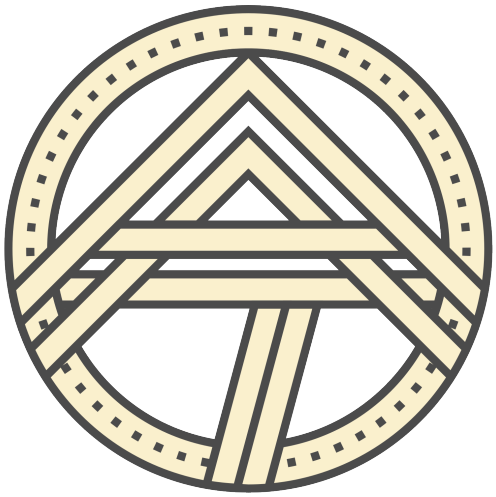

ARCHI7ECHS

Archi7echs - archi7echs@gmail.com

Progetto di **Ingegneria del Software**
A.A. 2024/2025

Specifica Tecnica

Autore: Il team

Ultima Modifica: 31/03/2025

Tipologia Documento: Interno

Tabella delle revisioni

Rev.	Data	Descrizione	Elaborazione	Verifica
0.2.0	31-03-2025	Stesura sezione componenti front-end	Gabriele Checchinato, Pietro Valdagno	Francesco Pozzobon, Giovanni Salvò
0.1.0	21-03-2025	Inizio stesura documento	Gabriele Checchinato	Giovanni Salvò, Pietro Valdagno

Indice

1) Introduzione

1.1) Finalità del documento

Questo documento ha l'obiettivo di fornire una descrizione dettagliata e strutturata degli aspetti tecnici fondamentali del progetto **3Dataviz**. In particolare, esso rappresenta una guida di riferimento per comprendere l'architettura del sistema, le scelte implementative adottate e le specifiche di deployment. Attraverso un'analisi approfondita, il documento illustra i principali componenti software e le tecnologie utilizzate. Inoltre, vengono descritte le motivazioni alla base delle decisioni progettuali, con un focus su scalabilità, manutenibilità e sicurezza del sistema. Gli obiettivi principali di questa specifica tecnica sono:

- Fornire una documentazione chiara e dettagliata a supporto dello sviluppo e della manutenzione del software.
- Garantire l'allineamento con i requisiti funzionali e non funzionali definiti nel documento *Analisi dei Requisiti v1.0.0*.
- Definire una base comune di conoscenza per tutti i membri del team, facilitando l'integrazione e l'evoluzione del sistema.

1.2) Scopo del progetto

L'obiettivo è realizzare una piattaforma web di visualizzazione tridimensionale dei dati, che consenta all'utente che la utilizza di navigare e interagire con grafici a barre verticali 3D rappresentanti dati complessi, utili per l'analisi e la presentazione di informazioni. Il prodotto deve essere progettato per poter rappresentare dati, in un modello 3D, navigabile e interattivo.
Dunque le sue funzionalità$_G$ principali includono:

- **Funzionalità di un ambiente 3D**:
  - **Rotazione**: permettere la rotazione del grafico per osservarlo da diverse angolazioni.
  - **Pan**: consentire lo spostamento del grafico sul piano orizzontale.
  - **Zoom**: abilitare l'avvicinamento e l'allontanamento dal grafico.
  - **Auto-positioning**: posizionare automaticamente il grafico in una vista ottimale.
- **Visualizzazione del valore medio globale**: il sistema deve consentire di visualizzare un piano parallelo alla base, che rappresenta il valore medio globale dei dati.
- **Opacizzazione o nascondimento delle barre**: il sistema deve offrire la possibilità di opacizzare o nascondere le barre con valori superiori o inferiori rispetto a:
  - una barra selezionata;
  - il valore medio globale rappresentato dal piano visualizzato.

  Inoltre, deve permettere di lasciare visibili o non opacizzati solo i valori di minimo o di massimo delle y, ossia i punti estremi.
- **Visualizzazione dei valori corrispondenti a una barra**: il sistema deve consentire di visualizzare i valori corrispondenti a una barra quando questa è soggetta a un evento « hover$_G$ « del mouse.
- **[Opzionale] Visualizzazione del valore medio del singolo elemento**: il sistema deve consentire di visualizzare un piano parallelo alla base, che rappresenta il valore medio di un singolo elemento di un asse (X o Z).

1.3) Glossario

All'interno del documento saranno spesso utilizzati degli acronimi o termini tecnici per semplificare la scrittura e la lettura. Per garantire che quanto scritto sia comprensibile a chiunque, è possibile usufruire del *glossario*. Tutte le parole consultabili nel glossario saranno identificate da una «G» in colore blu. Facendo click sul collegamento si aprirà una scheda del browser con il glossario

1.4) Riferimenti

1.4.1) Riferimenti normativi

- Norme di Progetto (v 1.0.0)
- Riferimento al capitolato$_G$ 5 di ***Sanmarco Informatica SPA - 3Dataviz***: https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C5.pdf - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Regolamento del progetto$_G$ didattico***: https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/PD1.pdf - *Ultimo accesso 20/03/2025*

1.4.2) Riferimenti informativi

- Riferimento documentazione: ***Svelte***: https://svelte.dev/docs/svelte/overview
  Ultimo accesso 20/03/2025
- Riferimento documentazione: ***Threlte***: https://threlte.xyz/
  Ultimo accesso 20/03/2025
- Riferimento documentazione: ***Sping_Boot*** https://spring.io/projects/spring-boot
  Ultimo accesso 20/03/2025
- Riferimento documentazione: ***Maven***: https://maven.apache.org/
  Ultimo accesso 20/03/2025
- Riferimento documentazione: ***PostgreSQL***: https://www.postgresql.org/
  Ultimo accesso 20/03/2025
- Riferimento documentazione: ***Docker***: https://docs.docker.com/
  Ultimo accesso 20/03/2025
- Riferimento alle slide IS: ***Progettazione: le dipendenze tra componenti***:
  https://www.math.unipd.it/~rcardin/swea/2022/Dependency%20Management%20in%20Object-Oriented%20Programming.pdf - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Analisi e descrizione delle funzionalità: Use Case e relativi diagrammi UML***: https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione e programmazione: Diagrammi delle classi (UML)***:
  https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Analisi e descrizione delle funzionalità: Diagrammi delle attività (UML)***: https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20di%20Attivit%C3%A0.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione: I pattern architetturali***: https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf
  - *Ultimo accesso 20/03/2025*

- Riferimento alle slide IS: ***Progettazione: Il pattern Dependency Injection***: https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Architetturali%20-%20Dependency%20Injection.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione: il pattern Model-View-Controller e derivati***: https://www.math.unipd.it/~rcardin/sweb/2022/L02.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione: i pattern creazionali (GoF)***: https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Creazionali.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione: I pattern strutturali (GoF)***: https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Strutturali.pdf
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Progettazione: I pattern di comportamento (GoF)***: https://drive.google.com/file/d/1cpi6rORMxFtC91nI6_sPrG1Xn-28z8eI/view?usp=sharing
  - *Ultimo accesso 20/03/2025*
- Riferimento alle slide IS: ***Programmazione: SOLID programming***: https://drive.google.com/file/d/1o1Xun2dVVc3mDiaGyN0FrDJhhoO3lfLQ/view?usp=sharing
  - *Ultimo accesso 20/03/2025*

2) Tecnologie

In questa sezione vengono elencate le tecnologie utilizzate all'interno del progetto **3Dataviz**, dalla fase di progettazione alla sua implementazione.
Ogni tecnologia utilizzata, verrà descritta tramite:

1. Nome della tecnologia
2. Descrizione della tecnologia e del suo utilizzo
3. Versione della tecnologia utilizzata
4. Link di riferimento alla sua documentazione

3) Front-end

3.1) Utilities

In questa sezione vengono documentate in dettaglio le utilities e i moduli di supporto utilizzati nell'applicazione. Questi file gestiscono la preparazione, il calcolo e la distribuzione dei dati e delle impostazioni globali, fornendo le basi per il funzionamento dell'interfaccia 3D e dei filtri. Per ciascuna utility verranno illustrati i seguenti aspetti:

- **Descrizione**: una breve spiegazione del ruolo e dello scopo della utility all'interno dell'applicazione.
- **Struttura e Funzionalità**: la composizione del file, evidenziando le funzioni chiave e la logica implementata per elaborare i dati e gestire lo stato globale.

3.1.1) index.svelte.ts

3.1.1.1) Descrizione

Questo file è un modulo per la gestione dello stato e la fornitura di dati computati per l'applicazione. Esso definisce i dati grezzi, ne calcola le proprietà (come la media, i valori minimi e massimi, e le dimensioni della matrice) e li espone tramite funzioni ed oggetti reattivi. Inoltre, il file gestisce la selezione degli elementi tramite la classe Selection e definisce un oggetto di filtro che raccoglie le impostazioni di visualizzazione e filtraggio.

3.1.1.2) Struttura e Funzionalità

- **Struttura**:
  - Vengono importati moduli fondamentali da **three** e funzioni di stato reattivo da Svelte.
  - Definisce una variabile `fetchedData` che contiene una matrice di dati grezzi.
  - Utilizza una store derivata per creare l'oggetto `data`, che include sia i valori originali che le proprietà computate, come:
    - La media dei valori,
    - I valori minimo e massimo,
    - Il numero di righe e colonne,
    - Il target predefinito per la visualizzazione e la posizione di default della camera.
  - Esporta la funzione `getData()`, che restituisce l'oggetto `data`, e la funzione `getValueFromId()`, utile per estrarre un valore dalla matrice in base a un identificatore.
  - Definisce la classe **Selection** che gestisce la selezione degli elementi con metodi per aggiungere, rimuovere, verificare e alternare gli elementi selezionati.
  - Crea un'istanza di **Selection** e definisce un oggetto di filtro che include impostazioni quali lo spacing, il range di valori, le opzioni di colorazione, i filtri per la media e le barre, e il flag per il display del filtro.
- **Funzionalità**:
  - Calcola in modo reattivo proprietà importanti dai dati grezzi per normalizzare e posizionare gli elementi 3D.
  - Fornisce un meccanismo per gestire la selezione degli elementi, utile per applicare filtri e evidenziare barre specifiche nel grafico.

- Esporta uno stato globale (filter) che raccoglie tutte le impostazioni di filtraggio e visualizzazione, da utilizzare nei componenti dell'applicazione.

3.1.1.3) Props e Variabili Reattive

- **Variabili Reattive:**
  - **fetchedData:** contiene i dati grezzi iniziali.
  - **data:** store derivata che combina i dati grezzi con le proprietà computate (media, min, max, righe, colonne, defaultTarget e defaultPosition).
  - **utils:** raccoglie le proprietà computate, utili per il posizionamento e il filtraggio.
- **Oggetto di Filtro:**
  - L'oggetto filter include impostazioni quali:
    - spacing,
    - rangeValue (min e max),
    - colorSelection,
    - avgFilter e avgEnabled,
    - barFilterSelection,
    - displayBarFilter,
    - selection (istanza della classe Selection).

3.1.1.4) Eventi e Comunicazione

- Pur non gestendo direttamente eventi, questo modulo fornisce le funzioni e gli oggetti di stato che altri componenti importano per comunicare e sincronizzare i dati e le impostazioni a livello globale.

3.1.1.5) Stili e Layout

- Non applica stili, in quanto funge esclusivamente da modulo di gestione dei dati e dello stato.

3.1.1.6) Esempi di Utilizzo

- In altri componenti, come App.svelte o Bar.svelte, il modulo viene importato per accedere ai dati e alle impostazioni.

```
import { getData, filter, getValueFromId } from '$lib/index.svelte';
let data = getData();
```

- Questo esempio evidenzia come il modulo fornisca le basi per il calcolo dei dati e la gestione dei filtri che vengono poi utilizzati per aggiornare dinamicamente la visualizzazione del grafico 3D.

3.1.1.7) Dipendenze Esterne

- **three:** Utilizzato per operazioni matematiche e vettoriali (ad es. Vector3) necessarie per il calcolo delle proprietà della scena 3D.
- **svelte (State Management):** Le funzioni state (state, derived, effect) sono impiegate per gestire la reattività e aggiornare automaticamente i dati e le proprietà computate.
- **lib/index.svelte:** Il modulo stesso funge da fonte centrale per le funzioni getData, getValueFromId e per la gestione dei filtri tramite l'oggetto filter.

Questa descrizione strutturata fornisce una panoramica completa del file index.svelte.ts, evidenziando come esso gestisca la preparazione e la distribuzione dei dati e delle impostazioni globali, e fungendo da riferimento essenziale per gli sviluppatori e i manutentori dell'applicazione.

3.2) Componenti

In questa sezione vengono documentati in dettaglio i componenti front-end[G] sviluppati in Svelte per il prodotto. Ogni file .svelte rappresenta un componente autonomo che implementa una specifica funzionalità dell' interfaccia utente[G] . Per ciascun componente verranno illustrati i seguenti aspetti:

- **Descrizione**: una breve spiegazione del ruolo e dello scopo del componente all'interno dell'applicazione.
- **Struttura e Funzionalità**: la composizione del componente, evidenziando gli elementi chiave e la logica implementata.
- **Props e Variabili Reattive**: i dati in ingresso (props) e le variabili reattive che gestiscono lo stato interno.
- **Eventi e Comunicazione**: come il componente comunica con altri elementi, tramite eventi custom o binding.
- **Stili e Layout**: le regole CSS o l'utilizzo di framework di styling adottati per garantire un'interfaccia coerente.
- **Esempi di Utilizzo**: un esempio pratico su come il componente viene importato e integrato nell'applicazione.
- **Dipendenze Esterne**: eventuali librerie aggiuntive utilizzate e spunti per ottimizzazioni future.

Questa sezione permette di comprendere il funzionamento e l'interazione dei vari componenti, fornendo un riferimento utile per sviluppatori e manutentori.

3.2.1) App.svelte

3.2.1.1) Descrizione

Il componente App.svelte è il punto di ingresso dell'applicazione e coordina il rendering della scena 3D e dei pannelli di controllo. Gestisce il canvas principale, carica i dati tramite la funzione getData() e imposta il target della visualizzazione, offrendo inoltre la possibilità di resettare il target alla configurazione di default.

3.2.1.2) Struttura e Funzionalità

- **Struttura**:
  - Il componente è contenuto in un `<div>` che occupa l'intera finestra, con un background scuro per evidenziare la scena 3D.
  - All'interno del `<Canvas>` (fornito da threlte/core) vengono integrati tre componenti principali:
    - SettingsPane.svelte: pannello per il controllo delle impostazioni e per il reset del target.
    - BarPane.svelte: componente che gestisce i filtri e la selezione delle barre.
    - Scene.svelte: responsabile del rendering della scena 3D in base al target impostato.
- **Funzionalità**:
  - Recupera i dati e le configurazioni computate tramite getData() e li memorizza in una variabile derivata, utils.

- Imposta il target della visualizzazione con il valore default ottenuto da getData().computed.defaultTarget e lo aggiorna in modo reattivo.
- Fornisce la funzione resetTarget() che ripristina il target al valore predefinito in utils.

3.2.1.3) Props e Variabili Reattive

- **Props**:
  - I componenti SettingsPane.svelte, BarPane.svelte e Scene.svelte ricevono tramite props i dati e le funzioni necessari per il loro corretto funzionamento.
  - In particolare, Scene.svelte riceve il prop target, fondamentale per centrare la visualizzazione 3D.
- **Variabili reattive**:
  - **utils**: variabile derivata (derived) che contiene i valori computati ottenuti da getData().computed.
  - **target**: stato reattivo (state) inizializzato al valore predefinito e aggiornato dalla funzione resetTarget().

3.2.1.4) Eventi e Comunicazione

- Il componente comunica con i suoi figli tramite binding delle props:
  - SettingsPane.svelte utilizza resetTarget() per permettere agli utenti di ripristinare il target della camera.
  - BarPane.svelte e Scene.svelte ricevono i dati e le impostazioni necessari per sincronizzare il rendering della scena 3D e la gestione dei filtri.

3.2.1.5) Stili e Layout

- Il componente è avvolto in un `<div>` che imposta:
  - Posizione: relative
  - Dimensioni: height: 100vh e width: 100vw
  - Background: rgb(14, 22, 37) per creare un ambiente visivo immersivo che esalta gli elementi 3D.

3.2.1.6) Esempi di Utilizzo

- Integrazione nell'applicazione: Il componente App.svelte viene utilizzato come punto di ingresso principale.

```
<script>
  import App from './App.svelte';
</script>

<App />
```

3.2.1.7) Dipendenze Esterne

- **threlte/core**: fornisce il componente Canvas e le funzionalità per il rendering 3D integrando Three.js in Svelte.
- **lib/index.svelte**: contiene la funzione getData(), che restituisce i dati e le configurazioni computate necessarie per inizializzare lo stato del componente.
- Svelte State Management: utilizza state e derived per gestire lo stato reattivo e derivare i valori computati nel componente.

3.2.2) Bar.svelte

3.2.2.1) Descrizione

Il componente Bar.svelte rappresenta una singola barra del grafico 3D. Esso visualizza un dato specifico mediante la sua altezza e il colore dinamico, rispondendo alle interazioni dell'utente (hover e click) per applicare filtri e gestire la selezione.

3.2.2.2) Struttura e Funzionalità

- **Struttura**:
  - Il componente utilizza un elemento T.Mesh per creare il mesh della barra, basato su una BoxGeometry e un MeshStandardMaterial, che gestisce trasparenza e colorazione dinamica.
  - Un elemento Text, posizionato sopra la barra, mostra il valore numerico della barra.
- **Funzionalità**:
  - Calcola l'opacità della barra in base a condizioni di filtraggio: se l'altezza rientra nell'intervallo definito e rispetta i filtri (basati sulla media e sui filtri specifici per le barre), l'opacità è impostata a 1, altrimenti a 0.2.
  - Utilizza un raycaster per rilevare le interazioni del mouse e applica un effetto hover mediante un tween definito con durata ed easing personalizzati.
  - La funzione getBarColor determina il colore della barra in base alla selezione del criterio di colorazione (per righe, per colonne o in base al valore normalizzato).

3.2.2.3) Props e Variabili Reattive

- **Props**:
  - **id**: identificatore univoco della barra, utilizzato per la gestione delle selezioni.
  - **coordinates**: posizione 3D della barra.
  - **height**: valore che determina l'altezza della barra.
  - **currentCameraQuaternionArray**: array che rappresenta l'orientamento corrente della camera, usato per orientare il testo.
- **Variabili reattive**:
  - **utils**: derivata contenente dati calcolati tramite `getData().computed`.
  - **inRange**: derivata che verifica se l'altezza rientra nei limiti definiti da `filter.rangeValue`.
  - **passesFilter**: calcola, in modo derivato, se la barra soddisfa ulteriori condizioni di filtro (in base a media e selezione).
  - **opacity**: impostata in modo derivato in base alla combinazione dei filtri `inRange` e `passesFilter`.
  - **mesh e text**: riferimenti allo stato dei componenti `T.Mesh` e `Text` per applicare gli effetti di interazione.

3.2.2.4) Eventi e Comunicazione

- Il componente gestisce eventi di pointer (movimento e uscita) per attivare o disattivare l'effetto hover, modificando il valore del tween hover.
- Gli eventi click, applicati sia al mesh della barra che al testo, invocano metodi di toggle o set sulla proprietà `filter.selection`, aggiornando lo stato di selezione.
- La comunicazione con il componente genitore avviene tramite il binding delle props essenziali (id, coordinates, height), mentre i filtri sono ora gestiti centralmente dal modulo `filter`.

3.2.2.5) Stili e Layout

- La posizione della barra è determinata dalle props coordinates e height, che definiscono il posizionamento 3D e la scala del mesh.
- Il componente Text è posizionato sopra la barra (offset in altezza) e ruotato in modo da garantire la leggibilità, in funzione dell'orientamento della camera.

3.2.2.6) Esempi di Utilizzo

- Il componente Bar.svelte viene utilizzato all'interno di un ciclo, dove per ogni elemento della matrice di dati viene creato un componente Bar.

```
{#each data as row, rowIndex}
  {#each row as height, colIndex}
    <Bar
      id={`${rowIndex}-${colIndex}`}
      coordinates={[
        rowIndex * spacing,
        height / 2,
        colIndex * spacing
      ]}
      {height}
      {currentCameraQuaternionArray}
    />
  {/each}
{/each}
```

3.2.2.7) Dipendenze Esterne

- **threlte/core**: Fornisce il framework per la creazione degli elementi 3D, come T.Mesh, BoxGeometry e MeshStandardMaterial.
- **threlte/extras**: Offre il componente Text e le funzioni per abilitare l'interattività nella scena.
- **three**: Libreria base per la manipolazione degli oggetti 3D, utilizzata per Raycaster, Vector2 e per operazioni vettoriali.
- **svelte/motion** e **svelte/easing**: Utilizzate per creare animazioni fluide (tweening) per l'effetto hover.
- **three/tsl**: Fornisce, se necessario, funzioni aggiuntive per la gestione delle selezioni.
- **lib/index.svelte**: modulo contenente `getData` e `filter`, che centralizza la gestione dei filtri applicati al grafico

3.2.3) BarPane.svelte

3.2.3.1) Descrizione

Il componente BarPane.svelte gestisce il filtro di selezione per le barre del grafico 3D. Se l'opzione di visualizzazione del filtro è attiva, viene mostrato un pannello fisso che contiene quattro pulsanti, ciascuno dei quali imposta un criterio di filtro diverso per le barre (solo la barra selezionata, barre con valori maggiori, barre con valori minori, o reset del filtro).

3.2.3.2) Struttura e Funzionalità

- **Struttura:**
  - Il componente importa i componenti Pane e Button dalla libreria **svelte-tweakpane-ui**.

    - Utilizza un controllo condizionale per rendere il pannello solo se la proprietà filter.displayBarFilter è attiva.
    - All'interno del pannello vengono resi quattro pulsanti, ognuno con una specifica etichetta e azione associata.
- **Funzionalità:**
    - Ogni pulsante, tramite l'evento on:click, modifica direttamente il valore di filter.barFilterSelection, impostando il criterio di filtraggio per le barre del grafico.
    - Il pulsante «Filter reset» ripristina il filtro annullando ogni criterio precedentemente applicato.

3.2.3.3) Props e Variabili Reattive

- **Props:**
    - Il componente utilizza l'oggetto filter importato da lib/index.svelte, che contiene le impostazioni per il filtraggio delle barre.
- **Variabili reattive:**
    - **filter.displayBarFilter**: determina se il pannello di filtro deve essere visualizzato.
    - **filter.barFilterSelection**: viene aggiornato in base al pulsante cliccato e controlla il criterio di filtro applicato al grafico.

3.2.3.4) Eventi e Comunicazione

- Il componente gestisce gli eventi on:click per ciascun pulsante, aggiornando il valore di filter.barFilterSelection.
- Queste modifiche vengono propagate al sistema, in modo che il grafico 3D si aggiorni dinamicamente in base al filtro selezionato.

3.2.3.5) Stili e Layout

- Il pannello è posizionato in maniera fissa (position fixed) con coordinate specifiche (y = 210, x = 120), garantendo che rimanga visibile nell'interfaccia utente.
- I pulsanti ereditano gli stili predefiniti della libreria **svelte-tweakpane-ui**, assicurando un aspetto coerente con il resto dell'interfaccia.

3.2.3.6) Esempi di Utilizzo

- Il componente BarPane.svelte viene renderizzato condizionatamente nel layout principale dell'applicazione quando il filtro delle barre è attivo.

```
<div>
  <Canvas>
    <BarPane />
  </Canvas>
</div>
```

3.2.3.7) Dipendenze Esterne

- **svelte-tweakpane-ui**: Fornisce i componenti Pane e Button, essenziali per la creazione dell'interfaccia di filtro.
- **lib/index.svelte**: Fornisce l'oggetto filter, che contiene le impostazioni e i metodi per la gestione dei filtri e delle selezioni.

3.2.4) CameraSettings.svelte

3.2.4.1) Descrizione

Il componente CameraSettings.svelte fornisce i controlli per regolare la posizione della camera nella scena 3D. Permette all'utente di effettuare lo zoom in, lo zoom out e di resettare la posizione della camera alla configurazione predefinita, influenzando direttamente la visualizzazione dell'ambiente 3D.

3.2.4.2) Struttura e Funzionalità

- **Struttura:**
  - Il componente importa il Button dalla libreria **svelte-tweakpane-ui** e utilizza il hook useThrelte per accedere al riferimento della camera.
  - Utilizza una variabile zoomValue e una costante zoomStep per controllare l'incremento o il decremento dello zoom.
  - Le funzioni zoomIn() e zoomOut() aggiornano zoomValue e richiamano updateCamera() per modificare la posizione della camera lungo la sua direzione attuale.
  - La funzione resetPosition() ripristina la posizione della camera al valore default ottenuto da utils.defaultPosition e invoca la funzione resetTarget() passata tramite props.
- **Funzionalità:**
  - **Zoom In**: diminuisce zoomValue e sposta la camera in avanti lungo la sua direzione corrente.
  - **Zoom Out**: aumenta zoomValue e sposta la camera indietro lungo la stessa direzione.
  - **Reset**: ripristina la posizione della camera e aggiorna il target della visualizzazione.

3.2.4.3) Props e Variabili Reattive

- **Props:**
  - **resetTarget**: funzione passata tramite props per ripristinare il target della camera.
- **Variabili reattive:**
  - **utils**: variabile derivata che contiene i valori computati ottenuti da getData().computed, inclusa la posizione default della camera.
  - **zoomValue**: stato numerico che controlla il livello di zoom corrente.
  - **zoomStep**: costante che definisce l'incremento/decremento per ogni operazione di zoom.

3.2.4.4) Eventi e Comunicazione

- Il componente gestisce gli eventi on:click sui pulsanti:
  - Il pulsante «Zoom In» invoca la funzione zoomIn(), spostando la camera in avanti.
  - Il pulsante «Zoom Out» invoca la funzione zoomOut(), spostando la camera indietro.
  - Il pulsante «Resetta» invoca resetPosition(), ripristinando la posizione della camera al valore di default e aggiornando il target.
- La comunicazione con il componente genitore avviene tramite la funzione resetTarget() e il binding dello stato della camera, garantendo un aggiornamento dinamico della visualizzazione.

3.2.4.5) Stili e Layout

- Il componente utilizza i pulsanti forniti da **svelte-tweakpane-ui**, che mantengono uno stile coerente con il resto dell'interfaccia.

- Non sono specificati ulteriori stili custom; lo styling è gestito dalla libreria e dal tema globale dell'applicazione.

3.2.4.6) Esempi di Utilizzo

- Integrazione in un pannello di controllo: Il componente viene inserito all'interno di un pannello di impostazioni (ad es. in SettingsPane.svelte) per consentire agli utenti di regolare la posizione della camera.

```
<TabPage title="Camera options">
      <CameraSettings {resetTarget} />
</TabPage>
```

- Questo esempio mostra come il componente permetta di controllare lo zoom e di ripristinare la vista della scena 3D.

3.2.4.7) Dipendenze Estere

- **threlte/core:** Fornisce il hook useThrelte e il contesto necessario per accedere alla camera e agli altri elementi della scena 3D.
- **svelte-tweakpane-ui:** Fornisce i componenti Button per la creazione dei controlli dell'interfaccia.
- **three:** Utilizzato per la gestione di vettori nello spazio 3D (Vector3) e per operazioni geometriche sulla posizione della camera.
- **lib/index.svelte:** (indiretto) Fornisce la funzione getData() per ottenere le configurazioni computate necessarie per inizializzare il valore default della camera.

3.2.5) Chart.svelte

3.2.5.1) Descrizione

Il componente Chart.svelte è responsabile del rendering della visualizzazione 3D del grafico. Esso gestisce la costruzione della scena, inclusa la griglia di fondo, il piano medio (quando abilitato), le etichette per righe e colonne e la generazione dinamica delle barre 3D tramite il componente Bar.svelte. Il componente aggiorna inoltre in tempo reale l'orientamento della camera per garantire che le etichette siano sempre leggibili.

3.2.5.2) Struttura e Funzionalità

- **Struttura:**
  - Il componente si struttura all'interno di un elemento T.Group, che raggruppa tutti gli elementi 3D della scena.
  - Viene renderizzata una Mesh che funge da griglia di fondo, creata con PlaneGeometry e MeshStandardMaterial, posizionata centralmente in base al numero di righe, colonne e allo spacing.
  - Se il flag `filter.avgEnabled` è attivo, viene renderizzato un piano medio (Mesh) posizionato a livello della media dei dati, con un materiale semitrasparente.
  - Le etichette delle righe e delle colonne vengono generate utilizzando il componente Text, posizionato e ruotato per garantire la leggibilità nell'ambiente 3D.
  - Infine, il componente itera sui dati per creare, per ogni elemento della matrice, un componente Bar.svelte che visualizza la barra corrispondente.
- **Funzionalità:**

- Calcola dinamicamente le dimensioni della griglia (rows, cols) e normalizza i valori dei dati per definire l'altezza delle barre.
- Utilizza la funzione `truncateText` per abbreviare le etichette se troppo lunghe.
- Aggiorna in modo reattivo i dati, i valori computati (`utils`), lo spacing e il target della scena, basandosi sui dati ottenuti tramite `getData()`.
- Monitora lo stato di `filter.displayBarFilter` aggiornando in base allo stato della selezione.

3.2.5.3) Props e Variabili Reattive

- **Props**:
  - Nessuna props per i filtri, in quanto ora vengono gestiti direttamente tramite `filter`.
- **Variabili reattive**:
  - **dt** derivato da `getData()`, contiene i dati grezzi e i valori computati.
  - **data**: matrice dei dati numerici, usata per definire le altezze e il posizionamento delle barre.
  - **utils**: derivata dai dati computati tramite `getData()`, contiene informazioni come la media e le dimensioni della griglia.
  - **spacing**: valore derivato da `filter.spacing` che determina la distanza tra le barre.
  - **rows e cols**: ottenuti da utils e rappresentano il numero di righe e colonne della matrice data.
  - **currentCameraQuaternionArray**: array che conserva l'orientamento corrente della camera, aggiornato continuamente per sincronizzare le etichette.

3.2.5.4) Eventi e Comunicazione

- Il componente utilizza `onMount` per avviare un ciclo di aggiornamento che cattura l'orientamento della camera (quaternion) in tempo reale, utilizzando `requestAnimationFrame`.
- Su `onDestroy`, l'animazione viene interrotta per liberare le risorse.
- Le variabili reattive vengono propagate ai componenti figli, in particolare a Bar.svelte, per sincronizzare la visualizzazione delle barre.

3.2.5.5) Stili e Layout

- Gli elementi 3D sono organizzati all'interno di T.Group per mantenere una struttura gerarchica chiara della scena.
- La griglia (Mesh con PlaneGeometry) e il piano medio sono centrati in base a rows, cols e spacing, garantendo un layout bilanciato.
- Le etichette (Text) sono posizionate e ruotate in modo da essere sempre leggibili, indipendentemente dall'orientamento della camera.

3.2.5.6) Esempi di Utilizzo

- Il componente Chart.svelte viene solitamente integrato all'interno della scena principale dell'applicazione per visualizzare il grafico 3D.

```
<Chart />
```

3.2.5.7) Dipendenze Esterne

- **threlte/core**:
  - Fornisce il framework per il rendering 3D, compreso il componente T.Group e altri elementi base come Mesh e PlaneGeometry.

- **threlte/extras**:
  - Offre il componente Text per la creazione di etichette 3D e funzioni per l'interattività.
- **three**:
  - Libreria fondamentale per la manipolazione degli oggetti 3D, utilizzata per operazioni come il calcolo delle dimensioni della griglia e per il raycasting.
- **svelte**:
  - Gestisce il ciclo di vita del componente tramite `onMount` e `onDestroy`, e supporta la reattività attraverso state, derived ed effect.
- **lib/index.svelte**:
  - Fornisce la funzione `getData()` e l'oggetto `filter`, che contengono i dati, le configurazioni computate e le impostazioni di filtraggio per la scena.

3.2.6) Color.svelte

3.2.6.1) Descrizione

Il componente **Color.svelte** fornisce un'interfaccia per la selezione del criterio di colorazione utilizzato nella visualizzazione 3D. Utilizza un controllo List della libreria **svelte-tweakpane-ui** per offrire opzioni predefinite come colorazione per righe, colonne o valori. Il valore selezionato è ora gestito tramite l'oggetto filter importato da `$lib/index.svelte`.

3.2.6.2) Struttura e Funzionalità

- **Struttura**:
  - Il componente importa e utilizza il componente `List` da **svelte-tweakpane-ui**, che fornisce un'interfaccia utente sotto forma di un menu a tendina per la selezione di un'opzione.
  - L'opzione selezionata è collegata a `filter.colorSelection`, che gestisce lo stato centralizzato del filtro.
  - L'oggetto `options` definisce le modalità di selezione disponibili per la colorazione.
- **Funzionalità**:
  - Ogni volta che l'utente cambia la selezione nel menu, il valore di `colorSelection` si aggiorna automaticamente grazie al binding bidirezionale, riflettendo la scelta dell'utente.
  - Il valore selezionato viene visualizzato in tempo reale all'interno di un elemento `<pre>`, che permette di visualizzare il numero corrispondente all'opzione scelta, fornendo un feedback immediato all'utente.

3.2.6.3) Props e Variabili Reattive

- **Props**:
  - **filter.colorSelection**: ora il valore selezionato è gestito attraverso l'oggetto filter importato, invece che come prop indipendente..
- **Variabili reattive**:
  - **options**: oggetto che contiene le possibili scelte per la modalità di colorazione.

3.2.6.4) Eventi e Comunicazione

- L'uso di bind:value={filter.colorSelection} permette al componente di aggiornare direttamente lo stato del filtro colore all'interno di `$lib/index.svelte`.
- Il valore selezionato viene immediatamente riflesso nella UI e nello stato centralizzato.

3.2.6.5) Stili e Layout

- Il componente non definisce stili personalizzati, poiché si affida alla libreria **svelte-tweakpane-ui** per la gestione dell'interfaccia utente.

3.2.6.6) Esempi di Utilizzo

Esempio di integrazione:

```
<TabPage title="Color filter">
      <Color />
</TabPage>
```

Nell'esempio il componente è incluso come una delle schede (TabPage) all'interno del pannello delle impostazioni, e permette di modificare il tipo di colore utilizzato nell'applicazione.

3.2.6.7) Dipendenze Esterne

- **svelte-tweakpane-ui:** fornisce il componente List per la selezione delle opzioni in modo strutturato e interattivo.
- **lib/index.svelte**: fornisce la gestione centralizzata dello stato del filtro colore.

3.2.7) DataFilter.svelte

3.2.7.1) Descrizione

Il componente **DataFilter.svelte** permette di filtrare i dati in base a intervalli specifici e valori relativi alla media. Gestisce un intervallo visibile di dati e consente all'utente di filtrare i valori inferiori o superiori alla media globale e include un'opzione per visualizzare il piano della media.

3.2.7.2) Struttura e Funzionalità

- **Struttura:**
  - Utilizza un `IntervalSlider` per consentire la selezione di un intervallo di valori da visualizzare.
  - Fornisce un `Checkbox` per attivare o disattivare la visualizzazione del piano della media.
  - Fornisce tre `Button`:
    - Uno per filtrare i **valori inferiori alla media**.
    - Uno per filtrare i **valori superiori alla media**.
    - Uno per **resettare** il filtro e riportare i valori all'intervallo completo.
- **Funzionalità:**
  - I valori minimi e massimi dell'intervallo vengono ottenuti dinamicamente tramite la funzione `getData().computed` e assegnati a `filter.rangeValue`.
  - Il `Checkbox` consente di attivare o disattivare la visualizzazione del piano della media attraverso la variabile `filter.avgEnabled`.
  - La variabile `value` controlla l'intervallo visualizzato.
  - I pulsanti modificano il valore di `filter.avgFilter`, che definisce i valori da visualizzare in base alla media globale.
  - Il reset del filtro riporta i valori di `filter.rangeValue` ai limiti minimi e massimi calcolati dinamicamente.

3.2.7.3) Props e Variabili Reattive

- **Props:**

    - **filter.rangeValue**: oggetto che contiene i valori minimo e massimo per l'intervallo di visualizzazione, aggiornato dinamicamente in base ai dati.
    - **filter.avgFilter**: variabile che memorizza il tipo di filtro applicato (0 = nessun filtro, 1 = sotto media, 2 = sopra media).
    - **filter.avgEnabled**: booleano che abilita/disabilita la visualizzazione del piano della media.
- **Variabili reattive:**
    - **utils:** contiene i dati calcolati dinamicamente tramite `getData().computed`.

3.2.7.4) Eventi e Comunicazione

- Il `Checkbox` modifica il valore di `filter.avgEnabled`, permettendo di attivare/disattivare la visualizzazione del piano della media.
- I Button aggiornano `filter.avgFilter` per applicare i filtri o resettare l'intervallo.
- L'intervallo di valori può essere modificato tramite il `IntervalSlider`, con il valore legato reattivamente a `filter.rangeValue`.

3.2.7.5) Stili e Layout

- Non sono specificati stili personalizzati. Viene utilizzato **svelte-tweakpane-ui** per il layout e l'interfaccia utente del filtro e dei pulsanti.

3.2.7.6) Esempi di Utilizzo

Esempio di integrazione:

```
<TabPage title="Data filter">
      <DataFilter />
</TabPage>
```

Nell'esempio il componente **DataFilter** viene utilizzato in una TabPage all'interno di un'interfaccia a schede (Tab Group), permettendo la gestione dei filtri dei dati nell'applicazione.

3.2.7.7) Dipendenze Esterne

- **svelte-tweakpane-ui:** non sono specificati stili personalizzati. Viene utilizzato **svelte-tweakpane-ui** che fornisce i componenti `IntervalSlider`, `Checkbox` e `Button` per la selezione e gestione dei filtri.
- **lib/index.svelte**: fornisce la gestione dei dati e delle variabili di filtro tramite `filter` e `getData().computed`.

3.2.8) Scene.svelte

3.2.8.1) Descrizione

Il componente **Scene.svelte** è responsabile per la visualizzazione della scena 3D, includendo la gestione della fotocamera, delle luci, e della rappresentazione grafica dei dati attraverso il componente **Chart**. Permette l'interazione tramite controlli di orbita e visualizza i dati in un contesto 3D dinamico.

3.2.8.2) Struttura e Funzionalità

- **Struttura:**
    - **Camera e controlli:**

    - Utilizzo di una `PerspectiveCamera` per visualizzare la scena 3D da una posizione predefinita.
    - `OrbitControls` per abilitare l'interazione dell'utente con la scena, includendo il damping e la rotazione automatica.
    - `Gizmo` per visualizzare gli assi e migliorare la comprensione spaziale della scena.
  - **Luci:** due luci direzionali e una luce ambientale, per creare un'illuminazione dinamica e realistica nella scena.
  - **Componente grafico:** `Chart` è il componente visualizza i dati in un grafico 3D, permettendo l'interazione.

- **Funzionalità:**
  - La fotocamera `PerspectiveCamera` viene utilizzata per configurare la visualizzazione della scena da un'angolazione iniziale predefinita, con la possibilità di manipolare la scena grazie agli `OrbitControls`.
  - `OrbitControls` permette di interagire con la scena tramite il mouse, consentendo operazioni come zoom, panoramica e rotazione.
  - Il `Gizmo` fornisce una rappresentazione visiva degli assi XYZ, migliorando l'interazione e la comprensione della scena da parte dell'utente.

3.2.8.3) Props e Variabili Reattive

- **Props:**
  - **target**: posizione della fotocamera, passata dinamicamente per centrare la scena 3D.

- **Variabili reattive:**
  - **autoRotate:** flag (settato su `false`) che abilita o disabilita la rotazione automatica della scena.

3.2.8.4) Eventi e Comunicazione

- Il componente **Scene.svelte** riceve il parametro `target` tramite props e lo passa al componente `OrbitControls` per manipolare la posizione della fotocamera..
- `OrbitControls` consente di interagire con la scena tramite il mouse per zoom, rotazione e panoramica.

3.2.8.5) Stili e Layout

- Il layout della scena è gestito tramite il sistema di **Threlte** e **Three.js**, con una configurazione di luci e camera integrata. Non sono presenti stili CSS personalizzati nel componente.

3.2.8.6) Esempi di Utilizzo

Esempio di integrazione:

```
<div>
  <Canvas>
    <Scene {target} />
  </Canvas>
</div>
```

L'esempio utilizzato in App.svelte mostra l'integrazione del componente **Scene** tramite il tag `<Scene>` all'interno del `<Canvas>` per rendere la scena 3D interattiva e visibile nell'applicazione.

3.2.8.7) Dipendenze Esterne

- **threlte/core**: gestisce la scena 3D, la fotocamera e il rendering.
- **threlte/extras**: fornisce i controlli `OrbitControls` e il `Gizmo` per la manipolazione della scena.
- **Three.js**: usato per la gestione di oggetti 3D come la fotocamera e le luci.

3.2.9) SettingsPane.svelte

3.2.9.1) Descrizione

Il componente **SettingsPane.svelte** gestisce la sezione di impostazioni dell'applicazione, organizzando diverse opzioni tramite una serie di schede. Fornisce agli utenti il controllo su vari parametri come le impostazioni della fotocamera, la fonte dei dati, i filtri, e la selezione del colore.

3.2.9.2) Struttura e Funzionalità

- **Struttura**:
  - Utilizza il componente `Pane` di **svelte-tweakpane-ui** per creare un pannello fisso con un gruppo di schede (TabGroup).
  - Ogni **scheda** (TabPage) contiene un componente separato per gestire un gruppo di impostazioni, tra cui:
    - `CameraSettings` per il controllo della posizione della fotocamera
    - `DataSource` per la gestione della fonte dei dati
    - `DataFilter` per il controllo dei filtri sui dati
    - `Color` per selezionare il tipo di colorazione
- **Funzionalità**:
  - Le schede offrono un'interfaccia interattiva per modificare e configurare parametri come la posizione della fotocamera, i filtri sui dati e la colorazione.
  - Sincronizzazione dei dati tra il componente e il resto dell'app tramite binding bidirezionale, permettendo che le modifiche alle impostazioni influenzino la visualizzazione dei dati in tempo reale.

3.2.9.3) Props e Variabili Reattive

- **Props**:
  - **resetTarget**: variabile passata da componenti esterni, utilizzata per ripristinare la posizione della fotocamera.
- **Variabili reattive**:
  - **bindable**: per legare variabili reattive come `mediaFilter`, `colorSelection`, `rangeValue`, `avgEnabled` a proprietà e parametri specifici, aggiornando automaticamente la visualizzazione quando cambiano.

3.2.9.4) Eventi e Comunicazione

- Il componente comunica con gli altri tramite le props, passando la variabile `resetTarget` al componente `CameraSettings`.

3.2.9.5) Stili e Layout

- Il pannello delle impostazioni ha una posizione fissa, impostato tramite la proprietà `position="fixed"`.

- Le schede sono organizzate in un layout tabellare tramite `TabGroup`, con ciascuna scheda (`TabPage`) che consente l'accesso a diverse configurazioni.

3.2.9.6) Esempi di Utilizzo

Esempio di integrazione:

```
<div>
  <Canvas>
    <SettingsPane {resetTarget} />
  </Canvas>
</div>
```

Nel'"esempio, il componente **SettingsPane** viene usato all"interno di un , con il parametro `resetTarget` passato tramite props.

3.2.9.7) Dipendenze Esterne

- **svelte-tweakpane-ui:** libreria per creare interfacce utente con pannelli di configurazione avanzati.
- **ThemeUtils:** usato per applicare temi globali e predefiniti.