1.

```java
@RestController
@RequestMapping("/rest")
public class StudentRestController
{
    @Autowired
    StudentService studentService;

    @RequestMapping("/student/view/{npm}")
    public StudentModel view(@PathVariable("npm") String npm)
    {
        StudentModel student = studentService.selectStudent(npm);
        return student;
    }

    @RequestMapping("/student/viewall")
    public List<StudentModel> viewAll()
    {
        List<StudentModel> students = studentService.selectAllStudents();
        return students;
    }
}
```

Jadi disini saya membuat Class StudentRestController yang memiliki dua method yaitu view dan viewAll; View sendiri merupakan method yang digunakan untuk menampilkan hasil query suatu mahasiswa dengan npm tertentu. Data hasil query tersebut nantinya akan disimpan dalam bentuk json. Sama halnya dengan method viewAll yang mengembalikan hasil query dari semua mahasiswa yang ada di dalam database.

2.

```java
@RestController
@RequestMapping("/rest")
public class CourseRestController
{
    @Autowired
    CourseService courseService;

    @RequestMapping("/course/view/{id}")
    public CourseModel view(@PathVariable("id") String id)
    {
        CourseModel course = courseService.selectCourse(id);
        return course;
    }

    @RequestMapping("/course/viewall")
    public List<CourseModel> viewAll()
    {
        List<CourseModel> courses = courseService.selectAllCourse();
        return courses;
    }
}
```

Sama seperti nomor 1, saya memiliki dua method untuk melihat course dengan id tertentu dan juga melihat semua course yang ada di database dengan method view dan viewAll. Method view akan mengembalikan hasil query course dengan id tertentu dari database, dan viewAll akan mengembalikan hasil query semua course yang ada di dalam database.

3.

```java
public interface StudentDAO
{
    StudentModel selectStudent(String npm);
    List<StudentModel> selectAllStudents();
}
```

```java
@Service
public class StudentDAOImpl implements StudentDAO
{
    @Autowired
    private RestTemplate restTemplate;

    @Override
    public StudentModel selectStudent(String npm) {
        StudentModel student = restTemplate.getForObject( url: "http://localhost:8080/rest/student/view/" + npm,
                StudentModel.class);
        return student;
    }

    @Override
    public List<StudentModel> selectAllStudents() {
        List<StudentModel> students = restTemplate.getForObject( url: "http://localhost:8080/rest/student/viewall", List.class);
        return students;
    }
}
```

```java
@Slf4j
@Service
@Primary
public class StudentServiceRest implements StudentService
{
    @Autowired
    private StudentDAO studentDAO;

    @Override
    public StudentModel selectStudent(String npm) { return studentDAO.selectStudent(npm); }

    @Override
    public List<StudentModel> selectAllStudents() {
        return studentDAO.selectAllStudents();
    }
```

Jadi disini saya pertama-tama membuat interface class StudentDAO yang memiliki dua method selectStudent dan selectAllStudents. Kemudian saya membuat class StudentDaoImpl yang akan mengimplementasi class StudentDAO. Dimana nantinya method selectStudent akan mengambil hasil query yg sudah di simpan dalam producer yang sudah kita buat sebelumnya dan menyimpannya dalam bentuk object yang diinginkan yaitu Student, sama halnya dengan method selectAllStudents yang berfungsi untuk mengambil hasil query dan menyimpannya semua data siswa yang sudah ada di producer.

Kemudian saya membuat class service tambahan yang mengimplementasi StudentService. Class itu menggunakan anotasi primary sehingga nantinya jika terdapat method yang ada di interface StudentService akan di prioritaskan terlebih dahulu menggunakan implementasi method di class StudentServiceRest. StudentServiceRest sendiri memiliki dua method yaitu selectStudent dan selectAllStudents yang akan mengembalikan hasil pemanggilan method dengan nama yang sama yang ada di StudentDAO yang sudah dibuat tadi.

4.

```java
public interface CourseDAO
{
    CourseModel selectCourse(String id_course);
    List<CourseModel> selectAllCourse();
}
```

```java
@Service
public class CourseDAOImpl implements CourseDAO
{
    @Autowired
    private RestTemplate restTemplate;

    @Override
    public CourseModel selectCourse(String id_course) {
        CourseModel course = restTemplate.getForObject( url: "http://localhost:8080/rest/course/view/" + id_course,
                CourseModel.class);
        return course;
    }

    @Override
    public List<CourseModel> selectAllCourse() {
        List<CourseModel> courses = restTemplate.getForObject( url: "http://localhost:8080/rest/course/viewall", List.class);
        return courses;
    }
}
```

```java
@Slf4j
@Service
@Primary
public class CourseServiceRest implements CourseService
{
    @Autowired
    private CourseDAO courseDAO;

    @Override
    public CourseModel selectCourse(String id_course) { return courseDAO.selectCourse(id_course); }

    @Override
    public List<CourseModel> selectAllCourse() {
        return courseDAO.selectAllCourse();
    }
}
```

Sama seperti nomor 3, saya membuat tiga class yaitu courseDAO courseDAOImpl dan courseServiceRest dimana courseDAOImpl mengimplementasi courseDAO dan courseServiceRest mengimplementasi courseService. CourseDAOImpl memiliki dua method yaitu selectCourse dan selectAllCourse yang nantinya akan mengambil dan menyimpan hasil object yang ada di producer yang sudah dibuat pada latihan nomor 2. Selanjutnya CourseServiceRest akan mengembalikan hasil dari method yang ada di Class CourseDAO dengan nama method yang sama.

Disini saya mempelajari bagaimana caranya menjalankan dua aplikasi dengan fugsi yang berbeda dimana producer sendiri lebih terfokus terkait backend dan consumer sendiri berfokus terhadap front end. Hal ini akan lebih mempermudah dalam pengembangan aplikasi nantinya.