Alma Mater Studiorum · Università di Bologna

Corso di Laurea in Informatica per il Management

Analisi delle principali librerie per la generazione di grafici in React, Angular e Vue.js

Prof: Angelo Di Iorio

Presentata da:
Alessandro Modelli

Seconda Sessione

2022/2023

Indice

INTRODUZIONE

Alla base di questo studio vi è l'analisi di alcune delle principali librerie di grafici dei framework React, Angular e Vue.js.

In particolare, si pone l'attenzione sui vantaggi e svantaggi di ciascuna libreria in termini di semplicità di utilizzo, funzionalità offerte e compatibilità e prestazione con il framework utilizzato, poiché questi aspetti sono fondamentali per garantire un'esperienza utente fluida.

Una libreria di grafici è uno strumento software che, attraverso diverse librerie JavaScript, fornisce allo sviluppatore, componenti predefiniti e personalizzabili che semplificano la creazione e l'integrazione di grafici in un'applicazione.

Le motivazioni che mi hanno spinto ad approfondire tale tema sono incentrate principalmente attorno all'importanza della scelta da effettuare nel momento in cui è richiesta la generazione di grafici all'interno di un'applicazione; scegliere la giusta libreria da integrare in un progetto porta un'elevata conseguente riduzione del tempo richiesto per lo sviluppo. Dopo essermi dedicato allo studio della documentazione di ciascuna libreria, l'utilizzo di ciascuna di esse ha rappresentato la base su cui ho fondato la mia analisi.

L'obiettivo di questa tesi è quello di fornire un'analisi accurata per ciascuna libreria attraverso l'integrazione e il confronto delle diverse tipologie di grafici per ciascuna di esse, nel caso di React facendo uso di un dataset esterno, mentre nel caso di Angular e Vue utilizzando dati predefiniti in una struttura dati ad array, vedi *2.3: Dati di test*. I dati selezionati, per React, provengono da un dataset scaricato dalla piattaforma Kaggle e riguarda prezzi e informazioni sulle abitazioni in California.

La tesi è articolata in cinque capitoli: nel primo capitolo si tratta il concetto di frameworks, con una descrizione dettagliata delle tecnologie utilizzate, della struttura e del funzionamento. Il secondo capito punta ad illustrare una panoramica dell'analisi e degli strumenti utilizzati. Il terzo capitolo, invece, tratta l'analisi di tre librerie di grafici di React, con descrizione generale, vantaggi e svantaggi di ciascuna di esse e il processo di utilizzo. Con il quarto e quinto capitolo si pone l'attenzione sui framework Angular e Vue.js, con tre librerie di grafici per ciascuno, la struttura di questo capitolo rispecchia quella del precedente, di conseguenza verrà descritta ciascuna libreria con la medesima modalità. Infine, vengono commentati e confrontati i risultati ottenuti dalle analisi svolte, esponendo le caratteristiche che portano alla scelta di ciascuna libreria. Grazie a questo lavoro è stato possibile confrontare ciascuna libreria nei minimi dettagli così da capire qual è la scelta migliore da utilizzare per il proprio progetto.

CAPITOLO PRIMO – FRAMEWORK PER LO SVILUPPO WEB

Quando si decide di sviluppare un'applicazione, solitamente si hanno due possibilità, si può scegliere di scrivere il proprio codice completamente da zero, oppure si può scegliere di utilizzare strumenti e tecnologie già esistenti da assemblare tra loro per costruire il proprio progetto riducendo il tempo di sviluppo. Un framework, nell'ambito dello sviluppo Web, è una struttura di supporto per lo sviluppo software costituito da strumenti, librerie e componenti predefiniti progettati per velocizzare e rendere più semplice lo sviluppo di applicazioni di maggiore complessità. Offrono una struttura di partenza solida e conveniente per lo sviluppo di applicazioni web, consentendo agli sviluppatori di dedicarsi alla logica dell'applicazione e alle funzionalità specifiche anziché dover affrontare problematiche di base già risolte.

1.1 Caratteristiche principali e funzionalità

Ciascun framework per applicazioni Web è basato su alcuni aspetti principali che ne caratterizzano la semplificazione nell'utilizzo.
Generalmente sono costituiti da una struttura organizzativa predefinita basata sui modelli di sviluppo MVC (Model-View-Controller) o MVVM (Model-View-ViewModel). In questo modo semplificano il lavoro degli sviluppatori nel separare le differenti parti che si occupano della gestione della logica dei dati, della presentazione della grafica e del controllo delle interazioni dell'utente. In questo modo l'applicazione risulta più organizzata e di conseguenza la ricerca di risorse più rapida. L'utilizzo di componenti riutilizzabili è un altro degli aspetti principali di un framework; consentire agli sviluppatori di creare componenti che possono essere riutilizzati per l'interfaccia utente (GUI) riduce il tempo di sviluppo poiché evita di dover riscrivere codice utilizzato in precedenza. Un altro aspetto molto importante è la community e la quantità di risorse accessibili agli sviluppatori. Un framework con una grande comunità di programmatori che condividono le proprie risorse e soluzioni ai problemi che si riscontrano maggiormente, permette a tutti gli altri sviluppatori di utilizzare soluzioni già esistenti senza dover perdere ulteriore tempo e deviando l'attenzione dal problema principale.
Al giorno d'oggi esistono diversi tipi di framework per applicazioni web, che utilizzano linguaggi di programmazione differenti come JavaScript, Php, Java, Python e tanti altri. In questa analisi andremo ad affrontare tre tipologie di framework, molto simili tra loro, che utilizzano JavaScript: React, Angular e Vue.js.

1.2 React

Attualmente, React è considerato uno dei principali framework impiegati nello sviluppo di applicazioni web. Piu dettagliatamente però, React è una libreria open-source creata da Facebook, per lo sviluppo di interfacce utente di tipo dinamico e reattivo basata sul concetto di "componenti", ovvero blocchi di codice che rappresentano elementi specifici dell'interfaccia e che possono essere riutilizzati.

A rendere React particolarmente potente è il suo approccio dichiarativo alla costruzione delle interfacce. Una volta che viene definito lo stato attuale dell'interfaccia, esso si occupa automaticamente di gestire gli aggiornamenti necessari quando lo stato cambia.

Al fine di ottimizzare le prestazioni, React utilizza il concetto di "DOM Virtuale". Invece di aggiornare direttamente il DOM (Document Object Model) al cambio dello stato, crea una rappresentazione virtuale dell'interfaccia utente. Quando si verifica un cambiamento dello stato, React effettua un confronto tra il DOM e la rappresentazione virtuale e successivamente aggiorna solo le parti necessarie.

In altre parole, permette di aggiornare ciascun singolo componente al cambio dello stato, senza dover aggiornare l'intera pagina. Questo rende il codice più prevedibile, facile da testare e mantenibile.

Sebbene React sia la libreria principale, molti sviluppatori affiancano React con altre librerie e strumenti per creare applicazioni complete. Una delle principali è, ad esempio, React Router, che viene spesso utilizzata per gestire le route, anche dette "percorsi", dell'applicazione.

Ricapitolando, React è una libreria JavaScript potente e popolare che semplifica lo sviluppo delle interfacce utente, contribuendo a ridimensionare e facilitare la progettazione delle applicazioni web moderne.

1.3 Angular

Angular è l'evoluzione di AngularJS, ed è un framework flessibile di sviluppo web creato da Google attualmente molto in uso per lo sviluppo front-end per interfacce dinamiche ed efficienti. Il linguaggio principalmente utilizzato per lo sviluppo è TypeScript, una variante di JavaScript tipizzata. Anch'esso è un framework con una struttura basata su componenti che possono essere riutilizzati e che rappresentano parti specifiche dell'interfaccia utente semplificando la gestione del codice e agevolando lo sviluppo in team.
Una delle caratteristiche principali che differenziano Angular dagli altri framework è il Two-Way Data Binding, ovvero Angular sincronizza automaticamente i dati tra il modello dell'applicazione e l'interfaccia utente. In questo modo le modifiche fatte nei dati vengono riflesse immediatamente nell'interfaccia e viceversa, rendendo le interazioni utente fluide e reattive. Un'altra delle principali caratteristiche è il Dependency Injection ovvero la possibilità di passare le dipendenze da un componendo all'altro semplificandone la gestione e rendendo il codice più modulare e scalabile.
Per quanto riguarda la navigazione, Angular offre un sistema di Routing già implementato che permette la navigazione tra diverse pagine.
Anche in questo caso, questo framework rileva automaticamente i cambiamenti nello stato e aggiorna solo le parti dell'interfaccia che sono state effettivamente modificate, migliorando l'efficienza dell'applicazione. Nonostante Angular possieda già un proprio sistema per la gestione degli stati spesso viene affiancato ad altre librerie come Redux per gestire lo stato globale dell'applicazione in modo efficiente.
Angular è diventato uno dei framework più popolari per lo sviluppo di applicazioni web moderne. Grazie alla sua architettura basata su componenti, al two-way data binding e ad altri strumenti integrati, Angular semplifica la creazione di interfacce utente dinamiche, reattive e performanti, rendendolo una scelta preferita per gli sviluppatori di tutto il mondo.

1.4 Vue.js

Vue.js è un framework JavaScript open-source che ha ottenuto rapidamente popolarità grazie alla sua flessibilità, semplicità ed eleganza nello sviluppo web.
Questo framework è stato progettato per agevolare la creazione di interfacce utente dinamiche e interattive, in modo da semplificare lo sviluppo di applicazioni web più sofisticate.
Vue.js è basato su un'architettura a componenti. Anche in questo framework l'interfaccia utente è suddivisa in componenti modulari e riutilizzabili, in questo modo favorisce una maggiore chiarezza della complessità delle interfacce e permette di avere una struttura modulare e scalabile.
Una delle caratteristiche che denota questo framework è la sua reattività dichiarativa. Permette, infatti, agli sviluppatori di dichiarare l'influenza dei dati sull'interfaccia utente e, in autonomia, esso si occupa di mantenere l'interfaccia sincronizzata con lo stato dei dati.
Anche in questo framework è presente il two-way data binding che permette di sincronizzare automaticamente i dati tra il modello e la vista, in questo modo le modifiche apportate all'interfaccia vengono riflesse nel modello e viceversa.
Per quanto riguarda il routing, Vue.js è dotato di una router ufficiale chiamato Vue Router che permette la gestione della navigazione tra le diverse pagine.
La gestione degli stati dell'applicazione, viene anch'essa gestita internamente al framework attraverso Vuex, ovvero uno stato di gestione centralizzato basato sui concetti di Redux e che consente di gestire lo stato globale dell'applicazione in modo prevedibile e scalabile.
Un'altra caratteristica principale di Vue.js è la progressività, è un framework progressivo e in quanto tale, offre la possibilità di essere utilizzato anche solo in alcuni punti di applicazioni già esistenti.
Vue.js ha guadagnato una rapida e grande popolarità soprattutto grazie alla sua semplicità di apprendimento, alla sua reattività dichiarativa e alla sua flessibilità. Grazie alle sue caratteristiche si presenta come un'ottima opzione per creare interfacce web reattive e moderne senza dover conoscere diversi complessi concetti di sviluppo.

CAPITOLO SECONDO – LIBRERIE PER LA GENERAZIONI DI GRAFICI: PANORAMA E CONFRONTO

2.1 Librerie di grafici

Con l'aumentare dell'importanza dei processi di raccolta e analisi dei dati, la necessità di visualizzare questi ultimi, attraverso grafici e/o tabelle, è diventata un componente critico per ciascuna organizzazione o azienda. Per deviare a questa necessità, la maggior parte di esse, richiede agli sviluppatori di creare applicazioni personalizzate che facilitino la lettura dei dati. Questo ha portato allo sviluppo di librerie di grafici interattivi personalizzabili dagli sviluppatori.

Le librerie di grafici per applicazioni web sono strumenti software progettati per semplificare la creazione di grafici interattivi e visivamente accattivanti all'interno delle applicazioni web. Queste librerie forniscono un insieme di componenti e funzionalità predefinite che consentono agli sviluppatori di rappresentare i dati di interesse, rendendo più facile per gli utenti comprenderne il significato. Diversi settori richiedono l'utilizzo di grafici, come il settore scientifico, quello di business, data science e tanti altri, ed è proprio qui che prevale l'importanza delle librerie di grafici con i propri vantaggi e svantaggi.

L'integrazione di grafici all'interno della propria applicazione significa offrire all'utente che la utilizza la possibilità di comprendere più intuitivamente i dati di interesse, piuttosto che doverli interpretare da una complessa tabella descrittiva. Inoltre, permette di visualizzare e comprendere rapidamente l'andamento dei dati stessi.

La maggior parte delle librerie utilizzate offre la possibilità di rendere i propri grafici interattivi, questo consente agli utenti di esplorare i dati, ingrandire specifiche aree dei grafici, attivare/disattivare serie di dati e altro ancora, incrementando il coinvolgimento nell'attività di lettura. La personalizzazione dei grafici e la loro consistenza sono altri aspetti di grande importanza nell'utilizzo delle librerie. Esse offrono la possibilità di personalizzare colori, stili, etichette, puntatori e tanto altro, consentendo di adattare i grafici allo stile dell'applicazione o alle richieste del cliente e mantenere tale stile per ciascun grafico.

Uno dei vantaggi principali nell'utilizzo di queste librerie è l'elevata riduzione del tempo richiesto per la generazione dei grafici, specialmente quando si presentano di elevata complessità. Questo è possibile poiché gli sviluppatori possono implementare le funzionalità

predefinite invece di dovere scrivere il codice completamente da zero. Come tutte, o gran parte, delle tecnologie, oltre ai vantaggi si presentano anche alcuni svantaggi.
La complessità di apprendimento da parte degli sviluppatori per l'utilizzo è uno di quelli. Alcune librerie possono presentarsi complesse da utilizzare, soprattutto se ne viene richiesto l'utilizzo di funzionalità avanzate, di conseguenza aumenta il tempo sviluppo, in quanto, lo sviluppatore è costretto ad imparare ad utilizzare efficacemente la libreria. Un ulteriore svantaggio che può presentarsi è la personalizzazione limitata. Alcune librerie non presentano un'elevata flessibilità a livello di personalizzazione, di conseguenza lo sviluppatore è costretto ad analizzare la coerenza di stile della libreria con lo stile generale dell'applicazione, ed in caso non siano esattamente compatibili, per mantenere uno stile coerente, è costretto a cambiare libreria. Librerie complesse ed elevatamente interattive, possono influire sulle prestazioni dell'applicazione, in quanto è possibile che le risorse grafiche e gli script richiesti per le funzionalità possano aggiungere peso ai file del progetto causando rallentamenti generali e altre problematiche. Un importante aspetto da analizzare durante la scelta della libreria da utilizzare è verificare la compatibilità di quest'ultima con i browser utilizzati dagli utenti, che siano i dipendenti dell'azienda che richiede l'applicazione o utenti generici. La scelta di una libreria compatibile con solo alcuni browser potrebbe portare ad un ulteriore adattamento futuro conforme a tutte le piattaforme. Un altro possibile svantaggio che può verificarsi è la creazione di dipendenze esterne dovute all'utilizzo di alcune librerie. Integrare librerie utilizzate da piccole community, che non vengono mantenute e aggiornate o che rischiano di non essere più supportate, potrebbe portare alla completa ristrutturazione dell'applicazione, in quanto è necessario utilizzare una nuova libreria e, di conseguenza, riscrivere il codice di tutti i grafici utilizzati all'interno del progetto.
In generale, le librerie di grafici sono strumenti preziosi per arricchire le applicazioni web con visualizzazioni dei dati coinvolgenti. Per quanto riguarda la gestione degli eventuali svantaggi, questi ultimi possono essere contenuti scegliendo attentamente la libreria da utilizzare, valutando le esigenze richieste dall'applicazione e verificando che la libreria, che si intende utilizzare, abbia una buona documentazione e un opportuno supporto dalla community di sviluppatori.
In questa analisi sono state selezionate alcune delle principali librerie per i tre framework scelti con la consultazione di alcuni siti accessibili dalla bibliografia.
Nel caso di React sono state selezionate le seguenti librerie: Recharts, Victory, React-chartjs-2. Per quanto riguarda Angular sono state selezionate: Highcharts, FusionCharts, AnyCharts

Mentre per il framework Vue.js sono state selezionate le seguenti librerie: ApexCharts, Vue-ECharts, JSCharting.

FRAMEWORK	LIBRERIA	DOCUMENTAZIONE
React	Recharts	https://recharts.org/en-US
React	Victory	https://formidable.com/open-source/victory/docs
React	React-chartjs-2	https://react-chartjs-2.js.org/
Angular	Highcharts	https://www.highcharts.com/docs/index
Angular	FusionCharts	https://www.fusioncharts.com/dev/fusioncharts
Angular	AnyCharts	https://docs.anychart.com/Quick_Start/Quick_Start
Vue	ApexCharts	https://apexcharts.com/docs/installation/
Vue	Vue-ECharts	https://vue-echarts.dev/
Vue	JSCharting	https://jscharting.com/tutorials/creating-js-charts/

Tabella 1: Tabella Indicativa delle librerie di grafici utilizzate

Non tutte le librerie prese in analisi sono strettamente specifiche per il framework in cui sono state utilizzate. Alcune librerie come Highcharts, FusionCharts, AnyCharts, ApexCharts e JSCharting sono librerie compatibili con tutti e tre i framework in questione; in questo caso, il framework in cui sono state integrate è indicato nella *Tabella 2*. Ci sono poi alcune librerie che sono specifiche per alcuni di essi, come Recharts, Victory e React-chartjs-2 che sono specifiche per React e Vue-ECharts che è prettamente specifica per Vue.

2.2 Parametri di confronto

Partendo dal presupposto che tutte le librerie in questione hanno lo scopo di implementare grafici, per poter effettuare un'analisi dettagliata per ciascuna di esse è necessario andare a valutare alcuni parametri predefiniti. Con questo studio si va ad analizzare le diverse caratteristiche e funzionalità per ciascuna di esse in modo da ottenere un resoconto dettagliato di ciò che contraddistingue una libreria da un'altra.

Ciascuna libreria è stata valutata secondo i seguenti parametri:

- **Compatibilità con i framework React, Angular, Vue.js**: In questo modo è possibile verificare la compatibilità di ciascuna libreria con i diversi framework, così da avere una maggiore scelta in caso di compatibilità multipla
- **Architettura basata su componenti**: Applicazioni sviluppate con un'architettura basata su componenti, agevolano l'utilizzo di librerie con la medesima struttura
- **Responsività**: Grafici che si adattano automaticamente alle dimensioni del contenitore agevolano l'integrazione di rappresentazioni di dati in applicazioni disponibili per dispositivi di dimensioni diverse
- **Tipologie di grafici**
    - **Semplici:** Grafici principali come grafici a linee, a barre, a torta, ad area, a donut, a radar, scatter, funnel, a candela, composizioni di grafici e altre tipologie
    - **Avanzati:** Grafici specifici come diagrammi, organigrammi, Gantt, mappe di calore, mappe geografiche e tanti altri
    - **Tridimensionali:** Grafici tridimensionali di diverse tipologie tra cui le principali
- **Personalizzazione**
    - **Semplice:** Personalizzazione di titoli, etichette, colori, legenda e tooltip
    - **Avanzata:** Personalizzazione a 360° con simboli, animazioni e tanto altro
- **Interattività:**
    - **Semplice:** Funzionalità di base come dettagli al passaggio del mouse, eventi al click del mouse
    - **Avanzata:** Funzionalità avanzate per una maggiore analisi dei dati, come zoom, selezione dei dati, attivazione/disattivazione serie di dati e tanto altro

- **Supporto a dati complessi:** Supporto per la rappresentazione di dati complessi come dati temporali e multidimensionali
- **Supporto ai Big Data:** Supporto per la rappresentazione di grandi quantità di dati in modo chiaro e comprensivo
- **Rappresentazione SVG:** Rappresentazione dei grafici in formato SVG (Scalable Vector Graphics), così da permettere una rappresentazione di alta qualità e scalabilità
- **Esportazione:** Funzionalità di esportazione dei grafici in formati diversi, tra cui, PNG, JPG, PDF, SVG, etc.
- **Compatibilità con versione mobile:** Compatibilità dei grafici applicazioni mobile tra cui iOS e Android
- **Licenza per uso**
    - **A pagamento:** Licenza a pagamento per uso commerciale e accesso completo alle funzionalità dei grafici
    - **Versione gratuita:** Licenza gratuita per uso non commerciale e possibilità di funzionalità ridotte
- **Documentazione**
    - **Completa:** Documentazione altamente dettagliata e ricca di esempi che facilitano l'utilizzo e la personalizzazione
    - **Scadente:** Documentazione poco chiara e a volte confusionale con pochi esempi di utilizzo e personalizzazioni
- **Community:** Community di sviluppatori che condividono i problemi riscontrati e le eventuali soluzioni trovate
- **Facilità di apprendimento:** Facilità nell'apprendere, dalla documentazione, il funzionamento e le diverse funzionalità messe a disposizione.
- **Facilità di utilizzo:** Facilità nell'integrare i grafici disponibili con diverse personalizzazioni

Per poter rappresentare un quadro generale dei parametri descritti sopra, è stata creata una tabella descrittiva per riassumere le caratteristiche di ciascuna libreria.

2.3 Dati di test

Nel paragrafo precedente sono stati indicati i parametri di confronto di questa analisi. In questo paragrafo, invece, vengono illustrati i dati utilizzati per le rappresentazioni grafiche.
Come accennato nell'introduzione, per l'utilizzo delle librerie per React è stato utilizzato un dataset di 20000 righe in formato *.csv* scaricato dalla piattaforma online Kaggle[1].
Questo dataset tratta i prezzi medi delle abitazioni in California dal 1990. Il suo contenuto è strutturato su diverse colonne rappresentanti diverse informazioni, tra cui:

- Longitudine
- Latitudine
- Età media dell'abitazione
- Numero totale di stanze
- Numero totale di camere da letto
- Popolazione
- Famiglie
- Reddito medio
- Valore medio dell'abitazione
- Prossimità all'oceano

L'integrazione dei grafici viene fatta sulla base di alcuni di questi valori rispetto al valore medio dell'abitazione. Un esempio è la rappresentazione dell'età media dell'abitazione rispetto al valore medio dell'abitazione. Per ciascun grafico, di ciascuna tipologia di ogni libreria, vengono selezionate un certo numero predefinito di righe del dataset, ad esempio: il primo grafico utilizza 40 righe, il secondo 60, il terzo 110 e il quarto 10. In questo modo è possibile avere rappresentazioni più o meno popolate da dati. Ciascuna libreria è poi stata testata con una quantità maggiore di dati, all'incirca 2000 righe, per verificare il supporto per i big data; tale integrazione però non è stata mantenuta nel codice in modo da non compromettere le performance dell'applicazione.
Per quanto riguarda l'utilizzo delle librerie per Angular e Vue, in questi due casi, sono stati utilizzati dati statici salvati in strutture dati ad array di una decina di elementi e successivamente passati a ciascun grafico per la rappresentazione. In questi due casi si è cercato di proporre esempi di personalizzazioni differenti e di tipologie avanzate con una maggiore attenzione verso l'aspetto grafico e meno verso i dati.

Un esempio di struttura dati utilizzata nei grafici a linee è la seguente:

[1] Nugent C., Dataset California Housing Prices, https://www.kaggle.com/datasets/camnugent/california-housing-prices

```
const data = [
       { name: "Jan", value: 28 },
       { name: "Feb", value: 29 },
       { name: "Mar", value: 33 },
       { name: "Apr", value: 36 },
       { name: "May", value: 23 },
       { name: "Jun", value: 23 },
       { name: "Jul", value: 28 }
]
```

Per ciascuna libreria utilizzata sono state integrate quattro differenti varianti, con diverse personalizzazioni, per ognuna delle quattro tipologie di grafici principali, ovvero grafici a linee, a barre, a torta e ad area.

Tutti e tre i progetti alla base di questa analisi sono stati caricati in tre repository differenti pubblicate su GitLab, così da permettere l'accesso alle risorse.

Progetto	URL GitLab
React	https://gitlab.com/alessandromodelli/charts-libraries-analysis
Angular	https://gitlab.com/alessandromodelli/angular-chart-analysis
Vue.js	https://gitlab.com/alessandromodelli/vuechartanalysis

Tabella 2: Tabella dei link alle repository dei progetti su GitLab

CAPITOLO TERZO - ANALISI E CONFRONTO DELLE LIBRERIE PER REACT

Questo capitolo si propone di esaminare da vicino tre delle principali librerie di grafici disponibili per React e di confrontare più approfonditamente funzionalità, flessibilità e performance. Come accennato in precedenza, sono state selezionate come librerie da utilizzare Recharts, Victory e React-chartjs-2, cercando di esplorare i vari tipi di grafici disponibili, tra cui grafici a barre, grafici a torta e tanti altri. Durante questa analisi, verranno evidenziate le caratteristiche distintive di ciascuna libreria, come la facilità d'uso, la personalizzazione, la documentazione dettagliata e la capacità di gestire volumi significativi di dati, così da offrire un quadro chiaro delle opzioni disponibili per l'integrazione di grafici in un'applicazione. In questo caso viene utilizzato, come base di dati, il dataset proposto precedentemente su prezzi e valori riguardanti alcune abitazioni in California. I dati vengono letti dal file *housing.csv* dato in input dall'utente. Successivamente vengono memorizzati e manipolati in base ai dati che si intendono utilizzare per ciascun grafico. Arrivati a questo punto, i grafici vengono automaticamente riempiti con i dati selezionati.

Di seguito viene mostrata la tabella con i parametri di valutazione e i relativi risultati delle tre librerie utilizzate nel progetto in React.

	React			Angular			Vue		
Parametri di valutazione	Recharts	Victory	React-Chartjs-2	Highcharts	FusionCharts	AnyChart	ApexCharts	VueECharts	JSCharting
Compatibilità React	SI	SI	SI						
Compatibilità Angular	NO	NO	NO						
Compatibilità Vue	NO	NO	NO						
Architettura a componenti	SI	SI	SI						
Responsività	NO	SI	SI						
Grafici semplici	SI	SI	SI						
Grafici avanzati	NO	NO	NO						
Grafici tridimensionali	NO	NO	NO						
Personalizzazione semplice	SI	SI	SI						
Personalizzazione avanzata	NO	SI	SI						
Interattività semplice	SI	SI	SI						
Interattività avanzata	NO	SI	SI						
Supporto dati complessi	SI	SI	SI						
Supporto Big Data	SI	SI	SI						
Rappresentazione SVG	SI	SI	NO						
Esportazione	NO	NO	NO						
Compatibilità mobile	SI	SI	SI						
Licenza a pagamento	NO	NO	NO						
Licenza con versione gratuita	-	-	-						
Documentazione completa	COMPLETA	COMPLETA	SCADENTE						
Community	SI	SI	SI						
Facilità di apprendimento	FACILE	FACILE	MEDIA						
Facilità di implementazione	FACILE	FACILE	MEDIA						

Tabella 3: Tabella dei parametri di confronto con i risultati delle librerie relative a React

3.1 Recharts

Recharts[2] è una libreria di grafici che semplifica la creazione di grafici interattivi e dinamici all'interno delle applicazioni React. Questa libreria si basa sulla semplicità d'uso e sulla facilità di configurazione, agevolando la creazione di strumenti per la visualizzazione di dati più accattivanti e intrattenitrici.
Recharts è una libreria che offre una grande varietà di tipologie di grafici, tra cui grafici a linee, barre, torte, aree e tanti altri. Questa sua vasta gamma di tipologie di grafici disponibili offre agli sviluppatori la possibilità di scegliere il tipo di grafico più adatto allo stile e alle esigenze dell'applicazione su cui stanno lavorando.

3.1.1 Caratteristiche principali

Una delle caratteristiche principali di questa libreria è la sua compatibilità con React. Poiché Recharts è stata progettata appositamente per React, la sua architettura la rende estremamente semplice da integrare nel proprio progetto. Non a caso, Recharts utilizza un approccio basato su componenti, esattamente come React. Questa caratteristica permette agli sviluppatori di integrare i differenti componenti di Recharts all'interno della loro applicazione senza problematiche, rendendo semplificato l'utilizzo dei grafici. I grafici hanno una caratteristica particolare, non sono reattivi di default, ovvero non si riadattano in automatico alle dimensioni del contenitore, è possibile però utilizzare un componente responsive abilitando questa funzionalità. Inoltre, sono disponibili funzionalità di interattività con i dati. Per quanto riguarda la configurazione dei grafici, Recharts offre un approccio dichiarativo. Questo permette agli sviluppatori di specificare le proprietà, che si intendono utilizzare, all'interno delle proprietà dei componenti, rendendo tutto il processo di configurazione più chiaro e comprensibile. La flessibilità, per quanto riguarda la personalizzazione dei grafici, è un'altra delle caratteristiche principali di Recharts. È possibile effettuare semplici personalizzazioni come colori, stili, scritte, etichette, assi cartesiani e altro, offrendo la possibilità di adattare al meglio i grafici allo stile o alle esigenze dell'applicazione. Un altro aspetto, molto importante, è la sua compatibilità con dati complessi. Questa libreria è in grado di gestire dati più complessi come dati temporali o multidimensionali, rendendola un'ottima opzione per applicazioni che necessitano la visualizzazione di dati di questi tipi. A renderla una delle migliori librerie per grafici è anche

[2] Recharts Group, Recharts, 2016-2023, https://recharts.org/en-US

la sua documentazione, Recharts offre una documentazione completa e ricca di esempi, da poter testare, con codice e struttura dei dati condivisi.

3.1.2 Vantaggi e Svantaggi

Questa libreria è caratterizzata da diversi vantaggi, la sua interfaccia è uno di quelli. Recharts offre un'interfaccia semplice, intuitiva e professionale per creare grafici reattivi e interattivi per applicazioni in React. È una libreria dotata di una documentazione completa e ricca di esempi dettagliati che ne agevolano l'utilizzo a partire dall'integrazione alla completa personalizzazione. Infine, la varietà di tipologie di grafici e moderata flessibilità nella personalizzazione permettono di adattare i grafici alle diverse esigenze degli sviluppatori.
Arrivati a questo punto si può osservare come questa libreria porti differenti vantaggi nel suo utilizzo, però non sempre può rivelarsi la scelta migliore. Infatti, Recharts potrebbe non essere la scelta migliore per progetti che richiedono personalizzazioni di elevata complessità o molto dettagliate, poiché questa libreria offre personalizzazioni complete, ma standardizzate nei limiti della sua architettura. Proprio per questo gli sviluppatori optano per altre librerie quando gli viene richiesta una personalizzazione a 360 gradi.

In generale però, Recharts è una libreria di grafici ben progettata e adatta per la creazione di grafici interattivi e accattivanti all'interno delle applicazioni React. La sua facilità d'uso, reattività e varietà di opzioni di personalizzazione la rendono una scelta popolare tra gli sviluppatori che cercano di integrare grafici di alta qualità nelle loro applicazioni React.

3.1.3 Utilizzo e test

L'utilizzo della libreria Recharts in un progetto in React risulta molto semplice.
In questo caso, il progetto è strutturato secondo una logica molto semplice, ciascuna libreria è stata raccolta in un proprio componente che a sua volta contiene un componente per ciascuna tipologia di grafico, ad esempio, il grafico a barre viene integrato con il componente RechartBarChart all'interno del componente Recharts con le diverse proprietà.
L'integrazione dei grafici è molto semplice ed è costruita con i componenti base della libreria Recharts, tra cui componente grafico, assi cartesiani e tooltip con piccole personalizzazioni per quanto riguarda i colori e la rappresentazione dei dati.

Di seguito viene mostrato il codice sorgente del componente del grafico a barre BarChart con alcune ulteriori possibili personalizzazioni commentate:

```
const Barchart = (props: BarChartProps) => {
    const colors = ['#7ED6FB', '#20ACD9', '#09556F', '#06152B', '#7ED6FB',
'#20ACD9', '#09556F', '#06152B', '#7ED6FB', '#20ACD9', '#09556F', '#06152B'];
    return (
        <BarChart
            width={props.widthBarC}
            height={props.heightBarC}
            data={props.data}
            margin={{
                top: props.top,
                right: props.right,
                left: props.left,
                bottom: props.bottom,
            }}
            layout={props.vertical ? "vertical" : "horizontal"}
        >
            {/* <CartesianGrid strokeDasharray="3 3" /> */}
            <CartesianGrid horizontal={true} vertical={false} opacity={0.2} />

            {props.vertical ?
                <>
                    <YAxis dataKey={"name"} type="category" />
                    <XAxis type="number" />
                </> :
                <>
                    <XAxis dataKey={"name"} />
                    <YAxis />
                    {/* <YAxis type="category" /> */}
                </>
            }
            <Tooltip />
            {/* <Legend /> */}
            <Bar dataKey="uv" barSize={props.barSize} >
                {props.data.map((entry, index) => (
                    <Cell key={`cell-${index}`} fill={colors[index % 20]} />
                ))}

            </Bar>
        </BarChart>
    );
}
```

Con un risultato finale di questa tipologia:

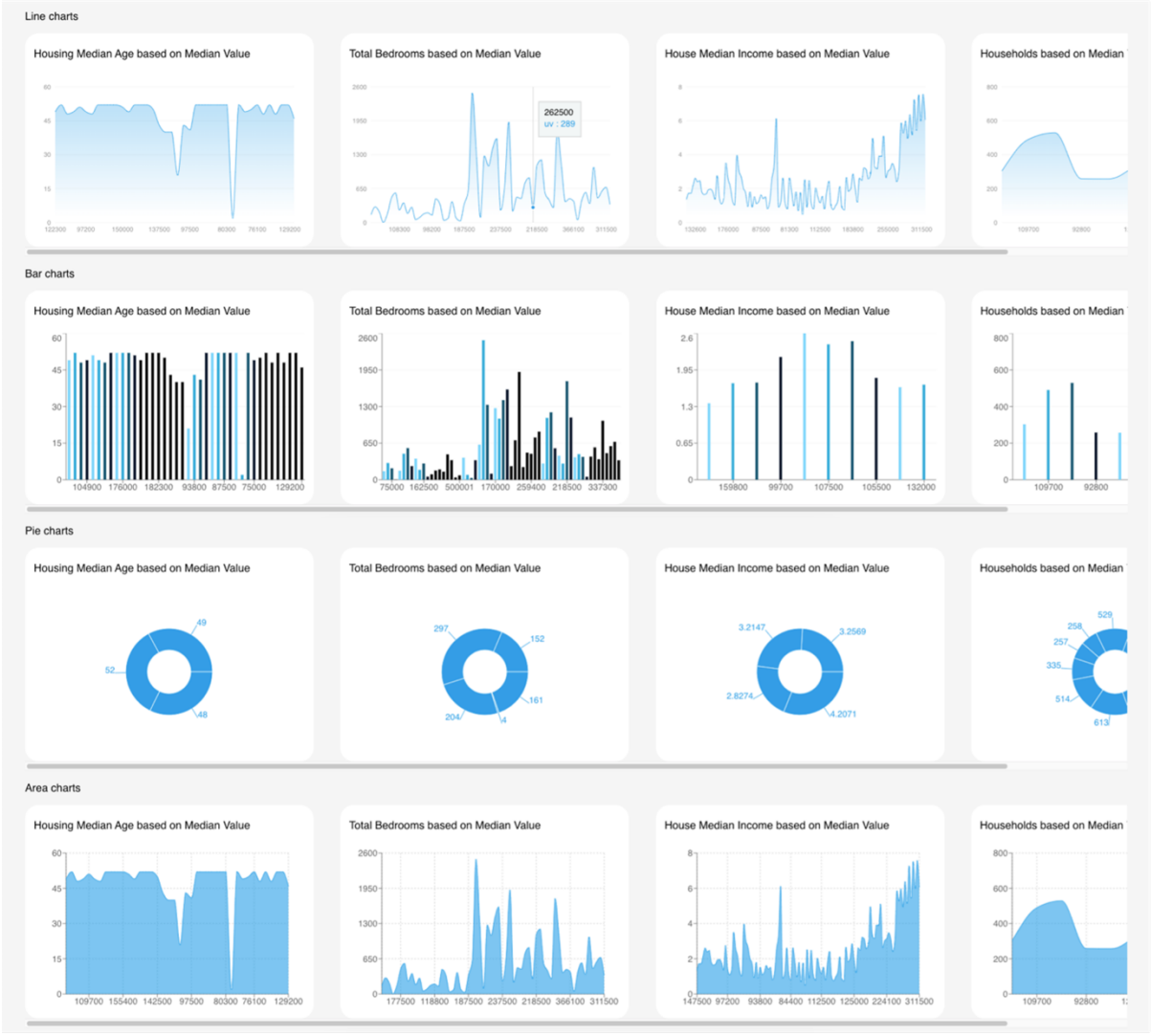

Figura 3.1.1: Risultato finale dell'utilizzo di Recharts

3.2 Victory

Victory[3] è una potente libreria di grafici progettata per l'ambiente di sviluppo React. Questa libreria offre una vasta gamma di componenti estremamente personalizzabili per la creazione di grafici accattivanti e interattivi all'interno della propria applicazione. Grazie alla sua flessibilità e alla sua architettura basata su componenti, Victory semplifica il processo di creazione di visualizzazioni di dati. È una libreria che offre diverse tipologie di grafici, tra cui grafici a linee, barre, torte, aree e tanti altri, come grafici a candela o rappresentanti errori.
Tutto questo offre la possibilità di integrare grafici con un aspetto in linea con lo stile generale dell'applicazione.

3.2.1 Caratteristiche principali

Victory, come accennato precedentemente, è una libreria che offre un'elevata flessibilità nella personalizzazione dei i propri grafici, sia per tipologia di grafico che per il suo stile.
L'ampia varietà di tipologie di grafici permette agli sviluppatori di rappresentare dati di diversi domini in modi più specifici. Ciascun grafico è rappresentato da un proprio componente React importabile dalla libreria. Questo permette di integrare più facilmente i grafici all'interno del componente di interesse, agevolando la gestione dello stato delle interazioni.
Una delle caratteristiche principali, di questa libreria, è l'interattività integrabile all'interno dei propri grafici. Ciascuno di essi può essere racchiuso in un componente *"contenitore"* che può avere funzionalità predefinite differenti o anche essere personalizzato. Esistono diverse tipologie di *"contenitori"* predefiniti tra cui, contenitori per la selezione di una porzione di grafico personalizzata o lungo gli assi cartesiani, contenitori con cursore per ispezionare le coordinate, fino ad arrivare a contenitori con funzionalità di zoom per analizzare i dati con maggior precisione. All'interno di ciascun componente grafico è a sua volta possibile inserire tooltip per rappresentare informazioni più dettagliate relative ai punti con stili e forme differenti, aggiungere animazioni per rendere la rappresentazione dei dati più coinvolgente e aggiungere etichette lungo gli assi.
Un altro degli aspetti fondamentali di questa libreria, come accennato in precedenza, è la sua personalizzazione. La possibilità di personalizzare ogni aspetto, come colori, assi cartesiani,

[3] Formidable, Victory, 2015-2023, https://formidable.com/open-source/victory/docs

simboli e sfondi in base alle proprie esigenze rende questa libreria un'ottima candidata per applicazioni che richiedono personalizzazioni avanzate. Per quanto riguarda la gestione di dati, Victory offre la possibilità di gestire e rappresentare grandi quantità e tipologie di dati, tra cui dati complessi e multidimensionali.

3.2.2 Vantaggi e Svantaggi

Victory è una libreria che offre senza dubbi differenti vantaggi. La sua semplicità di utilizzo la rende una libreria adatta anche a nuovi utenti. È una libreria dotata di una completa e dettagliata documentazione che mette a disposizione, degli sviluppatori, esempi di codice semplici e più complessi per semplificare l'apprendimento sull'utilizzo della libreria. Una community attiva caratterizza un altro dei vantaggi di questa libreria; soluzioni condivise in rete da altri sviluppatori permettono di ridurre le tempistiche di risoluzione di eventuali problemi già affrontati da altri colleghi.
Alcuni svantaggi che potrebbe presentarsi con l'utilizzo di Victory potrebbero essere relativi alla sua complessità, all'aumento delle dimensioni dell'applicazione e alla manutenzione. Utenti che hanno appena iniziato ad utilizzare React potrebbero percepire questa libreria come tecnologia complessa e quindi richiedere maggior tempo per imparare ad utilizzarla. Applicare personalizzazioni avanzate ai grafici integrati potrebbe causare un notevole aumento delle dimensioni del progetto. Infine, uno degli ultimi svantaggi che potrebbero presentarsi potrebbe essere la manutenzione dell'applicazione, utilizzando i grafici di una libreria esterna questo la rende dipendente da essa, di conseguenza, in caso di problemi alla libreria, è opportuno essere preparati ad eventuali aggiornamenti.

In generale, Victory risulta una scelta ottimale per applicazioni React che necessitano sia di personalizzazioni che di funzionalità interattive di tipo avanzato. Grazia alla sua semplicità agevola l'integrazione di grafici agli sviluppatori che hanno esperienza con React.
Rimane opportuno, comunque, analizzare i diversi vantaggi e svantaggi in corrispondenza delle esigenze del proprio progetto.

3.2.3 Utilizzo e test

L'utilizzo di Victory in un progetto React non risulta un procedimento complesso.
Le tre librerie di grafici per React sono state utilizzate nello stesso progetto, di conseguenza la struttura dell'applicazione risulta sempre la stessa così come l'integrazione dei componenti di Victory. Per questa libreria prendiamo come esempio il grafico a linee, quest'ultimo viene utilizzato nel component VictoryLineChart. In questo caso le proprietà richieste sono i dati, il titolo che si vuole dare al grafico e le eventuali etichette descrittive i dati degli assi cartesiani. L'integrazione dei grafici di Victory risulta molto simile a quella della libreria Recharts; avendo entrambe un'architettura basata su componenti, il grafico e le funzionalità interattive come tooltip e personalizzazioni, vengono integrati attraverso quest'ultimi. In questo caso il grafico a linee viene costruito all'interno di un contenitore a selezione libera che permette all'utente di selezionare la parte di grafico interessata.

Di seguito, viene mostrato il codice dello sviluppo del componente VictoryLineChart per l'integrazione del grafico a linee:

```
const VictoryLineChart = (props: VictoryChartProps) => {

    return (
        <div className={styles.chartBox}>

            <div className={styles.chartBoxSubcontainer}>

                <Label text={props.title} className={styles.chartTitleLabel} />

                <VictoryChart
                    containerComponent={<VictorySelectionContainer />}
                >
                    <VictoryLine

                        style={{
                            data: { stroke: "tomato" },
                        }}

                        labels={({ datum }) => `${props.y ? props.y : "y"}:
${datum.y}\n ${props.x ? props.x : "x"}: ${datum.x}`}

                        labelComponent={<VictoryTooltip />}

                        //data={houseMedianAgeChartsData.slice(0, 100)}
                        data={props.data}

                        // data accessor for x values
                        y="y"
                        x="x"
                    />
                </VictoryChart>
            </div>
        </div>
    )
}
```

Con un risultato di questa tipologia:

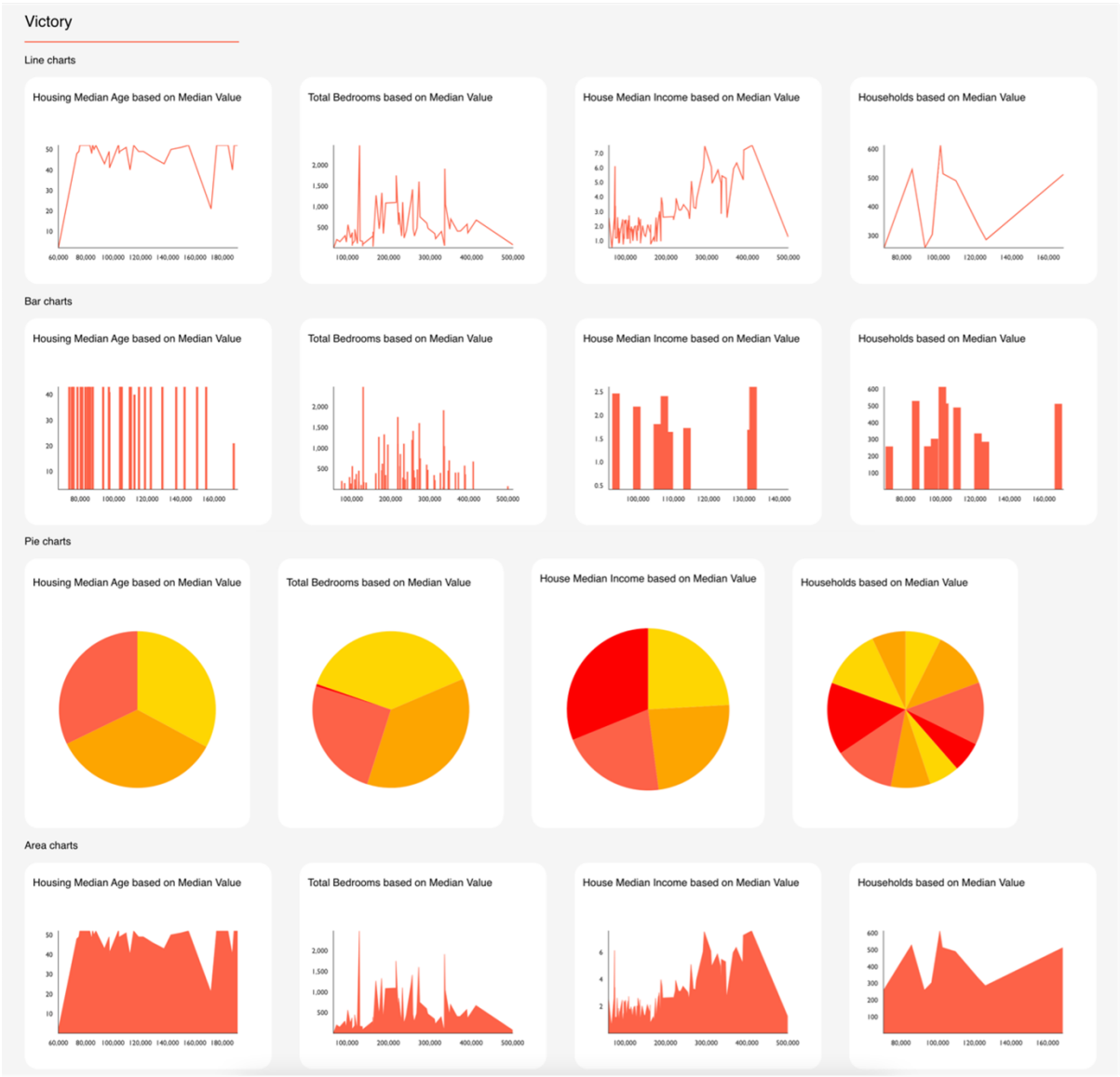

Figura 3.2.1: Risultato finale dell'utilizzo di Victory

3.3 React-chartjs-2

React-Chartjs-2[4] è una libreria per la rappresentazione di grafici, progettata per applicazioni in React, basata su Chart.js, ovvero una libreria di grafici JavaScript molto conosciuta e utilizzata dagli sviluppatori. Piu dettagliatamente, questa libreria è un wrapper di Chart.js, ovvero una libreria di avvolgimento che rende l'integrazione di quest'ultimo, in applicazioni React, più semplificata. Grazie a React-Chartjs-2 è possibile creare grafici personalizzati e interattivi di diverse tipologie, in modo chiaro e accattivante, rendendo l'esperienza utente più interessante.

3.3.1 Caratteristiche principali

React-Chartjs-2 è una libreria che mette a disposizione, degli sviluppatori, diverse tipologie di grafici da utilizzare. A partire dai più comuni e utilizzati, come grafici a linee, a barre, a torta, ad area, fino ad arrivare a grafici raggruppati, a radar, a dispersione e tanti altri.
L'architettura di questa libreria è anch'essa basata sull'utilizzo di componenti e questo la rende un'ottima candidata per progetti in React.
È una libreria che offre diverse opzioni di personalizzazione. È possibile personalizzare i colori delle serie di dati, personalizzare le legende sia a livello grafico che interattivo, personalizzare le etichette e gli assi cartesiani impostando titoli o range di dati, fino a personalizzare i tooltip per descrivere i dati in modo più dettagliato e tante altre opzioni. In questo modo viene offerta la possibilità di creare rappresentazioni dei dati con uno stile più adeguato a quello generale dell'applicazione.
L'interattività è un altro aspetto importante di questa libreria, infatti gli utenti possono interagire con i grafici attraverso eventi come clic, passaggi del mouse e altro ancora. Questo consente di creare grafici interattivi che offrono agli utenti la possibilità di analizzare i dati in modo più intuitivo.
La reattività è uno degli aspetti principali di questa libreria, grazie a questa caratteristica i grafici si riadattano in modo automatico alle dimensioni del contenitore, così da rendere possibile la rappresentazione di dati su diversi dispositivi di diverse dimensioni. React-Chartjs-2 permette di essere utilizzata in modo semplice in applicazioni React, anche se le

[4] Ayerst J., React-chartjs-2, 2020, https://react-chartjs-2.js.org/

personalizzazioni possono apparire inizialmente complesse, in quanto avvengono in modo differente dalle librerie analizzate precedentemente.
Uno degli ultimi aspetti principali di questa libreria è il supporto dato dalla sua documentazione e dalla sua community, React-Chartjs-2 è supportata da una documentazione completa e con esempi con codice condiviso per ciascuna tipologia di grafico che si intende utilizzare, in più possiede una grande community di sviluppatori che condividono i propri lavori facilitando la risoluzione di eventuali problemi riscontrati.

3.3.2 Vantaggi e Svantaggi

L'utilizzo di React-Chartjs-2 può portare diversi vantaggi visti i suoi aspetti principali. Uno di questi è dato già dal fatto che è predisposta all'integrazione in un progetto React. La sua varietà di tipologie di grafici disponibili e flessibilità nelle personalizzazioni la rende un'ottima scelta da parte degli sviluppatori. Un grande vantaggio è dato anche dalla sua funzionalità di aggiornare i dati dei grafici dinamicamente, rendendola un'ottima scelta per applicazioni che necessitano una rappresentazione di dati in tempo reale.
Infine, un vantaggio di grande importanza è la possibilità di trovare online esempi dettagliati pronti da utilizzare nel proprio progetto, riducendo eventuali tempi di sviluppo.
A questo punto è opportuno analizzare anche gli svantaggi che possono presentarsi. React-Chartjs-2 è una libreria che per quanto riguarda la personalizzazione dei suoi grafici, inizialmente può apparire alquanto complessa, motivo per il quale potrebbe presentarsi la necessità di maggior tempo per comprenderne in modo più approfondito il suo utilizzo. Allo stesso tempo, per quanto possa essere personalizzabile, potrebbe non essere la scelta migliore per progetti che necessitano grafici con personalizzazioni avanzate. Un ulteriore svantaggio che potrebbe verificarsi è la diminuzione delle prestazioni dell'applicazione dovute alla dimensione di questa libreria.

In generale, React-Chartjs-2 risulta una valida scelta per la creazione di grafici in applicazioni React, specialmente se si possiede familiarità con React e si necessita di grafici comuni come linee, barre o a torta. Tuttavia, potrebbe non rivelarsi la scelta migliore per necessità avanzate e personalizzazioni complesse, di conseguenza, in questi casi, potrebbe rivelarsi più utile valutare altre librerie.

3.3.3 Utilizzo e test

L'integrazione dei grafici di React-Chartjs-2 inizialmente potrebbe presentarsi come un'operazione più complessa rispetto alle librerie viste precedentemente.
In questo caso analizziamo l'integrazione del grafico ad area. Questa tipologia di grafico rappresenta l'area interessata dai dati utilizzati e viene integrato all'interno del componente ChartJSAreaChart. A differenza delle altre librerie utilizzate, React-Chartjs, oltre all'importazione dei componenti della libreria che si intendono utilizzare, richiede che questi componenti vengano registrati all'interno del sistema di ChartJs. Per fare ciò è necessario scrivere i componenti all'interno della funzione *Chart.register()*.
A questo punto è possibile utilizzare il grafico con le opzioni desiderate. L'integrazione del grafico avviene tramite la dichiarazione di un'oggetto relativo ai dati con le diverse proprietà del grafico e un'oggetto opzioni relativo alle diverse personalizzazioni.

Di seguito viene mostrato il codice per lo sviluppo del componente ChartJSAreaChart per l'utilizzo del grafico ad area:

```
const ChartJSAreaChart = (props: ChartJSProps) => {
  const labels = props.data.map((record) => record.name);
  const [data, setData] = useState({
    labels: labels,
    datasets: [
      {
        fill: true,
        label: 'Dataset 2',
        data: props.data.map((record) => record.uv),
        borderColor: 'rgb(53, 162, 235)',
        backgroundColor: 'rgba(53, 162, 235, 0.5)',
      },
    ]});
  useEffect(() =>{
    setData({
      labels: labels,
      datasets: [
        {
          fill: true,
          label: props.legend,
          data: props.data.map((record) => record.uv),
          borderColor: 'rgb(153, 102, 255)',
          backgroundColor: 'rgba(153, 102, 255, 0.5)',
        }]})
  }, [props.data])

  const options = {
    responsive: true,
    layout: {
      padding: 20
    }
  };
  return(
    <div className={styles.chartBox}>
    <div className={styles.chartBoxSubcontainer}>
```

```
        <Label text={props.title} className={styles.chartTitleLabel} />
        <Line data={data} options={options}/>

    </div>
    </div>
  );
};
```

Con un risultato di questa tipologia:

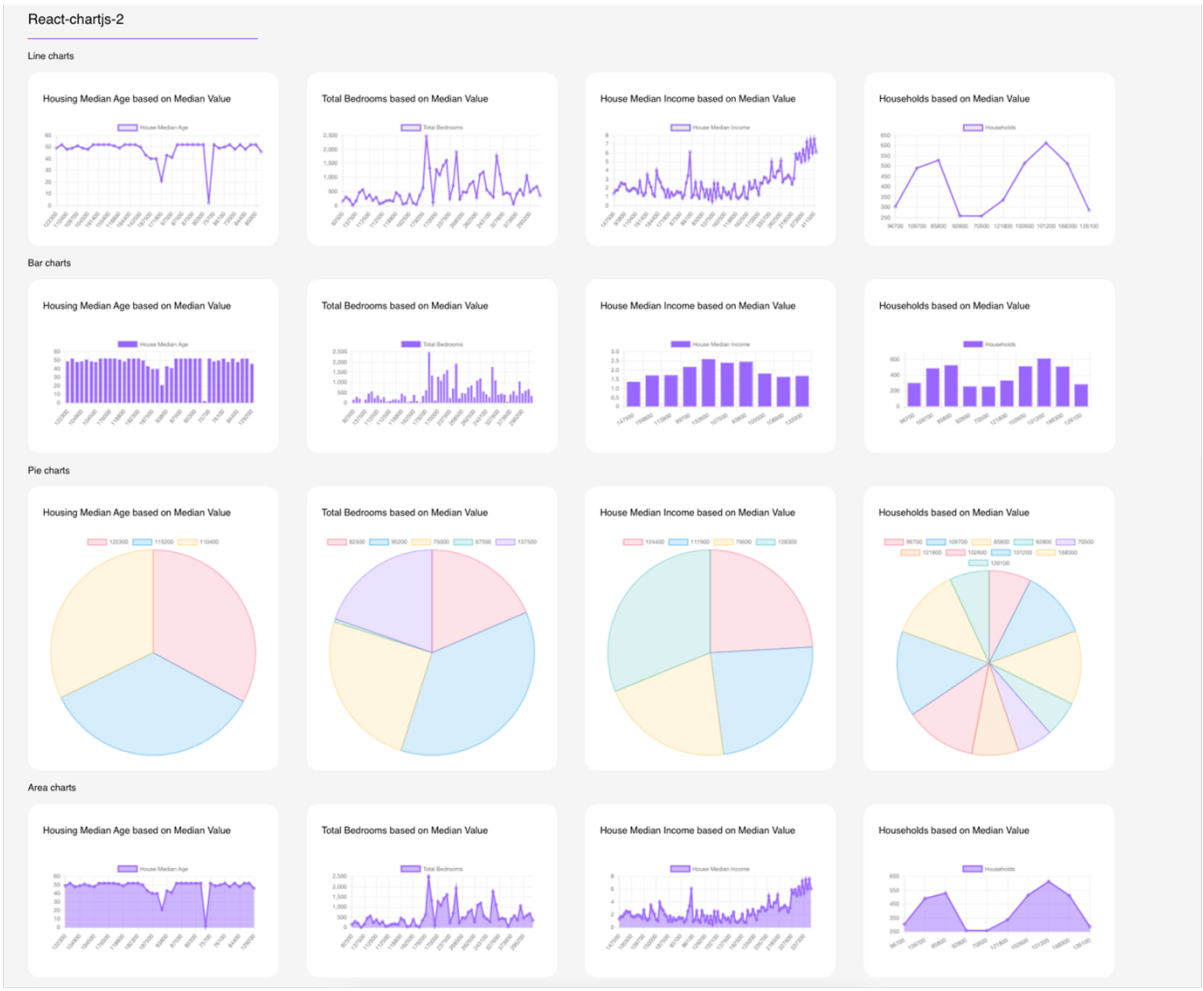

Figura 3.3.1: Risultato finale dell'utilizzo di React-chartjs-2

CAPITOLO QUARTO - ANALISI E CONFRONTO DELLE LIBRERIE PER ANGULAR

Il quarto capitolo è dedicato all'analisi di tre librerie di grafici compatibili con il framework Angular, confrontando funzionalità, flessibilità e performance per ciascuna di esse. Le librerie prese in considerazione per questa analisi sono Highcharts, FusionCharts e AnyCharts. Ciascuna libreria è stata utilizzata nel progetto alla base di questa tesi così da permettere il confronto delle differenti tipologie di grafici disponibili, cercando di evidenziare le caratteristiche principali e distintive per ciascuna di esse, tra cui personalizzazione, semplicità d'uso e tanto altro.
In questo caso i dati rappresentati sono dati predefiniti negli esempi della documentazione per ciascuna libreria, in questo modo è possibile visualizzare differenti rappresentazioni per ciascuna tipologia di grafico.

Di seguito viene mostrata la tabella con i parametri di valutazione e i relativi risultati delle tre librerie utilizzate nel progetto in Angular.

	React			Angular			Vue		
Parametri di valutazione	Recharts	Victory	React-Chartjs-2	Highcharts	FusionCharts	AnyChart	ApexCharts	VueECharts	JSCharting
Compatibilità React	SI	SI	SI	SI	SI	SI			
Compatibilità Angular	NO	NO	NO	SI	SI	SI			
Compatibilità Vue	NO	NO	NO	SI	SI	SI			
Architettura a componenti	SI	SI	SI	NO	NO	NO			
Responsività	NO	SI	SI	SI	SI	SI			
Grafici semplici	SI	SI	SI	SI	SI	SI			
Grafici avanzati	NO	NO	NO	SI	SI	SI			
Grafici tridimensionali	NO	NO	NO	SI	SI	SI			
Personalizzazione semplice	SI	SI	SI	SI	SI	SI			
Personalizzazione avanzata	NO	SI	SI	SI	SI	SI			
Interattività semplice	SI	SI	SI	SI	SI	SI			
Interattività avanzata	NO	SI	SI	SI	SI	SI			
Supporto dati complessi	SI	SI	SI	SI	SI	SI			
Supporto Big Data	SI	SI	SI	SI	SI	SI			
Rappresentazione SVG	SI	SI	NO	SI	NO	SI			
Esportazione	NO	NO	NO	SI	SI	SI			
Compatibilità mobile	SI	SI	SI	SI	SI	SI			
Licenza a pagamento	NO	NO	NO	SI	SI	SI			
Licenza con versione gratuita	-	-	-	SI	SI	SI			
Documentazione completa	COMPLETA	COMPLETA	SCADENTE	COMPLETA	COMPLETA	COMPLETA			
Community	SI	SI	SI	SI	SI	SI			
Facilità di apprendimento	FACILE	FACILE	MEDIA	FACILE	MEDIA	FACILE			
Facilità di implementazione	FACILE	FACILE	MEDIA	FACILE	MEDIA	FACILE			

Tabella 4: Tabella dei parametri di confronto con i risultati delle librerie di Angular

4.1 Highcharts

Highcharts[5] è una libreria di grafici JavaScript, open-source, sviluppata appositamente per migliorare l'interfaccia grafica di applicazioni web attraverso la rappresentazione di dati con grafici interattivi. È una libreria semplice, intuitiva e di elevata personalizzazione, che permette di creare e personalizzare grafici di diverse tipologie, tra cui i più comuni come grafici a linee, a barre, a torta fino ad arrivare a quelli più complessi. Highcharts offre, agli sviluppatori, la possibilità di utilizzare tipologie di grafici maggiormente adatte allo stile generale dell'applicazione su cui operano.

4.1.1 Caratteristiche principali

Una delle caratteristiche principali che distingue questa libreria dalle altre è l'architettura dei grafici. Highcharts, infatti, rappresenta i suoi grafici attraverso il formato SVG, ovvero grafiche vettoriali scalabili che rappresentano i dati dell'immagine come una serie di oggetti geometrici, come linee, curve, cerchi e tanto altro. È una libreria molto leggera e di conseguenza un'ottima opzione per applicazioni che necessitano di un'elevata performance.
Highcharts offre una vasta varietà di tipologie di grafici, a partire dai classici a linee, a barre e ad area fino ad arrivare a grafici 3D per una rappresentazione molto più accattivante e interattiva per l'utente. Questa libreria è caratterizzata dalla sua elevata flessibilità nella personalizzazione. Lo stile e le opzioni, di ciascun grafico, vengono, infatti, descritti attraverso un oggetto JavaScript semplicissimo da configurare. È possibile personalizzare colori, scritte, sfondi, qualsiasi tipo di asse e tanto altro. Per quando riguarda l'interazione con l'utente, i tooltip permettono di configurare azioni che vengono eseguite automaticamente quando quest'ultimo passa con il mouse sopra qualsiasi punto del grafico.
Un'altra caratterista importante è la sua compatibilità con tutti i tipi di Browser e sistemi mobile con Android e iOS.
Highcharts è, inoltre, dotata di una documentazione completa e dettagliata con diversi esempi con stili differenti per ciascuna tipologia di grafico. Questo offre agli sviluppatori la possibilità di scegliere, tra le diverse opzioni, quella maggiormente compatibile con la propria applicazione.

[5] Highcharts, Highcharts, 2023, https://www.highcharts.com/docs/index

4.1.2 Vantaggi e Svantaggi

Highcharts è una libreria che presenta numerosi vantaggi. La varietà di tipologie di grafici meno comuni come i grafici 3D la rende una grande opzione per applicazioni che necessitano rappresentazioni di dati alternative. Altri vantaggi sono dati dalla sua flessibilità nella personalizzazione e dalle diverse funzionalità interattive quali eventi come click, movimenti del mouse o zoom. Infine, Highcharts presenta una documentazione molto dettagliata con una grande varietà di esempi che rende più semplice l'integrazione dei grafici in un'applicazione e la risoluzione di eventuali problemi riscontrati. Un ulteriore vantaggio è la licenza di questa libreria. Highcharts è una libreria commerciale a pagamento, che mette a disposizione una licenza gratuita per l'uso non commerciale, di conseguenza è una libreria che risulta molto conveniente per progetti personali o accademici. A causa di questa licenza, però, questo vantaggio risulta, allo stesso tempo, un grande svantaggio per aziende e sviluppatori. Questi ultimi, infatti, per poter integrare i grafici Highcharts sono costretti ad acquistare la licenza. Un ulteriore possibile svantaggio potrebbe essere dato dal fatto che, per funzionalità avanzate, la documentazione potrebbe non essere molto approfondita.

In generale, Highcharts è una libreria che porta con se numerosi vantaggi e che permette la creazione di grafici completi e personalizzabili in base alle proprie esigenze. Risulta un'ottima soluzione per lo sviluppo di progetti a scopo personale o accademico. Conviene, invece, considerare altre librerie in caso di necessità commerciali.

4.1.3 Utilizzo e test

L'utilizzo della libreria Highcharts nel progetto Angular, alla base di questa tesi, risulta relativamente semplice. La struttura del progetto in Angular risulta uguale alla struttura del progetto in React, di conseguenza le tre librerie prese in analisi sono utilizzate tutte nello stesso progetto con la medesima struttura. Per ciascuna libreria sono state integrate le principali tipologie di grafici cercando di presentare differenti varianti. In questo caso, il componente HighChartComponent è il componente principale, il quale, attraverso il template *app-high-chart*, richiama i quattro componenti in cui sono integrate le diverse tipologie di grafici. Prendiamo come esempio il grafico a barre. Questa tipologia di grafico è utilizzata all'interno del componente HighChartBarComponent, con selettore *app-high-chart-bar*. Al suo interno sono presenti le diverse configurazioni di quattro tipologie di grafici a barre differenti, in modo

da rappresentare differenti personalizzazioni. La configurazione del grafico viene descritta sotto forma di oggetto, all'interno del quale vengono indicate proprietà come titolo ed eventuali scritte, colori, dimensioni, tooltip e tutte le diverse proprietà che si intendono utilizzare. In questo caso specifico sono state descritte proprietà come il titolo del grafico, categorie dei dati rappresentati, bordi e dati.

Di seguito viene mostrato il codice implementativo del componente HighChartBarComponent per l'utilizzo di uno solo dei quattro grafici a barre:

```
@Component({
  selector: 'app-high-chart-bar',
  templateUrl: './high-chart-bar.component.html',
  styleUrls: ['./high-chart-bar.component.css']
})
export class HighChartBarComponent implements AfterViewInit, OnInit {

  public optionsChart: Highcharts.Options = {}
  public data = [
    { name: 'US', value: 311591917 },
    { name: 'Canada', value: 34482779 },
    { name: 'Mexico', value: 112336538 },
    { name: 'UK', value: 62641000 },
  ]
  public title = "World Populations";
  ngOnInit(): void {
    this.optionsChart = {
      chart: {
        type: 'column'
      },
      title: {
        text: this.title
      },
      xAxis: {
        type: 'category',
        labels: {
          rotation: -45,
          style: {
            fontSize: '13px',
            fontFamily: 'Verdana, sans-serif'
          }
        }
      },
      yAxis: {
        min: 0,
        title: {
          text: 'Population (millions)'
        }
      },
      legend: {
        enabled: false
      },
      tooltip: {
        pointFormat: 'Population in 2021: <b>{point.y:.1f} millions</b>'
      },
      series: [{
        type: 'column',
        name: 'Population',
        colorByPoint: true,
        groupPadding: 0,
```

```
        data: this.data.map(data => { return [data.name, data.value] }),

      }]
    }}

  ngAfterViewInit() {
    Highcharts.chart('container', this.optionsChart);

  }
}
```

Con un risultato di questo tipo:

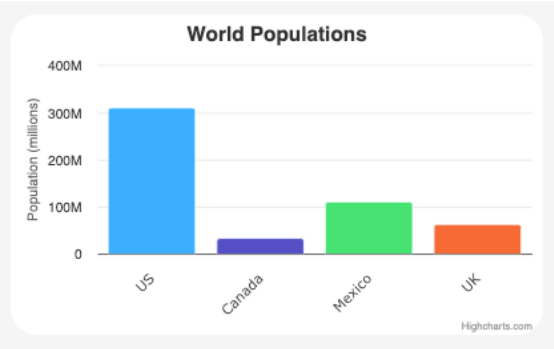

Figura 4.1.1: Risultato finale dell'integrazione di una variante di grafico a barre di Highcharts

Per quanto riguarda il risultato finale dell'utilizzo dei differenti grafici nel progetto, il risultato finale risulta:

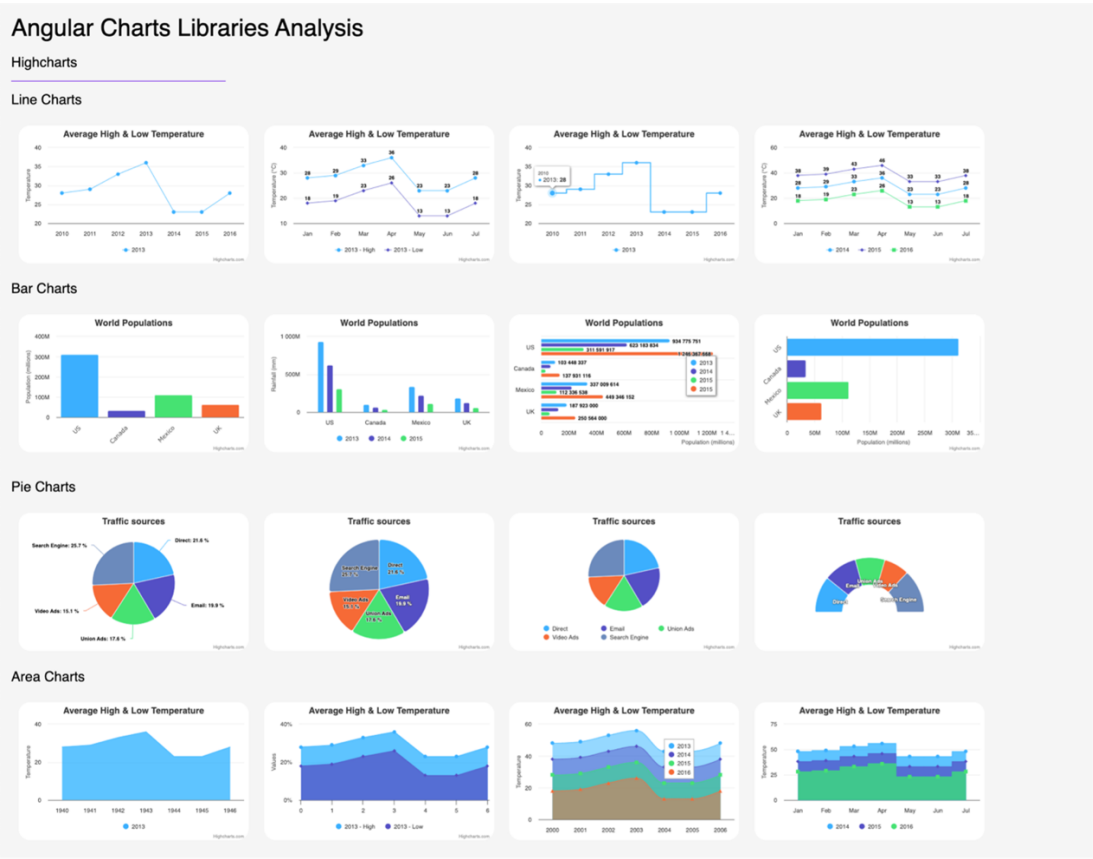

Figura 4.1.2: Risultato finale dell'utilizzo di Highcharts

4.2 FusionCharts

FusionCharts[6] è una popolare libreria JavaScript per la creazione di grafici interattivi e accattivanti progettata per applicazioni web sviluppate attraverso diversi framework. In questo caso è stata presa in esame come libreria per Angular. È una libreria che permette agli sviluppatori di rappresentare i dati dell'applicazione attraverso una vasta gamma di tipologie di grafici personalizzabili, così da rendere la lettura dei dati più semplice e intuitiva.

4.2.1 Caratteristiche principali

Una delle caratteristiche principali di questa libreria è la differenziazione di tipologie di grafici messe a disposizione degli sviluppatori. FusionCharts offre più di cento tipologie di grafici interattivi e personalizzabili attraverso i quali è possibile rappresentare qualsiasi tipologia di dati. A caratterizzate questa libreria è anche la possibilità di utilizzare una rappresentazione dei dati in 3D; infatti, FusionCharts offre la possibilità di utilizzare modelli tridimensionali per la maggior parte delle tipologie di grafici, tra cui grafici a barre, a colonne, a torta, mappe geografiche e tanti altri. Per quanto riguarda l'aspetto della personalizzazione, questa libreria offre una personalizzazione a 360°, a partire dal tema del grafico, fino ad arrivare ai minimi dettagli. FusionCharts offre, infatti, diverse tipologie di temi, con personalizzazioni predefinite, da applicare ai propri grafici, rendendoli maggiormente in linea con lo stile generale dell'applicazione. Oltre all'utilizzo di temi è possibile personalizzare singoli aspetti dei grafici, come colori, titoli e caratteri, etichette, legende, fino ad arrivare ad animazioni ed eventi. FusionCharts offre la possibilità di personalizzare funzioni che vengono eseguite quando si verifica un determinato evento, come l'utente che passa con il mouse sul grafico, o anche un semplice click. Questo permette di creare funzioni per suggerimenti, rendendo la comprensibilità dei dati più semplice e intuitiva. FusionCharts è una libreria che offre diverse funzionalità anche per l'utilizzo dei dati; infatti, è possibile creare grafici in tempo reale che si aggiornano dinamicamente man mano che i dati diventano disponibili. Per quando riguarda la gestione dei dati, questa libreria permette di rappresentare dati provenienti da diverse fonti come json, file .csv, ma anche da semplici database. Una caratteristica importante di questa libreria è anche la sua compatibilità con diverse tipologie di framework. Infatti, è possibile

[6] An Idera, Inc. Company, FusionCharts, 2023, https://www.fusioncharts.com/dev/fusioncharts

utilizzare i grafici di FusionCharts in applicazioni che utilizzano altri framework JavaScript come React e Vue.js.
Per quanto riguarda, invece, l'aspetto documentativo, è una libreria che offre una documentazione completa e popolata di esempi e tutorial semplici da utilizzare.

4.2.2 Vantaggi e Svantaggi

FusionCharts è una libreria che offre numerosi vantaggi, infatti la sua ampia varietà di tipologie di grafici con opzioni più avanzate come grafici a mappe permette di rappresentare i dati in modi più creativi e adeguati alle esigenze e allo stile generale dell'applicazione.
Il supporto per le mappe geografiche porta a questa libreria un grande vantaggio rispetto alle altre, infatti la possibilità di creare mappe interattive la rende un'ottima scelta per applicazioni che necessita visualizzazioni geografiche. L'elevata personalizzazione e la possibilità di integrare funzionalità interattive avanzate offre agli utenti la capacità di analizzare e interagire con i dati in modo più intuitivo, semplice e completo.
FusionCharts, inoltre, mette a disposizione di principianti o sviluppatori di applicazioni ad uso privato e di dimensioni ridotte, una versione gratuita che permette di utilizzare un numero inferiore di tipologie di grafici e con funzionalità di base.
Proprio come con Highcharts, questo vantaggio genera anche uno svantaggio piuttosto importante. Infatti, se si necessita di funzionalità e personalizzazioni avanzate, non incluse nella versione gratuita, è necessario acquistare una licenza commerciale generando un aumento di costi dell'applicazione. Un ulteriore svantaggio può essere dato dalla community, infatti, FusionCharts, a differenza di altre librerie, presenta una comunità ridotta e questo può portare a una maggiore difficoltà nella risoluzione di problemi nel suo utilizzo.

In generale, FusionCharts è un'ottima libreria ricca di funzionalità che permette agli sviluppatori di creare rappresentazioni di dati fuori dal comune e differenti da altre applicazioni. La flessibilità, e l'ampia varietà di tipologie di grafici, di questa libreria, la rendono un'ottima scelta per lo sviluppo di dashboard aziendali, piattaforme di analisi, piattaforme e-commerce e applicazioni che richiedono visualizzazioni dati geografiche.

4.2.3 Utilizzo e test

L'utilizzo di FusionCharts è molto simile all'utilizzo delle librerie precedenti. All'interno del progetto sono state integrate le principali tipologie di grafico della libreria, ovvero, differenti configurazioni di grafici a linee, a barre, a torte e ad area.
Per seguire come avviene l'integrazione dei grafici prendiamo come esempio il grafico a torta. Quest'ultimo viene utilizzato nel componente FusionChartPieComponent e al suo interno vengono configurate quattro varianti di grafici a torta con rappresentazioni bidimensionali e tridimensionali.
Ciascuna configurazione viene scritta attraverso l'utilizzo di un oggetto che contiene i dati del grafico e le impostazioni personalizzate. Per quanto riguarda le personalizzazioni del grafico, è possibile personalizzare titolo, sottotitolo, angolazione di partenza del grafico, legenda, visualizzazione di dati, etichette, temi, tooltip e tanto altro in base alla tipologia utilizzata.

Di seguito viene mostrato il codice del componente FusionChartPieComponent per l'integrazione di una sola delle quattro differenti tipologie di grafico a torta:

```
@Component({
  selector: 'app-fusion-chart-pie',
  templateUrl: './fusion-chart-pie.component.html',
  styleUrls: ['./fusion-chart-pie.component.css']
})
export class FusionChartPieComponent {
  chart2: Object;
  public data = [
    { value: 335, name: 'Direct' },
    { value: 310, name: 'Email' },
    { value: 274, name: 'Union Ads' },
    { value: 235, name: 'Video Ads' },
    { value: 400, name: 'Search Engine' }
];
public title = "Traffic sources"
  constructor() {
    const dataSource2 = {
        "chart": {
            "caption": "Split of Visitors by Age Group",
            "subCaption": "Last year",
            "startingAngle": "0",
            "showPercentValues": "1",
```

```
            "decimals": "1",
            "useDataPlotColorForLabels": "1",
            "theme": "fusion"
        },
        "data": [
            {
                "label": "Teenage",
                "value": "1250400"
            },
            {
                "label": "Adult",
                "value": "1463300"
            },
            {
                "label": "Mid-age",
                "value": "1050700"
            },
            {
                "label": "Senior",
                "value": "491000"
            }
        ]
    }
    this.chart2 = dataSource2;
  }
}
```

Come è possibile notare, sono state personalizzate impostazioni, tra cui il titolo del grafico, sottotitolo, angolazione di partenza, label con percentuale dei dati, una cifra decimale, il tema e i colori di ciascuna label in base al colore della porzione di grafico.

Ottenendo un grafico 3D di questa tipologia:

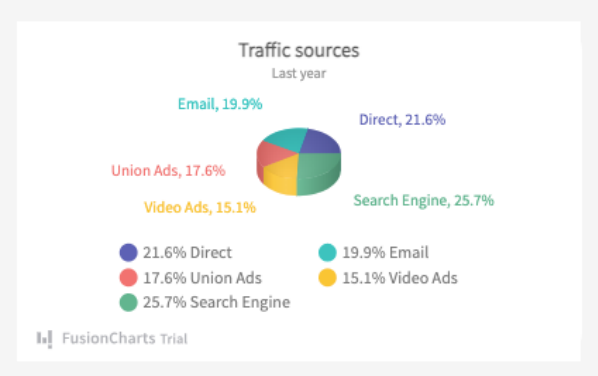

Figura 4.2.1: Risultato finale dell'integrazione di una delle varianti di grafico a torta di FusionCharts

Per quanto riguarda l'integrazione completa della libreria, il risultato finale nell'applicazione risulta:

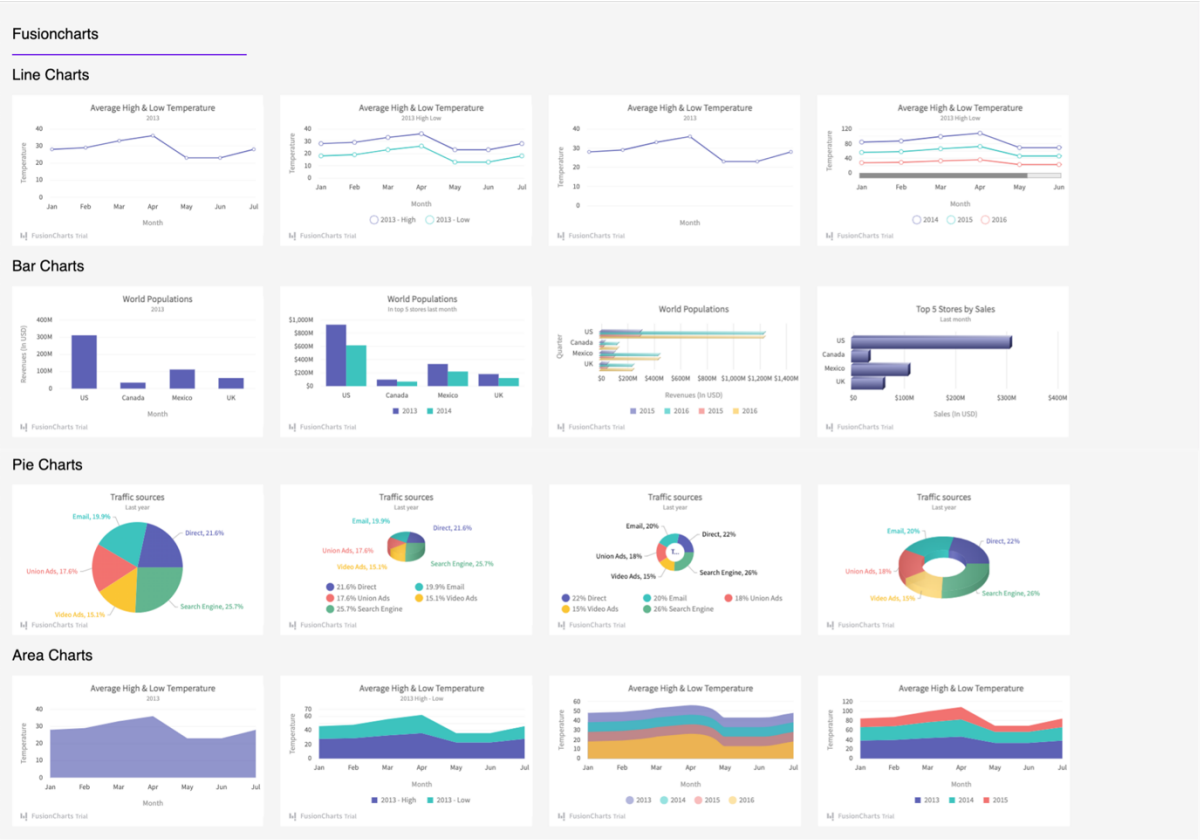

Figura 4.2.2: Risultato finale dell'utilizzo di FusionCharts

4.3 AnyChart

AnyChart[7] è una popolare libreria JavaScript che permette agli sviluppatori di creare grafici interattivi e visualizzazioni di dati di elevata qualità. È una libreria supportata da diverse piattaforme e framework, in questo caso Angular. AnyChart offre una vasta varietà di opzioni nella scelta della tipologia di grafico da utilizzare, a partire dai classici grafici comuni, fino ad arrivare a rappresentazioni tridimensionali, mappe e tanto altro.

4.3.1 Caratteristiche principali

AnyChart, come accennato precedentemente, è una libreria caratterizzata dalla sua vasta compatibilità con diverse piattaforme e framework, come React, Vue.js, .NET o come in questo caso Angular. AnyChart è caratterizzata principalmente dalla sua ampia varietà di scelta tra le sue tipologie di grafici. Offre, infatti, un centinaio di tipologie di grafici differenti e personalizzabili che permettono di creare applicazioni e strumenti avanzati e professionali.
Le tipologie di grafici variano dalle principali, tra cui grafici a linee, a barre, a torta e ad area, fino ad arrivare a rappresentazioni più avanzate come per mappe geografiche, radar, errori, sankey per la rappresentazione di flussi e tanto altro. Un altro aspetto distintivo di questa libreria è la sua personalizzazione. AnyChart, infatti, permette di personalizzare ogni singolo aspetto del grafico in base allo stile generale dell'applicazione e alle esigenze del progetto. È possibile personalizzare titoli, scritte, colori, assi cartesiani, legende, indicatori, tooltip e tanto altro, fino ad arrivare all'integrazione di strumenti e funzionalità per l'analisi dei dati come zoom, strumenti di disegno e molte altre funzionalità. Queste funzionalità permettono di rendere i grafici integrati altamente interattivi, offrendo la possibilità, all'utente che li utilizza, di comprendere i dati e il loro andamento in modo semplice, intuitivo e con maggiore precisione.
È una libreria che offre la possibilità di operare e rappresentare dati di elevate dimensioni, offrendo la capacità di creare applicazioni per la gestione, di dati e risorse, di alto livello.
Un ulteriore feature che caratterizza AnyChart è la sua funzionalità di esportazione dei suoi grafici in diversi formati. Questo permette la condivisione delle rappresentazioni grafiche attraverso file PDF e SVG, integrabili all'interno di documenti e rapporti.

[7] AnyChart.Com, AnyChart, 2023, https://docs.anychart.com/Quick_Start/Quick_Start

Infine, AnyChart offre una documentazione dettagliata e ricca di numerosi esempi di integrazione che aiutano costantemente gli sviluppatori nella risoluzione di problemi.

4.3.2 Vantaggi e Svantaggi

L'utilizzo di AnyChart in un progetto porta con sé svariati vantaggi. Innanzitutto, la sua ampia varietà di scelta nella tipologia di grafico costituisce uno dei vantaggi principali. Scegliere una libreria che supporti diverse tipologie di grafici, permette agli sviluppatori di utilizzare la rappresentazione migliore rispetto alle esigenze dell'applicazione, oltre che ad offrire un'interfaccia grafica distinguente. Ad aggiungere ulteriori vantaggi sono la flessibilità nella personalizzazione e l'interattività dei grafici. La libertà nel personalizzare i propri grafici rende una libreria un'ottima scelta per gli sviluppatori. Per quanto riguarda l'interattività, rappresentazioni interattive permettono, all'utente finale, di avere un'esperienza migliore nella lettura e nella comprensione dei grafici. Infine, un ulteriore vantaggio è dato dalla possibilità di esportare i diversi grafici utilizzati per poterli integrare in eventuali documenti. Alcuni svantaggi che possono essere generati dall'utilizzo di AnyChart, invece, sono principalmente il costo e l'apprendimento delle funzionalità di questa libreria. È una libreria a pagamento che offre una prova gratuita con funzionalità limitate, di conseguenza è necessario acquistare una delle licenze offerte per poter usufruire dei grafici per lo sviluppo di applicazioni commerciali. Ulteriori eventuali svantaggi potrebbero essere legati alle prestazioni, la rappresentazione di grafici complessi e interattivamente avanzati potrebbe richiedere un numero di risorse di sistema significativo, influenzando le prestazioni dell'applicazione.

In generale, AnyChart resta un'ottima libreria per la creazione di grafici di qualità e professionalità. Risulta una valida opzione per progetti che necessitano rappresentazioni dati di elevata interattività e che possiedono la possibilità di un investimento finanziario iniziale per l'acquisto di una licenza AnyChart. In caso contrario, conviene utilizzare altre librerie open-source, ma in ogni caso, prima di scegliere una libreria da integrare nel proprio progetto, è sempre opportuno comprendere le necessità dell'applicazione stessa.

4.3.3 Utilizzo e test

Come per le librerie precedenti, per ciascuna tipologia di grafico sono state integrate quattro varianti con personalizzazioni differenti. Prendiamo come esempio il grafico ad area, utilizzato all'interno del componente AnyChartAreaComponent. Al suo interno vengono dichiarate le quattro varianti di grafico, tra cui rappresentazioni bidimensionali e tridimensionali. Ciascun grafico viene poi configurato e mostrato attraverso la propria funzione. Per quanto riguarda la personalizzazione, in questo caso sono state personalizzate diverse proprietà, tra cui il titolo del grafico, colori, etichette e dati.

Di seguito viene mostrato il codice del componente AnyChartAreaComponent per l'integrazione del grafico ad area per la variante tridimensionale:

```
@Component({
  selector: 'app-any-chart-area',
  templateUrl: './any-chart-area.component.html',
  styleUrls: ['./any-chart-area.component.css']
})
export class AnyChartAreaComponent {
  public chart1 = anychart.area3d();
  public data = [
    { name: "Jan", value: 28 },
    { name: "Feb", value: 29 },
    { name: "Mar", value: 33 },
    { name: "Apr", value: 36 },
    { name: "May", value: 23 },
    { name: "Jun", value: 23 },
    { name: "Jul", value: 28 }
  ];

  public title = 'Average High & Low Temperature'

    drawChart2(container: string) {
    const data1 = this.data.map(data => { return [data.name, data.value] })

    const data2 = this.data.map(data => { return [data.name, data.value - 10] })

    this.chart1.title(this.title).enabled(true)

    const series1 = this.chart1.area(data1);

    const series2 = this.chart1.area(data2);

    // set the container id
    this.chart1.container(container);
    // initiate drawing the chart
    this.chart1.draw();
  }
ngOnInit(): void {
    this.drawChart2("anyChartArea2");
  }}
```

Come è possibile notare, questa variante di grafico, ad area tridimensionale, utilizza due serie di dati, entrambe con colori personalizzati e con semplici funzionalità di interazione come click del mouse e passaggio del cursore sull'area interessata.

Il codice precedente integra un grafico ad area con un risultato di questa tipologia:

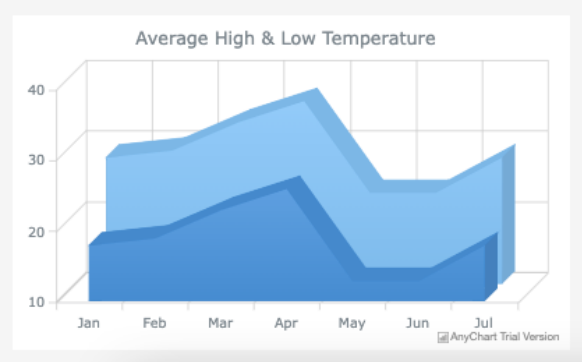

Figura 4.3.1: Risultato finale dell'integrazione di una delle varianti di grafico ad area di AnyChart

Per quanto riguarda l'integrazione completa della libreria nell'applicazione, invece, il risultato finale risulta della seguente tipologia:

Figura 4.3.2: Risultato finale dell'utilizzo di AnyChart

CAPITOLO QUINTO - ANALISI E CONFRONTO DELLE LIBRERIE PER VUE

Il quarto capitolo punta ad analizzare i dettagli di tre librerie di grafici compatibili con il framework Vue.js. Lo scopo di questo capitolo è fornire una valutazione dettagliata, secondo i parametri di valutazione, delle funzionalità e delle diverse proprietà che le caratterizzano.
Le librerie prese in analisi sono ApexCharts, Vue-ECharts e JSCharting. Per ciascuna libreria sono state integrate le quattro tipologie di grafici principali con diverse varianti di personalizzazioni. In questo modo è possibile confrontare le diverse tipologie di grafici evidenziando le diverse caratteristiche di ciascuna libreria. Per questa analisi, i dati rappresentati, sono dati statici come negli esempi della documentazione di ciascuna libreria.

Di seguito viene mostrata la tabella con i parametri di valutazione e i relativi risultati delle tre librerie utilizzate nel progetto in Vue.js.

	React			Angular			Vue		
Parametri di valutazione	Recharts	Victory	React-Chartjs-2	Highcharts	FusionCharts	AnyChart	ApexCharts	VueECharts	JSCharting
Compatibilità React	SI	SI	SI	SI	SI	SI	SI	NO	SI
Compatibilità Angular	NO	NO	NO	SI	SI	SI	SI	NO	SI
Compatibilità Vue	NO	NO	NO	SI	SI	SI	SI	SI	SI
Architettura a componenti	SI	SI	SI	NO	NO	NO	NO	SI	NO
Responsività	NO	SI	SI	SI	SI	SI	SI	SI	SI
Grafici semplici	SI	SI	SI	SI	SI	SI	SI	SI	SI
Grafici avanzati	NO	NO	NO	SI	SI	SI	SI	SI	SI
Grafici tridimensionali	NO	NO	NO	SI	SI	SI	NO	NO	NO
Personalizzazione semplice	SI	SI	SI	SI	SI	SI	SI	SI	SI
Personalizzazione avanzata	NO	SI	SI	SI	SI	SI	SI	SI	SI
Interattività semplice	SI	SI	SI	SI	SI	SI	SI	SI	SI
Interattività avanzata	NO	SI	SI	SI	SI	SI	SI	SI	SI
Supporto dati complessi	SI	SI	SI	SI	SI	SI	SI	SI	SI
Supporto Big Data	SI	SI	SI	SI	SI	SI	SI	SI	SI
Rappresentazione SVG	SI	SI	NO	SI	NO	SI	SI	SI	SI
Esportazione	NO	NO	NO	SI	SI	SI	SI	NO	SI
Compatibilità mobile	SI	SI	SI	SI	SI	SI	SI	SI	SI
Licenza a pagamento	NO	NO	NO	SI	SI	SI	NO	NO	SI
Licenza con versione gratuita	-	-	-	SI	SI	SI	-	-	SI
Documentazione completa	COMPLETA	COMPLETA	SCADENTE	COMPLETA	COMPLETA	COMPLETA	COMPLETA	SCADENTE	COMPLETA
Community	SI	SI	SI	SI	SI	SI	SI	SI	SI
Facilità di apprendimento	FACILE	FACILE	MEDIA	FACILE	MEDIA	FACILE	FACILE	MEDIA	FACILE
Facilità di implementazione	FACILE	FACILE	MEDIA	FACILE	MEDIA	FACILE	FACILE	MEDIA	FACILE

Tabella 5: Tabella dei parametri di confronto con i risultati delle librerie di Vue

5.1 ApexCharts

ApexCharts[8] è una moderna e popolare libreria di grafici che permette agli sviluppatori di creare rappresentazioni di dati interattive e accattivanti per applicazioni Web. È una libreria open-source con licenza del MIT completamente gratuita anche per utilizzi commerciali. ApexCharts offre diverse tipologie di grafici completamente personalizzabili in base alle esigenze e allo stile dell'applicazione.

5.1.1 Caratteristiche principali

Una delle caratteristiche principali di questa libreria è la sua semplicità, ApexCharts è una libreria estremamente semplice da comprendere e altrettanto da utilizzare. É compatibile con diversi framework, tra i quali React, Angular e in questo caso Vue. I grafici di ApexCharts sono responsivi di default, ovvero si riadattano al contenitore, semplificando il lavoro degli sviluppatori. È una libreria che permette di integrare diverse tipologie di grafici, a partire dai più semplici, fino ad arrivare a quelli più avanzati come mappe di calore.
ApexCharts offre una personalizzazione avanzata per i propri grafici. È, infatti, possibile personalizzare ogni singolo aspetto di ciascun grafico; in questo modo è possibile creare dei grafici con uno stile in linea con quello generale dell'applicazione. Lo stesso discorso vale per l'interattività. ApexCharts offre la possibilità di creare grafici con funzionalità di interazione avanzata, offrendo all'utente finale la possibilità di analizzare e comprendere i dati rappresentati in modo più semplice e definito. Inoltre, è possibile integrare grafici che rappresentano dati complessi e/o grandi quantità di dati. Un'ulteriore caratteristica di questa libreria è la rappresentazione dei grafici; ciascuno di essi viene rappresentato come SVG. In questo modo è possibile avere grafici scalabili e di elevata qualità. ApexCharts offre, inoltre, la possibilità di esportare i propri grafici in diversi formati così da poterli integrare in eventuali documenti; funzionalità che la rende un'ottima libreria per applicazioni che necessitano l'eventuale documentazione dell'andamento dei dati. Un'altra importante caratteristica è che tutti i grafici hanno il supporto per la compatibilità per applicazioni in versione mobile.
Per quanto riguarda il supporto documentativo, ApexCharts offre una documentazione completa, ben dettagliata e ricca di esempi da utilizzare nella propria applicazione.

[8] ApexCharts, ApexCharts, 2023, https://apexcharts.com/docs/installation/

È, inoltre, supportata da una grande community di sviluppatori che condividono le proprie risorse facilitando la risoluzione di problemi.

5.1.2 Vantaggi e Svantaggi

ApexCharts, come si può vedere, dispone di diversi vantaggi. In primis la sua semplicità nell'apprendere il suo funzionamento e le diverse funzionalità. Successivamente si presentano la vasta scelta di tipologie di grafici e la personalizzazione avanzata di ciascuno di essi. Un forte vantaggio è dato anche dalla documentazione completa e dettagliata, oltre che dalla community. Oltre ad essere semplice da capire, risulta anche semplice da utilizzare, un vantaggio non da poco per sviluppatori alle prime armi.
Possibili svantaggi potrebbero presentarsi per personalizzazioni complesse, poiché personalizzazioni altamente avanzate potrebbero richiedere maggior tempo di sviluppo. Un ulteriore svantaggio è la sua mancanza di tipologie di grafici tridimensionali. Questo potrebbe portare problemi nell'eventuale richiesta di utilizzo di grafici di questa tipologia.

In generale, ApexCharts è un'ottima libreria per la generazione di grafici semplici, accattivanti, interattivi e che richiedono personalizzazioni avanzate. Prima di effettuare la scelta di questa libreria, rimane comunque opportuno valutare altre librerie.

5.1.3 Utilizzo e test

L'utilizzo di ApexCharts risulta estremamente semplice.
Il progetto in Vue è strutturato esattamente come il progetto in React e Angular. Per ciascuna tipologia di grafico vengono integrate quattro varianti di personalizzazione all'interno di un componente designato. Quest'ultimo viene poi richiamato nel componente principale della libreria, mostrando tutte le quattro principali tipologie.
Le configurazioni di ciascun grafico vengono effettuate con un oggetto contenente le diverse proprietà personalizzate. Queste configurazioni vengono poi passate come attributi del componente all'interno del suo template (*:options: "" :series:"")*.
Prendiamo come esempio il componente ApexChartsLine, quest'ultimo genera le quattro tipologie di grafici a linee. In questo caso, il primo grafico è un grafico a linea singola con la funzionalità interattiva di zoom e le etichette dei dati disabilitate. Vengono poi personalizzati il titolo del grafico, l'asse delle ascisse e le righe della griglia rappresentante i punti.

Di seguito viene mostrato il codice del componente ApexChartsLine per la generazione del primo grafico a linee:

```
<script>
import VueApexCharts from "vue3-apexcharts";
export default {
    name: 'ApexChartsLine',
    components: {
        apexchart: VueApexCharts
    },
data: function () {
        const data = [
            { name: "Jan", value: 28 },
            { name: "Feb", value: 29 },
            { name: "Mar", value: 33 },
            { name: "Apr", value: 36 },
            { name: "May", value: 23 },
            { name: "Jun", value: 23 },
            { name: "Jul", value: 28 }
        ];
        const title = 'Average High & Low Temperature'

        return {

            chart1: {
                series: [{
                    name: "Desktops",
                    data: data.map(data => data.value)
                }],
                chart: {
                    type: 'line',
                    zoom: {
                        enabled: false
                    }
                },
                dataLabels: {
                    enabled: false
                },
                stroke: {
                    curve: 'straight'
                },
                title: {
                    text: title,
                    align: 'left'
                },
                grid: {
                    row: {
                        colors: ['#f3f3f3', 'transparent'],
                        opacity: 0.5
                    },
                },
                xaxis: {
                    categories: data.map(data => data.name),
                }
            }
       }
}
</script>
```

Con un risultato di questa tipologia:

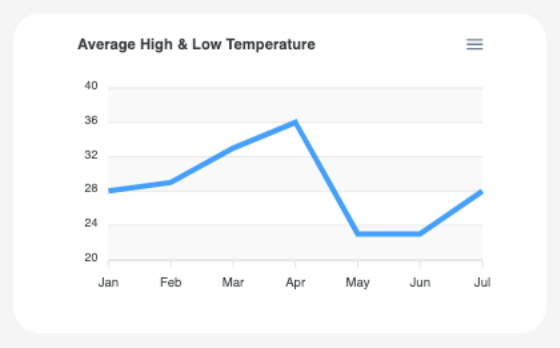

Figura 5.1.1: Risultato finale dell'integrazione di una delle varianti di grafico a linee di ApexCharts

Il risultato finale dell'utilizzo delle principali quattro tipologie di grafici risulta, invece, di questa tipologia:

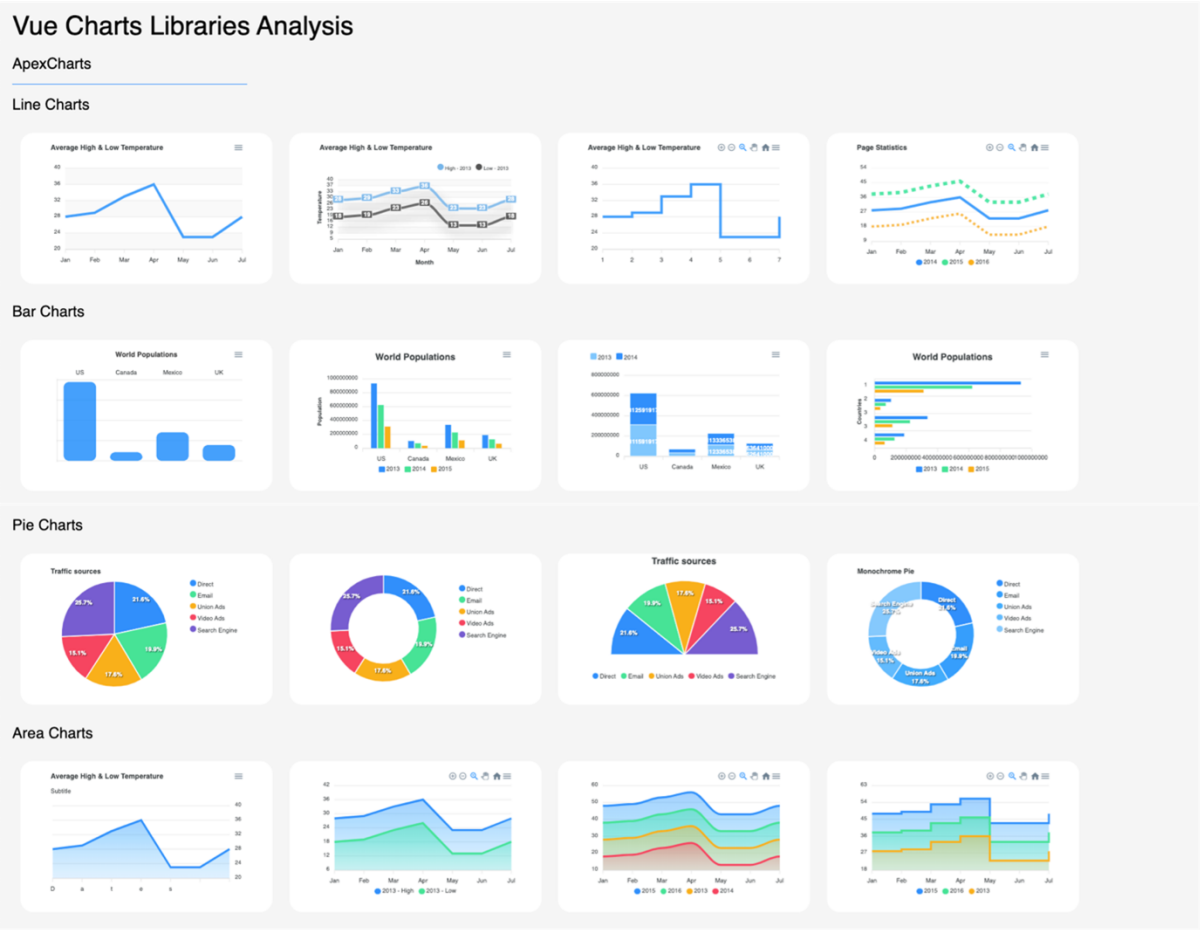

Figura 5.1.2: Risultato finale dell'utilizzo di ApexCharts

5.2 Vue-ECharts

Vue-ECharts[9] è una libreria open-source che consente di integrare i grafici della libreria ECharts all'interno di applicazioni Vue.js. ECharts è una popolare libreria JavaScript per la creazione di grafici interattivi e altamente personalizzabili, con una vasta gamma di tipologie di rappresentazioni e ampiamente utilizzata in applicazioni Web.

5.2.1 Caratteristiche principali

Vue-ECharts è progettata prettamente per essere utilizzata in applicazioni Vue.js; di conseguenza non risulta compatibile con ulteriori framework. In questo caso specifico è stata volutamente presa in analisi una libreria strettamente compatibile con Vue, poiché ECharts sarebbe risultata compatibile con anche tutti gli altri framework, ma non specifica per quest'ultimo. Vue-ECharts è caratterizzata dalla sua architettura a componenti predefiniti per diverse tipologie di grafici, completamente responsive, così da semplificarne lo sviluppo e compatibili con applicazioni in versione mobile. È una libreria che permette l'integrazione di diverse tipologie di grafici, semplici e avanzati, anche se in numero ridotto rispetto alla libreria principale ECharts. Permette di rappresentare tipologie di grafici che vanno dai principali, fino a grafici polari e mappe geografiche.
Vue-ECharts consente di creare personalizzazioni avanzate all'interno dei grafici con funzionalità interattive avanzate quali zoom, selezione di dati, attivazione e disattivazione di serie e tanto altro. Grazie a queste caratteristiche, è una libreria che permette di configurare il design e le funzionalità in base alle necessità dell'applicazione. In più, è una libreria che offre supporto per dati complessi e big data, consentendo la rappresentazione di grandi quantità di dati in modo chiaro e comprensivo. È caratterizzata da una grande community che contribuisce alla risoluzione di eventuali problematiche in fase di utilizzo, in quanto, la documentazione, è molto limitata e altamente specifica, di conseguenza nella fase inziale può risultare complessa da apprendere.

[9] VueEcharts, Vue-ECharts, 2023, https://vue-echarts.dev/

5.2.2 Vantaggi e Svantaggi

Vue-ECharts è caratterizzata da diversi vantaggi, tra cui la sua diretta compatibilità con Vue, il che rende questa libreria adatta ad applicazioni Vue.js. Ulteriori vantaggi sono l'elevata personalizzazione e le funzionalità di interazione avanzata, tra cui la possibilità di esportazione dei grafici in diversi formati. Inoltre, Vue-ECharts è rilasciata sotto una licenza open-source, il che la rende completamente gratuita per utilizzi a scopo commerciale. Purtroppo, è una libreria che presenta anche alcuni svantaggi, tra cui proprio la documentazione. Quest'ultima, infatti, è altamente specifica e in una prima fase iniziale può portare a perdite di tempo per apprendere le funzionalità; inoltre, in fase di utilizzo, può risultare complessa poiché nella documentazione sono presenti pochi esempi dettagliati.

In generale, Vue-ECharts è una buona libreria per la generazione di grafici personalizzati e interattivi specifica per applicazioni Vue. Dati i suoi vantaggi e svantaggi, converrebbe, in ogni modo, confrontare anche altre librerie in modo da avere ulteriori opzioni.

5.2.3 Utilizzo e test

L'utilizzo di Vue-ECharts potrebbe risultare un processo leggermente complesso in fase iniziale. Esattamente come nel caso della libreria React-chartjs-2, i componenti predefiniti dei grafici vanno "registrati" con la funzione use() e solo dopo possono essere utilizzati.

```
import { use } from 'echarts/core';
import { CanvasRenderer } from 'echarts/renderers';
import { PieChart } from 'echarts/charts';
import {TitleComponent,TooltipComponent,LegendComponent,} from 'echarts/components';
import VChart from 'vue-echarts';

use([CanvasRenderer,PieChart,TitleComponent,TooltipComponent,LegendComponent]);
```

Prendiamo per esempio il componente VueEchartsPie che genera le quattro varianti personalizzate di grafici a torta. La configurazione di ciascun grafico avviene tramite un oggetto che viene passato come attributo (*:option*) al componente nel template.
Nel caso del primo grafico, vengono configurate proprietà come titolo, tooltip, legenda e serie di dati con interattività al passaggio del mouse sul grafico.

Di seguito viene mostrato il codice del componente VueEchartsPie che integra la prima delle quattro varianti di grafici a torta:

```
export default {
    name: 'VueEchartsPie',
    components: {
        'vueChart': VChart
    },
    data: function () {
        const data = [
            { value: 335, name: 'Direct' },
            { value: 310, name: 'Email' },
            { value: 274, name: 'Union Ads' },
            { value: 235, name: 'Video Ads' },
            { value: 400, name: 'Search Engine' }
        ];

        return {
            chart1: {
                title: {
                    text: 'Traffic Sources',
                    left: 'center',
                },
                tooltip: {
                    trigger: 'item',
                    formatter: '{a} <br/>{b} : {c} ({d}%)',
                },
                legend: {
                    orient: 'horizontal',
                    top: 'bottom',
                    data: ['Direct', 'Email', 'Union Ads', 'Video Ads', 'Search
Engine'],
                },
                series: [
                    {
                        name: 'Traffic Sources',
                        type: 'pie',
                        radius: '55%',
                        center: ['50%', '60%'],
                        data: data,
                        emphasis: {
                            itemStyle: {
                                shadowBlur: 10,
                                shadowOffsetX: 0,
                                shadowColor: 'rgba(0, 0, 0, 0.5)',
                            },
                        },
                    },
                ],
            }
    }
}
```

Con un risultato di questa tipologia:

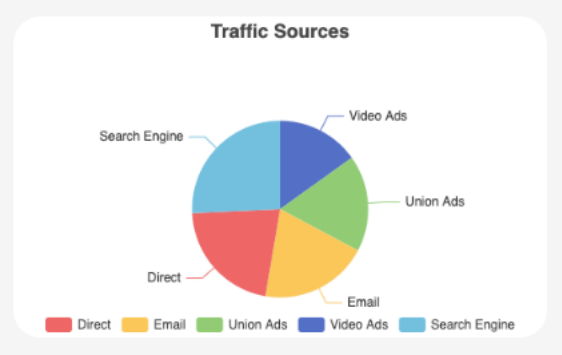

Figura 5.2.1: Risultato finale dell'integrazione di una delle varianti di grafico a torta di Vue-ECharts

Il risultato finale della generazione delle principali quattro tipologie di grafici a torta risulta, invece, di questa tipologia:

Figura 5.2.2: Risultato finale dell'utilizzo di Vue-ECharts

5.3 JSCharting

JSCharting[10] è una popolare libreria JavaScript per la creazione di grafici e rappresentazioni di dati altamente interattive e personalizzabili in modo avanzato. È una libreria che permette di creare rappresentazioni utilizzando il formato SVG, così da avere un'elevata qualità e scalabilità allo stesso tempo. Offre tantissime tipologie di grafici con differenti varianti per ciascuna di esse, consentendo di avere una grande scelta per rispettare lo stile della propria applicazione.

5.3.1 Caratteristiche principali

La principale caratteristica, di questa libreria, è proprio la sua ampia varietà di tipologie di grafici e rappresentazioni di dati. Offre centinaia di tipologie che vanno, dalle più semplici, alle più avanzate, come Gantt, Stocks, organigrammi e tanto altro. È una libreria compatibile con diverse tipologie di framework, in questo caso con React, Angular e Vue. Tutti i grafici di JSCharting sono realizzati attraverso SVG, questo permette di avere grafici responsivi e compatibili con applicazioni in versioni mobile su dispositivi di dimensioni differenti. La sua interattività è uno degli ulteriori aspetti principali. Consente di integrare nei propri grafici funzionalità di elevata interattività, dalla più semplice funzione di zoom o selezione di una certa area di dati, alla possibilità di connettere tipologie diverse di grafici tra loro con azioni specifiche. Per quanto riguarda la personalizzazione, è possibile modificare ogni singolo aspetto dei grafici. Inoltre, viene data la possibilità di integrare funzionalità aggiuntive attraverso i widget, rendendo i grafici maggiormente accattivanti e interattivi.
JSCharting offre supporto per la realizzazione di rappresentazioni di dati complessi e/o di grandi dimensioni in modo chiaro e comprensibile, così da garantire all'utente finale un'esperienza di analisi di alto livello. Per quanto riguarda la condivisione dei propri grafici, JSCharting offre la possibilità di esportare i grafici in diversi formati, consentendo l'integrazione delle rappresentazioni in documenti esterni.
JSCharting possiede diverse licenze di utilizzo, tra cui la versione gratuita con funzionalità e tipologie di grafici limitate. Acquistare una licenza JSCharting consente di avere accesso a tutte le funzionalità offerte dalla libreria.

[10] Corporate Web Solutions Ltd. & WebAvail Productions Inc, JSCharting, 2023, https://jscharting.com/tutorials/creating-js-charts/

Per quanto riguarda la fase iniziale di apprendimento, è una libreria dotata di un'ottima documentazione, chiara, dettagliata e ricca di esempi pronti da utilizzare nei propri progetti. Lo stesso discorso vale per la community, JSCharting possiede una grande community di utenti e sviluppatori che condividono le proprie risorse, problematiche e soluzioni.

5.3.2 Vantaggi e Svantaggi

Come è possibile vedere, JSCharting possiede numerosi vantaggi. Uno dei primi è proprio la sua varietà di scelta per quanto riguarda le tipologie di grafici. Personalizzazione e interattività avanzate sono altri dei principali vantaggi, sono caratteristiche che offrono agli sviluppatori un ampio margine di sviluppo nell'implementazione delle esigenze dell'applicazione.
La possibilità di esportare le rappresentazioni di dati costituisce un altro dei vantaggi di questa libreria, in quanto non tutte le ulteriori opzioni lo permettono.
Infine, la documentazione e la community sono gli ultimi aspetti vantaggiosi di JSCharting, il che rende questa libreria altamente supportata da diversi fronti.
Per quanto riguarda eventuali svantaggi, uno di essi potrebbe essere la sua mancanza di grafici tridimensionali; in caso di necessità di utilizzo di grafici di questa tipologia, sarà necessario fare ricorso ad ulteriori librerie. Il discorso della licenza potrebbe anch'esso generare alcuni svantaggi, poiché l'acquisto di una licenza completa aumenterebbe i costi di sviluppo del progetto.

In generale JSCharting risulta un'ottima opzione per la generazione di grafici interiettivi e personalizzabili all'interno di un'applicazione Web. I suoi vantaggi la rendono molto valida per lo sviluppo di diversi progetti sia in Vue che con altri framework. Rimane sempre un'ottima scelta, analizzare eventuali alternative nella fase iniziale di sviluppo del progetto.

5.3.3 Utilizzo e test

L'utilizzo di questa libreria, in un progetto Vue, risulta estremamente semplice.
Prendiamo come esempio il componente JSChartingArea che integra la tipologia di grafico ad area. Quest'ultimo genera quattro varianti di personalizzazione di grafici ad area. La configurazione di ciascun grafico avviene attraverso un oggetto contenente tutte le proprietà che si intendono integrare. Nel caso del primo grafico vengono personalizzate proprietà come il titolo del grafico, gli assi cartesiani, serie e punti dei dati rappresentati.

Ciascuna configurazione viene poi passata come parametro (*:options*) al componente richiamato nel template.

Di seguito viene mostrato il codice del componente JSChartingArea che integra la prima delle quattro varianti di grafici ad area:

```
export default {
    name: 'JSChartingArea',
    components: {
        JSChart: JSCharting
    },
    data: function () {
        const data = [
            { name: "Jan", value: 28 },
            { name: "Feb", value: 29 },
            { name: "Mar", value: 33 },
            { name: "Apr", value: 36 },
            { name: "May", value: 23 },
            { name: "Jun", value: 23 },
            { name: "Jul", value: 28 }
        ];

        const title = 'Average High & Low Temperature'

        return {

            chart1: {
                debug: true,
                type: 'area',
                title_label_text: title,
                legend_visible: false,
                // yAxis: { formatString: 'c' },

                xAxis: {
                    crosshair_enabled: true,
                    scale: { type: 'month' }
                },

                defaultSeries: {
                    shape_opacity: 0.3,
                    defaultPoint_marker: {
                        fill: 'white',
                        type: 'circle',
                        outline: { width: 2 }
                    }
                },

                series: [
                    {
                        name: '2013',

                        points: data.map(data => {return [data.name, data.value]})
                    }
                ]
            }
       }
}
```

Con un risultato di questa tipologia:

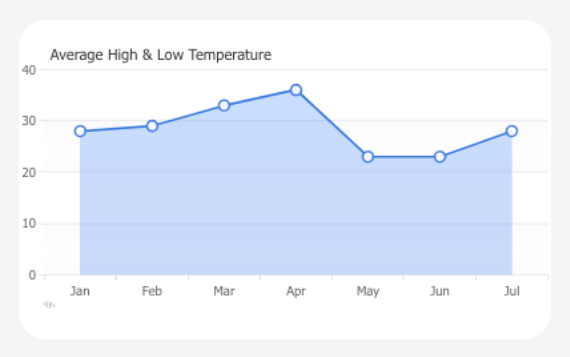

Figura 5.3.1: Risultato finale dell'integrazione di una delle varianti di grafico ad area di JSCharting

Il risultato finale dell'integrazione delle principali quattro tipologie di grafici ad area risulta, invece, di questa tipologia:

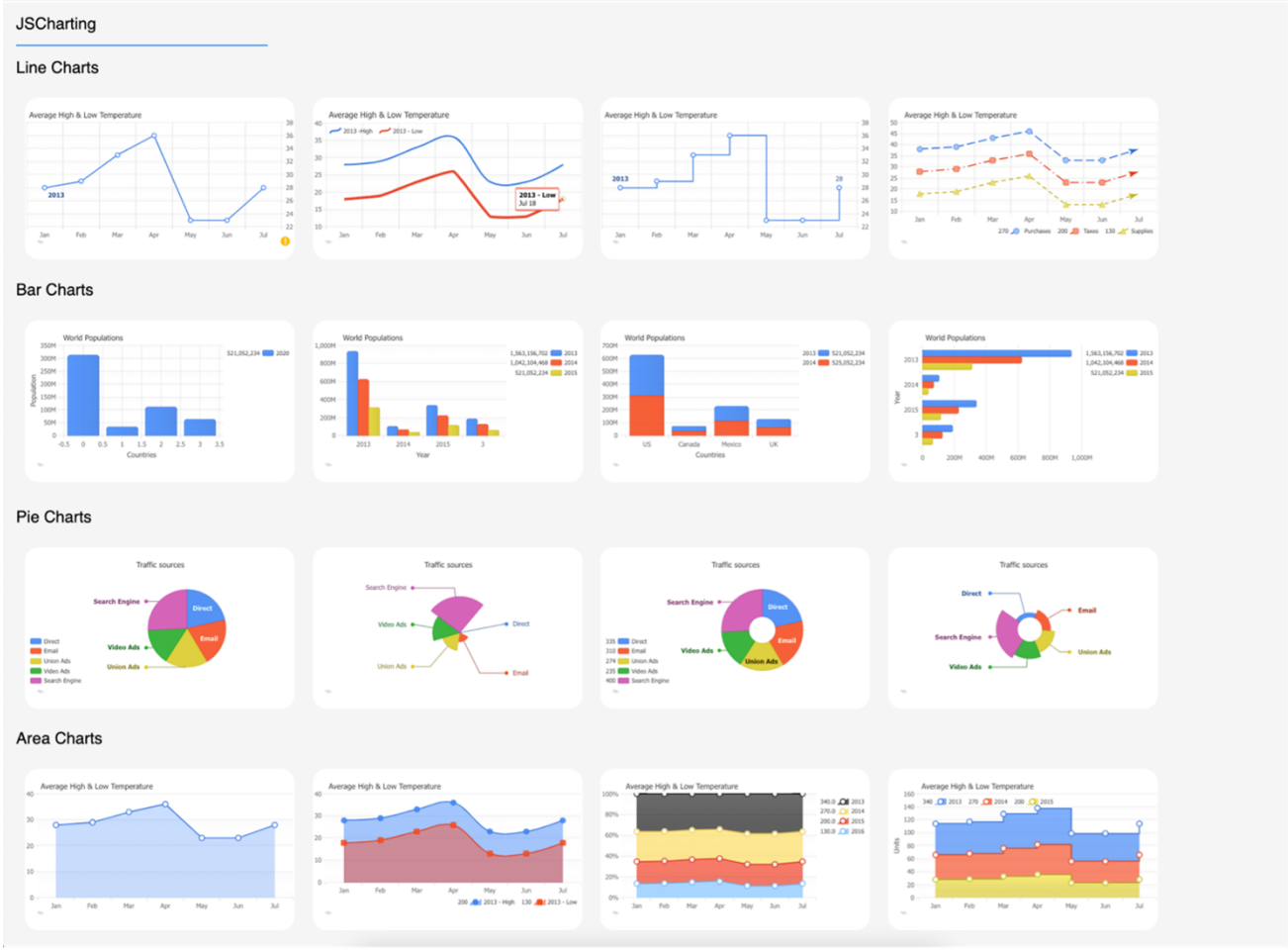

Figura 5.3.2: Risultato finale dell'utilizzo di JSCharting

CONCLUSIONI

Per concludere, attraverso questa analisi dettagliata delle librerie per la generazione di grafici per i framework React, Angular e Vue, è stato possibile evidenziare diverse opzioni disponibili per la visualizzazione dei dati in applicazioni web moderne. Ciascun framework utilizzato porta con sé le sue particolarità, che siano opportunità o difficolta; in questo caso, le librerie di grafici specificamente progettate per ciascun ambiente, possono offrire vantaggi significativi.

Come primo framework è stato utilizzato React, con l'utilizzo delle tre librerie Recharts, Victory e React-Chartjs-2. Ciò che è risultato dall'analisi di queste tre librerie è, innanzitutto, la specifica compatibilità e progettazione per React. Tutte e tre le librerie forniscono solide basi per la creazione di grafici e rappresentazioni dati interattive e responsive per applicazioni sviluppate con questo framework. Per semplici rappresentazioni di dati, che non presentano personalizzazioni avanzate e che non richiedono elevata interattività, Recharts potrebbe risultare un'ottima opzione. Per applicazioni che richiedono, invece, grafici con personalizzazioni avanzate e funzionalità interattive, per una migliore lettura e comprensione dei dati, Victory e React-Chartjs-2 si presentano come le opzioni migliori. Le differenze tra Victory e React-Chartjs-2 si presentano nel formato rappresentativo dei grafici e nella complessità nel comprendere la documentazione. Victory rappresenta i suoi grafici attraverso il formato SVG, così da garantire una migliore qualità e scalabilità che il Canvas di React-Chartjs-2 non offre. Inoltre, Victory presenta una documentazione completa e dettagliata che rende la generazione dei suoi grafici semplice e veloce, a differenza di React-Chartjs-2 che possiede una documentazione meno dettagliata e che rende l'utilizzo più complesso.
Per lo sviluppo di questo progetto, la libreria che è risultata migliore, in base alle capacità e conoscenze di sviluppo, è stata Victory. In generale, la scelta su quale libreria utilizzare dipenderà dalla complessità del progetto e dalle capacità e preferenze del team di sviluppo.

Per il secondo progetto, invece, è stato utilizzato Angular, con l'integrazione delle tre librerie Highcharts, FusionCharts ed AnyChart. Tutte e tre le librerie consentono la creazione di grafici e rappresentazioni di dati semplici, avanzate e tridimensionali, compatibili con tutti e tre i framework di sviluppo e in questo caso specifico con Angular. L'analisi di queste tre librerie ha portato alla luce un potenziale vincolo di grande importanza. Tutte e tre le librerie sono proprietarie e richiedono l'acquisto di una licenza per l'utilizzo commerciale.

Sono tre librerie che consentono di generare grafici e rappresentazioni di dati con personalizzazioni avanzate e di elevata interattività, oltre ad offrire una scelta di tipologia di grafici di ampia varietà. FusionCharts è l'unica libreria delle tre, che potrebbe risultare più problematica in fase di utilizzo, ma possiede un'ottima documentazione per la risoluzione di problemi. In questo caso tra le tre librerie, quella che è risultata migliore per lo sviluppo di questo progetto è Highcharts, sia per preferenza che per capacità di sviluppo personali. La scelta, su quale libreria utilizzare, dovrà essere basata sulle specifiche del progetto e sulle preferenze del team di sviluppo, poiché è possibile personalizzare qualsiasi aspetto.

Come ultimo framework è stato utilizzato Vue, con le tre librerie ApexCharts, Vue-ECharts e JSCharting. Tutte e tre le librerie consentono di creare grafici responsive con personalizzazioni avanzate e funzionalità interattive di alto livello. In questo caso, Vue-ECharts è una libreria specifica per Vue, mentre ApexCharts e JSCharting sono compatibili anche con gli altri framework. Tutte e tre le librerie consentono di creare grafici di diverse tipologie, semplici e avanzate, con l'esclusione dei grafici tridimensionali. Per applicazioni che necessitano diverse tipologie di grafici personalizzabili, interattivi ed esportabili, ApexCharts e JSCharting, in questo caso, sono le due librerie che costituiscono la scelta migliore. L'unica differenza, che vincola la scelta, è la licenza di JSCharting; infatti, quest'ultima necessita l'acquisto di una licenza per il suo utilizzo commerciale.
Un'opzione, invece, ad ApexCharts, potrebbe essere Vue-ECharts, che però non consente l'esportazione dei propri grafici. In questo caso, potrebbero sorgere problematiche legate alla sua documentazione altamente specifica, che potrebbe risultare complessa da comprendere e, di conseguenza, da integrare. Per lo sviluppo di questo progetto, tra tutte e tre le librerie, quella che è risultata migliore è ApexCharts; per le funzionalità che offre è estremamente semplice da comprendere e da utilizzare. In generale, però, la scelta della libreria, dovrà essere effettuata in base alle richieste specifiche del progetto, in base ad un eventuale budget messo a disposizione e in base alle capacità del team di sviluppo.

Infine, per quanto riguarda la libreria migliore e compatibile con tutti e tre i framework utilizzati, si presentano tre potenziali candidate. Il vincolo di scelta è dato dal budget fornito al progetto per l'eventuale acquisto di una licenza; in questo caso, la scelta verrebbe effettuata tra Highcharts ed AnyChart. In caso questo budget non sia disponibile, la scelta più ottimale, risulta essere ApexCharts con la sua licenza open-source.

In conclusione, l'analisi delle librerie per la generazione di grafici per React, Angular e Vue ha dimostrato che ogni framework offre soluzioni valide per la rappresentazione dei dati. La scelta della libreria più adatta dovrebbe essere basata sulle esigenze specifiche del progetto, sulle capacità del team di sviluppo e sulla preferenza personale. Indipendentemente dalla scelta, l'uso di queste librerie può migliorare notevolmente la capacità di presentare dati in modo efficace e comprensivo all'interno delle moderne applicazioni web.

BIBLIOGRAFIA E SITOGRAFIA

A

AnyChart.Com, AnyChart, 2023,
https://docs.anychart.com/Quick_Start/Quick_Start

An Idera, Inc. Company, FusionCharts, 2023,
https://www.fusioncharts.com/dev/fusioncharts

ApexCharts, ApexCharts, 2023,
https://apexcharts.com/docs/installation/

Ayerst J., React-chartjs-2, 2020,
https://react-chartjs-2.js.org/

B

Bampakos A., Deeleman P., Learning Angular: A no-nonsense guide to building web applications with Angular 15, Packt Publishing, 23/02/2023, 4th Edition,

C

Copelli D., React 100% Operativo, 28/02/2019, Independently Published

Corporate Web Solutions Ltd. & WebAvail Productions Inc, JSCharting, 2023,
https://jscharting.com/tutorials/creating-js-charts/

D

Durga P. A., I 40 Migliori Framework e Librerie JavaScript per il 2023, 08/06/2023,
https://kinsta.com/it/blog/librerie-javascript/

F

Formidable, Victory, 2015-2023,
https://formidable.com/open-source/victory/docs

H

Highcharts, Highcharts, 2023,
https://www.highcharts.com/docs/index

J

Janani A. K., 15 Best JavaScript Chart Libraries in 2023, 17/08/2023,
https://www.atatus.com/blog/javascript-chart-libraries/

N

Nugent C., Dataset California Housing Prices,
https://www.kaggle.com/datasets/camnugent/california-housing-prices

R

Recharts Group, Recharts, 2016-2023,
https://recharts.org/en-US

Repository GitLab, Progetto Angular,
https://gitlab.com/alessandromodelli/angular-chart-analysis

Repository GitLab, Progetto React,
https://gitlab.com/alessandromodelli/charts-libraries-analysis

Repository GitLab, Progetto Vue.js,
https://gitlab.com/alessandromodelli/vuechartanalysis

V

VueEcharts, Vue-ECharts, 2023,
https://vue-echarts.dev/