В отчёте 2015-12-29 нарисовал совершенно не годную схему. Теперь схема состоит из двух таблиц :
request_tree — дерево запроса содержит список всех узлов, у каждого узла есть поле parentid — ссылка на старший узел, обеспечивающая структуру дерева.
 node_types — типы узлов.
 Сама схема приведена в тексте ниже.
Много времени ушло на редактирование запроса. Собственно это единственное, что у меня сейчас работает. Теперь подробнее чем занимался и где сейчас остановился.
1. Плагин jstree (https://www.jstree.com/). На его основе построено редактирование дерева запроса. При реализации возникало много неясностей, на преодоление которых уходило значительное время.
2. Когда казалось, что цикл редактирование → сохранение → восстановление → …. заработал, при передаче на сохранение терялись узлы из конца списка. Я не сталкивался с ограничением объёма данных, передаваемых методом POST. Теперь знаю, что это ограничение существует. Как пишут умные люди, ограничение определяется параметром в php.ini и составлят 8mb. Структура узла jstree достаточно громоздкая и этого объёма хватает на примерно 25 узлов. Поэтому разбил список узлов на пачки по 20 штук. На стороне backEnd в index.php пачки накапливаются и по окончанию передачи собираются в единый массив и обрабатываются дальше. При передаче в обратную сторону таких проблем нет.
3. Оформление. При раскрытии/закрытии узлов дерева должна меняться высота фоновой панели. Иначе дерево выползает за его границы. Если сделать высоту заведомо большой, то при малом числе видимых узлов будет не хорошо. Решение напрашивается следующее: надо считать число видимых узлов и переводить это число в высоту панели. Этот подсчёт можно реализовать отлавливая события открытия/закрытия узла. При открытии видимыми становятся узлы, подчинённые данному. При закрытии наоборот — все подчинённые скрываются.
4. Редактирование структуры запроса надо отлаживать и совершенствовать. Но цель работы - разбор запроса.
    Запрос считается выполнимым (допустимым), если распознаны (установлены) его части : ВОПРОС, СУБЪЕКТ, ДЕЙСТВИЕ, ОБЪЕКТ. Рассмотрим каждую из этих частей.
4.1. ВОПРОС. Не обязательная часть запроса. Например в запросе: «цены на холодильники», вопроса нет вовсе, хотя этот же запрос может звучать так: «каковы цены на холодильники?». Здесь вопрос есть. Существуют вопросы, которые сразу делают запрос невыполнимым, например, «почему(!) люди не летают?», «зачем(!) нужен иностранный язык?», «что(!) будет с курсом рубля?». Т.е. вопросы зачем, почему, что можно считать не выполнимыми.
4.2. СУБЪЕКТ. Субъектом запроса всегда является пользователь. Поэтому не важно задан он явно или не задан вовсе.
4.3. ДЕЙСТВИЕ. Может быть задано явно, например, купить, заказать, ознакомиться. При отсутствии явного задания действия, можно считать, что требуется просмотр.
4.4. ОБЪЕКТ. Определяет цель запроса. Можно разделить на составляющие: цена, товар, услуги, место. Если в запросе присутствует «цена», то обязательно должна быть одна из следующих составляющих: «товар» или «услуги». «Место» если не задано, можно принять город откуда пришёл запрос (для этого должна быть регистрация пользователя). В то же время наличие в запросе определённых товаров или услуг делает его сразу не выполнимым.

Если в результате разбора запрос признан не выполнимым, то включается механизм случайного выбора ответа.
Запросы надо хранить в БД вместе с датой и оценкой (выполнимый или нет). Для пользователя это примеры выполненных запросов, для разработчика материал для дальнейшей работы.
Ссылки. http://mnudelman.xyz/elize , тексты https://github.com/mnudelman/elize.git .

Схема БД.

```
-- Создание схемы БД elize
-- ------------------------------------
-- CREATE DATABASE IF NOT EXISTS elize ;
– drop table request_tree ;

–  таблица  request_tree – Дерево запроса.

CREATE TABLE IF NOT EXISTS request_tree (
  nodeid   INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
  parentid INTEGER DEFAULT 0,                  // родительский узел
  node_name    VARCHAR(100) ,                  // имя узла
  typeid INTEGER REFERENCES node_types(typeid),    // тип узла
  node_valid INTEGER DEFAULT 1,              // допустимость узла
  node_default INTEGER DEFAULT 0,            // ветка по умолчанию
  comment VARCHAR(100) ,
  KEY index_parent (parentid)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
-- ------------------------------------


–  таблица  node_types –  типы узлов дерева.

CREATE TABLE IF NOT EXISTS node_types (
  typeid   INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
  type_name  VARCHAR (50),
  comment VARCHAR (100)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

Содержимое таблиц

Запрос: SELECT * FROM node_types - Типы узлов запроса
Результат-записей: 8

typeid	type_name	comment
1	root	корень дерева запроса
2	question	раздел запроса - ВОПРОС
3	subject	раздел запроса - СУБЪЕКТ
4	action	ДЕЙСТВИЕ запроса
5	object	раздел запроса - ОБЪЕКТ
6	sub_object	подраздел раздела ОБЪЕКТ
7	concept	понятие
8	synonym	слово, определяющее понятие

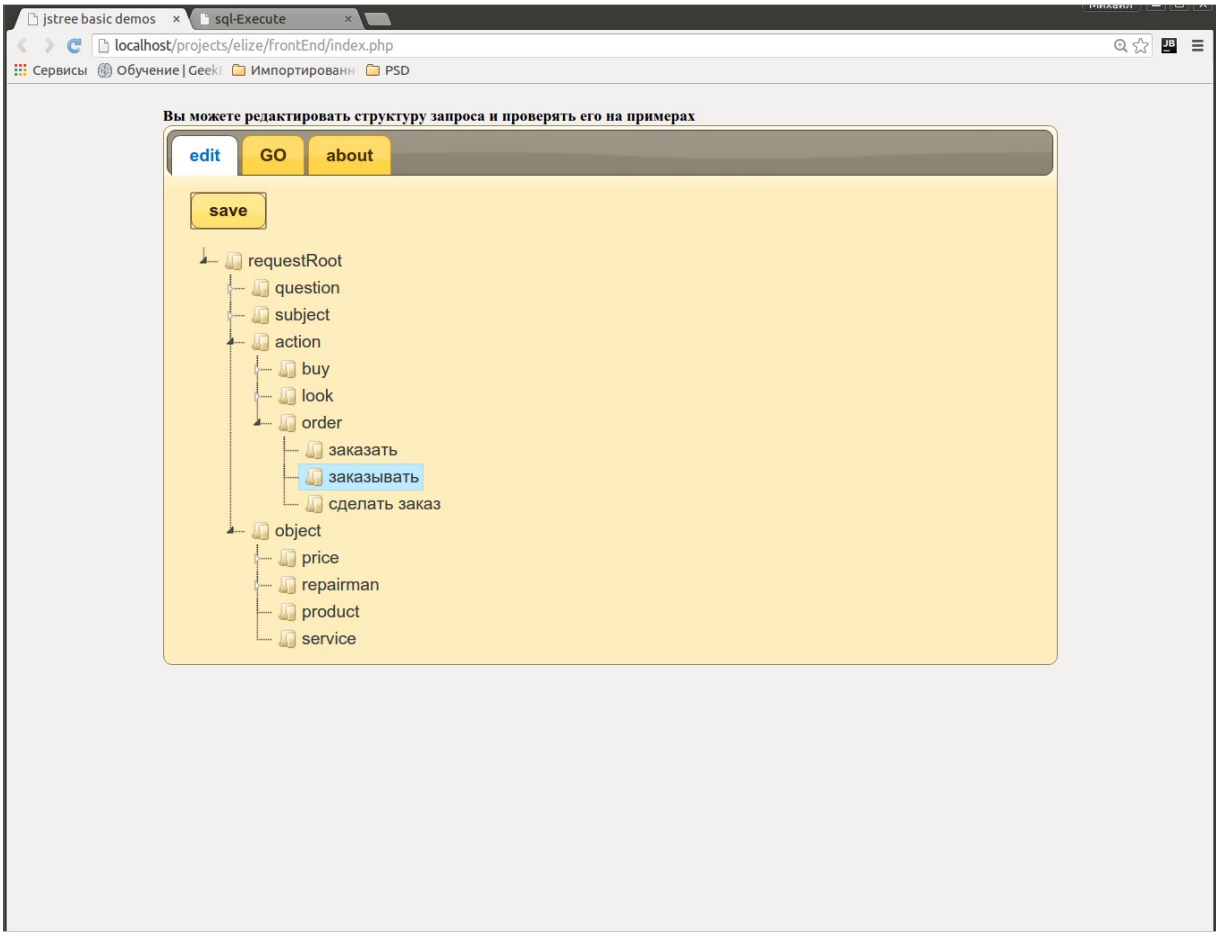
Запрос: SELECT * FROM request_tree - дерево запроса
Результат-записей: 61

nodeid	parentid	node_name	typeid	node_valid	node_default	comment
1		requestRoot	1	1		КОРЕНЬ запроса
2	1	question	2	1		ВОПРОС запроса
3	1	subject	3	1		СУБЪЕКТ запроса
4	1	action	4	1		ДЕЙСТВИЕ запроса

5	1	object	5	1	ОБЪЕКТ запроса
6	2	where	7	1	вопрос ГДЕ
7	2	what	7	1	вопрос ЧТО
8	2	why	7	1	вопрос ПОЧЕМУ
9	2	when	7	1	вопрос КОГДА
13	3	user	7	1	ПОЛЬЗОВАТЕЛЬ - субъект запроса
14	13	хочу	8	1	синоним USER
15	13	хотелось бы	8	1	синоним USER
16	13	покажите	8	1	синоним USER
17	6	где	8	1	синоним WHERE
18	6	куда	8	1	синоним WHERE
19	6	откуда	8	1	синоним WHERE
20	7	что	8	1	синоним WHAT
21	7	какой	8	1	синоним WHAT
22	7	каков	8	1	синоним WHAT
23	7	который	8	1	синоним WHAT
24	8	почему	8	1	синоним WHY
25	8	зачем	8	1	синоним WHY
26	8	как	8	1	синоним WHY
27	9	когда	8	1	синоним WHEN
28	9	укажите дату	8	1	синоним WHEN
29	4	buy	7	1	действие КУПИТЬ
30	4	look	7	1	действие ПРОСМОТРЕТЬ
31	4	order	7	1	действие ЗАКАЗАТЬ
32	29	купить	8	1	синоним BUY
33	29	приобрести	8	1	синоним BUY
34	29	получить	8	1	синоним BUY
35	30	узнать	8	1	синоним LOOK
36	30	посмотреть	8	1	синоним LOOK
37	30	покажите	8	1	синоним LOOK
38	31	заказать	8	1	синоним ORDER
39	31	заказывать	8	1	синоним ORDER
40	31	сделать заказ	8	1	синоним ORDER
41	29	купить	8	1	синоним BUY
42	29	приобрести	8	1	синоним BUY
43	29	получить	8	1	синоним BUY
44	30	узнать	8	1	синоним LOOK
45	30	посмотреть	8	1	синоним LOOK
46	30	покажите	8	1	синоним LOOK
50	5	price	7	1	ЦЕНА
51	5	repairman	7	1	МАСТЕР
52	5	product	6	1	под-объект ПРОДУКТ, ИЗДЕЛИЕ
53	5	service	6	1	под-объект УСЛУГИ
67	50	цена	8	1	синоним PRICE
68	50	цены	8	1	синоним PRICE
69	50	ценами	8	1	синоним PRICE
70	50	по цене	8	1	синоним PRICE
71	50	по какой цене	8	1	синоним PRICE
72	50	сколько стоит,	8	1	синоним PRICE
73	50	стоимость	8	1	синоним PRICE

74	51	мастер по ремонту	8	1	синоним REPAIRMAN
75	51	специалист по ремнту	8	1	синоним REPAIRMAN
76	51	элекрик	8	1	синоним REPAIRMAN
77	51	сантехник	8	1	синоним REPAIRMAN
89	52	place	6	1	под-объект МЕСТО выполнения запроса(город,..)
90	52	fridge	7	1	ХОЛОДИЛЬНИК
92	52	washer	7	1	СТИРАЛЬНАЯ МАШИНА

Фрагмент дерева запроса



Работа с деревом запроса. Левая кнопка мыши — открыть/закрыть узел. Правая кнопка открывает контекстное меню : создать узел, переименовать узел, удалить узел, перенести/копировать в другую ветку. Создать и переименовать узел открывают форму редактирования узла, где задаётся имя, тип узла, пишется комментарий. Указывается допустимость узла. После внесения изменений кнопкой «save» можно отправить текущее состояние дерева в БД.

Следующая таблица определяет возможность комбинировать узлы разных типов.

```
nodeTypes = {                          // типы узлов дерева запросов
    '#': {
        'max_children': 1,
        'max_depth': -1,
        'valid_children': ['root']
    },
    'root': {
        'max_children': -1,
        'max_depth': -1,
        'valid_children': ['question', 'subject', 'action', 'object']
    },
    'question': {                       // раздел - ВОПРОС
        'max_children': -1,
        'max_depth': 2,
        'valid_children': ['concept']
    },
    'subject': {                       // раздел - СУБЪЕКТ - 'user' only
        'max_children': 1,
        'max_depth': 2,
        'valid_children': ['concept']
    },
    'action': {                         // раздел - ДЕЙСТВИЯ
        'max_children': -1,
        'max_depth': 2,
        'valid_children': ['concept']
    },
    'object': {                         // раздел - ОБЪЕКТ ЗАПРОСА
        'max_children': -1,
        'max_depth': -1,
        'valid_children': ['sub_object', 'concept']
    },
    'sub_object': {                     // подраздел - ОБЪЕКТа ЗАПРОСА
        'max_children': -1,
        'max_depth': -1,
        'valid_children': ['sub_object', 'concept']
    },
    'concept': {                        // ПОНЯТИЕ
        'max_children': -1,
        'max_depth': 1,
        'valid_children': ['synonym']
    },
    'synonym': {                        // синонимы - слова, определяющие
понятие
        'max_children': 0,
        'max_depth': 1,
        'valid_children': []
    }
};
```

закладка GO — исполнить запрос

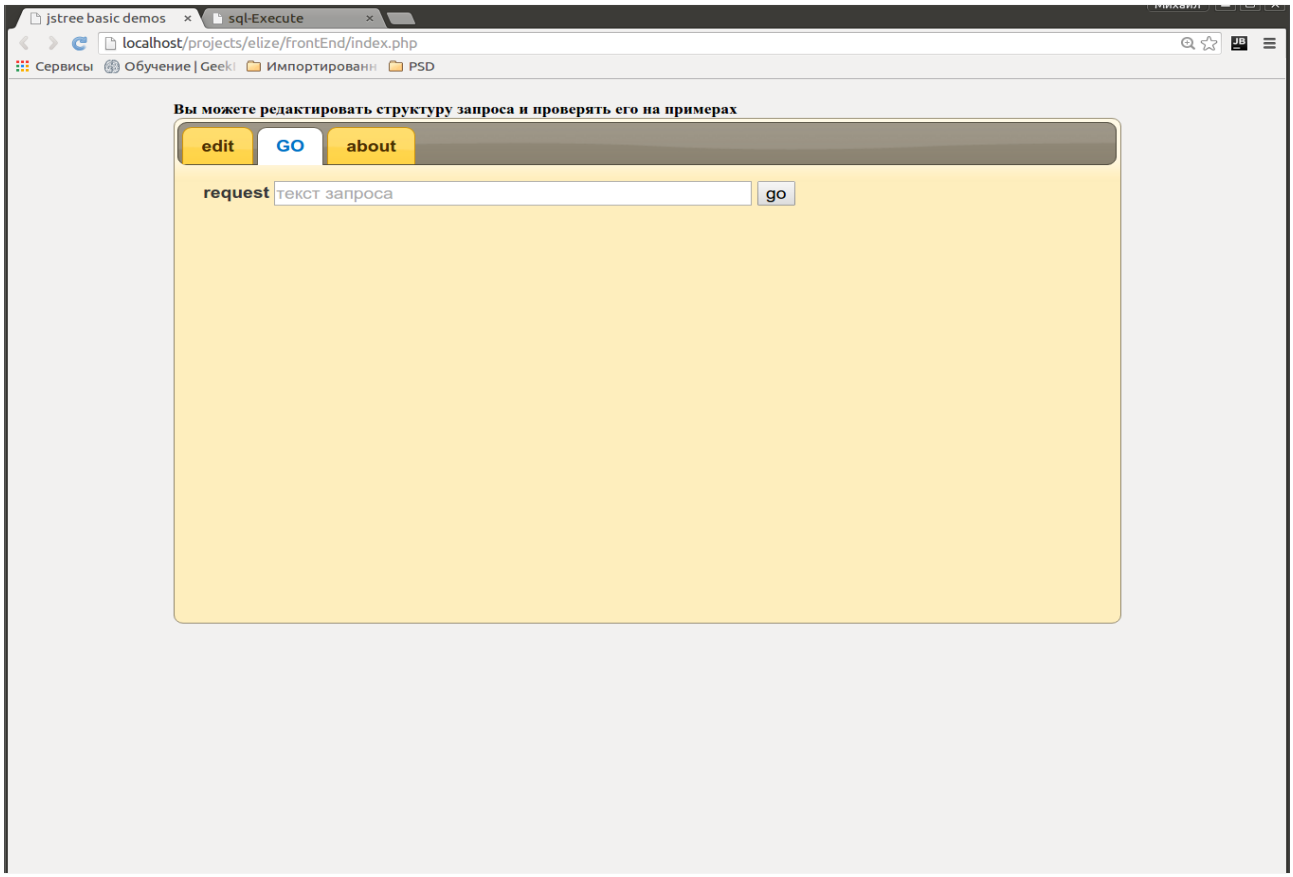
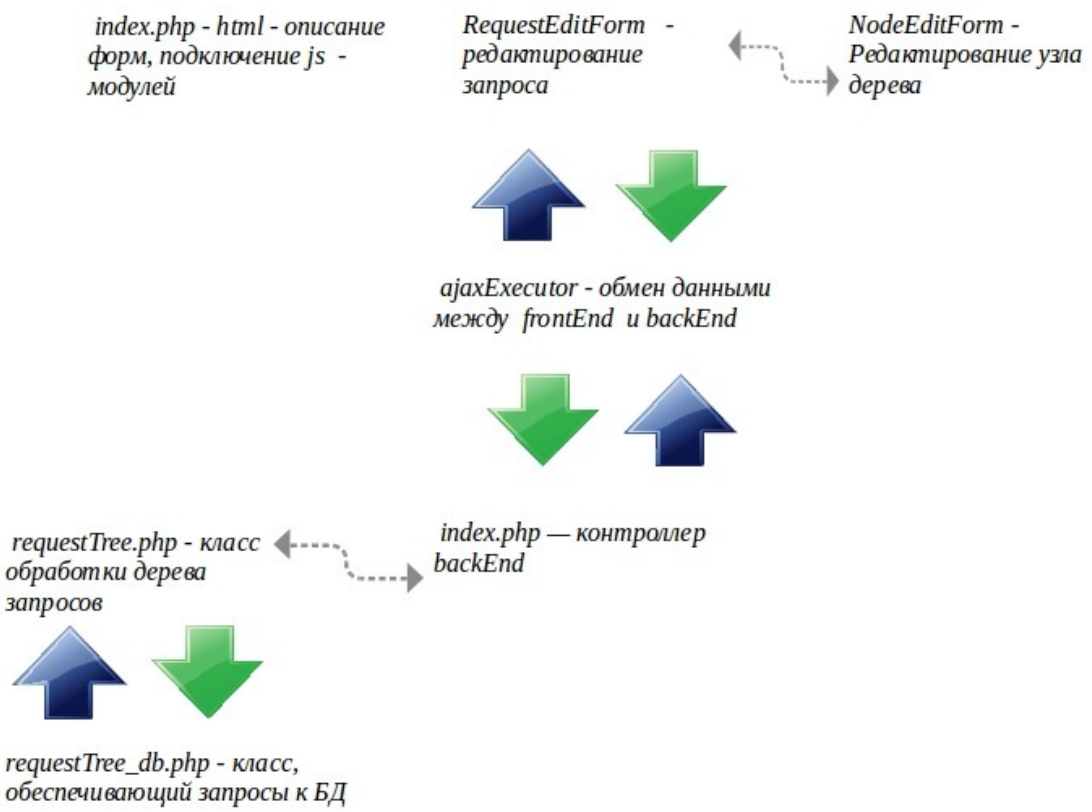


Схема работы. Получилось не очень красиво, но смысл понятен.


index.php - html - описание форм, подключение js - модулей
RequestEditForm - редактирование запроса
NodeEditForm - Редактирование узла дерева
ajaxExecutor - обмен данными между frontEnd и backEnd
index.php — контроллер backEnd
requestTree.php - класс обработки дерева запросов
requestTree_db.php - класс, обеспечивающий запросы к БД