

**Федеральное государственное автономное образовательное
учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники**



**Лабораторная работа №1
по дисциплине
Сервисо-ориентированная архитектура**

Выполнили Студенты группы Р3412
Кобелев Р.П. Балин А.А.
к. т. н. Преподаватель:
Кривоносов Е.Д.

г. Санкт-Петербург
2025г.

Содержание

1	Задание	2
2	Выполнение	3
2.1	MusicBand Service	3
2.2	Grammy Service	15
3	Вывод	17

1 Задание

Разработать спецификацию в формате OpenAPI для набора веб-сервисов, реализующего следующую функциональность:

Первый веб-сервис должен осуществлять управление коллекцией объектов. В коллекции необходимо хранить объекты класса **MusicBand**, описание которого приведено ниже:

MusicBandService.java

```
1 public class MusicBand {
2     private Integer id; //Поле не может быть null, Значение поля должно быть больше
   ↳ 0, Значение этого поля должно быть уникальным, Значение этого поля должно
   ↳ генерироваться автоматически
3     private String name; //Поле не может быть null, Строка не может быть пустой
4     private Coordinates coordinates; //Поле не может быть null
5     private java.time.LocalDate creationDate; //Поле не может быть null, Значение
   ↳ этого поля должно генерироваться автоматически
6     private Integer numberOfParticipants; //Поле не может быть null, Значение поля
   ↳ должно быть больше 0
7     private Integer albumsCount; //Поле не может быть null, Значение поля должно
   ↳ быть больше 0
8     private String description; //Поле не может быть null
9     private MusicGenre genre; //Поле не может быть null
10    private Album bestAlbum; //Поле может быть null
11 }
12 public class Coordinates {
13     private float x; //Значение поля должно быть больше -975
14     private long y;
15 }
16 public class Album {
17     private String name; //Поле не может быть null, Строка не может быть пустой
18     private Long tracks; //Поле не может быть null, Значение поля должно быть
   ↳ больше 0
19 }
20 public enum MusicGenre {
21     PSYCHEDELIC_CLOUD_RAP,
22     SOUL,
23     POP;
24 }
```

Веб-сервис должен удовлетворять следующим требованиям:

- API, реализуемый сервисом, должен соответствовать рекомендациям подхода RESTful.
- Необходимо реализовать следующий базовый набор операций с объектами коллекции: добавление нового элемента, получение элемента по ИД, обновление элемента, удаление элемента, получение массива элементов.
- Операция, выполняемая над объектом коллекции, должна определяться методом HTTP-запроса.
- Операция получения массива элементов должна поддерживать возможность сортировки и фильтрации по любой комбинации полей класса, а также возможность分页 вывода результатов выборки с указанием размера и порядкового номера выводимой страницы.
- Все параметры, необходимые для выполнения операции, должны передаваться в URL запроса.
- Информация об объектах коллекции должна передаваться в формате **xml**.
- В случае передачи сервису данных, нарушающих заданные на уровне класса ограничения целостности, сервис должен возвращать код ответа http, соответствующий произошедшей ошибке.

Помимо базового набора, веб-сервис должен поддерживать следующие операции над объектами коллекции:

- Удалить все объекты, значение поля `description` которого эквивалентно заданному.
- Удалить один (любой) объект, значение поля `genre` которого эквивалентно заданному.
- Вернуть количество объектов, значение поля `bestAlbum` которых больше заданного.

Эти операции должны размещаться на отдельных URL.

Второй веб-сервис должен располагаться на URL `/grammy`, и реализовывать ряд дополнительных операций, связанных с вызовом API первого сервиса:

- `/band/band-id/singles/add` : добавить новый сингл указанной группе
- `/band/band-id/participants/add` : добавить в группу нового участника

Эти операции также должны размещаться на отдельных URL.

Для разработанной спецификации необходимо сгенерировать интерактивную веб-документацию с помощью Swagger UI. Документация должна содержать описание всех REST API обоих сервисов с текстовым описанием функциональности каждой операции. Созданную веб-документацию необходимо развернуть на сервере *helios*.

2 Выполнение

Для выполнения была выбрана 3.1.1 версия OpenAPI спецификации.

2.1 MusicBand Service

music.yaml

```
1 openapi: 3.1.1
2
3 info:
4   title: Musicband Service Endpoints - OpenAPI 3.1.1
5
6   description: |-
7     **OpenAPI** specification for **Laboratory work №1** of **Service-oriented
8     ↪ architecture** at **ITMO-University**.
9
10  contact:
11    name: Roman Kobelev & Balin Artem
12    url: https://github.com/ta4ilka69/Service-oriented-architecture
13
14  license:
15    name: The MIT License
16    url: https://mit-license.org/
17
18  version: 0.0.1
19
20  servers:
21    - url: 'https://localhost:5252'
22      description: Helios
23
24  paths:
25    /music-bands:
```

```

25  get:
26      summary: Get list of musicbands.
27      description: Get list of musicbands with possible filtering, sorting and
    ↪ pagination.
28
29      parameters:
30          - name: sort
31            in: query
32            description: Fields to sort by. Prefix with '-' for descending order. If
    ↪ field is used multiple times (both with and without prefix), the last
    ↪ occurrence takes precedence.
33            required: false
34            schema:
35                type: array
36                items:
37                    $ref: '#/components/schemas/SortValues'
38
39          - name: page
40            in: query
41            description: Page number. If provided, size must also be provided.
    ↪ Otherwise, page is ignored.
42            required: false
43            schema:
44                type: integer
45                format: int32
46                minimum: 1
47                default: 1
48
49          - name: size
50            in: query
51            description: Number of items per page. If page is provided, size must
    ↪ also be provided. Otherwise, size is ignored.
52            required: false
53            schema:
54                type: integer
55                format: int32
56                minimum: 1
57                default: 10
58
59          - name: filter
60            in: query
61            description: Filter conditions in the format field(>|<=>|<≠|=|^|$=|
    ↪ @=)value. "^=" is for prefix matching, "$=" is for suffix matching, and "@="
    ↪ is for substring matching. If a non-numeric field is used with a numeric
    ↪ operator or date format is invalid, an error is returned.
62            required: false
63            schema:
64                type: array
65                items:
66                    $ref: '#/components/schemas/FilterCondition'
67
68      responses:
69          '200':
70              description: A list of musicbands.
71              content:
72                  application/xml:

```

```

73         schema:
74             type: array
75             xml:
76                 name: musicBands
77                 wrapped: true
78             items:
79                 $ref: '#/components/schemas/MusicBandAllSchema'
80
81     '400':
82         $ref: '#/components/responses/BadRequestInvalidQueryParameters'
83     '500':
84         $ref: '#/components/responses/InternalServerError'
85     default:
86         description: Internal Server Error
87         content:
88             application/xml:
89                 schema:
90                     $ref: '#/components/schemas/Error'
91
92 post:
93     summary: Create a new musicband.
94     description: Create a new musicband with the provided details.
95
96     requestBody:
97         description: Musicband to add.
98         required: true
99         content:
100             application/xml:
101                 schema:
102                     $ref: '#/components/schemas/MusicBand'
103
104     responses:
105         '201':
106             description: Musicband created successfully.
107             content:
108                 application/xml:
109                     schema:
110                         $ref: '#/components/schemas/MusicBandAllSchema'
111
112         '400':
113             $ref: '#/components/responses/BadRequestInvalidInput'
114
115         '500':
116             $ref: '#/components/responses/InternalServerError'
117     default:
118         description: Internal Server Error
119         content:
120             application/xml:
121                 schema:
122                     $ref: '#/components/schemas/Error'
123
124 /music-bands/{id}:
125 get:
126     summary: Get a musicband by ID.
127     description: Retrieve a single musicband by its unique ID.
128

```

```

129     parameters:
130     - name: id
131       in: path
132       description: The unique identifier of the musicband.
133       required: true
134       schema:
135         $ref: '#/components/schemas/MusicBandId'
136
137     responses:
138       '200':
139         description: A single musicband.
140         content:
141           application/xml:
142             schema:
143               $ref: '#/components/schemas/MusicBandAllSchema'
144
145       '400':
146         $ref: '#/components/responses/BadRequestInvalidID'
147
148       '404':
149         $ref: '#/components/responses/MusicBandNotFound'
150
151       '422':
152         $ref: '#/components/responses/InvalidIdFormat'
153
154       '500':
155         $ref: '#/components/responses/InternalServerError'
156     default:
157       description: Internal Server Error
158       content:
159         application/xml:
160           schema:
161             $ref: '#/components/schemas/Error'
162
163   put:
164     summary: Update an existing musicband (with all fields).
165     description: Update an existing musicband by its ID. All fields must be
166     ↪ provided. Missing fields will be set to null.
167
168     parameters:
169     - name: id
170       in: path
171       description: The unique identifier of the musicband to update.
172       required: true
173       schema:
174         $ref: '#/components/schemas/MusicBandId'
175
176     requestBody:
177       description: Updated musicband details.
178       required: true
179       content:
180         application/xml:
181           schema:
182             $ref: '#/components/schemas/MusicBand'
183
184     responses:
185       '200':

```

```

184         description: Musicband updated successfully.
185         content:
186             application/xml:
187                 schema:
188                     $ref: '#/components/schemas/MusicBandAllSchema'
189
190         '400':
191             $ref: '#/components/responses/BadRequestInvalidInput'
192
193         '404':
194             $ref: '#/components/responses/MusicBandNotFound'
195         '422':
196             $ref: '#/components/responses/InvalidIdFormat'
197         '500':
198             $ref: '#/components/responses/InternalServerError'
199     default:
200         description: Internal Server Error
201         content:
202             application/xml:
203                 schema:
204                     $ref: '#/components/schemas/Error'
205 delete:
206     summary: Delete a musicband by ID.
207     description: Delete a single musicband by its unique ID.
208
209     parameters:
210         - name: id
211           in: path
212           description: The unique identifier of the musicband to delete.
213           required: true
214           schema:
215               $ref: '#/components/schemas/MusicBandId'
216
217     responses:
218         '204':
219             description: Musicband deleted successfully. No content returned.
220
221         '400':
222             $ref: '#/components/responses/BadRequestInvalidID'
223
224         '404':
225             $ref: '#/components/responses/MusicBandNotFound'
226         '422':
227             $ref: '#/components/responses/InvalidIdFormat'
228         '500':
229             $ref: '#/components/responses/InternalServerError'
230     default:
231         description: Internal Server Error
232         content:
233             application/xml:
234                 schema:
235                     $ref: '#/components/schemas/Error'
236
237 patch:
238     summary: Partially update a musicband by ID.

```



```

239     description: Update one or more fields of an existing musicband by its ID.
    ↳ Only the provided fields will be updated; other fields will remain unchanged.
240
241     parameters:
242     - name: id
243       in: path
244       description: The unique identifier of the musicband to update.
245       required: true
246       schema:
247         $ref: '#/components/schemas/MusicBandId'
248     requestBody:
249       description: Fields to update in the musicband.
250       required: true
251       content:
252         application/xml:
253           schema:
254             $ref: '#/components/schemas/MusicBandPatch'
255     responses:
256       '200':
257         description: Musicband updated successfully.
258         content:
259           application/xml:
260             schema:
261               $ref: '#/components/schemas/MusicBandAllSchema'
262
263       '400':
264         $ref: '#/components/responses/BadRequestInvalidInput'
265
266       '404':
267         $ref: '#/components/responses/MusicBandNotFound'
268
269       '422':
270         $ref: '#/components/responses/InvalidIdFormat'
271
272       '500':
273         $ref: '#/components/responses/InternalServerError'
274     default:
275       description: Internal Server Error
276       content:
277         application/xml:
278           schema:
279             $ref: '#/components/schemas/Error'
280
281 /music-bands/all-with-description:
282 delete:
283   summary: Delete all musicbands with a specific description.
284   description: Delete all musicbands whose description matches the provided
    ↳ value. If nothing is deleted, it means success too.
285
286   parameters:
287   - name: description
288     in: query
289     description: The description to filter musicbands for deletion.
290     required: true
291     schema:
292       $ref: '#/components/schemas/MusicBandDescription'
293
294   responses:

```

```

293     '204':
294         description: Musicbands deleted successfully. No content returned.
295
296     '400':
297         description: Bad Request - Missing or invalid description parameter.
298         $ref: '#/components/responses/BadRequestInvalidQueryParameters'
299
300     '500':
301         $ref: '#/components/responses/InternalServerError'
302     default:
303         description: Internal Server Error
304         content:
305             application/xml:
306                 schema:
307                     $ref: '#/components/schemas/Error'
308
309 /music-bands/one-with-genre:
310     delete:
311         summary: Delete one musicband with a specific genre.
312         description: Delete one random musicband whose genre matches the provided
313         ↪ value. Input is case-sensitive. If nothing is deleted, it means success too.
314
315     parameters:
316     - name: genre
317       in: query
318       description: The genre to filter musicbands for deletion.
319       required: true
320       schema:
321         $ref: '#/components/schemas/MusicBandGenre'
322
323     responses:
324     '204':
325         description: Musicband deleted successfully. No content returned.
326
327     '400':
328         description: Bad Request - Missing or invalid genre parameter.
329         $ref: '#/components/responses/BadRequestInvalidQueryParameters'
330
331     '500':
332         $ref: '#/components/responses/InternalServerError'
333     default:
334         description: Internal Server Error
335         content:
336             application/xml:
337                 schema:
338                     $ref: '#/components/schemas/Error'
339
340 /music-bands/count-best-album:
341     get:
342         summary: Count musicbands with best album greater than provided.
343         description: Get the count of musicbands whose best album has more tracks
344         ↪ than the specified number. If equals, uses alphabetical comparison of the
345         ↪ names.
346
347     parameters:
348     - name: albumName

```

```

346         in: query
347         required: true
348         description: The name of the album to compare against.
349         schema:
350             type: string
351             minLength: 1
352     - name: albumTracks
353       in: query
354       required: true
355       description: The number of tracks in the album to compare against.
356       schema:
357         type: integer
358         format: int64
359         minimum: 1
360
361     responses:
362       '200':
363         description: Count of musicbands with best album tracks greater than the
364         ↪ content: specified number.
365           application/xml:
366             schema:
367               xml:
368                 name: count
369                 type: integer
370                 format: int64
371                 example: 5
372
373       '400':
374         description: Bad Request - Missing or invalid tracks parameter.
375         $ref: '#/components/responses/BadRequestInvalidQueryParameters'
376
377       '500':
378         $ref: '#/components/responses/InternalServerError'
379     default:
380       description: Internal Server Error
381       content:
382         application/xml:
383           schema:
384             $ref: '#/components/schemas/Error'
385
386 components:
387   schemas:
388     Album:
389       type: object
390       xml:
391         name: album
392       properties:
393         name:
394           type: string
395           minLength: 1
396         tracks:
397           type: integer
398           format: int64
399           minimum: 1
400       required:

```

```

401         - name
402         - tracks
403
404     Coordinates:
405         type: object
406         properties:
407             x:
408                 type: number
409                 format: float
410                 exclusiveMinimum: -975
411             y:
412                 type: integer
413                 format: int64
414                 minimum: 1
415         required:
416             - x
417             - y
418
419     MusicBand:
420         type: object
421         xml:
422             name: musicBand
423         properties:
424             name:
425                 type: string
426                 minLength: 1
427             coordinates:
428                 $ref: '#/components/schemas/Coordinates'
429             numberOfParticipants:
430                 type: integer
431                 format: int32
432                 minimum: 1
433             albumsCount:
434                 type: integer
435                 format: int32
436                 minimum: 1
437             description:
438                 $ref: '#/components/schemas/MusicBandDescription'
439             genre:
440                 $ref: '#/components/schemas/MusicBandGenre'
441             bestAlbum:
442                 $ref: '#/components/schemas/Album'
443             nullable: true
444         required:
445             - name
446             - coordinates
447             - numberOfParticipants
448             - albumsCount
449             - description
450             - genre
451     MusicBandAllSchema:
452         allOf:
453             - $ref: '#/components/schemas/MusicBand'
454             - type: object
455               required:
456                 - id

```

```

457         properties:
458             id:
459                 type: integer
460                 format: int32
461                 minimum: 1
462             creationDate:
463                 type: string
464                 format: date
465     xml:
466         name: musicBandAllSchema
467 MusicBandPatch:
468     type: object
469     xml:
470         name: musicBand
471     properties:
472         name:
473             type: string
474             minLength: 1
475         coordinates:
476             $ref: '#/components/schemas/Coordinates'
477         numberOfParticipants:
478             type: integer
479             format: int32
480             minimum: 1
481         albumsCount:
482             type: integer
483             format: int32
484             minimum: 1
485         description:
486             $ref: '#/components/schemas/MusicBandDescription'
487         genre:
488             $ref: '#/components/schemas/MusicBandGenre'
489         bestAlbum:
490             $ref: '#/components/schemas/Album'
491         nullable: true
492
493 Error:
494     type: object
495     required:
496         - code
497         - message
498     xml:
499         name: error
500     properties:
501         code:
502             type: integer
503             format: int32
504             example: 500
505         message:
506             type: string
507             example: Internal Server Error
508
509 InternalServerError:
510     allOf:
511         - $ref: '#/components/schemas/Error'
512         - type: object

```

```

513         properties:
514             code:
515                 example: 500
516             message:
517                 example: Internal Server Error
518 NotFoundError:
519     allOf:
520     - $ref: '#/components/schemas/Error'
521     - type: object
522       properties:
523           code:
524               example: 404
525           message:
526               example: Not Found
527 BadRequestError:
528     allOf:
529     - $ref: '#/components/schemas/Error'
530     - type: object
531       properties:
532           code:
533               example: 400
534           message:
535               example: Bad Request
536 UnprocessableEntityError:
537     allOf:
538     - $ref: '#/components/schemas/Error'
539     - type: object
540       properties:
541           code:
542               example: 422
543           message:
544               example: Unprocessable Entity
545
546 FilterCondition:
547     type: string
548     pattern: '^((name|creationDate|description|genre|bestAlbum\.name)
↪ (≥|<>|≠|=|~|=$|@=).+)|(coordinates\.x(≥|<>|<≠|=)-?\d+(\.\d+)?)|(
↪ id|coordinates\.y|numberOfParticipants|albumsCount|bestAlbum\.tracks)
↪ (≥|<>|<≠|=)-?\d+)$'
549
550 SortValues:
551     type: string
552     pattern: '^((id|-id|name|-name|coordinates\.x|-coordinates\.x|coordinates\.y|-
↪ coordinates\.y|creationDate|-creationDate|numberOfParticipants|-
↪ numberOfParticipants|albumsCount|-albumsCount|description|-description|genre
↪ |-genre|bestAlbum\.name|-bestAlbum\.name|bestAlbum\.tracks|-bestAlbum\.tracks
↪ )$'
553
554 PatchOption:
555     type: string
556     xml:
557         name: field
558     pattern: '^((name|creationDate|description|genre|bestAlbum\.name)=.+)| (
↪ coordinates\.x=-?\d+(\.\d+)?)|(id|coordinates\.y|numberOfParticipants|
↪ albumsCount|bestAlbum\.tracks)=-?\d+)$'
559

```

```

560 MusicBandId:
561     type: integer
562     format: int32
563     minimum: 1
564
565 MusicBandDescription:
566     type: string
567     minLength: 0
568
569 MusicBandGenre:
570     type: string
571     enum:
572         - PSYCHEDELIC_CLOUD_RAP
573         - SOUL
574         - POP
575
576 responses:
577     InternalServerError:
578         description: Internal Server Error
579         content:
580             application/xml:
581                 schema:
582                     $ref: '#/components/schemas/InternalServerError'
583
584 MusicBandNotFound:
585     description: Not Found - Musicband not found.
586     content:
587         application/xml:
588             schema:
589                 $ref: '#/components/schemas/NotFoundError'
590
591 BadRequestInvalidID:
592     description: Bad Request - Invalid ID supplied.
593     content:
594         application/xml:
595             schema:
596                 $ref: '#/components/schemas/BadRequestError'
597 BadRequestInvalidQueryParameters:
598     description: Bad Request - Invalid Query Parameters.
599     content:
600         application/xml:
601             schema:
602                 $ref: '#/components/schemas/BadRequestError'
603
604 BadRequestInvalidInput:
605     description: Bad Request - Invalid input.
606     content:
607         application/xml:
608             schema:
609                 $ref: '#/components/schemas/BadRequestError'
610
611 InvalidIdFormat:
612     description: Parameter 'id' must be a positive integer.
613     content:
614         application/xml:
615             schema:

```

616 `$ref: '#/components/schemas/UnprocessableEntityError'`

2.2 Grammy Service

grammy.yaml

```
1 openapi: 3.1.1
2 info:
3   title: Grammy Service Endpoints - OpenAPI 3.1.1
4   description: |-
5     **OpenAPI** specification for **Laboratory work №1** of **Service-oriented
6     ↪ architecture** at **ITMO-University**.
7   contact:
8     name: Roman Kobelev & Balin Artem
9     url: https://github.com/ta4ilka69/Service-oriented-architecture
10
11  license:
12    name: The MIT License
13    url: https://mit-license.org/
14
15  version: 0.0.1
16
17  servers:
18    - url: 'https://localhost:5252/api/v1/grammy'
19      description: Helios
20
21  paths:
22    /band/{band-id}/singles/add:
23      post:
24        summary: Add a single to the band
25        description: Add a single to the band. Accepts a string in body
26
27        requestBody:
28          required: true
29          content:
30            application/xml:
31              schema:
32                $ref: '#/components/schemas/SingleSchema'
33
34        responses:
35          '201':
36            description: Created. Returns created object
37            content:
38              application/xml:
39                schema:
40                  $ref: '#/components/schemas/Single'
41
42          '400':
43            description: Bad Request - Invalid name parameter.
44            $ref: './music.yaml#/components/responses/BadRequestInvalidInput'
45          '404':
46            $ref: './music.yaml#/components/responses/MusicBandNotFound'
47          '422':
48            $ref: './music.yaml#/components/responses/InvalidIdFormat'
49          '500':
```



```

50     $ref: './music.yaml#/components/responses/InternalServerError'
51   '503':
52     $ref: './#/components/responses/ServiceUnavailable'
53   default:
54     description: Returns error code and message in case of error
55     content:
56       application/xml:
57         schema:
58           $ref: './music.yaml#/components/schemas/Error'
59
60   parameters:
61     - name: band-id
62       in: path
63       description: ID of the band to add a single to
64       required: true
65       schema:
66         $ref: './music.yaml#/components/schemas/MusicBandId'
67
68 /band/{band-id}/participants/add:
69   post:
70     summary: Add a participant to the band
71     description: Add a participant to the band
72
73   responses:
74     '201':
75       description: Created. Returns new object
76       content:
77         application/xml:
78           schema:
79             $ref: './music.yaml#/components/schemas/MusicBandAllSchema'
80     '400':
81       description: Bad Request - Invalid name parameter.
82       $ref: './music.yaml#/components/responses/BadRequestInvalidInput'
83     '404':
84       $ref: './music.yaml#/components/responses/MusicBandNotFound'
85     '422':
86       $ref: './music.yaml#/components/responses/InvalidIdFormat'
87     '500':
88       $ref: './music.yaml#/components/responses/InternalServerError'
89     '503':
90       $ref: './#/components/responses/ServiceUnavailable'
91   default:
92     description: Returns error code and message in case of error
93     content:
94       application/xml:
95         schema:
96           $ref: './music.yaml#/components/schemas/Error'
97
98   parameters:
99     - name: band-id
100       in: path
101       description: ID of the band to add a participant to
102       required: true
103       schema:
104         $ref: './music.yaml#/components/schemas/MusicBandId'
105

```

```

106     requestBody:
107       required: true
108       content:
109         application/xml:
110           schema:
111             $ref: '#/components/schemas/ParticipantSchema'
112
113
114 components:
115   schemas:
116     Single:
117       allOf:
118         - $ref: './music.yaml#/components/schemas/Album'
119       xml:
120         name: single
121     SingleSchema:
122       type: object
123       properties:
124         name:
125           type: string
126         xml:
127           name: name
128       required:
129         - name
130     ParticipantSchema:
131       type: object
132       properties:
133         name:
134           description: Имя нового участника.
135           type: string
136           minLength: 1
137       required:
138         - name
139     ServiceUnavailableError:
140       allOf:
141         - $ref: './music.yaml#/components/schemas/Error'
142         - type: object
143           properties:
144             code:
145               example: 503
146             message:
147               example: Service Unavailable
148 responses:
149   ServiceUnavailable:
150     description: Service Unavailable
151     content:
152       application/xml:
153         schema:
154           $ref: '#/components/schemas/ServiceUnavailableError'

```

3 Вывод

В ходе выполнения лабораторной работы было расписаны спецификации двух сервисов в виде **yaml** файлов. Разобрались с синтаксисом языка для написания спецификаций.