



OUT OF BOUNDS

MANUALE UTENTE
PROGETTO CAPTCHA
Versione 1.0.0

Responsabile	Michele Cazzaro
Redattori	Alberto Matterazzo Michele Cazzaro Jacopo Angeli Edoardo Retis Simone Bisortole
Verificatori	Michele Cazzaro Valentina Caputo Simone Bisortole Alberto Matterazzo
Uso	Esterno
Destinatari	Out of Bounds prof. Tullio Vardanega Prof. Riccardo Cardin Zucchetti S.p.A.

CONTATTI
sweoutofbounds@gmail.com
REPOSITORIES
[orgs/SWE-OutOfBounds/repositories](https://github.com/SWE-OutOfBounds/repositories)

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
1.0.0	2023/05/26	Michele Cazzaro	Responsabile	Approvazione e rilascio del documento.
0.1.0	2023/05/23	Alberto Matteredazzo	Verificatore	Verifica generale del documento.
0.0.8	2023/05/18	Simone Bisortole, Alberto Matteredazzo	Programmatore, Verificatore	Modifiche migliorative a §4 e verifica.
0.0.7	2023/05/12	Edoardo Retis, Simone Bisortole	Programmatore, Verificatore	Espansione §4, espansione §3, aggiunta ulteriori immagini e verifica.
0.0.6	2023/05/10	Jacopo Angeli, Simone Bisortole	Programmatore, Verificatore	Espansione §1, espansione §2, §3 e verifica.
0.0.5	2023/05/09	Edoardo Retis, Simone Bisortole	Programmatore, Verificatore	Aggiunta immagini e verifica.
0.0.4	2023/05/05	Jacopo Angeli, Simone Bisortole	Programmatore, Verificatore	Inizio stesura §4 e verifica
0.0.3	2023/04/26	Michele Cazzaro, Valentina Caputo	Programmatore, Verificatore	Inizio stesura §2 e verifica.
0.0.2	2023/04/03	Alberto Matteredazzo, Michele Cazzaro	Amministratore, Verificatore	Stesura §1 e verifica.
0.0.1	2023/03/31	Alberto Matteredazzo	Amministratore	Creazione Scheletro del documento.

Tabella 1: Registro delle modifiche

Indice

1	Introduzione	4
1.1	Scopo del documento	4
1.2	Scopo del prodotto	4
1.3	Glossario	4
1.4	Riferimenti	4
2	Utilizzo di clockCAPTCHA	5
2.1	Requisiti	5
2.1.1	Browser supportati	5
2.2	Installazione	5
2.3	Importazione degli oggetti	6
2.4	Creazione, inserimento e utilizzo di ClockCAPTCHAView	6
2.4.1	Struttura di ClockCAPTCHAView	7
2.5	Generazione di un CAPTCHA	7
2.5.1	Algoritmi di generazione	9
2.6	Verifica di un CAPTCHA	10
3	Utilizzo dell'applicazione	12
3.1	Requisiti	12
3.2	Installazione	12
3.3	Configurazione e avvio	13
3.3.1	MySQL	13
3.4	Dettagli Applicazione	14
3.4.1	Home Page	14
3.4.2	Login	14
3.4.3	Registrazione	16

Elenco delle figure

1	Risultato dell'utilizzo di HTMLCanvasGenerator.	9
2	Risultato dell'utilizzo di NoiseDecorator.	9
3	Risultato dell'utilizzo di ShapesDecorator.	9
4	Risultato dell'utilizzo dei decorator combinati tra loro.	10
5	Home page dell'applicazione.	14
6	Dettagli della Home page.	14
7	Dettagli pagina di accesso dell'applicazione.	15
8	Pagina di registrazione.	17

Elenco delle tabelle

1	Registro delle modifiche	1
---	------------------------------------	---

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di illustrare le funzionalità fornite dalla libreria *clock-captcha* e fornire le istruzioni per il suo utilizzo. Il documento inoltre riporta i requisiti minimi necessari per il corretto funzionamento dell'applicazione elencandone alcuni aspetti funzionali.

1.2 Scopo del prodotto

La richiesta dell'azienda Zucchetti è un servizio *web CAPTCHA_G*, ovvero un prodotto che discrimini esseri umani da *bot_G* artificiali. Il servizio deve essere erogato attraverso una libreria *open-source_G*.

Clock-captcha è un servizio di *CAPTCHA_G* basato su immagini. Si differenzia dall'offerta di servizi *CAPTCHA* già disponibili fornendo un tipo di sfida poco visto: il riconoscimento di un orario da un orologio analogico. La lettura di un orologio analogico è culturalmente universale e un processo rapido, pertanto si presta bene a questo scopo.

1.3 Glossario

I termini che possono generare dubbi riguardo al loro significato vengono contrassegnati con una lettera G al pedice, a indicare che il termine si può trovare nel documento *Glossario v2.0*.

1.4 Riferimenti

- [Regolamento del progetto didattico](#) [Ultima consultazione: 2023/05/23],
- [Documentazione Capitolato presentato dall'azienda](#) [Ultima consultazione: 2023/05/23].

2 Utilizzo di clockCAPTCHA

2.1 Requisiti

Per l'utilizzo della libreria è necessario avere installati:

- **npm:** *npm* (*Node Package Manager*) è un gestore di pacchetti per *Node.js*. Viene installato automaticamente insieme a *Node.js*. *npm* è utilizzato per gestire le dipendenze del progetto e installare i pacchetti necessari.
- **TypeScript Compiler (tsc):** *TypeScript* è un *superset* di *JavaScript* che offre funzionalità aggiuntive come la tipizzazione statica. Il compilatore *TypeScript* (*tsc*) è necessario per convertire il codice *TypeScript* in *JavaScript* eseguibile dai browser.

2.1.1 Browser supportati

L'applicazione e il servizio *clock-CAPTCHA* supportano i seguenti *browser*:

- **Google Chrome:** versione 110 e successive,
- **Microsoft Edge:** versione 110 e successive,
- **Mozilla Firefox:** versione 110 e successive.

2.2 Installazione

Per utilizzare la libreria *clock-captcha*, è necessario scaricare e compilare il codice sorgente disponibile nelle *release* della [repository pubblica](#) seguendo nell'ordine i seguenti punti:

1. Clonare il contenuto della *repository*_G,
2. Installare le dipendenze della libreria, attraverso il comando `npm install`,
3. Compilare il codice sorgente utilizzando il comando `tsc`,
4. Inserire la cartella `dist` all'interno del progetto.

I metodi di libreria offerti da *clock-captcha*_G sono utilizzabili da qualsiasi applicativo *JavaScript*.

La libreria si limita a tre funzioni principali:

- Fornire dati per la somministrazione di *CAPTCHA* ad applicazioni *web*
 - Fornire dei dati criptati e la loro rappresentazione grafica quanto più capibile per l'essere umano e quanto meno interpretabile da sistemi informatici,
 - Fornire un metodo per la validazione di una interpretazione di una rappresentazione grafica generata in precedenza,
- Offrire in fine una vista da utilizzare per l'esposizione della rappresentazione grafica generata e per il recupero di un'interpretazione utente.

2.3 Importazione degli oggetti

Front-end

Per la parte di *front-end* la libreria fornisce un oggetto da utilizzare per rappresentare i dati visivamente e per raccogliere l'inserimento di un utente. L'oggetto si chiama `ClockCAPTCHAView` e per importarlo nel codice si può utilizzare

```
import { ClockCAPTCHA } from 'clock-captcha/dist'
```

Back-end

Per utilizzare gli oggetti utili alla generazione dei dati e alla loro consecutiva verifica, forniti dalla libreria utilizzando si può utilizzare:

```
const cc = require("path/to/clock-captcha/dist/index");
```

Se il parametro `"type"` contenuto nel file `package.json` è impostato a `"module"` allora utilizzare la sintassi

```
import * as cc from 'path/to/clock-captcha/dist';
```

Ovviamente il nome `cc` può essere modificato a proprio gradimento.

2.4 Creazione, inserimento e utilizzo di `ClockCAPTCHAView`

L'oggetto `ClockCAPTCHAView` rappresenta un elemento *HTML* utile per visualizzare i dati generati da *ClockCAPTCHA*, raccogliere l'input utente e visualizzare eventuali errori. L'elemento *HTML* ha la seguente forma:



Per utilizzare `ClockCAPTCHAView` il procedimento è il seguente:

```
//Costruisce il modulo CAPTCHA e inizia l'animazione di caricamento  
captchaModule: ClockCAPTCHAView = new ClockCAPTCHAView();
```

```
//Inserisce il modulo nell'elemento HTML specificato  
captchaModule.inject(document.getElementById('clock-captcha'));
```

```
//Imposta l'immagine CAPTCHA e il token associato  
captchaModule.fill(img_src,token);
```

dove `img_src` e `token` sono rispettivamente la rappresentazione grafica generata dalla funzione specifica dell'oggetto `ClockCAPTCHA` una stringa criptata che contiene il valore effettivo del `CAPTCHA`

Alcuni degli altri metodi utili forniti dalla libreria sono:

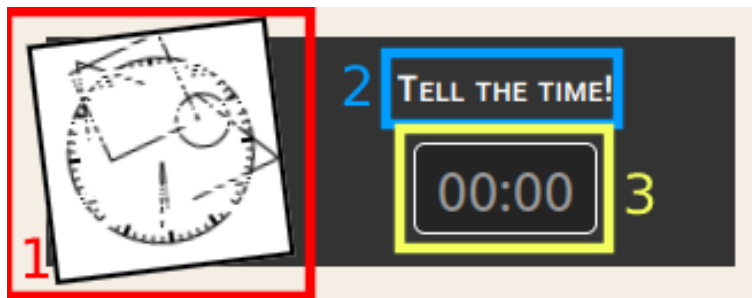
```
//Visualizza un messaggio di errore all'interno del modulo.
captchaModule.error("CAPTCHA errato, riprova.");

//Ripristina l'animazione di caricaento rimuovendo eventuali messaggi o errori
//precedentemente visualizzati.
captchaModule.clear()

//Visualizza un messaggio all'interno del modulo clock-captcha
captchaModule.message("Inserisci utilizzando questo formato: HH:MM");
```

2.4.1 Struttura di ClockCAPTCHAView

Di seguito illustrati gli elementi principali della vista `clockCAPTCHAView`



1. **Immagine CAPTCHA:** riquadro dedicato alla visualizzazione dell'immagine sfida da risolvere per superare il test,
2. **Campo di testo per istruzioni o errori:** di *default* invita l'utente a indovinare l'orario ma, può essere sostituito con un messaggio di errore o con un messaggio informativo,
3. **Campo input:** utilizzato per raccogliere la risposta alla sfida dell'utente.

2.5 Generazione di un CAPTCHA

La generazione dei dati da utilizzare in `ClockCAPTCHAView` avviene mediante l'utilizzo della funzione `generateData(image_generator: ClockImageGenerator)` dell'oggetto `ClockCAPTCHA`, in combinazione con un oggetto `ClockImageGenerator`.

La creazione di quest'ultimo necessita di un algoritmo di genesi dell'immagine, la cui definizione può variare a seconda del caso d'utilizzo. La libreria fornisce infatti un algoritmo di base che genera solo un orologio analogico e due oggetti che gli applicano una decorazione, così da poter utilizzare un grado di disturbo personalizzato:

```
// Orologio base
base_strategy: ClockImageGenerationStrategy =
    new HTMLCanvasGenerator();
```




```
//Aggiunta di forme geometriche all'immagine
second_strategy: ClockImageGenerationStrategy =
    new ShapeDecorator(base_strategy, <quantità_forme>);

// Aggiunta di disturbo all'immagine
third_strategy: ClockImageGenerationStrategy =
    new NoiseDecorator(base_strategy, <fattore_disturbo>);

// Concatenazione di disturbi
fourth_strategy: ClockImageGenerationStrategy =
    new ShapeDecorator(
        new NoiseDecorator(
            base_strategy,
            <fattore_disturbo>
        ),
        <quantità_forme>
    )

// Creazione del generatore d'immagine
base_generator: ClockImageGenerator =
    new ClockCImageGenerator(strategy);
high_security_generator: ClockImageGenerator =
    new ClockImageGenerator(fourth_strategy);

// Genesi dei dati per ClockCAPTCHAView
base_data: Object = ClockCAPTCHA.generateData(
    <SECRET_PWD>,
    base_generator
);
high_security_data: Object = ClockCAPTCHA.generateData(
    <SECRET_PWD>,
    high_security_generator
);
```

Gli oggetti `base_data` e `high_security_data` contengono la coppia:

```
{
    image: string
    token: string
}
```

che viene utilizzata per inizializzare un oggetto `ClockCAPTCHAView`.

Il valore di `token` viene poi inviato al *back-end* in combinazione con quanto inserito dall'utente e viene utilizzata la funzione `validateData()` per verificare la natura dell'utente.

2.5.1 Algoritmi di generazione

Come spiegato in precedenza la libreria fornisce nella versione v1.x.x i seguenti oggetti da utilizzare alla creazione dell'oggetto `ClockImageGenerator`

- **HTMLCanvasGenerator**: Algoritmo base per la generazione di un orologio. Utilizzato da solo genera la seguente immagine:



Figura 1: Risultato dell'utilizzo di `HTMLCanvasGenerator`.

- **NoiseDecorator**: aggiunge un disturbo rimuovendo casualmente alcune linee dall'immagine. La quantità di linee rimosse è personalizzabile attraverso il secondo parametro del costruttore. Utilizzato in combinazione con `HTMLCanvasGenerator` genera la seguente immagine:



Figura 2: Risultato dell'utilizzo di `NoiseDecorator`.

- **ShapeDecorator**: aggiungere delle figure geometriche di dimensione e posizione randomici. Il numero di figure è specificabile durante la costruzione dell'oggetto. Utilizzato in combinazione con `HTMLCanvasGenerator` genera la seguente immagine:



Figura 3: Risultato dell'utilizzo di `ShapesDecorator`.

Come dimostrato nella sezione precedente si possono combinare i decoratori per ottenere un risultato personalizzato ottenendo ad esempio:

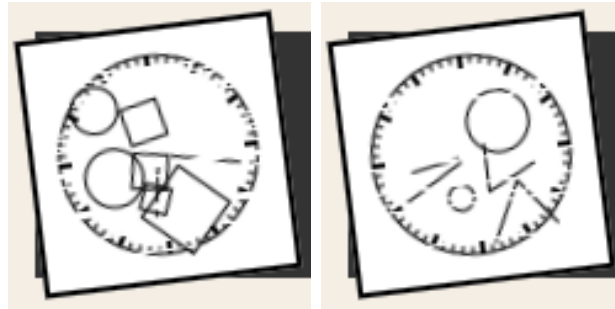


Figura 4: Risultato dell'utilizzo dei decoratori combinati tra loro.

Dove nel primo è stato utilizzato prima il `NoiseDecorator` con fattore 10 (su 100) e in seguito lo `ShapesDecorator` con quantità 8, mentre nella seconda il contrario.

Gli oggetti resi disponibili dalla libreria sono riassunti nella seguente tabella:

extends/implements	Type	Name	Constructor
-	Interface	ClockImageGenerationStrategy	-
ClockImageGenerationStrategy	Interface	HTMLCanvasStrategy	-
HTMLCanvasStrategy	class	HTMLCanvasGenerator	HTMLCanvasGenerator()
HTMLCanvasStrategy	abstract class	HTMLCanvasDecorator	-
HTMLCanvasDecorator	class	NoiseDecorator	NoiseDecorator(component:HTMLCanvasStrategy, noiseFactor: number)
HTMLCanvasDecorator	class	ShapesDecorator	ShapesDecorator(component:HTMLCanvasStrategy, shapePresence: number)

2.6 Verifica di un CAPTCHA

Una volta generati i dati e inseriti all'interno dell'oggetto `ClockCAPTCHAView`, all'occorrenza devono essere poi verificati utilizzando la funzione `validateData(data: object, password: string)`.

L'oggetto `data` contiene due valori:

- **token:** occorre un valore generato in precedenza mediante funzione `generateData()`, spiegata nel dettaglio la sezione precedente,
- **input:** stringa che rappresenta l'interpretazione dell'immagine, relativa ai dati criptati all'interno di `token`, fornita da un utente.

Dato quindi un oggetto come sopra definito, la funzione ritorna il valore booleano `true` se l'input corrisponde a dati nascosti all'interno di `token`, `false` altrimenti. Ovviamente per il corretto funzionamento della funzione il parametro `password` deve avere lo stesso valore del medesimo parametro fornito nella funzione `generateData()`.

```
if(ClockCAPTCHA.validateData(data,"SHARED_PWD")){
    //correct input handle
}else{
```



```
    //BAD_CAPTCHA handle  
}
```

3 Utilizzo dell'applicazione

Il *team Out of Bounds* ha sviluppato una *web-app* per dimostrare il funzionamento e l'efficacia della libreria sviluppata. In questa sezione vengono descritte le istruzioni per l'utilizzo dell'applicazione e le funzionalità che fornisce.

3.1 Requisiti

Per l'esecuzione di questa *web-app* devono essere soddisfatti alcuni requisiti:

- **Node.js:** *Angular* richiede *Node.js* per eseguire il suo ambiente di sviluppo. Assicurati di installare *Node.js* sul tuo sistema. Puoi scaricare l'ultima versione di *Node.js* dal sito ufficiale (<https://nodejs.org>),
- **npm:** *npm* (*Node Package Manager*) è un gestore di pacchetti per *Node.js*. Viene installato automaticamente insieme a *Node.js*. *npm* è utilizzato per gestire le dipendenze del progetto e installare i pacchetti necessari. Verrà utilizzato per installare *Angular CLI* e altre dipendenze del progetto.
- **Angular CLI:** *Angular CLI* (*Command Line Interface*) è uno strumento che semplifica la creazione e la gestione di progetti *Angular*. Puoi installare *Angular CLI* globalmente utilizzando il comando *npm* (*Node Package Manager*) dopo aver installato *Node.js*. Ecco il comando da eseguire nel tuo terminale:

```
npm install -g @angular/cli
```

3.2 Installazione

Per utilizzare la *web-app* clonare inizialmente i progetti localmente utilizzando `git clone` o scaricando manualmente i file `.zip` dalle corrispondenti *repository*:

- **Applicazione di front-end:** github.com/SWE-OutOfBounds/web-application,
- **Applicazione di back-end:** github.com/SWE-OutOfBounds/web-application-backend,
- **Libreria clock-captcha:** github.com/SWE-OutOfBounds/clock-captcha.

Assicurarsi di avere i tre progetti all'interno della stessa cartella ed eseguire i seguenti passi:

1. Clock-CAPTCHA:

- (a) Spostarsi sulla cartella `clock-captcha`,
- (b) Installare tutte le dipendenze richieste utilizzando il comando `npm install`,
- (c) Compilare il codice utilizzando il comando `npm run clean-build`

2. Back-end:

- (a) Spostarsi all'interno della cartella `web-application-backend`,
- (b) Installare tutte le dipendenze utilizzando `npm install`,



- (c) Assicurarsi di avere `mysql` installato nella macchina e di avere la seguente configurazione `{username: root, password: Password}`, per utilizzare utenti diversi modificare il file `.env` che si trova all'interno di `web-application-backend`,
- (d) Eseguire il codice reperibile nel file `DBINIT.sql` all'interno della cartella del progetto, in una console `mysql`,
- (e) Assicurarsi di utilizzare la versione `v19.x.x` di Node oppure impostare la versione utilizzando `nvm`,
- (f) Avviare il `server` utilizzando il comando
`npm run start`,
- (g) (Opzionale) Consultare la guida fornita collegandosi all'indirizzo `http://localhost/3000/api-docs`.

3. Front-end:

- (a) Spostarsi all'interno della cartella `web-application`,
- (b) (Opzionale) Consultare la guida fornita aprendo il file `/documentation/index.html` in un qualsiasi `browser`,
- (c) Installare tutte le dipendenze di `npm` utilizzando il comando `npm i`,
 - **Build:**
 - Eseguire il comando `ng build`,
 - `HostareG` l'applicazione utilizzando servizi di `hostingG` come *Netlify*, *Firebase Hosting*, *GitHub Pages* per o caricare i file sul `server` personale. Utilizzare in alternativa il pacchetto `http-host` disponibile in `npm` con il comando `http-server path/to/dist/web-application`.
 - **Testing:**
 - Eseguire il comando `ng serve -o` per far aprire in automatico il `browser` alla pagina corretta oppure utilizzare il comando `ng serve` e collegarsi al `link` generato.

3.3 Configurazione e avvio

3.3.1 MySQL

Per il funzionamento del *back-end* è necessario configurare il database utilizzando `mysql`. Per la sua inizializzazione eseguire il codice presente in `web-application-backend/DBINIT.sql`, reperibile all'interno della cartella del progetto in una *console SQL*.

Successivamente modificare opportunamente il file `.env` e aggiornare le voci `DB_USER` e `DB_PASSWORD` a seconda della propria configurazione.

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=Password
DB_NAME=PoCBackEnd
```

3.4 Dettagli Applicazione

3.4.1 Home Page

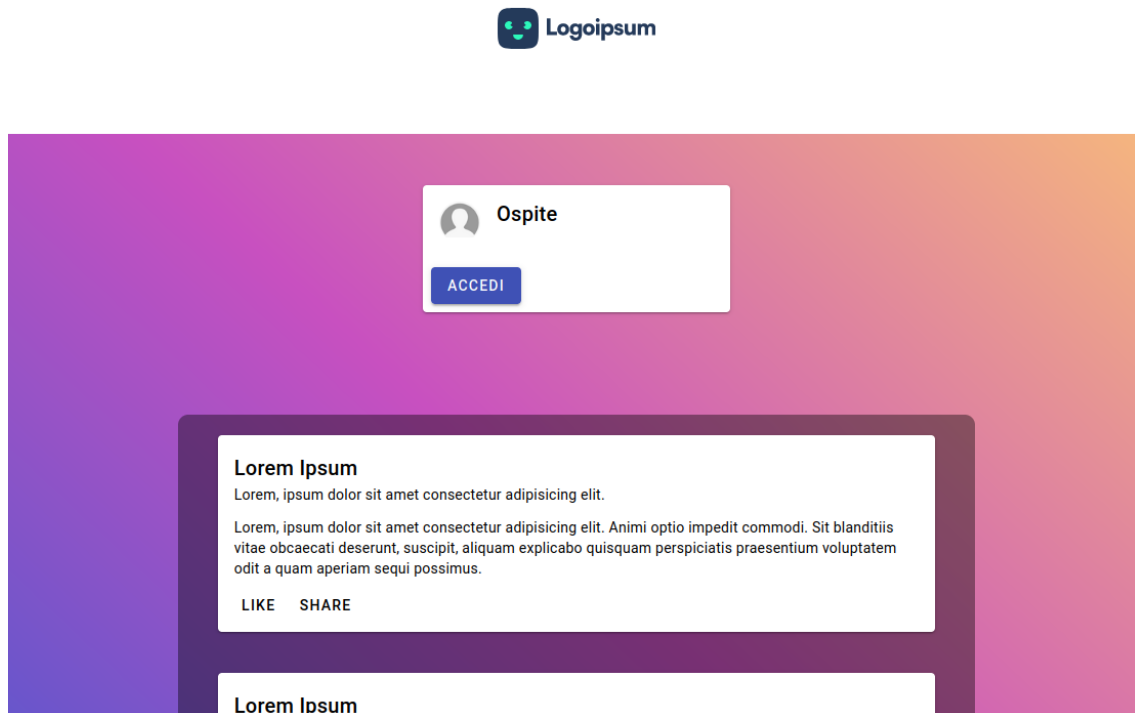


Figura 5: Home page dell'applicazione.

Nell'esempio riportato la navigazione avviene in modalità ospite, se la sessione è aperta la scritta ospite viene sostituita da nome utente e mail dell'utente in sessione e il pulsante **ACCEDI** viene sostituito dal pulsante **ESCI**. Per testare la funzione di accesso si può cliccare il pulsante "ACCEDI" (disponibile solamente all'utente Ospite).

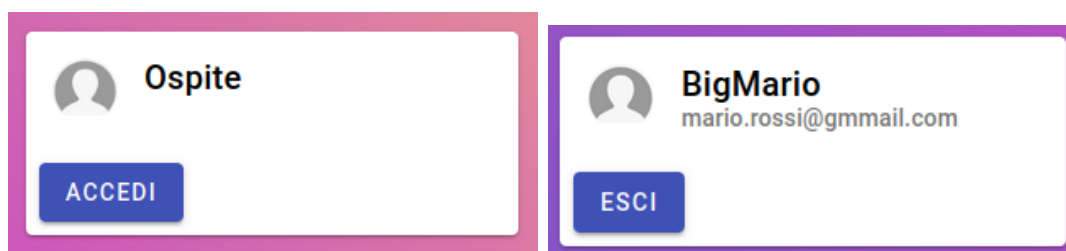


Figura 6: Dettagli della Home page.

3.4.2 Login

Per effettuare l'accesso collegarsi alla pagina dedicata mediante pulsanti o inserendo il valore `/login` in coda all'interno della barra di ricerca.

Per effettuare l'accesso inserire indirizzo *email* e *password*. La loro obbligatorietà viene notificata nel momento in cui si prova a lasciare un campo vuoto e comunque mediante disabilitazione

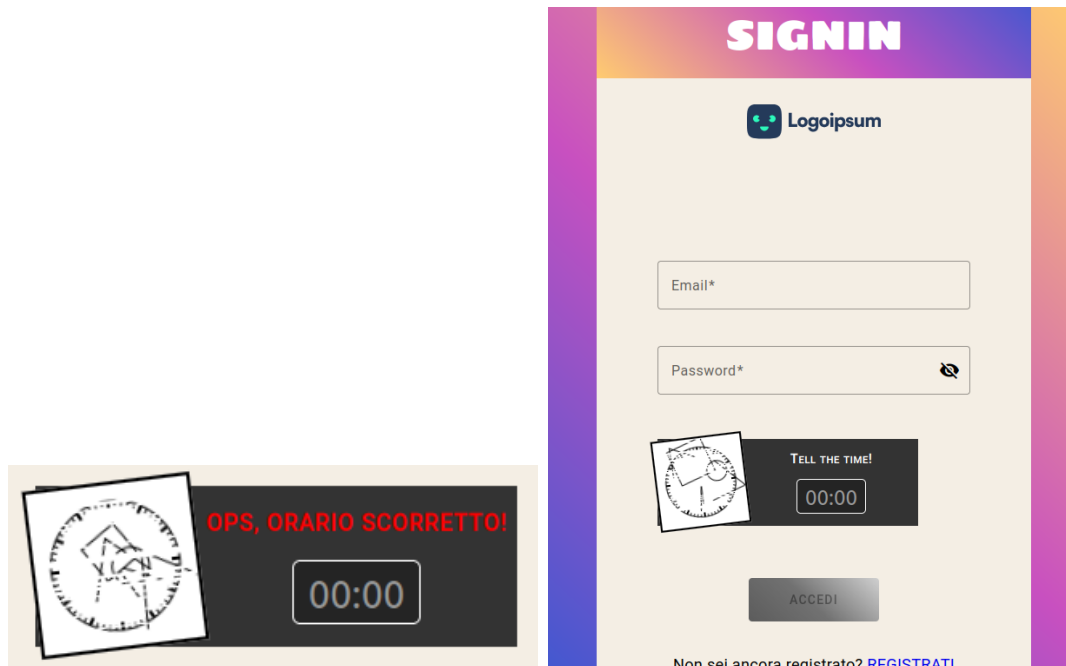


Figura 7: Dettagli pagina di accesso dell'applicazione.

del pulsante ACCEDI. Per poter effettivamente accedere occorre anche compilare il modulo di *CAPTCHA* utilizzando il campo d'inserimento sotto la dicitura "*Tell the time!*".

Sul campo *password* è presente l'icona per mostrare in chiaro oppure nascondere (*default*) la *password*.

Nel caso in cui le credenziali siano corrette e il compito venga risolto correttamente, l'utente verrà reindirizzato alla pagina principale dove potrà visualizzare i propri dati (*username* e *email*) e dove avrà la possibilità di fare il *log out* per chiudere la sessione. La funzione di accesso è inaccessibile a utenti con sessioni aperte.

Nel caso in cui le credenziali fornite non siano corrette, per formato o perché non presenti nel *database*, l'accesso al sistema sarà negato e l'utente visualizzerà il messaggio "Credenziali errate" in rosso sotto i campi "*email*" e "*password*".

L'accesso al sistema può fallire anche quando le credenziali sono corrette ma si verifica un errore nel *CAPTCHA*. In particolare, potrebbe verificarsi un errore quando:

- L'orario inserito è sbagliato, per l'accesso al sistema non viene eseguito e all'utente comparirà la scritta rossa "OPS, ORARIO SCORRETTO!",
- L'utente impiega troppo tempo per rispondere al test, per cui la risposta non sarà più ritenuta valida e comparirà la scritta rossa "Il tempo è volato! Riprova",
- Si registrano problemi con i dati forniti dal *CAPTCHA*, per cui la risposta non sarà più ritenuta valida e comparirà la scritta rossa "Qualcosa è andato storto. Riprova".

3.4.3 Registrazione

Per poter effettuare l'accesso occorre aver effettuato prima la registrazione utilizzando la pagina dedicata.

Collegatisi alla pagina di registrazione seguendo l'indirizzo `/signup` si possono utilizzare dei campi d'inserimento per inserire nome, cognome, *username*, *email* e *password*. Questi sono campi obbligatori e la loro obbligatorietà viene notificata all'utente nel momento in cui un campo viene lasciato vuoto e comunque mediante disabilitazione del pulsante REGISTRATI.

Sul campo *Password* è inoltre presente l'icona per mostrare oppure nascondere il contenuto del campo *password*. *Email* e *password* devono soddisfare determinati requisiti:

- L'indirizzo *email* inserito deve avere un formato valido definito dalla seguente espressione regolare

```
/^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$/
```

- La *password* invece deve rispettare il seguente formato

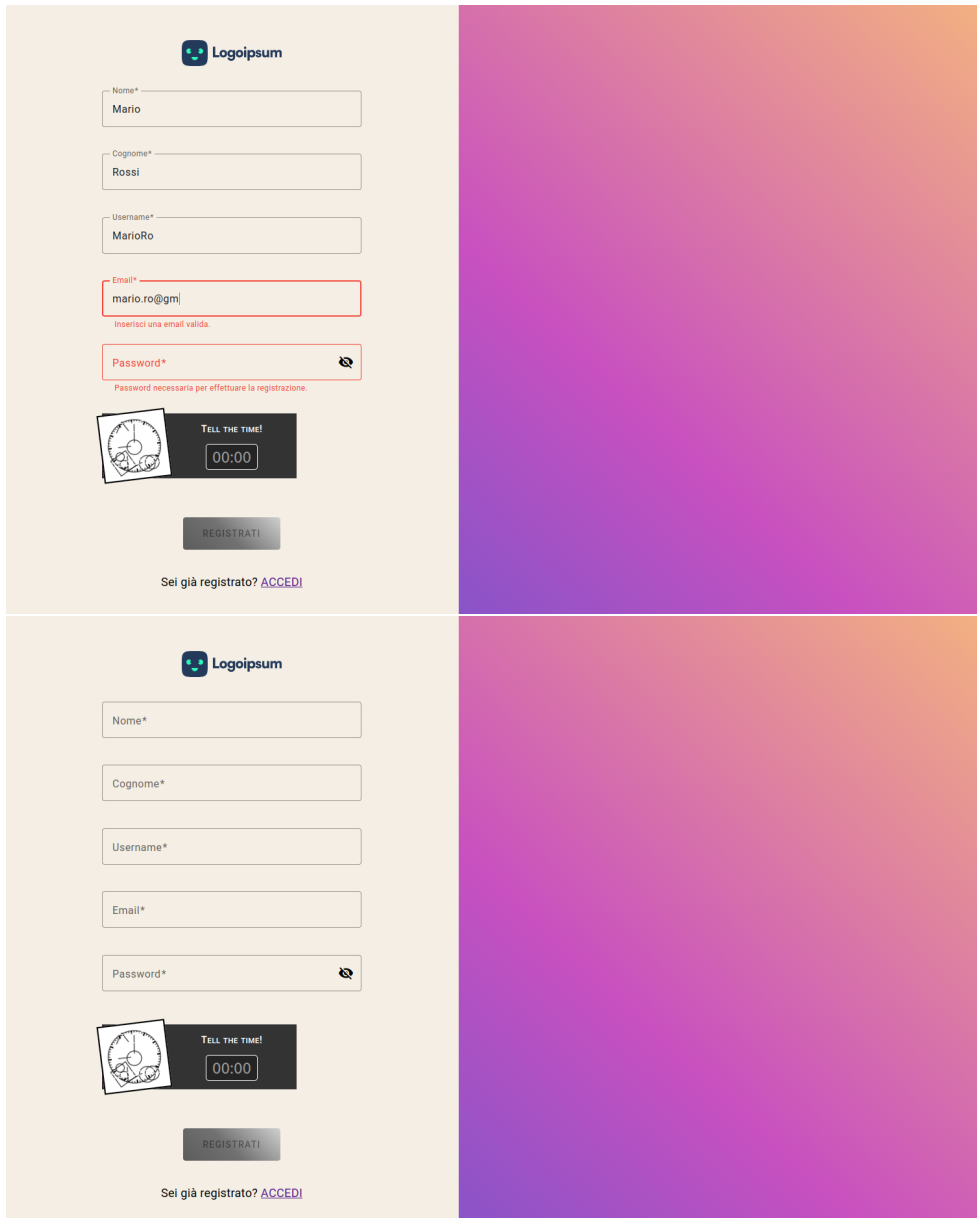
```
//8 caratteri  
//almeno un carattere maiuscolo  
//almeno un carattere minuscolo  
//almeno un numero
```

```
/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]).{8,}$/
```

In caso di valori inseriti non validi l'utente viene opportunamente informato.

Una volta inserite le credenziali e aver inserito correttamente l'ora indicata dall'orologio, cliccando sul pulsante "REGISTRATI" si effettua la registrazione al sistema.

Anche nella pagina di registrazione, nel caso in cui si rilevino errori nel *CAPTCHA* non sarà possibile portare a termine l'operazione di registrazione e l'utente visualizzerà un opportuno messaggio di errore.



Logoipsum

Nome*
Mario


Cognome*
Rossi

Username*
MarioRo

Email*
mario.ro@gm

Inserisci una email valida.

Password*
Password necessaria per effettuare la registrazione.

 **TELL THE TIME!**
00:00

REGISTRATI

Sei già registrato? [ACCEDI](#)

Figura 8: Pagina di registrazione.