura 1: Schema architetturale

### 4.1.2 Descrizione

Per questioni di estensibilità, si è deciso di sfruttare un'architettura a microservizi. Il sistema è costituito da:

- due *Spring Boot App*, una per il *back-end* ed una per il servizio CAPTCHA;
- un'app React che fa da *client* (*front-end*);
- un *database* PostgreSQL che sfrutta il sistema delle API REST, gestendo degli *endpoint* soggetti a richieste HTTP.

## 4.2 Diagramma delle classi

### 4.2.1 Interfaccia Utente

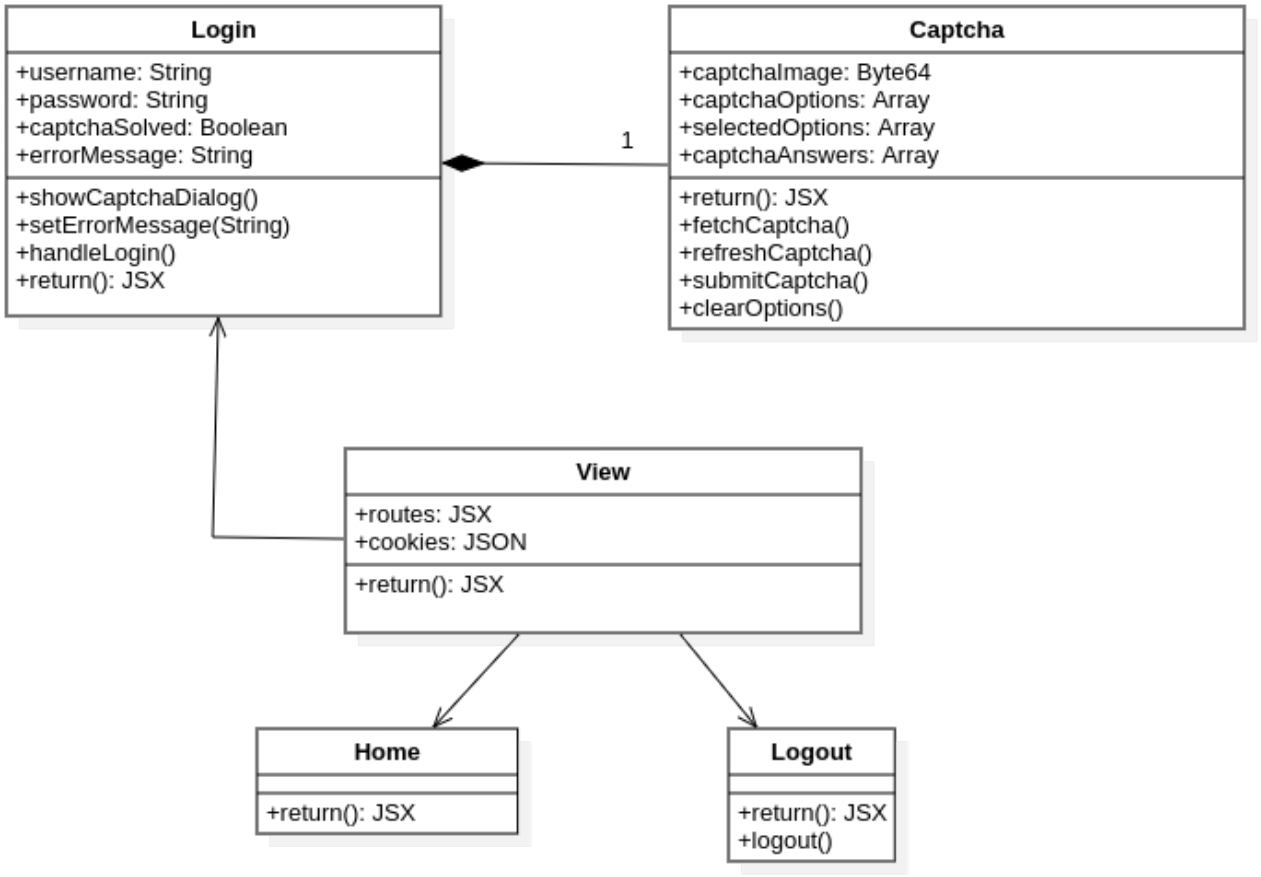


Figura 2: Diagramma delle classi dell'interfaccia utente

Il diagramma riportato rappresenta le componenti React utilizzate per la creazione dell'interfaccia utente, la quale è formata dalle seguenti componenti:

- **View**: componente principale che si occupa di creare le pagine dell'interfaccia e i vari collegamenti tra di esse;
- **Home**: pagina iniziale del sito, senza funzionalità disponibili:
- **Login**: pagina dove l'utente può effettuare l'accesso al sito dopo aver compilato tutti i campi correttamente ed aver risolto il CAPTCHA;
- **Logout**: pagina dove l'utente può effettuare il *logout* al sito se è autenticato;
- **Captcha**: componente che viene mostrato all'utente per completare l'accesso o la registrazione, sottoponendolo ad un *test* CAPTCHA;

### 4.2.2 Back-end

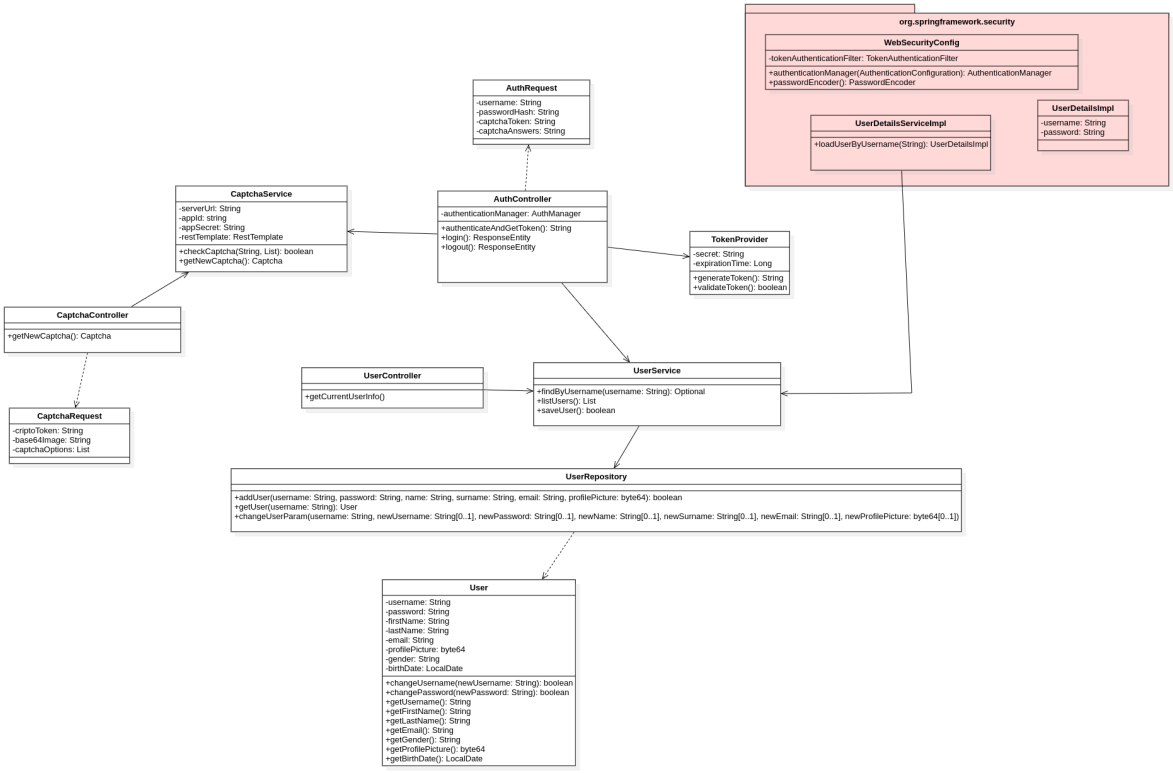


Figura 3: Diagramma delle classi dell'interfaccia utente

Per supportare l'interfaccia utente, l'applicazione *web* dispone di un *backend* creato con Spring Boot. Tale *backend*, oltre ad occuparsi della gestione degli utenti, si occupa di mettere in contatto l'interfaccia utente con il servizio CAPTCHA, al fine di poter permettere l'utilizzo di *test* CAPTCHA. In particolare, è formato dalle seguenti classi:

- **AuthController**: classe che gestisce la comunicazione tra interfaccia utente e servizio CAPTCHA. Contiene il CAPTCHA recuperato dal servizio e i dati necessari per connettersi allo stesso. Inoltre, ha un riferimento ad UserService per l'autenticazione dell'utente;
- **UserService**: classe che permette l'autenticazione di un utente, contiene il repository di utenti da cui effettua la ricerca di un utente;
- **UserRepository**: classe che si interfaccia con il database. Permette la creazione, la ricerca e la modifica di un utente;
- **UserController**: classe che gestisce le richieste ai dati degli utenti;
- **CaptchaController**: classe che gestisce le richieste di test CAPTCHA;
- **CaptchaService**: classe che si interfaccia con il database per verificare i test CAPTCHA o richiederne di nuovi;
- **AuthRequest**: classe che rappresenta le richieste di login;
- **CaptchaRequest**: classe che rappresenta le richieste di test CAPTCHA;
- **TokenProvider**: classe che fornisce la logica dei Token;

- **User**: classe che rappresenta l'utente;
- **WebSecurityConf, UserDetailsServiceImpl, UserDetailsImpl**: classi del *package org.springframework.security* per gestire la sicurezza del sistema.

### 4.2.3 Servizio CAPTCHA

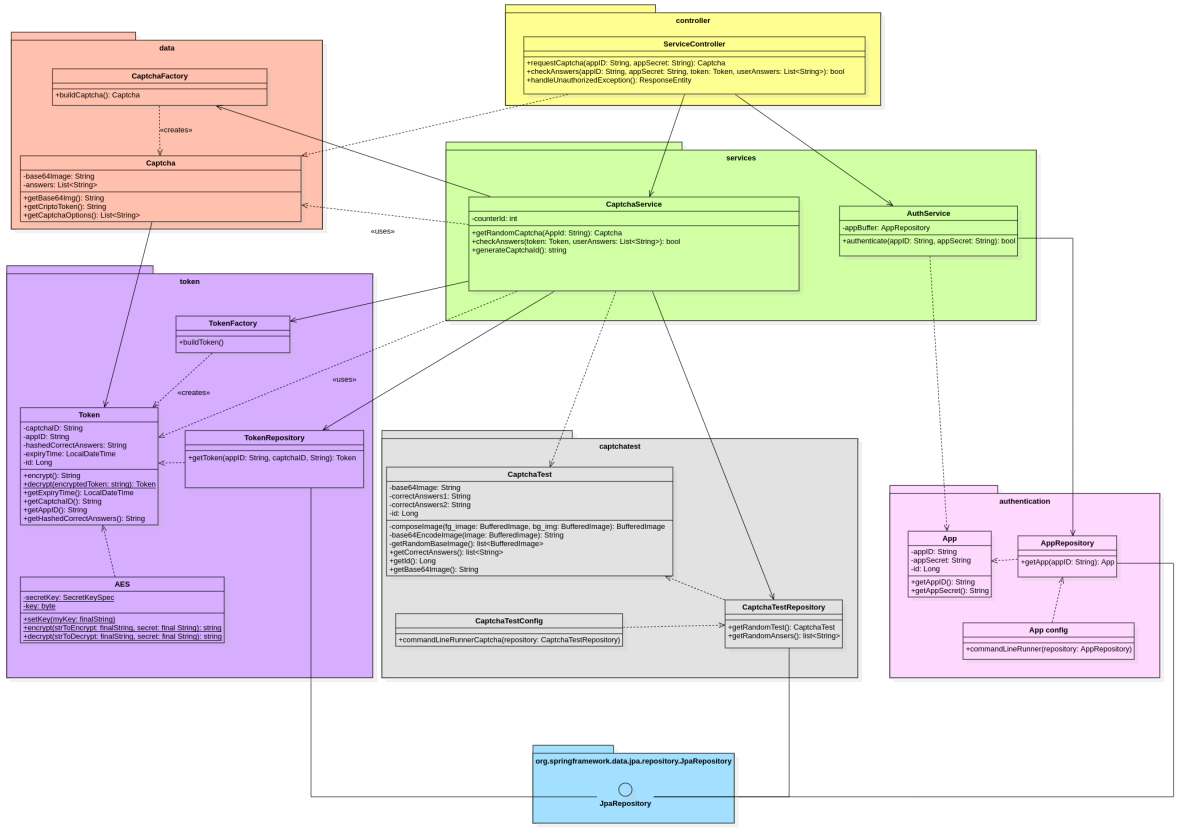


Figura 4: Diagramma delle classi del servizio CAPTCHA

Il servizio CAPTCHA prevede la seguente suddivisione:

- **ServiceController**: classe che gestisce la suddivisione delle richieste, inoltrandole ai servizi corretti;
- **AuthService**: classe che si occupa dell'autenticazione di un client web che richiede l'accesso ai test CAPTCHA;
- **AppRepository**: classe che contiene tutte le app autorizzate ad accedere al servizio;
- **CaptchaService**: classe che si occupa della creazione e della verifica dei test CAPTCHA;
- **TokenRepository**: classe che contiene tutti i *Token* generati
- **App**: classe che rappresenta una applicazione web autenticata al servizio;
- **CaptchaTest**: classe che rappresenta un test, formato dall'immagine e dalle risposte esatte;
- **CaptchaTestRepository**: classe che contiene tutti i *CaptchaTest* generati;
- **Captcha**: classe che rappresenta un CAPTCHA;
- **CaptchaFactory**: Factory Method per la creazione di oggetti CAPTCHA;
- **TokenFactory**: Factory Method per la creazione di oggetti Token;
- **Token**: classe che rappresenta un token, ovvero un elemento contenente gli identificativi di un CAPTCHA e dell'applicazione web a cui è stato inviato, assieme alle risposte corrette criptate e ad un tempo di scadenza.

## 4.3 Architettura di dettaglio

### 4.3.1 Interfaccia Utente

In linea con la filosofia di React, il *frontend* impiega il *pattern Composition*, presentando diversi componenti per ogni elemento grafico o funzionale, il *Conditional Rendering*, per renderizzare diversi componenti a seconda delle condizioni specificate ed i React *Hooks*, introdotti con la versione 16.8, come modello di ogni componente. Insieme agli *Hooks* viene utilizzato anche il *pattern Render Props*, nello specifico, a condividere il modello della pagina di *login* con il componente del CAPTCHA e viceversa.

### 4.3.2 *Back-end* e Servizio CAPTCHA

Sia il *back-end* che il servizio CAPTCHA sono *app* Spring Boot e condividono la stessa serie di *design pattern*, comunemente utilizzati durante lo sviluppo con Spring:

- *Repository* per gestire l'accesso ai dati facilmente con Spring Data JPA;
- *Service Layer* per incapsulare la logica di *business* delle REST API, separando la logica di accesso ai dati dalla gestione delle richieste HTTP;
- *Data Transfer Object (DTO)*, utilizzati dal *back-end* per rappresentare i dati che vengono inviati attraverso le richieste e le risposte API con maggiore flessibilità nel modellare i dati e separando la logica di dominio dalla rappresentazione dei dati;
- *Factory Method* per creare oggetti senza specificare la classe esatta dell'oggetto che deve essere creato;
- *Controller* utilizzato per gestire ed elaborare le richieste HTTP.

### 4.3.3 *Database*

Il *database* è composto dalle seguenti tabelle:

- Token;
- Captcha_test;
- App;
- Users.

La tabella "Token" contiene i dati relativi all'entità *token*, necessaria per verificare la validità del CAPTCHA inviato al *client*. Contiene i riferimenti al CAPTCHA inviato al *client* ed i riferimenti generati dall'applicativo in uso.
La tabella "Captcha_test" colleziona i CAPTCHA inviati al *client*, affinché possa rispondere. Oltre alla chiave primaria (campo "id"), contiene l'immagine in stringa (campo "base64_image") e le due risposte corrette ("correct_answer1" e "correct_answer2"). I dati permettono di poter verificare la risposta ricevuta dal *client*.
La tabella "App" riguarda l'entità *app* e contiene il codice *secret* generato per identificare l'applicativo che richiede un CAPTCHA (campo "app_secret").
L'oggetto User è rappresentato nel *database* dalla tabella "Users", avente lo "*username*" come chiave primaria.

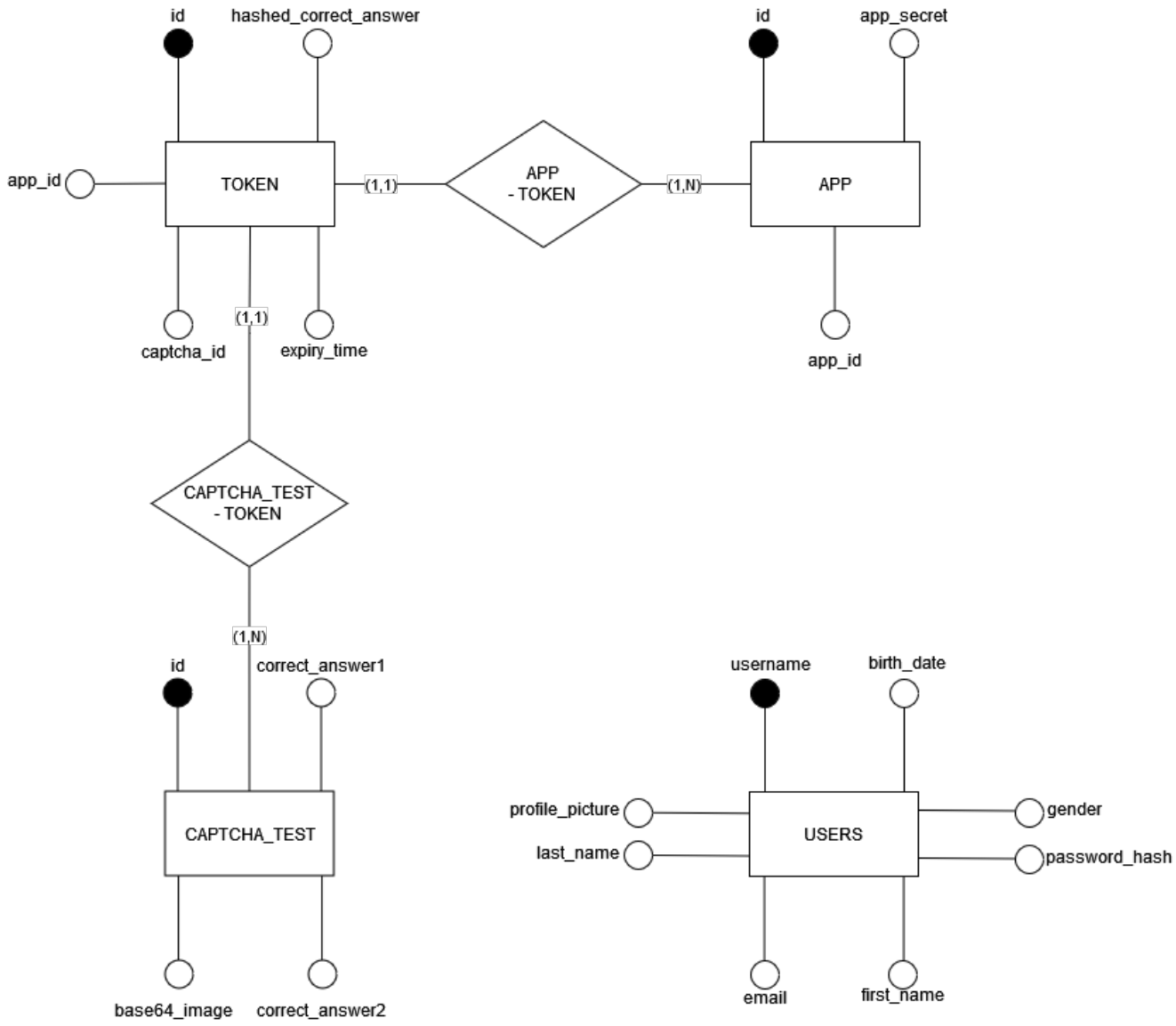


Figura 5: Diagramma ER ristrutturato del *database*