Key Components

1. **Data Structure**: We use the provided JSON structure with Observations containing multiple Data’s (projects)
2. **Service Layer**: Handles API communication to fetch and save data
3. **Summary View Component**: Displays the data in a table format
4. **Type Definitions**: Strong typing for our data models

Implementation Details

1. Data Models (models/observation.ts)

export interface Observation {

Id: number;

Name: string;

Datas: ObservationData[];

}

export interface ObservationData {

SamplingTime: string;

Properties: Property[];

}

export interface Property {

Label: string;

Value: any;

}

export interface SummaryViewData {

samplingTime: string;

projectName: string;

constructionCount: number | null;

isConstructionCompleted: boolean | null;

lengthOfTheRoad: number | null;

}

2. Observation Service

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

import { Observation } from '../models/observation';

@Injectable({

providedIn: 'root'

})

export class ObservationService {

private apiUrl = 'http://localhost:5000/api/observations'; // Adjust port as needed

constructor(private http: HttpClient) { }

// Fetch all observation data from the API

getObservations(): Observable<Observation> {

return this.http.get<Observation>(this.apiUrl);

}

// Save updated observation data

updateObservation(observation: Observation): Observable<any> {

return this.http.post(this.apiUrl, observation);

}

}

3. Summary View Component

import { Component, OnInit } from '@angular/core';

import { ObservationService } from '../../services/observation.service';

import { Observation, SummaryViewData } from '../../models/observation';

@Component({

selector: 'app-summary-view',

templateUrl: './summary-view.component.html',

styleUrls: ['./summary-view.component.css']

})

export class SummaryViewComponent implements OnInit {

summaryData: SummaryViewData[] = [];

displayedColumns: string[] = [

'samplingTime',

'projectName',

'constructionCount',

'isConstructionCompleted',

'lengthOfTheRoad'

];

constructor(private observationService: ObservationService) { }

ngOnInit(): void {

this.loadData();

}

// Load and transform data from API

loadData(): void {

this.observationService.getObservations().subscribe({

next: (data) => {

this.summaryData = this.transformData(data);

},

error: (err) => {

console.error('Error loading data:', err);

// Handle error (show message to user)

}

});

}

// Transform raw API data into table-friendly format

private transformData(data: Observation): SummaryViewData[] {

return data.Datas.map(item => {

// Helper function to safely get property values

const getProperty = (label: string): any => {

const prop = item.Properties.find(p => p.Label === label);

return prop ? prop.Value : null;

};

return {

samplingTime: item.SamplingTime,

projectName: getProperty('Project Name') || 'N/A',

constructionCount: getProperty('Construction Count'),

isConstructionCompleted: getProperty('Is Construction Completed'),

lengthOfTheRoad: getProperty('Length of the road')

};

});

}

// Refresh data (could be connected to a refresh button)

refreshData(): void {

this.loadData();

}

}

**summary-view.component.html**

<div class="summary-container">

<div class="header">

<h2>Construction Projects Summary</h2>

<button (click)="refreshData()" class="refresh-button">

<span class="refresh-icon">↻</span> Refresh

</button>

</div>

<div class="table-container">

<table>

<thead>

<tr>

<th \*ngFor="let column of displayedColumns" [ngSwitch]="column">

<span \*ngSwitchCase="'samplingTime'">Sampling Time</span>

<span \*ngSwitchCase="'projectName'">Project Name</span>

<span \*ngSwitchCase="'constructionCount'">Construction Count</span>

<span \*ngSwitchCase="'isConstructionCompleted'">Completed</span>

<span \*ngSwitchCase="'lengthOfTheRoad'">Length (KM)</span>

</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let item of summaryData">

<td>{{ item.samplingTime | date:'medium' }}</td>

<td>{{ item.projectName }}</td>

<td>{{ item.constructionCount !== null ? item.constructionCount : 'N/A' }}</td>

<td>

<span [class.completed]="item.isConstructionCompleted"

[class.not-completed]="item.isConstructionCompleted === false">

{{ item.isConstructionCompleted !== null ?

(item.isConstructionCompleted ? 'Yes' : 'No') : 'N/A' }}

</span>

</td>

<td>{{ item.lengthOfTheRoad !== null ? item.lengthOfTheRoad : 'N/A' }}</td>

</tr>

</tbody>

</table>

<div \*ngIf="summaryData.length === 0" class="no-data">

No construction data available. Please check back later.

</div>

</div>

</div>

**summary-view.component.css**

.summary-container {

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 20px;

}

.refresh-button {

padding: 8px 16px;

background-color: #f0f0f0;

border: 1px solid #ddd;

border-radius: 4px;

cursor: pointer;

display: flex;

align-items: center;

}

.refresh-button:hover {

background-color: #e0e0e0;

}

.refresh-icon {

margin-right: 5px;

}

.table-container {

overflow-x: auto;

}

table {

width: 100%;

border-collapse: collapse;

}

th, td {

padding: 12px 15px;

text-align: left;

border-bottom: 1px solid #ddd;

}

th {

background-color: #f8f9fa;

font-weight: 600;

}

tr:hover {

background-color: #f5f5f5;

}

.completed {

color: #28a745;

font-weight: 500;

}

.not-completed {

color: #dc3545;

font-weight: 500;

}

.no-data {

padding: 20px;

text-align: center;

color: #6c757d;

font-style: italic;

}

Key Logic Explained

1. Data Transformation

The transformData() method converts the nested JSON structure into a flat format suitable for table display:

private transformData(data: Observation): SummaryViewData[] {

return data.Datas.map(item => {

const getProperty = (label: string): any => {

const prop = item.Properties.find(p => p.Label === label);

return prop ? prop.Value : null;

};

return {

samplingTime: item.SamplingTime,

projectName: getProperty('Project Name') || 'N/A',

constructionCount: getProperty('Construction Count'),

isConstructionCompleted: getProperty('Is Construction Completed'),

lengthOfTheRoad: getProperty('Length of the road')

};

});

}

2. Data Loading

The loadData() method handles the API call and data processing:

loadData(): void {

this.observationService.getObservations().subscribe({

next: (data) => {

this.summaryData = this.transformData(data);

},

error: (err) => {

console.error('Error loading data:', err);

// In a production app, show a user-friendly error message

}

});

}

3. Table Display

The HTML template dynamically generates the table based on the displayedColumns array and the transformed data:

<table>

<thead>

<tr>

<th \*ngFor="let column of displayedColumns">

<!-- Column headers with friendly names -->

</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let item of summaryData">

<!-- Display each data item with proper formatting -->

</tr>

</tbody>

</table>

4. Error Handling and Empty States

The component includes handling for:

* API errors (logged to console)
* Empty data states (shows "No data available" message)
* Missing properties (displays 'N/A')

Features

1. **Responsive Table**: Works on different screen sizes
2. **Data Formatting**:
   * Dates formatted using Angular's date pipe
   * Boolean values shown as Yes/No with color coding
   * Null values displayed as 'N/A'
3. **Refresh Capability**: Manual refresh button
4. **Type Safety**: Strong typing throughout the application
5. **Error Handling**: Basic error handling for API calls

Integration with the Rest of the Application

The summary view is integrated with:

* The tab navigation component for switching between views
* The observation service for data fetching
* The detailed view through shared data models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sampling Time | Project name | Construction count | Is Construction cmp | Length of road |
| 16-04-2025 10:25:00 Am | Road const-1 | 2 | False | 5.4 |
| 16-04-2025 10:30:20 AM | Road const2 | 2 | true | 4.2 |
|  |  |  |  |  |

A screen shot of a project

AI-generated content may be incorrect.