.7

Scelta e motivazione delle tecnologie

Versione 1.0.0

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

Progetto Ingegneria del Software

Dipartimento di Matematica
Università degli Studi di Padova

31 gennaio 2023

Contatti: dotseventeam@gmail.com

Registro delle versioni

Versione	Data	Autore	Ruolo	Motivazione
1.0.0	31/01/2023	Martina Garon	Responsabile	Validazione
0.2.0	31/01/2023	Noemi Liva	Verificatore	Stesura punti relativi a PostgreSQL e React
0.1.1	31/01/2023	Ruth Genevieve Bousapnameme	Analista	Stesura punti relativi a PostgreSQL e React
0.1.0	13/01/2023	Ivan Antonino Arena	Verificatore	Verifica generale stesura premessa, classi di tecnologie e motivazione delle scelte
0.0.1	13/01/2023	Marco Marchiante	Analista	Stesura premessa, classi di tecnologie e motivazione delle scelte

Tabella 1: Registro di versionamento del documento

Indice

1 Premessa

La scelta delle tecnologie da utilizzare nella realizzazione del progetto avviene in più fasi di seguito documentate.

La prima sezione **§2** deriva direttamente da una prima fase di brainstorming sul testo del capitolato e una prima analisi del problema individuando così delle "classi astratte" di tecnologie. Quest'astrazione verrà successivamente concretizzata in tecnologie specifiche basate sulle necessità funzionali del proponente e tecniche del progetto.

La seconda sezione **§3** deriva dall'esperienza di stesura di un PoC esplorativo dove sono state testate e validate alcune tecnologie concrete che calzano le necessità specifiche del caso in esame. La scelta definitiva delle singole tecnologie e relative versioni è avvenuta dopo la presentazione del PoC al proponente che ha ritenuto opportune le scelte in base ad un analisi pesata sui requisiti proposti, rendendo la scelta una conseguenza indiretta dei requisiti analizzati. L'analisi delle specifiche tecnologie basata sui requisiti fa parte di questo documento.

2 Classi di tecnologie

Inizialmente è stata individuata una bozza di architettura del sistema (derivante dai requisiti lato soluzione) per cui un **applicazione Web** si interfaccia con un **servizio** che fornisce le funzionalità di CAPTCHA (quindi fornitura e validazione del risultato).
Di seguito sono elencate le classi di tecnologie individuate in una fase iniziale di analisi del problema:

- **Framework di sviluppo web**: Essendo richiesta una piattaforma web protetta da CAPTCHA, è opportuno l'utilizzo di un framework per costruirla utilizzando costrutti pre-validati che possano agevolare lo sviluppo e garantire un certo livello di manutenibilità del codice.
- **REST API**: Viene richiesto un sistema CAPTCHA che protegga determinate funzionalità di un'applicazione web. Considerando la scalabilità del sistema, è sensato che esso sia un servizio a se stante che possa servire più applicazioni web in maniera semplice. Riteniamo che la tecnologia REST sia un metodo efficace di ottenere questo obiettivo poiché fornisce una interfaccia standard, affidabile, potente e molto flessibile.
- **Server web**: L'applicazione e il servizio devono essere serviti in maniera tale da essere accessibili dalla rete globale rendendo un server web una necessità insostituibile.
- **Containerizzazione**: Riteniamo inoltre che confinare ogni componente del sistema in un ambiente standard ed immutabile di esecuzione possa giovare all'affidabilità e scalabilità del sistema portando solo benefici.
- **Database relazionale**: Un database è una componente essenziale in ogni sistema moderno in quanto fornisce un sistema potente ed affidabile di immagazzinare dati ed averne accesso in maniera organizzata.
- **Libreria di elaborazione di immagini**: L'intendo di realizzare un sistema CAPTCHA basato su immagini (in particolare sovrapposizione in trasparenza) richiede l'abilità di elaborare e modificare immagini, serve dunque una libreria.
- **Dataset di immagini**: Per realizzare un sistema CAPTCHA basato su immagini, è necessario avere a disposizione un set di immagini da utilizzare che siano propriamente etichettate per permettere al motore di CAPTCHA di verificare la risposta senza inciampare nel suo stesso scopo.

3 Scelta e motivazione delle tecnologie:

3.1 Java

Riferimenti: Java, Java 17 Docs

Java è la *tecnologia di backend* scelta in base ad un analisi ponderata su requisiti individuati in principio, basata sulle seguenti considerazioni:

PRO:

- Supporto nativo alla *programmazione di rete / web / API*;
- *Ampia collezione di librerie di prima e terze parti* che forniscono funzionalità aggiuntive e iper-specializzate;
- *Libreria standard* molto ampia che incoraggia un buono stile di programmazione;
- Ottima ed ampia *documentazione* di ogni componente sviluppata nel tempo;
- Disponibilità di documentazione non ufficiale in forma di *"tutorial", "articoli" e "domande e risposte" curate dalla community* e disponibili in quantità sul web;
- *Standard de facto* per lo sviluppo di sistemi enterprise e REST API.

CONTRO:

- *Probabilmente lento* durante l'esecuzione se interpretato dalla JVM.

La scelta è ricaduta su Java per la sua versatilità e la disponibilità abbondante di risorse, librerie, framework e guide che possono aiutare nello sviluppo. Il principale problema di questa tecnologia é l'intensivo utilizzo di risorse derivante dall'interpretazione del linguaggio, ma ciò può essere mitigato in gran parte utilizzando tecniche di compilazione JIT.
La specifica versione scelta è la 17 in quanto la sua natura LTS garantisce supporto per un lungo periodo [1].

3.2 Spring Boot

Ref: Spring Boot

Spring boot è stato scelto come *framework di sviluppo web* in seguito alla scelta della tecnologia di backend (*Java*), in quanto framework principalmente utilizzato oggi per sviluppare per il Web in *Java*.
Supporta lo sviluppo di applicazioni web tradizionali HTTP e REST API, riducendo così tempi e costi di sviluppo.

[1] Come possiamo vedere da Oracle Java SE Support Roadmap, Java 17 è l'ultima versione a cui è garantito un supporto prolungato LTS (Long Time Support) con un attuale EOL minima stimata per Settembre 2029 garantendo almeno altri 6,5 anni di patch di sicurezza.

3.3 Java2D

Ref: Java 2D

Java 2D è l'API che la libreria standard di Java mette a disposizione per l'interazione con le immagini.
Permette interazioni anche avanzate con le immagini e copre le necessità dell'applicazione, cioè la composizione e l'alpha-transparency che sono essenziali per la produzione delle immagini che l'utente dovrà "decifrare".
Oltre a ciò, essendo disponibile da molto tempo, è molto affidabile e ben documentata [2] .

3.4 PostgreSQL

Riferimenti: PostgreSQL

PostgreSQL è un database relazionale, completamente open source, esistente nel mercato dal oltre 20 anni.
Rispetto ad altri database relazionali open source [3] offre:

- Un'elevata scalabilità sia in termine di quantità di dati, sia in quantità di utenti che si possono soddisfare contemporaneamente.
- Una maggiore efficienza nella lettura e scrittura dei dati, in particolare:
  - Supporto di vista materializzata per join costosi frequentemente eseguiti;
  - Supporto di index parziali per migliorare l'efficienza di query relative a dati frequentemente richiesti.
- Replicazione primaria-secondaria, replicazione streaming e replicazione sincrona su altri server PostgreSQL, permettendo una migliore tutela dei dati.

I punti appena elencati e la sua popolarità nel mercato [4] lo rendono un perfetto strumento per lo sviluppo del nostro progetto.

3.5 React.js

Riferimenti: React.js

React è una libreria JavaScript front-end gratuita e open source, presente nel mercato da oltre 10 anni, ideato per la creazione di interfacce utente basate sui componenti della UI. Confrontato con i suoi rivali nel mercato [5], React risulta la scelta migliore in quanto:

- É il piú utilizzato tra i front-end;
- Ha una curva di apprendimento meno ripida rispetto ai rivali;
- La documentazione é esaustiva.

[2] Questa API è disponibile fin da Java 8 e non è cambiata molto nel tempo. Di conseguenza gli sviluppatori hanno avuto il tempo di scrivere documentazione molto completa e le guide prodotte nel tipo e pubblicate sul Web sono ancora attuali.
[3] Confronto: PostgreSQL vs MariaDB
[4] Al momento della stesura di questo documento, é quarto in DB-Engines
[5] Confronto: React vs Angular vs Vue

3.6 HTML5 e CSS3

Riferimenti: HTML5, CSS3

Quando si parla di sviluppo per il web, **HTML** e **CSS** sono il linguaggio universale costruzione delle pagine essendo di conseguenza una scelta obbligata.
Riguardo alle specifiche versioni dello standard da adottare, possiamo tranquillamente [6] utilizzare le ultime HTML5 e CSS3 dato che sono supportate dalla maggior parte dei dispositivi al giorno d'oggi.

3.7 Docker

Riferimento: Docker

La scelta è ricaduta su questo particolare *strumento di containerizzazione* principalmente perché è un ottimo strumento con il quale parte del team ha confidenza ed è in grado di gestire perfettamente l'uso che ne verrà fatto.

PRO:

- Anche nella versione *free* non costituisce ostacoli;
- Ottimo supporto fornito dalla community;
- Ricca libreria di container pre-configurati;
- Si integra facilmente con la maggior parte degli ambienti di sviluppo;
- Possibilità di utilizzo in locale;
- Possibilità quasi illimitate;
- Scalabilità quasi infinita con software come **kubernetes**.

CONTRO:

- Curva di apprendimento ripida;
- Può essere complesso gestire sistemi particolarmente complessi;
- Non essendo progettato per memoria persistente richiede qualche configurazione aggiuntiva.

3.8 THINGS 4 All

Ref: THINGS 4 All

Il database di immagini scelto è chiamato "THINGS 4 All" ed è uno dei migliaia di dataset per il training di AI. La scelta di un dataset di questo tipo è stata totalmente intenzionale in quanto queste raccolte sono strettamente categorizzate e gli elementi meticolosamente etichettati per via della loro destinazione d'uso molto specifica.
Questo dataset in particolare ha catturato la nostra attenzione perché oltre ad essere enorme assicurando scalabilità, contiene immagini abbastanza definite da non perdere troppa qualità quando viene composta l'immagine, e sono in rapporto di forma abbastanza simili da non causare eccessive distorsioni nell'immagine composta.
La particolare variante è l'archivio 'THINGS.zip' (diviso in 5 parti nel repository sorgente), ri-arrangiato manualmente per adattarlo al caso d'uso specifico. Ad esempio, l'eliminazione di alcune categorie ambigue e la fusione di alcune categorie troppo simili tra loro.

[6] Come possiamo vedere da Can I use: HTML5 e https://caniuse.com/?search=html5, la maggior parte (più del 95%) dei dispositivi che si connettono al Web supportano le funzionalità principali di queste tecnologie rendendo l'accessibilità un problema di entità trascurabile.