Πανεπιστήμιο Πειραιώς

Σχολή Πληροφορικής

Τμήμα Πληροφορικής

Ιανουάριος 2023

ΣΥΓΧΡΟΝΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ - ΛΟΓΙΣΜΙΚΟ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ

ΤΕΧΝΙΚΟ ΕΓΧΕΙΡΙΔΙΟ

Ονοματεπώνυμο: Γεωργιάδης Ευθύμιος Πάτροκλος, Π19031

ΕΙΣΑΓΩΓΗ

Η εφαρμογή αυτή εστιάζει κυρίως στην χρήση της αρχιτεκτονικής MVC για την παροχή υπηρεσιών σε ένα πανεπιστήμιο για την βαθμολόγηση των μαθητών του. Η βαθμολόγηση των μαθητών πραγματοποιείται από τους καθηγητές του πανεπιστημίου, ενώ η γενικότερη διαχείριση των μαθητών, των καθηγητών και των εξεταζόμενων μαθημάτων, θα καλύπτεται από τους υπάλληλους της γραμματείας.

Για την λειτουργία της πλατφόρμας, είναι αναγκαία η μορφοποίηση μιας βάσης δεδομένων (gradDB.bacpac). Με βάση τους πίνακες της βάσης αυτής, διαμορφώνονται και τα μοντέλα (Models) της εφαρμογής. Με την χρήση των μοντέλων, πλέον είναι δυνατή η δημιουργία και τροποποίηση των Controllers, προσδίδοντας λειτουργικότητα στην εφαρμογή και επιτρέποντας στους χρήστες να δουν αλλά και να επεξεργαστούν κάποια από τα δεδομένα των Models, μέσω των αντίστοιχων Views.

ΕΡΓΑΛΕΙΑ

Τα εργαλεία που χρησιμοποιήθηκαν για την εργασία αυτή είναι το Microsoft Visual Studio 2022, για την ανάπτυξη της κύριας εφαρμογής, το Microsoft SQL Server + Microsoft SQL Server Management Studio 2018/2019, για την δημιουργία και λειτουργικότητα της βάσης δεδομένων, καθώς και για το παραπάνω διάγραμμα της βάσης και το Mockaroo για την εμπλούτιση της βάσης με αληθοφανή δεδομένα.

ΚΥΡΙΑ ΜΟΝΤΕΛΑ

Οι χρήστες της εφαρμογής χωρίζονται σε 3 ρόλους: Μαθητές (Students), Καθηγητές (Professors), Γραμματεία (Secretaries). Τα μοντέλα τους είναι σημαντικά για την διαμόρφωση των υπόλοιπων μοντέλων, τις

λειτουργίες των ίδιων των χρηστών αλλά και στην αρχική διαδικασία Login.

Students:

```
[Table("students")]
[Index("RegistrationNum", Name = "UQ__students__DD1F272C84155ED5", IsUnique = true)]
public partial class Student
{
    [Key]
    [Column("username")]
    [StringLength(50)]
    [Unicode(false)]
    public string Username { get; set; } = null!;

    [Column("password")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Password { get; set; }

    [Column("name")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Name { get; set; }

    [Column("surname")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Surname { get; set; }

    [Column("registrationNum")]
    [StringLength(50)]
    [Unicode(false)]
    public string RegistrationNum { get; set; } = null!;

    [Column("department")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Department { get; set; }
    [InverseProperty("RegistrationNumNavigation")]
    public virtual ICollection<CourseHasStudent> CourseHasStudents { get; } = new List<CourseHasStudent>();
}
```

Το Data Annotation [Key] υποδηλώνει την χρήση της μεταβλητής ως πρωτεύον κλειδί του πίνακα-μοντέλου. Στην συγκεκριμένη περίπτωση το πρωτεύον κλειδί είναι η στήλη “Username”. Τα υπόλοιπα data annotations, υποδεικνύουν την μορφή των δεδομένων των μεταβλητών του μοντέλου μας.

Όλοι οι Public virtual getters/setters αντιπροσωπεύουν ένα πεδίο που χρησιμοποιείται σε ένα άλλο μοντέλο ως ξένο κλειδί π.χ. RegistrationNumNavigation στο CourseHasStudents.

Professors:

```
[Table("professors")]
[Index("Afm", Name = "UQ__professo__C6906E626E9F47D0", IsUnique = true)]
public partial class Professor
{
    [Key]
    [Column("username")]
    [StringLength(50)]
    [Unicode(false)]
    public string Username { get; set; } = null!;

    [Column("password")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Password { get; set; }

    [Column("AFM")]
    [StringLength(50)]
    [Unicode(false)]
    public string Afm { get; set; } = null!;

    [Column("name")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Name { get; set; }

    [Column("surname")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Surname { get; set; }

    [Column("department")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Department { get; set; }
    [InverseProperty("ProfessorsAfmNavigation")]
    public virtual ICollection<Course> Courses { get; } = new List<Course>();
}
```

Θα ήταν εφικτό, τα πεδία “Username” και “Password”, να ανήκαν σε ένα ξεχωριστό μοντέλο (και αντίστοιχα σε έναν ξεχωριστό πίνακα) π.χ. Users, για μια πιο εύκολη και με λιγότερο φόρτο διαδικασία Login. Όμως, έκανα την παραδοχή πως η έλλειψη ενός μοντέλου User θα ωφελήσει τις υπόλοιπες δραστηριότητες και θα μειώσει τον φόρτο εργασιών της πλατφόρμας με την συνεχή συγχώνευση πινάκων για την ανάκτηση των δεδομένων των Students και Professors, που γίνεται πολύ πιο συχνά.

Secretaries:

```
[Table("secretaries")]
[Index("PhoneNumber", Name = "UQ__secretar__4849DA01B682E493", IsUnique = true)]
public partial class Secretary
{
    [Key]
    [Column("username")]
    [StringLength(50)]
    [Unicode(false)]
    public string Username { get; set; } = null!;

    [Column("password")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Password { get; set; }

    [Column("phoneNumber")]
    [StringLength(50)]
    [Unicode(false)]
    public string? PhoneNumber { get; set; }

    [Column("name")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Name { get; set; }

    [Column("surname")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Surname { get; set; }

    [Column("department")]
    [StringLength(50)]
    [Unicode(false)]
    public string? Department { get; set; }
}
```

LOGIN

Αξιοποιώντας των HomeController που υπάρχει ήδη για τον έλεγχο της λειτουργικότητας της αρχικής σελίδας, η υλοποίηση της σελίδας Login είναι ακόμα πιο απλή. Η Login(), μπορεί με την χρήση ενός μοντέλου Student ως είσοδο, να ταυτοποιήσει οποιονδήποτε χρήστη της

εφαρμογής, Student, Professor ή Secretary. Το μοντέλο που χρησιμοποιείται ως είσοδος, δεν είναι αναγκαίο να είναι Student, αλλά οποιοδήποτε μοντέλο που περιέχει τα πεδία “Username” και “Password”, απλώς έκανα την παραδοχή να χρησιμοποιήσω το συγκεκριμένο. Το μοντέλο αυτό δεν ξαναχρησιμοποιείται ή αποθηκεύεται στην βάση έτσι και αλλιώς.

```
public async Task<IActionResult> Login(Student model)
{
    var student = from m in _context.Students select m;
    student = student.Where(s => s.Username == model.Username);
    var professor = from m in _context.Professors select m;
    professor = professor.Where(s => s.Username == model.Username);
    var secretary = from m in _context.Secretaries select m;
    secretary = secretary.Where(s => s.Username == model.Username);
    if (student.Count() != 0)
    {
        if (student.First().Password == model.Password)
        {
            return View("Index");
        }
        return View("Login");
    }
    else if (professor.Count() != 0)
    {
        if (professor.First().Password == model.Password)
        {
            return View("Index");
        }
        return View("Login");
    }
    else if(secretary.Count() != 0)
    {
        if (secretary.First().Password == model.Password)
        {
            return View("Index");
        }
        return View("Login");
    }
    else
    {
        return View("Login");
    }
}
```

LAYOUT/ΚΕΝΤΡΙΚΗ ΜΠΑΡΑ

Οι χρήστες μπορούν να έχουν πρόσβαση στις λειτουργίες της εφαρμογής, μέσω μιας μπάρας (με υπομενού για συγκεκριμένες λειτουργίες).

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" asp-area="" asp-controller="StudentsGrades" asp-action="Index">Students Menu</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" asp-area="" asp-controller="ProfessorsCourses" asp-action="Index">Professors Menu</a>
        </li>
        <li class="nav-item dropdown">
         <a class="nav-link dropdown-toggle" href="#" id="dropdown03" data-bs-toggle="dropdown" aria-expanded="false">Secretaries Menu</a>
            <ul class="dropdown-menu" aria-labelledby="dropdown03">
                <li><a class="dropdown-item" asp-area="" asp-controller="Students" asp-action="Index">Students</a></li>
                <li><a class="dropdown-item" asp-area="" asp-controller="Professors" asp-action="Index">Professors</a></li>
                <li><a class="dropdown-item" asp-area="" asp-controller="Courses" asp-action="Index">Courses</a></li>
                <li><a class="dropdown-item" asp-area="" asp-controller="CourseHasStudents" asp-action="Index">Assign Grades</a>
        </li>
            </ul>
        </li>
    </ul>
</div>
```

STUDENTS

(Pagination/Order By/Search)

Οι μαθητές έχουν την δυνατότητα να δουν τους βαθμούς τους ανά μάθημα ή ανά εξάμηνο με την χρήση των μεθόδων OrderBy και OrderByDescending, καθώς και να δουν την συνολική βαθμολογία των μαθημάτων τους για όλα τα μαθήματα που έχουν εξεταστεί, με την χρήση μιας μπάρας αναζήτησης, ψάχνοντας τον αριθμό μητρώου τους.

StudentsGradesView:

```
<form asp-action="Index" method="get">
    <div class="form-actions">
        <p>
            Find by Registration Number: <input type="text" name="search" value="@ViewData["CurrentFilter"]" />
            <input type="submit" value="search" class="btn btn-primary" /> |
            <a asp-action="Index">Back to Full List</a>
        </p>
    </div>
</form>
<table class="table">
    <thead>
        <tr>
            <th>
                <a asp-action="Index" asp-route-page="@ViewData["Page"]" asp-route-sortOrder="@ViewData["CourseTitleSortParm"]" asp-route-search="@ViewData["CurrentFilter"]">Course
            </th>
            <th>
                <a asp-action="Index" asp-route-page="@ViewData["Page"]" asp-route-sortOrder="@ViewData["CourseSemesterSortParm"]" asp-route-search="@ViewData["CurrentFilter"]">Cour
            </th>
            <th>
                <a asp-action="Index" asp-route-page="@ViewData["Page"]" asp-route-sortOrder="@ViewData["RegistrationNumSortParm"]" asp-route-search="@ViewData["CurrentFilter"]">Reg
            </th>
```

```
<nav>
    @Html.PagedListPager(Model, page => Url.Action("index", new { page = page }), new PagedListRenderOptions()
    {
    ActiveLiElementClass = "active",
    PageClasses = new[]{ "page-link"},
    LiElementClasses=new[] { "page-item" },
    UlElementClasses = new[] { "pagination","justify-content-center", "mt-3" },
    LinkToNextPageFormat = "Next",
    LinkToPreviousPageFormat = "Previous",
    MaximumPageNumbersToDisplay = 5,
    DisplayLinkToPreviousPage = PagedListDisplayMode.Always,
    DisplayLinkToNextPage = PagedListDisplayMode.Always,
    })
</nav>
```

StudentsGradesController:

```
// GET: StudentsGrades
public IActionResult Index(int? page, string? search, string sortOrder)
{
    ViewData["CurrentSortOrder"] = sortOrder;
    ViewData["CourseTitleSortParm"] = String.IsNullOrEmpty(sortOrder) ? "title_desc" : "";
    ViewData["CourseSemesterSortParm"] = sortOrder == "semester" ? "semester_desc" : "semester";
    ViewData["RegistrationNumParm"] = sortOrder == "regnum" ? "regnum_desc" : "regnum";

    ViewData["CurrentFilter"] = search;

    var grades = from e in _context.CourseHasStudents
                 select e;

    //Search
    if (!String.IsNullOrEmpty(search))
    {
        grades = grades.Where(e => e.RegistrationNum.Contains(search));
    }

    //SortOrder
    switch (sortOrder)
    {

        case "title_desc":
            grades = grades.OrderByDescending(e => e.IdCourseNavigation.CourseTitle);
            break;

        case "semester":
            grades = grades.OrderBy(e => e.IdCourseNavigation.CourseSemester);
            break;

        case "semester_desc":
            grades = grades.OrderByDescending(e => e.IdCourseNavigation.CourseSemester);
            break;

        case "regnum":
            grades = grades.OrderBy(e => e.RegistrationNum);
            break;
```

```
        case "regnum_desc":
            grades = grades.OrderByDescending(e => e.RegistrationNum);
            break;
        default:
            grades = grades.OrderBy(e => e.IdCourseNavigation.CourseTitle);
            break;
    }

    // Pagination
    if (page != null && page < 1)
    {
        page = 1;
    }

    int PageSize = 10;
    var gradesData = grades.Include(e => e.IdCourseNavigation).ToPagedList(page ?? 1, PageSize);

    return View(gradesData);
}
```

PROFESSORS

(Details + Added Information/Create)

Οι καθηγητές έχουν την δυνατότητα να δουν όλα τα μαθήματα και μετά να δουν αναλυτικά σε μορφή λίστας, την βαθμολογία των μαθητών για κάθε μάθημα. Έπειτα, μπορούν να προσθέσουν βαθμό σε όσους μαθητές δεν έχουν βαθμό σε κάποιο μάθημα.

ProfessorsGradesController

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("IdCourse,RegistrationNum,Grade")] CourseHasStudent courseHasStudent)
{
    if (ModelState.IsValid)
    {
        _context.Add(courseHasStudent);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index", "ProfessorsCourses");
    }
    ViewData["IdCourse"] = new SelectList(_context.Courses, "IdCourse", "IdCourse", courseHasStudent.IdCourse);
    ViewData["RegistrationNum"] = new SelectList(_context.Students, "RegistrationNum", "RegistrationNum", courseHasStudent.RegistrationNum);
    return View();
}
```

ProfessorsCourses/Details

```
<div>
    <h4>Course</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.CourseTitle)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.CourseTitle)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.CourseSemester)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.CourseSemester)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.ProfessorsAfmNavigation.Surname)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ProfessorsAfmNavigation.Surname)
        </dd>
    </dl>
</div>

<p>
    <a asp-action="Create" asp-route-id="@ViewData["c_id"]">Grade a student</a>
</p>

<table class="table">
    <thead>
        <tr>
            <th>
                Registration Number
            </th>
            <th>
                Grade
            </th>
                <th></th>
            </tr>
        </thead>
        <tbody>
    @foreach (var item in Model.CourseHasStudents) {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.RegistrationNum)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Grade)
                </td>
            </tr>
    }
        </tbody>
    </table>

    <div> |
        <a asp-action="Index">Back to List</a>
    </div>
```

SECRETARIES

Όλα τα μέλη της γραμματείας μπορούν να δουν και να καταχωρήσουν μαθήματα, νέους φοιτητές ή καθηγητές στην βάση, να αναθέσουν καθηγητές ως υπεύθυνους των μαθημάτων αλλά και να δηλώσουν μαθητές σε αυτά τα μαθήματα. Η επιλογή αυτών των διεργασιών γίνεται από μια μπάρα-υπόμενου της κύριας μπάρας εργασιών.(Οι λειτουργίες εκτελούνται παρόμοια με αυτές των άλλων ρόλων)

METADATA

Για την καλύτερη προβολή των δεδομένων, χρησιμοποιήθηκαν μοντέλα Metadata, σε λίγες περιπτώσεις.

Models/Metadata/ProfessorsCoursesMetadata

```
namespace GradingPlatformMVC.Models.Metadata
{
    1 reference | Πάτροκλος Γεωργίαδης, 17 hours ago | 1 author, 1 change
    public class ProfessorsCoursesMetadata
    {
        [Display(Name = "Course Title")]
        0 references | Πάτροκλος Γεωργίαδης, 17 hours ago | 1 author, 1 change
        public string? CourseTitle { get; set; }

        [Display(Name = "Course Semester")]
        0 references | Πάτροκλος Γεωργίαδης, 17 hours ago | 1 author, 1 change
        public int? CourseSemester { get; set; }

        [Display(Name = "Professor's Afm")]
        0 references | Πάτροκλος Γεωργίαδης, 17 hours ago | 1 author, 1 change
        public string? ProfessorsAfm { get; set; }
    }
}
```

PartialClasses

```
[ModelMetadataType(typeof(ProfessorsCoursesMetadata))]
36 references | Πάτροκλος Γεωργίαδης, 6 hours ago | 1 author, 6 changes
public partial class Course { }
```