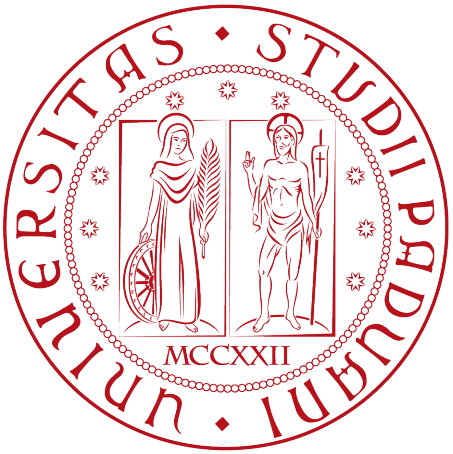
Università degli Studi di Padova

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

Anno Accademico 2017-2018

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 verrano descritte in dettaglio l'analisi dei requisiti, la progettazone, l'implementazione e la validazione dell'applicazione CS-template. L'applicazione in questione è stato realizzato utilizzanod le tecnolgie web inovvative sia per quanto riguarda lato frant-end dell'applicazione che quello back-end.
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 Professional.

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

Il secondo capitolo descrive e tecnologie utlizzate durante tutto il periodo di stage.

Il terzo capitolo formalizza tutti 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 approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

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: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

"Life is really simple, but we insist on making it complicated"

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. NomeDelProfessore, 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 . . . 2
- 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 editor drag-and-drop . . . 15
  - 3.1.3 Casi d'uso pagina contenente 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 28

5 Prodotto realizzato **29**
5.1 Editor 29
5.2 Widget 30

6 Conclusioni **31**
6.1 Consuntivo finale 31
6.2 Raggiungimento degli obiettivi 31
6.3 Conoscenze acquisite 31
6.4 Valutazione personale 31

A Appendice A **33**

Bibliografia **37**

Elenco delle figure

1.1 Logo di Nextep: immagine tratta dal sito dell'azienda 1

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 applicazione 22
4.6 Struttura applicazione 23
4.7 Struttura applicazione 24
4.8 Elementi atomic design 25
4.9 Componenti Angular Material che formano gli atomi e le molecole dell'applicazione 25
4.10 Componenti Angular Material che formano gli atomi e le molecole dell'applicazione 27
4.11 Archittetura 28

5.1 Editor realizzato 29
5.2 Editor realizzato 30

Elenco delle tabelle

3.1 Tabella del tracciamento dei requisti funzionali 18
3.2 Tabella del tracciamento dei requisiti di vincolo 18

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 Italia, Allos Sud Africa, Allos USA e Zero12. 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 Ltd, 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.

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

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 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'aziedna 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 strutta della applicazione ad alto livello:

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

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 piattaforam opensourse realizzato da Google nel 2016 che permette di creare le SPA, sfruttando i 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 web viene costruitta da diversi componenti. Un componente in Angular in generale è una piccola parte della view che reppressenta una specifica funzionalità(esempio la navbar). Ogni component ha una propria logica strutturale (scritta tramite appositi marcatori HTML), di presentazione (scritta con appositi fogli di stile CSS oppure SCSS) 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 Materia DesignG. Questo libreria è stata sviluppata sempre da Google e permette di realizzare delle UI molto avvanzata in maniena molto semplice. Fornendo una serie di semplici componenti(come i pulsanti, inputbox ecc) di Angular già fatta permette algli sviluppatori di risparmiare una notevole quantità di tempo. Tutti i componenti sono testati da Google garantendo cosi un corretto funzionamento.

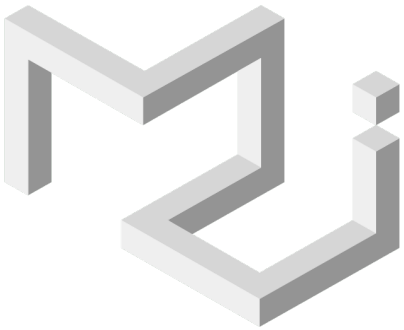


Figura 2.2: Logo di Material UI

2.1.3 Typescript

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

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

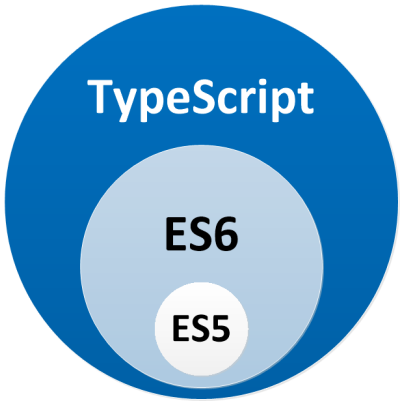


Figura 2.3: Typescript rispetto ES6 e ES5

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 front-end dell'applicazione. Una delle principali vantaggi di Angular sta proprio nel utilizzo di HTML pure 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/CSS è un linguaggio utilizzato per definire la presentazione dell'applicazione. Poichè 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 oggigiorno 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. La scelta è stata fatta soprattutto per la pololarità dei servizi Amazon, essendo Amazon primo a fornire questo tipo di servizio sono attualmente molto più utilizzati rispetto gli altri due. Per quanto riguarda i prezzi, quantità e qualità di servizi offerti da tuti i provider non è stato trovato un rirevante 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 richiesta dalle applicazioni, dopo ogni richiesta questo servizio genera un evento che esegue qualcosa(una funzione lambda, un altra chiamata ecc).

Figura 2.7: Logo API Gateway

2.2.3 AWS Lambda

Sono delle semplici funzioni che ricevano come input un evento e possono ritornare qualche valore. Possono essere scritte in molti linguaggii dprogrammazione, come Nodejs(javascript), Java, C++ e Phython.

Queste funzioni possono comunicare con qualsiasi servizio di Amazon utilizzando AWS-SDK. In questo progetto sono state utilizzate diverse funzioni di questo tipo, princiaplamente per scrivere o leggere i dati su database.

Figura 2.8: Logo di AWS Lambda

2.2.4 AWS DynomoDB

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 tempalte sono stati salvati su questo database. L'acceso 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 stassa è stata caricata su questo servizio.

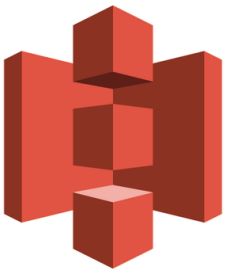


Figura 2.10: Logo di AWS S3

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 garatire l'accesso alla pagina di amministratore 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 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 cosi di lavorare su una repository privata.

Figura 2.13: Logo di Gitlab

2.3.3 Atom

Atom è un editor di testo Open-Sourse 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.

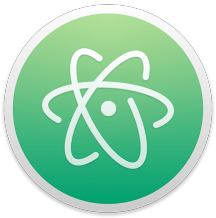


Figura 2.14: Logo di Atom

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 rappressentono un app di Zendesk. Take package può essere installato sulla propria piattaforma Zendesk come una applicazioen privata oppure caricata(gratuitamente oppure a pagamento) su Market place di Zendesk rendendola cosi disponibile a tutti gli utenti nel mondo. Il package è formato da due file JSON(manifest.json) e una cartella assets.La cartella contiene le icone da visualizzare dopo l'installazione dell'applllicazione sulla piattaforma, mentre il file manifest.json contiene tutte le informazioni riguardo l'applicazione. Il seguente codice mostra la struttuta 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'atrtibuto "location" è quello più importatnte. Questo attributo specifica dove visualizzare l'appllicazione sulla piattaforma dopo l'installazione. Si ha la possibilità di scegliere tra piu di dieci possizioni differenti, e per ogni posizione speficato bisgona 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. Nel contesto del applicazione è richiesto di inserire obbligatoriamente un token da 16 cifre, l'inserimento sbaglaito di tale valore comporta inutilizzo dell'applicazione CS-Template.

2.5.1 ZAF Client SDK

Ovviamente non avrebbe senso semplcemente 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 Zenedesk, è reso disponibile una libreria(ZAF Client SDK) da utilizzare nella propria applicazione web. Questa libreria fornisce un oggetto speciale(ZAFClient) con una ciquantina di metodi che permettono(una volta caricata la pagina nella piattaforma) di interagire con le diverse funzionalità di Zendesk.

Seguente codice invoca il metodo "invoke" dell'istanza "client" del oggetto ZAFClient per aggiungere 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 fornire una descrizione completa e precisadi tutti i requisitiG 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 Unified Modeling Language (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 autori del sistema saranno così classificati:
  - Utente Zendesk: sono gli utenti della piattaforma Zendesk. Possono essere gli agenti(persone che gestiscono le richieste dei cleinti) oppure gli amministratore di Zendesk;
  - Utente generico: utente qualsiasi che non ha ancora effetuato l'accesso alla pagina degli amministratori;
  - Amministratore: 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 qui scopo princiaplae 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 uenti amministratori di Nextep con le credenziali valide.

UC1: Login pagina amministratori

Attori Principali: Utente generico.

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

Precondizioni: L'utente non autenticato.

Postcondizioni: Il sistema riconosce l'utente amministratore.

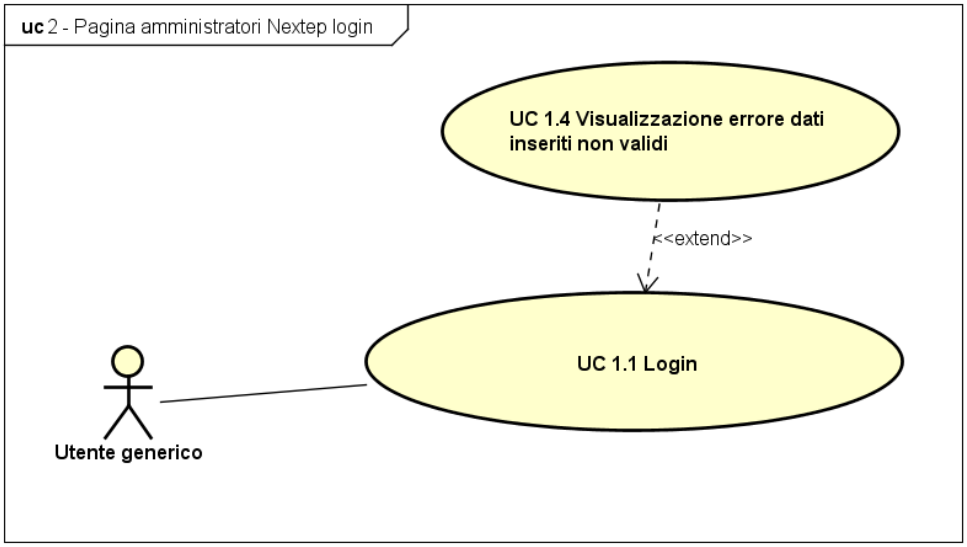


Figura 3.1: UC 1 - pagina login

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 nuovoi.

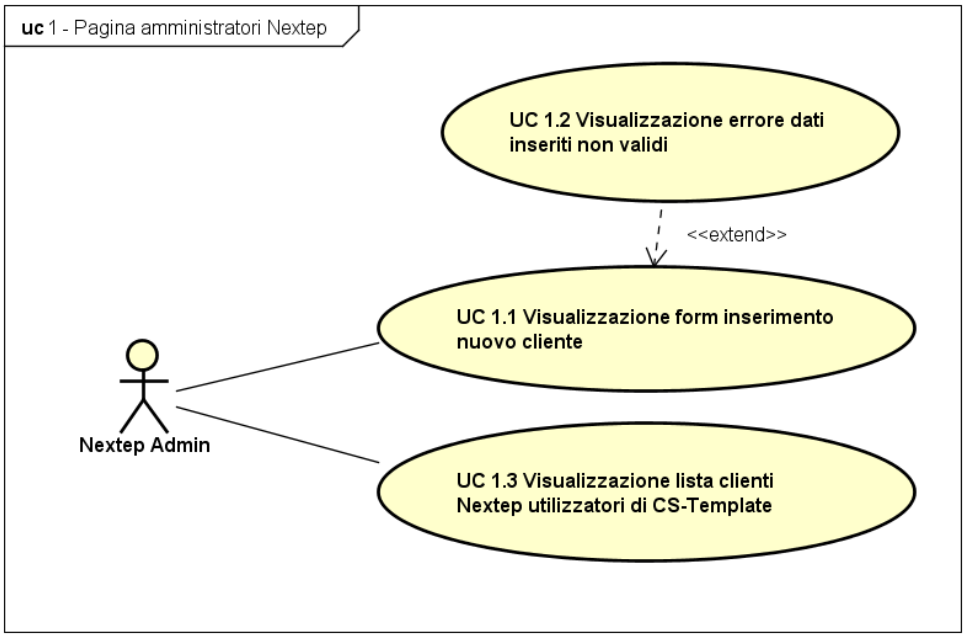


Figura 3.2: UC 2 - funzionalità pagina admin

3.1.2 Casi d'uso pagina contenente 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 del editor.

UC3: Editor drag-and-drop

Attori Principali: Utente Zendesk.

Descrizione: Caso d'uso descrive tutte le funzionalità ad oalto 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 contentuto HTML e CSS.

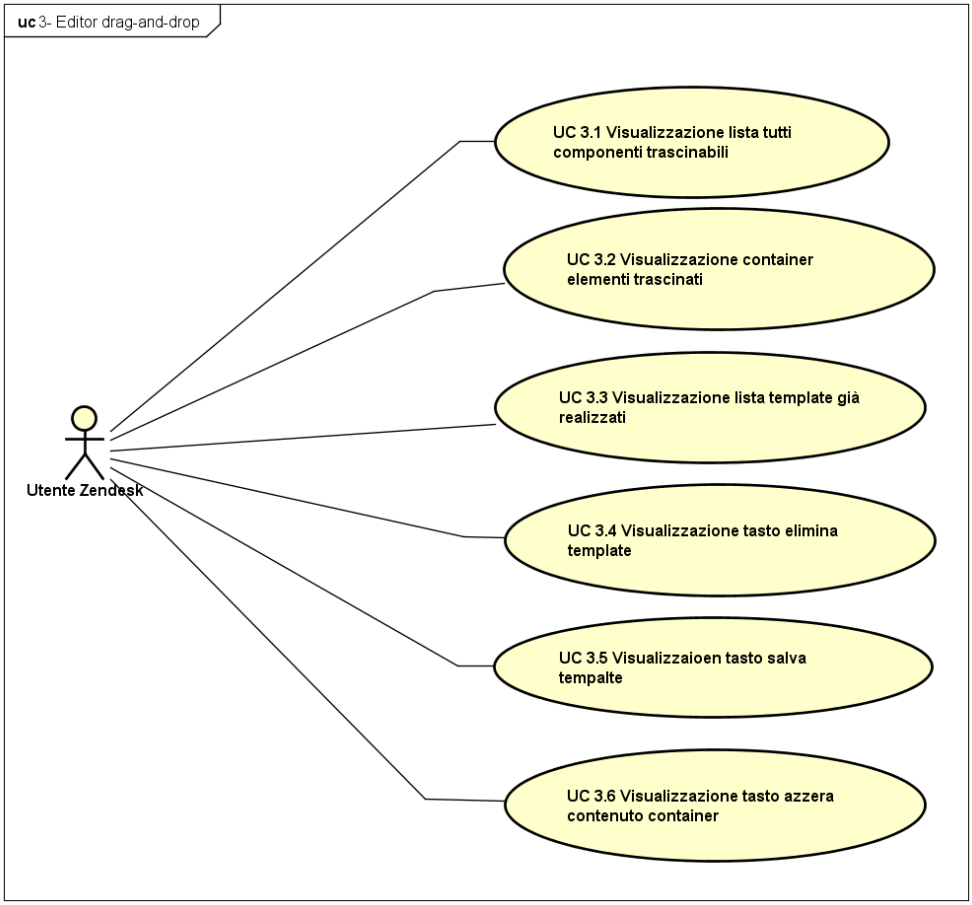


Figura 3.3: UC3 - pagina contenene l'editor

3.1.3 Casi d'uso pagina contenente widget

Questa pagina web contiene il widget che verrà visualizzato successivamente nella piattaforma Zendesk in un iframe. Il widget verrà mostrato quando verrà aperto una richeista 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.

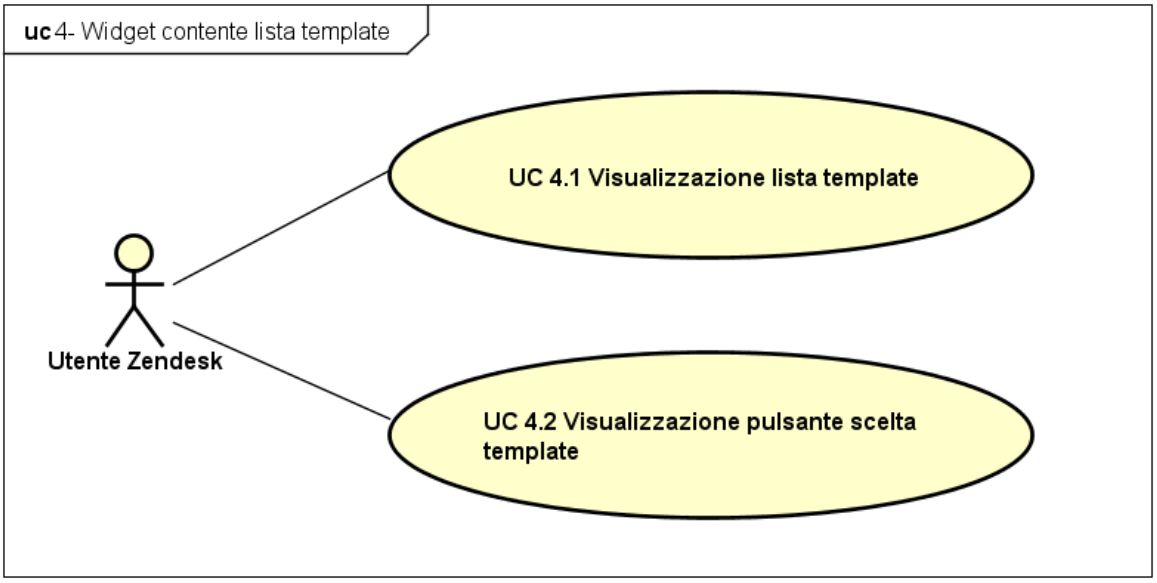


Figura 3.4: UC3 - pagina contenente il widget

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/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

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 requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	Personale di Nextep può effetuare il login nella pagina degli amministraori	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	Utenet Zendesk può utilizzare l'editor drag-and-drop	UC3
RFN-5	Utenet Zendesk può creare nuovi template	UC3
RFN-6	Utenet Zendesk può salvare il template creato	UC3
RFN-7	Utenet Zendesk può eliminare template creati	UC3
RFN-8	Utenet Zendesk può azzerrare il contenuto del editor	UC3
RFN-10	Utenet Zendesk può impostare il nome del template	UC3
RFN-11	Utenet Zendesk può utilizzare il temaplte creato nella risposta verso il cliente	UC4
RFD-12	Utenet Zendesk può esportare tutto i template creati in un file json come backup	UC4
RFD-13	Utenet 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
RVO-1	Il backend dell'applicazioend deve essere realizzato utilizzando gli servizi cloud	-
RVO-2	L'applicazione deve essere utilizzabile solo dai clienti aggiunti da Nextep, implementando un sistema d' accesso tramite il token	-
RVO-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 pagina, rendendo cosi molto semplice la progettazione architetturale di tale pagine.

4.1.1 Pagina cotenente 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 rappresenta la zona dove tutti gli elementi vengono trascinati. In questo contenitore viene visualizzato a schermo il contenuto HTML e CSS contenuto in ogni elemento. 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 un semplice lista che dovrà contenere tutti i template da visualizzare nella pagina dei tickets, in modo che essa posono essere scelti dagli utenti da Zendesk.

4.1.3 Pagina login

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

Figura 4.2: Mock pagina di login

4.1.4 Pagina degli amministratori

Questa pagina dovrà essere realizzata per gli amministratore di Nextep, per permettere a loro di gestire tutti i clienti utilizzatore del 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 informazione di un singolo cliente.

4.1.5 Struttura applicazione

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

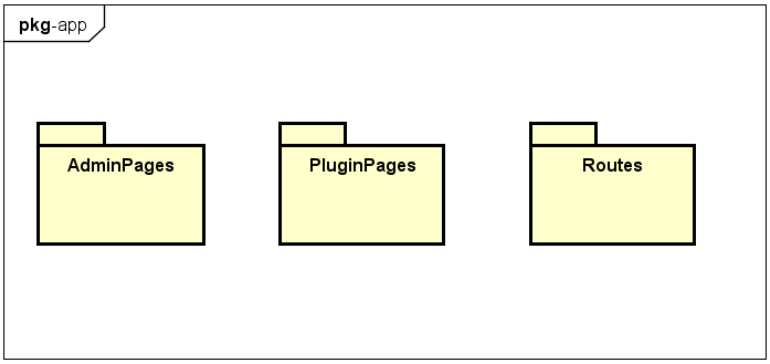


Figura 4.4: Struttura applicazione

* **AdminPages:** contiente tutti i componenti e i servizi 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 pagine contenente l'editor e la pagina contenente il widget;
* **Routes:** contiene tutto il codice per gestione della navgazione dell'applicazione.

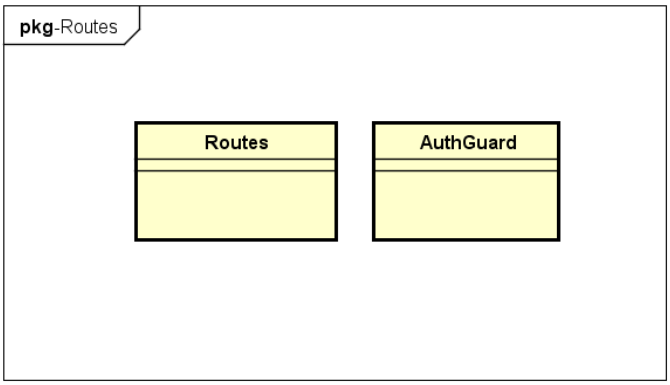


Figura 4.5: Struttura applicazione

* 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 contenente l'editor e il widget.

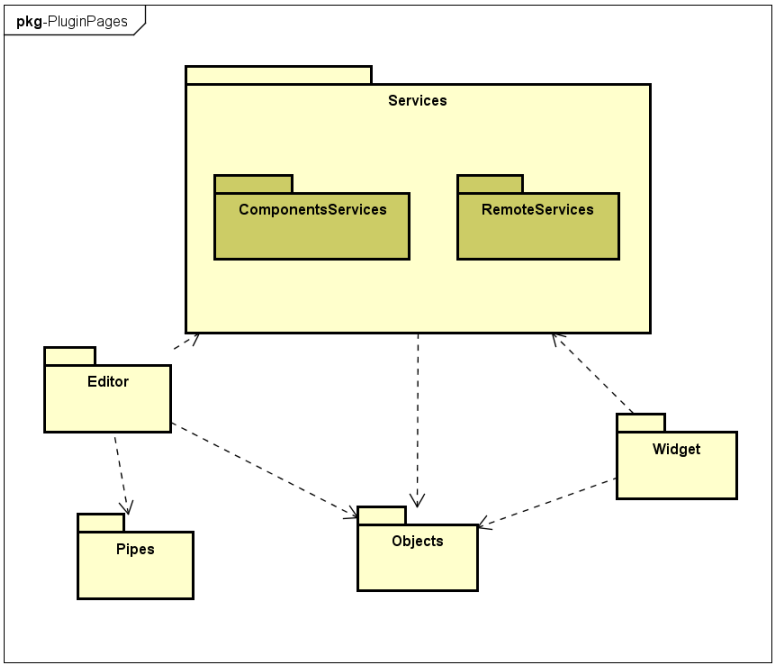


Figura 4.6: Struttura applicazione

* **ComponentsServices:** contiente tutti i servizi che permettono di scambiare i dati tri componenti locali;
* **RemoteServices:** contiene tutti i servizi utilizzati per communicare con il lato backend dell'applicazione. Quindi principalmente per leggere, aggiungere o rimuovere i tempalte dal database su AWS;
* **Objects:** contiene tutti i tipi creati per rappresentare diversi tipi di dati;
* **Editor:** contiene tutti i componenti Angular che formano la pagine contenente l'editor;
* **Widget:** contiene tutti i componenti Angular che formano la pagina contenente 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.

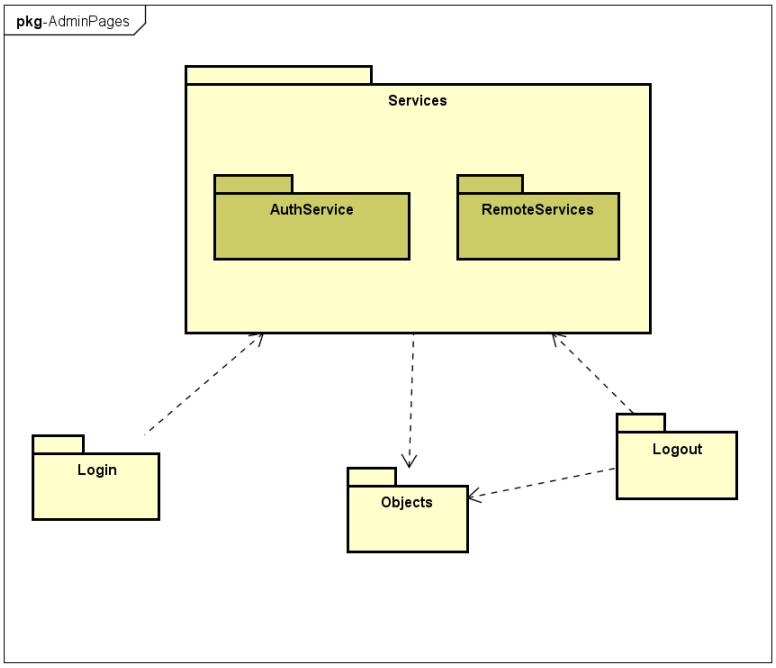


Figura 4.7: Struttura applicazione

* **AuthService:** contiene il servizio che permette di effetuare il login;
* **RemoteServices:** contiene i servizi per aggiungere, leggere o rimuovere i cliente dal database presente su AWS;
* **Objects:** contiene tutti i tipi creati per rappresentare diversi tipologia di dati;
* **Admin:** contiene tutti i componenti Angular che formano la pagine degli amministratori;
* **Login:** contiene tutti i componenti Angular che formano la pagine 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 metologia si parte dai componenti piu basilari possibili, fino ad arrivare alle pagine finali. Quindi è 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 della applicazione gli atomi e le molecole sono rappresentate dai elementi della libreria Angular Material, che successivamente sono utilizzate per realizzare l'interfaccia di intera applicazione.

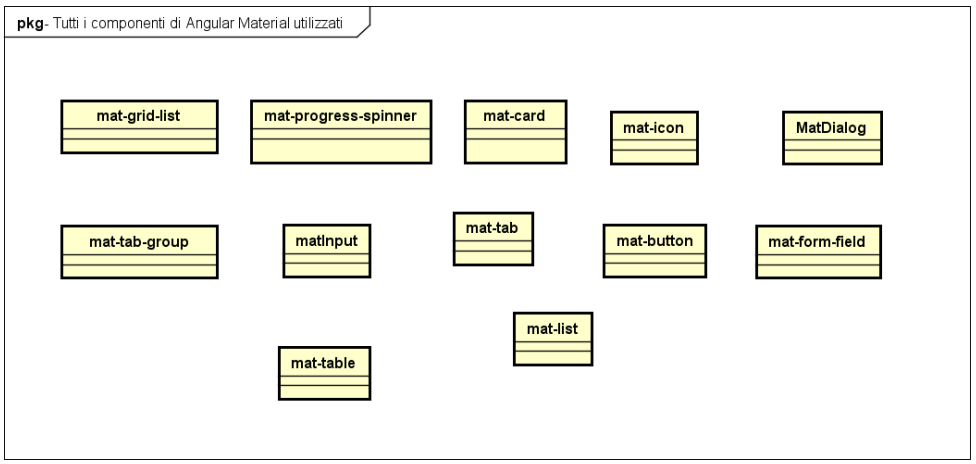


Figura 4.9: Componenti Angular Material che formano gli atomi e le molecole dell'applicazione

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 dai nostri atomi, molecole e organismi. Creando così la prima idea di 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 sutilizzando i servizi web di Amazon. I servizi scelti sono stati descritti nel capitolo 2. In questa sezione viene descritto come questi servizi communicano tra di loro. Il seguente immagine mostra la panoramica dell'architettura della applicazione Zendesk realizzata(per la pagina degli amministratori la situazione è leggermente diversa). L'immagine di seguito descrive come un template viene caricato nel database nosql presente su Amazon.

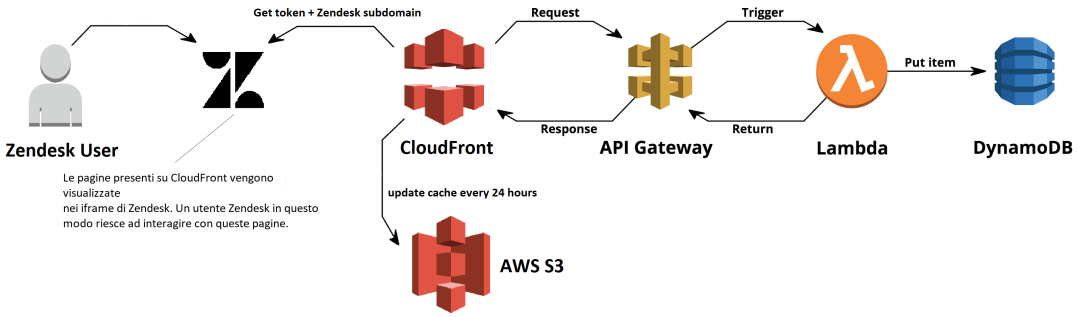


Figura 4.10: Componenti Angular Material che formano gli atomi e le molecole dell'applicazione

* Zendesk User interagisce con la applicazione tramitte 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 dalla piattforma in qui si trovano utilizzando lo oggetto ZAFClient(descritto in capitolo 2);
* Utente crea un nuovo template e lo salva;
* L'editor manda la richiesta al API Gateway inviando nel header il token e il nome del dominio;
* API Gateway come prima cosa verifica il token e il nome del sottodominio se sono validi. Se sono validi genera un eventi che inesa una funzione lambda altrimenti(token e nomedominio non validi) ritorna un messaggio d'erroe;
* La funzione lambda riceve dati dal evento generato da API Gateway. Utilizzando AWS-SDK interagisce con il database nosql e salva il nuovo item(template).
* Funazione 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 effetuare il login. Il login è stato implementato utilizzanod il servizio Cognito, che permette di gestire un pool di utenti in maniera molto facile. Il seguente immagine descrive il funzionamento di questa pagina.

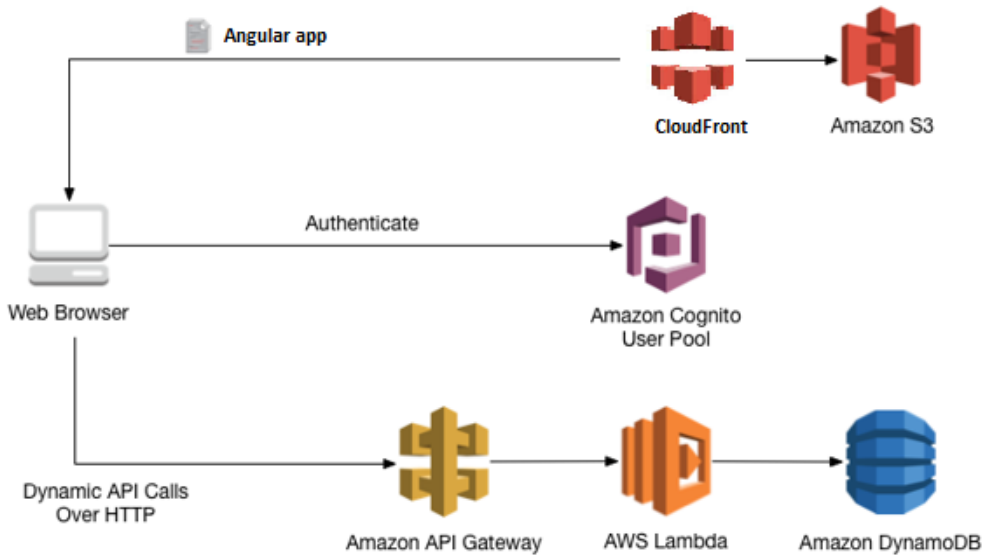


Figura 4.11: Architettura

* La pagina di login viene caricato nel browser;
* L'utente inserisce le sue credenziali corrette per effetuare il login;
* AWS Cognito verifica le credenziali, se sono valide ritorna il token di accesso altrimenti un errore;
* Le credenziali sono valide, l'uente accede nella pagina degli amministratori;
* L'utente ora può communicare con il API Gateway inviando il token ritornato da AWS Cognito. Questo token permette all'utente di inserire oppure eliminare uenti dal database nosql presente su AWS;
* Inserimento oppure l'eliminazione di un cliente esegue stesso flusso visto sopra per il template. Tranne per il AWS Cognito la pagina degli ammministraoti 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 del utente Nextep con la pagina degli amministratori.

5.1 Editor

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

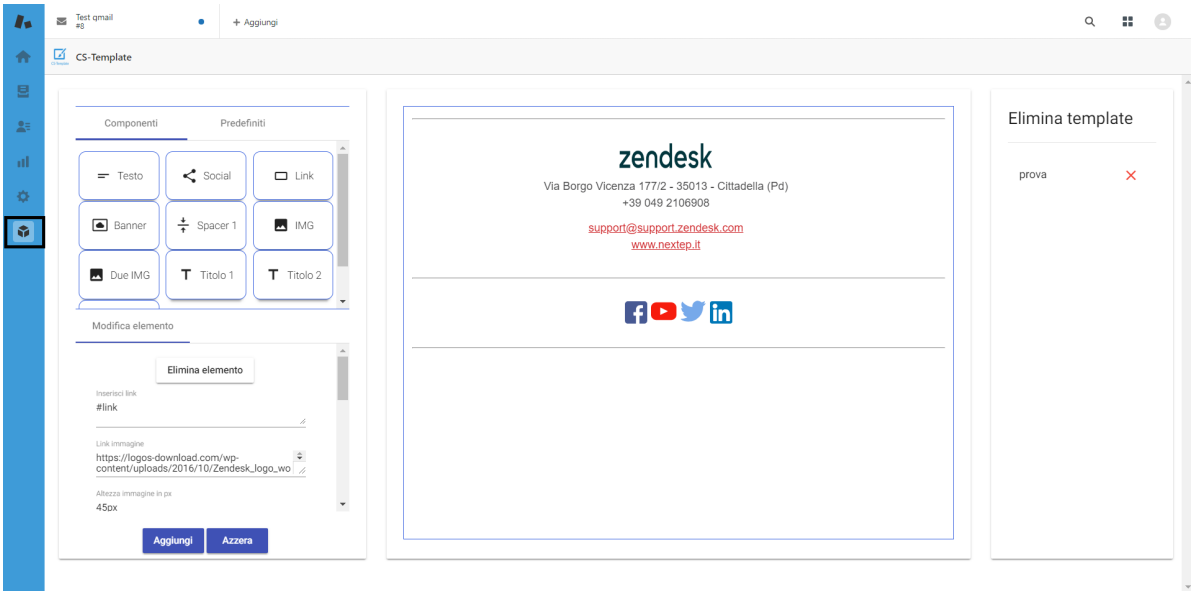


Figura 5.1: Editor realizzato

- 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;

* Faccendo 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 azzerrare il contenuto del container semplicemente cliccando il pulsante "Azzerra";
* Una volta realizzato il template deriderato l'utente può salvarlo semplicemente cliccando il pulsante "Aggiungi". Verrà chiesto all'utente di inserire il nome del template, inserito il nome valido il template verrà automaticamente salvato sul database nosql di Amazon;
* A destra nel "Elimina template" è possibile visualizzare tutti template realizzati e se necessario eliminarli semplicemente cliccano sull'icona "X".

5.2 Widget

Il widget permette al agente di Zendesk di utilizzare i template realizzati nelle risposte ai clienti.

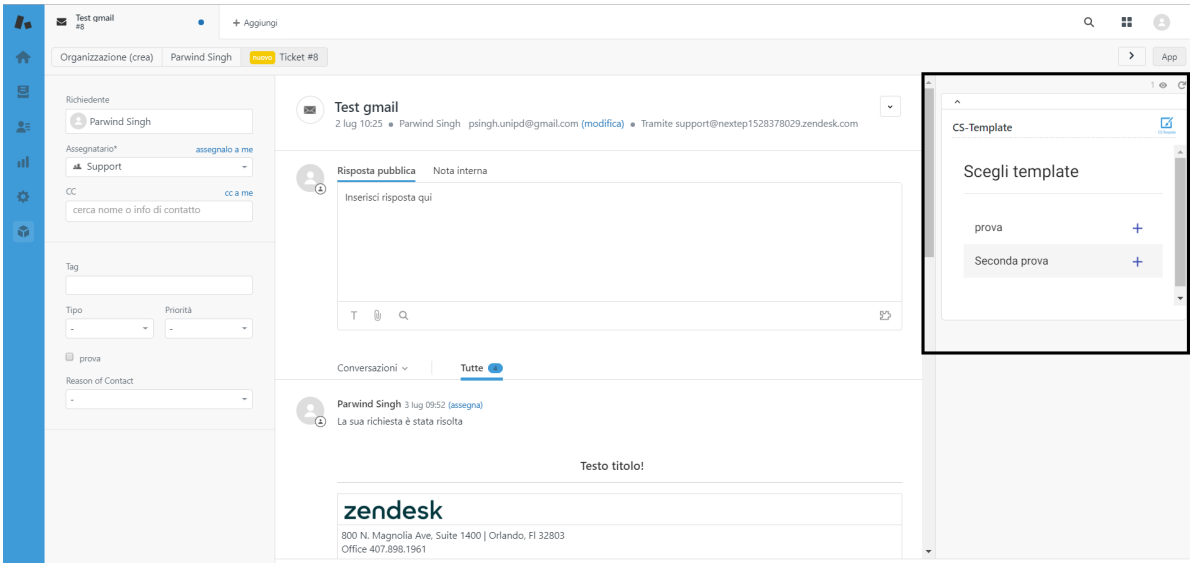


Figura 5.2: Editor realizzato

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

5.3 Login pagina amministratore

Capitolo 6

Conclusioni

6.1 Consuntivo finale

6.2 Raggiungimento degli obiettivi

6.3 Conoscenze acquisite

6.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia