Università degli Studi di Padova

Catch em All - ***CAPTCHA: Umano o Sovraumano?***
Email: catchemallswe3@gmail.com

Specifica architetturale

Versione	1.0.0
Approvazione	Zhen Wei Zheng
Redazione	Zhen Wei Zheng
Verifica	Zhen Wei Zheng
Stato	Approvato
Uso	Esterno
Distribuzione	Zucchetti S.p.A, Prof. Vardanega Tullio,
	Prof. Cardin Riccardo, Gruppo Catch Em All

Registro delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
0.0.6	05/06/2023	Inizio fine di §3	Zhen Wei Zheng	Progettista
0.0.5	02/06/2023	Inizio stesura di §3	Zhen Wei Zheng	Progettista
0.0.4	30/05/2023	Inizio fine di §2	Zhen Wei Zheng	Progettista
0.0.3	25/05/2023	Inizio stesura di §2	Zhen Wei Zheng	Progettista
0.0.2	20/05/2023	Stesura di §1	Zhen Wei Zheng	Progettista
0.0.1	17/05/2023	Creazione bozza e struttura del documento	Zhen Wei Zheng	Progettista

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di servire da linea guida per gli sviluppatori che andranno ad estendere o manutenere il prodotto. Di seguito lo sviluppatore troverà nel documento tutte le informazioni riguardanti i linguaggi e le tecnologie utilizzate, l'architettura del sistema e le scelte progettuali effettuate per il prodotto.

1.2 Scopo del prodotto

Dal proponente Zucchetti S.p.A. viene evidenziato, nel capitolato da loro proposto, una criticità negli attuali sistemi di sicurezza sulla rilevazione dei bot_G rispetto agli esseri umani. Oggi giorno il meccanismo più utilizzato per risolvere questo problema è il test CAPTCHA_G.
Un bot_G non è altro che una procedura automatizzata che, in questo caso, ha fini malevoli, come per esempio:

- Registrazione presso siti web;
- Creazione di spam_G;
- Violare sistemi di sicurezza.

I bot_G, grazie alle nuove tecnologie sviluppate con sistemi che utilizzano principalmente l'intelligenza artificiale, riescono a svolgere compiti che fino a poco tempo fa venivano considerati impossibili da svolgere per una macchina.
Ciò evidenzia che i CAPTCHA_G attuali risultano sempre più obsoleti, non andando a individuare correttamente tutti i bot_G, se non quasi nessuno.
Un'altra criticità individuata dal proponente è il sistema di classificazione delle immagini che sta effettuando Google grazie al proprio $\text{reCAPTCHA}_\text{G}$, che attualmente è il sistema più diffuso.
Questa criticità nasce dal beneficio che questa big tech_G ottiene dall'interazione degli utenti nel risolvere le task_G proposte, che portano alla creazione di enormi dataset_G di immagini classificate che possono essere utilizzate per l'apprendimento dei propri sistemi di machine learning o vendibili a terzi.
Il capitolato C1 richiede di sviluppare una applicazione web costituita da una pagina di login provvista di questo sistema di rilevazione in grado di distinguere un utente umano da un bot_G.
L'utente quindi, dopo aver compilato il form in cui inserirà il nome utente e la password, dovrà svolgere una task_G che sarà il cosiddetto test CAPTCHA_G.

1.3 Glossario

Per evitare ambiguità relative al linguaggio utilizzato nei documenti prodotti, viene fornito il **Glossario v 1.0.0**. In questo documento sono contenuti tutti i termini tecnici, i quali avranno una definizione specifica per comprenderne al meglio il loro significato.
Tutti i termini inclusi nel Glossario, vengono segnalati all'interno del documento Piano di qualifica con una G a pedice.

1.4 Standard di progetto

Per lo svolgimento del progetto il gruppo *Catch Em All* ha scelto di utilizzare come norme di riferimento informativo la serie di standard **ISO/IEC 25000 SQuaRE** per definire i requisiti$_G$ e le metriche per valutazione della qualità di un prodotto e lo standard **ISO/IEC 15504 SPICE** per definire al meglio la qualità e le metriche di un processo.

1.5 Riferimenti

1.5.1 Riferimenti normativi

Riferimenti normativi utilizzati:

- Norme di Progetto v1.0.0;
- Capitolato d'appalto C1 *CAPTCHA: Umano o Sovrumano?* : `https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C1.pdf`.

1.5.2 Riferimenti informativi

Riferimenti informativi utilizzati:

- Diagrammi delle Classi - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf`;
- Software Architecture Patterns - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf`;
- Design Pattern Architetturali - Materiale didattico del corso di Ingegneria del Software:
  - Dependency Injection: `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Architetturali%20-%20Dependency%20Injection.pdf`;
  - Model View Controller: `https://www.math.unipd.it/~rcardin/sweb/2022/L02.pdf`.
- Design Pattern Creazionali: `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Creazionali.pdf`;
- Design Pattern Strutturali: `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Strutturali.pdf`;
- Design Pattern Comportamentali: `https://www.math.unipd.it/~rcardin/swea/2021/Design%20Pattern%20Comportamentali_4x4.pdf`;
- SOLID Principles of Object-Oriented Design: `https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf`.

2 Tecnologie coinvolte

2.1 Tecnologie per la codifica

2.1.1 Linguaggi

Tecnologia	Versione	Descrizione
PHP	8.1	Linguaggio principale per lo sviluppo del progetto, usato per gestire tutti componenti dei model-view-controller. E' il linguaggio del framework Laravel
HTML	Versione	Linguaggio di markup utilizzato per impostare la struttura delle pagine web.
CSS	Versione	Utilizzato per la formattazione e la definizione dello stile dei documenti HTML.
JavaScript	Versione	Utilizzato per implementare il POW (proof of work) una volta richiesta la generazione del catpcha.
Python	3.10.9	Utilizzato per creazioni degli script di download e elaborazione delle immagini da Unsplash.

Tabella 2.1: Linguaggi utilizzati

2.1.2 Strumenti

Tecnologia	Versione	Descrizione
Composer	2.5.5	Gestore di dipendenze per il linguaggio di programmazione PHP.
NodeJS(npm)	8.1.0	Consente di installare, gestire e condividere pacchetti e moduli di terze parti per lo sviluppo di applicazioni Node.js.
Sqlite	Versione	Utilizzato per salvare i dati;
Scribe	4.19	Utilizzato per la generazione della documentazione.

Tabella 2.2: Strumenti utilizzati

2.1.3 Framework e librerie

Tecnologia	Versione	Descrizione
Laravel	10.10	Framework open-source scritto in PHP, semplifica lo sviluppo delle applicazioni web offrendo una vasta gamma di funzionalita'.
Sanctum	3.2	Descrizione

Tabella 2.3: Framework e librerie utilizzati

2.2 Strumenti per l'analisi del codice

Strumento	Versione	Descrizione
GitHub Action	-	Strumento fornito da GitHub che permette di definire workflows personalizzati all'interno di repository.
Phpunit	10.1	Strumento per l'analisi statica del codice;
Xdebug	-	Strumento per il code coverage.

Tabella 2.4: Strumenti per analisi utilizzati

3 Architettura

L'obiettivo principale del progetto è la creazione di un servizio captcha insieme a un'applicazione web dimostrativa che ne faccia uso, al fine di illustrarne il funzionamento. Per raggiungere questo obiettivo, il prodotto è stato realizzato come un'API REST.

La componente centrale dell'applicazione risiede nel back-end, dove è stato utilizzato il framework Laravel per la codifica.

La parte back-end è responsabile principalemnte della gestione delle richieste della generazione e della verifica del servizio captcha. D'altro canto, per la parte front-end, è stata creata una semplice pagina HTML che utilizza JavaScript per fare richieste all'API REST. Questa pagina fornisce un'interfaccia utente elementare attraverso la quale gli utenti possono interagire con il servizio captcha.

3.1 Diagrammi delle classi

Il diagramma delle classi si può suddividere in varie sezioni, le quali svolgono ognuna una parte portante del progetto. Le sezioni sono:

- Generazione del Captcha;
- Captcha Resource;
- Verifica del Captcha;
- Gestione delle chiavi per l'encrypt e decrypt delle soluzioni del captcha;
- Download e elaborazione delle immagini.

Per lo sviluppo delle varie funzionalità elencate il gruppo ha sfruttato diversi strumenti forniti dal framework Laravel, come gli eloquent Model che forniscono un metodo semplice ed intuitivo per interagire con le entità presenti nel DB, e delle classi Controller per poter gestire la logica di gestione delle richieste nel modo più chiaro possibile. Infatti i Controllers possono raggruppare la logica di gestione delle richieste correlate in una singola classe.

3.1.1 Generazione del Captcha

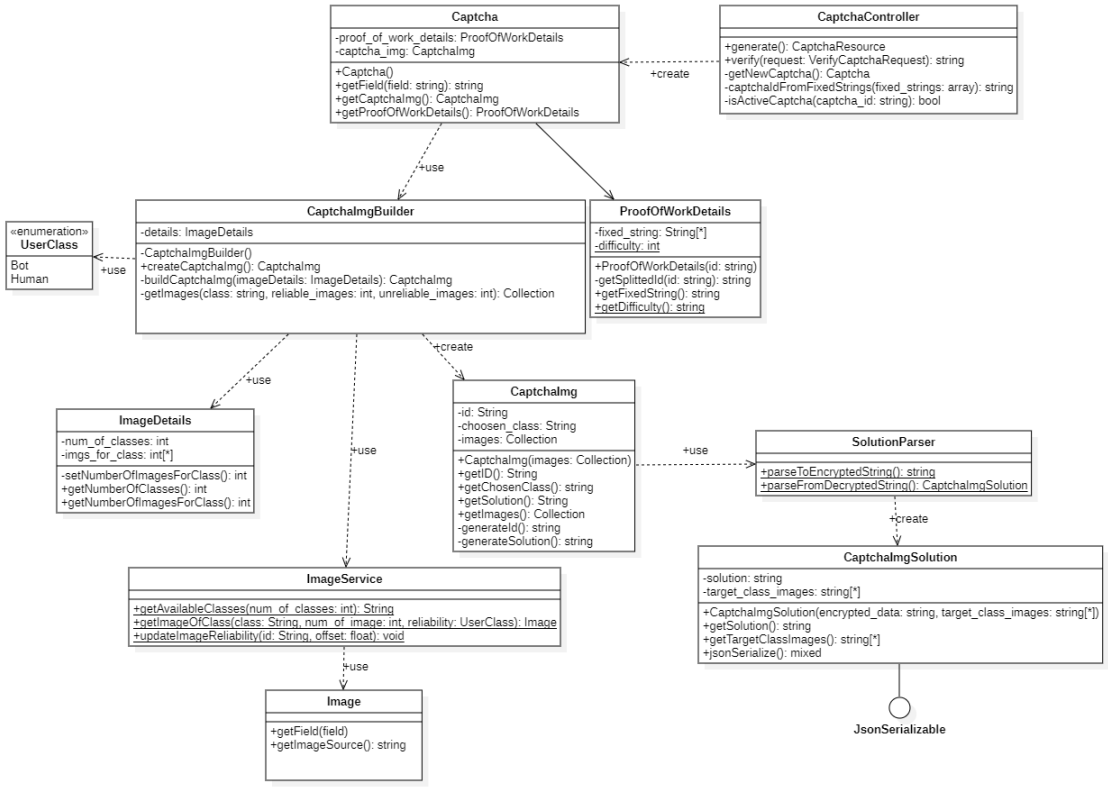

Figura 3.1: Diagramma delle classi. Generazione del Captcha

Nella figura sopra viene mostrata la logica utilizzata per la generazione di un Captcha.

- Ogni Captcha è composto da:
  - 9 immagini in bianco e nero con solo i contorni visibili;
  - L'immagine honeypot, invisibile all'utente;
  - Il proof of work.

In particolare:

- **CaptchaController**: Classe cardine del progetto, la quale si occupa di gestire tutte le richieste che verranno fatte alla route dell'API. In questa classe è anche contenuto il metodo che ritornerà l'oggetto CaptchaResource a seguito di una richiesta da parte dell'utente;
- **Captcha**: Classe che si occupa di mettere insieme tutti i pezzi che compongono il Captcha, ovvero immagini, honeypot e proof of work. Questa classe viene creata dal Controller quando viene richiesto un Captcha;
- **CaptchaImgBuilder**: Classe singleton che si occupa dell'effettiva creazione della parte del Captcha che contiene le immagini e l'honeypot. Fa inoltre utilizzo di una enumerazione per la gestione dell'affidabilità delle immagini;
- **ProofOfWorkDetails**: Classe che contiene i dati necessari per il calcolo dei nonce nel proof of work;
- **ImageDetails**: Classe che contiene i dettagli utili alla creazione delle 9 immagini che comporranno il Captcha, quali il numero di classi e di immagini per classe che dovrà avere. Viene utilizzata dalla classe CaptchaImgBuilder;
- **CaptchaImg**: Classe che contiene le varie informazioni che la parte del captcha composta dalle immagini deve avere. È l'oggetto che la classe CaptchaImgBuilder va a creare;
- **ImageService**: Classe che contiene i metodi per le chiamate al DB riguardanti il Model delle immagini. Viene dalla classe CaptchaImgBuilder per recuperare le immagini da inserire nel Captcha;
- **Image**: Eloquent Model che contiene le informazioni dell'oggetto immagine. Viene utilizzato da ImageService;
- **Solution Parser**: Classe che gestisce l'encrypting della soluzione, che è in formato json, così che l'utente non possa decifrarla;
- **CaptchaImgSolution**: Classe che contiene le informazioni della soluzione della parte del Captcha composta dalle immagini.

3.1.2 Captcha Resource

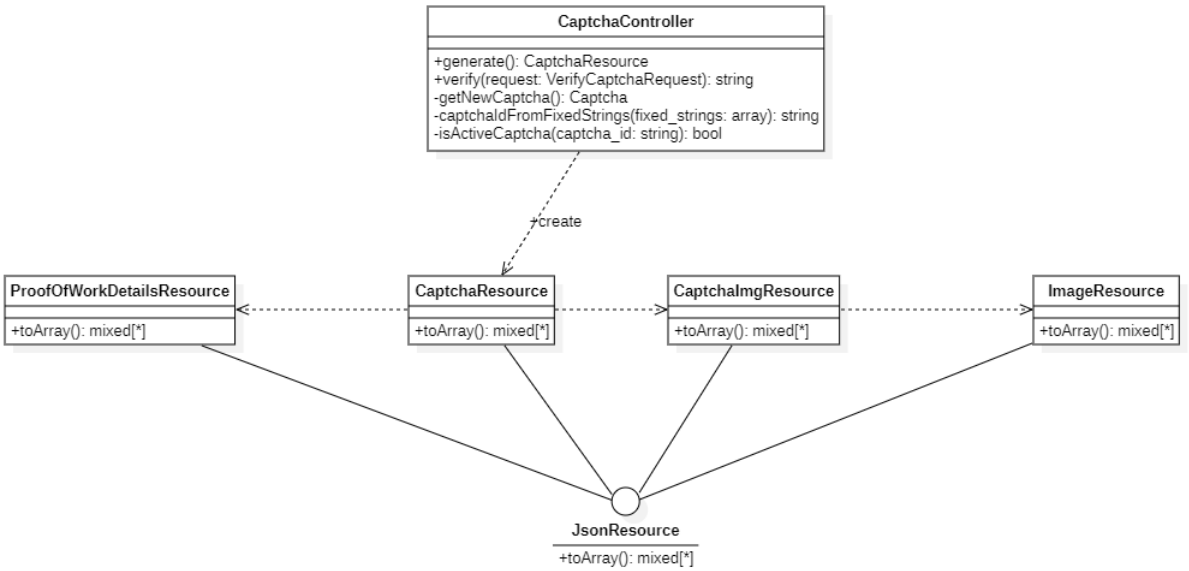

Figura 3.2: Diagramma delle classi. Captcha resource

Nella figura sopra viene mostrata la logica utilizzata per la gestione delle informazioni che si vogliono passare all'utente al seguito di una richiesta. Infatti dopo la creazione di un captcha, viene creato dal Controller l'oggetto CaptchaResource, il quale sarà un json che contiene solamente le informazioni chiave per il caricamento del captcha lato front-end e la soluzione criptata che servirà durante la verifica del Captcha.

In particolare:

- **CaptchaController**: Dopo aver creato il Captcha, genera l'oggetto CaptchaResource il quale contiene le varie informazioni da passare in risposta alla richiesta dell'utente;
- **CaptchaResource**: Classe Eloquent Resource che utilizza le classi CaptchaImgResource e ProofOfWorkDetailsResource per generare il json che verrà poi inviato all'utente;
- **CaptchaImgResource**: Classe Eloquent Resource che seleziona le informazioni da inviare all'utente per la parte del Captcha immagini. Utilizza la classe ImageResource per le informazioni delle immagini contenute nel Captcha;
- **ProofOfWorkDetailsResource**: Classe Eloquent Resource che seleziona le informazioni da inviare all'utente per la parte del proof of work;
- **ImageResource**: Classe Eloquent Resource che seleziona le informazioni che ogni immagine dovrà avere nella risposta all'utente.

3.1.3 Verifica del Captcha

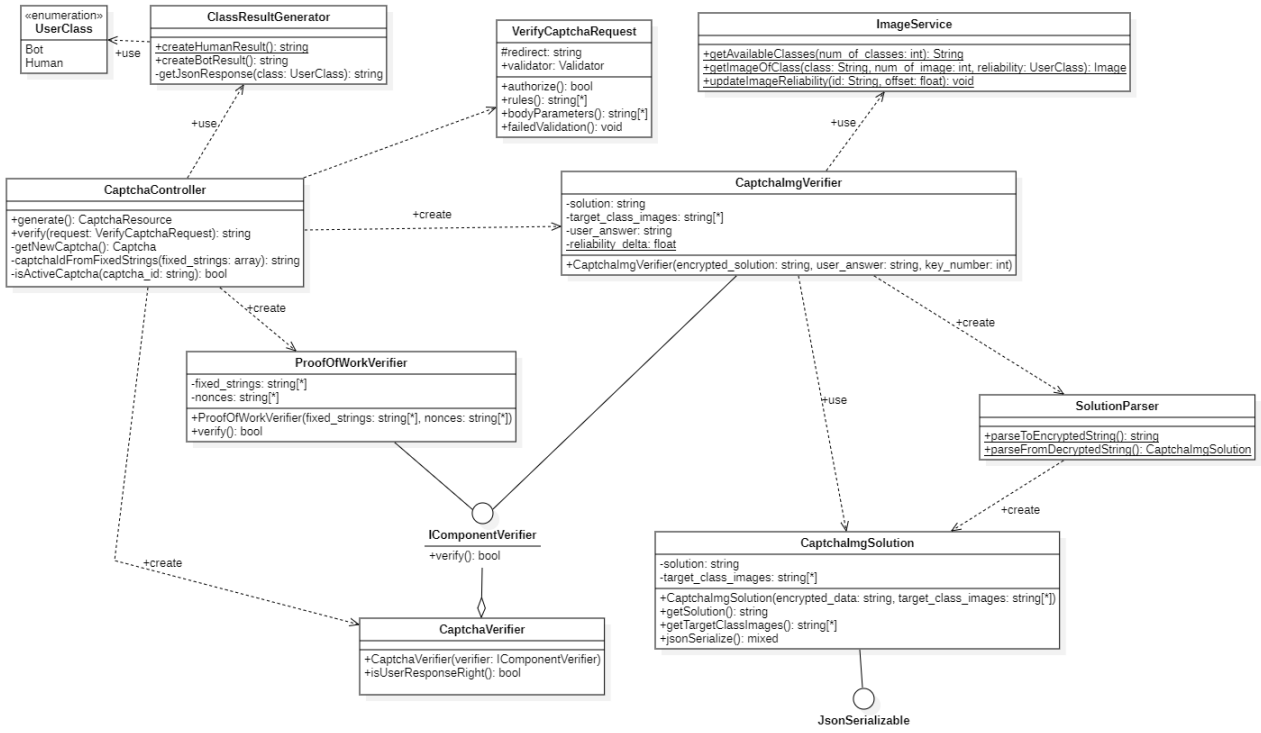

Figura 3.3: Diagramma delle classi. Verifica del Captcha

Nella figura sopra viene mostrata la logica utilizzata per la verifica dei Captcha inviati dagli utenti. Come punto centrale c'è sempre il CaptchaController che gestisce tutte le richieste. Quest'ultimo andrà a creare i vari verificatori che controlleranno se l'utente ha dato una risposta valida o no. Viene poi generato un json che verrà criptato e inviato come risposta. In particolare:

- **CaptchaController**: Questa classe si occupa anche di gestire le richieste per la verifica di un Captcha, andando prima a validare se la risposta inviata coerente a quella aspettata e poi andando a creare i verificatori che andranno a controllare la correttezza effettiva del Captcha;
- **VerifyCaptchaRequest**: Classe che controlla se la risposta inviata dall'utente è nel formato richiesto;
- **CaptchaVerifier**: Questa classe utilizza i verificatori CaptchaImgVerifier e ProofOfWorkVerifier per controllare se la risposta dell'utente è corretta o meno;
- **IComponentVerifier**: Questa classe possiede il metodo verify che verrà implementato dai due verificatori;

- **CaptchaImgVerifier**: Classe che verifica la correttezza della soluzione dell'utente per la parte del Captcha composto dalle immagini, andando inoltre ad utilizzare la classe ImageService per aggiornare l'affidabilità delle immagini nel DB;
- **ProofOfWorkVerifier**: Classe che verifica la correttezza dei nonce calcolati dall'utente durante il proof of work;
- **SolutionParser**: Classe utilizzata da CaptchaImgVerifier per decriptare la soluzione corretta del Captcha inviata alla'utente;
- **CaptchaImgSolution**: Classe che contiene gli attributi della soluzione del Captcha. Creata da SolutionParser per fornire la soluzione decriptata a CaptchaImgVerifier;
- **ImageService**: Classe utilizzata dal CaptchaImgVerifier per aggiornare l'affidabilità delle immagini all'interno del DB;
- **ClassResultGenerator**: Classe che genera il json criptato che sarà dato in risposta alla soluzione inviata dall'utente. Questo json contiene il risultato dell'operazione di verifica.

3.1.4 Gestione delle chiavi e dell'encrypting

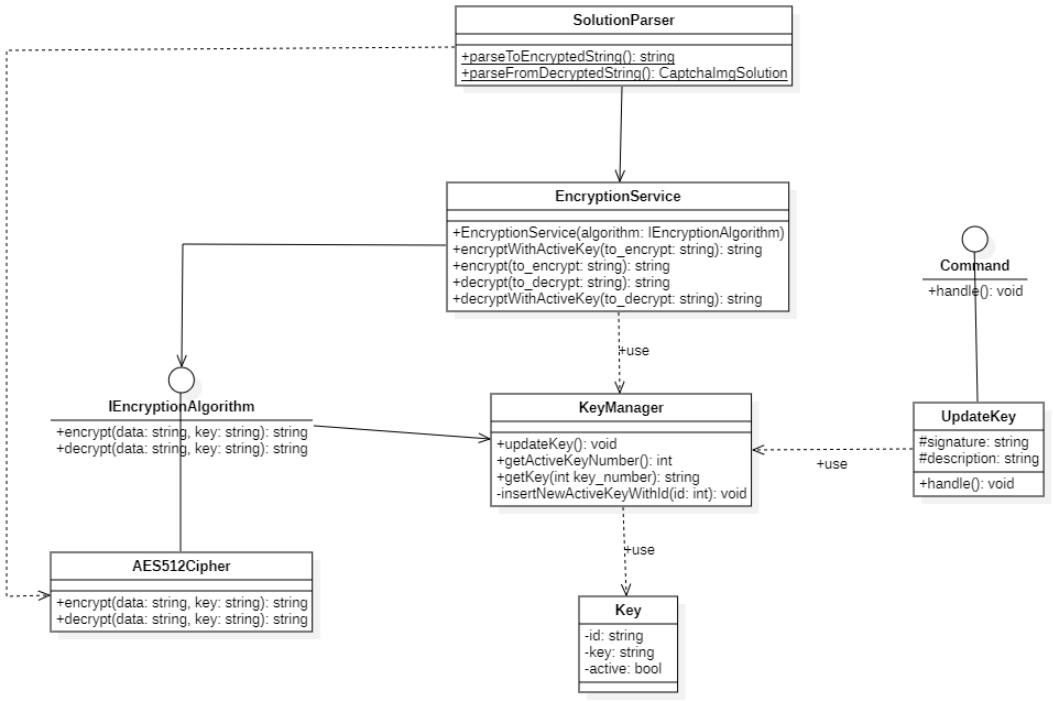

Figura 3.4: Diagramma delle classi. Gestione delle chiavi e dell'encrypting

3.1.5 Download e elaborazione delle immagini

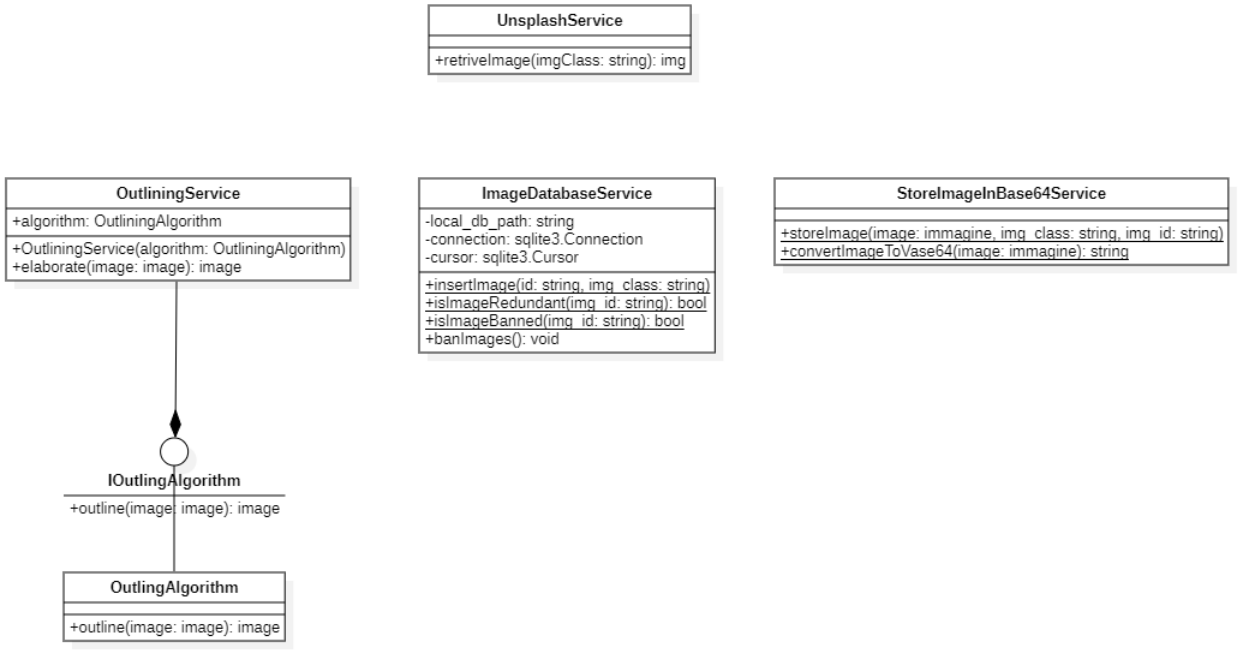

Figura 3.5: Diagramma delle classi. Recupero delle immagini da Unsplash

La figura sopra riportata rappresenta il metodo di download delle immagini dal sito di Unsplash, la rielaborazione di tali immagini e il metodo di salvataggio. Vengono eseguiti con il seguente ordine:

1. Viene scaricato l'immagine di una determina classe;
2. Rielaborazione dell'immagine eliminando i colori e tenendo solo il contorno.
3. Conversione dell'immagine in base64;
4. Salvataggio dell'immagine del database.

In particolare:

- **UnsplashService**: download dell'immagine;
- **OutliningService**: elaborazione dell'immagine;
- **OutilningAlgorithm**: algoritmo di elaborazione dell'immagine;

- **StoreImageInBase64Service**: salvataggio dell'immagine in formato base64 usando come path *classe/id immagine*;
- **ImageDatabaseService**: salvataggio dell'immagine nel database.

3.2 Architettura di dettaglio

3.2.1 Strategy pattern

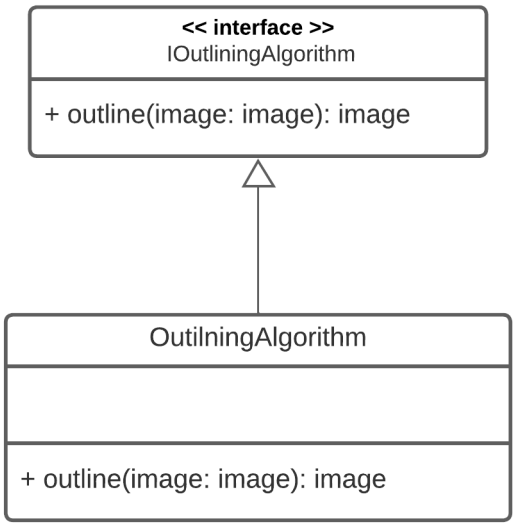

Figura 3.6: Strategy pattern dell'algoritmo di rielaborazione dell'immagine

Attualmente, la classe astratta "IOutliningAlgorithm" ha una sola classe concreta che rappresenta l'unico algoritmo implementato. Tuttavia, è progettata in modo da consentire l'estensione per l'implementazione di ulteriori algoritmi in futuro. Ciò significa che se si desidera aggiungere nuovi algoritmi, sarà possibile creare nuove classi che estendono l'interfaccia astratta "IOutliningAlgorithm". Questa flessibilità consente di espandere il sistema per includere più algoritmi di scontornaggio senza dover modificare la struttura di base.

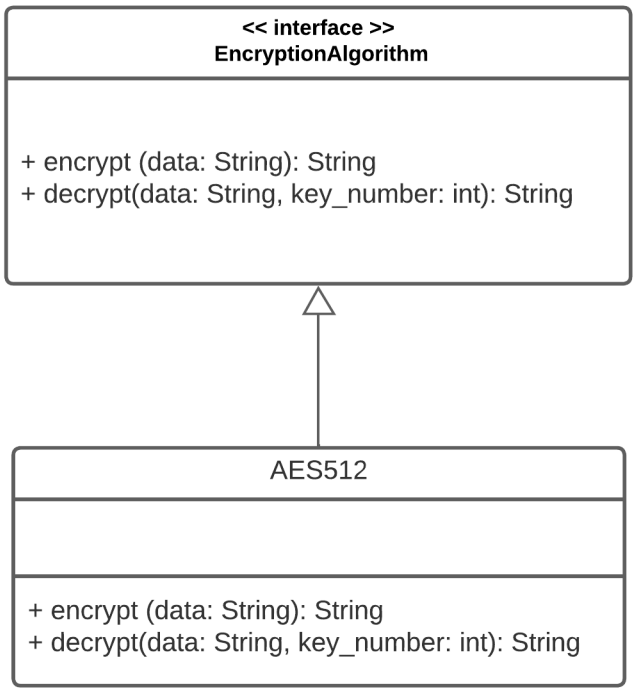

Figura 3.7: Strategy patter dell'algoritmo di criptaggio

Come sopra, la classe astratta "EncryptionAlgorithm" ha una sola classe conreta per il motivo sopra citata.

3.2.2 Dependency Injection

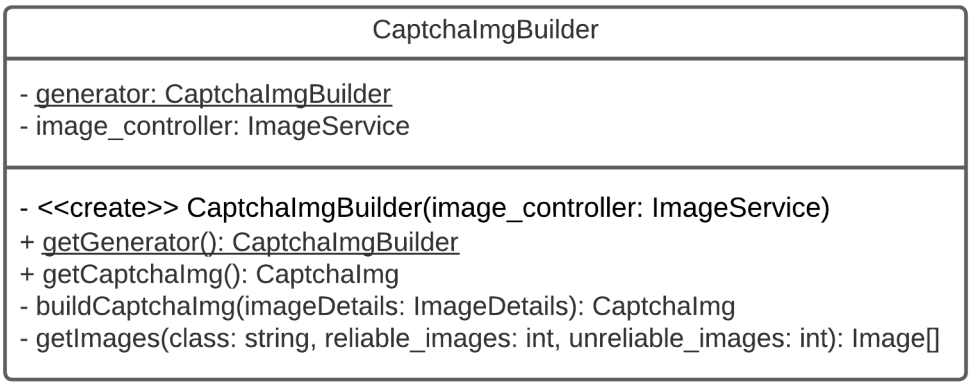

Figura 3.8: Singleton pattern

La costruzione di un'instanza di CaptchaImg è un'operazione che può essere considerata complessa in quanto bisogna tenere in considerazione variabili come numero di classi di immagine presenti e numero di immagini per classe, tali variabili devono a loro volta rispettare delle condizioni:

- numero di classi tra 2 e 4 estremi compresi
- 9 immagini totali
- le immagini della classe target devono essere la metà + 1 affidabili
- le immagini delle classi non target devono essere la metà affidabili

Per quanto sopra si è deciso si spezzare la complessità e creare una classe il cui compito è quello di rifornire il costruttore di CaptchaImg.

3.2.3 Strategy Pattern

Il pattern Strategy è stato utilizzato per l'algoritmo di scontornamento delle immagini e per quello di cifratura. L'utilizzo di tale pattern permette una futura estensione delle famiglie degli algoritmi in questione.

3.2.4 Model-View-Controller

Il pattern architetturale su cui si basa il framework Laravel è *Model-View-Controller*. Lo sviluppo del servizio API per la generazione e verifica del captcha sfrutta la classi messe a disposizione del framework:

- Illuminate
  Database
  Eloquent
  Model: classe astratta per lo sviluppo di modelli la quale fornisce attributi e metodi integrati per la comunicazione con il database

- Illuminate
  Routing
  Controller: classe astratta per lo sviluppo di classi il cui compito è la gestione delle route e dei modelli

Le viste trattandosi di un API non sono state utilizzate.

Questo pattern inoltre è alla base dello sviluppo dell'applicazione web utilizzata per testare il servizio di Captcha.