

School-AUEB Project

Ο πλήρης κώδικας του Project μπορεί να βρεθεί στο GitHub στον παρακάτω σύνδεσμο:

<https://github.com/KStatharas/FinalProject-CSharp.git>

By Konstantinos Statharas

Εισαγωγή

Αυτό το Project αφορά τον σχεδιασμό και την υλοποίηση μιας εφαρμογής με σκοπό την πλοήγηση σε έναν πίνακα ελέγχου της βάσης δεδομένων ενός σχολείου.

Ο Backend κώδικας της εφαρμογής είναι γραμμένος στην γλώσσα C# αξιοποιώντας το σύστημα Entity Framework της Microsoft. Το Frontend μέρος της εφαρμογής χρησιμοποιεί HTML, CSS, Javascript με την βοήθεια Bootstrap και JQuery ενώ για την δημιουργία της βάσης χρησιμοποιήθηκε η ενσωματωμένη κλάση DbContext η οποία αναλαμβάνει και αυτοματοποιεί τα SQL Queries για τη λήψη ή αποστολή δεδομένων στον Server.

Ο σχεδιασμός της εφαρμογής υλοποιήθηκε με βάση το MVC μοντέλο το οποίο περιλαμβάνει Service και DAO layers, μοντέλα αντικειμένων και DTOS με όλες τις CRUD μεθόδους (Create, Read, Update, Delete).

Ανάλυση Προβλήματος

Το σχολείο ΑUEB έχει την ανάγκη να καταχωρεί τους μαθητές, τους καθηγητές και τα μαθήματα στα οποία συμμετέχουν σε μια βάση δεδομένων καθώς και να παρέχει τη δυνατότητα στους χρήστες να δημιουργούν λογαριασμούς, να δηλώνουν συμμετοχή σε νέα μαθήματα και να έχουν πρόσβαση σε λίστες μαθητών και καθηγητών.

Η εφαρμογή για να λύσει το παραπάνω πρόβλημα παρέχει την δυνατότητα δημιουργίας χρηστών με τρεις διαφορετικούς ρόλους:

- Διαχειριστής
- Καθηγητής
- Μαθητής

Οι διαχειριστές είναι χρήστες στους οποίους παρέχεται πλήρης δυνατότητα να βλέπουν, να τροποποιούν, να διαγράφουν και να προσθέτουν χρήστες και μαθήματα καθώς και την συμμετοχή τους σε αυτά.

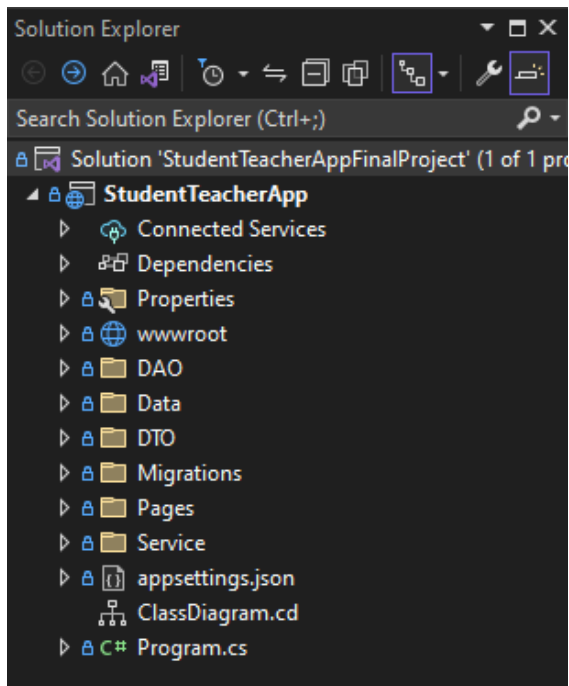
Οι καθηγητές έχουν την δυνατότητα να δημιουργούν μαθήματα και να βλέπουν λίστες άλλων καθηγητών και μαθητών.

Οι μαθητές έχουν την δυνατότητα να δηλώσουν την συμμετοχή τους σε μαθήματα και να βλέπουν λίστες άλλων καθηγητών και μαθητών.

Όλοι οι χρήστες έχουν την δυνατότητα να τροποποιήσουν τις προσωπικές τους πληροφορίες και να δουν το προφίλ τους.

Σχεδιασμός

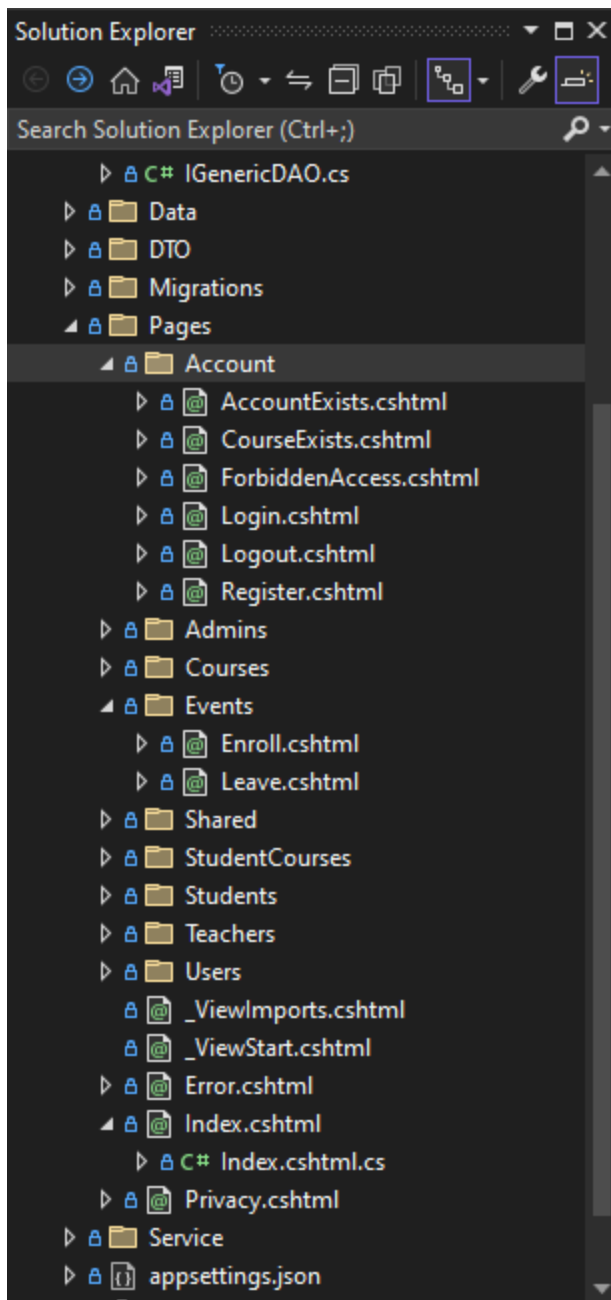
- Οργάνωση εφαρμογής



Η εφαρμογή περιέχει 6 βασικούς φακέλους:

1. **DAO:** Λαμβάνει εντολές από το Service και εισάγει δεδομένα στη βάση.
2. **Data:** Περιέχει τα μοντέλα (κύριες κλάσεις) και το DbContext.
3. **DTO:** Τα DTOs των μοντέλων μεταφέρουν τις πληροφορίες του χρήστη.
4. **Migrations:** Μεταφράζει το DbContext μοντέλο σε εντολές SQL.
5. **Pages:** Όλα τα Razor pages με τις CRUD μεθόδους.
6. **Service:** Το Service layer που με την χρήση APIs μεταφράζει τα DTOs σε μοντέλα και εν συνεχεία καλεί λειτουργίες του DAO Layer.

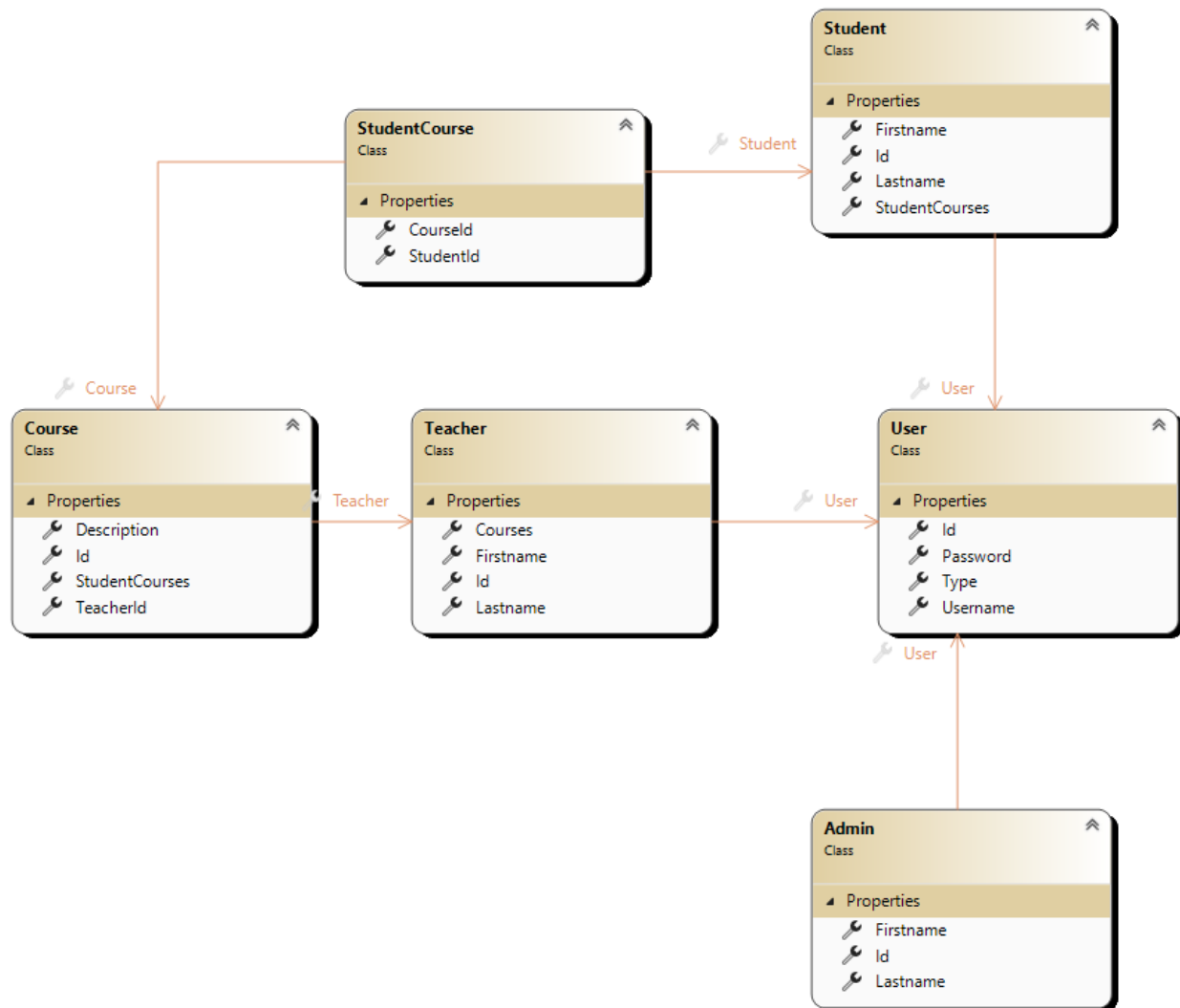
- Οργάνωση Razor Pages



Τα Razor pages πέρα από τις βασικές CRUD μεθόδους περιέχουν επίσης και επιπλέον λειτουργίες στον φάκελο Account που έχουν να κάνουν με την δημιουργία λογαριασμού και είσοδο του χρήστη στο σύστημα.

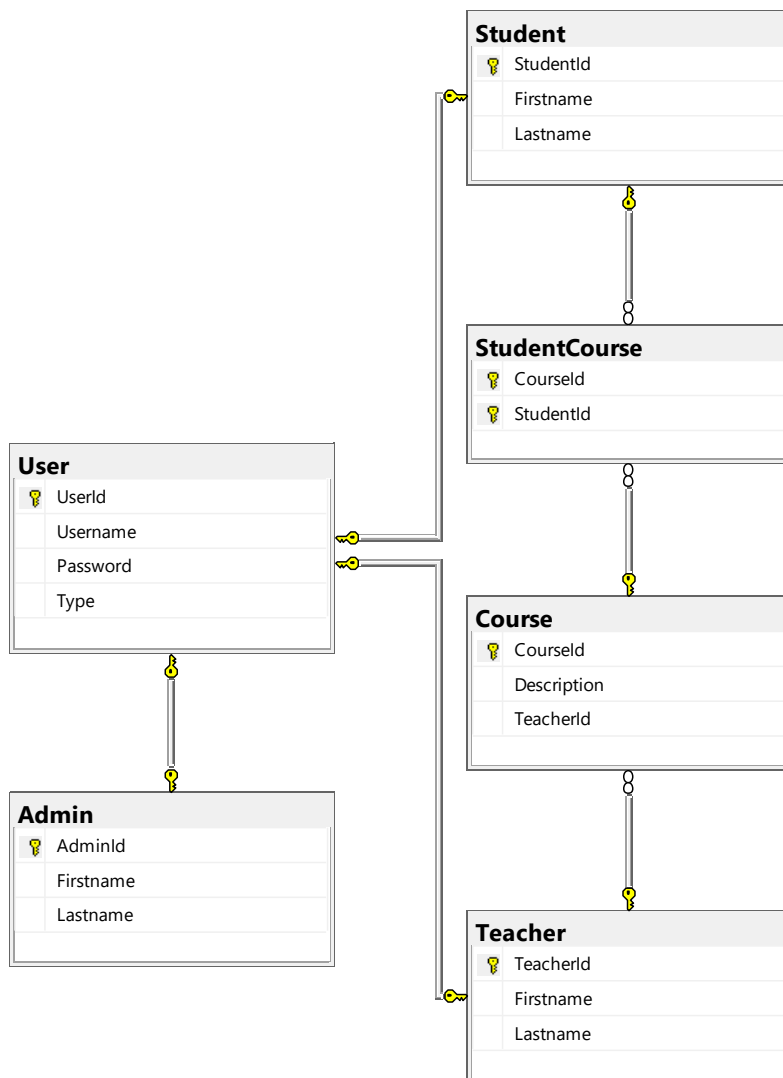
Επίσης φροντίζουν και για την διαδικασία της συμμετοχής ή αποχώρησης των μαθητών από τα μαθήματα (φάκελος Events).

- Διάταξη UML



Ο κώδικας της εφαρμογής βασίζεται στις κλάσεις που δείχνει το παραπάνω σχήμα UML. Πιο συγκεκριμένα οι κλάσεις Admin, Teacher και Student βασίζουν τα δεδομένα τους στα δεδομένα της κλάσης User. Η κλάση Teacher παρέχει δεδομένα στην κλάση Course η οποία λαμβάνει δεδομένα και από την κλάση StudentCourse που λειτουργεί ως συνδετικός κρίκος ανάμεσα στην Student και την Course.

- Βάση Δεδομένων



Το παραπάνω διάγραμμα της βάσης δεδομένων αποτυπώνει την σχέση μεταξύ των μοντέλων.

Πιο συγκεκριμένα οι πίνακες Admin, Teacher, Student έχουν σχέση **Ένα-Προς-Ένα** με τον User.

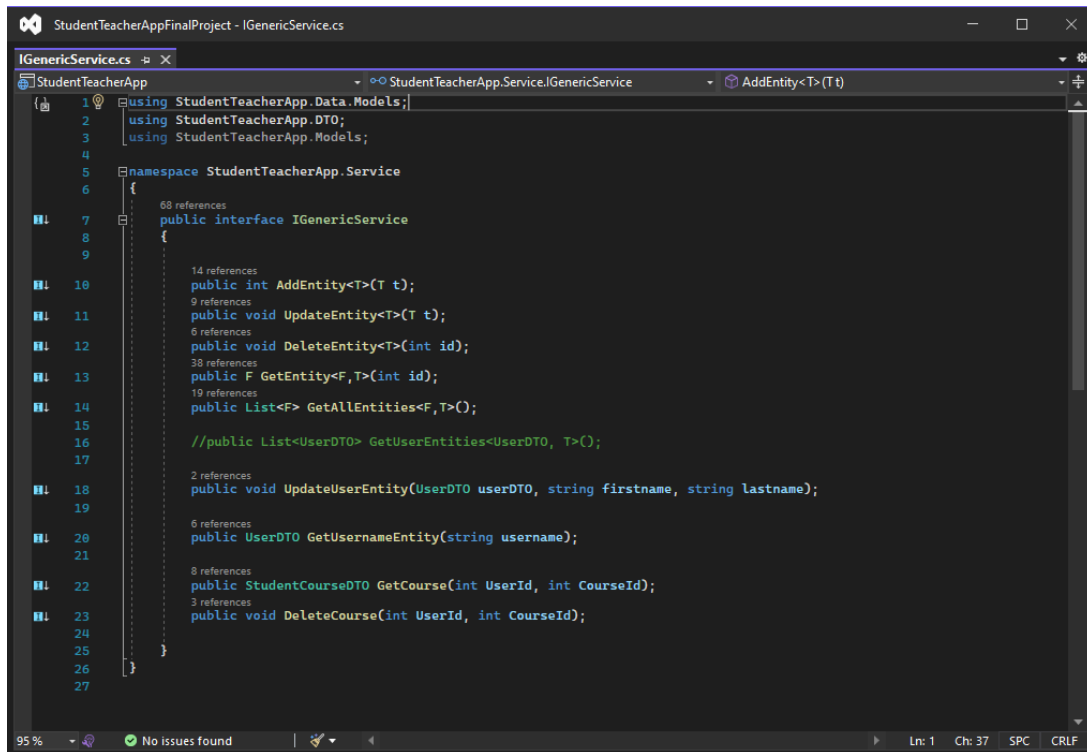
Επίσης ένας Καθηγητής μπορεί να διδάσκει σε πολλά μαθήματα όπως και ένας μαθητής μπορεί να συμμετέχει σε πολλά από αυτά και επομένως έχουν σχέσεις **Ένα-Προς-Πολλά**.

Υλοποίηση

Η περισσότερη λειτουργικότητα της εφαρμογής προέρχεται από το Service Layer που περιέχει όλα τα απαραίτητα APIs, από το DAO Layer που επικοινωνεί μέσω του DbContext με την βάση δεδομένων και εννοείται από τα Razor Pages που μέσω των Controllers πραγματοποιούν όλα τα APIs του Service και λαμβάνουν ή παρέχουν πληροφορίες στον χρήστη.

Πιο συγκεκριμένα:

• Service Layer



Βασικά CRUD APIs:

- **AddEntity:** Εισαγωγή αντικειμένου
- **UpdateEntity:** Ενημέρωση αντικειμένου
- **DeleteEntity:** Διαγραφή αντικειμένου
- **GetEntity:** Λήψη αντικειμένου
- **GetAllEntities:** Λήψη όλων των αντικειμένων

Extra CRUD APIs που δημιουργήθηκαν για λόγους επιπλέον ή ειδικής λειτουργικότητας:

- **UpdateUserEntity:** Ενημέρωση αντικειμένου χρήστη
- **GetUsernameEntity:** Λήψη αντικειμένου χρήστη με βάση δεδομένων από Cookie.
- **GetCourse:** Λήψη αντικειμένου μαθήματος με χρήση 2 παραμέτρων
- **DeleteCourse:** Διαγραφή αντικειμένου μαθήματος με χρήση 2 παραμέτρων

Στην συνέχεια τα παραπάνω APIs καλούν τις αντίστοιχες μεθόδους του DAO Layer:

```
1  using StudentTeacherApp.Data.Models;
2
3  namespace StudentTeacherApp.DAO
4  {
5      4 references
6      public interface IGenericDAO
7      {
8          7 references
9          public int Add<T>(T t);
10         8 references
11         public void Update<T>(T t);
12         5 references
13         public void Delete<T>(int id);
14         2 references
15         public T Get<T>(int id);
16         2 references
17         public List<T> GetAll<T>();
18
19         2 references
20         public User? GetUsername(string username);
21
22         2 references
23         public StudentCourse GetCourse(int UserId, int CourseId);
24         2 references
25         public void DeleteCourse(int UserId, int CourseId);
26     }
27 }
```

Μερικές υλοποιήσεις των APIs από το Service Layer:

```

17
18         this.genericDAO = genericDAO;
19     }
20
21     14 references
22     public int AddEntity<T>(T t)
23     {
24         int id = 0;
25
26         if (typeof(T) == typeof(TeacherDTO))
27         {
28             Teacher teacher = ConvertDTO<T, Teacher>(t);
29             id = genericDAO.Add(teacher);
30         }
31         else if (typeof(T) == typeof(StudentDTO))
32         {
33             Student student = ConvertDTO<T, Student>(t);
34             id = genericDAO.Add(student);
35         }
36         else if (typeof(T) == typeof(CourseDTO))
37         {
38             Course course = ConvertDTO<T, Course>(t);
39             id = genericDAO.Add(course);
40         }
41         else if (typeof(T) == typeof(StudentCourseDTO))
42         {
43             StudentCourse studentCourse = ConvertDTO<T, StudentCourse>(t);
44             id = genericDAO.Add(studentCourse);
45         }
46         else if (typeof(T) == typeof(AdminDTO))
47         {
48             Admin admin = ConvertDTO<T, Admin>(t);
49             id = genericDAO.Add(admin);
50         }
51         else if (typeof(T) == typeof(UserDTO))
52         {
53             User user = ConvertDTO<T, User>(t);
54             id = genericDAO.Add(user);
55         }
56
57         return id;
58     }
59
60     6 references
61     public void DeleteEntity<T>(int id) => genericDAO.Delete<T>(id);
62
63     19 references
64     public List<F> GetAllEntities<F, T>()
65     {
66         List<T> entities = genericDAO.GetAll<T>();
67         List<F> entityDTOS = new List<F>();
68
69         foreach (T entity in entities)
70         {
71             entityDTOS.Add(ConvertModel<T, F>(entity));
72         }
73
74         return entityDTOS;
75     }
76
77     38 references
78     public F GetEntity<F, T>(int id)
79     {
80         var daoResult = genericDAO.Get<T>(id);
81         if (daoResult is null) return default(F);
82
83         F entity = ConvertModel<T, F>(daoResult);
84         return entity;
85     }
86
87     9 references
88     public void UpdateEntity<T>(T t)
89     {
90         if (typeof(T) == typeof(TeacherDTO))
91         {
92             Teacher teacher = ConvertDTO<T, Teacher>(t);
93             genericDAO.Update(teacher);
94         }
95         else if (typeof(T) == typeof(StudentDTO))
96         {
97             Student student = ConvertDTO<T, Student>(t);
98             genericDAO.Update(student);
99         }
100         else if (typeof(T) == typeof(CourseDTO))
101         {
102             Course course = ConvertDTO<T, Course>(t);
103             genericDAO.Update(course);
104         }
105         else if (typeof(T) == typeof(StudentCourseDTO))
106         {
107             StudentCourse studentCourse = ConvertDTO<T, StudentCourse>(t);
108             genericDAO.Update(studentCourse);
109         }
110         else if (typeof(T) == typeof(AdminDTO))
111         {
112             Admin admin = ConvertDTO<T, Admin>(t);
113             genericDAO.Update(admin);
114         }
115         else if (typeof(T) == typeof(UserDTO))
116         {
117             User user = ConvertDTO<T, User>(t);
118             genericDAO.Update(user);
119         }
120     }
121
122     2 references
123     public void UpdateUserEntity(UserDTO userDTO, string firstname, string lastname)
124     {
125         User user = ConvertDTO<UserDTO, User>(userDTO);
126         genericDAO.Update(user);
127         switch (userDTO.Type)
128         {
129             case 1:
130                 user.Firstname = firstname;
131                 user.Lastname = lastname;
132                 genericDAO.Update(user);
133             case 2:
134                 user.Password = lastname;
135                 genericDAO.Update(user);
136             case 3:
137                 user.Password = lastname;
138                 genericDAO.Update(user);
139             case 4:
140                 user.Password = lastname;
141                 genericDAO.Update(user);
142             case 5:
143                 user.Password = lastname;
144                 genericDAO.Update(user);
145             case 6:
146                 user.Password = lastname;
147                 genericDAO.Update(user);
148             case 7:
149                 user.Password = lastname;
150                 genericDAO.Update(user);
151             case 8:
152                 user.Password = lastname;
153                 genericDAO.Update(user);
154             case 9:
155                 user.Password = lastname;
156                 genericDAO.Update(user);
157             case 10:
158                 user.Password = lastname;
159                 genericDAO.Update(user);
160             case 11:
161                 user.Password = lastname;
162                 genericDAO.Update(user);
163             case 12:
164                 user.Password = lastname;
165                 genericDAO.Update(user);
166             case 13:
167                 user.Password = lastname;
168                 genericDAO.Update(user);
169             case 14:
170                 user.Password = lastname;
171                 genericDAO.Update(user);
172             case 15:
173                 user.Password = lastname;
174                 genericDAO.Update(user);
175             case 16:
176                 user.Password = lastname;
177                 genericDAO.Update(user);
178             case 17:
179                 user.Password = lastname;
180                 genericDAO.Update(user);
181             case 18:
182                 user.Password = lastname;
183                 genericDAO.Update(user);
184             case 19:
185                 user.Password = lastname;
186                 genericDAO.Update(user);
187             case 20:
188                 user.Password = lastname;
189                 genericDAO.Update(user);
190             case 21:
191                 user.Password = lastname;
192                 genericDAO.Update(user);
193             case 22:
194                 user.Password = lastname;
195                 genericDAO.Update(user);
196             case 23:
197                 user.Password = lastname;
198                 genericDAO.Update(user);
199             case 24:
200                 user.Password = lastname;
201                 genericDAO.Update(user);
202             case 25:
203                 user.Password = lastname;
204                 genericDAO.Update(user);
205             case 26:
206                 user.Password = lastname;
207                 genericDAO.Update(user);
208             case 27:
209                 user.Password = lastname;
210                 genericDAO.Update(user);
211             case 28:
212                 user.Password = lastname;
213                 genericDAO.Update(user);
214             case 29:
215                 user.Password = lastname;
216                 genericDAO.Update(user);
217             case 30:
218                 user.Password = lastname;
219                 genericDAO.Update(user);
220             case 31:
221                 user.Password = lastname;
222                 genericDAO.Update(user);
223             case 32:
224                 user.Password = lastname;
225                 genericDAO.Update(user);
226             case 33:
227                 user.Password = lastname;
228                 genericDAO.Update(user);
229             case 34:
230                 user.Password = lastname;
231                 genericDAO.Update(user);
232             case 35:
233                 user.Password = lastname;
234                 genericDAO.Update(user);
235             case 36:
236                 user.Password = lastname;
237                 genericDAO.Update(user);
238             case 37:
239                 user.Password = lastname;
240                 genericDAO.Update(user);
241             case 38:
242                 user.Password = lastname;
243                 genericDAO.Update(user);
244             case 39:
245                 user.Password = lastname;
246                 genericDAO.Update(user);
247             case 40:
248                 user.Password = lastname;
249                 genericDAO.Update(user);
250             case 41:
251                 user.Password = lastname;
252                 genericDAO.Update(user);
253             case 42:
254                 user.Password = lastname;
255                 genericDAO.Update(user);
256             case 43:
257                 user.Password = lastname;
258                 genericDAO.Update(user);
259             case 44:
260                 user.Password = lastname;
261                 genericDAO.Update(user);
262             case 45:
263                 user.Password = lastname;
264                 genericDAO.Update(user);
265             case 46:
266                 user.Password = lastname;
267                 genericDAO.Update(user);
268             case 47:
269                 user.Password = lastname;
270                 genericDAO.Update(user);
271             case 48:
272                 user.Password = lastname;
273                 genericDAO.Update(user);
274             case 49:
275                 user.Password = lastname;
276                 genericDAO.Update(user);
277             case 50:
278                 user.Password = lastname;
279                 genericDAO.Update(user);
280             case 51:
281                 user.Password = lastname;
282                 genericDAO.Update(user);
283             case 52:
284                 user.Password = lastname;
285                 genericDAO.Update(user);
286             case 53:
287                 user.Password = lastname;
288                 genericDAO.Update(user);
289             case 54:
290                 user.Password = lastname;
291                 genericDAO.Update(user);
292             case 55:
293                 user.Password = lastname;
294                 genericDAO.Update(user);
295             case 56:
296                 user.Password = lastname;
297                 genericDAO.Update(user);
298             case 57:
299                 user.Password = lastname;
300                 genericDAO.Update(user);
301             case 58:
302                 user.Password = lastname;
303                 genericDAO.Update(user);
304             case 59:
305                 user.Password = lastname;
306                 genericDAO.Update(user);
307             case 60:
308                 user.Password = lastname;
309                 genericDAO.Update(user);
310             case 61:
311                 user.Password = lastname;
312                 genericDAO.Update(user);
313             case 62:
314                 user.Password = lastname;
315                 genericDAO.Update(user);
316             case 63:
317                 user.Password = lastname;
318                 genericDAO.Update(user);
319             case 64:
320                 user.Password = lastname;
321                 genericDAO.Update(user);
322             case 65:
323                 user.Password = lastname;
324                 genericDAO.Update(user);
325             case 66:
326                 user.Password = lastname;
327                 genericDAO.Update(user);
328             case 67:
329                 user.Password = lastname;
330                 genericDAO.Update(user);
331             case 68:
332                 user.Password = lastname;
333                 genericDAO.Update(user);
334             case 69:
335                 user.Password = lastname;
336                 genericDAO.Update(user);
337             case 70:
338                 user.Password = lastname;
339                 genericDAO.Update(user);
340             case 71:
341                 user.Password = lastname;
342                 genericDAO.Update(user);
343             case 72:
344                 user.Password = lastname;
345                 genericDAO.Update(user);
346             case 73:
347                 user.Password = lastname;
348                 genericDAO.Update(user);
349             case 74:
350                 user.Password = lastname;
351                 genericDAO.Update(user);
352             case 75:
353                 user.Password = lastname;
354                 genericDAO.Update(user);
355             case 76:
356                 user.Password = lastname;
357                 genericDAO.Update(user);
358             case 77:
359                 user.Password = lastname;
360                 genericDAO.Update(user);
361             case 78:
362                 user.Password = lastname;
363                 genericDAO.Update(user);
364             case 79:
365                 user.Password = lastname;
366                 genericDAO.Update(user);
367             case 80:
368                 user.Password = lastname;
369                 genericDAO.Update(user);
370             case 81:
371                 user.Password = lastname;
372                 genericDAO.Update(user);
373             case 82:
374                 user.Password = lastname;
375                 genericDAO.Update(user);
376             case 83:
377                 user.Password = lastname;
378                 genericDAO.Update(user);
379             case 84:
380                 user.Password = lastname;
381                 genericDAO.Update(user);
382             case 85:
383                 user.Password = lastname;
384                 genericDAO.Update(user);
385             case 86:
386                 user.Password = lastname;
387                 genericDAO.Update(user);
388             case 87:
389                 user.Password = lastname;
390                 genericDAO.Update(user);
391             case 88:
392                 user.Password = lastname;
393                 genericDAO.Update(user);
394             case 89:
395                 user.Password = lastname;
396                 genericDAO.Update(user);
397             case 90:
398                 user.Password = lastname;
399                 genericDAO.Update(user);
400             case 91:
401                 user.Password = lastname;
402                 genericDAO.Update(user);
403             case 92:
404                 user.Password = lastname;
405                 genericDAO.Update(user);
406             case 93:
407                 user.Password = lastname;
408                 genericDAO.Update(user);
409             case 94:
410                 user.Password = lastname;
411                 genericDAO.Update(user);
412             case 95:
413                 user.Password = lastname;
414                 genericDAO.Update(user);
415             case 96:
416                 user.Password = lastname;
417                 genericDAO.Update(user);
418             case 97:
419                 user.Password = lastname;
420                 genericDAO.Update(user);
421             case 98:
422                 user.Password = lastname;
423                 generic
```

Το Service Layer περιέχει επίσης μεθόδους που πραγματοποιούν την μετατροπή των DTOs σε Models αλλά και το αντίθετο όταν αυτό κριθεί απαραίτητο:

```
188 private U ConvertDTO<T, U>(T t)
189 {
190     U result = default;
191
192     if (typeof(T) == typeof(TeacherDTO))
193     {
194         TeacherDTO tDTO = t as TeacherDTO;
195         Teacher teacher = new()
196         {
197             Id = tDTO.Id,
198             Firstname = tDTO.Firstname,
199             Lastname = tDTO.Lastname,
200         };
201         result = (U)(object)teacher;
202     }
203     else if (typeof(T) == typeof(StudentDTO))
204     else if (typeof(T) == typeof(CourseDTO))
205     else if (typeof(T) == typeof(StudentCourseDTO))
206     else if (typeof(T) == typeof(AdminDTO))
207     else if (typeof(T) == typeof(UserDTO))
208
209     return result;
210 }
211
212 4 references
213 private T ConvertModel<U, T>(U u)
214 {
215     T result = default;
216
217     if (typeof(U) == typeof(Teacher))
218     {
219         Teacher teacher = u as Teacher;
220         TeacherDTO teacherDTO = new()
221         {
222             Id = teacher.Id,
223             Firstname = teacher.Firstname,
224             Lastname = teacher.Lastname,
225         };
226         result = (T)(object)teacherDTO;
227     }
228     else if (typeof(U) == typeof(Student))
229     {
230         Student student = u as Student;
231         StudentDTO studentDTO = new()
232         {
233             Id = student.Id,
234             Firstname = student.Firstname,
235             Lastname = student.Lastname,
236         };
237         result = (T)(object)studentDTO;
238     }
239     return result;
240 }
```

- DAO Layer

Μερικές μέθοδοι του DAO Layer:

```
166 ✓
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

8 references
public void Update<T>(T t)
{
    if (t == null) throw new ArgumentNullException(nameof(t));

    if (typeof(T) == typeof(Teacher))
    {
        Teacher teacher = (Teacher)(object)t;
        var local = _context.Set<Teacher>().Local.FirstOrDefault(entry => entry.Id.Equals(teacher.Id));
        if (local != null)
        {
            _context.Entry(teacher).State = Microsoft.EntityFrameworkCore.EntityState.Modified;
        }
        else if (typeof(T) == typeof(Student))
        {
        }
        else if (typeof(T) == typeof(Course))
        {
        }
        else if (typeof(T) == typeof(StudentCourse))
        {
        }
        else if (typeof(T) == typeof(Admin))
        {
        }
        else if (typeof(T) == typeof(User))
        {
        }
        _context.SaveChanges();
    }
}

2 references
public T Get<T>(int id)
{
    if (typeof(T) == typeof(Teacher))
    {
        Teacher? teacher = _context.Teacher.FirstOrDefault(x => x.Id == id);
        return (T)(object)teacher;
    }
    else if (typeof(T) == typeof(Student))
    {
        Student? student = _context.Student.FirstOrDefault(x => x.Id == id);
        return (T)(object)student;
    }
    else if (typeof(T) == typeof(Course))
    {
    }
    else if (typeof(T) == typeof(StudentCourse))
    {
    }
    else if (typeof(T) == typeof(Admin))
    {
    }
    else if (typeof(T) == typeof(User))
    {
    }
    return default(T);
}
```

```

18
19 // references
20 public int Add<T>(T t)
21 {
22     if (t == null) throw new ArgumentNullException(nameof(t));
23     int id = 0;
24
25     if (typeof(T) == typeof(Teacher))
26     {
27         var inserted = (Teacher)(object)t;
28         _context.Teacher.Add(inserted);
29         _context.SaveChanges();
30         id = inserted.Id;
31     }
32     else if (typeof(T) == typeof(Student))
33     {
34         var inserted = (Student)(object)t;
35         _context.Student.Add(inserted);
36         _context.SaveChanges();
37         id = inserted.Id;
38     }
39     else if (typeof(T) == typeof(Course))
40     {
41         var inserted = (Course)(object)t;
42         _context.Course.Add(inserted);
43         _context.SaveChanges();
44         id = inserted.Id;
45     }
46     else if (typeof(T) == typeof(StudentCourse))
47     {
48         var inserted = (StudentCourse)(object)t;
49         _context.StudentCourse.Add(inserted);
50         _context.SaveChanges();
51         id = inserted.Id;
52     }
53     else if (typeof(T) == typeof(Admin))
54     {
55         var inserted = (Admin)(object)t;
56         _context.Admin.Add(inserted);
57         _context.SaveChanges();
58         id = inserted.Id;
59     }
60     else if (typeof(T) == typeof(User))
61     {
62         var inserted = (User)(object)t;
63         _context.User.Add(inserted);
64         _context.SaveChanges();
65         id = inserted.Id;
66     }
67     return id;
68 }
69
70 // 5 references
71 public void Delete<T>(int id)
72 {
73     if (typeof(T) == typeof(Teacher))
74     {
75         Delete<Teacher>(id);
76     }
77     else if (typeof(T) == typeof(Student))
78     {
79         Delete<Student>(id);
80     }
81     else if (typeof(T) == typeof(Course))
82     {
83         Delete<Course>(id);
84     }
85     else if (typeof(T) == typeof(StudentCourse))
86     {
87         Delete<StudentCourse>(id);
88     }
89     else if (typeof(T) == typeof(Admin))
90     {
91         Delete<Admin>(id);
92     }
93     else if (typeof(T) == typeof(User))
94     {
95         Delete<User>(id);
96     }
97     else
98     {
99         User? user = _context.User.FirstOrDefault(x => x.Id == id);
100         switch (user.Type)
101         {
102             case "Admin":
103                 Delete<Admin>(id);
104                 break;
105             case "Teacher":
106                 Delete<Teacher>(id);
107                 break;
108             case "Student":
109                 Delete<Student>(id);
110                 break;
111             default:
112                 _context.User.Remove(user);
113                 break;
114         }
115     }
116 }

```

Μια περίπτωση χρήσης LINQ στο DAO Layer:

```

289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329

```

```

else if (typeof(T) == typeof(Course))
{
    return new List<T>((IEnumerable<T>)_context.Course.ToList());
}
else if (typeof(T) == typeof(StudentCourse))
{
    return new List<T>((IEnumerable<T>)_context.StudentCourse.ToList());
}
else if (typeof(T) == typeof(Admin))
{
    return new List<T>((IEnumerable<T>)_context.Admin.ToList());
}
else if (typeof(T) == typeof(User))
{
    return new List<T>((IEnumerable<T>)_context.User.ToList());
}
return null;
}

2 references
public StudentCourse? GetCourse(int UserId, int CourseId) => _context.StudentCourse.FirstOrDefault(x => x.CourseId == CourseId && x.StudentId == UserId);
2 references
public void DeleteCourse(int UserId, int CourseId)
{
    //StudentCourse studentCourse = _context.StudentCourse.FirstOrDefault(x => x.CourseId == CourseId && x.StudentId == UserId);
    //studentCourse.Student = null;
    //_context.StudentCourse.Remove(studentCourse);

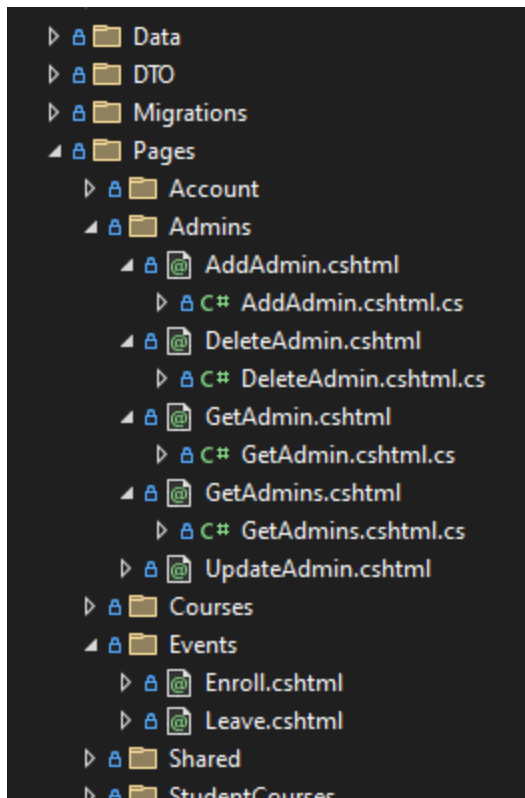
    _context.StudentCourse.Remove
    (
        from x in _context.StudentCourse
        where x.CourseId == CourseId && x.StudentId == UserId
        select x
    )?.FirstOrDefault(!);

    _context.SaveChanges();
}

```

- Razor Pages

Ο φάκελος Admins είναι ένα καλό παράδειγμα της υλοποίησης των CRUD λειτουργιών:



Παρακάτω είναι οι λειτουργίες που χειρίζονται οι Controllers του Admin Razor Page μαζί με τις cshtml σελίδες(frontend):

AddAdmin Razor page:

```
10 using StudentTeacherApp.Models;
11 using StudentTeacherApp.Service;
12 using Microsoft.AspNetCore.Authorization;
13 using System.Data;
14
15 namespace StudentTeacherApp.Pages.Admins
16 {
17     [Authorize(Roles = "Admin")]
18     public class AddAdminModel : PageModel
19     {
20
21         private readonly IGenericService _service;
22         public AddAdminModel(IGenericService service)
23         {
24             _service = service;
25         }
26
27         public IActionResult OnGet()
28         {
29             return Page();
30         }
31
32         [BindProperty]
33         public AdminDTO AdminDTO { get; set; }
34
35
36         public async Task<IActionResult> OnPostAsync()
37         {
38             if (!ModelState.IsValid)
39             {
40                 return Page();
41             }
42
43             _service.AddEntity(AdminDTO);
44
45             return RedirectToPage("/Index");
46         }
47     }
48 }
```

```
1 @page
2 @model StudentTeacherApp.Pages.Admins.AddAdminModel
3
4 @{
5     ViewData["Title"] = "Add Admin";
6 }
7
8 <h1>Add Admin</h1>
9
10 <h4>AdminDTO</h4>
11 <hr />
12 <div class="row">
13     <div class="col-md-4">
14         <form method="post">
15             <div asp-validation-summary="ModelOnly" class="text-danger"></div>
16             <div class="form-group">
17                 <label asp-for="AdminDTO.Firstname" class="control-label"></label>
18                 <input asp-for="AdminDTO.Firstname" class="form-control" />
19                 <span asp-validation-for="AdminDTO.Firstname" class="text-danger"></span>
20             </div>
21             <div class="form-group">
22                 <label asp-for="AdminDTO.Lastname" class="control-label"></label>
23                 <input asp-for="AdminDTO.Lastname" class="form-control" />
24                 <span asp-validation-for="AdminDTO.Lastname" class="text-danger"></span>
25             </div>
26             <div class="form-group">
27                 <input type="submit" value="Create" class="btn btn-primary" />
28             </div>
29         </form>
30     </div>
31 </div>
32
33 <div>
34     <a asp-page="Index">Back to List</a>
35 </div>
36
37 @section Scripts {
38     @await Html.RenderPartialAsync("_ValidationScriptsPartial");
39 }
40
```


DeleteAdmin Razor page:

```
19 public class DeleteAdminModel : PageModel
20 {
21     private readonly IGenericService _service;
22     public DeleteAdminModel(IGenericService service)
23     {
24         _service = service;
25     }
26
27     [BindProperty]
28     public AdminDTO AdminDTO { get; set; }
29
30     public async Task<IActionResult> OnGet(int id)
31     {
32         var admindto = _service.GetEntity<AdminDTO, Admin>(id);
33
34         if (admindto is default(AdminDTO))
35         {
36             return NotFound();
37         }
38         else
39         {
40             AdminDTO = admindto;
41         }
42         return Page();
43     }
44
45     public IActionResult OnPost(int id)
46     {
47         if (_service.GetEntity<AdminDTO, Admin>(id) is default(AdminDTO))
48         {
49             return NotFound();
50         }
51
52         _service.DeleteEntity<Admin>(id);
53         return RedirectToPage("/Index");
54     }
55 }
56
57 }
```

```
1 @page
2 @model StudentTeacherApp.Pages.Admins.DeleteAdminModel
3
4 @{
5     ViewData["Title"] = "Delete Admin";
6 }
7
8 <h1>Delete Admin</h1>
9
10 <h3>Are you sure you want to delete this Admin?</h3>
11 <div>
12     <hr />
13     <dl class="row">
14         <dt class="col-sm-2">
15             @Html.DisplayNameFor(model => model.AdminDTO.Id)
16         </dt>
17         <dd class="col-sm-10">
18             @Html.DisplayFor(model => model.AdminDTO.Id)
19         </dd>
20         <dt class="col-sm-2">
21             @Html.DisplayNameFor(model => model.AdminDTO.Firstname)
22         </dt>
23         <dd class="col-sm-10">
24             @Html.DisplayFor(model => model.AdminDTO.Firstname)
25         </dd>
26         <dt class="col-sm-2">
27             @Html.DisplayNameFor(model => model.AdminDTO.Lastname)
28         </dt>
29         <dd class="col-sm-10">
30             @Html.DisplayFor(model => model.AdminDTO.Lastname)
31         </dd>
32     </dl>
33
34     <form method="post">
35         <input type="hidden" asp-for="AdminDTO.Id" />
36         <input type="submit" value="Delete" class="btn btn-danger mt-5 mb-2"/>
37     </form>
38     <a asp-page="./GetAdmins">Back to List</a>
39 </div>
40
41 }
```

GetAdmin Razor page:

```
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6 using Microsoft.AspNetCore.Mvc.RazorPages;
7 using Microsoft.EntityFrameworkCore;
8 using StudentTeacherApp.DTO;
9 using StudentTeacherApp.Data;
10 using StudentTeacherApp.Models;
11 using StudentTeacherApp.Service;
12 using StudentTeacherApp.Data.Models;
13 using Microsoft.AspNetCore.Authorization;
14 using System.Data;
15
16 namespace StudentTeacherApp.Pages.Admins
17 {
18     [Authorize(Roles = "Admin")]
19     public class GetAdminModel : PageModel
20     {
21         private readonly IGenericService _service;
22
23         public GetAdminModel(IGenericService service)
24         {
25             _service = service;
26         }
27
28         // reference
29         public AdminDTO AdminDTO { get; set; }
30
31         // 0 reference
32         public IActionResult OnGet(int id)
33         {
34             if (_service.GetEntity<AdminDTO, Admin>(id) is default(AdminDTO))
35             {
36                 return NotFound();
37             }
38
39             AdminDTO = _service.GetEntity<AdminDTO, Admin>(id);
40
41             return Page();
42         }
43     }
44 }
```

```
1 @page "{id:int}"
2 @model StudentTeacherApp.Pages.Admins.GetAdminModel
3
4 @{
5     ViewData["Title"] = "Admin Details";
6 }
7
8 <h1>Admin Details</h1>
9
10 <div>
11     <hr />
12     <dl class="row">
13         @if (User.IsInRole("Admin"))
14         {
15             <dt class="col-sm-2">
16                 Admin Id
17             </dt>
18             <dd class="col-sm-10">
19                 @Html.DisplayFor(model => model.AdminDTO.Id)
20             </dd>
21         }
22         <dt class="col-sm-2">
23             @Html.DisplayNameFor(model => model.AdminDTO.Firstname)
24         </dt>
25         <dd class="col-sm-10">
26             @Html.DisplayFor(model => model.AdminDTO.Firstname)
27         </dd>
28         <dt class="col-sm-2">
29             @Html.DisplayNameFor(model => model.AdminDTO.Lastname)
30         </dt>
31         <dd class="col-sm-10">
32             @Html.DisplayFor(model => model.AdminDTO.Lastname)
33         </dd>
34     </dl>
35 </div>
36 <div>
37
38     <a asp-page="/UpdateCourse" asp-route-id="@Model.AdminDTO?.Id" class="btn btn-primary mt-5 mb-2">Edit</a>
39 }
40 <br />
41 <a asp-page="/GetCourses">Back to List</a>
42 </div>
43
```

GetAdmins Razor page:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Data;
4 using System.Linq;
5 using System.Threading.Tasks;
6 using Microsoft.AspNetCore.Authorization;
7 using Microsoft.AspNetCore.Mvc;
8 using Microsoft.AspNetCore.Mvc.RazorPages;
9 using Microsoft.EntityFrameworkCore;
10 using StudentTeacherApp.Data;
11 using StudentTeacherApp.Data.Models;
12 using StudentTeacherApp.DTO;
13 using StudentTeacherApp.Models;
14 using StudentTeacherApp.Service;
15
16 namespace StudentTeacherApp.Pages.Admins
17 {
18     [Authorize(Roles = "Admin")]
19     public class GetAdminsModel : PageModel
20     {
21         private readonly IGenericService _service;
22         public GetAdminsModel(IGenericService service)
23         {
24             _service = service;
25         }
26
27         4 references
28         public IList<AdminDTO> AdminDTO { get; set; } = default!;
29
30         0 references
31         public async Task OnGetAsync()
32         {
33             if (_service.GetAllEntities<AdminDTO, Admin>() != null)
34             {
35                 AdminDTO = _service.GetAllEntities<AdminDTO, Admin>(); ;
36             }
37         }
38     }
39 }
```

```
1 @page
2 @model StudentTeacherApp.Pages.Admins.GetAdminsModel
3
4 @{
5     ViewData["Title"] = "Admin Details";
6 }
7
8 <h1>Admins Details</h1>
9
10 @if (User.IsInRole("Admin"))
11 {
12     <p>
13         <a class="btn btn-success mt-1 mb-3 waves-effect" asp-page="/Account/Register">Create New</a>
14     </p>
15 }
16
17 <table class="table">
18     <thead>
19         <tr>
20             @if (User.IsInRole("Admin"))
21             {
22                 <th>
23                     @Html.DisplayNameFor(model => model.AdminDTO[0].Id)
24                 </th>
25                 <th>
26                     Full Name
27                 </th>
28                 <th>
29                     @Html.DisplayNameFor(model => model.AdminDTO[0].User)
30                 </th>
31                 <th></th>
32             }
33     </thead>
34     <tbody>
35         @foreach (var item in Model.AdminDTO)
36         {
37             <tr>
38                 @if (User.IsInRole("Admin"))
39                 {
40                     <td>
41                         @Html.DisplayFor(modelItem => item.Id)
42                     </td>
43                     <td>
44                         @Html.DisplayFor(modelItem => item.Firstname)
45                         @Html.DisplayFor(modelItem => item.Lastname)
46                     </td>
47                     <td>
48                         @Html.DisplayFor(modelItem => item.User.Id)
49                     </td>
50                     <td>
51                         @if (User.IsInRole("Admin"))
52                         {
53                             <a asp-page="/UpdateAdmin" asp-route-id="@item.Id" class="btn btn-info m-0 waves-effect">Edit</a>
54                             <a asp-page="/GetAdmin" asp-route-id="@item.Id" class="btn btn-info m-0 waves-effect">Details</a>
55                             <a asp-page="/DeleteAdmin" asp-route-id="@item.Id" class="btn btn-info m-0 waves-effect">Delete</a>
56                         }
57                     </td>
58                 }
59             </tr>
60         }
61     </tbody>
62 </table>
63 }
```

UpdateAdmin Razor page:

```
13 using StudentTeacherApp.Data.Models;
14 using Microsoft.AspNetCore.Authorization;
15 using System.Data;
16
17 namespace StudentTeacherApp.Pages.Admins
18 {
19     [Authorize(Roles = "Admin")]
20     public class UpdateAdminModel : PageModel
21     {
22         private readonly IGenericService _service;
23         public UpdateAdminModel(IGenericService service)
24         {
25             _service = service;
26         }
27
28         [BindProperty]
29         public AdminDTO AdminDTO { get; set; } = default!;
30
31         public async Task<IActionResult> OnGetAsync(int id)
32         {
33             if (_service.GetEntity<AdminDTO, Admin>(id) is default(AdminDTO))
34             {
35                 return NotFound();
36             }
37
38             AdminDTO = (AdminDTO)(object)_service.GetEntity<AdminDTO, Admin>(id);
39
40             return Page();
41         }
42
43         public async Task<IActionResult> OnPostAsync()
44         {
45             if (!ModelState.IsValid)
46             {
47                 return Page();
48             }
49
50             _service.UpdateEntity<AdminDTO>(AdminDTO);
51
52             return RedirectToPage("/Index");
53         }
54     }
55 }
```

```
1 @page "{id:int}"
2 @model StudentTeacherApp.Pages.Admins.UpdateAdminModel
3
4 @{
5     ViewData["Title"] = "Edit Admin";
6 }
7
8 <h1>Edit Admin</h1>
9
10 <hr />
11 <div class="row">
12     <div class="col-md-4">
13         <form method="post">
14             <div asp-validation-summary="ModelOnly" class="text-danger"></div>
15             <input type="hidden" asp-for="AdminDTO.Id" />
16             <div class="form-group">
17                 <label asp-for="AdminDTO.Firstname" class="control-label"></label>
18                 <input asp-for="AdminDTO.Firstname" class="form-control" />
19                 <span asp-validation-for="AdminDTO.Firstname" class="text-danger"></span>
20             </div>
21             <br />
22             <div class="form-group">
23                 <label asp-for="AdminDTO.Lastname" class="control-label"></label>
24                 <input asp-for="AdminDTO.Lastname" class="form-control" />
25                 <span asp-validation-for="AdminDTO.Lastname" class="text-danger"></span>
26             </div>
27             <br />
28             <div class="form-group">
29                 <input type="submit" value="Save" class="btn btn-primary mt-5 mb-2" />
30             </div>
31         </form>
32     </div>
33 </div>
34
35 <div>
36     <a asp-page="./GetAdmins">Back to List</a>
37 </div>
38
39 @section Scripts {
40     @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
41 }
42
```

Παρακάτω είναι μερικά επιπλέον Razor pages που χειρίζονται extra λειτουργίες:

Login Page και δημιουργία Cookie που συμβάλλει στο Authentication του χρήστη με τη χρήση Claims, Roles και Policies:

```
7 using System.Security.Claims;
8
9 namespace StudentTeacherApp.Pages.Account
10 {
11     8 references
12     public class LoginModel : PageModel
13     {
14         private readonly IGenericService _service;
15         0 references
16         public LoginModel(IGenericService service)
17         {
18             _service = service;
19         }
20
21         [BindProperty]
22         6 references
23         public Credential Credential { get; set; }
24         0 references
25         public void OnGet()
26         {
27         }
28
29         0 references
30         public async Task<IActionResult> OnPostAsync()
31         {
32             if (!ModelState.IsValid) return Page();
33
34             UserDTO UserDTO = _service.GetUsernameEntity(Credential.Username);
35             string id = Convert.ToString(UserDTO.Id);
36
37             if (UserDTO != default(UserDTO))
38             {
39                 var claims = new List<Claim>
40                 {
41                     new Claim(ClaimTypes.Name, Credential.Username),
42                     new Claim(ClaimTypes.Role, UserDTO.Type),
43                     new Claim("UserId", id)
44                 };
45                 var identity = new ClaimsIdentity(claims, "CredAuth");
46                 ClaimsPrincipal principal = new ClaimsPrincipal(identity);
47                 await HttpContext.SignInAsync("CredAuth", principal);
48                 return RedirectToPage("/Index");
49             }
50             return Page();
51         }
52     }
53     1 reference
54     public class Credential
55     {
56         [Required]
57         4 references
58         public string Username { get; set; }
59
60         [Required]
61         [DataType(DataType.Password)]
62         2 references
63         public string Password { get; set; }
64     }
65 }
```

Register Page:

```
47 public string Firstname { get; set; }
48
49 [BindProperty]
50 [Display(Name = "Last Name")]
51 [Required(ErrorMessage = "Last name is required.")]
52 [MinLength(3)]
53 [MaxLength(55)]
54
55 5 references
56 public string Lastname { get; set; }
57
58 0 references
59 public async Task<IActionResult> OnPostAsync()
60 {
61     if (!ModelState.IsValid) ...
62
63     if (_service.GetUsernameEntity(UserDTO.Username) != null) ...
64
65     string? type = UserDTO.Type;
66
67     switch (type)
68     {
69         case "Admin":
70             AdminDTO = new AdminDTO(); ...;
71             _service.AddEntity(AdminDTO);
72
73             break;
74         case "Student":
75             StudentDTO = new StudentDTO(); ...;
76             _service.AddEntity(StudentDTO);
77
78             break;
79         case "Teacher":
80             TeacherDTO = new TeacherDTO(); ...;
81             _service.AddEntity(TeacherDTO);
82
83             break;
84         default:
85             _service.AddEntity(UserDTO);
86             break;
87     }
88
89     return RedirectToPage("/Index");
90 }
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
```


Controller που δίνει την δυνατότητα να επιλέξει ο μαθητής την συμμετοχή του σε ένα μάθημα:

```
2 using Microsoft.AspNetCore.Mvc.RazorPages;
3 using StudentTeacherApp.Data.Models;
4 using StudentTeacherApp.DTO;
5 using StudentTeacherApp.Service;
6
7 namespace StudentTeacherApp.Pages.Events
8 {
9     6 references
10     public class EnrollModel : PageModel
11     {
12         private readonly IGenericService _service;
13         0 references
14         public EnrollModel(IGenericService service)
15         {
16             _service = service;
17         }
18         0 references
19         public UserDTO UserDTO { get; set; }
20         0 references
21         public async Task<IActionResult> OnGetAsync(int id)
22         {
23             string username = User.Identity.Name;
24             UserDTO UserDTO = _service.GetUsernameEntity(username);
25             StudentCourseDTO studentCourseDTO = new StudentCourseDTO()
26             {
27                 StudentId = UserDTO.Id,
28                 CourseId = id
29             };
30             _service.AddEntity(studentCourseDTO);
31             return RedirectToPage("/Courses/GetCourses");
32         }
33     }
34 }
35
36
37
38
39
```

Controller που δίνει την δυνατότητα να επιλέξει ο μαθητής την αποχώρησή του από ένα μάθημα:

```
1 using Microsoft.AspNetCore.Mvc;
2 using Microsoft.AspNetCore.Mvc.RazorPages;
3 using StudentTeacherApp.Data.Models;
4 using StudentTeacherApp.DTO;
5 using StudentTeacherApp.Service;
6
7 namespace StudentTeacherApp.Pages.Events
8 {
9     public class LeaveModel : PageModel
10     {
11         private readonly IGenericService _service;
12         public LeaveModel(IGenericService service)
13         {
14             _service = service;
15         }
16
17         public UserDTO UserDTO { get; set; }
18
19         public async Task<IActionResult> OnGetAsync(int id)
20         {
21             UserDTO = _service.GetUsernameEntity(User.Identity.Name);
22             _service.DeleteCourse(UserDTO.Id, id);
23
24             return RedirectToPage("/Courses/GetCourses");
25         }
26     }
27 }
28
29
```


Εδώ είναι το κυρίως πρόγραμμα που αναλαμβάνει την έναρξη της εφαρμογής. Σε αυτό το σημείο προστίθενται οι λειτουργίες των Authentications και των Policies:

```
2 using StudentTeacherApp.DAO;
3 using StudentTeacherApp.Data;
4 using StudentTeacherApp.Service;
5
6 var builder = WebApplication.CreateBuilder(args);
7
8 // Add services to the container.
9 builder.Services.AddRazorPages();
10
11 builder.Services.AddScoped<IGenericDAO, GenericDAOImpl>();
12 builder.Services.AddScoped<IGenericService, GenericServiceImpl>();
13
14 builder.Services.AddDbContext<ApplicationDbContext>(options =>
15 options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
16
17 builder.Services.AddAuthentication("CredAuth").AddCookie("CredAuth", options =>
18 {
19     options.Cookie.Name = "CredAuth";
20     options.LoginPath = "/Account/Login";
21     options.AccessDeniedPath = "/Account/ForbiddenAccess";
22 });
23
24 builder.Services.AddAuthorization(options =>
25 {
26     options.AddPolicy("UserSession", policy =>
27         policy.RequireAssertion(context =>
28             context.User.HasClaim(c => c.Value == "Admin") ||
29             context.User.HasClaim(c => c.Type == "UserId")));
30 });
31
32
33 var app = builder.Build();
34
35 // Configure the HTTP request pipeline.
36 if (!app.Environment.IsDevelopment())
37 {
38     app.UseExceptionHandler("/Error");
39     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
40     app.UseHsts();
41 }
42
43 app.UseHttpsRedirection();
44 app.UseStaticFiles();
45
46 app.UseRouting();
47
48 app.UseAuthentication();
49 app.UseAuthorization();
50
51 app.MapRazorPages();
52
53 app.Run();
```

Το DbContext δημιουργεί το σχήμα της Βάσης Δεδομένων:

```
7 namespace StudentTeacherApp.Data
8 {
9     7 references
10     public class ApplicationDbContext : DbContext
11     {
12         0 references
13         public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
14         {
15         }
16
17         0 references
18         protected override void OnModelCreating(ModelBuilder modelBuilder)
19         {
20             modelBuilder.Entity<Course>()
21                 .HasOne(p => p.Teacher)
22                 .WithMany(c => c.Courses)
23                 .HasForeignKey(fk => fk.TeacherId);
24
25             modelBuilder.Entity<StudentCourse>()
26                 .HasOne(p => p.Student)
27                 .WithMany(c => c.StudentCourses)
28                 .HasForeignKey(fk => fk.StudentId)
29                 .OnDelete(DeleteBehavior.Restrict);
30
31             modelBuilder.Entity<StudentCourse>()
32                 .HasOne(p => p.Course)
33                 .WithMany(c => c.StudentCourses)
34                 .HasForeignKey(fk => fk.CourseId)
35                 .OnDelete(DeleteBehavior.Restrict);
36
37             modelBuilder.Entity<StudentCourse>()
38                 .HasKey(c => new { c.CourseId, c.StudentId });
39
40             modelBuilder.Entity<User>()
41                 .Property(f => f.Id)
42                 .ValueGeneratedOnAdd();
43         }
44
45         5 references
46         public DbSet<Student> Student { get; set; }
47         5 references
48         public DbSet<Teacher> Teacher { get; set; }
49         7 references
50         public DbSet<Course> Course { get; set; }
51         14 references
52         public DbSet<StudentCourse> StudentCourse { get; set; }
53         5 references
54         public DbSet<Admin> Admin { get; set; }
55         12 references
56         public DbSet<User> User { get; set; }
57     }
58 }
```

Το μοντέλο του Student:

```
1 using StudentTeacherApp.Data.Models;
2 using System.ComponentModel.DataAnnotations;
3 using System.ComponentModel.DataAnnotations.Schema;
4
5 namespace StudentTeacherApp.Models
6 {
7     35 references
8     public class Student
9     {
10         [Key]
11         [Column("StudentId")]
12         [ForeignKey("User")]
13         [DatabaseGenerated(DatabaseGeneratedOption.None)]
14         6 references
15         public int Id { get; set; }
16
17         [Column("Firstname")]
18         2 references
19         public string? Firstname { get; set; }
20
21         [Column("Lastname")]
22         2 references
23         public string? Lastname { get; set; }
24
25         1 reference
26         public List<StudentCourse> StudentCourses { get; set; }
27
28         0 references
29         public User? User { get; set; }
30     }
31 }
```

Επίδειξη και Έλεγχος

Παρακάτω είναι μερικά στιγμιότυπα χρήσης της πλατφόρμας ως Administrator, Student και Teacher:

Login Screen:

School AUEB Home Privacy Courses LOGIN REGISTER

LOGIN REGISTER

Username
root

Password

SIGN IN

Not a member? Register

© 2022 - School AUEB - Privacy

Register Screen:

[School AUEB](#) [Home](#) [Privacy](#) [Courses](#) [LOGIN](#) [REGISTER](#)

LOGINREGISTER

First Name
Konstantinos

Last Name
Statharas

Username
kon

Password

Type
Student
Admin
Student
Teacher

SIGN IN

[Back to Login](#)

[School AUEB](#) [Home](#) [Privacy](#) [Courses](#) [LOGIN](#) [REGISTER](#)

LOGINREGISTER

First Name
Konstantinos

Last Name
Statharas

Username
kon

Password

Type
Student

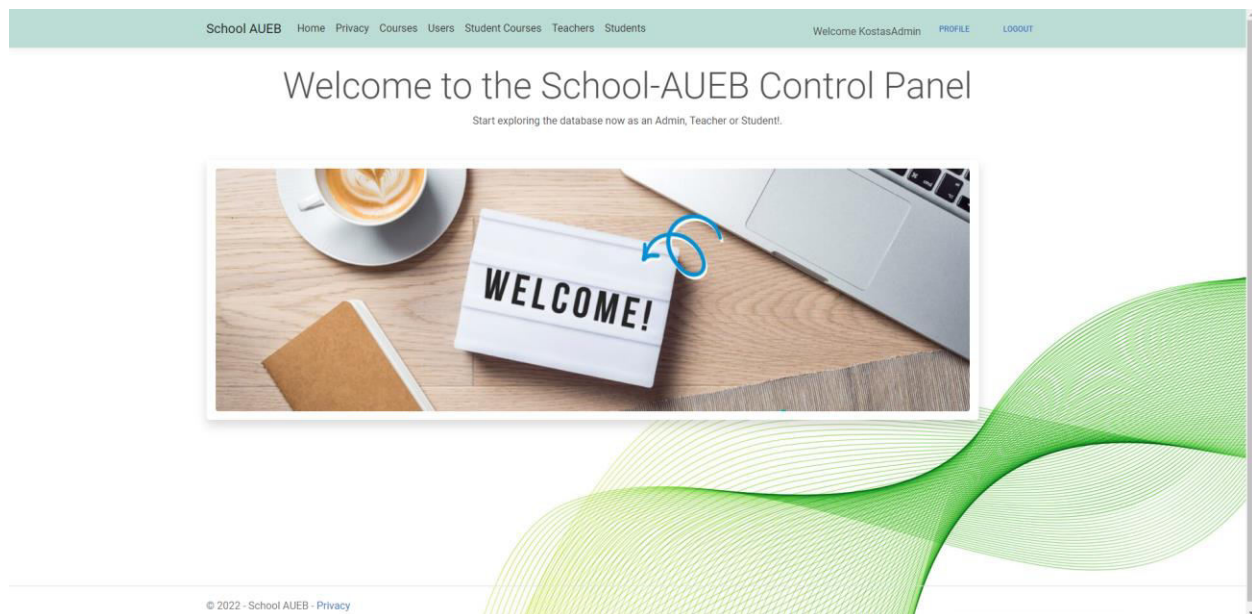
☒ I have read and agree to the [Privacy Policy](#)

SIGN IN

[Back to Login](#)

-Είσοδος ως Admin-

Όπως φαίνεται στο Navigation Bar ένας Admin έχει πρόσβαση σε όλες τις κατηγορίες:



Αλλά και σε όλες τις δυνατότητες (Create,Delete,Details,Edit):

School AUEB

Home

Privacy

Courses

Users

Student Courses

Teachers

Students

Welcome root

PROFILE

LOGOUT

Students

CREATE NEW

Id	Full Name	
10	Stondor Domin	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
16	Fabian Beristo	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
17	Birde Moggins	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
18	Enrico Cordon	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
19	Yordo Matlo	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
22	Selo Guccin	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>
23	Jonathan Jordan	<div>DETAILS</div> <div>EDIT</div> <div>DELETE</div>

https://localhost:7002/Students/CeStudents/10

School AUEB

Home

Privacy

Courses

Users

Student Courses

Teachers

Students

Welcome root

PROFILE

LOGOUT

Add Course

Description

Physics

TeacherId

13

CREATE

Back to List

© 2022 - School AUEB - Privacy

Όπως φαίνεται σε αυτήν την εικόνα ο Admin μπορεί να έχει πρόσβαση σε όλες τις πληροφορίες των χρηστών αλλά όχι στον κωδικό τους. Μόνο στον δικό του:

School AUEB Home Privacy Courses Users Student Courses Teachers Students Welcome root PROFILE LOGOUT

Users

CREATE NEW

Id	Username	Password	Full Name	Role	
9	root	3223	Mondes Chrysto	Admin	DETAILS EDIT DELETE
10	stud	*****	Stondor Donin	Student	DETAILS EDIT DELETE
11	teac	*****	David Sophor	Teacher	DETAILS EDIT DELETE
12	KostasAdmin	*****	Konstantinos Statharas	Admin	DETAILS EDIT DELETE
13	Zon	*****	Matias Denny	Teacher	DETAILS EDIT DELETE
14	Zonier	*****	Silvio Luxe	Teacher	DETAILS EDIT DELETE
15	Fadry12	*****	Kiken Lagor	Teacher	DETAILS EDIT DELETE
16	FadEd	*****	Fabian Beristo	Student	DETAILS EDIT DELETE
17	zero123	*****	Birgo Moggins	Student	DETAILS EDIT DELETE
18	Yode	*****	Enrico Cordon	Student	DETAILS EDIT DELETE

https://localhost:7002/Users/DeleteUser/13

School AUEB Home Privacy Courses Users Student Courses Teachers Students Welcome root PROFILE LOGOUT

Delete User

Are you sure you want to delete this User?

Id	13
Username	Zon
Password	*****
Role	Teacher

DELETE Back to List

© 2022 - School AUEB - Privacy

-Είσοδος ως Teacher-

Ένας Teacher δεν έχει πρόσβαση σε όλους τους χρήστες και μπορεί να εκτελέσει περιορισμένες ενέργειες:

The screenshot shows the 'Teachers' management page. At the top, there is a navigation bar with links: School AUEB, Home, Privacy, Courses, Student Courses, Teachers, and Students. On the right, it says 'Welcome teac' followed by 'PROFILE' and 'LOGOUT' links. The main heading is 'Teachers'. Below it, there is a table with the following data:

Full Name	
David Sopher	DETAILS
Matias Denny	DETAILS
Silvio Luxe	DETAILS
Kiken Lagor	DETAILS

At the bottom left, there is a copyright notice: '© 2022 - School AUEB - Privacy'. A URL bar at the bottom left shows 'https://localhost:7002/Teachers/GetTeacher/11'. A decorative green wavy graphic is on the right side.

Ωστόσο έχει πλήρεις δυνατότητες όσον αφορά τα μαθήματα:

The screenshot shows the 'Courses' management page. At the top, there is a navigation bar with links: School AUEB, Home, Privacy, Courses, Student Courses, Teachers, and Students. On the right, it says 'Welcome teac' followed by 'PROFILE' and 'LOGOUT' links. The main heading is 'Courses'. Below it, there is a green 'CREATE NEW' button. Below the button, there is a table with the following data:

Description	Theacher Id	Teacher Name	
Math	11	David Sopher	DETAILS EDIT DELETE
Programming	14	Silvio Luxe	DETAILS EDIT DELETE
Java	13	Matias Denny	DETAILS EDIT DELETE
Economics	15	Kiken Lagor	DETAILS EDIT DELETE

At the bottom left, there is a copyright notice: '© 2022 - School AUEB - Privacy'. A URL bar at the bottom left shows 'https://localhost:7002/Courses/GetCourse/11'. A decorative green wavy graphic is on the right side.

-Είσοδος ως Student-

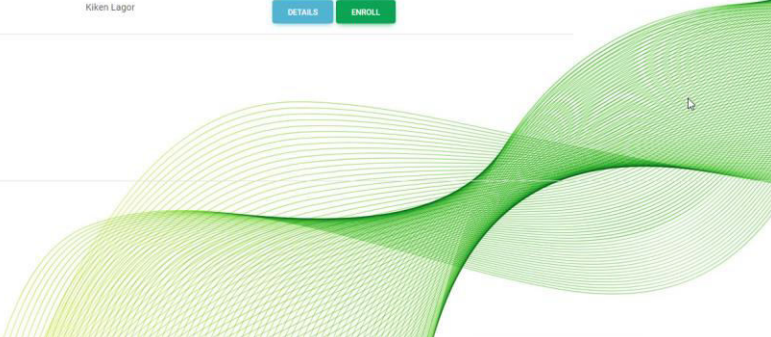
Σε έναν Student δεν του δίνεται η δυνατότητα να τροποποιήσει τις πληροφορίες των μαθημάτων όπως ένας Teacher, μπορεί όμως να επιλέξει σε ποια μαθήματα θέλει να πάρει μέρος και από ποια μαθήματα θέλει να αποχωρήσει :

School AUEBHomePrivacyCoursesTeachersStudentsWelcome studPROFILELOGOUT

Courses

Description	Theacher Id	Teacher Name		
Math	11	David Sopher	DETAILS	ENROLL
Programming	14	Silvio Luxe	DETAILS	LEAVE
Java	13	Matias Denny	DETAILS	ENROLL
Economics	15	Kiken Lagor	DETAILS	ENROLL

© 2022 - School AUEB - Privacy



Επίσης μπορεί όπως όλοι οι χρήστες να τροποποιήσει το προφίλ του:

School AUEBHomePrivacyCoursesTeachersStudentsWelcome studPROFILELOGOUT

Courses

Description	Theacher id	Teacher Name		
Math	11	David Sopher	DETAILS	ENROLL
Programming	14	Silvio Luxe	DETAILS	LEAVE
Java	13	Matias Denny	DETAILS	LEAVE
Economics	15	Kiken Lagor	DETAILS	ENROLL

© 2022 - School AUEB - Privacy

<https://localhost:7002/Users/GetUser/10>

School AUEBHomePrivacyCoursesTeachersStudentsWelcome studPROFILELOGOUT

User Profile

Username	stud
First Name	Stondor
Last Name	Donin
Password	22222323
Role	Student

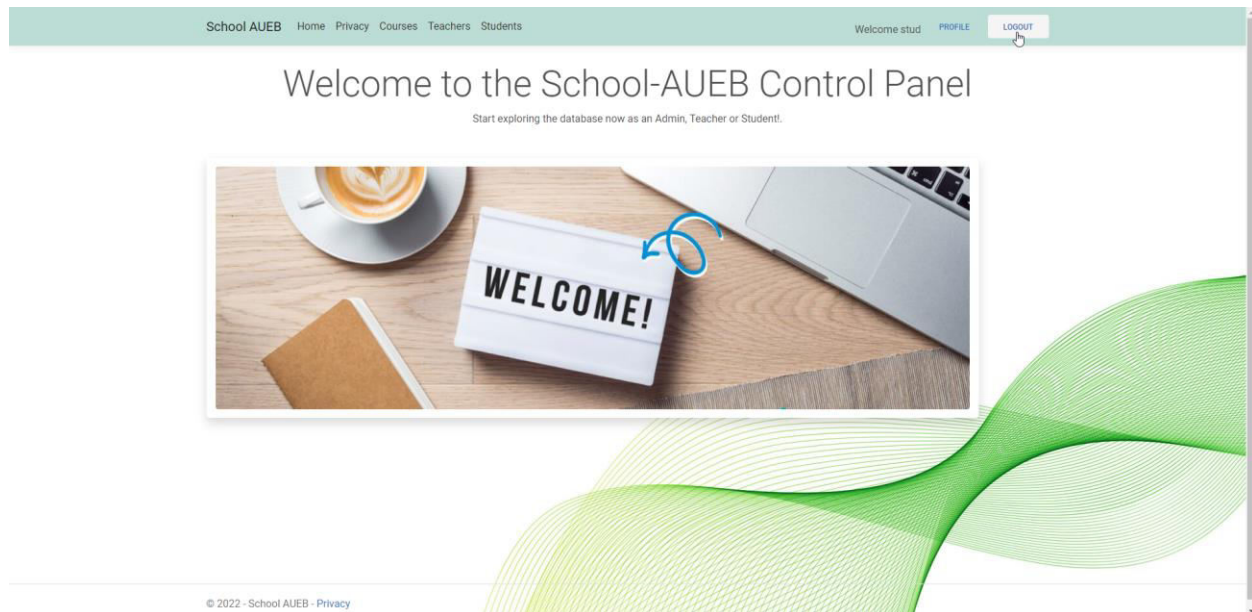
EDIT

Back to Home

© 2022 - School AUEB - Privacy

<https://localhost:7002/Users/UpdateUser/10>

Και τέλος οι χρήστες μπορούν αποσυνδεθούν από την πλατφόρμα:



Τέλος

Ο πλήρης κώδικας του Project μπορεί να βρεθεί στο GitHub στον παρακάτω σύνδεσμο:

<https://github.com/KStatharas/FinalProject-CSharp.git>

Βιβλιογραφία:

- Μαθήματα από το Ο.Π.Α. ως μέρος του ReGen Training Skills4Jobs που διοργανώθηκε από τον Σ.Ε.Β
- LinkedIn Course: [ASP-NET.NET Core: Razor Pages](#)

- Youtube: [ASP.NET Core Security Courses by Frank Liu](#)