# Рекомендуемые технологии и средства

* Spring boot– разворачивание и запуск приложения;
* Apache maven – сборка проекта;
* Mybatis – работа с БД;
* Slf4j – логгирование;
* Spring-webmvc – rest-сервисы, рендеринг страниц;
* Любой javascript-фреймворк для реализации клиентской логики;
* Любой css--фреймворк для стилевого оформления или самописные стили.

# Постановка задачи

Требуется написать веб-приложение для, например, библио-, аудио-, видеотеки и т.д. (предметную область выбрать самостоятельно), в котором будут реализованы CRUD-операции по работе с данными.

* Добавление записи
* Редактирование записи
* Удаление записи
* Просмотр записи
* Просмотр перечня записей с пейджингом
* Атрибутивный поиск записей

Для каждой из операций должен быть реализован rest-сервис. Соответственно, со стороны клиента rest-сервисы должны вызываться с помощью ajax-запросов. Опционально можно сделать служебную страницу, доступную только администратору, на которой будет отображаться перечень доступных rest-сервисов с кратким описанием.

В приложении должна быть реализована простейшая аутентификация и авторизация пользователей. В приложении должны быть 2 роли: администратор и пользователь.

Администратору доступны следующие действия

* Добавление записи
* Редактирование записи
* Удаление записи
* Просмотр записи
* Просмотр перечня записей с пейджингом
* Атрибутивный поиск записей

Пользователю доступны следующие действия

* Просмотр записи
* Просмотр перечня записей с пейджингом
* Атрибутивный поиск записей

Информация о пользователях должна храниться в БД. Для описания информации о пользователе достаточно 3 атрибутов:

* Логин
* Пароль
* Флаг «является администратором»

Для управления пользователями можно не делать служебную страницу, а создавать/изменять их на уровне БД вручную.

Готовое приложение должно представлять собой jar-файл с main-классом, зависимости должны лежать в папке lib рядом с jar-файлом. Пример структуры готового приложения:

|  |
| --- |
| /application  |--application.jar  |--/lib  |--dependency\_1.jar  |--dependency\_2.jar  |--dependency\_N.jar |

Приложение должно запускаться с помощью команды:

|  |
| --- |
| java –jar .\application.jar |

# Основные сценарии

После успешного входа в систему и пользователь и администратор попадают на страницу с перечнем записей. Перечень записей представляет собой таблицу с пейджингом. Перечень доступных действий формируется в зависимости от роли пользователя – администратор или пользователь.

Также на странице присутствует кнопка «выход», после нажатия на ее пользователь выходит из системы и происходит редирект на страницу входа в систему.

## Вход в систему

Актор: пользователь, администратор.

Пользователь вводит урл, происходит проверка, является ли этот пользователь авторизованным.

Если пользователь не авторизован, происходит редирект на страницу логина.

Если пользователь авторизован, он попадает на страницу с перечнем данных и доступными для этого пользователя действиями.

У администратора действий больше, у пользователя – меньше.

## Добавление записи

Актор: администратор

Пользователь нажимает кнопку «добавить запись», открывается карточка с атрибутами записи.

Пользователь заполняет карточку, нажимает «сохранить». Запись добавляется в БД, таблица динамически обновляется.

## Редактирование записи

Актор: администратор

Пользователь нажимает кнопку «редактировать» напротив выбранной в таблице записи, открывается карточка записи с заполненными атрибутами. Пользователь изменяет атрибуты, нажимает «сохранить». Запись в БД обновляется, таблица обновляется.

## Удаление записи

Актор: администратор

Пользователь нажимает кнопку «удалить» напротив выбранной в таблице записи, открывается всплывающее окно с предупреждением «Вы действительно хотите удалить запись %имя записи%?». Пользователь нажимает «Да» - запись удаляется, таблица обновляется. «Нет» - ничего не происходит

## Просмотр записи

Актор: пользователь, администратор

Пользователь нажимает на интересующую его запись, открывается карточка просмотра записи. Пользователь должен не просто кликнуть по строке в таблице, а у каждой записи должен быть атрибут выделенный в виде ссылки, по которой будет открывать карточка просмотра.

## Атрибутивный поиск

Актор: пользователь, администратор

Пользователь нажимает на кнопку «поиск», открывается карточка атрибутивного поиска, аналогичная карточке создания. Пользователь заполняет интересующие его атрибуты, нажимает кнопку «Поиск». Содержимое таблицы фильтруется согласно критериям поиска.

Поиск должен осуществляться на стороне сервера, поиск должен осуществляться учетом пейджинга. В поисковом запросе атрибуты соединяются между собой через И. Например: атрибут1=значение1 И атрибут2=значени2 и тд.

# Дополнительные требования

Пароль пользователя не должен храниться в открытом виде.

Различные настройки приложения должны быть вынесены в отдельный property-файл. С перечитыванием настроек «на лету» без перезапуска сервера можно не заморачиваться.

Все операции по работе данными должны быть реализованы с помощью ajax-запросов.

Приложение должно логгировать все основные операции и события. Максимальный размер лог-файла 10Мб, ротация логов суточная или по достижении максимального размера. Сохраняются последние 5 файлов логов.

В клиентском коде не должны использоваться inline-стили, использование inline-скриптов должно быть сведено к минимуму. То есть стили отдельно, верстка отдельно, скрипты отдельно.