**Арм для ветклиники.**

Стек технологий:

Java 8

Spring 4

Hibernate 5

MongoDB 3.2

PosrgreSql 9.5

Elasticsearch 2.3

AngularJS 2

Tomcat 8

Кратко:

Менеджмент пациентов, фиксация процесса оказания услуг, предоставление данных процесса клиентам.

Приложение должно обладать следующими ключевыми возможностями:

- добавление и редактирование животных

- добавление и редактирование записей обращений

- создание учетных записей персонала и клиентов

- личный кабинет клиента и персонала

- обмен сообщениями между персоналом и клиентами

Дизайн: над версткой не заморачиваться

**Фунционал постранично.**

**ГЛАВНАЯ.**

Урл - /

Логотип, краткое описание, пару фото, карта с маркером на какой-либо локации. Должна быть ссылка на страницу входа в личный кабинет.

**ЛОГИН**

урл - /login

страница входа в систему.

после авторизации клиенты идут на страницу для клиентов, сотрудники на свою.

**ЛИЧНЫЙ КАБИНЕТ КЛИЕНТ. (ЛК КЛИЕНТ)**

урл - /client

Отображаем последнее обращение клиента.

Ссылка на страницу животных клиента (стр. ВСЕ ЖИВОТНЫЕ).

**ВСЕ ЖИВОТНЫЕ К.**

урл - /client/pet

Список всех животных. Фото, краткое описание, кличка, кол-во обращений и статус последнего.

Ссылка на страницу ЖИВОТНОЕ К.

Сделать на ангуларе

**ЖИВОТНОЕ К.**

урл - /client/pet/{id}

История обращений по животному.

Фото, полное описание.

Отображать все обращения в обратном хронологическом порядке, т.е. последнее сверху. + ссылка на конкретное обращение.

**Личный кабинет сотрудник(ЛК СОТРУДНИК).**

урл - /employee

Имеет все те же страницы что и ЛК КЛИЕНТ с несколькими изменениями.

На главной лк отображаются последние 3 обращения, в которые Сотрудник делал изменения.

**ВСЕ ЖИВОТНЫЕ С**

урл - /employee/pet

Страница ВСЕ ЖИВОТНЫЕ - схожая с К, отображаются вообще все животные с пагинацией. Должно иметься поле ввода для поиска по животным и клиентам, при вводе данных должна появляться поисковая подсказка при выборе которой совершается переход на страницу ЖИВОТНОЕ С.

Так же должна быть ссылка на страницу ДОБАВЛЕНИЕ ЖИВОТНОГО.

Сделать на ангуларе

**ЖИВОТНОЕ С.**

урл - /employee/pet/{id}

Та же страница что и ЖИВОТНОЕ К, но есть кнопки редактирования и добавления обращений + редактирование самого животного.

**РЕДАКТИРОВАНИЕ ЖИВОТНОГО.**

урл - /employee/pet/{id}/edit

Редактирование существующего.

Страница должна иметь все необходимые поля для редактирования сущности животного.

Поле владельца после создания изменить нельзя.

**ДОБАВЛЕНИЕ ЖИВОТНОГО.**

урл - /employee/pet/new

Добавление нового.

Те же поля что и у редактирования. Поле владельца можно сделать поиском как на странице ВСЕ ЖИВОТНЫЕ С.

Поле владельца после создания изменить нельзя.

**ОБРАЩЕНИЕ РЕДАКТИРОВАНИЕ**

урл - /employee/pet/{id}/issue/{id}

Должны быть все поля необходимые для редактирования сущности обращения.

* не редактируемые поля с именем клиента и животного.

**ОБРАЩЕНИЕ ДОБАВЛЕНИЕ**

урл - /employee/pet/{id}/issue/new

Должны быть все поля необходимые для редактирования сущности обращения.

* не редактируемые поля с именем клиента и животного для которго создается обращение.

**АДМИНКА**

Урл - /admin

добавление и редактирование пользователей, как клиентов так и сотрудников.

Дополнения.

Все урл для примера, можно свои пути.

По сущностям - полей по минимуму.

Например пользователь - дата регистрации, фио, логин, мд5 пароля.

Питомец - фото, имя, описание, дата регистрации, пользователь

Обращение - дата, сотрудник, питомец, текст и статус.

Статус - это enum: new, in\_progress, done, rejected.

**Технические детали.**

Поскольку это приложение придумывалось с целью ознакомиться с определенным стеком технологий, здесь опишу что и где нужно применить.

**Java**

Приложение необходимо реализовать на Java 8. Можно использовать все её фичи.

**Spring**

Собственно все приложение должно быть реализовано на фреймворке Spring. Можно использовать Spring boot, но на выходе нужно получить war фаил который можно залить в Tomcat 8. Для конфигурации стоит отдать предпочтение JavaConfig, нежели XML, по той причине что JavaConfig проверяется компилятором и его проще рефакторить.

**Spring Security**

Собственно за безопасность отвечать будет Spring Security. Нужно сделать 3 группы прав доступа + не авторизованный:

* не авторизованый может смотреть только главную;
* СLIENT - просмотр своих животных и обращений;
* EMPLOYEE - имеет возможность просматривать все обращения и животных, а так же редактировать их;
* ADMIN - имеет все привелегии, включая создание пользователей.

**Hibernate**

Вся работа с Postgre должна идти через него.

**MongoDB**

В нем хранить личные сообщения и картинки.

Через Spring Data, смотреть MongoRepository.

С картинками и прочими файлами - GridFS.

**Elasticsearch**

Этот компонент отвечает за полнотекстовый поиск. У нас он будет использоваться для поисковых подсказок на страницах **ВСЕ ЖИВОТНЫЕ С** и **ДОБАВЛЕНИЕ ЖИВОТНОГО**.

**Автоматические тесты**

JUnit, Mockito и Spring Test - для тестирование контроллеров и сервисов и всего что не работает с БД

DBUnit - для тестирования работы c БД

Можно использовать embedded elasticsearch и mongodb для соответствующих целей.