Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea in Ingegneria Informatica

Progettazione e sviluppo di una piattaforma riutilizzabile in contesto aziendale

Relatore:
Prof. Francesco Guerra

Presentata da:
Matteo Sirri

Anno Accademico 2020-2021

Abstract

Indice

Capitolo 1

Introduzione

1.1 Obiettivo

testo

1.2 Campo di applicazione

testo

1.3 Panoramica

testo

Capitolo 2

Descrizione generale

2.1 Inquadramento

testo

2.2 Macrofunzionalità del sistema

testo

2.3 Caratteristiche degli utenti

testo

2.4 Vincoli generali

testo

2.5 Analisi future

testo

Capitolo 3

Tecnologie

3.1 Implementazione

In questa sezione verranno descritti gli strumenti utilizzati per implementare i componenti che permettono alla piattaforma di erogare i propri servizi. La motivazione principale che ha portato alla scelta delle tecnologie di seguito elencate è il *know-how* aziendale.

3.1.1 Linguaggio

Typescript

Typescript[1] è un linguaggio open-source sviluppato da Microsoft.

È un *super-set* del linguaggio JavaScript, permettendone l'estensione con l'introduzione di un meccanismo di tipizzazione statico e il supporto alla programmazione orientata agli oggetti. Per via della sua natura può essere utilizzato in tutti i contesti in cui viene usato JavaScript grazie ad un processo di transpilazione che traduce codice Typescript in codice JavaScript, permettendone così una successiva compilazione ed esecuzione.

3.1.2 Ambiente di sviluppo

Node.js

Node.js[2] è un ambiente runtime JavaScript open-source e multipiattaforma.

Le caratteristiche fondamentali sono: l'esecuzione dell'engine V8, sviluppato da Google, che permette di compilare ed eseguire codice JavaScript al di fuori di browser web, l'uso di un insieme di primitive I/O asincrone di tipo non bloccante e l'esecuzione di applicazioni su un solo processo, senza generazione di nuovi thread per ogni richiesta. Questo sta a significare che quando si deve eseguire una operazione I/O, come una richiesta ad un web server, Node.js non blocca il thread, mettendo in attesa la CPU, ma, al contrario, la lascia libera di portare avanti altri compiti e si occuperà di ripristinare l'operazione non appena arriverà una risposta utilizzando una *callback*

Grazie a queste peculiarità è possibile realizzare applicazioni performanti in grado di gestire connessioni concorrenti con un singolo server.

In questo ambiente è poi possibile utilizzare lo standard ECMAScript in modo flessibile in quanto è possibile modificare il set di funzionalità abilitate, potendo così adattarsi al meglio nei vari contesti di utilizzo.

Infine, Node.js permette anche di aumentare la produttività di un team di sviluppo perchè fornisce agli sviluppatori *front-end*, che conoscono il linguaggio JavaScript, la possibilità di sviluppare codice *server-side*; senza dover imparare un linguaggio del tutto nuovo. Grazie alle sue caratteristiche Node.js risulta essere un ottimo strumento per lo sviluppo di servizi web.

3.1.3 Node Package Manager

Node Package Manager[3] (NPM) è un *software registry* per applicazione Node.Js. Questo si compone di due parti principali: il registro e una *Command Line Interface*(CLI).

Il primo è una raccolta di librerie open-source che permettono di integrare in una applicazione numerose funzionalità e che può favorire la condivisione di codice, anche con l'uso di registri privati.

Il secondo permette di interagire con il registro e gestire le dipendenze del progetto. In particolare, il meccanismo di gestione delle dipendenze di NPM permette di gestire con semplicità i pacchetti sul quale dipendende una applicazione grazie all'utilizzo di un file particolare chiamato *package.json*. Al suo interno sono infatti raccolte tutte le informazioni relative alla applicazione, gli script eseguibili e l'elenco delle dipendenze. Questo risulta essere di fondamentale importanza perchè permette ad un team di sviluppo di avere un meccanismo che garantisce consistenza tra i vari ambienti usati dagli sviluppatori.

3.1.4 Framework

NestJS

NestJS [4] (Nest) è un framework basato su Node.js per realizzare delle web *Application programming interface* (API) e microservizi. Offre supporto sia il JavaScript che il TypeScript e combina elementi di programmazione ad oggetti e programmazione funzionale.

Nel dettaglio questo framework si pone come un layer di astrazione tra lo sviluppatore e un server HTTP basato su Express.js o Fastify (due framework per realizzare server web veloci e flessibili). Grazie a questo è inoltre possibile usufruire tutti i componenti aggiuntivi compatibili con la piattaforma sottostante, con ovvi vantaggi in termini di riusabilità e flessiblità.

Altro aspetto significativo di questo framework è che guida lo sviluppatore a realizzare una applicazione con una architettura *three-tier* ("a tre strati"), ovvero suddividendo gli elementi principali in 3 strati dedicati alla gestione delle richieste dell'utente, alla gestione della logica funzionale e alla gestione dei dati.

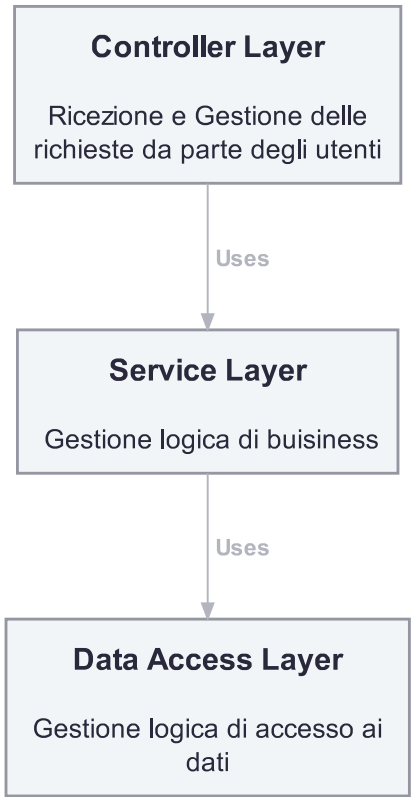

Figura 3.1: Schema riassuntivo della architettura *three-tier*

Questa architettura viene supportata grazie a dei componenti di base, offerti dal framework stesso, che possono essere estesi dallo sviluppatore in base alle proprie esigenze. Importante caratteristica di tutti questi componenti è che fanno largo uso della *Depenceny Injection*[1].

Pertanto l'utilizzo di Nest offre agli sviluppatori utili strumenti per velocizzare lo sviluppo di una web API prestando attenzione alle performance e alla architettura software del prodotto da realizzare risultando un'ottima scelta per la realizzazione si servizi web.

3.1.5 Testing

Jest

Jest[5] è un test runner JavaScript, sviluppato da Facebook, che fornisce un tool di strumenti per testare una applicazione basata su Node.js.

Permette di implementare con facilità unit test e integration test, con la possibilità di sfruttare i *mock objects* ("oggetti simulati"). Fornisce inoltre strumenti statistici per analizzare la *code coverage* ("copertura del codice").

[1]La *Dependency Injection* è un meccanismo che permette di applicare l'inversione del controllo ad un compoente software. In generale permette ad una classe di non dover configurare le proprie dipendenze in modo statico perchè vengono configurate dall'esterno. Ciò offre grossi vantaggi in termini di riusabilità e rende la fase di test molto più semplice.

3.2 Gestione dati

In questa sezione verranno descritti gli strumenti utilizzati per la gestione e l'aggiornamento delle informazioni necessare per il funzionamento della piattaforma. La motivazione principale che ha portato alla scelta delle tecnologie di seguito elencate è il *know-how* aziendale.

3.2.1 Database

MongoDB

MongoDB[6] è un database management system (DBMS) che offre la possibilità di gestire database NoSQL [2] basati sull'utilizzo di documenti flessibili per la gestione di dati in vario formato.[7]

L'unità fondamentale in un database basato su MongoDB sono i documenti. Questi rappresentano l'informazioni da memorizzare. Sono formattati in formato *Binary JSON* (BSON) e possono includere varie tipologie di dati che vanno dai comuni valori numerici o stringhe sino ai più complessi oggetti innidati e liste. La particolarità di questi documenti è che non hanno uno schema rigido: possono subire delle modifiche nel tempo. Permettono quindi ai dati di adattarsi alla applicazione e non viceversa. Questo offre un notevole vantaggio per lo sviluppatore in quanto può strutturare i dati nella maniera più consona per facilitarne l'utilizzo. Questi documenti vengono poi raccolti in collezioni che vengono usate come archivi per documenti facenti parte di una stessa categoria di informazione. Tutti questi dati possono poi essere ovviamente sottoposti a query e ad aggregazioni.

MongoDB risulta essere anche molto performante grazie alla possibilità di immagazzinare documenti innestati, riducendo l'attività I/O del sistema DB, e all'utilizzo del meccanismo di indicizzazione che permette di effettuare query molto velocemente.

Altre caratteristiche fondamentali di questa tecnologia sono l'alta disponibilità e la possiblità di scalare orizzontalmente il sistema. La prima è ottenuta con l'uso dei *replica set*, che permettono di gestire database replicati, in modo da garantire ridondanza. La seconda è invece realizzabile con il metodo dello *Sharding* distribuito che permette la distribuzione del carico computazionale, per la gestione delle richieste, su più server; fornendo così una esecuzione più efficente delle operazioni rispetto all'utilizzo di una solo server.

[2] dall'inglese *Not only SQL*; non solo SQL.

3.3 Servizi cloud esterni

In questa sezione verranno descritti i servizi cloud esterni integrati dalla piattaforma. La loro integrazione ha permesso alla piattaforma di erogare servizi complessi, riducendo spese e costi di gestione e sviluppo e aumentando la produttività del team.

3.3.1 Amazon Simple Email Service

Amazon Simple Email Service [8] (SES) è un servizio e-mail scalabile e conveniente che offre ad uno sviluppatore un'interfaccia semplice per inviare e-mail da qualisasi applicazione. Offre la possibilità di inviare e-mail transazionali, di business o in massa permettendo così di potersi adattare a vari contesti di utilizzo.

Questo servizio fornisce anche un pannello di controllo con il quale è possibile effettuare il monitoraggio e l'analisi dei problemi, che potrebbero diminuire l'efficacia del recapito delle e-mail, e delle statistiche relative all'invio delle comunicazioni, con le quali si può misurare il grado di coinvolgimento dei clienti Permette inoltre di gestire il piano di utilizzo con il quale è possibile ottimizzare le spese di gestione.

L'integrazione di questo servizio ha permesso di gestire l'invio di e-mail in modo flessibile, performante ed economicamente conveniente senza dover aggiungere complessità alla piattaforma.

3.4 Sicurezza

In questa sezione verranno descritti i meccanismi di autenticazione e autorizzazione scelti per garantire un adeguato livello di sicurezza agli utenti.

3.4.1 Autenticazione

L'autenticazione è un metodo che permette di identificare gli utenti sulla base di un set di credenziali e verificare che questi forniscano le giuste informazioni per accedere alla applicazione.

L'implementazione presente all'interno della piattaforma si basa su una coppia di credenziali: email e password. Per garantire una archiviazione sicura delle password è stato deciso di applicare una policy basata sull'uso di *hash* e *salt*.

- spiegazione di hash - - spiegazione di salt - - che problema risolve la loro introduzione -

3.4.2 Autorizzazione

L'autorizzazione è un metodo che permette di garantire la riservatezza e la disponibilità dei dati. Attraverso l'implementazione di policy è infatti possibile definire dei protocolli che permettono di restringere l'accesso alle risorse, rendendole disponibili solo agli utenti autorizzati.

Le policy definite all'interno della piattaforma si basano sul protocollo di autorizzazione Oauth 2.0 [9] e sull'utilizzo di un modello di controllo sugli accessi basato sui ruoli dei singoli utenti.

OAuth2.0

- definizione oauth2.0 - - descrizione categoria utilizzata da noi con riferimento alle altre come nota - - che problema vuole risolvere -

JWT

- definizione jwt - - struttura - - cifratura -

3.5 Gestione codice condiviso

In questa sezione verranno presentate le tecnologie utilizzate per supportare le operazioni del team di sviluppo ed aumentarne la produttività. La motivazione principale che ha portato alla scelta delle tecnologie di seguito elencate è il *know-how* aziendale.

3.5.1 Git

Git [10] è un sistema di controllo versione distribuito (DVCS), gratuito ed open-source. Nato nel 2005 grazie all'operato della comunità Linux e di Linus Torvalds, il creatore di Linux, è ad oggi uno strumento che permette ad un team di sviluppo di coopeare in modo facile, veloce ed efficente su grandi progetti.

Essendo un sistema di controllo versione permette di registrare, nel tempo, i cambiamenti ad un file o ad una serie di file, così da poter richiamare una specifica versione in un secondo momento. [11]. Ciò offre numerosi vantaggi tra cui quello di poter ripristinare un file o un intero progetto ad uno stato precedente ad una modifica, vedere chi e quando ha cambiato qualcosa e recuperare file cancellati. Inoltre offre un sistema di ramificazione (branching) per lo sviluppo non lineare.

È stato utilizzato nel contesto dello sviluppo della applicazione attraverso GitLab, una piattaforma web open source che consente la gestione di repository Git.

3.5.2 Monorepo

Una *monorepo*[12] (*mono repository*) è una strategia di sviluppo software dove il codice relativo a vari progetti o applicazioni risiede in uno stesso luogo, solitamente una *repository* condivisa gestita con un sistema di controllo versione. Si contrappone al modello di singola *repository* per progetto.

Di seguito alcuni vantaggi offerti dall'utilizzo della strategia monorepo.[13]

- Gestione semplificata delle dipendenze: le singole applicazioni condividono le stesse dipedenze. Questo stimola la coesione negli sturmenti e librerie utilizzati e aumenta la produttività degli sviluppatori.
- Organizzazione semplificata: è possibile progettare una gerarchia logica fra i progetti per renderli logicamente relazionati anche all'interno della repository.
- Semplificazione processi di release: esiste una pipeline condivisa per la build e il deploy del sistema.
- Codivisione codice: è possibile utilizzare delle librerie interne per condividere codice fra le varie applicazioni riducendo così la duplicazione e favorendo la creazione di codice migliore.
- Standard e convenzioni: è possibile introdurre un utilizzo coeso relativo alle convenzioni di sviluppo e all'utilizzo delle tecnologie usate per testing, debug e build dei progetti. Queste permette di creare codice consistente e di qualità.

L'utilizzo di una monorepo permette quindi di creare piattaforme complesse, composte da applicazioni e microservizi, in modo coeso, dal punto di vista degli strumenti e tecnologie usate, e permettendo agli sviluppatori di avere le conoscenze adatte per lavorare in modo efficente ed efficace ai vari progetti della organizzazione. Nonostante ciò l'utilizzo di questa strategia introduce un certo livello di rigidità nella gestione delle applicazioni e un grado di complessità, proporzionale alle dimensioni della monorepo, nell'introduzione di nuovi membri nel team di sviluppo.

Nello sviluppo della piattaforma è stato deciso di utilizzare questa strategia e ciò ha permesso al team di sviluppo di operare in modo efficente e coeso.

Nx

Nx [14] è uno strumento open source, creato dal team Nrwl [15], che permette ad una organizzazione di gestire una monorepo. Il suo funzionamento è basato sulla CLI di Angular, un framework per lo sviluppo di applicazioni front-end, ed è composto da un insieme di strumenti che permettono di gestire in modo flessibile una monorepo per progetti basati su Node.js, React o Angular. Alcune delle funzionalità offerte sono la generazione automatica della gerarchia di file e cartelle per generare una nuova applicazione o libreria, possibilità di eseguire script in un contesto globale o relegato alle singole applicazioni e meccanismi di ottimizzazione per le fasi di build e test grazie all'utilizzo dei un meccanismo di caching.

Nello sviluppo della piattaforma è stato decido di introdurre questo strumento in quanto permette di semplificare notevolmente la fase di design della monorepo stessa.

3.6 Integrazione e rilascio del software

In questa sezione sono descritte le tecnologie utilizzate nelle fasi di integrazione e rilascio del software.

3.6.1 Docker

Docker [16] è una tecnologia open source per creare, distribuire e gestire applicazioni sottoforma di *container*. Un container Docker è un'ambiente isolato in cui può essere messa in esecuzione una applicazione. Più nello specifico si crea un container basato su Linux in cui viene caricata un'immagine di una applicazione con le relative dipendenze.

Vantaggio offerto dalla piattaforma Docker è il fatto che questi container sono isolati li uni dagli altri e ciò permette di poterli eseguire in sicurezza e indipendentemente. Altra nota positiva è la leggerezza. Infatti i container Docker, a differenza delle macchine virtuali, non necessitano di un hypervisor ma vengono eseguiti direttamente dal kernel della macchina host.

L'utilizzo dei container permette poi di disaccopiare le fasi di rilascio e deployment delle applicazioni ed offre la garanzia di avere un'ambiente in cui si ha la certezza che il comportamento della applicazione sarà quello previsto. Questo risulta vantaggioso anche per la produttività degli sviluppatori che possono così condividere il proprio lavoro più facilmente.

Questa tecnologia è stata usata nello sviluppo della piattaforma per supportare le operazioni di integrazione e distribuzione. Viene inoltre utilizzato per effettuare il deployment, ovvero il rilascio dell'applicazione al reparto di produzione.

3.6.2 Jenkins

Jenkis [17] è uno strumento open source che fa parte della categoria degli *automation server*, ovvero applicazioni server utilizzate per automatizzare le task relative alle fasi di build, test, rilascio e deploy del software.

Basa il suo funzionamento su una pipeline, ovvero un insieme di step, detti *stage*, da eseguire in sequenza per portare a termine un processo. Ogni step potrà contenere a sua volte delle istruzioni, detti *jobs*, da eseguire per portare a termine un dato compito quale potrebbe essere la build, i test o il deployment. L'esecuzione della pipeline può avvenire in varie modalità pianificata o utilizzando dei trigger, come ad esempio il commit su una repository o il fallimento di un test.

3.7 Deployment

3.7.1 AWS

testo

Capitolo 4

Architettura

4.1 Descrizione generale

4.2 API Server

4.2.1 Descrizione generale

testo

4.2.2 Principi di design

testo

4.2.3 Auth module

testo

4.2.4 Demo module

testo

4.2.5 User module

testo

4.2.6 Mail module

testo

4.3 Mailer microservice

4.3.1 Descrizione generale

testo

4.3.2 Principi di design

testo

4.3.3 Template Service

testo

4.3.4 Transport Service

testo

4.4 Database Server

4.4.1 Descrizione generale

testo

4.4.2 Modellazione dati

testo

4.5 MQTT Server

4.5.1 Descrizione generale

testo

4.5.2 Principi di design

testo

Capitolo 5

Distribuzione

5.1 Descrizione generale

testo

5.2 CI/CD pipeline

5.2.1 Descrizione generale

testo

5.2.2 Motivazioni

testo

5.2.3 Integrazione continua

testo

5.2.4 Distribuzione continua

testo

5.2.5 Deployment continuo

Capitolo 6

Conclusioni

6.1 Valutazioni complessive

6.2 Sviluppi futuri

Fonti bibliografiche e sitografia

[1] Typescript. Sito web typescript. Visitato il 08/2021. [Online]. Available: https://www.typescriptlang.org/

[2] Node.js. Sito web node.js. Visitato il 08/2021. [Online]. Available: https://nodejs.org/en/about/

[3] NPM. Documentazione npm. Visitato il 08/2021. [Online]. Available: https://docs.npmjs.com

[4] NestJS. Documentazione nestjs. Visitato il 08/2021. [Online]. Available: https://docs.nestjs.com/

[5] Jest. Sito web jest. Visitato il 08/2021. [Online]. Available: https://jestjs.io/

[6] MongoDB. Sito web mongodb. Visitato il 08/2021. [Online]. Available: https://www.mongodb.com/

[7] IBM. What is mongodb. Visitato il 08/2021. [Online]. Available: https://www.ibm.com/cloud/learn/mongodb

[8] AWS. Sito web aws-ses. Visitato il 08/2021. [Online]. Available: https://aws.amazon.com/it/ses/

[9] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, Oct. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6749.html

[10] Git. Sito web git. Visitato il 08/2021. [Online]. Available: https://git-scm.com/

[11] S. Chacon and B. Straub, *Pro git: Everything you need to know about Git*, 2nd ed. Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2

[12] C. Jaspan, M. Jorde, A. Knight, C. Sadowski, E. K. Smith, C. Winter, and E. Murphy-Hill, "Advantages and disadvantages of a monolithic repository: A case study at google," *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2017.

[13] G. Brito, R. Terra, and M. T. Valente, "Monorepos: A multivocal literature review," *ArXiv*, vol. abs/1810.09477, 2018.

[14] Nrwl. Sito web nx. Visitato il 08/2021. [Online]. Available: https://nx.dev/

[15] ——. Sito web nrwl. Visitato il 08/2021. [Online]. Available: https://nrwl.io/

[16] Docker. Sito web docker. Visitato il 08/2021. [Online]. Available: https://www.docker.com/

[17] Jenkins. Sito web jenkins. Visitato il 08/2021. [Online]. Available: https://www.jenkins.io

Elenco delle figure