

УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Рефакторинг баз данных и приложений»

Этап №2

Рефакторинг существующего проекта

Выполнил:

Волненко Д.А.

Преподаватель

Логинов И. П.

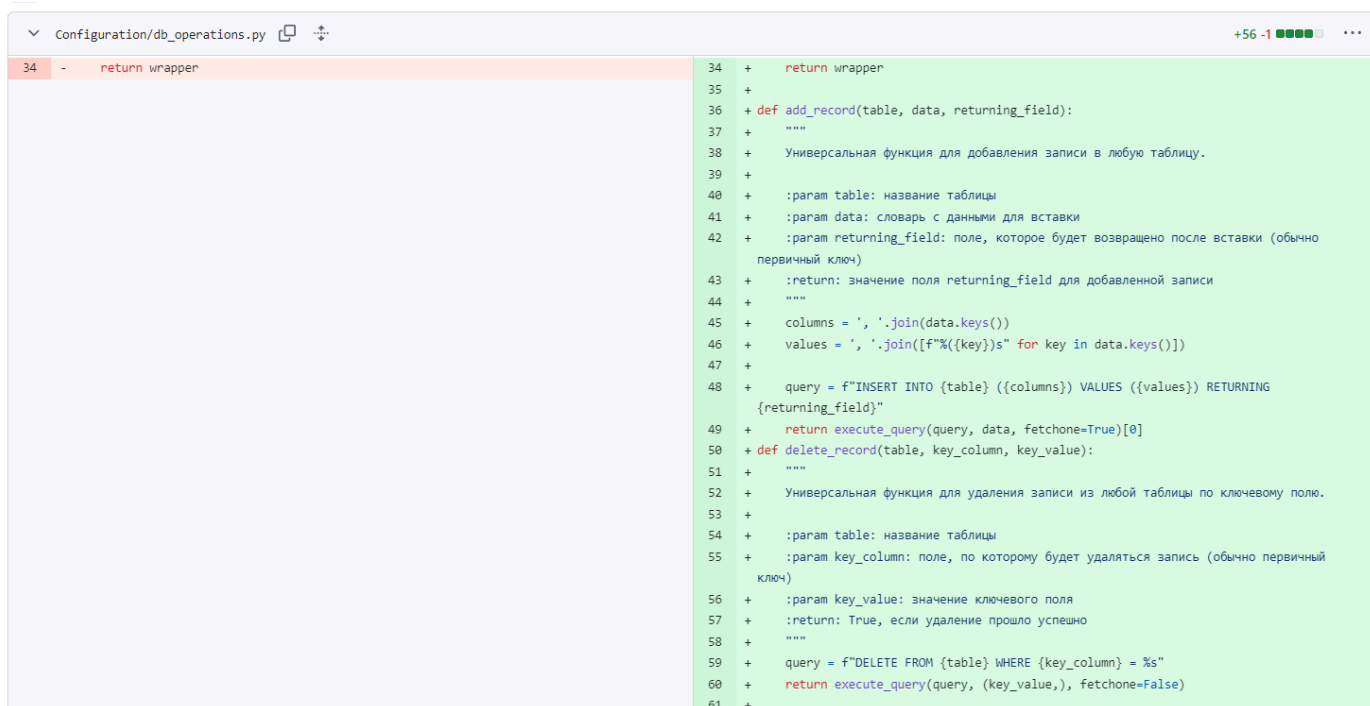
Санкт-Петербург, 2024 г.

Задание

Провести рефакторинг приложения, в качестве такого был взят курсовой проект по ИСБД – telegram-бот для управления библиотекой. Для второго этапа было запланировано сделать:

1. Унификация CRUD-операций: функции для работы с книгами, авторами, жанрами и т.д. имеют повторяющиеся операции по добавлению, удалению и обновлению записей в таблицах. Необходимо создать общие функции для этих операций.
2. Во многих обработчиках происходит валидация данных, введенных пользователем. Необходимо вынести проверки в отдельные функции.

Унификация CRUD-операций



```
34 - return wrapper

34 + return wrapper
35 +
36 + def add_record(table, data, returning_field):
37 +     """
38 +     Универсальная функция для добавления записи в любую таблицу.
39 +
40 +     :param table: название таблицы
41 +     :param data: словарь с данными для вставки
42 +     :param returning_field: поле, которое будет возвращено после вставки (обычно
        первичный ключ)
43 +     :return: значение поля returning_field для добавленной записи
44 +     """
45 +     columns = ', '.join(data.keys())
46 +     values = ', '.join([f"%({key})s" for key in data.keys()])
47 +
48 +     query = f"INSERT INTO {table} ({columns}) VALUES ({values}) RETURNING
        {returning_field}"
49 +     return execute_query(query, data, fetchone=True)[0]
50 + def delete_record(table, key_column, key_value):
51 +     """
52 +     Универсальная функция для удаления записи из любой таблицы по ключевому полю.
53 +
54 +     :param table: название таблицы
55 +     :param key_column: поле, по которому будет удаляться запись (обычно первичный
        ключ)
56 +     :param key_value: значение ключевого поля
57 +     :return: True, если удаление прошло успешно
58 +     """
59 +     query = f"DELETE FROM {table} WHERE {key_column} = %s"
60 +     return execute_query(query, (key_value,), fetchone=False)
61 +
```

```

62 + def update_record(table, update_data, key_column, key_value):
63 +     """
64 +     Универсальная функция для обновления записи в любой таблице.
65 +
66 +     :param table: название таблицы
67 +     :param update_data: словарь с обновляемыми данными
68 +     :param key_column: поле, по которому будет идентифицирована запись для
        обновления
69 +     :param key_value: значение ключевого поля
70 +     :return: True, если обновление прошло успешно
71 +     """
72 +     set_clause = ', '.join([f'{key} = %{key}s' for key in update_data.keys()])
73 +
74 +     query = f"UPDATE {table} SET {set_clause} WHERE {key_column} = %s"
75 +     update_data['key_value'] = key_value
76 +     return execute_query(query, update_data, fetchone=False)
77 +
78 + def get_record_by_id(table, key_column, key_value, fields="*"):
79 +     """
80 +     Получает запись из таблицы по идентификатору.
81 +
82 +     :param table: название таблицы
83 +     :param key_column: поле идентификатора (например, "book_id")
84 +     :param key_value: значение идентификатора
85 +     :param fields: поля, которые нужно выбрать (по умолчанию *)
86 +     :return: возвращает одну запись, если найдена
87 +     """
88 +     query = f"SELECT {fields} FROM {table} WHERE {key_column} = %s"
89 +     return execute_query(query, (key_value,), fetchone=True)

```

DatabaseInteractions/admin_database_updates.py

```

@@ -11,61 +11,51 @@ def add_book(name, ISBN, author_id, publisher_id, genre_id, department_id, copie
11     'department_id': department_id,
12     'copies': copies,
13 }
14 - query = '''
15 - INSERT INTO Books (name, ISBN, author_id, publisher_id, genre_id,
16 - department_id, copies)
17 - VALUES %(name)s, %(ISBN)s, %(author_id)s, %(publisher_id)s, %(genre_id)s,
18 - %(department_id)s, %(copies)s)
19 - RETURNING book_id;
20 - '''
21 - return execute_query(query, book_data, fetchone=True)[0]
22
23 def delete_book(book_id):
24 - query = "DELETE FROM Books WHERE book_id = %s"
25 - return execute_query(query, (book_id,), fetchone=False)
26
27 def change_copies(book_id, new_copies):
28 - query = "UPDATE Books SET copies = %s WHERE book_id = %s"
29 - return execute_query(query, (new_copies, book_id), fetchone=False)
30
31 def add_author(name):
32     if check_if_exists("SELECT * FROM Authors WHERE name = %s", (name,)):
33         return "Автор с таким именем уже существует!"
34 - query = '''INSERT INTO Authors(name) VALUES(%s) RETURNING author_id'''
35 - return execute_query(query, (name,), fetchone=True)[0]
36
37 def delete_author(author_id):
38 - query = "DELETE FROM Authors WHERE author_id = %s"
39 - return execute_query(query, (author_id,), fetchone=False)
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

| | |
|--|--|
| <pre> 39 def add_publisher(name): 40 if check_if_exists("SELECT * FROM Publishers WHERE name = %s", (name,)): 41 return "Издатель с таким именем уже существует!" 42 - query = '''INSERT INTO Publishers(name) VALUES(%s) RETURNING publisher_id''' 43 - return execute_query(query, (name,), fetchone=True)[0] 44 45 def delete_publisher(publisher_id): 46 - query = "DELETE FROM Publishers WHERE publisher_id = %s" 47 - return execute_query(query, (publisher_id,), fetchone=False) 48 49 def add_department(name): 50 if check_if_exists("SELECT * FROM LibraryDepartments WHERE name = %s", 51 (name,)): 52 return "Отдел с таким именем уже существует!" 52 - query = '''INSERT INTO LibraryDepartments(name) VALUES(%s) RETURNING 53 department_id''' 53 - return execute_query(query, (name,), fetchone=True)[0] 54 55 def delete_department(department_id): 56 - query = "DELETE FROM LibraryDepartments WHERE department_id = %s" 57 - return execute_query(query, (department_id,), fetchone=False) 58 59 def add_genre(genre_name): 60 - query = '''SELECT genre_id FROM Genres WHERE name = %s''' 61 - existing_genre = execute_query(query, (genre_name,), fetchone=True) 62 - 63 - if existing_genre: 64 - return existing_genre[0] 65 66 - query = "INSERT INTO Genres (name) VALUES (%s) RETURNING genre_id" 67 - return execute_query(query, (genre_name,), fetchone=True)[0] 68 </pre> | <pre> 33 def add_publisher(name): 34 if check_if_exists("SELECT * FROM Publishers WHERE name = %s", (name,)): 35 return "Издатель с таким именем уже существует!" 36 + 37 + publisher_data = {'name': name} 38 + return add_record("Publishers", publisher_data, "publisher_id") 39 40 def delete_publisher(publisher_id): 41 + return delete_record("Publishers", "publisher_id", publisher_id) 42 43 def add_department(name): 44 if check_if_exists("SELECT * FROM LibraryDepartments WHERE name = %s", 45 (name,)): 46 return "Отдел с таким именем уже существует!" 46 + 47 + department_data = {'name': name} 48 + return add_record("LibraryDepartments", department_data, "department_id") 49 50 def delete_department(department_id): 51 + return delete_record("LibraryDepartments", "department_id", department_id) 52 53 def add_genre(genre_name): 54 + if check_if_exists("SELECT genre_id FROM Genres WHERE name = %s", 55 (genre_name,)): 56 + return "Жанр с таким именем уже существует!" 56 57 + genre_data = {'name': genre_name} 58 + return add_record("Genres", genre_data, "genre_id") 59 </pre> |
|--|--|

| | |
|--|--|
| <pre> 68 69 def delete_genre(genre_id): 70 - query = "DELETE FROM Genres WHERE genre_id = %s" 71 - return execute_query(query, (genre_id,), fetchone=False) </pre> | <pre> 59 60 def delete_genre(genre_id): 61 + return delete_record("Genres", "genre_id", genre_id) </pre> |
|--|--|

| | | |
|---|---|-------|
| DatabaseInteractions/admin_user_interaction.py | | +2 -8 |
| <pre> 1 from Configuration.db import create_connection 2 3 4 def delete_user_reservation(reservation_id): 5 - conn = create_connection() 6 - with conn: 7 - with conn.cursor() as cursor: 8 - query = ''' 9 - DELETE FROM BookReservations 10 - WHERE reservation_id = %s 11 - ... 12 - cursor.execute(query, (reservation_id,)) </pre> | <pre> 1 from Configuration.db import create_connection 2 + from Configuration.db_operations import delete_record 3 4 5 def delete_user_reservation(reservation_id): 6 + return delete_record("BookReservations", "reservation_id", reservation_id) </pre> | |

| | | |
|---|---|---------|
| DatabaseInteractions/book_catalog.py | | +12 -11 |
| <pre> 1 - from Configuration.db_operations import execute_query 2 import datetime 3 4 def book_available(book_id): 5 """ 6 Проверяет наличие доступных копий книги. 7 8 @@ -17,8 +23,7 @@ def search_books(query): 17 rows = execute_query(book_query, ('%' + query + '%',), fetchall=True) 18 19 for row in rows: 20 - author_query = "SELECT name FROM Authors WHERE author_id = %s" 21 - author = execute_query(author_query, (row[2],), fetchone=True)[0] 22 books.append({'book_id': row[0], 'name': row[1], 'author': author}) 23 24 return books 27 @@ -27,15 +32,11 @@ def get_book_details(book_id): 27 """ 28 Получает подробную информацию о книге, включая автора и жанр. 29 """ 30 - book_query = "SELECT name, ISBN, author_id, genre_id, copies FROM Books WHERE 31 - book_id = %s" 32 - row = execute_query(book_query, (book_id,), fetchone=True) 32 </pre> | <pre> 1 + from Configuration.db_operations import * 2 import datetime 3 4 + def delete_user_reservation(reservation_id): 5 + """ 6 + Удаляет бронирование книги по идентификатору бронирования. 7 + """ 8 + return delete_record("BookReservations", "reservation_id", reservation_id) 9 + 10 def book_available(book_id): 11 """ 12 Проверяет наличие доступных копий книги. 23 rows = execute_query(book_query, ('%' + query + '%',), fetchall=True) 24 25 for row in rows: 26 + author = get_record_by_id("Authors", "author_id", row[2], "name") 27 28 books.append({'book_id': row[0], 'name': row[1], 'author': author}) 29 30 return books 32 """ 33 Получает подробную информацию о книге, включая автора и жанр. 34 """ 35 + row = get_record_by_id("Books", "book_id", book_id, "name, ISBN, author_id, 36 genre_id, copies") 36 </pre> | |

| | | | |
|-----|---|-----|---|
| 32 | | 36 | |
| 33 | if row: | 37 | if row: |
| 34 | - author_query = "SELECT name FROM Authors WHERE author_id = %s" | 38 | + author = get_record_by_id("Authors", "author_id", row[2], "name") |
| 35 | - author = execute_query(author_query, (row[2],), fetchone=True)[0] | 39 | + genre = get_record_by_id("Genres", "genre_id", row[3], "name") |
| 36 | - | | |
| 37 | - genre_query = "SELECT name FROM Genres WHERE genre_id = %s" | | |
| 38 | - genre = execute_query(genre_query, (row[3],), fetchone=True)[0] | | |
| 39 | | 40 | |
| 40 | - return {'name': row[0], 'ISBN': row[1], 'author': author, 'genre': genre, | 41 | - return {'name': row[0], 'ISBN': row[1], 'author': author, 'genre': genre, |
| 41 | 'copies': row[4]} | 42 | 'copies': row[4]} |
| 42 | | | |
| 117 | rows = execute_query(query, (reader_id,), fetchall=True) | 118 | rows = execute_query(query, (reader_id,), fetchall=True) |
| 118 | reservations = [{'reservation_id': row[0], 'book_name': row[1], | 119 | reservations = [{'reservation_id': row[0], 'book_name': row[1], |
| 119 | 'reservation_date': row[2]} for row in rows] | 120 | 'reservation_date': row[2]} for row in rows] |
| 120 | - return reservations | 121 | + return reservations |

| | | | |
|----|---|----|---|
| 10 | return True если пользователь уже зарегистрирован: | 10 | return True если пользователь уже зарегистрирован: |
| 11 | | 11 | |
| 12 | # Регистрация пользователя | 12 | # Регистрация пользователя |
| 13 | - query = "INSERT INTO Readers(name, contact_data, reader_number) | 13 | + user_data = { |
| 14 | - VALUES(%s, %s, %s) RETURNING reader_id" | 14 | + 'name': name, |
| 15 | - return execute_query(query, (name, contact_data, reader_number), fetchone=True) | 15 | + 'contact_data': contact_data, |
| | [0] | 16 | + 'reader_number': reader_number |
| | | 17 | + } |
| | | 18 | + return add_record("Readers", user_data, "reader_id") |
| 16 | | 19 | |
| 17 | def log_in_user(name, reader_number): | 20 | def log_in_user(name, reader_number): |
| 18 | """ | 21 | """ |
| 26 | """ | 29 | """ |
| 27 | Проверяет существование сотрудника в базе данных по имени. | 30 | Проверяет существование сотрудника в базе данных по имени. |
| 28 | """ | 31 | """ |
| 29 | - query = "SELECT staff_id FROM LibraryStaff WHERE name = %s" | 32 | + return get_record_by_id("LibraryStaff", "name", name, "staff_id") |
| 30 | - result = execute_query(query, (name,), fetchone=True) | | |
| 31 | - return result[0] if result else None | | |
| 32 | | 33 | |
| 33 | def get_user_profile(user_id): | 34 | def get_user_profile(user_id): |
| 34 | """ | 35 | """ |
| 35 | Возвращает профиль пользователя по его ID. | 36 | Возвращает профиль пользователя по его ID. |
| 36 | """ | 37 | """ |
| 37 | - query = "SELECT name, contact_data, reader_number FROM Readers WHERE | 38 | + return get_record_by_id("Readers", "reader_id", user_id, "name, contact_data, |
| 38 | reader_id = %s" | | reader_number") |
| 39 | - return execute_query(query, (user_id,), fetchone=True) | | |
| 40 | | 39 | |
| 40 | def user_exists(user_id): | 40 | def user_exists(user_id): |
| 41 | """ | 41 | """ |
| 42 | Проверяет, существует ли пользователь с данным ID. | 42 | Проверяет, существует ли пользователь с данным ID. |
| 43 | """ | 43 | """ |
| 44 | - return check_if_exists("SELECT 1 FROM Readers WHERE reader_id = %s", | 44 | + return check_if_exists("Readers", "reader_id", user_id) |
| | {user_id}) | | |

Рефакторинг CRUD-операций упростил и унифицировал взаимодействие с базой данных, заменив повторяющиеся запросы общими функциями. Это не только сократило дублирование кода, но и улучшило читаемость и поддержку кода, сделав его более гибким и модульным.

Рефакторинг валидации

▼ Handlers/bot_handlers.py

↕

↕

+47 -15

■■■■■

...

```
90 @router.message(Login.waiting_for_reader_number)
91 async def login_reader_number_entered(message: types.Message, state: FSMContext):
92     await state.update_data(reader_number=message.text)
93     data = await state.get_data()
94     reader_id = log_in_user(data['name'], data['reader_number'])

95

96     if reader_id:
97         await state.update_data(logged_in=True, reader_id=reader_id)
98         await ask_question(message, state, "Вход выполнен. Добро пожаловать, " +
99             data['name'] + "!",
100             UserLoggedIn.active)
101     else:
102         await message.answer("Неправильное имя пользователя или пароль")
103         await state.set_state(None)
104
105
106 @@ -174,23 +179,20 @@ async def cmd_reserve_book(message: types.Message, state: FSMContext):
107
108     @router.message(ReserveBook.waiting_for_book_id)
109     async def book_id_to_reserve_entered(message: types.Message, state: FSMContext):
110         user_data = await state.get_data()
111         reader_id = user_data.get("reader_id") # Используйте reader_id вместо user_id
112         try:
113             book_id = int(message.text)
114         except ValueError:
115             await message.answer("Пожалуйста, введите корректный ID книги (число).")
116             return
117         result = reserve_book(book_id, reader_id)
118
119
120 @router.message(ReserveBook.waiting_for_book_id)
121 async def book_id_to_reserve_entered(message: types.Message, state: FSMContext):
122     user_data = await state.get_data()
123     reader_id = user_data.get("reader_id")
124     book_id_error = validate_book_id(message.text)
125     if error:
126         await message.answer(error)
127     return
```

```
Handlers/library_handlers.py
63
64 # region Add Book
65 @router.message(Command(commands=["addbook"]))
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
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
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1
```

| | | | |
|-----|---|-----|---|
| 95 | | 112 | |
| 96 | await state.update_data(author_id=author_id) | 113 | await state.update_data(author_id=author_id) |
| 97 | await ask_question(message, state, "Введите ID издателя книги:", AddBook.waiting_for_publisher_id) | 114 | await ask_question(message, state, "Введите ID издателя книги:", AddBook.waiting_for_publisher_id) |
| 98 | | 115 | |
| 99 | - | | |
| 100 | @router.message(AddBook.waiting_for_publisher_id) | 116 | @router.message(AddBook.waiting_for_publisher_id) |
| 101 | @handle_db_errors | 117 | @handle_db_errors |
| 102 | async def publisher_id_entered(message: types.Message, state: FSMContext): | 118 | async def publisher_id_entered(message: types.Message, state: FSMContext): |
| 103 | - try: | 119 | + publisher_id, error = validate_int_input(message.text, "ID издателя") |
| 104 | - publisher_id = int(message.text) | 120 | + if error: |
| 105 | - except ValueError: | 121 | + await message.answer(error) |
| 106 | - await message.answer("Пожалуйста, введите корректный ID издателя (число).") | | |
| 107 | - return | 122 | return |
| 108 | | 123 | |
| 109 | await state.update_data(publisher_id=publisher_id) | 124 | await state.update_data(publisher_id=publisher_id) |
| 110 | await ask_question(message, state, "Введите ID жанра книги:", AddBook.waiting_for_genre_id) | 125 | await ask_question(message, state, "Введите ID жанра книги:", AddBook.waiting_for_genre_id) |
| 111 | | 126 | |
| 112 | - | | |
| 113 | @router.message(AddBook.waiting_for_genre_id) | 127 | @router.message(AddBook.waiting_for_genre_id) |
| 114 | @handle_db_errors | 128 | @handle_db_errors |
| 115 | async def genre_id_entered(message: types.Message, state: FSMContext): | 129 | async def genre_id_entered(message: types.Message, state: FSMContext): |
| 116 | - try: | 130 | + genre_id, error = validate_int_input(message.text, "ID жанра") |
| 117 | - genre_id = int(message.text) | 131 | + if error: |
| 118 | - except ValueError: | 132 | + await message.answer(error) |
| 119 | - await message.answer("Пожалуйста, введите корректный ID жанра (число).") | | |
| 120 | - return | 133 | return |
| 121 | | 134 | |
| 122 | await state.update_data(genre_id=genre_id) | 135 | await state.update_data(genre_id=genre_id) |
| 123 | await ask_question(message, state, "Введите ID отдела библиотеки для книги:", AddBook.waiting_for_department_id) | 136 | await ask_question(message, state, "Введите ID отдела библиотеки для книги:", AddBook.waiting_for_department_id) |
| 124 | | 137 | |
| 125 | - | | |

| | | | |
|------------------------------|--|---------|--|
| Handlers/library_handlers.py | | +33 -26 | |
| 125 | - | 138 | @router.message(AddBook.waiting_for_department_id) |
| 126 | @router.message(AddBook.waiting_for_department_id) | 139 | @handle_db_errors |
| 127 | @handle_db_errors | 140 | async def department_id_entered(message: types.Message, state: FSMContext): |
| 128 | async def department_id_entered(message: types.Message, state: FSMContext): | 141 | + department_id, error = validate_int_input(message.text, "ID отдела") |
| 129 | - try: | 142 | + if error: |
| 130 | - department_id = int(message.text) | 143 | + await message.answer(error) |
| 131 | - except ValueError: | | |
| 132 | - await message.answer("Пожалуйста, введите корректный ID отдела (число).") | 144 | return |
| 133 | - return | 145 | |
| 134 | | 146 | await state.update_data(department_id=department_id) |
| 135 | await state.update_data(department_id=department_id) | 147 | await ask_question(message, state, "Введите количество копий книги:", AddBook.waiting_for_copies) |
| 136 | await ask_question(message, state, "Введите количество копий книги:", AddBook.waiting_for_copies) | 148 | |
| 137 | | | |
| 138 | - | 149 | @router.message(AddBook.waiting_for_copies) |
| 139 | @router.message(AddBook.waiting_for_copies) | 150 | @handle_db_errors |
| 140 | @handle_db_errors | 151 | async def copies_entered(message: types.Message, state: FSMContext): |
| 141 | async def copies_entered(message: types.Message, state: FSMContext): | 152 | + copies, error = validate_int_input(message.text, "количество копий") |
| 142 | - try: | 153 | + if error: |
| 143 | - copies = int(message.text) | 154 | + await message.answer(error) |
| 144 | - except ValueError: | | |
| 145 | - await message.answer("Пожалуйста, введите корректное количество копий (число).") | 155 | return |
| 146 | - return | 156 | |
| 147 | | 157 | data = await state.get_data() |
| 148 | data = await state.get_data() | | |
| 149 | | 165 | await state.set_state(None) |
| 150 | | 166 | await state.set_state(None) |
| 151 | | 167 | |
| 152 | | | |
| 153 | | 168 | # endregion |
| 154 | | 169 | |
| 155 | | 170 | # region update amount |
| 156 | | | |
| 157 | | | |
| 158 | | | |
| 159 | - | | |
| 160 | - | | |
| 161 | # endregion | | |
| 162 | | | |
| 163 | # region update amount | | |

+

9