# **Требования по технологиям**

BE: Spring Boot, Spring Data Jpa (in memory db, auto fill), Spring Security, Spring Mvc, Gradle/ Maven, jUnit

FE: Angular/React/Vue.js (typescript), Bootstrap/Materialize - Опционально, возможно использовать любые удобные технологии)

# **Задание**

Создать веб приложение “Конференция”.

Пользователь должен иметь возможность регистрироваться на конференции и просматривать доклады. Докладчики могут создавать доклады и выбирать время в аудитории для доклада. Необходимо предусмотреть, чтобы доклады не перекрывали друг друга по времени в одной аудитории, т.е. не может быть 2 доклада в аудитории в одно время. Расписание также должно быть доступно по REST API.

Приложение должно содержать минимальный набор таблиц. Набор полей можно использовать по своему усмотрению. Таблицы:

* User - содержит данные о пользователях, пароль (зашифрованный), роль и тд.
* Talk - содержит данные о докладе. Один пользователь может быть спикером у более 1 доклада, и у одного доклада может быть много спикеров.
* Schedule - расписание, в какой аудитории, когда и какой доклад проходит
* Room - аудитория, где проходит доклад

**Роли:**

Admin, Speaker, Listener

**Страницы:**

* главная - содержит расписание докладов, разбитое по аудиториям.
* регистрация - только слушатели могут регистрироваться. Делать подтверждение регистрации не нужно.
* список докладов - доступна докладчику. отображает список докладов для докладчика с возможностями CRUD. Также на странице необходимо реализовать выбор времени доклада в расписании.
* список пользователей - доступна только администратору с возможностями CRUD. Только администратор может делать из пользователей докладчиков.

Страницу CRUD для ROOM можно не создавать, достаточно заполнить её при старте приложения

**Дополнительные требования:**

Данные заполнять через скрипт при старте приложения.

Для приложения использовать embedded tomcat.

Создать REST API которое в ответе будет содержать расписание, разбитое по аудиториям.

Покрыть тестами сервисы и контролеры - не обязательно покрывать все возможные кейсы, достаточно пару (2-3) теста для примера.

**Не обязательно:**

* После регистрации предусмотреть асинхронную отправку письма, подтверждения. Можно использовать любой ящик.
* Поддержать сборку в docker и деплой используя docker-compose
* Осуществлять сборку FE в отдельный docker контейнер с сервером nginx или npm
* FE часть. В случае отсутствия FE, необходимо предоставить механизм тестирования API, покрывающий все процессы, например выгрузку коллекции запросов из Postman <https://learning.postman.com/docs/getting-started/importing-and-exporting-data/#exporting-postman-data> или написанные тесты средствами Postman <https://learning.postman.com/docs/writing-scripts/test-scripts/>

**Время выполнения работы:** 8 дней.

**Подсказка:** выполнять задание проще с использованием Spring Boot и авто конфигурации. Можно начать с <http://start.spring.io/>.

**PS**

Это тестовое задание и не нужно доводить всё до полной готовности, достаточно создать рабочее приложение. Рабочее решение включает Оценивается умение использовать готовые решения.