**Instructions for creating a report**

1. Create view for report
2. Create entity for mapping view
3. On service layer realised logic for report (filters)
4. Create controller for generate response
5. On client layer create component and service
6. In component html use dev-extreme to create table with report

**Example**

For example I use ITA Tactical Plan by Group Stage. In this report we use 8 tables (academy, group\_info, profile\_info, language\_translations, technologies, employee, group\_info\_teachers, students) and five constant filters.

1. ***Create view for report***

**CREATE VIEW** tacticalreport **AS  
 SELECT** *year*(`a`.**`start\_date`**) **AS** `year`,  
 *month*(`a`.**`start\_date`**) **AS** `month`,  
 `a`.**`academy\_id` AS** `academy\_id`,  
 `gi`.**`group\_name` AS** `group\_name`,  
 `t`.**`name` AS** `cg`,  
 `pi`.**`profile\_name` AS** `profile`,  
 `lt`.**`trasnlation` AS** `location`,  
 `gi`.**`students\_planned\_to\_enrollment` AS** `requested`,  
 *`get\_student\_in\_progress`*(`a`.**`academy\_id`**) **AS** `student\_in\_progress`,  
 *`get\_student\_status`*(`a`.**`academy\_id`**, 8) **AS** `graduated`,  
 *`get\_student\_status`*(`a`.**`academy\_id`**, 9) **AS** `hired`,  
 `a`.**`start\_date` AS** `start\_date`,  
 `a`.**`end\_date` AS** `end\_date`,  
 *`get\_trainers`*(`a`.**`academy\_id`**) **AS** `trainers`,  
 `a`.**`free` AS** `payment\_status`  
 **FROM** ((((`ss\_ps\_db`.`academy` `a` **LEFT JOIN** `ss\_ps\_db`.`group\_info` `gi`  
 **ON** ((`a`.**`academy\_id`** = `gi`.**`academy\_id`**))) **LEFT JOIN** `ss\_ps\_db`.`profile\_info` `pi`  
 **ON** ((`gi`.**`profile\_id`** = `pi`.**`profile\_id`**))) **LEFT JOIN** `ss\_ps\_db`.`language\_translations` `lt`  
 **ON** ((`lt`.**`item\_id`** = `a`.**`city\_id`**))) **LEFT JOIN** `ss\_ps\_db`.`technologies` `t`  
 **ON** ((`t`.**`technology\_id`** = `a`.**`technology\_id`**)))  
 **WHERE** ((`lt`.**`local` LIKE 'en'**) **AND** (`lt`.**`tag` LIKE 'city'**))  
 **GROUP BY** `a`.**`academy\_id`**;

For generation this view I use four function *get\_student\_in\_progress, get\_student\_status, get\_trainers.*

**REATE FUNCTION** *get\_student\_in\_progress*(acid **INT**)  
 **RETURNS INT  
 BEGIN  
 RETURN** (**SELECT** *count*(s.**student\_status\_id**)  
 **FROM** students s  
 **WHERE** s.**academy\_id** = acid **AND** s.**student\_status\_id IN** (1, 2, 3));  
 **END**;

**CREATE FUNCTION** *get\_student\_status*(acid **INT**, ssid **INT**)  
 **RETURNS INT  
 BEGIN  
RETURN** (**SELECT** *count*(s.**id**)  
 **FROM** students s  
 **WHERE** s.**academy\_id** = acid **AND** s.**student\_status\_id** = ssid);  
 **END**;

**CREATE FUNCTION** *get\_trainers*(id **INT**)  
 **RETURNS VARCHAR**(100)  
 **BEGIN  
 RETURN** (**SELECT** *group\_concat*(*concat*(" ", e.**first\_name\_eng**, " ", e.**last\_name\_eng**)) **name  
 FROM** employee e  
 **JOIN** group\_info\_teachers git  
 **ON** git.**employee\_id** = e.**employee\_id  
 WHERE** git.**academy\_id** = **id AND** e.**employee\_id** = git.**employee\_id AND** git.**teacher\_type\_id IN** (1, 2));  
 **END**;

First two function I use for count of students with certain statuses. Third function I use to search and concatenation teachers and experts for group.

1. ***Create entity for mapping view***

@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@ToString  
@Entity  
@Table(name = **"tacticalreport"**)  
**public class** ItaTacticalReport {  
  
 @Id  
 @Column(name = **"academy\_id"**)  
 **private** Integer **academyId**;  
 @Column(name = **"year"**)  
 **private int year**;  
 @Column(name = **"month"**)  
 **private int month**;  
 @Column(name = **"group\_name"**)  
 **private** String **groupName**;  
 @Column(name = **"cg"**)  
 **private** String **CG**;  
 @Column(name = **"profile"**)  
 **private** String **profile**;  
 @Column(name = **"location"**)  
 **private** String **location**;  
 @Column(name = **"requested"**)  
 **private int requested**;  
 @Column(name = **"student\_in\_progress"**)  
 **private int studyInProgress**;  
 @Column(name = **"graduated"**)  
 **private int graduated**;  
 @Column(name = **"hired"**)  
 **private int hired**;  
 @Column(name = **"start\_date"**)  
 **private** Date **startDate**;  
 @Column(name = **"end\_date"**)  
 **private** Date **endDate**;  
 @Column(name = **"trainers"**)  
 **private** String **trainers**;@Column(name = **"payment\_status"**)  
 **private int paymentSatus**;  
 @Transient  
 **private** String **reportName**;  
}

In this entity I have field **private** String **reportName** with annotation @Transient. This annotation indicates that this field is not mapped to view **tacticalreport.** This field I use for indicate report on service layer.

1. Create service

@Service  
**public class** ItaTacticalReportImpl **implements** ItaTacticalReportService {  
 @Autowired  
 **private** ItaTacticalReportRepository **itaTacticalReportRepository**;  
 @Autowired  
 **private** AcademyRepository **academyRepository**;  
  
 **private final** Map<String, Predicate<Academy>> **FITERS\_FOR\_REPORTS**;  
  
 {  
 **FITERS\_FOR\_REPORTS** = **new** LinkedHashMap<>();  
 **FITERS\_FOR\_REPORTS**.put(**"List of planned launches of group (next 2 monthes)"**, a -> {  
 Calendar dateForComparison = **new** GregorianCalendar();  
 dateForComparison.add(Calendar.***MONTH***, 2);  
 Calendar academyStartDate = **new** GregorianCalendar();  
 academyStartDate.setTimeInMillis(a.getStartDate().getTime());  
 **return** a.getAcademyStages().getStageId() == ***ACADEMY\_STAGE\_PLANNED\_ID*** && academyStartDate.before(dateForComparison);  
 });  
 **FITERS\_FOR\_REPORTS**.put(**"In Process"**, a -> a.getAcademyStages().getStageId() == ***ACADEMY\_STAGE\_IN\_PROCESS\_ID***);  
 **FITERS\_FOR\_REPORTS**.put(**"Offering"**, a -> a.getAcademyStages().getStageId() == ***ACADEMY\_STAGE\_OFFERING\_ID***);  
 **FITERS\_FOR\_REPORTS**.put(**"Graduated"**, a -> a.getAcademyStages().getStageId() == ***ACADEMY\_STAGE\_GRADUATED\_ID***);  
 **FITERS\_FOR\_REPORTS**.put(**"List of planned releases"**, a -> a.getAcademyStages().getStageId() == ***ACADEMY\_STAGE\_BOARDING\_ID***);  
 }  
  
 @Override  
 **public** List<ItaTacticalReport> generateItaTacticalReport() {  
 List<ItaTacticalReport> itaTacticalReport = **new** LinkedList<>();  
 List<Academy> academies = **academyRepository**.findAll();  
 **for** (String reportName : **FITERS\_FOR\_REPORTS**.keySet()) {  
 List<Integer> ids = academies.stream()  
 .filter(**FITERS\_FOR\_REPORTS**.get(reportName))  
 .map(Academy::getAcademyId)  
 .collect(Collectors.*toList*());  
 **itaTacticalReportRepository**.findAllByAcademyIdIn(ids).stream()  
 .forEach(oneRow -> {  
 oneRow.setReportName(reportName);  
 itaTacticalReport.add(oneRow);  
 });  
 }  
 **return** itaTacticalReport;  
 }  
  
}

**FITERS\_FOR\_REPORTS** map I use to save the predicate for filters as the value and the title of the report as a key. In method *generateItaTacticalReport()* I use this map for stream filter.

1. Create controller for generate response

@RestController  
@RequestMapping(**"/reports"**)  
**public class** ControllerForReports {  
 @Autowired  
 **private** ItaTacticalReportService **itaTacticalReportService**;

@GetMapping(value = **"/ita\_tactical\_report"**)  
 **public** ResponseEntity<List<ItaTacticalReport>> getItaTacticalrReport() {  
 **return new** ResponseEntity<>(**itaTacticalReportService**.generateItaTacticalReport(), HttpStatus.***OK***);  
 }  
  
}

1. On client layer create component and service

Service:

@Injectable()  
**export class** ItaTacticalReportService {  
  
 **constructor**(**private http**: HttpClient) { }  
  
 getAll(): Observable<**any**> {  
 **return this**.**http**.get(environment.**serverUrl**+**'reports/ita\_tactical\_report'**);  
 }  
}

Component:

@Component({  
 **selector**: **'app-ita-tactical-report'**,  
 **templateUrl**: **'./ita-tactical-report.component.html'**,  
 **styleUrls**: [**'./ita-tactical-report.component.css'**]  
})  
**export class** ItaTacticalReportComponent **implements** OnInit {  
  
 **itaTacticalPlanReports** = [];  
 **loadIndicatorVisible** = **true**;  
  
 **constructor**(**private itaTacticalReportService**: ItaTacticalReportService,  
 **private router**: Router) {  
 }  
  
 calculateCellValue(data) {  
 **return** data.paymentSatus == 1 ? **"Founded by Softserve"** : **"Open group"**;  
 }  
  
 ngOnInit() {  
 **this**.**itaTacticalReportService**.getAll().subscribe(  
 data => {  
 **this**.**itaTacticalPlanReports** = data;  
 **this**.**loadIndicatorVisible** = **false**;  
 },  
 error => {  
 **if** (error.**status** === 403) {  
 **this**.**router**.navigate([**'ang/error'**]);  
 }  
 ***console***.log(error)  
 }  
 );  
 ;  
 }  
}

1. In component html use dev-extreme create table with report

First div block I use for add load indicator, property [visible] accepts boolean variable "loadIndicatorVisible", this variable equals true while load data, when the date is loaded the variable is equal to false and the load indicator is visible becomes false.

Second div I use for building table with report. For building report table I use dx-data-grid, when in property [dataSource] I specify an array from which to form a table ( itaTacticalPlanReports). The formation of this array is described in component.ts file. Tag dxo-export indicates that this table can be exported. Tag dxo-selection indicates that user can mark some fields for export. Tag dxo-paging in this tag properties you can change pagination or off pagination. Tag dxo-column-fixing indicates that column be fixing. In tag dxi-column in property dataField you will set name of variable and in property caption you will set column name, in property groupIndex you can set group index, this index uses for grouping table. Tag dxo-summary use for generate summary.

<**div id="block"**>  
 <**dx-load-indicator id="large-indicator" height="60" width="60" [visible]="loadIndicatorVisible"**></**dx-load-indicator**>  
 </**div**>  
  
  
<**div class="demo-container"**>  
 <**dx-data-grid id="gridContainer"  
 [dataSource]="itaTacticalPlanReports"  
 [allowColumnResizing]="true"  
 [visible]="!loadIndicatorVisible"**>  
 <**dxo-export [enabled]="true" fileName="report" [allowExportSelectedData]="true"**></**dxo-export**>  
 <**dxo-selection mode="multiple"**></**dxo-selection**>  
 <**dxo-group-panel [visible]="true"**></**dxo-group-panel**>  
 <**dxo-paging [enabled]="false"**></**dxo-paging**>  
 <**dxo-column-fixing [enabled]="true"**></**dxo-column-fixing**>  
  
 <**dxi-column dataField="year"  
 caption="Year"**></**dxi-column**>  
 <**dxi-column dataField="month"  
 caption="Moth"**></**dxi-column**>  
 <**dxi-column dataField="academyId"  
 caption="Group ID"**></**dxi-column**>  
 <**dxi-column dataField="groupName"  
 caption="Group/Group feedbacks"**></**dxi-column**>  
 <**dxi-column dataField="cg"  
 caption="CG"**></**dxi-column**>  
 <**dxi-column dataField="profile"  
 caption="Profile"**></**dxi-column**>  
 <**dxi-column dataField="location"  
 caption="Location"**></**dxi-column**>  
 <**dxi-column dataField="requested"  
 caption="requested"**></**dxi-column**>  
 <**dxi-column dataField="studyInProgress"  
 caption="study in progress"**></**dxi-column**>  
 <**dxi-column dataField="graduated"  
 caption="graduated"**></**dxi-column**>  
 <**dxi-column dataField="hired"  
 caption="hired"**></**dxi-column**>  
 <**dxi-column dataField="startDate"  
 caption="Group start Date"  
 dataType="date"**></**dxi-column**>  
 <**dxi-column dataField="endDate"  
 caption="Group finish Date"  
 dataType="date"**></**dxi-column**>  
 <**dxi-column dataField="trainers"  
 caption="Trainer"  
 [width]="350"**></**dxi-column**>  
 <**dxi-column dataField=""  
 aption="Comment"**></**dxi-column**>  
 <**dxi-column [calculateCellValue]="calculateCellValue"  
 capion="Payment status"**></**dxi-column**>  
 <**dxi-column dataField="reportName"  
 [groupIndex]="0"  
 aption="Report name"**></**dxi-column**>  
  
 <**dxo-summary**>  
 <**dxi-group-item  
 column="requested"  
 summaryType="sum"  
 displayFormat="{0}"  
 [alignByColumn]="true"**>  
 </**dxi-group-item**>  
 <**dxi-group-item  
 column="studyInProgress"  
 summaryType="sum"  
 displayFormat="{0}"  
 [alignByColumn]="true"**>  
 </**dxi-group-item**>  
 <**dxi-group-item  
 column="graduated"  
 summaryType="sum"  
 displayFormat="{0}"  
 [alignByColumn]="true"**>  
 </**dxi-group-item**>  
 <**dxi-group-item  
 column="hired"  
 summaryType="sum"  
 displayFormat="{0}"  
 [alignByColumn]="true"**>  
 </**dxi-group-item**>  
 </**dxo-summary**>  
  
 </**dx-data-grid**>  
</**div**>