

# Задание 2-8. Java Spring Project

## Темы проектов

Разработайте web приложение на основе Spring Framework :

1. **База пациентов медицинского учреждения.** Врач ищет пациента по базе, определяет диагноз, делает назначение. Пациент может получить свою историю болезни и посещений. В системе может быть роль администратора.
2. **База студентов ВУЗа.** Преподаватель может найти и выставить оценку студенту по предмету (оставить комментарий). Студент видит свою успеваемость. В системе может быть роль администратора.
3. **База личных контактов.** Можно искать, добавлять, редактировать контакты. Можно отправить сообщение по почте одному или нескольким контактом. Контакты могут быть связанные.
4. **База IT-проектов.** Пользователь может просмотреть/искать проекты. Может оставить заявку на участие в проекте. Администратор управляет базой проектов, добавляет пользователей на проект и отправляет им уведомления об отказе или приеме.
5. **Список задач.** Пользователь может просмотреть/искать задачи. Выбирать задачи для выполнения. Администратор фиксирует выполнение задачи пользователем.
6. **Список документов.** Пользователь может искать по описанию документы и скачивать. Может ставить рейтинг документу или оставлять комментарии. Администратор создает и загружает документы.
7. **Сервис подбора команды.** Администратор формирует требования к команде и информацию о мероприятии. Пользователи оставляют заявки на участие и заполняют формы. Администратор подтверждает или отклоняет заявки.
8. **Список игр и игроков (сервис поиска игрока, соперника).** Администратор оставляет заявку на игру. Пользователь ищет игру по критериям и заполняет форму на участие при желании. Администратор подтверждает или отклоняет заявки.
9. **Список программных приложений.** Администратор делает описание приложения и загружает его в систему. Пользователь может искать, фильтровать список и скачивать.
10. **Сервис по поиску/заказу блюд в ресторане(ах).** Пользователь выполняет заказ из меню, выполняет поиск блюд по фильтрам. Администратор подтверждает заказ, добавляет новые блюда.

- 11. Сервис услуг по аренде электросамокатов.** Пользователь выбирает самокат из списка доступных. Заполняет форму и срок аренды. Администратор оформляет возврат и выставляет счет.
- 12. Сервис по аренде компьютерной техники.** Пользователь выбирает/ ищет технику из списка доступных. Заполняет форму и срок аренды. Администратор оформляет возврат и выставляет счет.
- 13. Сервис поиска попутчиков (РБ, Европа, Америка).** Пользователь публикует предложение о поездке. Пользователь может искать, фильтровать предложения, подавать заявки. В системе может быть роль администратора.
- 14. Информационный ресурс поиска/публикации спортивных мероприятий.** Администратор публикует информацию о спортивном мероприятии. Пользователи ищут по критериям мероприятия. Оставляют отзывы (загружают фото).
- 15. Афиша концертров.** Администратор публикует информацию о концерте. Пользователи ищут по критериям концерты. Оставляют отзывы (загружают фото).
- 16. Афиша театров.** Администратор публикует информацию о спектаклях. Пользователи ищут по критериям спектакли (дата, театр, время, жанр, актеры и т.п.). Оставляют отзывы (загружают фото).

## Основные роли

- Должно поддерживаться минимум две роли, одна из которых администратор (используйте Spring Security)
- На основании ролей должны быть разграничены выполняемые функции

## Основные функциональные требования

- Регистрация и авторизация
- Возможность создания, просмотра, редактирования и удаления контента (в соответствии с темой)
- Поиск по различным полям / фильтрация
- Возможность отправлять уведомления по email на основании шаблона (критериев). Если приложение не предусматривает рассылку событий или информирование по email, то реализовать регистрацию/активацию пользователей через подтверждающую ссылку

## Основные формы приложения

- Формы регистрации/ авторизация
- Главная Форма – список (студентов/ контактов/ попутчиков/ техники....) с поддержкой постраничной навигации (10 или 20 на страницу)

- Форма Поиска / Фильтрации
- Форма Создания / Редактирования /Удаления

*При необходимости*

- Форма бронирования
- Форма выбора/загрузки фото
- Форма загрузки attachment (для документов, приложений)
- Форма отправки сообщений / email

Количество и структуру необходимых страниц форм/подформ определить самостоятельно.

## Технические требования

### Общие требования

- Java код должен соответствовать Java Code Convention
- Необходимо делать коммиты по завершению определенной задачи проекта для процентовки преподавателем.
- программа должна производить валидацию вводимых пользователем значений и не допускать данных которые не соответствуют формату поля или могут привести к ошибкам в работе системы

### Backend

- приложение должно иметь REST API, предоставляющие данные для frontend. Для REST endpoints контроллеров используйте разные методы (GET, PUT, PATCH, DELETE...) Добавьте разные статусы ответа, а также генерацию и обработку исключений на основе @ControllerAdvice.
- архитектура приложения должна соответствовать шаблону Layered architecture. При необходимости использовать шаблоны проектирования (Factory Method, Command, Builder, State, Observer и т.п.)
- сборка приложения должна производиться с использованием maven и может быть модульной, например, содержать модуль с логикой и web модуль.
- задокументируйте ваш REST с использованием OpenAPI 3. Выполните конфигурацию. Добавьте аннотации, описание, схемы.
- используйте технику создания и использования Bean. IoC, DI CDI.
- соблюдайте многоуровневую архитектуру: Controller, Service, Repository
- REST API должно быть защищено от неавторизованного доступа с помощью Spring Security / OAuth2.
- для авторизации используйте JWT ( JSON Web Token )

- информация должна храниться в базе данных, используйте СУБД PostgreSQL, MySQL или любую другую (Структуру и содержимое базы данных прикрепить к проекту в виде скрипта).
- для доступа к базе данных необходимо использовать Spring Data / JPA. Используйте как минимум два типа связей между сущностями @OneToMany, @ManyToOne, @ManyToMany.
- необходимо логировать основные действия администратора (удаление, добавление и т.д.), а также все ошибки, возникающие в системе с использованием log4j или logback. Используйте для этого Spring AOP (AspectJ). Конфигурацию аспектов выполните и через xml, и через аннотации. Создайте @Pointcut. Создайте советы around, before, after, after-throwing с конфигурированием для конкретных точек соединения
- используйте Java Bean Validation API или Spring Validation и объявите правила проверки полей. Напишите пользовательский валидатор (аннотацию) используя интерфейс Validator.
- для тестирования end points используйте Postman. В Postman для одного из контроллеров создайте коллекцию запросов. (по желанию - для проверки API проект можно поднимать в Docker контейнере)
- в проекте должны использоваться пользовательские типы исключений.
- в проекте используя JUNIT и JMock создайте 4 модульных и 2 интеграционных теста.

## Frontend

- Frontend часть представляет собой набор статических страничек возвращаемых сервером, каждая страничка реализует свою логику.
- Можно использовать чистый JS стандарта ES5, HTML5 и CSS3. При желании допускается использование сторонних Фреймворков.
- Web-приложение должно корректно работать в последних версиях всех основных браузеров).

## Вопросы для проверки

1. Spring как семейство проектов. SpringFramework - состав и назначение.
2. Жизненный цикл запроса в MVC Spring. Диспетчеризация. Настройка контекста
3. Spring MVC архитектура. Front Controller. Создание контроллера.
4. Конфигурация Spring. WebMvcConfigurer. Аннотации: @Controller, @Repository, @Service.
5. Адресация в Контроллере. @RequestMapping, @GetMapping и др.
6. Понятие Inversion of Control-контейнер (IoC) и Dependency Injection (DI)

7. JavaBean . Правила описания и использование JavaBean. Области действия управляемых бинов, аргументы, свойства. @Autowired, @Primary, @Qualifier, @Inject.
8. Жизненный цикл Bean Spring. @ComponentScan.
9. Spring Expression Language (SpEL): особенности и область использования.
10. Spring FrameworkValidation. Интерфейс Validator.
11. Правила валидации и ограничения.
12. Создание пользовательского валидатора.
13. Понятие ORM. Архитектура JPA: EntityManager, Persistence, ...
14. Требования в Entity. Жизненный цикл Entity. Типы связей.
15. Spring Data Annotations.
16. JPA механизм обратных вызовов (@Pre... @Post...). Запросы
17. Паттерн Service, Repository, Controller.
18. Аспектно-ориентированное программирование. Понятие аспекта, совета, срез и точки соединения, вплетение.
19. Архитектура АОП в Spring. ProxyFactory. AOP frameworks
20. Конфигурации Spring AOP. Пример определения аспекта. Аннотации и правила настройки @Pointcut @Before @AfterReturning @Around и др.
21. Понятие SPA и MPA приложений.
22. Entity – DTO конвертация. Модель Mapper (конфигурация, правила)
23. Понятие REST. Требования к RESP архитектуре.
24. HTTP-методы REST.
25. REST контроллер. Отображения запросов. Параметры запроса и ответа.
26. Отображение кодов ответа HTTP. Сопоставленные и не сопоставленные запросы.
27. Настраиваемые исключения при ошибках запроса REST. Форматы данных.
28. Тестирование REST. POSTMAN.
29. Понятие HATEOAS REST сервиса
30. Документирование REST на основе Open API. Аннотации.
31. Spring Security Framework. Request Security. Servlet filters. Security setting.
32. Authentication and authorization.
33. Interception of requests.
34. Security support in Spring Security at the method level.
35. Configure Spring Security for OAuth 2.0 Login and Resource Server
36. Spring Cloud