Тестовое задание Java Web разработчик

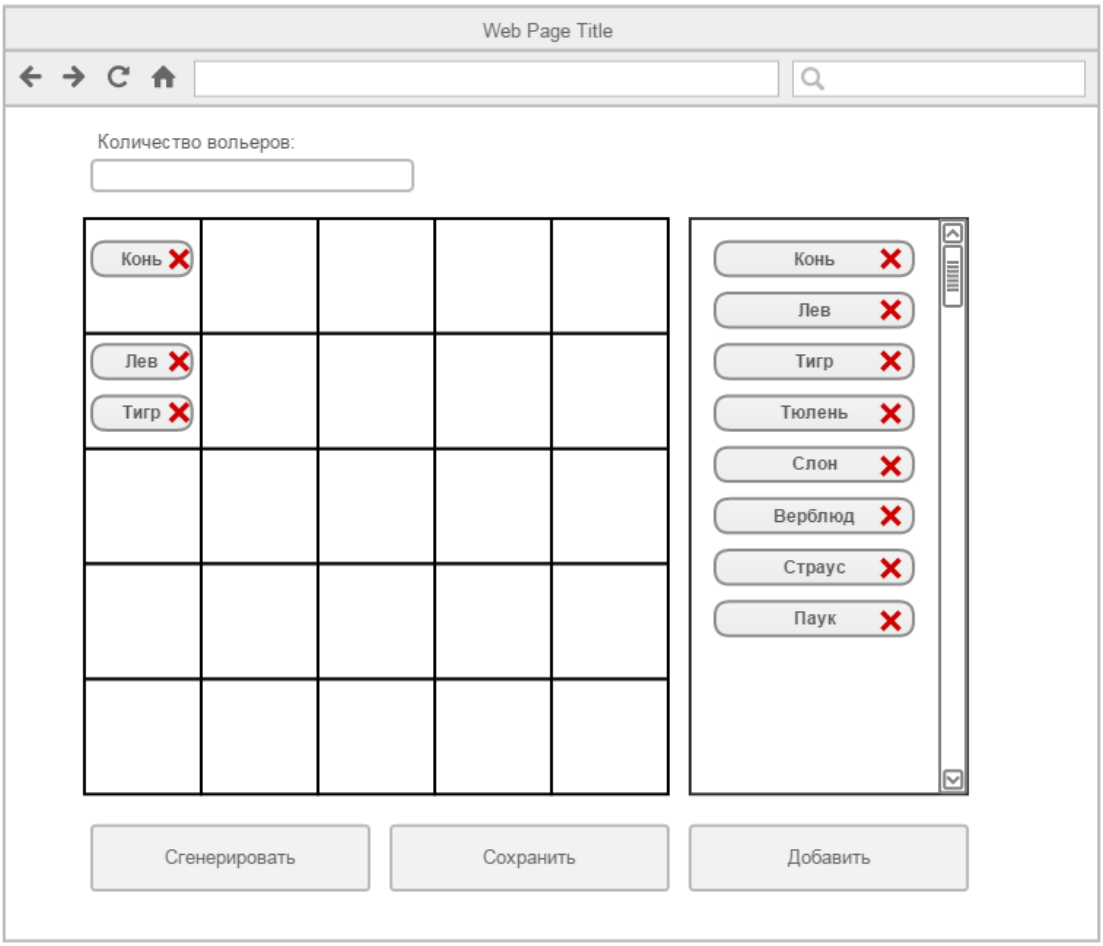
Постановка задачи

В городе N открывается новый зоопарк с довольно урезанным бюджетом.
Директор хочет понять, сколько ему нужно вольеров и каких животных ему можно поселить вместе, чтобы сэкономить - для этого он хочет небольшую программку,
в которой он сможет "потыкать", "пожамкать", "покрутить" и если что "бахнуть" животных в вольерах и сами вольеры.

Проектирование задачи

Интерфейс

Для данной задачи был спроектирован следующий интерфейс (дизайн совсем не важен):



User experience

1) Можно создавать животное с именем, которого еще нет в списке и признаком хищник/нет

2) Можно редактировать количество вольеров - при этом если в них уже были помещены животные, они остаются(если возможно)

3) При удалении животного из списка, животное удаляется из вольера

4) При удалении животного из вольера - оно остается в списке

5) Можно сохранять результат распределения животных (при этом просмотра сохраненного нет и запоминается только взаимное расположение животных в клетке, т.е. лев с тигром и т.п.)

6) Признак хищник/нет отображается цветом

7) При добавлении животного оно сохраняется и запоминается в системе

8) Нельзя поместить хищное животное в один вольер с не хищным

9) Нельзя поместить более двух животных в один вольер

10) Для добавления животного в вольер можно просто его перетянуть мышкой в нужный вольер (Drag`n`Drop)

11) Можно сгенерировать распределение животных

12) Каждого животного в зоопарке по одному

Генерация распределения животных

Распределение животных генерируется исходя из предыдущих сохраненных распределений.

Для обеспечения лучшего распределения был придуман следующий алгоритм:

Сначала для всех сохраненных связей между животными просчитывается количество (лев с тигром - 5 раз, панда с конем - 3 раза и т.д).

Далее список сортируется по убыванию количества связей и для каждого животного проставляется лучший напарник.

Если возникает спорная ситуация, то для каждого претендента происходит поиск лучшего и выбирается тот, у которого количество выбранных сочетаний меньше.

Пример (для льва):

Получился список:

Лев	4	Тигр
Медведь	4	Лев
Медведь	4	Волк
Гепард	3	Тигр

льва помещали с тигром и с медведем 4 раза.

тигра чаще всего помещали с гепардом (3 раза, при чем тут нужно понимать, что "чаще всего" - это за исключением льва)

медведя чаще всего помещали с волком (4 раза)

нужно поместить льва с тигром.

Таким образом, при возникновении спорных ситуаций выбирается лучшее решение.

Технологии

Нужно реализовать данное приложение с использованием RESTful архитектуры и следующих технологий на выбор:

UI: Dojo Framework, AngularJs, JQuery

Server Side: Spring, EJB 3.1+

DB: Hibernate ORM, Jdbc + Oracle/PostrgreSQL

Build: Maven, Gradle

Дальнейшее развитие системы

Система должна быть реализована из расчета на то, что:

1) Директор захочет просматривать ранее сохраненные планы распределения животных по вольерам

2) Могут появиться дополнительные атрибуты, которые будут влиять на признак - можно ли поместить животных вместе или нет (например размер животных, условия их жизни и проч.).

Требования к реализации

Процесс генерации должен быть покрыт Unit-тестом.

К проекту должна прилагаться инструкция по сборке и по развертыванию базы.

Предпочтительнее для развертывания базы использовать liquibase - если используется простой SQL-скрипт, то должен быть bat`ник для создания базы и таблиц.