Язык Java

Осенний семестр 2023, магистратура, 1 курс

Лекция 7: Spring MVC

Содержание

HTTP-запросы

- Удобная программа для выполнения HTTP запросов: Postman
- Пример API: Open Library Search API
- Пример GET запроса: поиск книг по названию и имени автора:

```
GET http://openlibrary.org/search.json?title=Clean%20Code&author=Martins
```

  - Протокол: `http`
  - Хост: `openlibrary.org`
  - Путь к ресурсу: `/search.json`
  - Параметры поиска (query params):
    - `title=Clean%20Code`
    - `author=Martins`

Структура HTTP сообщений

- Запрос (request)

```
GET /search.json?title=Clean%20Code&author=Martins HTTP/1.1
Accept: application/json
User-Agent: PostmanRuntime/7.29.0
Postman-Token: e8a1f21a-c7fd-4724-a0c6-3852d8a09483
Host: openlibrary.org
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

HTTP заголовки (http headers)

- Ответ (response)

```
HTTP/1.1 200 OK
```

HTTP код (HTTP status code)

```
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 06 Nov 2023 13:12:52 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
access-control-allow-origin: *
access-control-allow-method: GET, OPTIONS
access-control-max-age: 86400
x-ol-stats: "SR 1 0.060 TT 0 0.061"
Referrer-Policy: no-referrer-when-downgrade
```

HTTP заголовки

```
{
"numFound": 0,
"start": 0,
"numFoundExact": true,
"docs": [],
"num_found": 0,
"q": "",
"offset": null
}
```

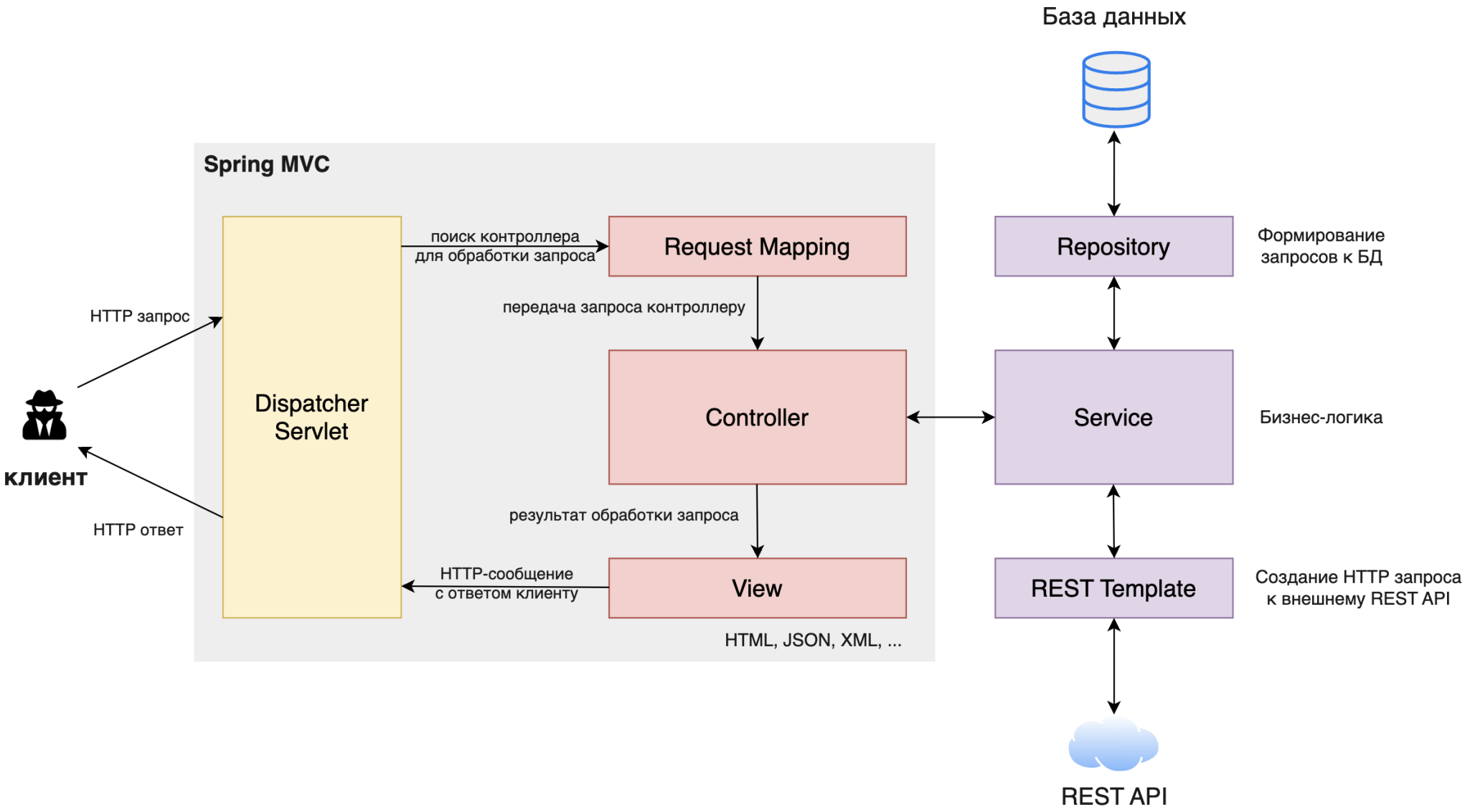
Тело HTTP ответа (в формате JSON)

Основные задачи веб-приложения

1. Следить за входящими HTTP запросами от клиентов
2. Чтение параметров HTTP-запроса
3. Валидация (проверка корректности) запроса (в т.ч. аутентификация и авторизация)
4. **Обработка запроса (чтение или запись в БД, вызов внешних API, ...)**
5. Формирование HTTP сообщения для ответа клиенту
6. Отправка HTTP-ответа клиенту

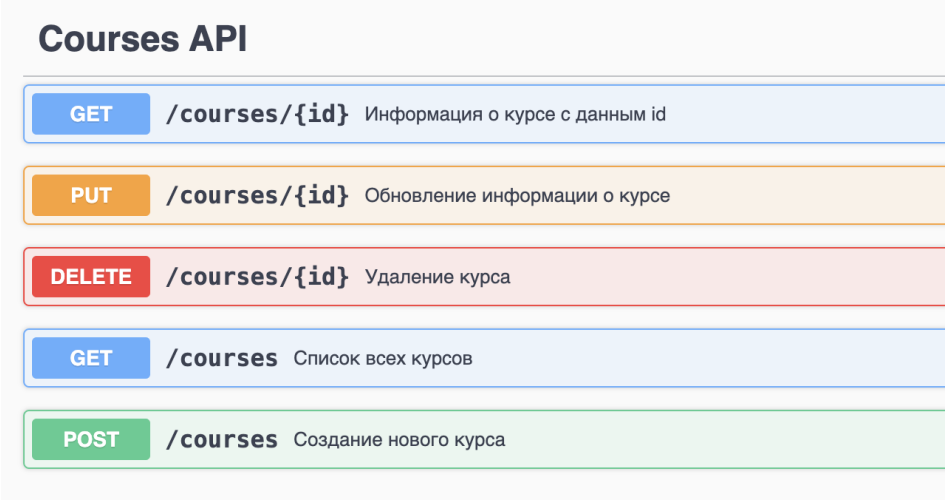
Spring MVC берет на себя обработку шагов 1-3 и 5-6.

Архитектура Spring MVC



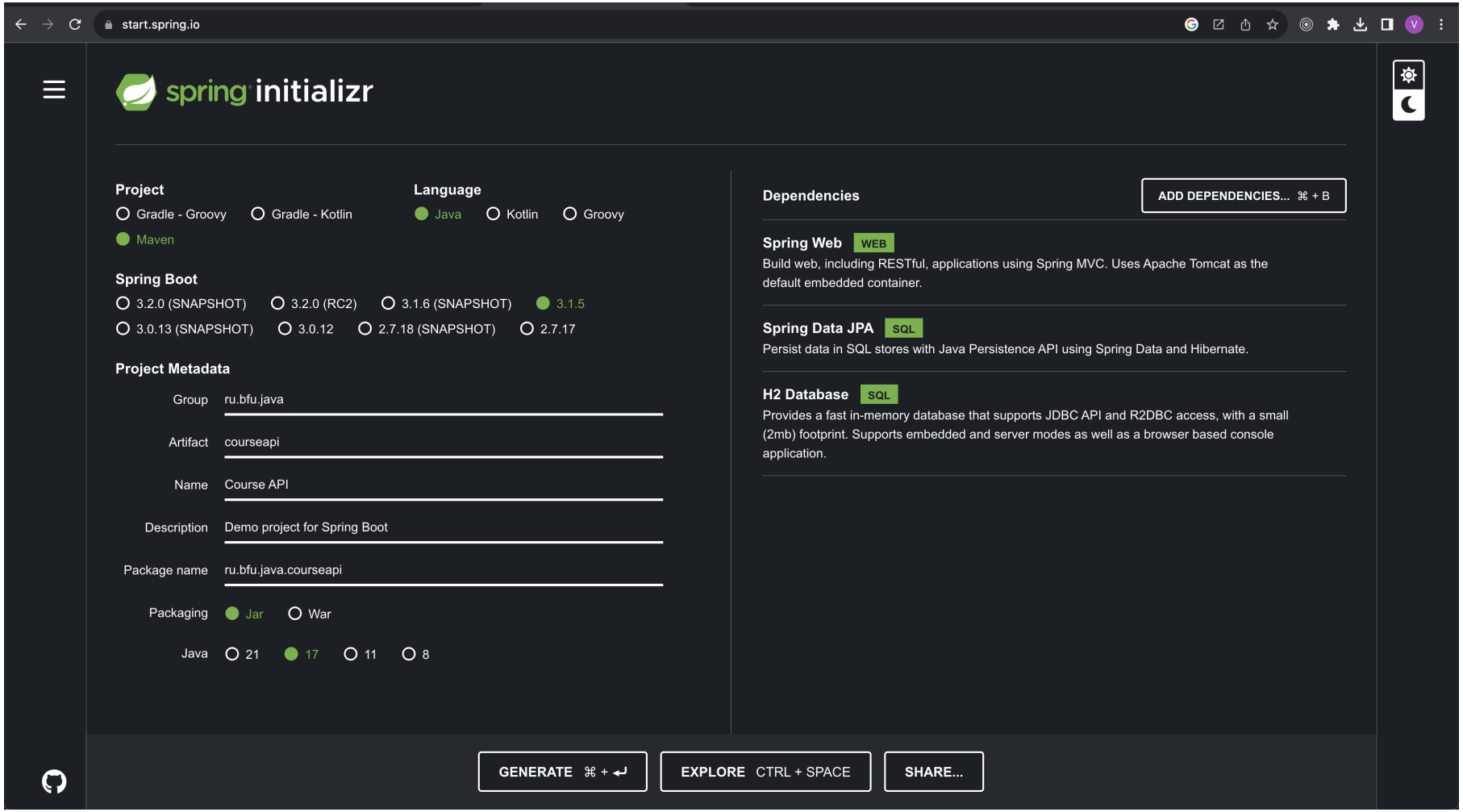
Демо-проект: "Course API"

- Создание REST API для чтения/создания/изменения/удаления информации о курсах

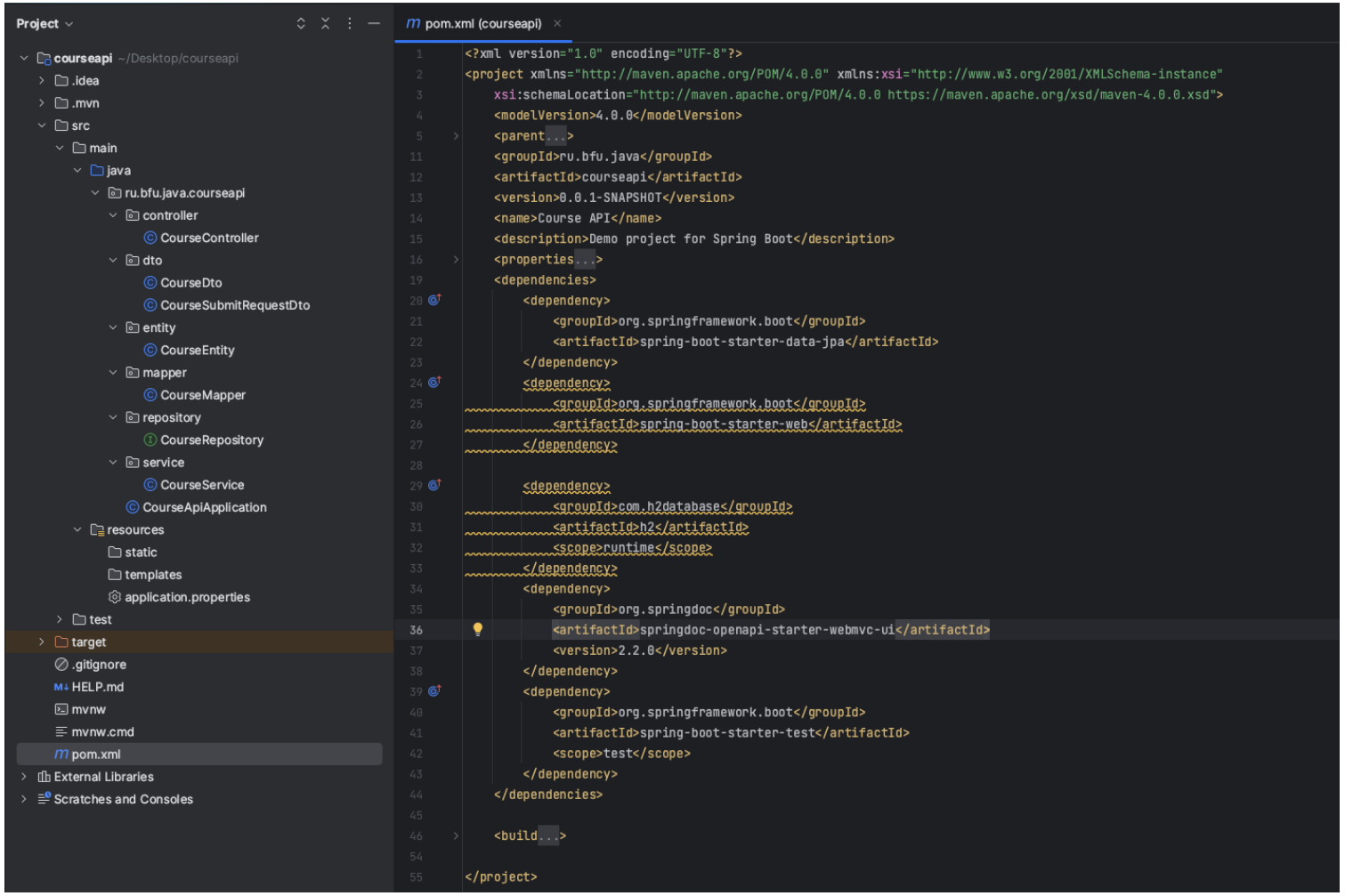


- Используемые технологии:
  - Spring MVC
  - Spring Data JPA
  - База данных H2

Создание приложения

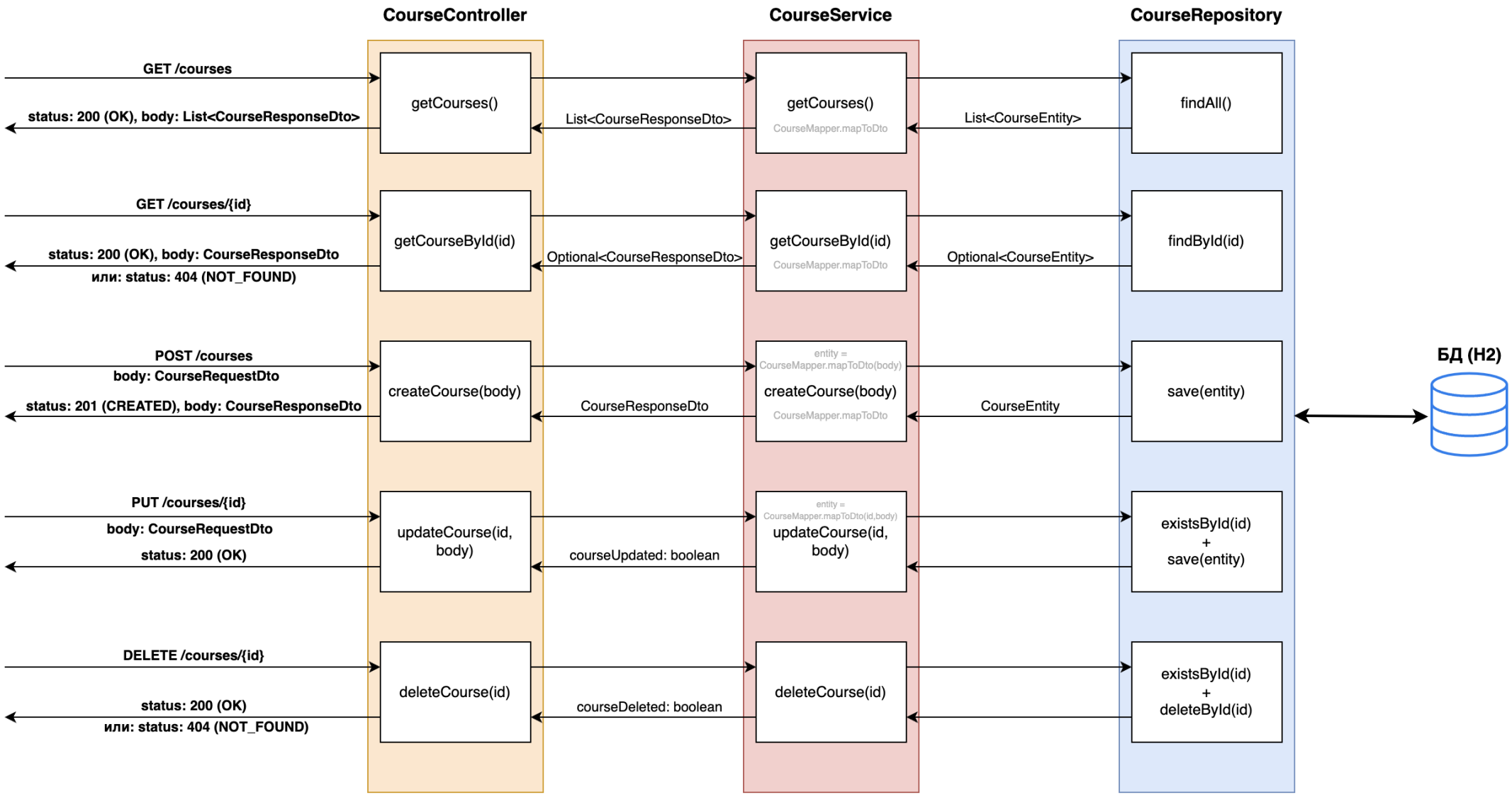


Структура приложения



Внимание: дополнительная зависимость `springdoc-openapi-starter-webmvc-ui` для автоматического создания документации API.

Архитектура приложения



CourseController

CourseController.java

```java
import ru.bfu.java.courseapi.dto.CourseRequestDto;
import ru.bfu.java.courseapi.service.CourseService;

import java.util.List;
import java.util.Optional;

no usages
@RestController
@RequestMapping("/courses")
@Tag(name = "Courses API")
public class CourseController {

    5 usages
    @Autowired
    private CourseService courseService;

    @Operation(summary = "Список всех курсов")
    @GetMapping
    public List<CourseResponseDto> getCourses() { return courseService.getCourses(); }

    @Operation(summary = "Информация о курсе с данным id")
    @GetMapping("/{id}")
    public ResponseEntity<CourseResponseDto> getCourseById(@PathVariable Long id) {...}

    @Operation(summary = "Создание нового курса")
    @PostMapping
    public ResponseEntity<CourseResponseDto> createCourse(@RequestBody CourseRequestDto courseCreateRequest) {...}

    @Operation(summary = "Обновление информации о курсе")
    @PutMapping("/{id}")
    public ResponseEntity<Void> updateCourse(@PathVariable Long id, @RequestBody CourseRequestDto courseUpdateRequest) {...}

    @Operation(summary = "Удаление курса")
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCourse(@PathVariable Long id) {...}

}
```

CourseRequestDto.java

```java
9 usages
public class CourseRequestDto {
    2 usages
    private String title;
    2 usages
    private String description;
    2 usages
    private Integer numberOfLectures;

    // GETTERS AND SETTERS
    1 usage
    public String getTitle() { return title; }

    no usages
    public void setTitle(String title) { this.title = title; }
```

CourseResponseDto.java

```java
14 usages
public class CourseResponseDto {
    2 usages
    private Long id;
    2 usages
    private String title;
    2 usages
    private String description;
    2 usages
    private Integer numberOfLectures;

    // GETTERS AND SETTERS
    no usages
    public Long getId() { return id; }

    1 usage
    public void setId(Long id) { this.id = id; }
```

`CourseEntity` - модель для таблицы БД

```java
@Entity
@Table(name = "courses")
public class CourseEntity {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    2 usages
    @Column(name = "title", nullable = false)
    private String title;

    2 usages
    @Column(name = "description", nullable = false)
    private String description;

    2 usages
    @Column(name = "number_of_lectures", nullable = false)
    private Integer numberOfLectures;

    // GETTERS AND SETTERS

    1 usage
    public Long getId() { return id; }

    1 usage
    public void setId(Long id) { this.id = id; }
```

CourseRepository

```java
@Repository
public interface CourseRepository extends JpaRepository<CourseEntity, Long> {
}
```

CourseService

```java
@Service
public class CourseService {
    7 usages
    @Autowired
    private CourseRepository courseRepository;

    1 usage
    public List<CourseResponseDto> getCourses() {
        List<CourseEntity> courseEntities = courseRepository.findAll();
        return courseEntities.stream() Stream<CourseEntity>
                .map(CourseMapper::mapToDto) Stream<CourseResponseDto>
                .toList();
    }
    1 usage
    public Optional<CourseResponseDto> getCourseById(Long id) {
        Optional<CourseEntity> courseEntityOptional = courseRepository.findById(id);
        return courseEntityOptional.map(CourseMapper::mapToDto);
    }
    1 usage
    public CourseResponseDto createCourse(CourseRequestDto courseCreateRequest) {
        CourseEntity courseEntity = CourseMapper.mapToEntity(courseCreateRequest);
        CourseEntity savedCourseEntity = courseRepository.save(courseEntity);
        return CourseMapper.mapToDto(savedCourseEntity);
    }
    1 usage
    public boolean updateCourse(Long id, CourseRequestDto courseUpdateRequest) {
        if (!courseRepository.existsById(id)) {
            return false;
        }
        CourseEntity courseEntity = CourseMapper.mapToEntity(id, courseUpdateRequest);
        courseRepository.save(courseEntity);
        return true;
    }
    1 usage
    public boolean deleteCourse(Long id) {
        if (!courseRepository.existsById(id)) {
            return false;
        }
        courseRepository.deleteById(id);
        return true;
    }
}
```

CourseMapper

```java
public final class CourseMapper {

    3 usages
    public static CourseResponseDto mapToDto(CourseEntity entity) {
        CourseResponseDto dto = new CourseResponseDto();

        dto.setId(entity.getId());
        dto.setTitle(entity.getTitle());
        dto.setDescription(entity.getDescription());
        dto.setNumberOfLectures(entity.getNumberOfLectures());

        return dto;
    }

    2 usages
    public static CourseEntity mapToEntity(CourseRequestDto requestDto) {
        CourseEntity entity = new CourseEntity();

        entity.setTitle(requestDto.getTitle());
        entity.setDescription(requestDto.getDescription());
        entity.setNumberOfLectures(requestDto.getNumberOfLectures());

        return entity;
    }

    1 usage
    public static CourseEntity mapToEntity(Long courseId, CourseRequestDto requestDto) {
        CourseEntity entity = mapToEntity(requestDto);

        entity.setId(courseId);

        return entity;
    }
}
```

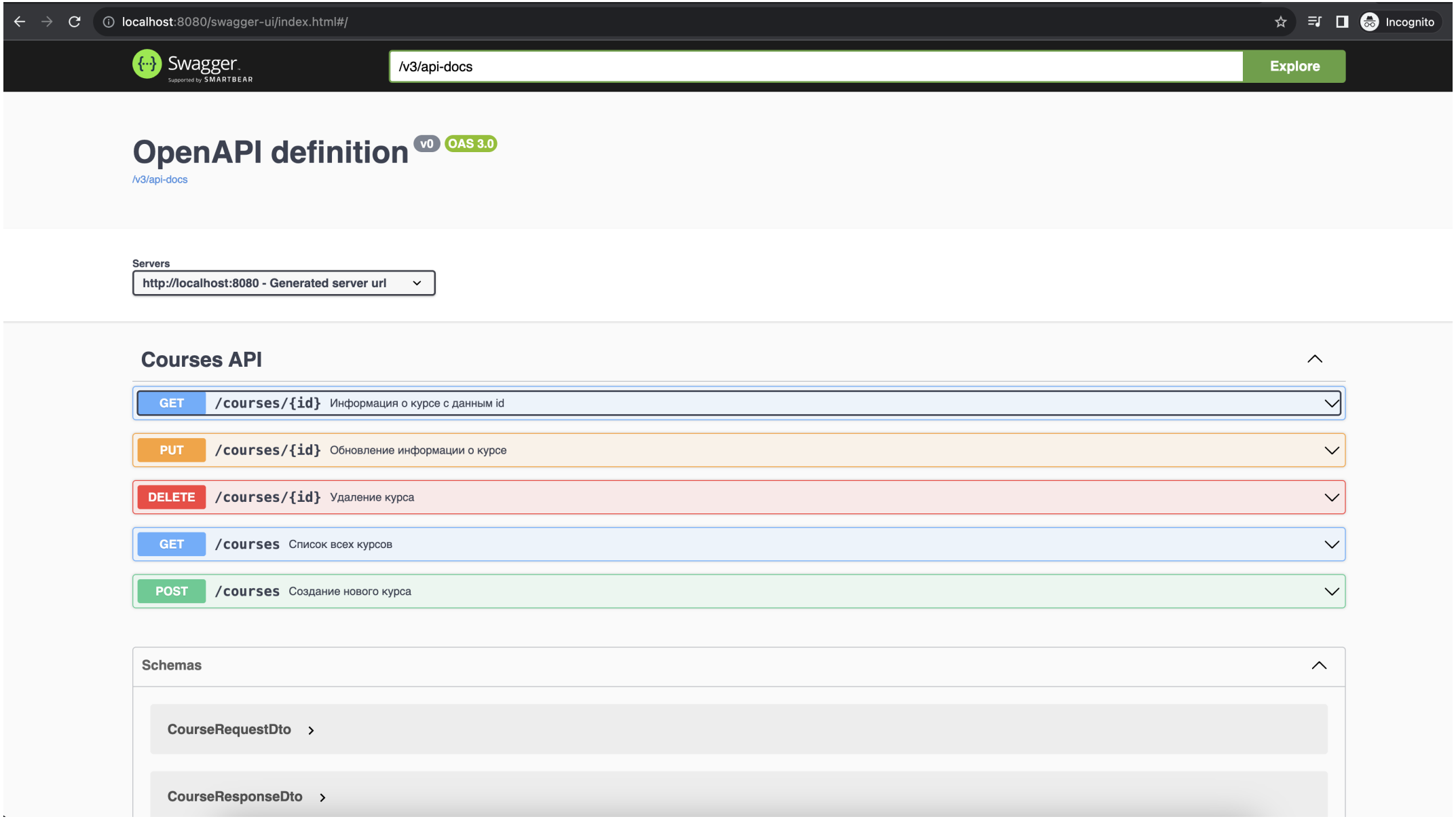
Конфигурационные параметры

application.properties

```
spring.datasource.url=jdbc:h2:file:/tmp/courseapi/db
spring.datasource.driverClassName=org.h2.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

SwaggerUI


localhost:8080/swagger-ui/index.html#/
Incognito
Swagger
Supported by SMARTBEAR
/v3/api-docs
Explore
OpenAPI definition
v0
OAS 3.0
/v3/api-docs
Servers
http://localhost:8080 - Generated server url
Courses API
GET /courses/{id} Информация о курсе с данным id
PUT /courses/{id} Обновление информации о курсе
DELETE /courses/{id} Удаление курса
GET /courses Список всех курсов
POST /courses Создание нового курса
Schemas
CourseRequestDto
CourseResponseDto

Упражнение

1. Добавить API для управления студентами (чтение данных, добавление, изменение, удаление)
2. (**Более сложное задание**) Добавить API для регистрации студентов на курсы

Демо-проект с Course API доступен здесь