```
1 using System;
 2 using System.Collections;
 3 using System.Collections.Generic;
 4 using System.ComponentModel;
 5 using System.IO;
 6 using System.Linq;
 7 using System.Text;
 8 using System.Threading.Tasks;
 9 using System.Windows.Forms;
10 using Nihulon2.Model;
11 using Nihulon2.Model.DbAccess;
12 using OfficeOpenXml;
13 using OfficeOpenXml.Style;
14
15 namespace Nihulon2.SupervisorsAdministration
16 {
17
        * The class that assumes the layer between the view and DataBase.
18
19
        * It provides methods that get data from the dbConnector and
        * calls the view methods to fill its controls with the new data.
        * This controller works with the exams
21
22
        */
       public class SupervisorsAdministration Controller
23
24
           private ISupervisorsAdministrationView _view; // instance of the view →
25
             for data visualization
           private DbConnector dbConn; // An instance of the class that provides →
26
              connection to the DB
27
           // The list of exams that are binded to the exam table
28
29
           private SortableBindingList<Exam> examsList;
30
           // The list with exams that have time overlaps
           private SortableBindingList<Exam> examsWithOverlap;
31
32
           #region Properties
33
34
35
            * Filter properties that define what data we need to get from the DB
36
37
38
           // When we need to show exams for a specific period of time,
39
           // this property is a date of the start of that period
40
           public string dateFromFilter
41
42
            {
43
                get;
44
               set;
45
           }
46
           // When we need to show exams for a specific period of time,
47
           // this property is a date of the start of that period
48
           public string dateToFilter
49
            {
50
                get;
51
                set;
52
           }
53
54
           // If true, show only the exams that refer to a certain division
```

```
... s Administration \backslash Supervisors Administration\_Controller.cs
 55
             public string divisionFilter
 56
             {
 57
                 get;
 58
                 set;
 59
             }
             // If true, show only the exams that refer to a certain course
 60
             public string courseFilter
 61
 62
 63
                 get;
 64
                 set;
 65
             }
             // If true, show only the exams that refer to a certain room
 66
             public string roomFilter
 67
 68
             {
 69
                 get;
 70
                 set;
 71
             }
             // If true, show the disabled exams
 72
 73
             public bool showDisabledFilter
 74
             {
 75
                 get;
 76
                 set;
             }
 77
 78
             // If true, show the exams that were added today
 79
             public bool showNewFilter
 80
 81
                 get;
 82
                 set;
 83
             }
 84
 85
             // Flag specifies in which mode to show exams, All or Overlaps only
 86
             public bool showOverlaps
 87
             {
 88
                 get;
 89
                 set;
 90
             }
 91
 92
             #endregion
 93
             // The constructor that gets the instance of the view
 94
             // that will be calling to the controller
 95
             public SupervisorsAdministration Controller
 96
               (ISupervisorsAdministrationView view)
 97
 98
                 // Initializing connection to DB
 99
                 dbConn = DbConnector.Instance;
100
                 examsList = new SortableBindingList<Exam>();
101
                 // Binding with the view
102
                 view = view;
103
                 _view.setController(this);
104
105
                 this.loadView();
106
```

dbConn.foundOverlaps = false;

107

108

109

}

```
...sAdministration\SupervisorsAdministration_Controller.cs
```

```
110
             #region Interface
111
             // Reload all comboboxes and fill the table
112
113
             public void reload()
114
             {
                 reloadFiltersComboBoxes();
115
                 fillTable();
116
117
             }
118
119
             // Get the exams from the DB and bind the list of exams with the table >
                at the view
120
             public void fillTable()
121
122
                 List<Exam> list;
123
124
                 // Show all exams according to the filters
125
                 if (!showOverlaps)
126
                 {
127
                     // Get exams form DB according to filters
                     list = dbConn.getExams(dateFromFilter, dateToFilter,
128
                       divisionFilter,
                         courseFilter, roomFilter, showDisabledFilter,
129
                         showNewFilter);
130
                     list.Reverse();
131
                     // Convert the list of exams to the Binding list
                     examsList = new SortableBindingList<Exam>(list);
132
133
                     // Ask the view to bind its table with controller's list of
134
                     _view.bindExamsWithTable(ref examsList);
135
136
                 // Show the exams that have time overlaps
137
                 else
138
                 {
139
                     // Get exams with time overlaps
                     list = dbConn.getExamsWithOverlaps();
140
                     // Convert the list of exams to the Binding list
141
                     examsList = new SortableBindingList<Exam>(list);
142
143
                     _view.bindExamsWithTable(ref examsList);
144
                 }
145
             }
146
             // Add a new exam to the DB and reload the table
147
             public void addExam(Exam newExam)
148
149
150
                 // Insert the new exam
151
                 dbConn.insertExam(newExam);
                 // Fill the table with exams
152
153
                 fillTable();
154
             }
155
             // Delete an exam from the DB
156
157
             public void deleteExam(int examId)
158
             {
159
                 dbConn.removeExam(examId);
160
             }
161
```

```
...sAdministration\SupervisorsAdministration_Controller.cs
```

```
2
```

```
162
              // Save changed exam to the DB and reload the table
163
              public void changeExam(Exam changedExam)
164
              {
165
                  dbConn.updateExam(changedExam);
                  // Check time overlaps if the flag "Show overlaps" is true
166
167
                  if(this.showOverlaps == true)
                       dbConn.markExamsWithOverlap();
168
169
                  fillTable();
170
              }
171
              // Takes a name of Excel file and create the file
172
173
              // with exam table
              public void createExcel(string excelName)
174
175
                  string filePath;
176
                  if(excelName != "")
177
178
179
                       // the path to the excel file
180
                      filePath = @"./Excel files/" + excelName + ".xlsx";
181
182
                      // Creating the directory for Excel files if doesn't exists
                       if (!Directory.Exists(@"./Excel files/"))
183
                           Directory.CreateDirectory(@"./Excel files/");
184
185
                       // Create an instance of the excelPackage, create the file and ₹
186
                          fill it with the exams
187
                      using (ExcelPackage excel = new ExcelPackage())
188
189
                           // create a workSheet
                           excel.Workbook.Worksheets.Add("Worksheet1");
190
191
                           // get the created workSheet
192
                           ExcelWorksheet excelWorksheet = excel.Workbook.Worksheets >
                           ["Worksheet1"];
193
                           // create the row of headers
194
195
                           List<string[]> headerRow = new List<string[]>()
196
                          new string[] { "תוספת זמן" } "שעת סיום", "שעת סיום" , "מקצוע", "חדר", "התייצבות ", "מסגמה/קורס", "קבוצה", "שם משגיח/ה", "חטיבה ", "בוטל", "מס", "תאריך בחינה", "שם משגיח/ה", "
197
                          מן", "זמן" },
198
                           };
199
                           // Insert header row data
                           excelWorksheet.Cells["A1"].LoadFromArrays(headerRow);
200
201
                           // Build the data for inserting from the exams at the
                           table
202
                           List<string[]> examsData = this.getExamsForExcel();
203
                           excelWorksheet.Cells["A2"].LoadFromArrays(examsData);
204
205
                           // format the workSheet
                           this.formatExcel(excelWorksheet, examsData.Count);
206
207
208
                           FileInfo excelFile = new FileInfo(filePath);
209
                           excel.SaveAs(excelFile);
210
                       }
211
                  }
```

```
...sAdministration\SupervisorsAdministration_Controller.cs
212
213
214
             // Load exams from Excel file
215
             // and save the changes into the DB
216
             public void loadFromExcel(string path)
217
                 // Create an instance of the excelPackage from the Excel file
218
219
                 using (ExcelPackage excel = new ExcelPackage(new FileInfo(path)))
220
221
                     int i;
222
                     // Get worksheet
                     ExcelWorksheet workSheet = excel.Workbook.Worksheets[1];
223
224
225
                     // Read the row of headers from the excel
                     char ch = 'A'; // Start from A
226
227
228
                     // Fill the array with headers
229
                     string[] headers = new string[14];
230
                     for(i = 1; i <= headers.Length; i++)</pre>
231
                     {
232
                         if (workSheet.Cells[$"{(char)(ch+i-1)}1"].Value != null)
                             headers[i-1] = workSheet.Cells[$"{(char)(ch + i - 1)} >
233
                         1"].Value.ToString();
234
                         else
235
                             headers[i-1] = "";
236
                     }
237
                     // Check if the headers at correct order and format
238
239
                     if (this.checkExcelFormat(headers))
240
241
                         // Save the exams from that worksheet to the DB
242
                         this.loadExamsFromExcelToDB(workSheet);
                                                                                      P
243
                     }
244
                 }
245
             }
246
247
             // Finds exams with time overlaps
248
             public void checkOverlaps()
249
             {
                 dbConn.markExamsWithOverlap();
250
251
                 this.fillTable();
252
             }
253
254
             #endregion
255
256
257
             #region Private methods for internal use
258
             // Get names of related items from DB and fill the combo boxes of the 🤝
               filters in the view
259
             private void reloadFiltersComboBoxes()
260
261
                 string[] divisions, courses, rooms;
262
```

// Get the names of related items from the DB

divisions = dbConn.getRelatedItemsNamesByType("חטיבות");

263264

```
...sAdministration\SupervisorsAdministration_Controller.cs
```

```
6
```

```
265
                 courses = dbConn.getRelatedItemsNamesByType("מגמות");
266
                 rooms = dbConn.getRelatedItemsNamesByType("חדרים");
267
268
                 // Clear combo boxes
269
                 view.clearComboBoxes();
270
                 // Fill the combo boxes with names of related items
271
                 _view.fillDivisionsCbo(divisions);
272
                 _view.fillCoursesCbo(courses);
273
274
                 _view.fillRoomsCbo(rooms);
275
             }
276
             // Set default values for the properties
277
278
             private void setDefaultFilters()
279
                 this.dateFromFilter = "";
280
                 this.dateToFilter = "";
281
                 this.divisionFilter = "הכול;
282
                 this.courseFilter = "הכול;
283
                 this.roomFilter = "הכול;
284
285
                 this.showDisabledFilter = false;
             }
286
287
288
             // Builds rows with exams for inserting into the excel file
289
             private List<string[]> getExamsForExcel()
290
291
                 List<string[]> data = new List<string[]>();
292
293
                 // Go through the exams at the table and fill the rows data
                 foreach(Exam exam in examsList)
294
295
                 {
296
                     string[] row = new string[14];
297
                     row[0] = exam.hasExtraTime ? ""; : "";
298
299
                     row[1] = exam.EndingTime;
300
                     row[2] = exam.StartTime;
301
                     row[3] = exam.room;
302
                     row[4] = exam.DisciplineName;
303
                     row[5] = exam.GroupName;
304
                     row[6] = exam.course;
305
                     row[7] = exam.division;
                     row[8] = exam.SupervisorName;
306
                     row[9] = exam.Date;
307
308
                     row[10] = exam.Id.ToString();
                     row[11] = exam.isCanceled ? "בוטל" : "";
309
310
                     row[12] = exam.Comments;
311
312
                     // Calculate difference between the start time and the ending >
313
                     // for getting the time scale of the exam
314
                     DateTime fromTime = DateTime.Parse(exam.StartTime);
315
                     DateTime toTime = DateTime.Parse(exam.EndingTime);
                     TimeSpan hours = (toTime - fromTime);
316
317
                     row[13] = hours.ToString();
318
319
                     data.Add(row);
```

```
...sAdministration\SupervisorsAdministration_Controller.cs
320
321
                 return data;
322
             }
323
324
             // Gets workSheet of Excel with exams and formats its cells
325
             private void formatExcel(ExcelWorksheet excelWorksheet, int numRows)
326
327
                 int i;
328
329
                 excelWorksheet.View.RightToLeft = true;
330
331
                 // Set width for the columns
                 excelWorksheet.Column(1).Width = 7;
332
333
                 excelWorksheet.Column(2).Width = 10;
334
                 excelWorksheet.Column(3).Width = 10;
335
                 excelWorksheet.Column(4).Width = 12;
336
                 excelWorksheet.Column(5).Width = 15;
337
                 excelWorksheet.Column(6).Width = 10;
338
                 excelWorksheet.Column(7).Width = 25;
                 excelWorksheet.Column(8).Width = 10;
339
340
                 excelWorksheet.Column(9).Width = 17;
341
                 excelWorksheet.Column(10).Width = 13;
342
                 excelWorksheet.Column(11).Hidden = true;
343
                 excelWorksheet.Column(12).Width = 7;
344
                 excelWorksheet.Column(13).Width = 25;
                 excelWorksheet.Column(14).Width = 10;
345
346
347
                 // Set wrap text
348
                 for (i = 1; i <= 14; i++)
349
                     excelWorksheet.Column(i).Style.WrapText = true;
350
351
                 // Set border style
                 excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Left.Style >
352
                    = OfficeOpenXml.Style.ExcelBorderStyle.Thin;
                 excelWorksheet.Cells[$"A1:N{numRows +
353
                                                                                      P
                   1}"].Style.Border.Right.Style =
                                                                                      P
                   OfficeOpenXml.Style.ExcelBorderStyle.Thin;
354
                 excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Top.Style
                   = OfficeOpenXml.Style.ExcelBorderStyle.Thin;
                 excelWorksheet.Cells[$"A1:N{numRows +
355
                                                                                      P
                   1}"].Style.Border.Bottom.Style =
                                                                                      P
                   OfficeOpenXml.Style.ExcelBorderStyle.Thin;
356
                 excelWorksheet.Cells[$"A1:N{numRows +
                                                                                      P
                   1}"].Style.Border.BorderAround
                                                                                      P
                   (OfficeOpenXml.Style.ExcelBorderStyle.Thick);
                 // Paint yellow the empty cells at the column with names of
357
                   supervisors
358
                 for (i = 2; i \le numRows + 1; i++)
359
                 {
360
                     if(excelWorksheet.Cells[$"I{i}"].Value == null)
361
                         excelWorksheet.Cells[$"I{i}"].Style.Fill.PatternType =
362
                         OfficeOpenXml.Style.ExcelFillStyle.Solid;
363
                         excelWorksheet.Cells[$"I
                                                                                      P
                         {i}"].Style.Fill.BackgroundColor.SetColor
                                                                                      P
                         (System.Drawing.Color.Yellow);
```

```
...sAdministration\SupervisorsAdministration_Controller.cs
364
365
                 }
366
             }
367
368
             // Checks format of the excel file
             private bool checkExcelFormat(string[] headers)
369
370
                 string[] patternHeaders = { "תוספת זמן", "תוספת ", "שעת סיום", "
371
                   "קבוצה", "מקצוע", "חדר", "התייצבות,",
372
                     "מס", "תאריך בחינה", "שם משגיח/ה", "חטיבה", "מגמה/קורס", →
                       "בוטל", "הערה", "בוטל" };
                 // Check the header row if the columns at correct order
373
374
                 for(int i = 0; i < 14; i++)
375
                     if (headers[i] != patternHeaders[i])
376
377
                         return false;
378
                 }
379
                 return true;
380
             }
381
             // Sets default filters and fill the table with exams
382
             private void loadView()
383
384
385
                 // Set default filters
                 setDefaultFilters();
386
                 // Get names of related items and fill the combo boxes of the
387
                   filters in the view
388
                 reloadFiltersComboBoxes();
389
                 // Bind the event handler that catch time overlaps with the
390
                   processing method of the view
391
                 dbConn.onOverlapsStateChanged += _view.showOverlapsWarning;
392
393
                 // Fill the table with exams
                 fillTable();
394
             }
395
396
397
             // Takes an Excel worksheet with exams data as a parameter
398
             // and saves the exams from that worksheet to the DB
399
             private void loadExamsFromExcelToDB(ExcelWorksheet workSheet)
400
                 // Array for saving id, supervisor name and comments
401
402
                 string[] examData = new string[3];
403
                 // Go through all rows at the excel file and save the changes into >
                 for (int i = 2; i <= workSheet.Dimension.End.Row; i++)</pre>
404
405
                 {
406
                     // Get data from the excel
407
408
                     if (workSheet.Cells[$"K{i}"].Value != null)
                         examData[0] = workSheet.Cells[$"K{i}"].Value.ToString();
409
410
                     else
                         examData[0] = "";
411
412
                     // Supervisor name
                     if (workSheet.Cells[$"I{i}"].Value != null)
413
```

examData[1] = workSheet.Cells[\$"I{i}"].Value.ToString();

414

```
\underline{\dots} s Administration \backslash Supervisors Administration \_Controller.cs
```

```
415
                     else
416
                         examData[1] = "";
417
                     // Comments
                     if (workSheet.Cells[$"M{i}"].Value != null)
418
419
                         examData[2] = workSheet.Cells[$"M{i}"].Value.ToString();
420
                    else
421
                         examData[2] = "";
422
                    // Save the changes into the DB
423
424
                    dbConn.updateExamFromExcel(examData);
425
                 }
426
             }
427
428
            #endregion
429
430
         }
431 }
432
```