|  |  |  |
| --- | --- | --- |
| 1 | using | System; |
| 2 | using | System.Collections.Generic; |
| 3 | using | System.ComponentModel; |
| 4 | using | System.Drawing; |
| 5 | using | System.Data; |
| 6 | using | System.Linq; |
| 7 | using | System.Text; |
| 8 | using | System.Threading.Tasks; |
| 9 | using | System.Windows.Forms; |
| 10 |  |  |
| 11 |  |  |
| 12 | using | Nihulon2.Model; |
| 13 | using | System.Collections; |
| 14 | using | DGVPrinterHelper; |
| 15 | using | System.IO; |
| 16 |  |  |

17 namespace Nihulon2.SupervisorsAdministration 18 {

19 /\*

1. \* The class of view that displays the data of exams
2. \* ISupervisorsAdministrationView - interface that provides methods to the controller
3. \* for displaying the data 23 \*/

24 public partial class SupervisorAdministration\_View : UserControl, ISupervisorsAdministrationView

25 {

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

// Instance of the controller that manages this view private SupervisorsAdministration\_Controller \_controller;

public SupervisorAdministration\_View()

{

InitializeComponent();

radioButtonShowAll.Checked = true; // set "show all exams" as a default filter

}

#region Form Events

// Adding a new exam

private void btnAddExam\_Click(object sender, EventArgs e)

{

// If all required fields at the form are filled if(cboCourse\_NewExam.SelectedIndex != -1 &&

cboDivision\_NewExam.SelectedIndex != -1 && cboRoom\_NewExam.SelectedIndex != -1)

44 {

45

46

47

48

49

50

51

52

// Create a new exam and fill its fields Exam newExam = new Exam(); newExam.course = cboCourse\_NewExam.Text; newExam.Date = DatePicker.Text;

newExam.DisciplineName = txtDiscipline\_NewExam.Text; newExam.division = cboDivision\_NewExam.Text; newExam.EndingTime = ToTimePicker.Text; newExam.GroupName = txtGroup\_NewExam.Text;

1. newExam.hasExtraTime = cbExtraTime\_NewExam.Checked;
2. newExam.room = cboRoom\_NewExam.Text;
3. newExam.StartTime = FromTimePicker.Text;
4. newExam.SupervisorName = txtSupervisor\_NewExam.Text;
5. newExam.dateOfCreation = DateTime.Today.ToString ("dd.MM.yyyy");

58

59 // Add the new exam to the list of exams and insert into the DB

|  |  |  |
| --- | --- | --- |
| 60 | \_controller.addExam(newExam); |  |
| 61 |  |
| 62 | clearAddingForm(); |
| 63 | } |
| 64 | } |
| 65 |  |
| 66 | // Clear the adding form |
| 67 | private void btnClearAddingForm\_Click(object sender, EventArgs e) |
| 68 | { |
| 69 | clearAddingForm(); |
| 70 | // Disable the button of clearing the form |
| 71 | btnClearAddingForm.Enabled = false; |
| 72 | } |
| 73 |  |
| 74 | // Create a copy of selected exam |
| 75 | private void btn\_CopyRow\_Click(object sender, EventArgs e) |
| 76 | { |
| 77 | // get selected exam as an instance of Exam |
| 78 | Exam selectedExam = dgvExamTable.CurrentRow.DataBoundItem as | Exam; |
| 79 | if(selectedExam != null) |  |
| 80 | \_controller.addExam(selectedExam); // Add exam |  |
| 81 | } |  |
| 82 |  |  |
| 83 | // Delete an exam |  |
| 84 | private void btnDeleteExam\_Click(object sender, EventArgs e) |  |
| 85 | { |  |
| 86 | int examId; |  |
| 87 | Exam exam; |  |
| 88 | // Go through all selected exams and delete them |  |
| 89 | foreach (DataGridViewRow row in dgvExamTable.SelectedRows) |  |
| 90 | { |  |
| 91 | exam = row.DataBoundItem as Exam; |  |
| 92 | if(exam != null) |  |
| 93 | { |  |
| 94 | examId = exam.Id; // get exam id from selected exam |  |
| 95 | \_controller.deleteExam(examId); // delete |  |
| 96 | } |  |
| 97 | } |  |
| 98 | // Get the updated data from the DB |  |
| 99 | \_controller.fillTable(); |  |
| 100 | } |  |
| 101 |  |  |
| 102 | // Open the form of changing an exam by click |  |
| 103 | // on the button "change exam" |  |
| 104 | private void btnChangeExam\_Click(object sender, EventArgs e) |  |
| 105 | { |  |

|  |  |
| --- | --- |
| 106 | this.callFormOfExamChanging(); |
| 107 | } |
| 108 | // Open the form of changing an exam by double click on the exam at |

the table

109 private void dgvExamTable\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

|  |  |
| --- | --- |
| 110 | { |
| 111 | this.callFormOfExamChanging(); |
| 112 | } |
| 113 |  |
| 114 | // When one of the checkboxes, "canceled" or "has extra time" changes |

its state,

1. // change the exam state and save into the DB
2. private void dgvExamTable\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

117 {

1. Exam changedExam = dgvExamTable.CurrentRow.DataBoundItem as Exam;
2. if(changedExam != null)
3. \_controller.changeExam(changedExam);

|  |  |  |
| --- | --- | --- |
| 121 | } |  |
| 122 |  |
| 123 | // | When a row in the table selected, fill textbox with the comments of |
| 124 | // | the selected exam  If there is multi-selected rows, disable the buttons of changing |
| and copying | | |
| 125 | private void dgvExamTable\_SelectionChanged(object sender, EventArgs e) | |
| 126 | { | |
| 127 | // If there is selected row at the table, fill the text box with | |

comments

128 if (dgvExamTable.CurrentRow != null)

129 {

1. Exam exam = dgvExamTable.CurrentRow.DataBoundItem as Exam;
2. if (exam != null)
3. txt\_Comments.Text = exam.Comments;

133 }

134

135

1. // If there are more than one selected row, disable the buttons
2. // of changing and copying exams
3. if(dgvExamTable.SelectedRows.Count != 1)

139 {

1. btnChangeExam.Enabled = false;
2. btn\_CopyRow.Enabled = false;

142 }

143 else

144 {

1. btnChangeExam.Enabled = true;
2. btn\_CopyRow.Enabled = true;

147 }

148 }

149

1. // Save changes after editing the comments
2. private void txt\_Comments\_Leave(object sender, EventArgs e)

152 {

1. // get selected exam
2. Exam changedExam = (dgvExamTable.CurrentRow.DataBoundItem as

Exam);

1. // If casted successful
2. if(changedExam != null)

158 {

1. changedExam.Comments = txt\_Comments.Text; // set new comments
2. \_controller.changeExam(changedExam); // change into the DB

161 }

162 }

163

164 /\*

1. \* When one of the radiobuttons at the time filters was checked,
2. \* enable the needed option and disable others 167 \*/
3. // Show all
4. private void radioButtonShowAll\_CheckedChanged(object sender, EventArgs e)

170 {

1. dateTimePickerOneDay\_Filter.Enabled = false;
2. dateTimePickerFrom\_Filter.Enabled = false;
3. dateTimePickerTo\_Filter.Enabled = false;

174 }

1. // Period
2. private void radioButtonPeriod\_CheckedChanged(object sender, EventArgs e)

177 {

1. dateTimePickerOneDay\_Filter.Enabled = false;
2. dateTimePickerFrom\_Filter.Enabled = true;
3. dateTimePickerTo\_Filter.Enabled = true;

181 }

1. // One day
2. private void radioButtonOneDay\_CheckedChanged(object sender, EventArgs e)

184 {

1. dateTimePickerOneDay\_Filter.Enabled = true;
2. dateTimePickerFrom\_Filter.Enabled = false;
3. dateTimePickerTo\_Filter.Enabled = false;

188 }

189

1. // When the button "show exams" has been pressed,
2. // set all values of the filters at the controller and ask to get the needed data

|  |  |
| --- | --- |
| 192 | private void btnShowExams\_Click(object sender, EventArgs e) |
| 193 | { |
| 194 | /\* Set the filters of period of time \*/ |
| 195 |  |
| 196 | // show all or show new |
| 197 | if (radioButtonShowAll.Checked == true || |
| 198 | radioButtonShowNew.Checked == true)  { |
| 199 | \_controller.dateFromFilter = ""; |
| 200 | \_controller.dateToFilter = ""; |
| 201 | } |
| 202 | // show exams for one day |
| 203 | else if (radioButtonOneDay.Checked == true) |
| 204 | { |

|  |  |  |
| --- | --- | --- |
| 205 |  | \_controller.dateFromFilter = dateTimePickerOneDay\_Filter.Text; |
| 206 |  | \_controller.dateToFilter = dateTimePickerOneDay\_Filter.Text; |
| 207 |  | } |
| 208 |  | // show exams for a period |
| 209 |  | else if (radioButtonPeriod.Checked == true) |
| 210 |  | { |
| 211 |  | \_controller.dateFromFilter = dateTimePickerFrom\_Filter.Text; |
| 212 |  | \_controller.dateToFilter = dateTimePickerTo\_Filter.Text; |
| 213 |  | } |
| 214 |  |  |
| 215 |  | // Set other filters from the form to the controller |
| 216 |  | \_controller.divisionFilter = cboDivisionFilter.Text; |
| 217 |  | \_controller.courseFilter = cboCourseFilter.Text; |
| 218 |  | \_controller.roomFilter = cboRoomFilter.Text; |
| 219 |  | \_controller.showDisabledFilter = cbShowDisabledExams.Checked; |
| 220 |  | \_controller.showNewFilter = radioButtonShowNew.Checked; |
| 221 |  | \_controller.showOverlaps = false; |
| 222 |  |  |
| 223 |  | // Ask the controller to fill the table according to the filters |
| 224 |  | \_controller.fillTable(); |
| 225 | } |  |
| 226 |  |  |
| 227 | // | If one of the dates at the filter form was changed, |
| 228 | // | check if the "From" date is earlier than the "To" date. |
| 229 | // | If not, set the "To" date equal to the "From" date |

230 private void dateTimePickerFilter\_ValueChanged(object sender, EventArgs e)

231 {

232 this.checkAndSetFilterDates(dateTimePickerTo\_Filter, dateTimePickerFrom\_Filter);

233 }

234

1. // The method that is called when something has been changed at the form
2. // of adding a new exam. It makes the button "clear the form" being enabled

248

|  |  |
| --- | --- |
| 237 | private void creationFormChanged(object sender, EventArgs e) |
| 238 | { |
| 239 | btnClearAddingForm.Enabled = true; |
| 240 | } |
| 241 |  |
| 242 | // Printing the Exam table |
| 243 | // The method uses DGVPrinter module that serves |
| 244 | // the printing of data Grid Views |
| 245 | private void btn\_Print\_Click(object sender, EventArgs e) |
| 246 | { |
| 247 | DGVPrinter printer = new DGVPrinter(); |
| 249 | // Settings of the printing |
| 250 | printer.HeaderCellAlignment = StringAlignment.Near; |
| 251 | printer.PageSettings.Landscape = true; |
| 252 | printer.PageSettings.Color = false; |
| 253 | printer.PageSettings.Margins.Left = 30; |
| 254 | printer.RowHeight = DGVPrinter.RowHeightSetting.CellHeight; |
| 256 | // Print |

255

257 printer.PrintDataGridView(dgvExamTable);

258 }

259

1. // Create a form for taking the name of Excel file
2. private void btn\_ExportToExcel\_Click(object sender, EventArgs e)

262 {

1. frmGetNameOfExcelFile getNameForm = new frmGetNameOfExcelFile();
2. // Event called when user entered the name of the file
3. getNameForm.onFileNameConfirm += excelNameEntered; 266
4. getNameForm.StartPosition = FormStartPosition.CenterParent;
5. getNameForm.ShowDialog();

269 }

1. // Loads exams from excel
2. private void btn\_ImportFromExcel\_Click(object sender, EventArgs e)

272 {

1. // Create a folder for Excel files if not exists
2. string location = System.Windows.Forms.Application.StartupPath;
3. location += @"\Excel files";
4. if (!Directory.Exists(location))

277 {

278 Directory.CreateDirectory(location);

279 }

280

1. // Open the file dialog from the folder with Excel files
2. OpenFileDialog openFileDialog = new OpenFileDialog();
3. openFileDialog.InitialDirectory = location;
4. openFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";
5. if (openFileDialog.ShowDialog() == DialogResult.OK && openFileDialog.CheckFileExists)

287 {

288 \_controller.loadFromExcel(openFileDialog.FileName);

289 }

290 }

291

1. // Checks each exam if it has time overlaps and paint its row red
2. private void dgvExamTable\_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)

294 {

295 foreach (DataGridViewRow dgvr in dgvExamTable.Rows)

296 {

1. if (Convert.ToBoolean(dgvr.Cells["hasOverlap"].Value))
2. dgvr.DefaultCellStyle.BackColor = Color.IndianRed;

299 }

300 }

301

1. // Show only the exams that have time overlaps
2. private void btnOverlaps\_Click(object sender, EventArgs e)

304 {

305 \_controller.showOverlaps = true; 306

307 \_controller.fillTable();

308 }

309

1. // Runs searching of exams with time overlaps
2. private void btnCheckOverlaps\_Click(object sender, EventArgs e)

312 {

313 \_controller.checkOverlaps();

314 }

315

316 #endregion 317

1. #region Interface
2. // binds the view with its controller
3. public void setController(SupervisorsAdministration\_Controller controller)

|  |  |  |
| --- | --- | --- |
| 321 | { |  |
| 322 | \_controller = controller; |  |
| 323 | } |  |
| 324 |  |  |
| 325 | // Binds the list of exams at the controller with the table at | the |

view

326 public void bindExamsWithTable(ref SortableBindingList<Exam> examsList)

327 {

1. dgvExamTable.DataSource = examsList; // bind
2. this.formatTable(); // set column sizes

330 }

331

1. // Fill the comboboxes of Divisions at the form of filters and at the form of adding
2. public void fillDivisionsCbo(string[] divisionsNames)

334 {

1. cboDivisionFilter.Items.Add("הכול");
2. // Fill the comboboxes with names

|  |  |  |  |
| --- | --- | --- | --- |
| 337 | for | | (int i = 0; i < divisionsNames.Length; i++) |
| 338 | { | |  |
| 339 |  | | cboDivisionFilter.Items.Add(divisionsNames[i]); |
| 340 |  | | cboDivision\_NewExam.Items.Add(divisionsNames[i]); |
| 341 | } | |  |
| 342 | // | | Set "show all exams" as selected |
| 343 |  | cboDivisionFilter.SelectedIndex = 0; | |
| 344 | } |  | |
| 345 | // | Fill the comboboxes of Courses at filters form and at the adding | |

form

346 public void fillCoursesCbo(string[] coursesNames)

347 {

1. cboCourseFilter.Items.Add("הכול");
2. // Fill the comboboxes with names

|  |  |  |  |
| --- | --- | --- | --- |
| 350 | for | | (int i = 0; i < coursesNames.Length; i++) |
| 351 | { | |  |
| 352 |  | | cboCourseFilter.Items.Add(coursesNames[i]); |
| 353 |  | | cboCourse\_NewExam.Items.Add(coursesNames[i]); |
| 354 | } | |  |
| 355 | // | | Set "show all exams" as selected |
| 356 |  | cboCourseFilter.SelectedIndex = 0; | |
| 357 | } |  | |
| 358 | // | Fill the comboboxes of Rooms at filters form and at the adding form | |

359 public void fillRoomsCbo(string[] roomsNames)

360 {

1. cboRoomFilter.Items.Add("הכול");
2. // Fill the comboboxes with names
3. for (int i = 0; i < roomsNames.Length; i++)

364 {

1. cboRoomFilter.Items.Add(roomsNames[i]);
2. cboRoom\_NewExam.Items.Add(roomsNames[i]);

367 }

1. // Set "show all exams" as selected
2. cboRoomFilter.SelectedIndex = 0;

370 }

371

1. // Clear all combo boxes
2. public void clearComboBoxes()

374 {

1. cboCourseFilter.Items.Clear();
2. cboDivisionFilter.Items.Clear();
3. cboRoomFilter.Items.Clear();
4. cboCourse\_NewExam.Items.Clear();
5. cboDivision\_NewExam.Items.Clear();
6. cboRoom\_NewExam.Items.Clear();

381 }

382

1. // Get updated data from the DB and reload the view
2. public void reloadView()

385 {

386 \_controller.reload();

387 }

388

1. // Set visible or hide the button and the label that show the overlaps at the table
2. public void showOverlapsWarning(bool hasOverlaps)

391 {

392 if (hasOverlaps == true && btnOverlaps.Visible == false)

393 {

1. btnOverlaps.Visible = true;
2. labelOverlaps.Visible = true;

396 }

397 else if (hasOverlaps == false)

398 {

1. btnOverlaps.Visible = false;
2. labelOverlaps.Visible = false;

401 }

402 }

403

404 #endregion 405

406 #region Private methods 407

408 // Clear txt fields and checkbox at the form for adding exams

409 private void clearAddingForm()

410 {

1. txtDiscipline\_NewExam.Clear();
2. txtGroup\_NewExam.Clear();
3. txtSupervisor\_NewExam.Clear();
4. cboRoom\_NewExam.SelectedIndex = -1;
5. cboCourse\_NewExam.SelectedIndex = -1;
6. cboDivision\_NewExam.SelectedIndex = -1; 417

418 cbExtraTime\_NewExam.Checked = false;

419 }

420

421 // Set column sizes and hide columns that user doesn't need

422 // like Id, date of creation etc.

423 private void formatTable()

424 {

1. // hide the columns that user doesn't need
2. dgvExamTable.Columns["comments"].Visible = false;
3. dgvExamTable.Columns["id"].Visible = false;
4. dgvExamTable.Columns["dateOfCreation"].Visible = false;
5. dgvExamTable.Columns["hasOverlap"].Visible = false; 430
6. // set fixed size for not textual columns
7. dgvExamTable.Columns["Date"].Width = 80;
8. dgvExamTable.Columns["isCanceled"].Width = 70;
9. dgvExamTable.Columns["hasExtraTime"].Width = 70;
10. dgvExamTable.Columns["startTime"].Width = 70;
11. dgvExamTable.Columns["endingTime"].Width = 70;
12. dgvExamTable.Columns["GroupName"].Width = 70;
13. dgvExamTable.Columns["room"].Width = 100;
14. dgvExamTable.Columns["division"].Width = 100; 440
15. // fill the entire space with textual columns
16. dgvExamTable.Columns["SupervisorName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
17. //dgvExamTable.Columns["division"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
18. dgvExamTable.Columns["course"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
19. //dgvExamTable.Columns["GroupName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
20. dgvExamTable.Columns["DisciplineName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
21. //dgvExamTable.Columns["room"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
22. // forbid editing for all columns excepting "canceled" and "has extra time"
23. dgvExamTable.Columns["Date"].ReadOnly = true;
24. dgvExamTable.Columns["startTime"].ReadOnly = true;
25. dgvExamTable.Columns["endingTime"].ReadOnly = true;
26. dgvExamTable.Columns["SupervisorName"].ReadOnly = true;
27. dgvExamTable.Columns["division"].ReadOnly = true;
28. dgvExamTable.Columns["course"].ReadOnly = true;
29. dgvExamTable.Columns["GroupName"].ReadOnly = true;
30. dgvExamTable.Columns["DisciplineName"].ReadOnly = true;
31. dgvExamTable.Columns["room"].ReadOnly = true;

459 }

460

461 // Checks if the date "From" is earlier than the "To" date

462 // If not, sets the "To" date equal to the "From" date

463 private void checkAndSetFilterDates(DateTimePicker dateTimePickerTo, DateTimePicker dateTimePickerFrom)

|  |  |  |  |
| --- | --- | --- | --- |
| 464 |  |  | { |
| 465 |  |  | if (dateTimePickerTo.Value < dateTimePickerFrom.Value) |
| 466 |  |  | { |
| 467 |  |  | dateTimePickerTo.Value = dateTimePickerFrom.Value; |
| 468 |  |  | } |
| 469 |  |  | } |
| 470 |  |  |  |
| 471 |  |  | // Event handler that is called when the user enter |
| 472 |  |  | // the name of the excel file for saving the table of exams |
| 473 |  |  | private void excelNameEntered(string excelName) |
| 474 |  |  | { |
| 475 |  |  | \_controller.createExcel(excelName); |
| 476 |  |  | } |
| 477 |  |  |  |
| 478 |  |  |  |
| 479 |  |  | // Called when one of the time pickers at the form of creating |
| 480 |  |  | // new exams has been changed. |
| 481 |  |  | // Checks if the "from" date is earlier than "to" date |
| 482 |  |  | private void TimePicker\_ValueChanged(object sender, EventArgs e) |
| 483 |  |  | { |
| 484 |  |  | checkAndSetFilterDates(ToTimePicker, FromTimePicker); |
| 485 |  |  | } |
| 486 |  |  |  |
| 487 |  |  | // Get changed exam and save the changes into the DB |
| 488 |  |  | private void saveChangedExam(Exam changedExam) |
| 489 |  |  | { |
| 490 |  |  | \_controller.changeExam(changedExam); |
| 491 |  |  | } |
| 492 |  |  |  |
| 493 |  |  | // Creates the form of changing exams, |
| 494 |  |  | // fills its controls and set the exam |
| 495 |  |  | // that is needed to be changed |
| 496 |  |  | private void callFormOfExamChanging() |
| 497 |  |  | { |
| 498 |  |  | // get selected exam as an instance of Exam |
| 499 |  |  | Exam selectedExam = (Exam)dgvExamTable.CurrentRow.DataBoundItem; |
| 501 |  |  | // Show form for changing exams |
| 502 |  |  | frmChangeExam changeForm = new frmChangeExam(); |
| 503 |  |  | // Subscribe on exam changes |
| 504 |  |  | changeForm.onSaveChanges += saveChangedExam; |
| 505 |  |  | // Fill the comboBoxes of the changeForme with related items |
| 506 |  |  | changeForm.fillComboBoxes(cboDivision\_NewExam.Items, |
| 507 |  |  | cboCourse\_NewExam.Items, cboRoom\_NewExam.Items);  // Set the exam that needed to be changed |
| 508 |  |  | changeForm.setExam(selectedExam); |
| 510 |  |  | // Show the form at the center |
| 511 |  |  | changeForm.StartPosition = FormStartPosition.CenterParent; |
| 512 |  |  | changeForm.ShowDialog(); |
| 513 |  |  | } |
| 514 |  |  |  |
| 515 |  |  | #endregion |
| 516 |  | } |  |
| 517 | } |  |  |
| 518 |  |  |  |

500

509