

Проект «elize»

Цель. Выполнение запросов пользователя с использованием оригинального интерфейса.

Руководство и шаблон оформления. Игнатичев Сергей. signatichiev@yandex.ru

Программирование. Нудельман Михаил. mnudelman@yandex.ru

Сценарий.

Фоновое изображение состоит из нескольких частей :

1. Центральный круг. В области центрального круга пользователь вводит запрос.
После ввода запроса в течении 2-3 сек происходит вращение круга и изображений обрамляющих центральный круг, символизирующее размышление системы на текст запроса.
2. Планка «спросить» выполняет функцию клавиши ввода. По клику по планке или нажатию «enter» запрос начинается обрабатываться.
3. Картинки, обрамляющие центральный круг. Правая половина символизирует реальный мир: животные, деревья, деньги, цвета(природа), цифры. Левая половина нечто эзотерическое: карты обычные и таро, астрология, земные стихии, домино, лунный календарь. После ввода запроса картинки начинают вращаться вокруг вертикальной оси.
4. Клубы дыма в нижней части изображения начинают клубиться.

Выполнение запроса. После клика по планке «спросить» начинается обработка запроса. Тип запроса определяется по ключевым словам. Запрос, содержащий вопросы: как, когда, где,зачем,... или имя собственное (из списка), например, «москва»,

передаётся на выполнение в поисковые системы (yandex.xml)

Запрос, содержащий слова: «стоимость», «сколько стоит» или просто существительное в именительном падеже, например, «холодильник» передаётся на выполнение в mnogonado.net.

Остальные запросы считаются философскими.

В качестве фона для результата выполнения поисковых запросов используется «свиток».

Ответ на «философский вопрос» оформляется следующим образом:

Центральный круг меняет внешний вид. И на новом фоне выводится абстрактный ответ, взятый случайным образом из списка таких ответов.

Картинки, обрамляющие центральный круг выбираются случайным образом из соответствующих множеств.

Деревья из набора изображений деревьев, карты таро из своего множества и т. д.

При нажатии «enter» или клику по планке «далее» разворачивается свиток с информацией по каждому изображению.

После клика по любой части изображения или клавишей «esc» система возвращается к исходному состоянию, т. е. Вводу нового запроса.

Структура проекта

- **Backend** - скрипты php

Index

- контроллер

AddSignals

- передача дополнительных Сигналов

AddSignal_db

- обращение к БД для выборки допСигналов

RequestGo

- разбор запроса

RequestType

- определить тип запроса

RequestTree

- дерево запроса

RequestTree_db

- взаимодействие с БД

ConceptFunction

- расширение для определения единиц запроса

local.php

- привязка к корневой папке проекта

MorphologyRu.php

- морфологический разбор на основе окончаний слов

Thoughts.php

- Выбрать из БД список «мыслей»

Thoughts_db.php

- Обращение к БД

GeoLocation.php

- определение ближайшего города по широте и долготе

GeoLocation_db.php

- Обращение к БД

XmlYandex - реализация запросов

YandexController — исполнитель запроса к поисковой системе

Yandex — реализация запроса в */yandex.ru/search/xml*

MainProjects — реализация запроса к проектам *mnogonado.net*

Service - вспомогательные классы

Utilites - утилиты работы с БД

- **Frontend** - скрипты js

js

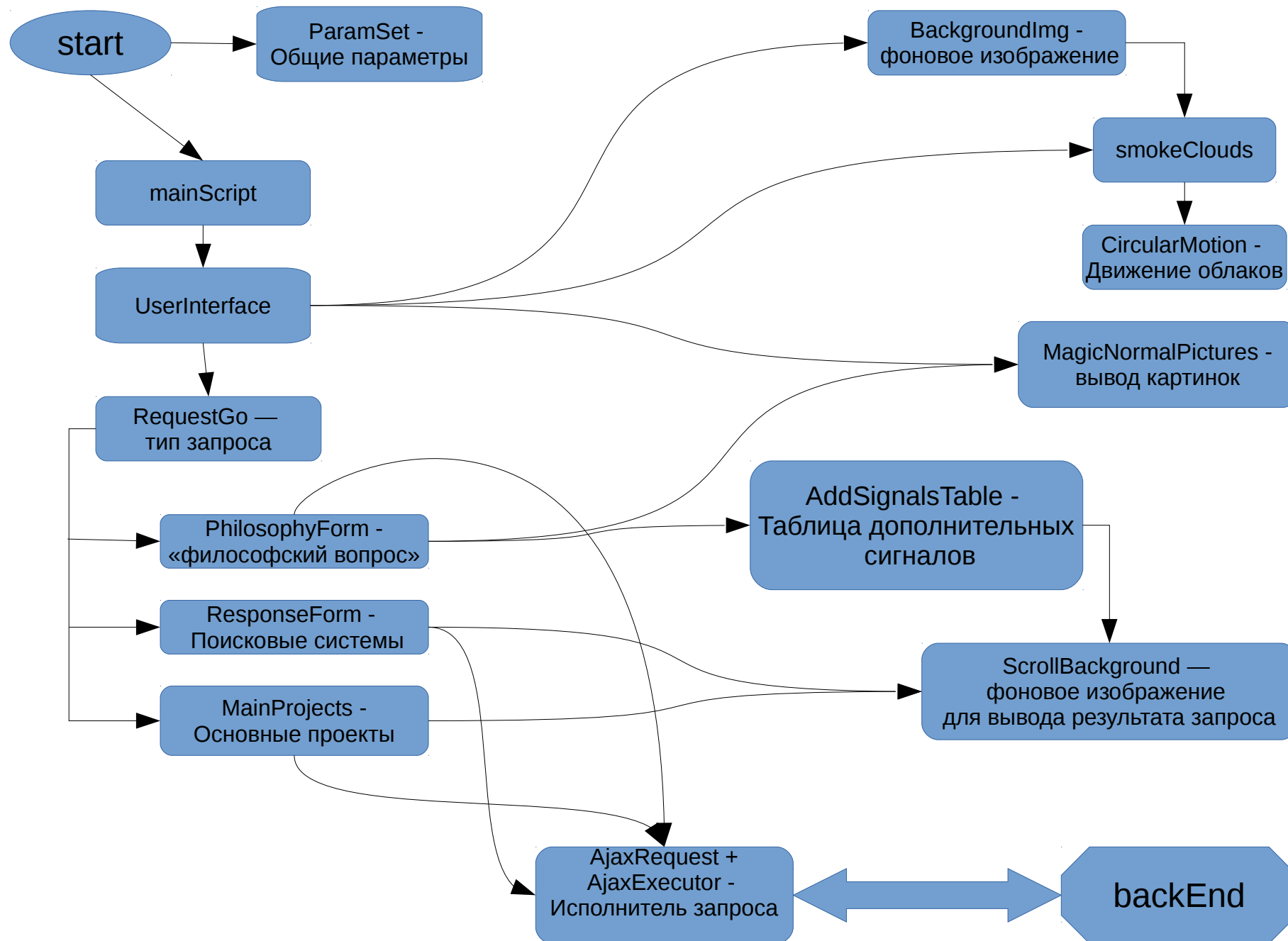
ActionSteps.js	- объект отслеживает последовательность шагов по реализации действия
AddSignalsTable.js	- контроллер таблицы дополнительных сигналов
AjaxExecutor.js	- запуск запросов к backEnd/index.php
AjaxRequest.js	- взаимодействие с ajaxExecutor
BackgroundImg.js	- элементы основного фонового изображения
CallStack.js	- обеспечивает возврат в нужную точку после завершения действия
CentralCircleText.js	- управление высотой области ввода текста запроса
CircularMotion.js	- управление движением облаков
GeoLocation.js	- определение координат пользователя по его ip-адресу
MagicNormalPictures.js	- элементы фонового изображения - картинки
MainProjectsForm.js	- запуск и вывод результата запроса к основным проектам
MainScript.js	- запуск приложения
Messages.js	- сообщения об ошибках
NodeEditForm.js	- редактирование узлов дерева запроса (только для администратора)
ParamSet.js	- общие параметры задачи.
PhilosophyForm.js	- вывод результата «философского запроса»
PhilosophyFormAttr.js	- атрибуты «философского запроса»
Placeholder.js	- имитатор placeholder
RequestForm.js	- форма редактирования запроса (только для администратора)
RequestGo.js	- определитель типа запроса
RequestGoForm.js	- форма определения типа запроса (только для администратора)
ResponseForm.js	- запуск и вывод результата запроса к поисковым системам
ScrollBackground.js	- фоновое изображение результата запроса к поисковым системам и проектам
SmokeClouds.js	- элемент изображения — клубы дыма(облака)
start.js	- точка входа в систему
UserInterface.js	- контроллер пользовательского представления

- **Images** - изображения
 - animals** - папки с тематическими картинками
 - astrology** - для представления результатов
 - cards.** - «философского запроса»
 -
 -
 - taro**
 - tree**
 - philosophy** - подвижные картинки — элементы шаблона *magic2.psd*
 - smoke** - облака — элементы шаблона *magic2.psd*
 - uselinterface** - элементы изображения основного фона и свитка — фона вывода результата
- **ARCHIV** - упакованные папки
 - db_dump** - дампы БД
 - lib** - плагины, используемые в проекте

FrontEnd — взаимодействие объектов

- Выполнение запроса

- Отображение



Технологические объекты.

- **CallStack** - *возврат в нужную точку объекта*
- **ActionSteps** — *последовательность шагов по реализации действия*
- **CallStack** - *возврат в нужную точку объекта.*

У объекта может быть точка входа(запуска), точка входа, точка повторного входа.

В точке входа выдаётся

callStack.pushItem(o_name,o_reEnter),

o_name — имя объекта

o_reEnter — функция(метод) для повторного входа в объект

В точке выхода:

callStack.pullItem() ; // убирает себя из стека

callStack.currentGo() ; // запускает o_reEnter объекта-родителя

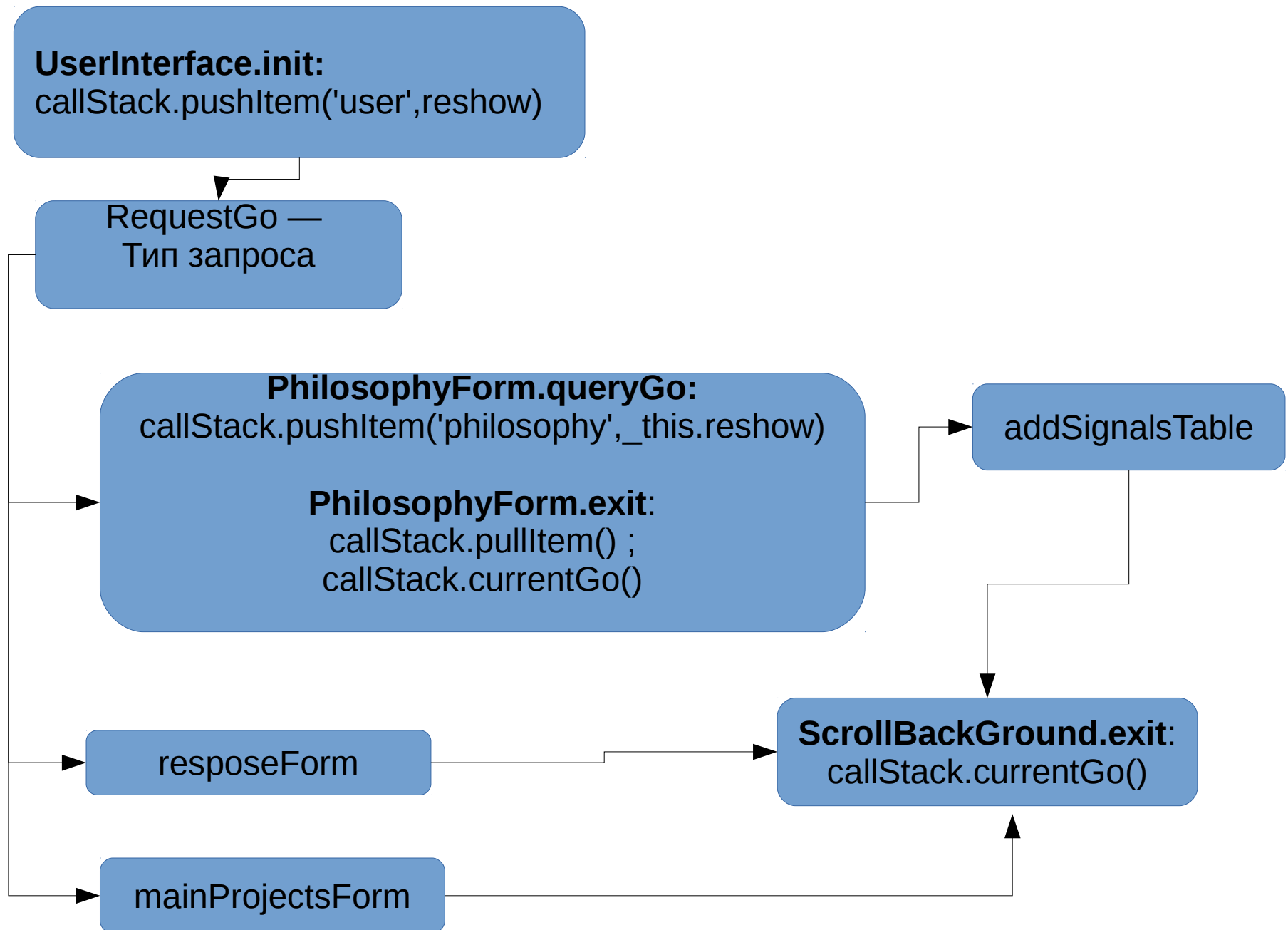
Из примера, приведённого на следующей странице, следует.

В зависимости от цепочки вызовов, после отработки scrollBackground,

управление вернётся в метод reshow() объекта userInterface или

В метод this.reshow() объекта philosophyForm.

Пример использования callStack



- **ActionSteps** — последовательность шагов по реализации действия

Объект отслеживает последовательность шагов по реализации действия
действие может остановиться в некотором состоянии. *callback* - используется
для запуска продолжения

пример. управляющийМодуль запускает выполнение запроса, требующего время на
выполнение.

по завершении стадии подготовки объект, реализующий запрос, выдаёт
`actionSteps.addStep('<имяПрограммы>', 'prepare', func)`

'prepare' - говорит, что данные для вывода готовы

func - метод, выполняющий вывод результата на экран

управляющийМодуль "ждёт" состояния 'prepare' и затем запускает *func* для
вывода результата

во время ожидания может отвлекать пользователя какими-то действиями.

UI.stampClickGo:

```
requestGo.requestExecute(auto); // выполнение запроса
Цикл { //- ожидание готовности для вывода
If (actionSteps.isConditionPrepare()) {// готовность к выводу
ActionSteps.stepGo() //- вывод
} else {
крутим центральный круг} }
```

ResponseForm.queryGo:

```
actionSteps.addStep('searchSystem','prepare',responseShow) ; -
Готовность данных для вывода
responseShow — метод вывода, к нему будет обращение
из управляющего объекта через actionSteps.stepGo()
```

Backend — взаимодействие классов

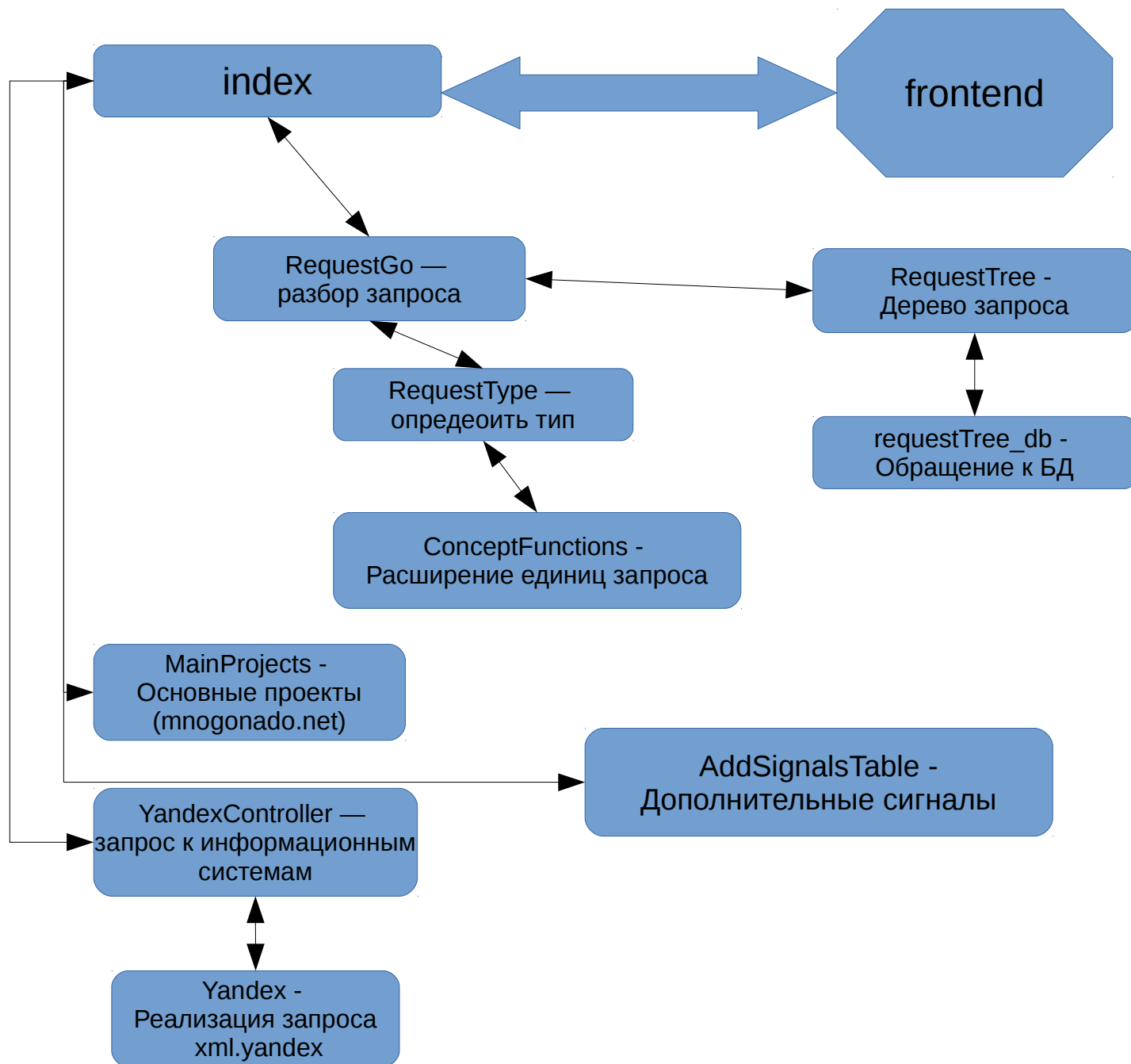


Схема базы данных

```
-- Создание схемы БД elize
```

```
-- CREATE DATABASE IF NOT EXISTS elize ;
```

```
--  
CREATE TABLE IF NOT EXISTS request_tree (  
  nodeid  INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  parentid INTEGER DEFAULT 0,  
  node_name  VARCHAR(255) ,  
  typeid INTEGER REFERENCES node_types(typeid),  
  node_valid INTEGER DEFAULT 1,  
  node_default INTEGER DEFAULT 0,      // ветка по умолчанию  
  comment VARCHAR(100) ,  
  KEY index_parent (parentid)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

– дерево запроса

– родительский узел

– тип узла

– допустимость узла

```
--  
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 ;  
-- таблица имён собственных
```

– типы узлов

```
CREATE TABLE IF NOT EXISTS ru_names (  
  nameid INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  name_text VARCHAR (100) NOT NULL DEFAULT "",  
  KEY index_name (name_text)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;  
-- таблица склонений по падежам имён собственных
```

– имена собственные

```
CREATE TABLE IF NOT EXISTS ru_name_synonyms (  
  id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  nameid INTEGER REFERENCES ru_names(nameid),  
  synonym VARCHAR (100) NOT NULL DEFAULT "",  
  KEY index_synonym (synonym)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;  
-- таблица типов дополнительных сигналов
```

– имена собственные - синонимы

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

– типы дополнительных сигналов

```
-- таблица дополнительных сигналов  
CREATE TABLE IF NOT EXISTS add_signals (  
  id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  typeid INTEGER REFERENCES add_signals_types(typeid),  
  file_name  VARCHAR (100),  
  name VARCHAR (100) UNIQUE ,  
  rang INTEGER ,  
  text TEXT  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

– дополнительные сигналы

```
-- таблица список мыслей
```

```
CREATE TABLE IF NOT EXISTS thoughts (  
  id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  text VARCHAR (255)  
)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

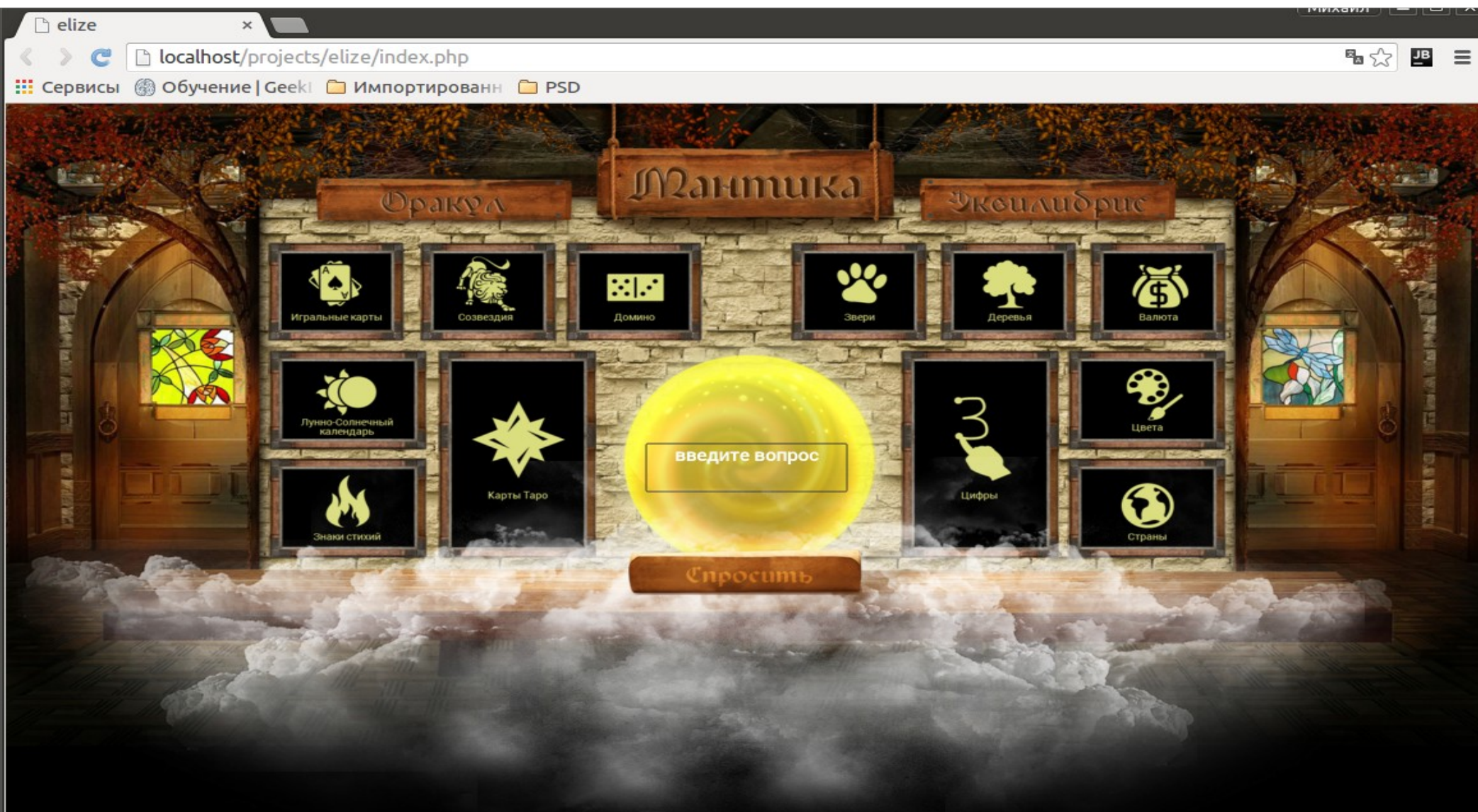
– список мыслей

```

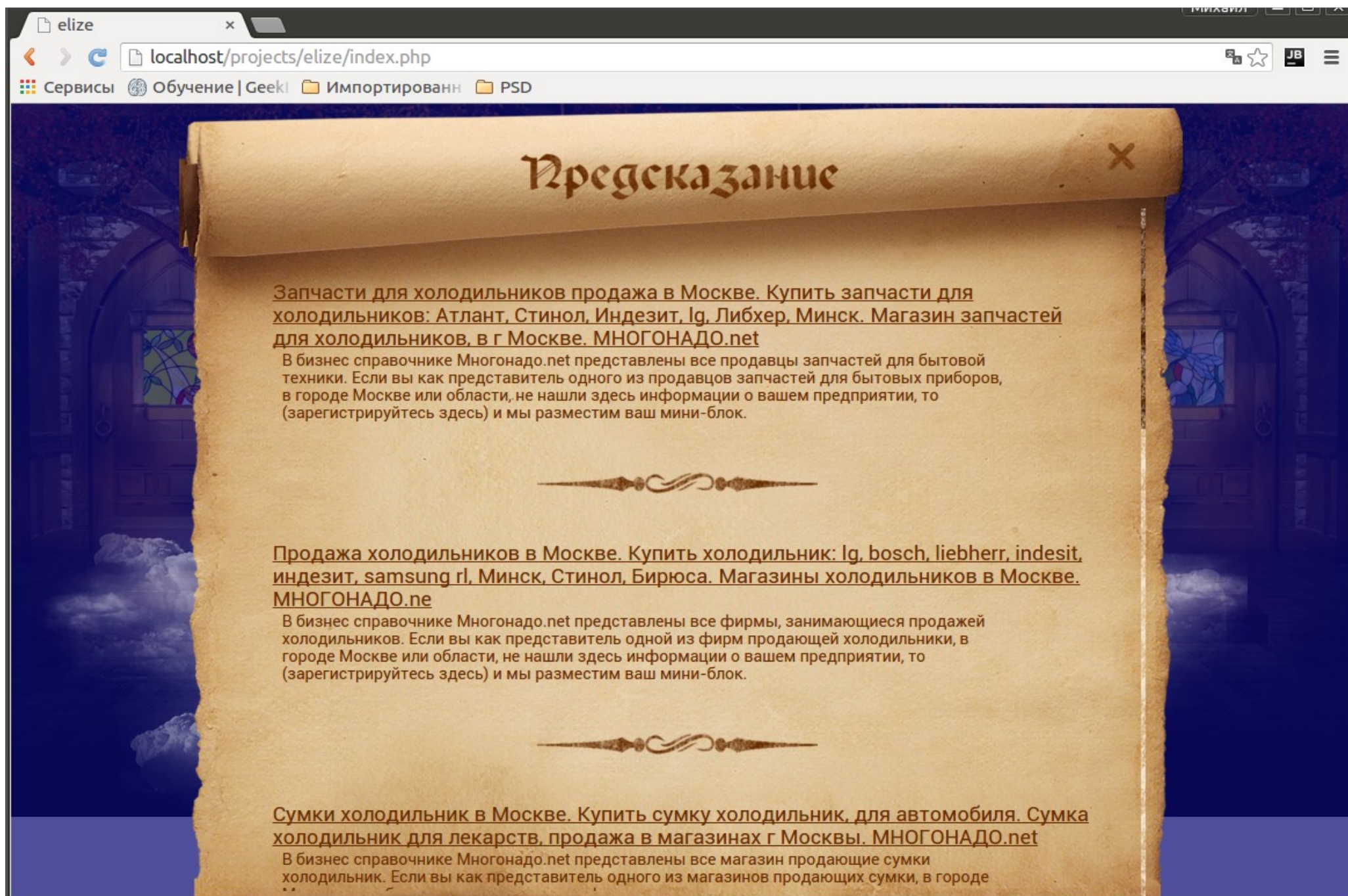
--
• -- Структура таблицы `mlt_city`
• --
• --DROP TABLE mlt_city ;
• CREATE TABLE IF NOT EXISTS `mlt_city` (                                -- список городов
•   `id` int(11) NOT NULL AUTO_INCREMENT,
•   `countryid` int(10) unsigned NOT NULL,
•   `regionid` int(11) NOT NULL,
•   `name` varchar(63) NOT NULL,
•   `size` enum('big','small','xsmall','tiny') NOT NULL,
•   `ordering` tinyint(3) NOT NULL DEFAULT '0',
•   `ordering2` int(6) NOT NULL,
•   `published` tinyint(1) NOT NULL DEFAULT '0',
•   `params` text CHARACTER SET cp1251 NOT NULL,
•   `alias` varchar(31) CHARACTER SET cp1251 NOT NULL,
•   PRIMARY KEY (`id`),
•   KEY `pub_region` (`published`,`regionid`),
•   KEY `alias` (`alias`(3)),
•   KEY `country_region_idx` (`countryid`,`regionid`)
• ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=130410 ;
• --
• -- Структура таблицы `mlt_city_II`
• --
• CREATE TABLE IF NOT EXISTS `mlt_city_II` (                                -- координаты городов (широта, долгота)
•   `cityid` int(10) unsigned NOT NULL,
•   `lat` float NOT NULL,                                                    -- широта
•   `long` float NOT NULL,                                                  -- долгота
•   PRIMARY KEY (`cityid`)
• ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
•

```

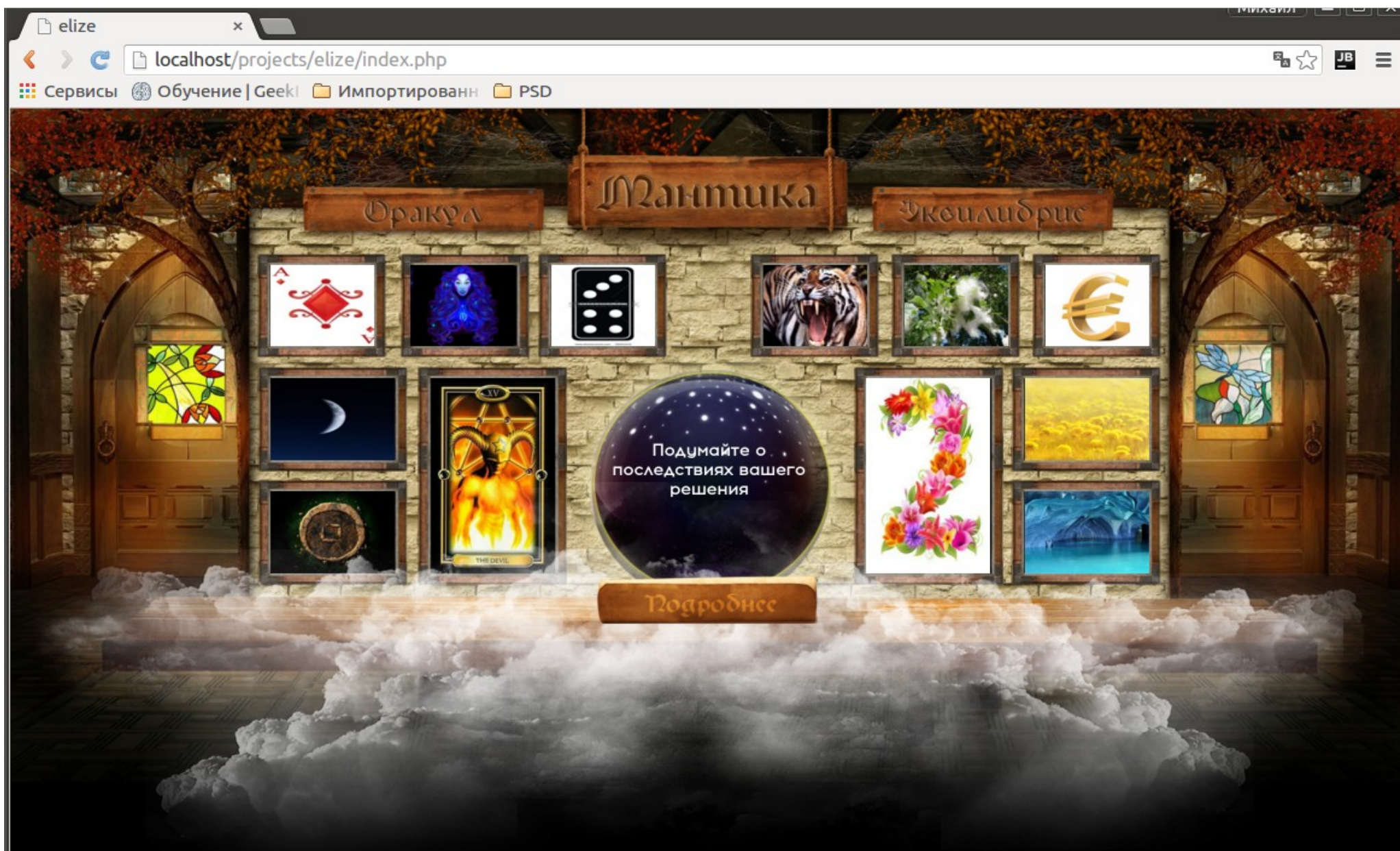
исходное фоновое изображение



Развёрнутый «свиток» - результат запроса



Результат запроса «философия»



«СВИТОК» - дополнительные сигналы

elize

localhost/projects/elize/index.php


Сервисы

Обучение | Geek!

Импортированн

PSD


Предсказание




игральные карты.
Бубновый туз

Бубновый туз говорит о том, что скоро придёт какое-то приятное сообщение, но стоит тщательно всё взвесить, чтобы определиться, необходимо ли принимать

предложение или нет. Письмо скорее всего будет любовным, и ситуация обернётся в хорошую сторону, но возможны и исключения. Лучше всё перепроверить.



За Против



Символ практичности, добродетели и помощи. Девы во всём проявляют аккуратность и за работу берутся с

Педантичны. Отлично планируют свои дела. Девы - самостоятельные, они индивидуалисты, в их мире всё всегда на

