## Architecture Overview

L’architettura della soluzione proposta è **coerente con quella desiderata da Aruba e descritta nel capitolato**. La stessa potrebbe essere oggetto di revisione in fase di design del progetto salvo accettazione da ambe le parti

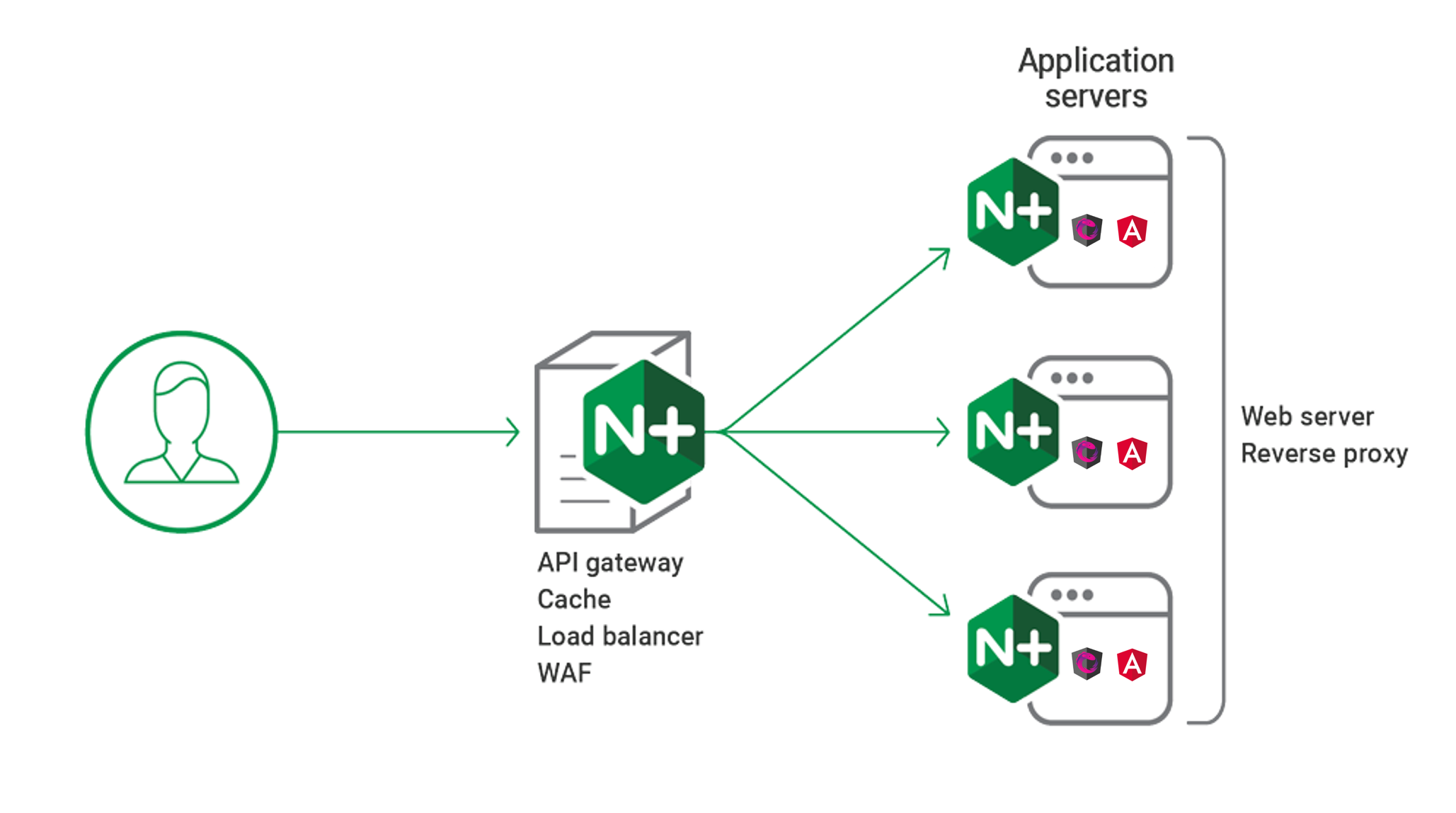


Figure 3 – Architettura dell’applicazione

L’applicazione sarà composta da diversi moduli implementati utilizzando una combinazione di framework Bootstrap, Angular.js e NgRx. Nello specifico è prevista la realizzazione di almeno un container Web Server che conterrà il componente di frontend che si interfaccerà direttamente con le API di DocFly3 tramite service Angular, escludendo in questo caso l’utilizzo della tecnologia Spring Cloud Gateway.

La gestione dei forms verrà eseguita mediante l’approccio dei Reactive Forms (RF) messi a disposizione da Angular, basati sul pattern Observable sono molto più flessibili, scalabili e robusti rispetto ai template-driven forms. I RF saranno implementati utilizzando la libreria NgRx che si occuperà della gestione dello stato.

Di seguito la descrizione dell’architettura dei componenti NgRx:

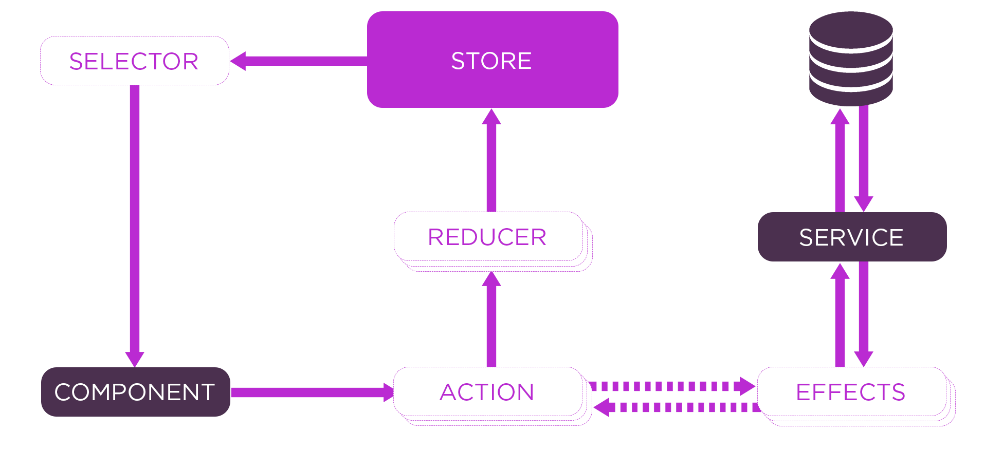


Figure 4 – NgRx – Gestione del ciclo di vita dello stato

* Store: è ciò che contiene lo stato dell'app.
* Action: un evento unico inviato da componenti e servizi che descrivono come modificare lo stato. Ad esempio, "Aggiungi cliente" può essere un'azione che cambierà lo stato (vale a dire aggiungere un nuovo cliente all'elenco).
* Reducer: tutti i cambiamenti di stato avvengono all'interno del Reducer; risponde all'azione e, sulla base di tale azione, creerà un nuovo stato immutabile e lo restituirà al Store.
* Selector: il selettore è una funzione utilizzata per ottenere una parte dello stato dallo Store.
* Effects: un meccanismo che ascolta le Action inviate in un flusso osservabile, elabora la risposta del server e restituisce nuove azioni immediatamente o in modo asincrono al Reducer per modificare lo stato.

## Front-End

### Tecnologie utilizzate

In accordo con le linee guida e con le richieste di Aruba, il Front End sarà un'applicazione Progressive Web Application (PWA), l’interfaccia del pannello atterrà ai mockup e alle linee guida grafico-stilistiche che verranno forniti da Aruba, così come i flussi di navigazione.

Le principali tecnologie utilizzate per l’implementazione saranno:

|  |  |
| --- | --- |
| Tecnologia | Descrizione |
| Angular 9.x | Framework Javascript per la creazione di Web Application dinamiche |
| NgRx | Libreria ispirata a Redux utilizzata per gestire lo stato applicativo in applicazioni Angular |
| Kubernetes | Software per l'automazione del deployment, scalabilità, e gestione di applicativi attraverso containers |
| Chart.js | Libreria open source per la creazione di grafici HTML |
| Bootstrap 4+ | Framework utilizzato principalmente per la creazione di siti e app responsive.  Permette di avere una migliore User Experience e tempi di sviluppo maggiori rispetto al Material Design. |
| Fontawesome | Tool utilizzato per le icone non SVG |
| Protractor | Tool utilizzato per gestire la parti di Code Coverage, Unit Test, E2E Test |
| Compodoc | Tool utilizzato per la documentazione del codice |
| Chart.js | Libreria open source per la creazione di grafici HTML |
| PM2 | Tool utilizzato per le attività di monitoring |

### Angular

La scelta di utilizzare Angular come tecnologia principale per il front-end permette di:

* avere un framework popolare e consolidato
* fornire feature come Ahead-of-Time compiling, Bundling, Lazyloading atte a garantire le performance delle componenti sviluppate;
* utilizzare TypeScript come linguaggio per garantire uno sviluppo solido e tipizzato;
* avere strumenti a supporto dello sviluppo (test e coverage) out-of-the-box;

Angular si basa principalmente su due costrutti: i components ed i services.

**Componenti**: sono i "pezzi del puzzle" e il loro assemblaggio costruisce una pagina web. Sono dei blocchi di codice HTML estesi, riempiti, modificati e potenziati da codice TypeScript, con la responsabilità di visualizzare i dati ricevuti dal server e gestire correttamente le interazioni con l’utente..

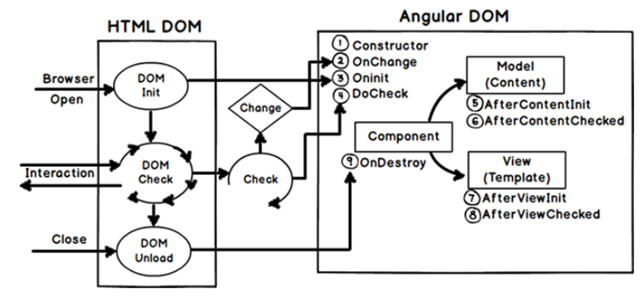


Figure 3 - Ciclo di vita di un component ed interazione con il DOM

**Servizi**: un service è un insieme di funzionalità di cui l’applicazione necessita. È utilizzato per il reperimento dei dati dal server, la gestione degli errori, lo scambio di dati tra component.

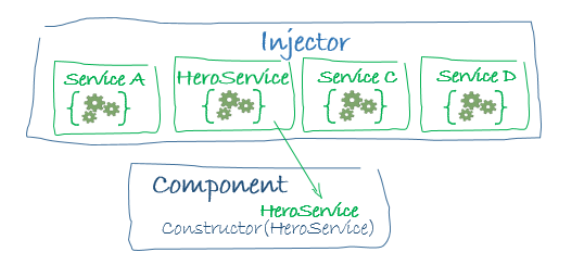
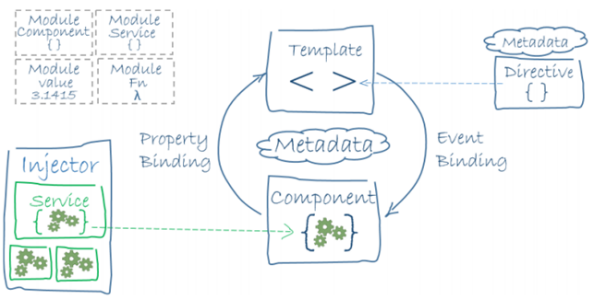


Figure 4 - Funzionamento di un component di Angular con servizio iniettato

Angular segue molti dei principi del Model-View-ViewModel. Questo design pattern è nato per migliorare la leggibilità e la manutenibilità del codice, dividendolo nelle tre parti menzionate nel nome. Uno schema della relazione tra le parti di tale pattern è visibile in figura 6.

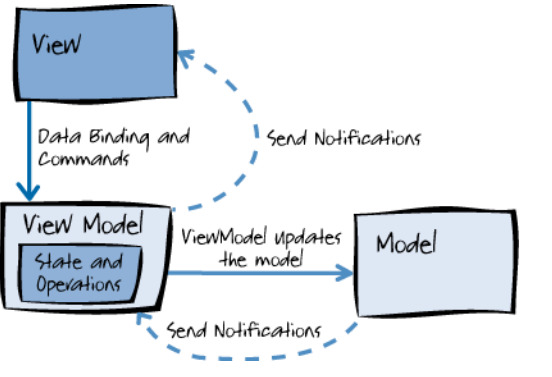


Figure 6 - Schema del design pattern Model-View-ViewModel

Le applicazioni web scritte con Angular sono definite anche **Single Page Application**, perché, come suggerisce il nome, vengono eseguite interamente in una sola pagina su browser.

L’utilizzo di tale pattern permetterà di avere migliori performance in termini soprattutto di velocità e scalabilità, migliorando drasticamente l’esperienza di navigazione dell’utente.

### Nginx

Nginx è un HTTP web server che può essere usato anche come reverse proxy, mail server, load balancer e server TCP/UDP. Tra le sue caratteristiche spiccano una ottimizzata gestione delle richieste verso file statici, un supporto per applicazioni Fast Common Gateway Interface (FastCGI), gestire il protocollo SSL e reindirizzare il traffico verso l'applicazione Node.js che contiene i file JavaScript dell’applicazione Angular.

L’estrema semplicità che contraddistingue il suo processo di configurazione è sicuramente uno dei suoi punti di forza.

Nel caso delle PWA, Nginx si occupa di indirizzare le richieste al server che fornisce i file statici generati da Angular, ed al server che fornisce l’API di back end.

### Storybook

Lo Storybook è un ambiente di sviluppo dell'interfaccia utente eseguito al di fuori della PWA principale in modo che gli utenti possano sviluppare i componenti dell'interfaccia utente in modo indipendente senza preoccuparsi delle dipendenze e dei requisiti specifici.

Lo Storybook sarà utilizzato per far fronte alle esigenze dell’approccio TDD in fase di sviluppo e per la generazione della documentazione tecnica dei componenti.

## Sicurezza

Il progetto verrà sviluppato seguendo le linee guida contenute all’interno dell’Allegato Linee guida sviluppo sicuro.pdf

## Scalabilità e Business Continuity

Grazie all’architettura proposta e messa a disposizione da Aruba (deploy su piattaforma Kubernetes con l'Horizontal Pod Autoscaler) i moduli sviluppati potranno essere istanziati in maniera indipendente sulla base del carico di lavoro richiesto, tale approccio garantirà la scalabilità del software.

## Performance minime

L’architettura e l’implementazione del sistema, garantiscono continuità di performance al crescere della richiesta elaborativa che saranno misurate e garantite dalla piattaforma WebPageTest.org.

## Monitoring, Logging, Tracing

PM2 è il più noto manager di processi per Node.js (framework per applicazioni JS lato server) utilizzato per eseguire e gestire i carichi di lavoro è anche un ottimo strumento di monitoring da CLI o da interfaccia Web, fornisce funzionalità come:

* Gestione di più applicazioni NodeJS su un singolo server
* Gestione del registro degli errori e delle applicazioni
* Integrazione dell'interfaccia Web per il monitoraggio dell'integrità e delle metriche dell'applicazione
* Clustering automatico per applicazioni NodeJS
* Ricarica a caldo dell'applicazione
* Supporto per script di avvio
* Registra lo streaming su un'interfaccia web