

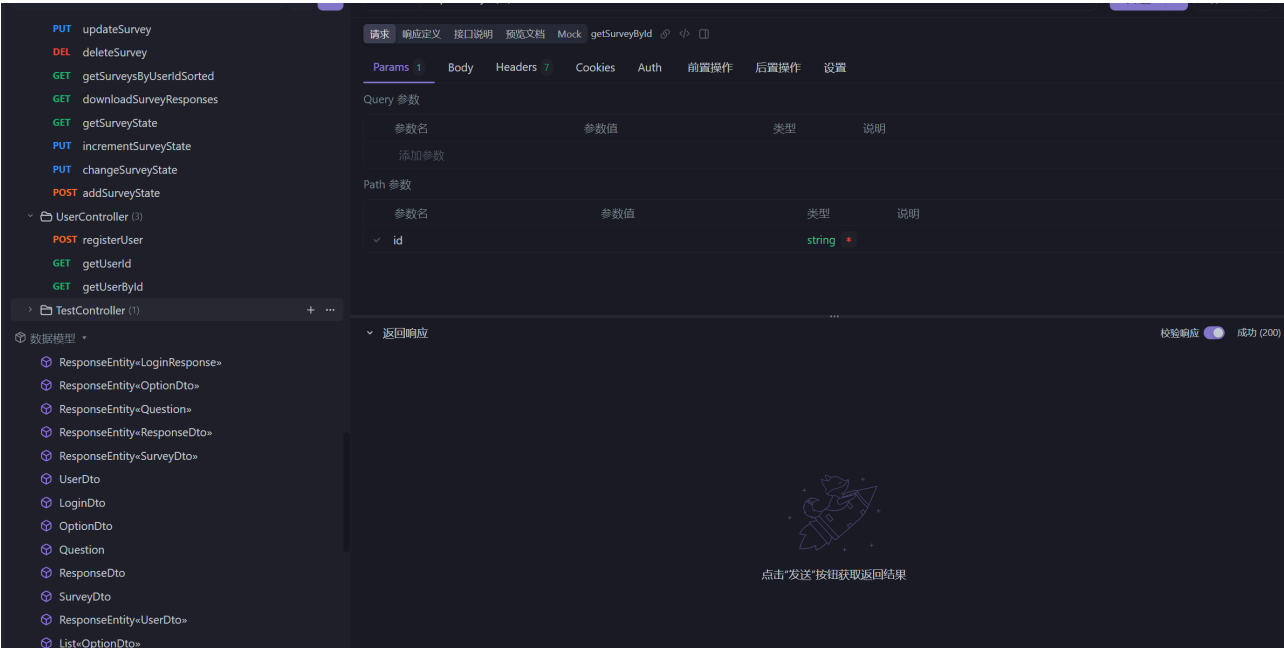
成果概要

本阶段进行了大量的功能开发，进行集中的功能实现，做为前后端的统筹，除了功能的实现规划，我实际工作中主要在于spring boot框架的后端数据库操作的api实现，同时和前端进行对接。

技术成功展示

成功沿用SSM框架这个标准MVC模式，通过在controller层提供简单直观的api和前端进行交互对接，同时在APIFOX软件进行api的实时同步共享以及api功能测试。

API共享



Controller层的代码实现

OptionController层

```
15 public class OptionController extends SecurityConfig{
22
23     //给某个问题添加选项
24     @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
25     @PostMapping("/{questionId}")
26     public ResponseEntity<OptionDto> addOptionToQuestion(@PathVariable Long questionId, @RequestBody OptionDto
27         OptionDto createdOption = optionServiceImpl.addOptionToQuestion(questionId, optionDto);
28     return ResponseEntity.ok(createdOption);
29 }
30
31     //根据问题的id获取选项
32     @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
33     @GetMapping("/{questionId}")
34     public ResponseEntity<List<OptionDto>> getOptionsForQuestion(@PathVariable Long questionId) {
35         List<OptionDto> options = optionServiceImpl.getOptionsForQuestion(questionId);
36         return ResponseEntity.ok(options);
37     }
38
39     //根据选项id更新选项
40     @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
41     @PutMapping("/{id}")
42     public ResponseEntity<OptionDto> updateOption(@PathVariable Long id, @RequestBody OptionDto optionDto) {
43         OptionDto updatedOption = optionServiceImpl.updateOption(id, optionDto);
44         return ResponseEntity.ok(updatedOption);
45     }
46
47     //根据选项id删除选项
48     @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
49     @DeleteMapping("/{id}")
50     public ResponseEntity<?> deleteOption(@PathVariable Long id) {
51         optionServiceImpl.deleteOption(id);
52         return ResponseEntity.ok().build();
53     }
54
55     //根据问题id获取选项
56     @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
57     @GetMapping("/question/{questionId}")
58     public ResponseEntity<List<OptionDto>> getOptionsByQuestionId(@PathVariable Long questionId) {
59         List<OptionDto> options = optionServiceImpl.getOptionsByQuestionId(questionId);
60         return ResponseEntity.ok(options);
61     }
62 }
```

QuestionController

```
17 public class QuestionController extends SecurityConfig {
46     return new ResponseEntity<>(questions, HttpStatus.OK);
47 }
48
49 //根据问题id更新问题
50 @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
51 @PutMapping("/{questionId}")
52 public ResponseEntity<Question> updateQuestion(@PathVariable Long questionId, @RequestBody Question question) {
53     question.setId(questionId);
54     questionServiceImpl.updateQuestion(question);
55     return new ResponseEntity<>(question, HttpStatus.OK);
56 }
57
58 //根据问题id删除问题
59 @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
60 @DeleteMapping("/{questionId}")
61 public ResponseEntity<Void> deleteQuestion(@PathVariable Long questionId) {
62     questionServiceImpl.deleteQuestion(questionId);
63     return new ResponseEntity<>(HttpStatus.OK);
64 }
65
66
67
68 //根据问题id和选项id更新选项
69 @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
70 @PutMapping("/{questionId}/options/{optionId}")
71 public ResponseEntity<OptionDto> updateOption(@PathVariable Long questionId, @PathVariable Long optionId, @RequestBody OptionDto option) {
72     option.setQuestionId(questionId);
73     option.setId(optionId);
74     OptionDto updatedOption = questionServiceImpl.updateOption(option);
75     return new ResponseEntity<>(updatedOption, HttpStatus.OK);
76 }
77
78 //交换两个问题的顺序
79 @CrossOrigin(origins = "http://localhost:8081")  xy-Summer-Sky
80 @PutMapping("/{questionId1}/swap/{questionId2}")
81 public ResponseEntity<Void> swapQuestions(@PathVariable Long questionId1, @PathVariable Long questionId2) {
82     questionServiceImpl.swapQuestions(questionId1, questionId2);
83     return new ResponseEntity<>(HttpStatus.OK);
84 }
85 }
```

ResponseController层

```
11 //ResponseController类是一个控制器类，用于处理回复相关的HTTP请求。
12 @RestController
13 @RequestMapping("/api/responses")
14 public class ResponseController extends SecurityConfig {
15
16     private final ResponseServiceImpl responseServiceImpl; 4个用法
17
18     public ResponseController(ResponseServiceImpl responseServiceImpl) {
19         this.responseServiceImpl = responseServiceImpl;
20     }
21
22     //给某个问题添加回复
23     @CrossOrigin
24     @PostMapping("/{questionId}")
25     public ResponseEntity<ResponseDto> addResponseToQuestion(@PathVariable Long questionId, @RequestBody ResponseDto responseDto) {
26         ResponseDto createdResponse = responseServiceImpl.addResponseToQuestion(questionId, responseDto);
27         return ResponseEntity.ok(createdResponse);
28     }
29
30     //根据问题id获取回复
31     @CrossOrigin
32     @GetMapping("/{questionId}")
33     public ResponseEntity<List<ResponseDto>> getResponsesForQuestion(@PathVariable Long questionId) {
34         List<ResponseDto> responses = responseServiceImpl.getResponsesForQuestion(questionId);
35         return ResponseEntity.ok(responses);
36     }
37
38     //根据回复id删除回复
39     @CrossOrigin
40     @DeleteMapping("/{id}")
41     public ResponseEntity<> deleteResponse(@PathVariable Long id) {
42         responseServiceImpl.deleteResponse(id);
43         return ResponseEntity.ok().build();
44     }
45 }
46
```

SurveyController层

```
13 import org.springframework.http.ResponseEntity;
14 import org.springframework.web.bind.annotation.*;
15 import java.io.FileInputStream;
16 import java.io.IOException;
17 import java.util.List;
18
19 //
20 @RestController
21 @RequestMapping("/api/surveys")
22 @CrossOrigin(origins = "http://localhost:8081")
23 public class SurveyController {
24
25     private final SurveyServiceImpl surveyService; 12个用法
26     private final ExcelFileCreator excelFileCreator; 2个用法
27     private static final Logger logger = LoggerFactory.getLogger(SurveyController.class); 1个用法
28
29     public SurveyController(SurveyServiceImpl surveyService, ExcelFileCreator excelFileCreator) {
30         this.surveyService = surveyService;
31
32
33         this.excelFileCreator = excelFileCreator;
34     }
35
36     // 问卷的增加
37     @CrossOrigin(origins = "http://localhost:8081")
38     @PostMapping("/create")
39     public ResponseEntity<SurveyDto> createSurvey(@RequestBody SurveyDto surveyDto) {
40         SurveyDto createdSurvey = surveyService.createSurvey(surveyDto);
41         return ResponseEntity.ok(createdSurvey);
42     }
43
44
45     // 根据id获取问卷
46     @CrossOrigin
47     @GetMapping("/{id}")
48     public ResponseEntity<SurveyDto> getSurveyById(@PathVariable Long id) {
49         SurveyDto survey = surveyService.getSurveyById(id);
50         return ResponseEntity.ok(survey);
51     }
52
53     // 根据用户id获取所有问卷
```


项目		
	© TestController	13
	© UserController	14
▼	DTO	15
	© LoginDto	16
	© LoginResponse	17
	© OptionDto	18
	© QuestionDto	19
	© QuestionResponse	20
	© ResponseDto	21
	© SurveyDto	22
	© SurveyStateDto	23
	© UserDto	24
▼	entity	25
	© Option	26
	© OptionExample	27
	© Question	28
	© QuestionExample	29
	© Response	30
	© ResponseExample	31
	© Survey	32
	© SurveyExample	33
	© SurveyState	34
	© SurveyStateExample	35
	© User	36
	© UserExample	37
	© UserWithRole	38
▼	mapper	39
		40
		41
		42

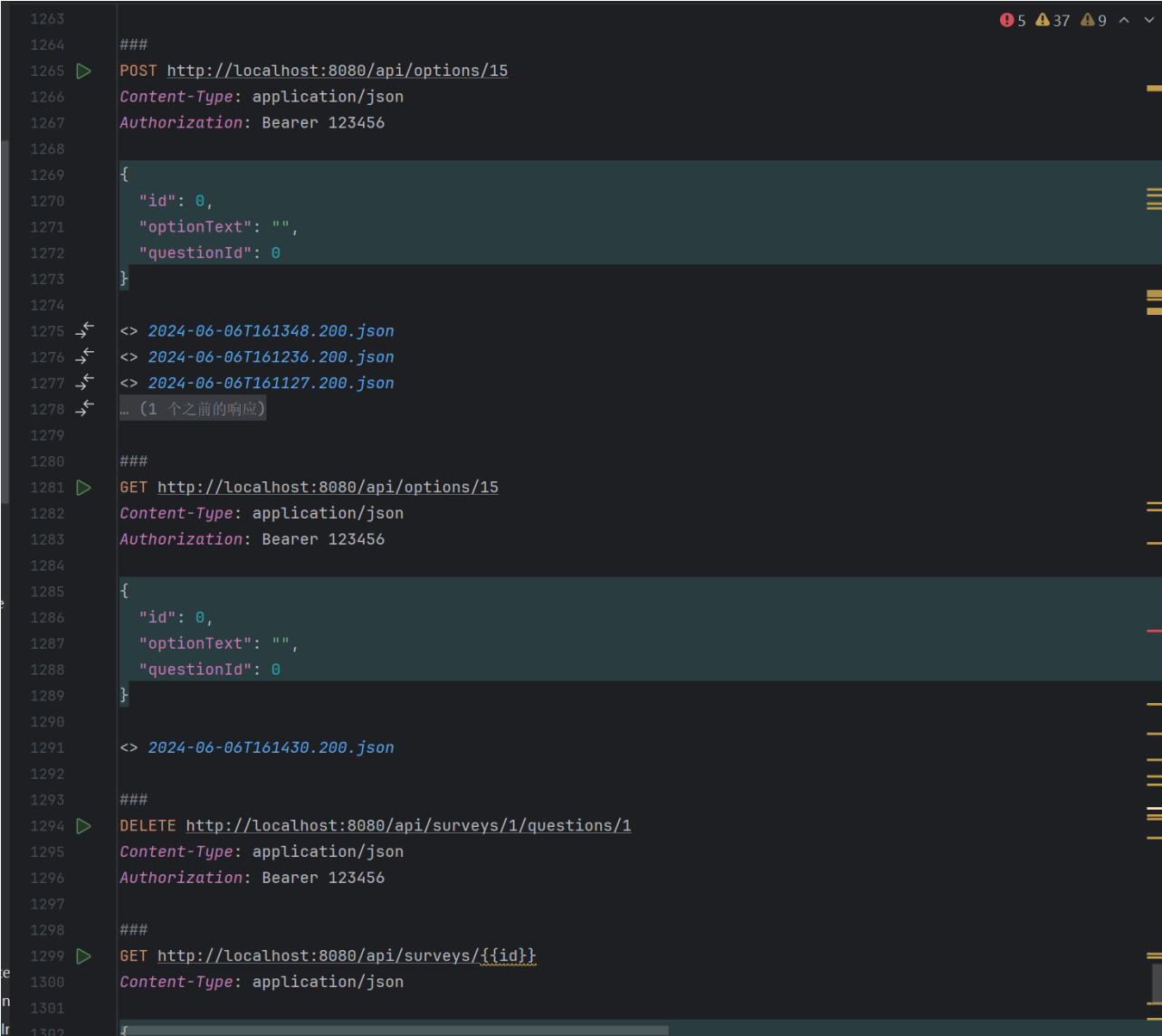
OptionMapper	43
QuestionMapper	44
ResponseMapper	45
SurveyMapper	46
SurveyStateMapper	47
UserMapper	48
service	49
serviceInterface	50
OptionServiceInte	51
QuestionServiceIn	52
ResponseServiceI	53

代码统计（后端全部代码）

Statistic										
Statistic										
Refresh Refresh on selection Settings										
Overview java properties xml										
Extension	Count	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG	Lines CODE
cmd (CMD files)	1x	7kB	7kB	7kB	7kB	205	205	205	205	169
java (Java classes)	62x	251kB	0kB	23kB	4kB	7752	1	762	125	4685
properties (Java properties files)	1x	0kB	0kB	0kB	0kB	2	2	2	2	2
sql (SQL files)	12x	11kB	0kB	3kB	0kB	265	3	58	22	230
xlsx (XLSX files)	1x	3kB	3kB	3kB	3kB	17	17	17	17	16
xml (XML configuration file)	8x	79kB	3kB	13kB	9kB	2124	68	355	265	1502
yaml (YML files)	2x	2kB	0kB	1kB	1kB	105	47	58	52	81
Total: 87x 355kB 15kB 52kB 27kB 10470 343 1457										

Squaretest Trial - 20 Days Remaining
 Please visit squaretest.com to purchase a license.

在IDE中进行http请求测试过程截图



```
1263
1264 ###
1265 POST http://localhost:8080/api/options/15
1266 Content-Type: application/json
1267 Authorization: Bearer 123456
1268
1269 {
1270   "id": 0,
1271   "optionText": "",
1272   "questionId": 0
1273 }
1274
1275 <> 2024-06-06T161348.200.json
1276 <> 2024-06-06T161236.200.json
1277 <> 2024-06-06T161127.200.json
1278 ... (1 个之前的响应)
1279
1280 ###
1281 GET http://localhost:8080/api/options/15
1282 Content-Type: application/json
1283 Authorization: Bearer 123456
1284
1285 {
1286   "id": 0,
1287   "optionText": "",
1288   "questionId": 0
1289 }
1290
1291 <> 2024-06-06T161430.200.json
1292
1293 ###
1294 DELETE http://localhost:8080/api/surveys/1/questions/1
1295 Content-Type: application/json
1296 Authorization: Bearer 123456
1297
1298 ###
1299 GET http://localhost:8080/api/surveys/{{id}}
1300 Content-Type: application/json
1301
1302
```

数据库表结构截图

在数据库设计过程中，为了避免一些id对应错误，从用户到选项/回答引入了一系列外键依赖，增加数据的可靠性

- ▼ 表 6
 - ▼ options
 - > 列 5
 - > 键 1
 - > 外键 1
 - > 索引 2
 - ▼ questions
 - ▼ 列 5
 - 🔑 id bigint (auto increment)
 - 🔑 text varchar(255)
 - 🔑 type varchar(255)
 - 🔑 survey_id bigint
 - 🔑 order bigint
 - > 键 1
 - > 索引 2
 - ▼ responses
 - > 列 3
 - > 键 1
 - > 外键 1
 - > 索引 2
 - ▼ surveys
 - > 列 5
 - > 键 1
 - > 外键 1
 - > 索引 2
 - ▼ surveys_state
 - ▼ 列 4
 - 🔑 survey_id bigint
 - 🔑 id int (auto increment)
 - 🔑 receiveNumber int
 - 🔑 state varchar(255)
 - > 键 2
 - > 外键 1
 - > 索引 2
 - ▼ users

代码数据统计

Extension ^	Count	Size SUM	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG	Lines CODE
cmd (CMD files)	1x	7kB	7kB	7kB	7kB	205	205	205	205	169
java (Java classes)	62x	251kB	0kB	23kB	4kB	7752	1	762	125	4685
properties (Java properties files)	1x	0kB	0kB	0kB	0kB	2	2	2	2	2
sql (SQL files)	12x	11kB	0kB	3kB	0kB	265	3	58	22	230
xlsx (XLSX files)	1x	3kB	3kB	3kB	3kB	17	17	17	17	16
xml (XML configuration file)	8x	79kB	3kB	13kB	9kB	2124	68	355	265	1502
yaml (YAML files)	2x	2kB	0kB	1kB	1kB	105	47	58	52	81
Total:	87x	355kB	15kB	52kB	27kB	10470	343	1457	688	

开发遇到的困难/坚决

数据库操作的熟悉

在开发过程中，由于对数据库中一些关键字的不熟悉（最经典的order），在书写过程中忘记加反引号，导致的各种错误

对数据库text类型的数据，进行传输时，变量类型的不匹配、查询方法的错误使用，导致数据的丢失（我觉得这也是分层复杂的一个比较容易导致问题的点）

http请求控制

在提出使用SpringSecurity和DynamicRoleFilter对http请求进行过滤和权限控制时，由于网络安全方面的知识欠缺，在和实际开发此功能的同学商议时遇到较大障碍，（例如如何区分游客、问卷创建者、开发人员的身份，以及对不同身份的人具体应该开放哪些api），并且在实际开发过程中由于知识的不全面，常误操作对测试过程造成较大阻碍。

但经过一番调教，我们约定开发者使用统一的内部的api，并且，经过配置，开发者可以无阻碍的使用所有api以开发测试。

响应式数据对页面的动态刷新困扰

在和前端开发者进行对接商议时，我们发现，在数据库数据发生更改或者重新获取数据用于动态渲染页面时，总是会发生数据刷新不及时，甚至在预计有数据的位置访问不到相应的数据。

基于此，我们决定将数据传输对象进行更改合并，让前端的数据获取和刷新减少循环、嵌套深度，尽力保证一次性全部相关数据的获取，将一定的操作放在后端服务层进行处理来反馈给前端

剪枝

在没有进行正式对接前，我们各自独立开发，后端有许多多余、重复的api实现，以及一些api的可用性不高，同时造成前端开发时对api产生混淆，多个api的组合使用，增加代码量

在不断的对接过程中，我进一步区分DTO和DAO层，避免将不必要的数据传送到前端，将DAO数据进行各种组合、拆分为更加适配需求的DTO，它们的差异化不断增大，各自服务于不同的需求，并且抛弃一些多余的api，对简单功能的api进行合并。

收获

开发过程中，我深刻理解了分层开发、前后端分离开发模式的优势所在，将功能和数据进行封装，前端和后端分别对应不同的数据模型，我选择将这二中数据模型的转换放在Service层，这样开发中利用数据库对应的数据库实体类，对数据进行直接快捷的操作，当数据传输模型因为前端或者后端的不同的要求需要做出改变时，对数据库层的影响很小，只需要在Service进行合适的转换即可；同时前后端分离，前端直接调用api，并且参考apifox共享的相关数据传输模型，可以与后端进行隔绝，无需理会后端的实现逻辑，当后端逻辑功能发生改变，只要api和数据传输模型(dto)不发生变化，前端开发就不会受到影响，前后端可以更加专注于自己负责板块的实现。

在和前端对接过程中，我也了解到，（vue3）前端的路由、页面跳转、事件监听、组件渲染

这种开发模式，极大的简化了代码的修改复杂程度，更加灵活多变，各个层次分工明确，代码不混杂在一起，从程序员角度也真是一种美如画的开发过程，