

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA IN INFORMATICA



**CS-Template: una applicazione per la
piattaforma Zendesk basata su moderne
tecnologie web**

Tesi di laurea triennale

Relatore

Prof. Francesco Ranzato

Laureando

Singh Parwinder
Matricola 1073318

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage (dal 03/06/2018 al 03/08/2018), della durata di circa 320 ore, dal laureando Singh Parwinder presso l'azienda Nextep alla sede di Cittadella.

In questo documento verranno descritte in dettaglio l'analisi dei requisiti, la progettazione, l'implementazione e la validazione dell'applicazione CS-template. L'applicazione in questione è stata realizzata utilizzando le tecnologie web innovative sia per quanto riguarda il lato frontend dell'applicazione che quello backend.

L'intero lavoro è stato svolto in ambiente Linux Ubuntu 18.04 LTS. Tutti i diagrammi delle classi, dei package e dei casi d'uso (presenti nei Capitoli 3 e 4) sono conformi allo standard UML 2.0. Per realizzarli è stato usato il software Astah Professional¹.

Il primo capitolo descrive l'azienda e il progetto di stage assegnato.

Il secondo capitolo descrive le tecnologie utilizzate durante tutto il periodo di stage.

Il terzo capitolo formalizza i casi d'uso e i requisiti ad alto livello raccolti in fase di analisi dei requisiti.

Il quarto capitolo illustra ad alto livello la progettazione.

Il quinto capitolo illustra il progetto realizzato.

Il sesto capitolo conclusione e risultati ottenuti.

Nel ultimo capitolo viene riportato il glossario ed alcuni concetti secondo me molto interessanti.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- * gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- * per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: **API**;
- * i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

¹<http://astah.net/editions/professional>.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Francesco Ranzato, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, settembre 2018

Singh Parwinder

Indice

1	Introduzione	1
1.1	Dominio aziendale	1
1.1.1	L'azienda ospitante	1
1.1.2	Prodotti e servizi	1
1.2	Lo stage	2
2	Tecnologie utilizzate	3
2.1	Frontend	3
2.1.1	Angular 6	3
2.1.2	Angular material	4
2.1.3	Typescript	4
2.1.4	HTML 5	5
2.1.5	SASS	5
2.2	Tecnologie Backend	6
2.2.1	Amazon Web Services	6
2.2.2	AWS API Gateway	6
2.2.3	AWS Lambda	7
2.2.4	AWS DynomoDB	7
2.2.5	AWS S3	7
2.2.6	AWS Cognito	8
2.3	Tecnologie di supporto	9
2.3.1	Git	9
2.3.2	GitLab	9
2.3.3	Atom	10
2.4	Figma	10
2.5	Zendesk Apps framework(ZAF)	10
2.5.1	ZAF Client SDK	11
3	Analisi dei requisiti	13
3.1	Casi d'uso	13
3.1.1	Casi d'uso pagina degli amministratori	13
3.1.2	Casi d'uso pagina contenente l'editor drag-and-drop	15
3.1.3	Casi d'uso pagina contenente il widget	16
3.2	Tracciamento dei requisiti	17
4	Progettazione	19
4.1	Progettazione Frontend	19
4.1.1	Pagina cotenente l'editor	19

4.1.2	Pagina contenente il widget	20
4.1.3	Pagina login	20
4.1.4	Pagina degli amministratori	21
4.1.5	Struttura applicazione	22
4.1.6	PluginPages	23
4.1.7	AdminPages	24
4.1.8	Atomic design	24
4.2	Progettazione Backend	27
4.2.1	Architettura a microservizi serverless	27
4.2.2	Architettura pagina degli amministratori	27
5	Prodotto realizzato	29
5.1	Editor	29
5.2	Widget	30
5.3	Pagina di login	31
5.4	Pagina degli amministratori	31
6	Conclusioni	33
6.1	Raggiungimento degli obiettivi	33
6.2	Valutazione personale sullo stage	34
A	Appendice A	35
A.1	Servizi di Angular 6	35
A.2	RxJS in Angular 6	36
	Glossario	37
	Bibliografia	41

Elenco delle figure

1.1	Logo di Nextep: immagine tratta dal sito dell'azienda	2
2.1	Logo di Angular	3
2.2	Logo di Material UI	4
2.3	Typescript rispetto ES6 e ES5	4
2.4	Logo di HTML 5	5
2.5	Logo di SASS	5
2.6	AWS cloud computing	6
2.7	Logo API Gateway	6
2.8	Logo di AWS Lambda	7
2.9	Logo di AWS DynamoDB	7
2.10	Logo di AWS S3	8
2.11	Logo di AWS Cognito	8
2.12	Logo di Git	9
2.13	Logo di Gitlab	9
2.14	Logo di Atom	10
2.15	Logo di Figma	10
3.1	UC 1 - pagina login	14
3.2	UC 2 - funzionalità pagina admin	15
3.3	UC3 - pagina contenene l'editor	16
3.4	UC3 - pagina contenente il widget	17
4.1	Mock pagina contenente l'editor	19
4.2	Mock pagina di login	20
4.3	Mock pagina degli amministratori	21
4.4	Struttura applicazione	22
4.5	Struttura Routes	22
4.6	Struttura PluginPages	23
4.7	Struttura AdminPages	24
4.8	Elementi atomic design	25
4.9	Componenti Angular Material che formano gli atomi e le molecole dell'applicazione	25
4.10	Flusso completo dell'architettura	27
4.11	Flusso completo dell'architettura pagina admin	28
5.1	Editor realizzato	29
5.2	Widget realizzato	30
5.3	Form login	31

5.4	Pagina degli amministratori	31
A.1	Injector di Angular	35
A.2	DI in Angular	36
A.3	Logo di Angular e RxJS	36

Elenco delle tabelle

3.1	Tabella del tracciamento dei requisiti funzionali	18
3.2	Tabella del tracciamento dei requisiti di vincolo	18
6.1	Soddisfacimento Requisiti	33

Capitolo 1

Introduzione

In questo capitolo viene brevemente descritta l'azienda ospitante in cui è stata svolta l'attività di stage e una descrizione ad alto livello del progetto realizzato.

1.1 Dominio aziendale

1.1.1 L'azienda ospitante

Nextep è una società fondata nel 2000 da Marco De Toni e Mirco Soffia, con sede attuale a Cittadella (PD). Opera nel settore informatico e si occupa di servizi web, web marketing e di infrastrutture per gestire le informazioni delle aziende, e più in generale ha come obiettivo quello di migliorare l'efficacia delle strategie di comunicazione web, delle aziende, dedicando particolare attenzione alla reputazione e all'identità digitale.

Nextep fa parte del gruppo Allos, insieme ad Allos Italia¹, Allos Sud Africa², Allos USA³ e Zero12⁴. Allos si occupa di progetti e tecnologie per lo sviluppo del capitale umano, mentre Zero12 si occupa dello sviluppo di soluzioni mobile e cloud based. Il gruppo Allos è stato recentemente acquisito da EOH Holdings Ltd⁵, una grande società sudafricana.

Nextep ha un organico di circa venti persone, tra dipendenti e collaboratori, con varie competenze: grafici, sviluppatori, esperti di web marketing e tecnici. Sono presenti tre gruppi principali di lavoro: quello di sviluppo, creativo e del supporto tecnico. In Nextep c'è un ambiente di lavoro giovane, dinamico ma allo stesso tempo professionale, ed è incentivata la collaborazione e la condivisione di conoscenze e idee tra le persone. Tutto questo favorisce sia la crescita individuale, dal punto di vista professionale, che la crescita e l'amalgamazione dei vari gruppi di lavoro.

1.1.2 Prodotti e servizi

Nextep lavora per clienti di diversa tipologia e conformazione, dalla piccola impresa privata alla multinazionale che si sta espandendo ulteriormente, e con questo offre

¹<https://www.allos.it/>.

²<http://www.allos.co.za/>.

³<http://www.allosamerica.com/>.

⁴<http://www.zero12.it/>.

⁵<http://www.eoh.co.za/>.



Figura 1.1: Logo di Nextep: immagine tratta dal sito dell'azienda

svariati prodotti e servizi in base alle esigenze e alle opportunità del mercato e proprie.

La maggior parte dei progetti riguarda la realizzazione di portali e siti web, ma vengono sviluppati anche diversi altri prodotti, tra cui soluzioni e-commerce e applicazioni mobile, sviluppo di progetti di virtualizzazione, e storage networking. Inoltre negli ultimi mesi l'azienda si sta dedicato molto anche ai prodotti di *machine learning*, come i chatbot.

Nextep offre diversi tipi di servizi tra questi l'installazione e assistenza del portale di *customer service* Zendesk. Guida le diverse società (piccole o grandi) verso la gestione dei proprio cliente in maniera semplice ed efficace.

1.2 Lo stage

Il progetto di stage è consistito principalmente nella realizzazione di una applicazione per la piattaforma di *customer service* Zendesk. La piattaforma Zendesk permette a un'azienda di gestire tutte le richieste (chiamate *tickets*) di propri clienti in unico posto. L'applicazione realizzata permette agli agenti (persone che gestiscono le richieste dei clienti) e agli amministratori di Zendesk di realizzare contenuti (chiamati *template*) HTML e CSS in maniera molto semplice e veloce, ovvero utilizzando un editor *drag-and-drop*. I template successivamente sono utilizzati nelle risposte verso i clienti. Questo permette di risparmiare una notevole quantità di tempo e non è necessario avere le conoscenze di HTML e CSS. Diverse aziende (clienti di Nextep) hanno fatta la richiesta esplicitamente di tale applicazione. .

Dopo una breve analisi insieme al tutor aziendale è stata definita la seguente struttura dell'applicazione ad alto livello:

- * L'applicazione deve essere realizzata utilizzando Angular;
- * Oltre all'applicazione che dovrà essere integrata su Zendesk, è richiesta la realizzazione (sempre in Angular) anche di una pagina di *admin* per la gestione di tutti i clienti che utilizzeranno tale applicazione;
- * La pagina di *admin* deve essere accessibile solo dopo aver effettuato il *login*;
- * Il backend dell'applicazione deve essere tutto realizzato nei sistemi Cloud (AWS, Azure ecc.).

Capitolo 2

Tecnologie utilizzate

In questo capitolo seguirà un elenco delle tecnologie di riferimento adottate per la realizzazione dell'applicazione CS-Template.

2.1 Frontend

2.1.1 Angular 6

Angular è una [framework](#) opensource realizzato da Google nel 2016 che permette di creare le [SPA](#), sfruttando i [design pattern](#) architetturali [MVC](#) e [MVVM](#).

Le applicazioni sviluppate in Angular vengono eseguite interamente dal *web browser* dopo essere state scaricate dal *web server*. Questo comporta il risparmio di dover spedire indietro la pagina web al *web-server* ogni volta che c'è una richiesta di azione da parte dell'utente. Il codice generato da Angular gira su tutti i principali *web browser* moderni.

Ogni pagina viene costruita da diversi componenti. Un componente in Angular in generale è una piccola parte della pagina che rappresenta una specifica funzionalità (esempio la navbar). Ogni componente ha una propria logica strutturale (scritta tramite appositi marcatori HTML), di presentazione (scritta con appositi fogli di stile CSS oppure SASS) e di *business* (scritta con il linguaggio di programmazione TypeScript). Tutti i componenti possono comunicare tra di loro scambiandosi oggetti, lo scambio viene fatto utilizzando diversi strumenti messi a disposizione da Angular. Oggi tale *framework* è alla versione 6.



Figura 2.1: Logo di Angular

2.1.2 Angular material

Angular material è una libreria che contiene una raccolta di componenti di Material Design¹. Questa libreria è stata sviluppata sempre da Google e permette di realizzare delle UI molto avanzate in maniera molto semplice. Fornendo una serie di semplici componenti (come i pulsanti, inputbox ecc) di Angular già fatta permette agli sviluppatori di risparmiare una notevole quantità di tempo. Tutti i componenti sono testati da Google garantendo così un corretto funzionamento.



Figura 2.2: Logo di Material UI

2.1.3 Typescript

TypeScript è un linguaggio di programmazione libero ed Open source (libero) sviluppato da Microsoft basato su ECMAScript 6. Esso estende la sintassi di Javascript aggiungendo il concetto di tipizzazione (interfacce, classi, enum ecc). Questo lo rende molto simile ai linguaggi di programmazione come Java oppure C++, e diventa anche molto semplice l'applicazione di molti *design pattern* conosciuti.

TypeScript nasce dal crescente bisogno di un linguaggio *frontend* per lo sviluppo di applicazioni JavaScript a larga scala. Il linguaggio è nato dalla necessità di sicurezza e robustezza, sia da sviluppatori interni a Microsoft sia clienti.

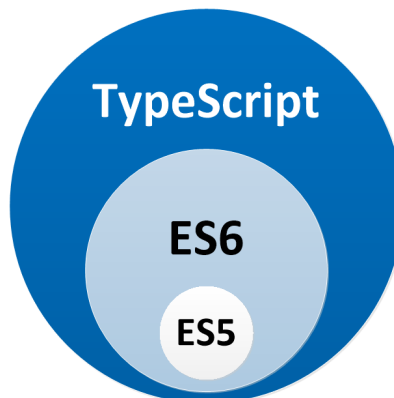


Figura 2.3: Typescript rispetto ES6 e ES5

¹<https://material.io/design/>.

2.1.4 HTML 5

L'HTML5 è un linguaggio di *markup* per la strutturazione delle pagine web, pubblicato come [W3C Recommendation](#) da ottobre 2014. HTML è stato usato come linguaggio per la definizione della logica strutturale del *frontend* dell'applicazione. Una delle principali vantaggi di Angular sta proprio nel utilizzo di HTML puro rispetto i *framework*/librerie revali come React oppure Vue.



Figura 2.4: Logo di HTML 5

2.1.5 SASS

Sass (Syntactically Awesome StyleSheets) è un'estensione del linguaggio CSS che permette di utilizzare variabili, di creare funzioni e di organizzare il foglio di stile in più file.

Il linguaggio Sass si basa sul concetto di preprocessore CSS, il quale serve a definire fogli di stile con una forma più semplice, completa e potente rispetto ai CSS e a generare file CSS ottimizzati, aggregando le strutture definite anche in modo complesso. SASS è un linguaggio utilizzato per definire la presentazione dell'applicazione. Poiché Angular Material fornisce molti componenti prefatti, questo linguaggio è utilizzato principalmente per definire il *layout* delle pagine.



Figura 2.5: Logo di SASS

2.2 Tecnologie Backend

Per quanto riguarda il lato *Backend* è stato deciso di utilizzare il *cloud computing*. Esso è la distribuzione di servizi di calcolo, come *server*, risorse di archiviazione, *database*, rete, *software*, analisi e molto altro, tramite *Internet* ("il cloud").

Le società che offrono questi servizi di calcolo sono dette *provider* di servizi *cloud* e in genere addebitano un costo per i servizi di *cloud computing* in base all'utilizzo (on demand). I provider più famosi oggi sono Amazon, Microsoft e Google.

Dopo una breve analisi e confronto tra i servizi offerti da questi provider è stato deciso di utilizzare i servizi offerti da Amazon, ovvero gli AWS (Amazon Web Services). La scelta è stata fatta soprattutto per la popolarità dei servizi Amazon: essendo Amazon il primo a fornire questo tipo di servizi, questi sono attualmente molto più utilizzati rispetto agli altri due. Per quanto riguarda i prezzi, quantità e qualità di servizi offerti da tutti i *provider* non è stato trovato una rilevante differenza.



Figura 2.6: AWS cloud computing

2.2.1 Amazon Web Services

Per l'applicazione in questione sono stati identificati specifici servizi di Amazon, che hanno permesso poi di realizzare un'architettura [serverless](#) facilmente scalabile. In seguito è fornita una descrizione generale dei servizi scelti.

2.2.2 AWS API Gateway

Amazon API Gateway è un servizio il cui scopo è quello di semplificare agli sviluppatori la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione delle [API](#) su qualsiasi scala. Questo servizio permette di creare un punto d'ingresso attraverso il quale le applicazioni posso accedere ai dati. Fornisce una API per ricevere le richieste dalle applicazioni, dopo ogni richiesta questo servizio genera un evento che esegue qualcosa (una funzione lambda oppure un'altra chiamata).



Figura 2.7: Logo API Gateway

2.2.3 AWS Lambda

Sono delle semplici funzioni che ricevono come *input* un evento e possono ritornare qualche valore. Possono essere scritte in molti linguaggi di programmazione, come Nodejs(javascript), Java, C++ e Python.

Queste funzioni possono comunicare con qualsiasi servizio di Amazon utilizzando [AWS-SDK](#). In questo progetto sono state utilizzate diverse funzioni di questo tipo, principalmente per scrivere o leggere i dati su/dal database.



Figura 2.8: Logo di AWS Lambda

2.2.4 AWS DynamoDB

Amazon DynamoDB è un database non relazionale che fornisce prestazioni affidabili su qualsiasi scala. Si tratta di un database *multi-master*, multiregione e completamente gestito che fornisce latenza costante di pochi millisecondi e sicurezza integrata, *backup* e ripristino e *cache* in memoria.

Tutti i dati degli utenti e i *template* sono stati salvati su questo database. L'accesso a tali avviene solo tramite le funzioni lambda.



Figura 2.9: Logo di AWS DynamoDB

2.2.5 AWS S3

Amazon Simple Storage Service è uno *storage* di oggetti creato per memorizzare e ripristinare qualsiasi volume di dati da qualunque origine: siti web, applicazioni per mobile o dati provenienti da diversi dispositivi. Può essere utilizzato per memorizzare file multimediali ed è ideale per acquisire dati quali foto, video o immagini di risoluzione elevata da dispositivi mobili, *backup* di dispositivi mobile o computer.

Nel contesto dello stage è stato utilizzato per garantire la persistenza di tutti i dati non strutturati dell'applicazione. Inoltre l'applicazione stessa è stata caricata su questo servizio.



Figura 2.10: Logo di AWS S3

2.2.6 AWS Cognito

Amazon Cognito permette di aggiungere strumenti di registrazione, accesso e controllo degli accessi alle *app Web* e per dispositivi mobili in modo rapido e semplice. Amazon Cognito permette di ricalibrare le risorse per milioni di utenti e supporta l'accesso con provider di identità *social* quali Facebook, Google e Amazon.

Nel contesto dello stage è stato utilizzato per garantire l'accesso alla pagina degli amministratori solo agli utenti con le credenziali valide.



Figura 2.11: Logo di AWS Cognito

2.3 Tecnologie di supporto

2.3.1 Git

Git è un software di controllo di versione distribuito, creato nel 2005 da Linus Torvalds (creatore di Linux). È usabile principalmente da interfaccia linea di comando, ed oggi è in assoluto uno dei software per il controllo di versione più utilizzati. Come tutti i software per il controllo di versione, si basa sul concetto di [repository](#), ovvero un ambiente in cui vengono immagazzinati i metadati che possono essere recuperati e aggiornati da chi può averne accesso (è possibile ripristinare versioni precedenti dei dati caricati nel repository, poichè essi non vengono sovrascritti con gli aggiornamenti).



Figura 2.12: Logo di Git

2.3.2 GitLab

GitLab è una piattaforma web open source(libera) che permette la gestione di repository Git. GitLab permette la creazione di repository pubblici o privati, in cui gli sviluppatori possono caricare il proprio codice e gestire le modifiche alle varie versioni in contemporanea al lavoro di più persone.

In GitLab è possibile lavorare parallelamente ad altre persone sullo stesso progetto senza generare conflitti, caricare il proprio lavoro nel repository remoto (operazione di *push*) e poter unire alla fine le modifiche di tutti in un unico progetto (operazione di *merge*). GitLab mette a disposizione diverse funzionalità a seconda del tipo di abbonamento e del prezzo pagato. È comunque possibile utilizzarlo gratuitamente, seppur con delle limitazioni. Nel contesto dello stage è stato utilizzato GitLab con l'account aziendale di Nextep, permettendo così di lavorare su una repository privata.



Figura 2.13: Logo di Gitlab

2.3.3 Atom

Atom è un *editor* di testo *open source* sviluppato nel 2014 da GitHub. Questo *editor* è stato scritto completamente utilizzando le tecnologie web (Html, CSS e Javascript). Una delle cose molto interessanti di questo editor è quello di avere Git già integrato, permettendo in questo modo di fare *commit* sin da subito.



Figura 2.14: Logo di Atom

2.4 Figma

Figma è una piattaforma online che permette di creare [mock](#) delle interfacce. Molto utile per definire la *view* delle pagine prima di iniziare a scrivere il codice, in questo modo si è in grado di capire sin da subito che tipo di interfaccia l'applicazione deve avere.



Figura 2.15: Logo di Figma

2.5 Zendesk Apps framework (ZAF)

ZAF è un semplice *framework* sviluppato da Zendesk Inc. Permette di integrare nella piattaforma Zendesk applicazione web realizzata con qualsiasi tecnologia web.

Questo framework genera dei package che rappresentano un app di Zendesk. Take package può essere installato sulla propria piattaforma Zendesk come una applicazione privata oppure caricata (gratuitamente o a pagamento) sul Market place di Zendesk rendendola così disponibile a tutti gli utenti nel mondo. Il package è formato da un file JSON (manifest.json) e una cartella assets. La cartella contiene le icone da visualizzare dopo l'installazione dell'applicazione sulla piattaforma, mentre il file manifest.json contiene tutte le informazioni riguardo l'applicazione. Il seguente codice mostra la struttura generale di un semplice file manifest.json.

```
{
  "name": "CS-Template",
  "author": {
    "name": "Singh Parwinder",
    "email": "mio@email.com",
  },
  "location": {
    "ticket_sidebar": "https://www.paginaweb.org"
  },
  "parameters": [
    {
      "name": "token",
      "type": "text",
      "required": true
    }
  ]
}
```

L'attributo "location" è quello più importante. Questo attributo specifica dove visualizzare l'applicazione sulla piattaforma dopo l'installazione. Si ha la possibilità di scegliere tra più di dieci posizioni differenti, e per ogni posizione specificata bisogna indicare l'indirizzo web della pagina da visualizzare. Quindi in parole povere queste posizioni sono dei semplici [iframe](#) che mostrano le pagine web presenti su un server remoto. Per questo motivo le applicazioni possono essere realizzate con qualsiasi tecnologia web.

L'attributo "parameters" indica i parametri da chiedere all'utente durante l'installazione dell'applicazione sulla piattaforma Zendesk. Nel contesto dell'applicazione è richiesto di inserire obbligatoriamente un token di 12 cifre, l'inserimento sbagliato di tale valore comporta inutilizzo dell'applicazione CS-Template.

2.5.1 ZAF Client SDK

Ovviamente non avrebbe senso semplicemente visualizzare le pagine web sulla piattaforma se queste non possono interagire con le funzionalità di Zendesk.

Per permettere all'applicazione web di interagire con la piattaforma Zendesk, è resa disponibile una libreria(ZAF Client SDK) da utilizzare nella propria applicazione web. Questa libreria fornisce un oggetto speciale(ZAFClient) con una cinquantina di metodi che permettono(una volta caricata la pagina nella piattaforma) di interagire con le diverse funzionalità di Zendesk.

Il seguente codice invoca il metodo "invoke" dell'istanza "client" dell'oggetto ZAF-Client per aggiungere il testo "Hello World!" nella risposta da inviare all'utente.

```
let client = ZAFClient.init();
client.invoke('ticket.comment.appendText', 'Hello world!').then
(function() {
  console.log('text has been appended');
});
```


Capitolo 3

Analisi dei requisiti

Il presente capitolo ha come scopo quello di fornire una descrizione completa e precisa di tutti i requisiti individuati e dei casi d'uso ad alto livello riguardanti il progetto CS-Template.

3.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo [UML](#) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Ogni caso d'uso è definito secondo la seguente struttura:

- * Nome: Il titolo del caso d'uso;
- * Attori: Indica gli attori principali e secondari del caso d'uso. In tutto il contesto dell'applicazione gli attori del sistema saranno così classificati:
 - Utente Zendesk: sono gli utenti della piattaforma Zendesk. Possono essere gli agenti (persone che gestiscono le richieste dei clienti) oppure gli amministratori di Zendesk;
 - Utente generico: utente qualsiasi che non ha ancora effettuato l'accesso alla pagina degli amministratori;
 - Nextep Admin: utente amministratore Nextep, che ha compito di gestire tutti le aziende che utilizzano l'applicazione CS-Template.
- * Descrizione: Riporta una breve descrizione del caso d'uso;
- * Precondizione: Specifica le condizioni che sono identificate come vere prima del verificarsi degli eventi del caso d'uso;
- * Postcondizione: Specifica le condizioni che sono identificate come vere dopo il verificarsi degli eventi del caso d'uso.

3.1.1 Casi d'uso pagina degli amministratori

Questa è la pagina web il cui scopo principale è quello di visualizzare la lista di tutti i clienti di Nextep che hanno l'applicazione CS-Template installata sulla propria

piattaforma Zendesk. Inoltre permette di aggiungerne dei nuovi. L'accesso a questa pagina è garantita solo agli utenti amministratori di Nextep con le credenziali valide.

UC1: Login pagina amministratori

Attori Principali: Utente generico.

Descrizione: Caso d'uso descrive login alla pagina degli amministratori. .

Precondizioni: L'utente non autenticato.

Postcondizioni: Il sistema riconosce l'utente amministratore.

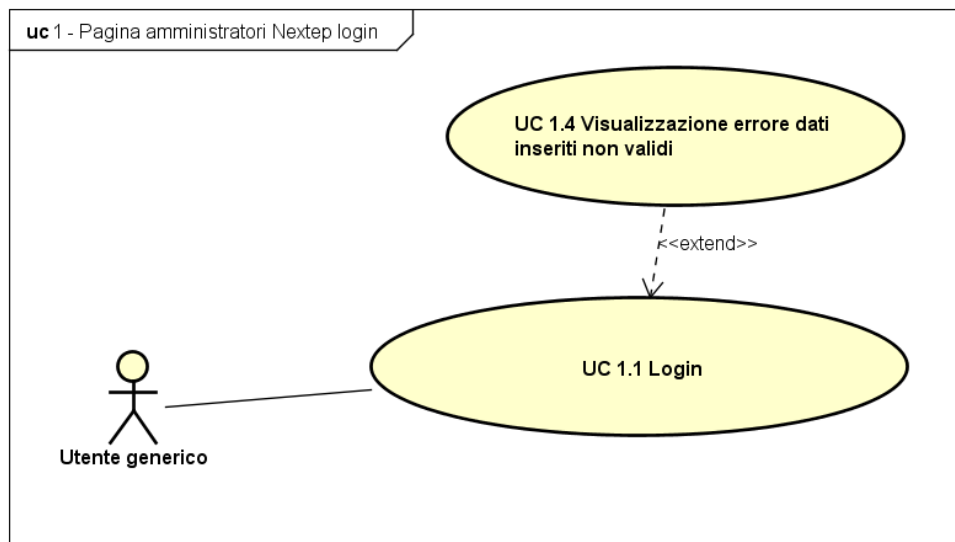


Figura 3.1: UC 1 - pagina login

UC2: Pagina degli amministratori

Attori Principali: Nextep Admin.

Descrizione: Caso d'uno descrive le funzionalità della pagina degli amministratori di Nextep. In questa pagina è possibile gestire tutti i clienti(aziende) di Nextep che utilizzano l'applicazione CS-Template.

Precondizioni: Il sistema riconosce l'amministratore.

Postcondizioni: L'amministratore visualizza una lista di tutti i clienti(utilizzatori di CS-Template) registrati nel sistema, visualizzando per ognuno di essi tutte le informazioni e un *form* per aggiungerne uno nuovo.

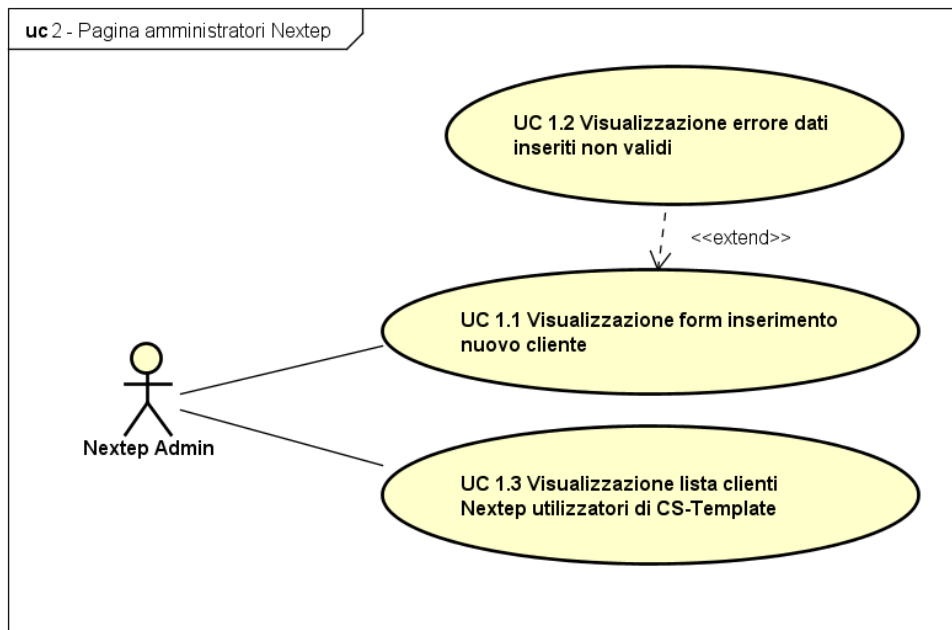


Figura 3.2: UC 2 - funzionalità pagina admin

3.1.2 Casi d'uso pagina contenente l'editor drag-and-drop

Questa pagina web contiene l'editor drag-and-drop che verrà visualizzato successivamente nella piattaforma Zendesk in un iframe. In seguito è riportato un caso d'uso generico che descrive tutte le proprietà ad alto livello dell'editor.

UC3: Editor drag-and-drop

Attori Principali: Utente Zendesk.

Descrizione: Caso d'uso descrive tutte le funzionalità ad alto livello che l'editor dovrà fornire.

Precondizioni: L'utente Zendesk apre l'editor.

Postcondizioni: L'editor permette all'utente Zendesk di realizzare qualsiasi tipo di contenuto HTML e CSS.

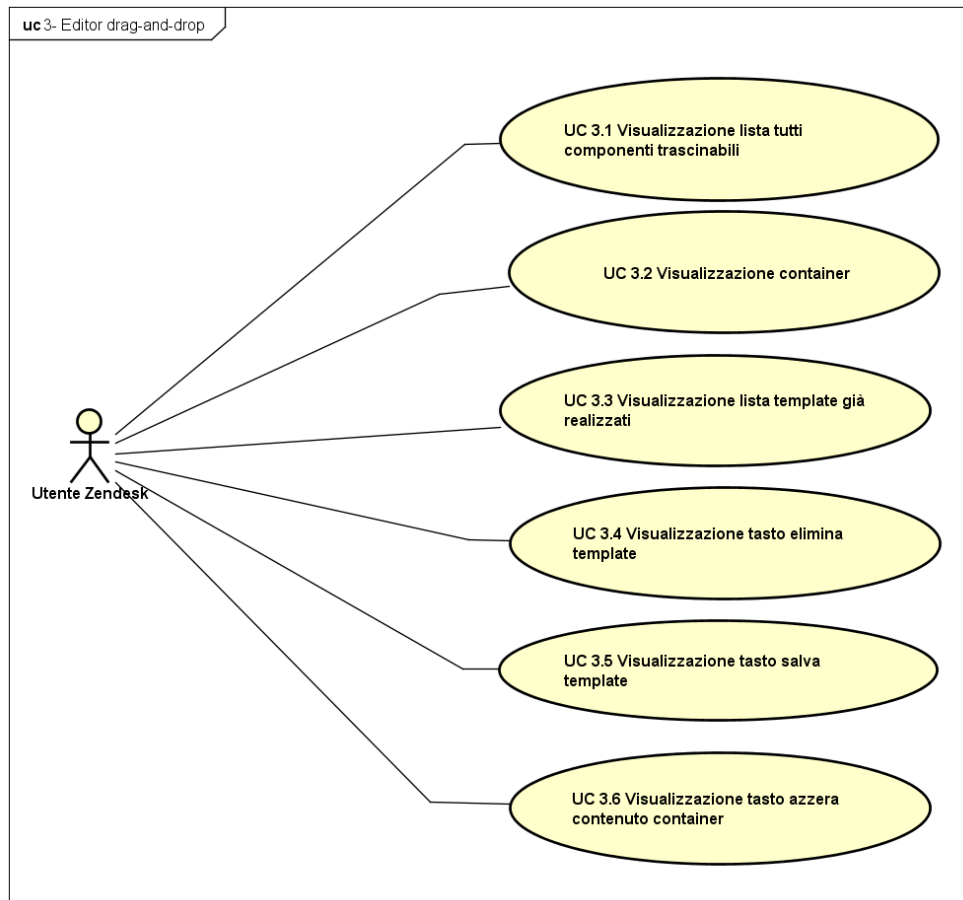


Figura 3.3: UC3 - pagina contenente l'editor

3.1.3 Casi d'uso pagina contenente il widget

Questa pagina web contiene il widget che verrà visualizzato successivamente nella piattaforma Zendesk in un iframe. Il widget verrà mostrato quando verrà aperta una richiesta qualsiasi del cliente. Esso permette di scegliere da una lista il contenuto HTML e CSS da utilizzare come risposta verso il cliente.

UC4: Widget dei template

Attori Principali: Utente Zendesk.

Descrizione: Caso d'uso descrive tutte le funzionalità ad alto livello che il widget dovrà fornire.

Precondizioni: L'utente Zendesk apre il widget.

Postcondizioni: Utente Zendesk sceglie il template da inviare come risposta al cliente.

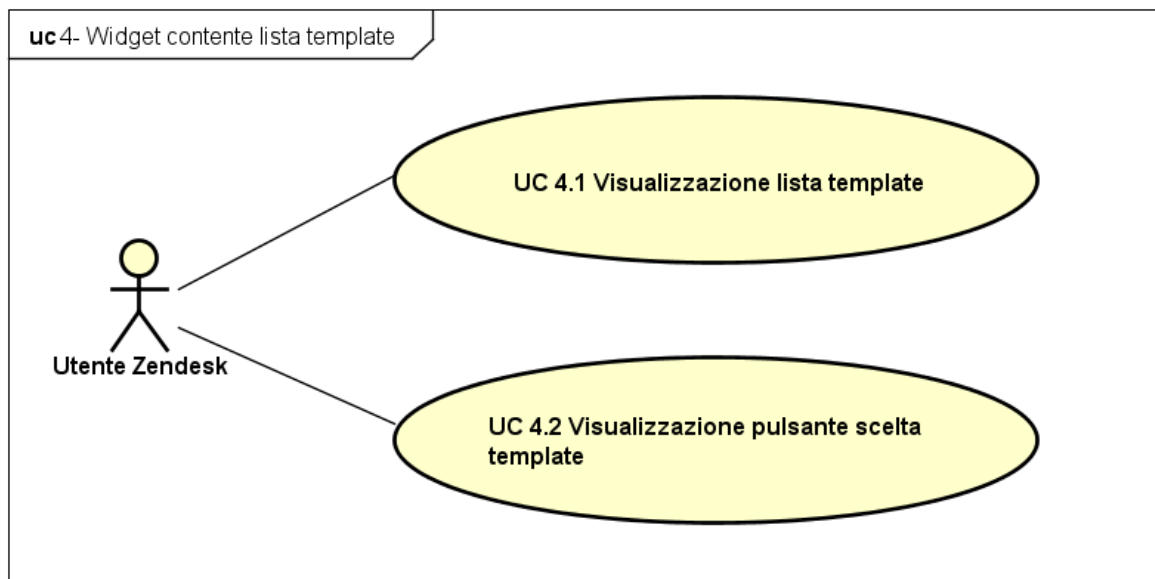


Figura 3.4: UC3 - pagina contenente il widget

3.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli *use case* effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli *use case*.

Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.

Il codice dei requisiti è così strutturato R(F/V)(N/D) dove:

R = requisito

F = funzionale

V = vincolo

N = obbligatorio (necessario)

D = desiderabile

Nelle tabelle 3.1 e 3.2 sono riassunti i requisiti e il loro tracciamento con gli *use case* delineati in fase di analisi.

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Use Case
RFN-1	Personale di Nextep può effettuare il login nella pagina degli amministratori	UC1
RFN-2	Nextep Admin può ottenere una lista con tutte le informazioni dei clienti che utilizzano l'applicazione CS-Template	UC2
RFN-3	Nextep Admin può eliminare un cliente dalla lista	UC2
RFN-4	Nextep Admin può aggiungere un nuovo cliente nella lista	UC2
RFN-4	Utente Zendesk può utilizzare l'editor drag-and-drop	UC3
RFN-5	Utente Zendesk può creare nuovi template	UC3
RFN-6	Utente Zendesk può salvare il template creato	UC3
RFN-7	Utente Zendesk può eliminare template creati	UC3
RFN-8	Utente Zendesk può azzerare il contenuto del editor	UC3
RFN-10	Utente Zendesk può impostare il nome del template	UC3
RFN-11	Utente Zendesk può utilizzare il template creato nella risposta verso il cliente	UC4
RFD-12	Utente Zendesk può esportare tutto i template creati in un file JSON come <i>backup</i>	UC4
RFD-13	Utente Zendesk può importare i template da un file JSON precedentemente creato	UC4

Tabella 3.2: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVN-1	Il backend dell'applicazione deve essere realizzato utilizzando i servizi cloud	-
RVN-2	L'applicazione deve essere utilizzabile solo dai clienti aggiunti da Nextep, implementando un sistema d'accesso tramite il <i>token</i>	-
RVN-3	L'intero progetto deve essere accompagnato da documentazione completa	-

Capitolo 4

Progettazione

In questo capitolo vengono illustrate le strategie di progettazione adottate per la realizzazione del prodotto in questione. La progettazione viene descritta ad alto livello senza descrivere in dettaglio tutti i diagrammi delle classi.

4.1 Progettazione Frontend

Come prima cosa sono stati realizzati utilizzando Figma i prototipi delle pagine da creare. In questo modo è stato possibile capire sin da subito la struttura delle pagine, rendendo così molto semplice la progettazione architetturale di tale pagine.

4.1.1 Pagina contenente l'editor

Questa pagina contiene l'editor *drag and drop*. Per creare la struttura di questa pagina sono stati studiati diversi editor online che forniscono le stesse funzionalità (in diversi contesti). Dopo una attenta analisi delle strutture dei diversi editor online è stato concepita la seguente struttura:



Figura 4.1: Mock pagina contenente l'editor

- * Il primo contenitore in alto a sinistra dovrà contenere tutti i componenti trascinabili. Ogni elemento rappresenta uno specifico tipo di oggetto HTML e CSS, come ad esempio il testo, banner, titolo, immagine ecc;
- * Il secondo contenitore a sinistra contiene tutte le proprietà modificabili per ogni elemento;
- * Il contenitore in centro(container) rappresenta la zona dove tutti gli elementi vengono trascinati. In questo contenitore viene visualizzato a schermo il contenuto HTML e CSS di ogni elemento trascinato. Inoltre dovrà essere possibile modificare tale contenuto ed eliminare un elemento se necessario;
- * Il contenitore a destra dovrà contenere tutti i template realizzati dagli utenti. Quindi esso contiene semplicemente una lista di tutti i template.

4.1.2 Pagina contenente il widget

Questa pagina contiene una semplice lista che dovrà contenere tutti i template da visualizzare nella pagina dei *tickets*, in modo che essi possono essere scelti dagli agenti di Zendesk.

4.1.3 Pagina login

Semplice pagina contenete il *form* per il login. Una volta inseriti i valori validi l'utente sarà autenticato come amministratore, caricandoli così la pagina degli amministratori.

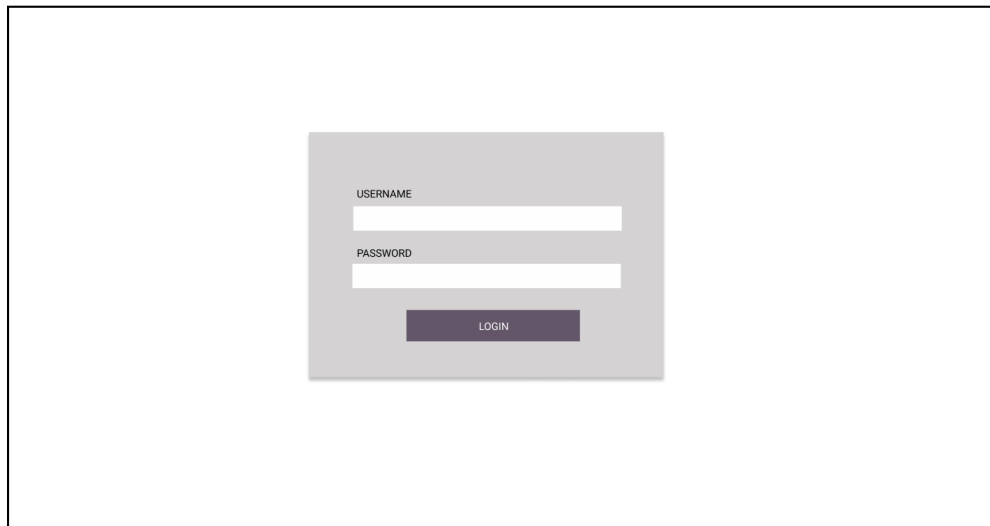


Figura 4.2: Mock pagina di login

4.1.4 Pagina degli amministratori

Questa pagina dovrà essere realizzata per gli amministratori di Nextep, per permettere a loro di gestire tutti i clienti utilizzatori dell'applicazione.



Figura 4.3: Mock pagina degli amministratori

- * Contiene un semplice form per aggiungere un nuovo cliente;
- * Contiene una semplice tabella, dove ogni elemento della tabella contiene le informazioni di un singolo cliente.

4.1.5 Struttura applicazione

Essendo un'unica applicazione che contiene sia le pagine degli amministratori di Nextep che le pagine dell'applicazione Zendesk, si è pensato quindi una struttura illustrata nel seguente diagramma.

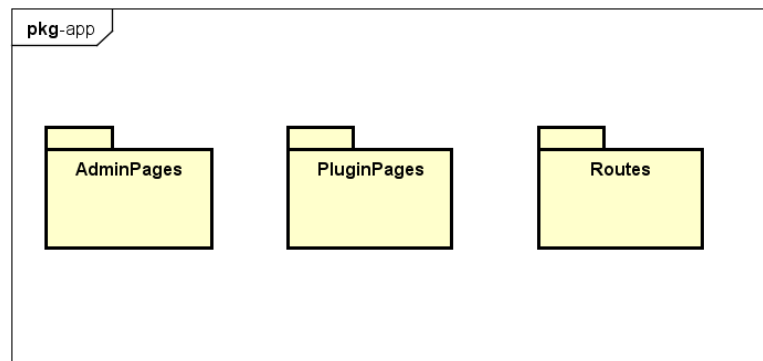


Figura 4.4: Struttura applicazione

- * **AdminPages:** contiene tutti i componenti e i servizi¹ che vanno a formare le pagine (login e di amministratori) per gli utenti di Nextep;
- * **PluginPages:** contiene tutti i componenti e i servizi per la pagina contenente l'editor e la pagina contenente il widget;
- * **Routes:** contiene tutto il codice per la gestione della navigazione dell'applicazione.

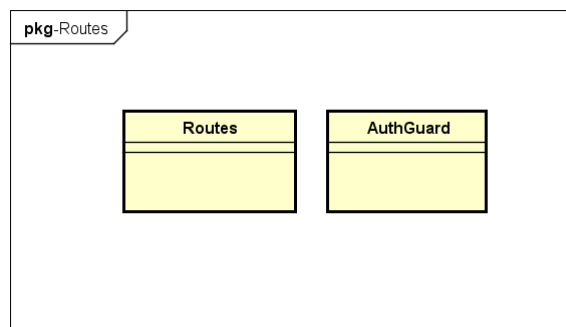


Figura 4.5: Struttura Routes

¹Nel Appendice si trova una descrizione completa di un servizio Angular.

- * L'oggetto **Routes** contiene un array di tutte le navigazioni dell'applicazione,
- * L'oggetto **AuthGuard** serve per bloccare la navigazione alla pagina degli amministratori finchè l'utente non effettua il login.

4.1.6 PluginPages

Contiene tutti i componenti e servizi per realizzare le pagine contenenti l'editor e il widget.

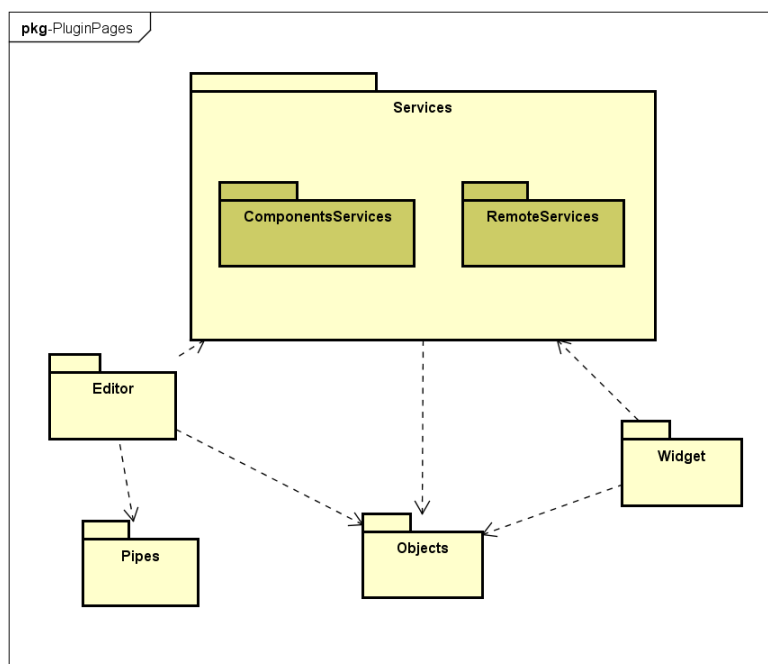


Figura 4.6: Struttura PluginPages

- * **ComponentsServices:** contiene tutti i servizi che permettono di scambiare i dati tra i componenti locali;
- * **RemoteServices:** contiene tutti i servizi utilizzati per comunicare con il lato backend dell'applicazione. Quindi principalmente per leggere, aggiungere o rimuovere i template dal database su AWS;
- * **Objects:** contiene tutti i tipi creati per rappresentare diversi tipi di dati;
- * **Editor:** contiene tutti i componenti Angular che formano le pagine contenenti l'editor;
- * **Widget:** contiene tutti i componenti Angular che formano le pagine contenenti il widget;
- * **Pipes:** contiene le pipes di Angular.

4.1.7 AdminPages

Contiene tutti i componenti e servizi per realizzare la pagina di login e la pagina degli amministratori.

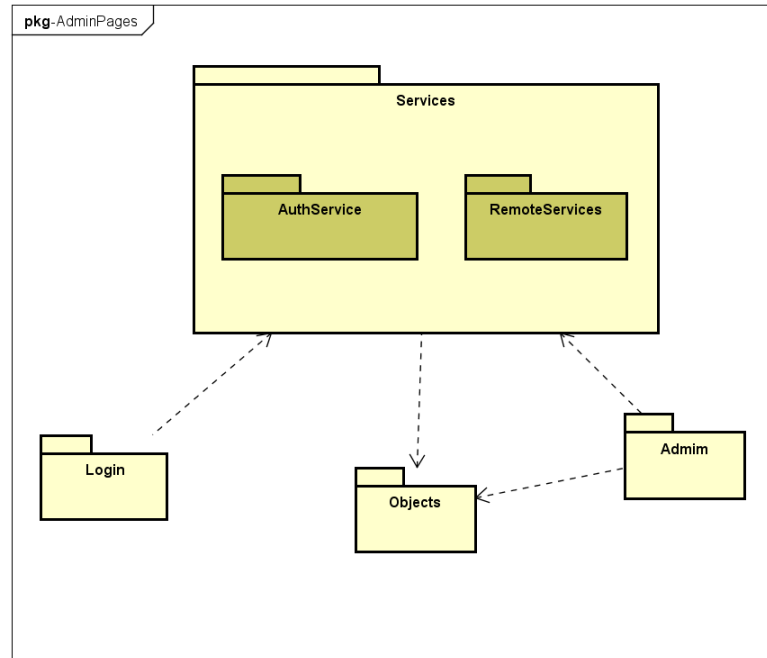


Figura 4.7: Struttura AdminPages

- * **AuthService:** contiene il servizio che permette di effettuare il login;
- * **RemoteServices:** contiene i servizi per aggiungere, leggere o rimuovere i clienti dal database presente su AWS;
- * **Objects:** contiene tutti i tipi creati per rappresentare diverse tipologie di dati;
- * **Admin:** contiene tutti i componenti Angular che formano la pagina degli amministratori;
- * **Login:** contiene tutti i componenti Angular che formano la pagina di login;

4.1.8 Atomic design

Per realizzare le diverse pagine dell'applicazione è stato utilizzato il concetto di Atomic Design. Creata da Brad Frost nel 2013, l'Atomic design è una metodologia composta da 5 differenti fasi, utile per creare un sistema di interfacce in maniera gerarchica. In questa metodologia si parte dai componenti più basilari possibili, fino ad arrivare alle pagine finali. E' quindi un approccio bottom-up.

Atomi: in fisica un atomo è la più piccola particella di un elemento che non subisce alterazioni nelle trasformazioni chimiche; nell'Atomic Design gli atomi sono i blocchi



Figura 4.8: Elementi atomic design

fondamentali che comprendono tutta l'interfaccia. Questi atomi comprendono elementi HTML come tipografia, palette colori, input, bottoni e altri elementi che non possono essere suddivisi ulteriormente senza cessare di essere funzionali.

Molecole: sono semplici gruppi di elementi d'interfaccia che funzionano uniti. Quando combiniamo due oppure più atomi, creiamo quindi una molecola.

Nel contesto dell'applicazione gli atomi e le molecole sono rappresentate dagli elementi della libreria Angular Material, che successivamente sono utilizzati per realizzare l'intera interfaccia dell'applicazione.

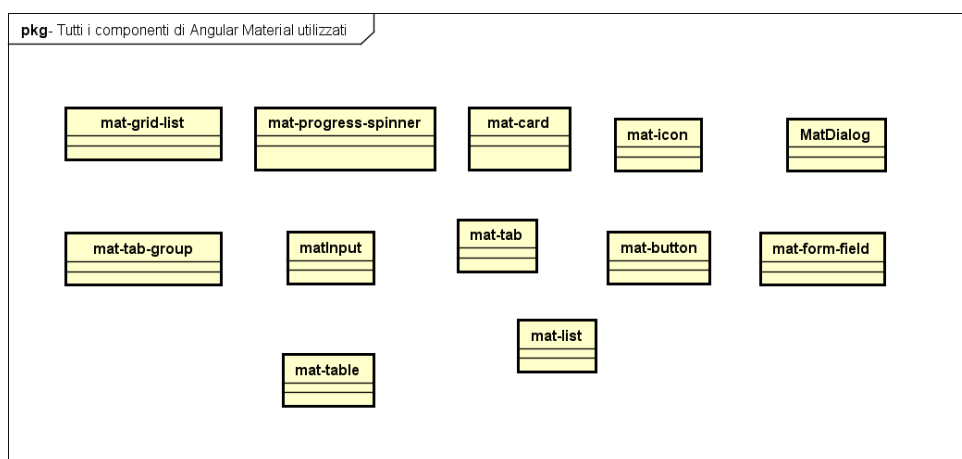


Figura 4.9: Componenti Angular Material che formano gli atomi e le molecole dell'applicazione

Gli organismi: sono dei componenti più o meno complessi, composti da gruppi di molecole e/o atomi e/o altri organismi. Questi organismi creano diverse sezioni all'interno della nostra interfaccia. Un esempio può essere un menu di navigazione, che è formato in media da diversi pulsanti/link.

I templates: sono creati dall'insieme dei nostri atomi, molecole ed organismi creando così la prima idea dello scheletro della pagina.

Le pagine: sono dei templates riempiti di contenuto reale, come immagini, testi, elementi grafici, advertising, ecc. Una pagina quindi è formata da molti template.

4.2 Progettazione Backend

4.2.1 Architettura a microservizi serverless

Il backend dell'applicazione è realizzato utilizzando i servizi web di Amazon, i servizi scelti sono stati descritti nel capitolo 2. In questa sezione viene descritto come questi servizi comunicano tra di loro. La seguente immagine mostra la panoramica dell'architettura della applicazione Zendesk realizzata(per la pagina degli amministratori la situazione è leggermente diversa). L'immagine descrive come un template viene caricato nel database NoSQL(database non relazionale) presente su Amazon.

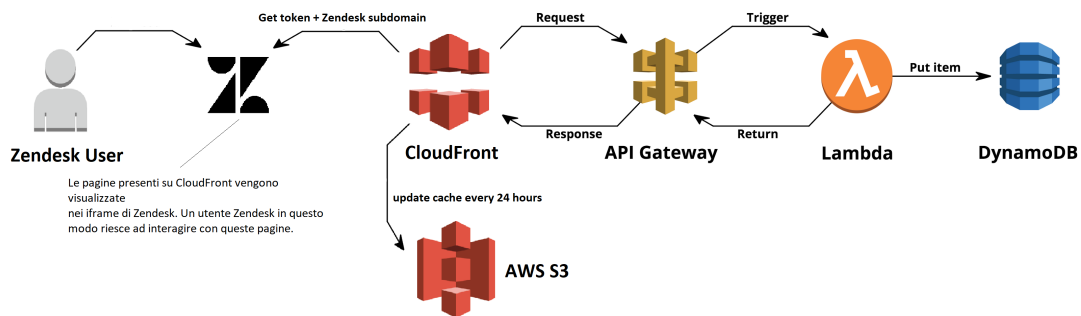


Figura 4.10: Flusso completo dell'architettura

- * Zendesk User interagisce con l'applicazione tramite l'interfaccia di Zendesk;
- * Le pagine dell'applicazione Angular sono caricate da CloudFront sui iframe presenti su Zendesk;
- * Le pagine appena caricate nei ifrmae leggono il token e il nome del sottodominio della piattaforma in cui si trovano utilizzando l'oggetto ZAFClient(descritto nel capitolo 2);
- * Utente crea un nuovo template e lo salva;
- * L'editor manda la richiesta all'API Gateway inviando nel header il token ed il nome del dominio;
- * API Gateway come prima cosa verifica il token ed il nome del sottodominio se sono validi. Se sono validi genera un evento che innesca una funzione lambda, altrimenti(token e nomedominio non validi) ritorna un messaggio d'errore;
- * La funzione lambda riceve dati dall'evento generato da API Gateway. Utilizzando AWS-SDK interagisce con il database nosql e salva il nuovo item(template).
- * Funzione lambda ritorna un messaggio di successo;

4.2.2 Architettura pagina degli amministratori

Il backend cambia leggermente per quanto riguarda la pagina degli amministratori di Nextep. Per accedere a questa pagina bisogna effettuare il login, il quale è stato

implementato utilizzando il servizio Cognito che permette di gestire un pool di utenti in maniera molto semplice. La seguente immagine descrive il funzionamento di questa pagina.

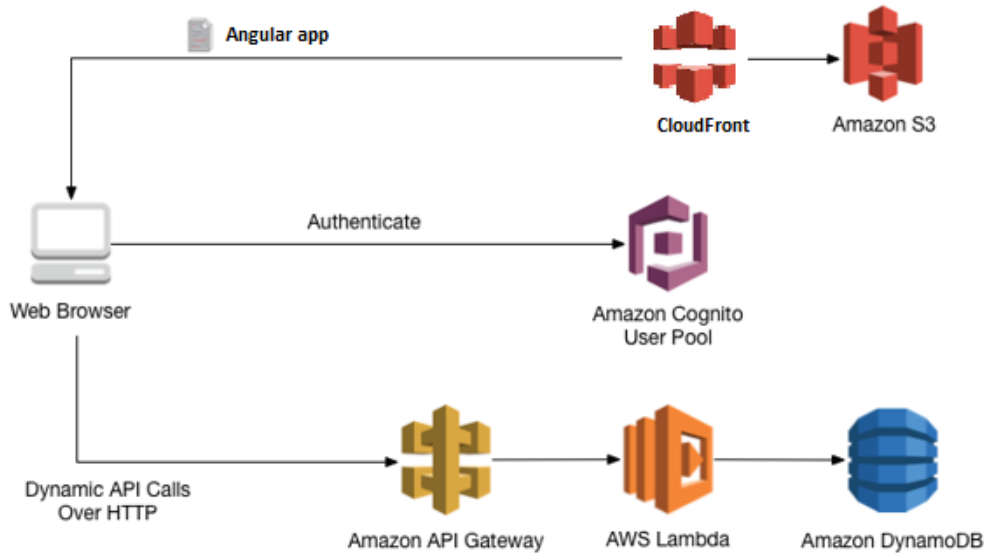


Figura 4.11: Flusso completo dell'architettura pagina admin

- * La pagina di login viene caricata nel browser;
- * L'utente inserisce le sue credenziali corrette per effettuare il login;
- * AWS Cognito verifica le credenziali, se sono valide ritorna il token di accesso altrimenti un errore;
- * Le credenziali sono valide, l'utente accede alla pagina degli amministratori;
- * L'utente ora può comunicare con l'API Gateway inviando il token ritornato da AWS Cognito. Questo token permette all'utente di inserire oppure eliminare utenti dal database NoSQL presente su AWS;
- * L'inserimento oppure l'eliminazione di un cliente esegue lo stesso flusso visto sopra per i template. Eccezione fatta per l'AWS Cognito, la pagina degli amministratori ha la stessa architettura.

Capitolo 5

Prodotto realizzato

In questo capitolo verrà spiegata in dettaglio come avviene l'interazione dell'utente Zendesk con l'applicazione Zendesk e dell'utente Nextep con la pagina degli amministratori.

5.1 Editor

Una volta installata l'applicazione sulla piattaforma Zendesk viene visualizzata automaticamente l'icona dell'applicazione nella sidebar della piattaforma.

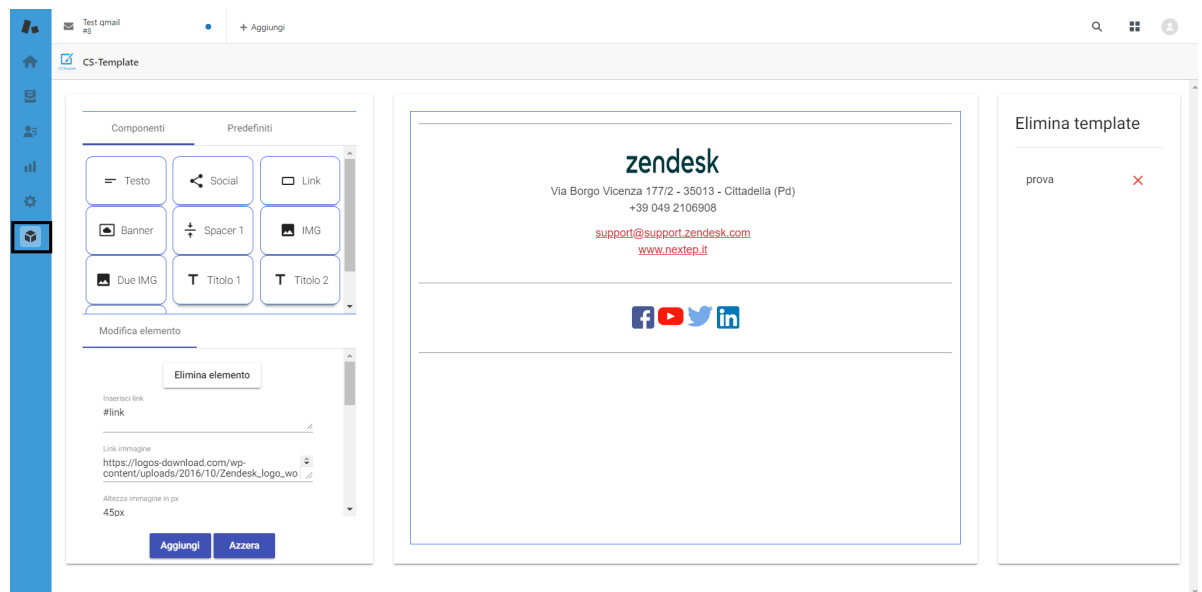


Figura 5.1: Editor realizzato

- * L'utente può trascinare qualsiasi elemento presente nei "componenti" nel *container* centrale;
- * Il *container* visualizza a schermo il contenuto HTML e CSS di ogni elemento trascinato in esso;

- * Facendo doppio click su qualsiasi elemento presente nel *container* centrale è possibile modificare il suo contenuto oppure eliminarlo dal *container*;
- * Gli elementi nel *container* possono essere ordinati in qualsiasi modo semplicemente facendo *drag-and-drop*;
- * E' possibile azzerare il contenuto del *container* semplicemente cliccando il pulsante "Azzerà";
- * Una volta realizzato il template desiderato l'utente può salvarlo cliccando il pulsante "Aggiungi". Verrà chiesto all'utente di inserire il nome del template, inserendo il nome valido il template verrà automaticamente salvato sul database NoSQL di Amazon;
- * A destra nell' "Elimina template" è possibile visualizzare tutti i template realizzati e se necessario eliminarli semplicemente cliccano sull'icona "X".

5.2 Widget

Il widget permette all'agente di Zendesk di utilizzare i template realizzati nelle risposte verso clienti.

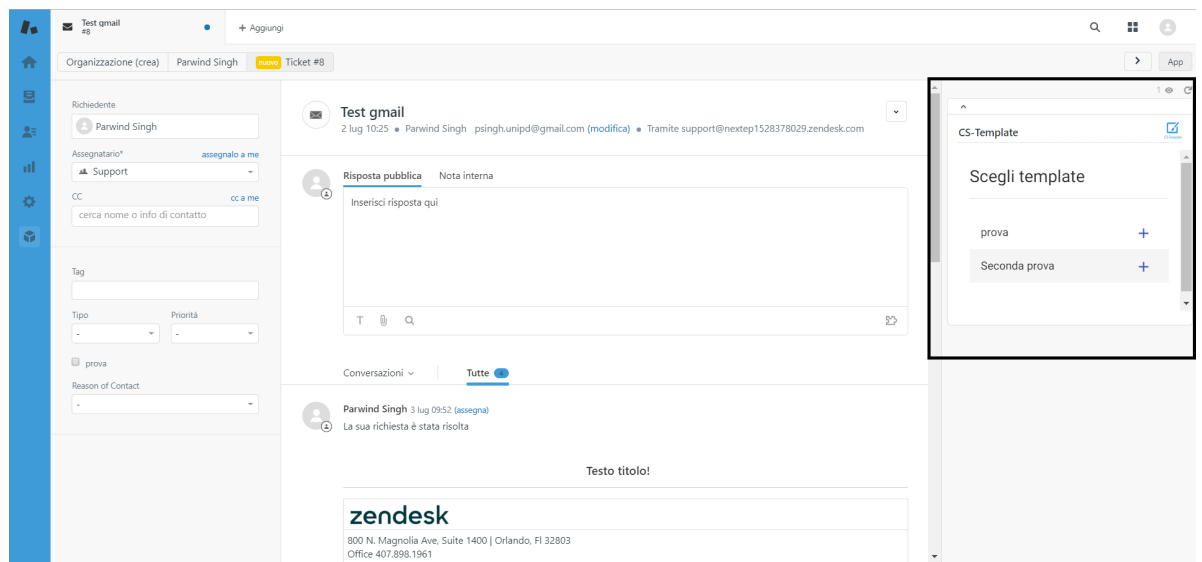
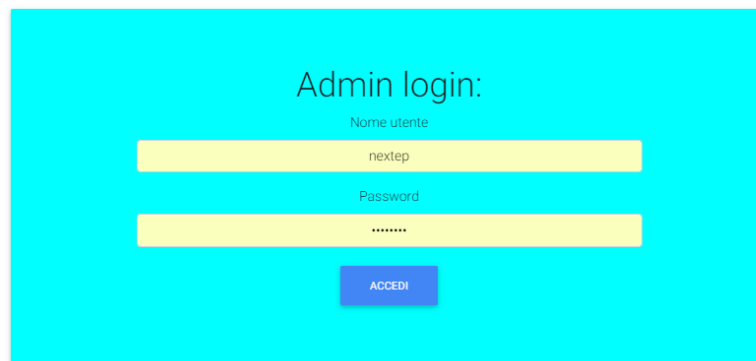


Figura 5.2: Widget realizzato

- * L'utente visualizza nel widget tutti i template realizzati;
- * L'utente può selezionare il template da utilizzare nella risposta;
- * Il template viene automaticamente aggiunto alla risposta.

5.3 Pagina di login

Nella immagine di seguito viene visualizzata il *form* di login contenuto nella pagina di login. Una volta inseriti i dati corretti, viene automaticamente caricata la pagina degli amministratori. Se i dati inseriti non sono corretti, viene mostrato a schermo un messaggio di errore.



The image shows a login form titled "Admin login:" on a light blue background. It contains two input fields: "Nome utente" (username) with the text "nextep" and "Password" with masked characters "*****". Below the fields is a blue button labeled "ACCEDI".

Figura 5.3: Form login

5.4 Pagina degli amministratori

Nella immagine di seguito viene visualizzata la pagina degli amministratori. In questa pagina è possibile aggiungere e rimuovere un cliente di Nextep che utilizza oppure andrà ad utilizzare l'applicazione CS-Template.



The image shows the admin page layout. At the top is a blue header with "CS-TEMPLATE" on the left and "Logout" on the right. The main content area has a light blue background and contains a form titled "Aggiungi nuovo dominio:". The form has five input fields: "Nome dominio", "Nome azienda", "Codice cliente Nextep", "Email", and a blue "AGGIUNGI" button. Below the form is a table with the following data:

#	Domaino	Token	Azienda	Codice Cliente	Email	Data inizio	Elimina
0	nextep1528378029	pln6872up465	nextep1528378029	nextep1528378029	nextep1528378029@gmail.com	2018-08-03	Elimina

At the bottom is a blue footer with the text "© 2018 Copyright: Nextep-Ecosistemi digitali".

Figura 5.4: Pagina degli amministratori

Capitolo 6

Conclusioni

Questa sezione è dedicata alle valutazioni e considerazioni finali effettuate al termine del periodo di Stage

6.1 Raggiungimento degli obiettivi

Gli obiettivi descritti nella sezione 3.2 dedicata al tracciamento dei requisiti individuati durante il periodo di Analisi, hanno avuto il seguente risultato:

Tipo	Individuati	Soddisfatti
Requisiti totali	16	14
Requisiti obbligatori	11	11
Requisiti desiderabili	2	0
Requisiti di vincolo	3	3

Tabella 6.1: Soddisfacimento Requisiti

Durante lo stage ho acquisito diverse conoscenze che mi hanno formato e portato al raggiungimento degli obiettivi prefissati. Di seguito verranno elencate le conoscenze principali che hanno caratterizzato il periodo svolto in azienda.

- * **Angular:** ho avuto modo di studiare molto bene il framework Angular. Le SPA sono sempre più richieste nella realtà aziendale.
- * **REST API:** ho avuto modo capire molto bene come funzionano le REST API.
- * **AWS:** sicuramente la tecnologia più interessante sono stati i servizi web di Amazon. Sempre più aziende puntano a queste tecnologie. Si sente sempre più parlare dell'architettura serverless utilizzando in frontend Angular, React ecc.
- * **Lavoro aziendale:** ho imparato cosa significa lavorare in un'azienda, la suddivisione dei ruoli e l'importanza del lavoro in team. Nel contempo ho sviluppato anche la capacità di essere autonomo in certe circostanze, in modo da non dipendere costantemente da altre persone.

6.2 Valutazione personale sullo stage

In conclusione l'esperienza che ho vissuto durante il periodo di stage è stata molto formativa ed interessante, soprattutto perché ho potuto affrontare nuove tematiche rispetto al mio corso di studi in Università. Nonostante ciò, grazie alle conoscenze apprese durante gli anni di studio, sono riuscito ad apprendere velocemente quanto necessario per svolgere il progetto. Inoltre, ho messo alla prova le mie capacità confrontandomi, per la prima volta, con il mondo del lavoro e posso affermare di essere rimasto molto soddisfatto. Mi è stato fornito tutto il materiale ed aiuto necessari per svolgere al meglio il mio lavoro ed ho potuto stabilire un buon rapporto, lavorativo ed umano, con gli altri dipendenti dell'azienda. Al termine dello stage ho acquisito consapevolezza delle mie capacità e ritengo che sia stato un ottimo periodo formativo.

Appendice A

Appendice A

A.1 Servizi di Angular 6

Un servizio in Angular 6 è una classe di tipo [Singleton](#) che implementa funzionalità condivise dai vari elementi di un'applicazione, siano essi componenti che altri servizi. Ne esiste solo un'istanza di uno servizio per tutta l'applicazione, in questo modo i dati condivisi dai diversi componenti sono gli stessi.

L'implementazione di un servizio avviene utilizzando il *decorator* `@Injectable()` di Angular. Questo *decorator* (design pattern [Decorator](#)) aggiunge alla classe dei metadati che permettono ad Angular di iniettare il servizio nei componenti come una dipendenza. Viene utilizzato il *Dependency injection* ([DI](#)) design pattern per iniettare le dipendenze dei servizi nei componenti.



Figura A.1: Injector di Angular

Dependency injection è utilizzato ovunque nel framework Angular. In un'applicazione Angular un componente in media consuma molti servizi utilizzando DI, che li fornisce accesso a tutti i metodi e ai dati forniti dal servizio.

Quando un componente viene creato come prima cosa vengono controllate le sue dipendenze. Se il componente dipende da qualche servizio viene generata l'istanza Singleton di tale servizio (se non esiste già), in questo modo vengono soddisfatte tutte le dipendenze del componente prima della sua creazione. La Dependency injection è gestito dall'oggetto *injector* di Angular, che ha il semplice compito di soddisfare le dipendenze.

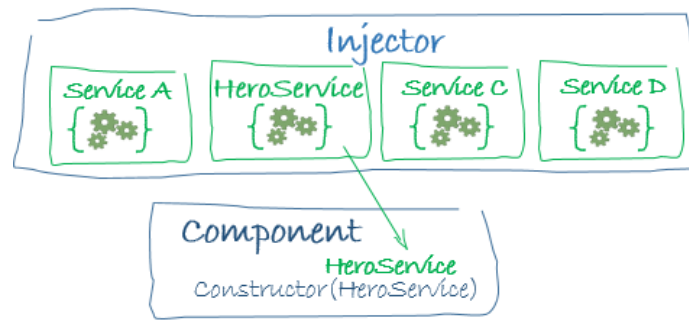


Figura A.2: DI in Angular

A.2 RxJS in Angular 6

RxJS (Reactive Extensions for JavaScript) è una libreria di *reactive programming* che permette di gestire le chiamate asincrone in maniera molto semplice. Questa libreria è già integrata dalla versione 4 in poi di Angular.



Figura A.3: Logo di Angular e RxJS

Questa libreria permette di implementare molto facilmente il design pattern **Observer**. Permette di trasformare qualsiasi tipo di Typescript in un *Subject*, ovvero qualcosa che può essere osservato dagli osservatori. Quando un *Subject* cambia nel tempo automaticamente tutti gli osservatori vengono avvisati del suo cambiamento.

Nel contesto dell'applicazione i *Subjects* sono utilizzati nei RemoteServices (capitolo 4) che sono dei servizi per le chiamate asincrone. I componenti si mettono in ascolto facendo *subscribe* a un oggetto Subject, appena la chiamata asincrona ritorna i dati il *Subject* si aggiorna e di conseguenza anche i componenti che lo stavano osservando, aggiornando così la *view* in modo reattivo.

Glossario

API Acronimo di Application Programming Interface (ovvero interfaccia di programmazione di una applicazione) si indicano un insieme di procedure (raggruppate assieme per formare un insieme di strumenti specifici) per il compimento di un determinato compito all'interno di un certo programma. [v](#), [6](#)

Decorator Decorator è un design pattern che permette di decorare un'oggetto utilizzando le funzioni d'altri oggetti. Utilizzato per diminuire l'esagerato utilizzato dell'ereditarietà. [. 35](#)

design pattern Questo termine fa riferimento a una soluzione progettuale generale ad un problema ricorrente che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software. [3](#)

DI Acronimo di Dependency Injection, è un design pattern che risolve il problema di dipendenze. Utilizzato soprattutto nella programmazioni ad oggetti, dove spesso molti oggetti contengono altri oggetti come parametri. Questo pattern permette di iniettare le dipendenze tra gli oggetti invece di avere istanze interne, in questo modo ogni oggetto può essere modificato senza causare danni in altri oggetti. [35](#)

EMCAScript 6 Spesso conosciuto con acronimo ES6 è una versione del linguaggio di scripting Javascript standardizzata [. 4](#)

framework Un framework è un'architettura logica di supporto (spesso un'implementazione logica di un particolare design pattern) su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore. Inoltre fornisce una raccolta di librerie ed oggetti già fatti. [3](#)

iframe L'iframe in informatica è un elemento HTML. Si tratta infatti di un frame "ancorato" all'interno della pagina, equivale cioè ad un normale frame, ma con la differenza di essere un elemento inline (interno) della pagina, non esterno. L'iframe viene generalmente utilizzato per mostrare il contenuto di una pagina web, o di una qualsivoglia risorsa, all'interno di un riquadro in una seconda pagina principale. [11](#)

Javascript JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso. [4](#)

JSON Acronimo di JavaScript Object Notation, è un formato dichiarativo basato sullo Standard ECMA-262 adatto all'interscambio di dati fra applicazioni client-server. [18](#)

mock Dal inglese "finto". Utilizzato nello sviluppo per simulare il comportamento degli oggetti reali, in questo modo è possibile provare delle parti di un sistema senza necessariamente implementare tutto. [10](#)

MVC Model-view-controller in informatica, è un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business. [3](#)

MVVM Una variante del pattern MVC è MVVM, Model View ViewModel. Questo pattern propone un ruolo più attivo della View rispetto a MVC: la View è in grado di gestire eventi, eseguire operazioni ed effettuare il data-binding. In questo contesto, quindi, alcune delle funzionalità del Controller vengono inglobate nella View, la quale si appoggia su un'estensione del Model: il ViewModel. Il ViewModel è quindi un Model esteso con funzionalità per la manipolazione dei dati e per l'interazione con la View. [3](#)

Observer E' un design patter che sta alla base di MVC, permette a molti osservatori di osservare un'unica fonte di verità. In questo modo tutti gli oggetti hanno i stessi dati. [36](#)

repository E' un ambiente condiviso di sviluppo software. In generale è una cartella condivisa da molti utenti nel quale è possibile lavorare insieme in maniera molto semplice. [9](#)

SDK Acronimo di Software Development Kit, si riferisce genericamente a un insieme di strumenti per migliorare lo sviluppo o la documentazione di software. [7](#)

serverless Nel contesto dell'applicazione, serverless è un metodo di creazione ed esecuzione di applicazioni e servizi che non richiede la gestione dell'infrastruttura. L'applicazione sarà comunque eseguita su server, ma la gestione di quest'ultimo sarà a carico di AWS. [6](#)

Singleton Singleton è un design pattern che permette di creare un'unica istanza di un oggetto in tutta l'applicazione. [35](#)

SPA Single Page Application è una metodologia per la creazione delle interfacce web. In questo caso esiste una sola pagina HTML, tutto il resto (navigazione, pagine) viene gestito tramite Javascript. I più famosi framework/librerie sono Angular, React e Vue. [3](#)

UI User Interface è un'interfaccia uomo-macchina, ovvero ciò che si frappone tra una macchina e un utente, consentendone l'interazione reciproca. [4](#)

UML Unified Modeling Language, linguaggio di modellizzazione e specifica basato sul paradigma orientato agli oggetti. [13](#)

W3C Recommendation La parola W3C significa World Wide Web Consortium, cioè il Consorzio Internazionale che si occupa di definire linee guida e standard non proprietari, le tecnologie, i protocolli e tutto quanto necessario per lo sviluppo del Web . [5](#)

Bibliografia

Riferimenti bibliografici

Architecting Angular Applications with Redux, RxJS, and NgRx. Packt Publishing Ltd., 2018.

Design Patterns, prima edizione italiana. Pearson, 2002.

HTML5 CSS3 Javascript - Pellegrino Principe. Apogeo, 2012.

Siti web consultati

Angular Docs. URL: <https://angular.io/docs>.

AWS. URL: <https://aws.amazon.com/>.

Typescript. URL: <http://www.typescriptlang.org/>.

W3School. URL: <https://www.w3schools.com/>.

Zendesk Developer. URL: <https://developer.zendesk.com/>.