# Sourcecode Winkler Olivier – ProbeIPA

Grün markierter Code wurde während der Probezeit hinzugefügt

Gelb markierter Code wurde während der Probezeit angepasst

## Frontend

abotype.component.html

<apx-chart  
 [chart]="aboTypeChartOptions.chart"  
 [colors]="aboTypeChartOptions.colors"  
 [labels]="aboTypeChartOptions.labels"  
 [series]="aboTypeChartOptions.series"  
></apx-chart>

abotype.component.ts

import {Component, Input} from '@angular/core';  
import {ApexChart, ApexNonAxisChartSeries} from 'ng-apexcharts';  
import {AboType} from '../../abotype.model';  
import {ChartColors} from '../../chart-colors';  
  
export type ChartOptions = {  
 series: ApexNonAxisChartSeries;  
 chart: ApexChart;  
 colors: string[];  
 labels: string[];  
};  
  
@Component({  
 selector: 'app-abotype',  
 templateUrl: './abotype.component.html',  
 styleUrls: ['./abotype.component.scss']  
})  
export class AbotypeComponent {  
  
 public aboTypeChartOptions: Partial<ChartOptions>;  
  
 @Input() set aboType(aboType: AboType) {  
 this.initChart(aboType);  
 }  
  
 constructor() {  
 }  
  
 private initChart(aboType: AboType) {  
 this.aboTypeChartOptions = {  
 series: [  
 aboType.aboGa,  
 aboType.aboHalf,  
 aboType.aboRegional,  
 aboType.aboP2p,  
 aboType.aboOther  
 ],  
 chart: {  
 width: 500,  
 type: 'donut'  
 },  
 colors: ChartColors,  
 labels: [  
 $localize`GA`,  
 $localize`Halbtax`,  
 $localize`Regionales Abonnement`,  
 $localize`Direktbillett`,  
 $localize`Direktbillett`  
 ]  
 };  
 }  
}

agegroup.component.html

<apx-chart  
 [chart]="ageGroupChartOption.chart"  
 [colors]="ageGroupChartOption.colors"  
 [dataLabels]="ageGroupChartOption.dataLabels"  
 [legend]="ageGroupChartOption.legend"  
 [plotOptions]="ageGroupChartOption.plotOptions"  
 [series]="ageGroupChartOption.series"  
 [states]="ageGroupChartOption.states"  
 [subtitle]="ageGroupChartOption.subtitle"  
 [title]="ageGroupChartOption.title"  
 [tooltip]="ageGroupChartOption.tooltip"  
 [xaxis]="ageGroupChartOption.xaxis"  
 [yaxis]="ageGroupChartOption.yaxis"  
></apx-chart>

agegroup.component.ts

import {Component, Input} from '@angular/core';  
import {  
 ApexAxisChartSeries,  
 ApexChart,  
 ApexDataLabels,  
 ApexGrid,  
 ApexLegend,  
 ApexPlotOptions,  
 ApexStates,  
 ApexTitleSubtitle,  
 ApexYAxis  
} from 'ng-apexcharts';  
import {AgeGroup} from '../../agegroup.model';  
import {ChartColors} from '../../chart-colors';  
  
export type ChartOptions = {  
 series: ApexAxisChartSeries;  
 chart: ApexChart;  
 dataLabels: ApexDataLabels;  
 plotOptions: ApexPlotOptions;  
 yaxis: ApexYAxis;  
 xaxis: ApexXAxis;  
 grid: ApexGrid;  
 subtitle: ApexTitleSubtitle;  
 colors: string[];  
 states: ApexStates;  
 title: ApexTitleSubtitle;  
 legend: ApexLegend;  
 tooltip: any; *//ApexTooltip;*};  
  
type ApexXAxis = {  
 type?: 'category' | 'datetime' | 'numeric';  
 categories?: any;  
 labels?: {  
 style?: {  
 colors?: string | string[];  
 fontSize?: string;  
 };  
 };  
};  
  
@Component({  
 selector: 'app-agegroup',  
 templateUrl: './agegroup.component.html',  
 styleUrls: ['./agegroup.component.scss']  
})  
export class AgegroupComponent {  
  
 public ageGroupChartOption: Partial<ChartOptions>;  
  
 @Input() set ageGroup(ageGroup: AgeGroup) {  
 this.initChart(ageGroup);  
 }  
  
 constructor() {  
 }  
  
 private initChart(ageGroup: AgeGroup) {  
 this.ageGroupChartOption = {  
 series: [  
 {  
 data: [  
 {  
 x: '0 - 20 Jahre',  
 y: ageGroup.genZ,  
 },  
 {  
 x: '21 - 40 Jahre',  
 y: ageGroup.millennial,  
 },  
 {  
 x: '41 - 60 Jahre',  
 y: ageGroup.genX,  
 },  
 {  
 x: '61 - 80 Jahre',  
 y: ageGroup.babyBoomer,  
 },  
 ]  
 }  
 ],  
 chart: {  
 height: 300,  
 width: '100%',  
 type: 'bar',  
 toolbar: {  
 show: false  
 }  
 },  
 plotOptions: {  
 bar: {  
 distributed: true,  
 horizontal: true,  
 barHeight: '75%',  
 dataLabels: {  
 position: 'bottom'  
 }  
 }  
 },  
 legend: {  
 show: false  
 },  
 colors: ChartColors,  
 dataLabels: {  
 textAnchor: 'start',  
  
 formatter: function(val, opt) {  
 return opt.w.globals.labels[opt.dataPointIndex];  
 },  
  
 },  
 tooltip: {  
 x: {  
 show: false  
 },  
 y: {  
 title: {  
 formatter: function(val, opts) {  
 return opts.w.globals.labels[opts.dataPointIndex];  
 }  
 }  
 }  
 },  
 yaxis: {  
 labels: {  
 show: false  
 }  
 }  
 };  
 }  
}

journeyrating.component.html

<apx-chart  
 [chart]="jounreyRatingChartOptions.chart"  
 [colors]="jounreyRatingChartOptions.colors"  
 [dataLabels]="jounreyRatingChartOptions.dataLabels"  
 [grid]="jounreyRatingChartOptions.grid"  
 [legend]="jounreyRatingChartOptions.legend"  
 [plotOptions]="jounreyRatingChartOptions.plotOptions"  
 [series]="jounreyRatingChartOptions.series"  
 [tooltip]="jounreyRatingChartOptions.tooltip"  
 [xaxis]="jounreyRatingChartOptions.xaxis"  
 [yaxis]="jounreyRatingChartOptions.yaxis"  
></apx-chart>

journeyrating.component.ts

import {Component, Input} from '@angular/core';  
import {ApexAxisChartSeries, ApexChart, ApexDataLabels, ApexGrid, ApexLegend, ApexPlotOptions, ApexXAxis, ApexYAxis} from 'ng-apexcharts';  
import {JourneyRating} from '../../journey-rating.model';  
import {ChartColors} from '../../chart-colors';  
  
export type ChartOptions = {  
 series: ApexAxisChartSeries;  
 chart: ApexChart;  
 dataLabels: ApexDataLabels;  
 plotOptions: ApexPlotOptions;  
 yaxis: ApexYAxis;  
 xaxis: ApexXAxis;  
 tooltip: any;  
 grid: ApexGrid;  
 colors: string[];  
 legend: ApexLegend;  
 labels: any;  
};  
  
@Component({  
 selector: 'app-journeyrating',  
 templateUrl: './journeyrating.component.html',  
 styleUrls: ['./journeyrating.component.scss']  
})  
export class JourneyratingComponent {  
  
 public jounreyRatingChartOptions: Partial<ChartOptions>;  
  
 constructor() {  
 }  
  
 @Input() set journeyRating(journeyRating: JourneyRating) {  
 this.initChart(journeyRating);  
 }  
  
 private initChart(journeyRating: JourneyRating) {  
  
 this.jounreyRatingChartOptions = {  
 series: [  
 {  
 data: [  
 journeyRating.totalHappinessFactorAwesome,  
 journeyRating.totalHappinessFactorGood,  
 journeyRating.totalHappinessFactorOk,  
 journeyRating.totalHappinessFactorBad,  
 journeyRating.totalHappinessFactorWorst  
 ]  
 }  
 ],  
 chart: {  
 height: 300,  
 width: '100%',  
 type: 'bar',  
 toolbar: {  
 show: false  
 }  
 },  
 colors: ChartColors,  
 plotOptions: {  
 bar: {  
 columnWidth: '45%',  
 distributed: true  
 }  
 },  
 dataLabels: {  
 enabled: false  
 },  
 legend: {  
 show: false  
 },  
 tooltip: {  
 x: {  
 show: false  
 },  
 y: {  
 title: {  
 formatter: function(val, opts) {  
 return `Bewertung ${opts.w.globals.labels[opts.dataPointIndex]}`;  
 }  
 }  
 }  
 },  
 xaxis: {  
 categories: [  
 '5',  
 '4',  
 '3',  
 '2',  
 '1'  
 ],  
 labels: {  
 style: {  
 colors: [  
 '#008FFB',  
 '#00E396',  
 '#FEB019',  
 '#FF4560',  
 '#775DD0',  
 '#546E7A',  
 '#26a69a',  
 '#D10CE8'  
 ],  
 fontSize: '12px'  
 }  
 }  
 }  
 };  
 }  
}

dashboard.component.html

<form [formGroup]="dashboardForm" class="container">  
 <div class="row d-flex align-items-baseline">  
 <div class="col-sm">  
 <h2 i18n>Dashboard</h2>  
 </div>  
  
 <div class="col-3">  
 <sbb-form-field class="sbb-form-field-long">  
 <sbb-select formControlName="studies" placeholder="Auswahl Studie">  
 <sbb-option *\*ngFor*="let study of studies"  
 [value]="study.studyId">{{ study.name }}</sbb-option>  
 </sbb-select>  
 </sbb-form-field>  
 </div>  
 </div>  
  
 <div class="row line">  
 <h3 i18n>Statistiken von {{ title }}</h3>  
 </div>  
  
 <div class="row line">  
 <div class="col">  
 <div class="row">  
 <h4 i18n>Studiendauer</h4>  
 <sbb-icon svgIcon="kom:calendar-medium"></sbb-icon>  
 </div>  
 <div class="row d-flex align-items-center">  
 <div class="col">  
 <p i18n>  
 Beginn: {{ studyStatistics.studyDetails.length < 2 ? studyStatistics.studyDetails[0].startDate : '' }}</p>  
 <p i18n>Ende: {{ studyStatistics.studyDetails.length < 2 ? studyStatistics.studyDetails[0].endDate : '' }}</p>  
 </div>  
 </div>  
 </div>  
  
 <div class="col">  
 <div class="row">  
 <h4 i18n>Aufteilung Zugklassen</h4>  
 <sbb-icon svgIcon="kom:tickets-class-medium"></sbb-icon>  
 </div>  
 <div class="row d-flex align-items-center">  
 <div>  
 <p i18n>1. Klasse Tickets: {{ studyStatistics.trainClass }}%</p>  
 </div>  
 </div>  
 </div>  
  
 <div class="col">  
 <div class="row">  
 <h4 i18n>Aufteilung Geschlecht</h4>  
 <sbb-icon svgIcon="kom:toilet-medium"></sbb-icon>  
 </div>  
 <div class="row d-flex align-items-center">  
 <div>  
 <p i18n>Anzahl Männer: {{ studyStatistics.gender.male }}</p>  
 <p i18n>Anzahl Frauen: {{ studyStatistics.gender.female }}</p>  
 </div>  
 </div>  
 </div>  
  
 <div class="col">  
 <div class="row">  
 <h4 i18n>Häufigster Reisegrund</h4>  
 <sbb-icon svgIcon="kom:switzerland-route-medium"></sbb-icon>  
 </div>  
 <div class="row d-flex align-items-center">  
 <div>  
 <p i18n>{{ studyStatistics.journeyRating.mostFrequentJourneyReason }}</p>  
 </div>  
 </div>  
 </div>  
  
 <div class="col">  
 <div class="row">  
 <h4 i18n>∅ Zufriedenheit</h4>  
 <sbb-icon svgIcon="kom:face-grinning-medium"></sbb-icon>  
 </div>  
 <div class="row d-flex align-items-center">  
 <div>  
 <p i18n>Bewertung {{ studyStatistics.journeyRating.overallHappinessFactor }}</p>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="row diagrams">  
 <div class="col-sm">  
 <p i18n>Anzahl Teilnehmende pro Altersklasse</p>  
 <app-agegroup [ageGroup]="studyStatistics.ageGroup" class="col"></app-agegroup>  
 </div>  
  
 <div class="col-sm">  
 <p i18n>Zufriedenheit der Reise</p>  
 <app-journeyrating [journeyRating]="studyStatistics.journeyRating" class="col"></app-journeyrating>  
 </div>  
  
 <div class="col-sm">  
 <p i18n>Verteilung der Abonnemente</p>  
 <app-abotype [aboType]="studyStatistics.aboType" class="col"></app-abotype>  
 </div>  
 </div>  
  
 <div class="col-12">  
 <sbb-expansion-panel>  
 <sbb-expansion-panel-header i18n>Bewertungen der einzelnen Touchpoints</sbb-expansion-panel-header>  
 <ng-container [formGroup]="touchpointFilterForm">  
 <table [dataSource]="dataSource" sbbSort sbbTable>  
 <ng-container *\*ngFor*="let column of displayedColumns" [sbbColumnDef]="column.name">  
 <th *\*sbbHeaderCellDef* [sbbSortHeader]="column.name" sbbHeaderCell style="width: 20%">  
 {{ column.label }}  
 </th>  
 <td *\*sbbCellDef*="let element" sbbCell>{{ element[column.name] }}</td>  
 </ng-container>  
  
 <ng-container sbbColumnDef="filter-touchpoints">  
 <th *\*sbbHeaderCellDef* class="sbb-table-filter" sbbHeaderCell>  
 <input formControlName="type"/>  
 </th>  
 </ng-container>  
  
 <ng-container sbbColumnDef="empty">  
 <th *\*sbbHeaderCellDef* sbbHeaderCell></th>  
 </ng-container>  
  
 <tr *\*sbbHeaderRowDef*="['type', 'rating']" sbbHeaderRow></tr>  
 <tr *\*sbbHeaderRowDef*="['filter-touchpoints', 'empty']" sbbHeaderRow></tr>  
 <tr *\*sbbRowDef*="let row; columns: ['type', 'rating']" sbbRow></tr>  
 </table>  
 </ng-container>  
  
 <p *\*ngIf*="dataSource?.filteredData.length === 0" i18n>Keine Touchpoints mit Eingabe gefunden!</p>  
  
 <sbb-paginator [pageSize]="20" class="d-flex justify-content-center"></sbb-paginator>  
 </sbb-expansion-panel>  
 </div>  
</form>  
  
<sbb-loading *\*ngIf*="showLoading"  
 aria-valuetext="Loading, please wait"  
 mode="fullscreen"  
></sbb-loading>

dashboard.component.scss

.line {  
 h3, div.col {  
 margin-bottom: 15px;  
 }  
  
 border-bottom: 1px solid #DCDCDC;  
}  
  
.diagrams {  
 margin-top: 42px;  
}  
  
sbb-icon {  
 display: flex;  
 align-items: flex-end;  
 padding: 22px 0 0 10px;  
}

dashboard.component.ts

import {AfterViewInit, Component, OnDestroy, OnInit, ViewChild} from '@angular/core';  
import {DashboardService} from '../dashboard.service';  
import {Dashboard} from '../dashboard.model';  
import {ActivatedRoute, Router} from '@angular/router';  
import {FormBuilder, FormControl, FormGroup} from '@angular/forms';  
import {StudyDetails} from '../studydetails.model';  
import {SbbSortDirective, SbbTable, SbbTableDataSource, SbbTableFilter,} from '@sbb-esta/angular-business/table';  
import {SbbPaginatorComponent} from '@sbb-esta/angular-business/pagination';  
import {Subject} from 'rxjs';  
import {TouchpointRating} from '../touchpoint-rating.model';  
import {*takeUntil*} from 'rxjs/operators';  
  
interface TouchpointFilter extends SbbTableFilter {  
 type: string  
}  
  
@Component({  
 selector: 'app-dashboard',  
 templateUrl: './dashboard.component.html',  
 styleUrls: ['./dashboard.component.scss']  
})  
export class DashboardComponent implements OnInit, AfterViewInit, OnDestroy {  
 @ViewChild(SbbPaginatorComponent) paginator: SbbPaginatorComponent;  
 @ViewChild(SbbSortDirective) sort: SbbSortDirective;  
 @ViewChild(SbbTable) table: SbbTable<TouchpointRating>;  
  
 dashboardForm: FormGroup;  
 studyStatistics: Dashboard = <Dashboard> {};  
 studies: StudyDetails[];  
 studyId: number;  
 title: string = $localize`Studien`;  
 showLoading: boolean = false;  
  
 displayedColumns: any[] = [  
 {  
 name: 'type',  
 label: $localize`Touchpoint`  
 },  
 {  
 name: 'rating',  
 label: $localize`Bewertung`  
 }  
 ];  
  
 dataSource = new SbbTableDataSource<TouchpointRating, TouchpointFilter>();  
 touchpointFilterForm = new FormGroup({  
 type: new FormControl()  
 });  
  
 private \_destroyed = new Subject<void>();  
  
 constructor(private dashboardService: DashboardService, private route: ActivatedRoute, private fb: FormBuilder, private router: Router) {  
 }  
  
 ngOnInit(): void {  
 this.showLoading = true;  
 this.studyId = +this.route.snapshot.params['studyId'];  
 this.initDashboard();  
  
 if (this.studyId) {  
 this.getStatisticsOfStudy();  
 } else {  
 this.getAllStatistics();  
 }  
  
 this.dashboardService.getAllStudies().subscribe(studies => {  
 this.studies = studies;  
 });  
 }  
  
 ngAfterViewInit() {  
 this.initDataSource();  
 }  
  
 ngOnDestroy(): void {  
 this.\_destroyed.next();  
 }  
  
 private getAllStatistics() {  
 this.dashboardService.getStatisticsOfAllStudies().subscribe((dashboard: Dashboard) => {  
 this.setDashboardValue(dashboard);  
 });  
 }  
  
 private getStatisticsOfStudy() {  
 this.dashboardService.getStatisticsOfStudy(this.studyId).subscribe((dashboard: Dashboard) => {  
 this.title = $localize`Studie «${dashboard.studyDetails[0].name}»`;  
 this.setDashboardValue(dashboard);  
 });  
 }  
  
 private initDataSource() {  
 this.dataSource.paginator = this.paginator;  
 this.dataSource.sort = this.sort;  
 this.table.dataSource = this.dataSource;  
 this.touchpointFilterForm.valueChanges  
 .pipe(takeUntil(this.\_destroyed))  
 .subscribe((touchpointFilterForm: TouchpointFilter) => {  
 this.dataSource.filter = touchpointFilterForm;  
 });  
 }  
  
 private setDashboardValue(dashboard: Dashboard) {  
 this.dashboardForm.get('studies').valueChanges.subscribe((study) => {  
 this.router.navigateByUrl(`dashboard/${study}`);  
 this.ngOnInit();  
 });  
  
 this.studyStatistics.studyDetails = dashboard.studyDetails;  
 this.studyStatistics.gender = dashboard.gender;  
 this.studyStatistics.ageGroup = dashboard.ageGroup;  
 this.studyStatistics.aboType = dashboard.aboType;  
 this.studyStatistics.trainClass = dashboard.trainClass;  
 this.studyStatistics.journeyRating = dashboard.journeyRating;  
 this.studyStatistics.touchpointRatings = dashboard.touchpointRatings;  
  
 this.dataSource = new SbbTableDataSource<TouchpointRating, TouchpointFilter>(this.studyStatistics.touchpointRatings);  
  
 this.initDataSource();  
 this.showLoading = false;  
 }  
  
 private initDashboard() {  
 this.dashboardForm = this.fb.group({  
 studies: ['']  
 });  
  
 this.studyStatistics = {  
 studyDetails: [{  
 studyId: 0,  
 name: '',  
 startDate: new Date(),  
 endDate: new Date()  
 }],  
 gender: {  
 male: 0,  
 female: 0  
 },  
 ageGroup: {  
 babyBoomer: 0,  
 genX: 0,  
 millennial: 0,  
 genZ: 0  
 },  
 aboType: {  
 aboGa: 0,  
 aboHalf: 0,  
 aboRegional: 0,  
 aboP2p: 0,  
 aboOther: 0  
 },  
 trainClass: 0,  
 journeyRating: {  
 mostFrequentJourneyReason: '',  
 totalJourneys: 0,  
 overallHappinessFactor: 0,  
 totalHappinessFactorAwesome: 0,  
 totalHappinessFactorGood: 0,  
 totalHappinessFactorOk: 0,  
 totalHappinessFactorBad: 0,  
 totalHappinessFactorWorst: 0  
 },  
 touchpointRatings: [{  
 type: '',  
 rating: 0  
 }]  
 };  
 }  
}

abotype.model.ts

export interface AboType {  
 aboGa: number;  
 aboHalf: number;  
 aboRegional: number;  
 aboP2p: number;  
 aboOther: number;  
}

agegroup.model.ts

export interface AgeGroup {  
 babyBoomer: number;  
 genX: number;  
 millennial: number;  
 genZ: number;  
}

chart-colors.ts

export const ChartColors = [  
 '#EB0000',  
 '#F27E00',  
 '#FFDE15',  
 '#00973B',  
 '#2d327d'  
];

dashboard-routing.module.ts

import {NgModule} from '@angular/core';  
import {RouterModule, Routes} from '@angular/router';  
import {DashboardComponent} from './dashboard/dashboard.component';  
  
const routes: Routes = [  
 {  
 path: '',  
 component: DashboardComponent,  
 },  
 {  
 path: ':studyId',  
 component: DashboardComponent,  
 }  
];  
  
@NgModule({  
 imports: [RouterModule.*forChild*(routes)],  
 exports: [RouterModule],  
})  
export class DashboardRoutingModule {  
}

dashboard.model.ts

import {StudyDetails} from './studydetails.model';  
import {Gender} from './gender.model';  
import {AgeGroup} from './agegroup.model';  
import {AboType} from './abotype.model';  
import {JourneyRating} from './journey-rating.model';  
import {TouchpointRating} from './touchpoint-rating.model';  
  
export interface Dashboard {  
 studyDetails: StudyDetails[];  
 gender: Gender;  
 ageGroup: AgeGroup;  
 aboType: AboType;  
 trainClass: number;  
 journeyRating: JourneyRating;  
 touchpointRatings: TouchpointRating[];  
}

dashboard.module.ts

import {NgModule} from '@angular/core';  
import {DashboardComponent} from './dashboard/dashboard.component';  
import {DashboardRoutingModule} from './dashboard-routing.module';  
import {SharedModule} from '../shared/shared.module';  
import {SbbAccordionModule, SbbAutocompleteModule, SbbSelectModule} from '@sbb-esta/angular-business';  
import {NgApexchartsModule} from 'ng-apexcharts';  
import {AgegroupComponent} from './charts/agegroup/agegroup.component';  
import {JourneyratingComponent} from './charts/journeyrating/journeyrating.component';  
import {AbotypeComponent} from './charts/abotype/abotype.component';  
  
  
@NgModule({  
 declarations: [  
 DashboardComponent,  
 AgegroupComponent,  
 JourneyratingComponent,  
 AbotypeComponent  
 ],  
 imports: [  
 SharedModule, DashboardRoutingModule, SbbSelectModule, NgApexchartsModule, SbbAccordionModule, SbbAutocompleteModule  
 ]  
})  
export class DashboardModule {  
}

dashboard.service.ts

import {Injectable} from '@angular/core';  
import {HttpClient} from '@angular/common/http';  
import {Observable} from 'rxjs';  
import {environment} from '../../environments/environment';  
import {Dashboard} from './dashboard.model';  
import {StudyDetails} from './studydetails.model';  
  
@Injectable({  
 providedIn: 'root'  
})  
export class DashboardService {  
  
 constructor(private httpClient: HttpClient) {  
 }  
  
 getStatisticsOfAllStudies(): Observable<Dashboard> {  
 return this.httpClient  
 .get<Dashboard>(`${environment.backendUrl}/dashboard`);  
 }  
  
 getStatisticsOfStudy(studyId: number): Observable<Dashboard> {  
 return this.httpClient  
 .get<Dashboard>(`${environment.backendUrl}/dashboard/${studyId}`);  
 }  
  
 getAllStudies(): Observable<StudyDetails[]> {  
 return this.httpClient  
 .get<StudyDetails[]>(`${environment.backendUrl}/dashboard/studies`);  
 }  
}

gender.model.ts

export interface Gender {  
 male: number;  
 female: number;  
}

journey-rating.model.ts

export interface JourneyRating {  
 mostFrequentJourneyReason: string;  
 totalJourneys: number;  
 overallHappinessFactor: number;  
 totalHappinessFactorAwesome: number;  
 totalHappinessFactorGood: number;  
 totalHappinessFactorOk: number;  
 totalHappinessFactorBad: number;  
 totalHappinessFactorWorst: number;  
}

studydetails.model.ts

export interface StudyDetails {  
 studyId: number;  
 name: string;  
 startDate: Date;  
 endDate: Date;  
}

touchpoint-rating.model.ts

export interface TouchpointRating {  
 type: string;  
 rating: number;  
}

header.component.html

<sbb-header [environmentColor]="'red'" [environment]="stage !== 'prod' ? stage : ''" [label]="'Admintool SBB go'"  
 i18n-label>  
 <a class="nav" i18n routerLink="studies" routerLinkActive="sbb-active">Studien</a>  
 <a class="nav" i18n routerLink="touchpoints" routerLinkActive="sbb-active">Touchpoints</a>  
 <a class="nav" i18n routerLink="dashboard" routerLinkActive="sbb-active">Dashboard</a>  
  
 <sbb-usermenu  
 [displayName]="this.authService.name"  
 [userName]="this.authService.username">  
  
  
 <a href="/de/" sbb-usermenu-item>Deutsch</a>  
 <a href="/fr/" sbb-usermenu-item>Français</a>  
 <a href="/it/" sbb-usermenu-item>Italiano</a>  
 <a href="/en/" sbb-usermenu-item>English</a>  
  
 </sbb-usermenu>  
</sbb-header>

app-routing.module.ts

import {NgModule} from '@angular/core';  
import {RouterModule, Routes} from '@angular/router';  
import {AuthGuard} from './auth/auth-guard';  
  
*// Use the AuthGuard in routes that should require a logged in user.  
// Do NOT use it for the root route. If the user should always be logged in,  
// see comment in the AppComponent constructor.*const routes: Routes = [  
 {  
 path: 'studies',  
 loadChildren: () =>  
 import('./studies/studies.module').then((m) => m.StudiesModule),  
 canActivate: [AuthGuard],  
 },  
 {  
 path: 'touchpoints',  
 loadChildren: () =>  
 import('./analytics/analytics.module').then((m) => m.AnalyticsModule),  
 canActivate: [AuthGuard],  
 },  
 {  
 path: 'dashboard',  
 loadChildren: () =>  
 import('./dashboard/dashboard.module').then((m) => m.DashboardModule),  
 canActivate: [AuthGuard],  
 },  
 {  
 path: '',  
 pathMatch: 'full',  
 redirectTo: 'studies',  
 },  
];  
  
@NgModule({  
 imports: [RouterModule.*forRoot*(routes, {relativeLinkResolution: 'legacy'})],  
 exports: [RouterModule],  
})  
export class AppRoutingModule {  
}

## Backend

DashbaordController.java

package ch.sbb.kd.kom.sbbgo.controller.admin;  
  
import ch.sbb.kd.kom.sbbgo.service.DashboardService;  
import ch.sbb.kd.kom.sbbgo.service.dto.DashboardDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.tags.Tag;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.Set;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@RestController  
@Tag(name = "Dashboard")  
@RequestMapping(path = "api/v1/dashboard")  
public class DashboardController {  
  
 private final DashboardService dashboardService;  
  
 @Autowired  
 public DashboardController(DashboardService dashboardService) {  
 this.dashboardService = dashboardService;  
 }  
  
 */\*\*  
 \* @return Returns DashboardDto with statistics of all studies  
 \*\*/* @GetMapping  
 @Operation(summary = "Returns statistics of all studies")  
 public DashboardDto getStatisticsOfAllStudies() {  
 return dashboardService.getStatisticsAllOfStudies();  
 }  
  
 */\*\*  
 \* @return Returns DashboardDto with statistics of a single studies  
 \*\*/* @GetMapping("/{studyId}")  
 @Operation(summary = "Returns statistics of one study")  
 public DashboardDto getStatisticsOfStudy(@PathVariable("studyId") Long studyId) {  
 return dashboardService.getStatisticsOfStudy(studyId);  
 }  
  
 */\*\*  
 \* @return Returns a Set of StudyDetailsDto for select of every study in frontend  
 \*\*/* @GetMapping("/studies")  
 @Operation(summary = "Returns all studies")  
 public Set<StudyDetailsDto> getAllStudies() {  
 return dashboardService.getAllStudies();  
 }  
}

DashboardControllerTest.java

package ch.sbb.kd.kom.sbbgo.controller;  
  
  
import ch.sbb.kd.kom.sbbgo.controller.admin.DashboardController;  
import ch.sbb.kd.kom.sbbgo.service.DashboardService;  
import ch.sbb.kd.kom.sbbgo.service.dto.AboType;  
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;  
import ch.sbb.kd.kom.sbbgo.service.dto.DashboardDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;  
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;  
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import lombok.extern.slf4j.Slf4j;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.springframework.test.web.servlet.MockMvc;  
import org.springframework.test.web.servlet.setup.MockMvcBuilders;  
  
import java.util.List;  
import java.util.Set;  
  
import static org.hamcrest.Matchers.*containsString*;  
import static org.mockito.Mockito.doReturn;  
import static org.mockito.Mockito.mock;  
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;  
  
@Slf4j  
class DashboardControllerTest {  
  
 private final ObjectMapper objectMapper = new ObjectMapper();  
 private MockMvc mvc;  
 private DashboardService dashboardServiceMock;  
  
 @BeforeEach  
 void beforeEach() {  
 dashboardServiceMock = mock(DashboardService.class);  
  
 mvc = MockMvcBuilders.standaloneSetup(new DashboardController(dashboardServiceMock)).build();  
 }  
  
 @Test  
 void getStatisticsOfAllStudies\_returns200() throws Exception {  
 doReturn(DashboardDto.from(  
 Set.of(StudyDetailsDto.from(mock(TouchpointDto.class))),  
 Gender.from(10, 10),  
 AgeGroup.from(5, 3, 5, 1),  
 AboType.from(10, 5, 1, 6, 0),  
 50.0,  
 JourneyRating.from("Freizeit", 10, 4, 4, 3, 2, 5, 1),  
 List.of(new TouchpointRating())  
 )).when(dashboardServiceMock).getStatisticsAllOfStudies();  
  
 mvc.perform(get("/api/v1/dashboard")).andExpect(status().isOk())  
 .andExpect(content().string(containsString("Freizeit")));  
  
 }  
  
 @Test  
 void getAllStudies\_returns200() throws Exception {  
 doReturn(Set.of(StudyDetailsDto.from(mock(TouchpointDto.class)))).when(dashboardServiceMock).getAllStudies();  
  
 String expectedResult = objectMapper.writeValueAsString(Set.of(StudyDetailsDto.from(mock(TouchpointDto.class))));  
  
 mvc.perform(get("/api/v1/dashboard/studies")).andExpect(status().isOk()).andExpect(content().string(expectedResult));  
 }  
}

AboType.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
@Builder  
public class AboType {  
  
 private int aboGa;  
 private int aboHalf;  
 private int aboRegional;  
 private int aboP2p;  
 private int aboOther;  
  
 public static AboType from(int aboGa, int aboHalf, int aboRegional, int aboP2p, int aboOther) {  
 return *builder*()  
 .aboGa(aboGa)  
 .aboHalf(aboHalf)  
 .aboRegional(aboRegional)  
 .aboP2p(aboP2p)  
 .aboOther(aboOther)  
 .build();  
 }  
}

AgeGroup.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
@Builder  
public class AgeGroup {  
  
 private int babyBoomer;  
 private int genX;  
 private int millennial;  
 private int genZ;  
  
 public static AgeGroup from(int boomer, int genX, int millennial, int genZ) {  
 return builder()  
 .babyBoomer(boomer)  
 .genX(genX)  
 .millennial(millennial)  
 .genZ(genZ)  
 .build();  
 }  
}

DashbaordDto.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
import java.util.List;  
import java.util.Set;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
@Builder  
public class DashboardDto {  
  
 private Set<StudyDetailsDto> studyDetails;  
 private Gender gender;  
 private AgeGroup ageGroup;  
 private AboType aboType;  
 private double trainClass;  
 private JourneyRating journeyRating;  
 private List<TouchpointRating> touchpointRatings;  
  
 public static DashboardDto from(Set<StudyDetailsDto> studyDetails, Gender gender, AgeGroup ageGroup, AboType aboType, double trainClass, JourneyRating journeyRating, List<TouchpointRating> touchpointRatings) {  
 return *builder*()  
 .studyDetails(studyDetails)  
 .gender(gender)  
 .ageGroup(ageGroup)  
 .aboType(aboType)  
 .trainClass(trainClass)  
 .journeyRating(journeyRating)  
 .touchpointRatings(touchpointRatings)  
 .build();  
 }  
}

Gender.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
@Builder  
public class Gender {  
  
 private int male;  
 private int female;  
  
 public static Gender from(int male, int female) {  
 return *builder*()  
 .male(male)  
 .female(female)  
 .build();  
 }  
}

Journey.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Data;  
  
import java.util.List;  
  
@Data  
public class Journey {  
  
 private Long journeyId;  
 private List<String> journeyReasons;  
 private int overallHappinessFactor;  
}

JourneyRating.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
@Builder  
public class JourneyRating {  
  
 private String mostFrequentJourneyReason;  
 private int totalJourneys;  
 private double overallHappinessFactor;  
 private int totalHappinessFactorAwesome;  
 private int totalHappinessFactorGood;  
 private int totalHappinessFactorOk;  
 private int totalHappinessFactorBad;  
 private int totalHappinessFactorWorst;  
  
 public static JourneyRating from(String mostFrequentJourneyReason, int totalJourneys, double overallHappinessFactor, int totalHappinessFactorAwesome, int totalHappinessFactorGood, int totalHappinessFactorOk, int totalHappinessFactorBad, int totalHappinessFactorWorst) {  
 return *builder*()  
 .mostFrequentJourneyReason(mostFrequentJourneyReason)  
 .totalJourneys(totalJourneys)  
 .overallHappinessFactor(overallHappinessFactor)  
 .totalHappinessFactorAwesome(totalHappinessFactorAwesome)  
 .totalHappinessFactorGood(totalHappinessFactorGood)  
 .totalHappinessFactorOk(totalHappinessFactorOk)  
 .totalHappinessFactorBad(totalHappinessFactorBad)  
 .totalHappinessFactorWorst(totalHappinessFactorWorst)  
 .build();  
 }  
}

StudyDetailsDto.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
  
import lombok.Builder;  
import lombok.Data;  
  
import java.util.Date;  
  
@Data  
@Builder  
public class StudyDetailsDto {  
  
 private Long studyId;  
 private String name;  
 private Date startDate;  
 private Date endDate;  
  
 public static StudyDetailsDto from(TouchpointDto touchpoint) {  
 return *builder*()  
 .studyId(touchpoint.getStudyId())  
 .name(touchpoint.getName())  
 .startDate(touchpoint.getStartDate())  
 .endDate(touchpoint.getEndDate())  
 .build();  
 }  
}

TouchpointRating.java

package ch.sbb.kd.kom.sbbgo.service.dto;  
  
import lombok.Data;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Data  
public class TouchpointRating {  
  
 private String type;  
 private double rating;  
}

DashboardService.java

package ch.sbb.kd.kom.sbbgo.service;  
  
import ch.sbb.kd.kom.sbbgo.service.dto.AboType;  
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;  
import ch.sbb.kd.kom.sbbgo.service.dto.DashboardDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;  
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;  
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;  
import ch.sbb.kd.kom.sbbgo.service.dto.SearchParamsDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Map;  
import java.util.Optional;  
import java.util.Set;  
import java.util.stream.Collectors;  
  
*/\*\*  
 \* Copyright (C) Schweizerische Bundesbahnen SBB, 2021.  
 \*  
 \* @author E502439 (Winkler Olivier)  
 \* @since January 2021.  
 \*/*@Service  
public class DashboardService {  
  
 private final TouchpointService touchpointService;  
  
 private List<String> genders;  
 private List<Map<String, Object>> aboTypes;  
 private List<Integer> ageGroups;  
 private List<String> trainClass;  
 private List<TouchpointRating> touchpointRatings;  
 private List<Journey> journeys;  
  
 @Autowired  
 public DashboardService(TouchpointService touchpointService) {  
 this.touchpointService = touchpointService;  
 }  
  
 private void initCalculation() {  
 genders = new ArrayList<>();  
 aboTypes = new ArrayList<>();  
 ageGroups = new ArrayList<>();  
 trainClass = new ArrayList<>();  
 touchpointRatings = new ArrayList<>();  
 journeys = new ArrayList<>();  
 }  
  
 public DashboardDto getStatisticsAllOfStudies() {  
 this.initCalculation();  
  
 List<TouchpointDto> touchpointDtos = this.touchpointService.searchTouchpoints(new SearchParamsDto());  
  
 this.iterateOverTouchpoints(touchpointDtos);  
  
 return DashboardDto.from(  
 getStudyDetails(touchpointDtos),  
 getGender(genders),  
 getAgeGroup(ageGroups),  
 getAboType(aboTypes),  
 getTrainClass(trainClass),  
 getJourneyRating(journeys),  
 touchpointRatings  
 );  
 }  
  
 public DashboardDto getStatisticsOfStudy(Long studyId) {  
 this.initCalculation();  
  
 SearchParamsDto searchParamsDto = new SearchParamsDto();  
 searchParamsDto.setStudyId(studyId);  
  
 List<TouchpointDto> touchpointDtos = this.touchpointService.searchTouchpoints(searchParamsDto);  
  
 this.iterateOverTouchpoints(touchpointDtos);  
  
 return DashboardDto.from(  
 getStudyDetails(touchpointDtos),  
 getGender(genders),  
 getAgeGroup(ageGroups),  
 getAboType(aboTypes),  
 getTrainClass(trainClass),  
 getJourneyRating(journeys),  
 touchpointRatings  
 );  
 }  
  
 public Set<StudyDetailsDto> getAllStudies() {  
 Set<StudyDetailsDto> studyDetailsDtos = new HashSet<>();  
  
 List<TouchpointDto> touchpointDtos = this.touchpointService.searchTouchpoints(new SearchParamsDto());  
  
 touchpointDtos.forEach(touchpointDto -> studyDetailsDtos.add(  
 StudyDetailsDto.from(touchpointDto)  
 ));  
  
 return studyDetailsDtos;  
 }  
  
 private void iterateOverTouchpoints(List<TouchpointDto> touchpointDtos) {  
 for (TouchpointDto touchpoint : touchpointDtos) {  
  
 if (touchpoint.getCodingName() != null && touchpoint.getCodingId() != null) {  
 touchpointRatings.add(this.setTouchpointRating(touchpoint));  
 }  
  
 genders.add(touchpoint.getGender());  
  
 ageGroups.add(touchpoint.getAgeGroup());  
  
 HashMap<String, Object> aboType = new HashMap<>();  
 aboType.put("aboGa", touchpoint.isHasAboGa());  
 aboType.put("aboHalf", touchpoint.isHasAboHalf());  
 aboType.put("aboRegional", touchpoint.isHasAboRegional());  
 aboType.put("aboP2p", touchpoint.isHasAboP2p());  
 aboType.put("aboOther", touchpoint.getAboOther());  
 aboTypes.add(aboType);  
  
 trainClass.add(touchpoint.getTrainClass());  
  
 journeys.add(this.setJourney(touchpoint));  
 }  
 }  
  
 public TouchpointRating setTouchpointRating(TouchpointDto touchpoint) {  
 TouchpointRating touchpointRating = new TouchpointRating();  
 touchpointRating.setRating(touchpoint.getHappinessFactor());  
 touchpointRating.setType(touchpoint.getCodingName());  
 return touchpointRating;  
 }  
  
 public Journey setJourney(TouchpointDto touchpoint) {  
 Journey journey = new Journey();  
 journey.setJourneyId(touchpoint.getJourneyId());  
 journey.setJourneyReasons(touchpoint.getJourneyReasons().entrySet().stream().filter(Map.Entry::getValue).map(Map.Entry::getKey).collect(Collectors.toList()));  
 journey.setOverallHappinessFactor(touchpoint.getOverallHappinessFactor());  
 return journey;  
 }  
  
 public Set<StudyDetailsDto> getStudyDetails(List<TouchpointDto> touchpointDtos) {  
 Set<StudyDetailsDto> studyDetailsDtos = new HashSet<>();  
  
 touchpointDtos.forEach(touchpointDto -> {  
 studyDetailsDtos.add(StudyDetailsDto.from(touchpointDto));  
 });  
  
 return studyDetailsDtos;  
 }  
  
 public Gender getGender(List<String> genders) {  
 int male = (int) genders.stream().filter(gender -> gender.equals("male")).count();  
 int female = genders.size() - male;  
  
 return Gender.from(male, female);  
 }  
  
 public AgeGroup getAgeGroup(List<Integer> ageGroups) {  
 int boomer = 0, genX = 0, millenial = 0, genZ = 0;  
  
 for (Integer age : ageGroups) {  
 if (age <= 1946) boomer++;  
 else if (age <= 1976) genX++;  
 else if (age <= 1995) millenial++;  
 else if (age <= 2010) genZ++;  
 }  
  
 return AgeGroup.from(boomer, genX, millenial, genZ);  
 }  
  
 public AboType getAboType(List<Map<String, Object>> aboTypes) {  
 int aboGa = 0, aboHalf = 0, aboRegional = 0, aboP2p = 0, aboOther = 0;  
  
 for (Map<String, Object> aboType : aboTypes) {  
 Map<String, Object> result = aboType.entrySet()  
 .stream()  
 .filter(map -> map.getValue().equals(true))  
 .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));  
  
 for (String s : result.keySet()) {  
 if (result.get(s).equals(true)) {  
 switch (s) {  
 case "aboGa":  
 aboGa++;  
 break;  
 case "aboHalf":  
 aboHalf++;  
 break;  
 case "aboRegional":  
 aboRegional++;  
 break;  
 case "aboP2p":  
 aboP2p++;  
 break;  
 case "aboOther":  
 aboOther++;  
 break;  
 default:  
 break;  
 }  
 }  
 }  
 }  
  
 return AboType.from(aboGa, aboHalf, aboRegional, aboP2p, aboOther);  
 }  
  
 public double getTrainClass(List<String> trainClasses) {  
 double total = trainClasses.size();  
 double firstClass = (double) trainClasses.stream().filter(trainClass -> trainClass.equals("first")).count();  
  
 return (firstClass / total) \* 100;  
 }  
  
 public JourneyRating getJourneyRating(List<Journey> journeys) {  
 Set<Long> totalJourney = new HashSet<>();  
 double overallHappinessFactor;  
 int overallHappinessFactorJourney = 0, awesome = 0, good = 0, ok = 0, bad = 0, worst = 0;  
 List<String> journeyReasons = new ArrayList<>();  
  
 for (Journey journey : journeys) {  
 totalJourney.add(journey.getJourneyId());  
 overallHappinessFactorJourney += journey.getOverallHappinessFactor();  
 journeyReasons.addAll(journey.getJourneyReasons());  
  
 switch (journey.getOverallHappinessFactor()) {  
 case 1:  
 worst++;  
 break;  
 case 2:  
 bad++;  
 break;  
 case 3:  
 ok++;  
 break;  
 case 4:  
 good++;  
 break;  
 case 5:  
 awesome++;  
 break;  
 default:  
 break;  
 }  
 }  
  
 Map<String, Long> reasonsWithNumberOfJourneys = journeyReasons.stream().collect(Collectors.groupingBy(reason -> reason, Collectors.counting()));  
  
 Optional<Map.Entry<String, Long>> mostFrequentJourneyReason = reasonsWithNumberOfJourneys.entrySet()  
 .stream()  
 .max(Map.Entry.comparingByValue());  
  
 overallHappinessFactor = overallHappinessFactorJourney / totalJourney.size();  
  
 return JourneyRating.from(mostFrequentJourneyReason.get().getKey(), totalJourney.size(), overallHappinessFactor, awesome, good, ok, bad, worst);  
 }  
}

DashboardServiceTest.java

package ch.sbb.kd.kom.sbbgo.service;  
  
import ch.sbb.kd.kom.sbbgo.service.dto.AboType;  
import ch.sbb.kd.kom.sbbgo.service.dto.AgeGroup;  
import ch.sbb.kd.kom.sbbgo.service.dto.Gender;  
import ch.sbb.kd.kom.sbbgo.service.dto.Journey;  
import ch.sbb.kd.kom.sbbgo.service.dto.JourneyRating;  
import ch.sbb.kd.kom.sbbgo.service.dto.StudyDetailsDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointDto;  
import ch.sbb.kd.kom.sbbgo.service.dto.TouchpointRating;  
import lombok.extern.slf4j.Slf4j;  
import org.junit.jupiter.api.Test;  
  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;  
import java.util.Map;  
import java.util.Optional;  
import java.util.Set;  
import java.util.stream.Collectors;  
  
import static org.junit.jupiter.api.Assertions.assertEquals;  
import static org.mockito.Mockito.mock;  
  
@Slf4j  
class DashboardServiceTest {  
  
 @Test  
 void getTouchpointRating\_returnsTouchpointRating() {  
 TouchpointRating touchpointRating = new TouchpointRating();  
 touchpointRating.setRating(5);  
 touchpointRating.setType("Perron");  
  
 assertEquals("Perron", touchpointRating.getType());  
 }  
  
 @Test  
 void getJourney\_returnsJourney() {  
 Journey journey = new Journey();  
 journey.setJourneyId(1L);  
 journey.setJourneyReasons(List.of("Freizeitreise", "Geschäftsreise"));  
 journey.setOverallHappinessFactor(4);  
  
 assertEquals(List.of("Freizeitreise", "Geschäftsreise"), journey.getJourneyReasons());  
 }  
  
 @Test  
 void getStudyDetails\_returnsStudyDetailsDto() {  
 Set<StudyDetailsDto> studyDetailsDtos = new HashSet<>();  
  
 for (int i = 0; i < 10; i++) {  
 studyDetailsDtos.add(StudyDetailsDto.from(mock(TouchpointDto.class)));  
 }  
  
 assertEquals(1, studyDetailsDtos.size());  
 }  
  
 @Test  
 void getGenders\_returnsGender() {  
 List<String> genders = List.of("male", "male", "male", "male", "female", "female", "female", "female");  
  
 int male = (int) genders.stream().filter(gender -> gender.equals("male")).count();  
 int female = genders.size() - male;  
  
 Gender gender = Gender.from(male, female);  
  
 assertEquals(4, gender.getMale());  
 assertEquals(4, gender.getFemale());  
 }  
  
 @Test  
 void getAgeGroups\_returnsAgeGroup() {  
 List<Integer> ageGroups = List.of(1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010);  
  
 int boomer = 0, genX = 0, millenial = 0, genZ = 0;  
  
 for (Integer age : ageGroups) {  
 if (age <= 1946) boomer++;  
 else if (age <= 1976) genX++;  
 else if (age <= 1995) millenial++;  
 else if (age <= 2010) genZ++;  
 }  
  
 AgeGroup ageGroup = AgeGroup.from(boomer, genX, millenial, genZ);  
  
 assertEquals(3, ageGroup.getBabyBoomer());  
 assertEquals(3, ageGroup.getGenX());  
 assertEquals(2, ageGroup.getMillennial());  
 assertEquals(2, ageGroup.getGenZ());  
 }  
  
 @Test  
 void getAboTypes\_returnsAboType() {  
 List<String> aboTypes = List.of("aboGa", "aboGa", "aboGa", "aboGa", "aboHalf", "aboHalf", "aboHalf");  
  
 int aboGa = 0, aboHalf = 0, aboRegional = 0, aboP2p = 0, aboOther = 0;  
  
 for (String s : aboTypes) {  
 switch (s) {  
 case "aboGa":  
 aboGa++;  
 break;  
 case "aboHalf":  
 aboHalf++;  
 break;  
 case "aboRegional":  
 aboRegional++;  
 break;  
 case "aboP2p":  
 aboP2p++;  
 break;  
 case "aboOther":  
 aboOther++;  
 break;  
 default:  
 break;  
 }  
 }  
  
  
 AboType aboType = AboType.from(aboGa, aboHalf, aboRegional, aboP2p, aboOther);  
  
 assertEquals(4, aboType.getAboGa());  
 assertEquals(3, aboType.getAboHalf());  
 assertEquals(0, aboType.getAboRegional());  
 assertEquals(0, aboType.getAboP2p());  
 assertEquals(0, aboType.getAboOther());  
 }  
  
 @Test  
 void getTrainClasses\_returnsTrainClass() {  
 List<String> trainClasses = List.of("first", "first", "second", "second", "second", "second", "second", "second", "second", "second");  
  
 double total = trainClasses.size();  
 double firstClass = (double) trainClasses.stream().filter(trainClass -> trainClass.equals("first")).count();  
  
 assertEquals(20, (firstClass / total) \* 100);  
 }  
  
 @Test  
 void getJourneys\_returnsJourney() {  
 List<Journey> journeys = new ArrayList<>();  
  
 for (int i = 0; i < 5; i++) {  
 Journey journey = new Journey();  
 journey.setJourneyId((long) i);  
 journey.setJourneyReasons(List.of("Freizeit", "Freizeit", "Geschäftsreise"));  
 journey.setOverallHappinessFactor(i);  
 journeys.add(journey);  
 }  
  
 Set<Long> totalJourney = new HashSet<>();  
 double overallHappinessFactor;  
 int overallHappinessFactorJourney = 0, awesome = 0, good = 0, ok = 0, bad = 0, worst = 0;  
 List<String> journeyReasons = new ArrayList<>();  
  
 for (Journey journey : journeys) {  
 totalJourney.add(journey.getJourneyId());  
 overallHappinessFactorJourney += journey.getOverallHappinessFactor();  
 journeyReasons.addAll(journey.getJourneyReasons());  
  
 switch (journey.getOverallHappinessFactor()) {  
 case 1:  
 worst++;  
 break;  
 case 2:  
 bad++;  
 break;  
 case 3:  
 ok++;  
 break;  
 case 4:  
 good++;  
 break;  
 case 5:  
 awesome++;  
 break;  
 default:  
 break;  
 }  
 }  
  
 Map<String, Long> reasonsWithNumberOfJourneys = journeyReasons.stream().collect(Collectors.groupingBy(reason -> reason, Collectors.counting()));  
  
 Optional<Map.Entry<String, Long>> mostFrequentJourneyReason = reasonsWithNumberOfJourneys.entrySet()  
 .stream()  
 .max(Map.Entry.comparingByValue());  
  
 overallHappinessFactor = overallHappinessFactorJourney / totalJourney.size();  
  
 JourneyRating journeyRating = JourneyRating.from(mostFrequentJourneyReason.get().getKey(), totalJourney.size(), overallHappinessFactor, awesome, good, ok, bad, worst);  
  
 assertEquals(0, journeyRating.getTotalHappinessFactorAwesome());  
 assertEquals(1, journeyRating.getTotalHappinessFactorGood());  
 assertEquals(1, journeyRating.getTotalHappinessFactorOk());  
 assertEquals(1, journeyRating.getTotalHappinessFactorBad());  
 assertEquals(1, journeyRating.getTotalHappinessFactorWorst());  
 assertEquals(5, journeyRating.getTotalJourneys());  
 assertEquals(2.0, journeyRating.getOverallHappinessFactor());  
 assertEquals("Freizeit", journeyRating.getMostFrequentJourneyReason());  
 }  
}