Федеральное государственное образовательное бюджетное учреждение
Сибирский государственный университет телекоммуникаций и информатики
Кафедра прикладной математики и кибернетики

Контрольная работа
«Базы данных»

Выполнил:
студент 4 курса
группы ЗП-51
специальность ПОСВТ
Васильев Е. В.
Проверил:
доцент каф. ПмиК
Барахнин В. Б.

Новосибирск, 2018

Содержание

Задача контрольной работы

Написать последовательность команд:

- создания таблицы;
- начального заполнения таблицы данными;
- выборки из нее итоговой информации.

Написать три PHP-скрипта:

- создания таблицы в СУБД MySQL и начального заполнения таблицы данными,
- создания HTML-формы (указанного вида) для выбора имени столбца таблицы,
- вывод в браузере содержимого выбранного столбца.

Задания, согласно варианту 15:

- Вывести число строк, не содержащих данные о языках, тип которой задавать с помощью параметра.
- Вывести список языков, кроме тех , что относятся к первому или последнему (задавать с помощью параметра) по алфавиту типу.
- Вывести названия языков, имеющих те же типы, что и типы языков, выпускаемые фирмой, указанной в параметре.

HTML-форма: флажки-переключатели.

Примечания

1. Имя таблицы должно быть в точности таким, как написано в задании.
2. Отчет по КР должен отвечать всем требованиям к оформлению КР, содержать текст задания, тексты сценариев и результаты выполнения команд.
3. Таблица должна содержать данные согласно вашему варианту задания, номер варианта соответствует номеру персонального шифра (зачетной книжки).
4. Имя таблицы должно быть в точности таким, как написано в задании.
5. Изменять данные в приведенных таблицах НЕ РАЗРЕШАЕТСЯ!

Таблица 3 - Языки программирования (ЯП).

Обязательные поля: номер, название языка, тип языка, фирма-разработчик (выбрана условно).

N	Назв.	Тип	Фирма
1	Pascal	Процед	Borland
2	C	Процед	Borland
3	Java	Процед	Java inc
4	C++	Объект	Java inc
5	Visual C	Объект	Microsoft
6	Visual Basic	Объект	Microsoft
7	Delphi	Объект	Borland
8	Lisp	Сценарн	IBM
9	Prolog	Сценарн	IBM
10	XML	Сценарн	Borland

Практическая часть

Создание таблицы

```
CREATE TABLE `Языки` (
  `N` int(2) UNSIGNED NOT NULL,
  `Назв` char(12) NOT NULL,
  `Тип` enum('Процед','Объект','Сценарн','') NOT NULL,
  `Фирма` char(9) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `Языки`
  ADD PRIMARY KEY (`N`);

ALTER TABLE `Языки`
  MODIFY `N` int(2) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;
```

Результат

```
MariaDB [mydb]> CREATE TABLE `Языки` (
    ->   `N` int(2) UNSIGNED NOT NULL,
    ->   `Назв` char(12) NOT NULL,
    ->   `Тип` enum('Процед','Объект','Сценарн','') NOT NULL,
    ->   `Фирма` char(9) NOT NULL
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

Query OK, 0 rows affected (2.902 sec)

MariaDB [mydb]>
MariaDB [mydb]> ALTER TABLE `Языки`
    ->   ADD PRIMARY KEY (`N`);
Query OK, 0 rows affected (1.136 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [mydb]>
MariaDB [mydb]> ALTER TABLE `Языки`
    ->   MODIFY `N` int(2) UNSIGNED NOT NULL AUTO_INCREMENT;
Query OK, 0 rows affected (0.285 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [mydb]> COMMIT;
Query OK, 0 rows affected (0.000 sec)

MariaDB [mydb]>
```

Начальное заполнение данными

```
INSERT INTO `Языки` (`N`, `Назв`, `Тип`, `Фирма`) VALUES
(NULL, 'Pascal', 'Процед', 'Borland'),
(NULL, 'C', 'Процед', 'Borland'),
(NULL, 'Java', 'Процед', 'Java inc'),
(NULL, 'C++', 'Объект', 'Java inc'),
(NULL, 'Visual C', 'Объект', 'Microsoft'),
(NULL, 'Visual Basic', 'Объект', 'Microsoft'),
(NULL, 'Delphi', 'Объект', 'Borland'),
(NULL, 'Lisp', 'Сценарн', 'IBM'),
(NULL, 'Prolog', 'Сценарн', 'IBM'),
(NULL, 'XML', 'Сценарн', 'Borland');
```

Результат

```
MariaDB [mydb]> INSERT INTO `Языки` (`N`, `Назв`, `Тип`, `Фирм
a`) VALUES
    -> (NULL, 'Pascal', 'Процед', 'Borland'),
    -> (NULL, 'C', 'Процед', 'Borland'),
    -> (NULL, 'Java', 'Процед', 'Java inc'),
    -> (NULL, 'C++', 'Объект', 'Java inc'),
    -> (NULL, 'Visual C', 'Объект', 'Microsoft'),
    -> (NULL, 'Visual Basic', 'Объект', 'Microsoft'),
    -> (NULL, 'Delphi', 'Объект', 'Borland'),
    -> (NULL, 'Lisp', 'Сценарн', 'IBM'),
    -> (NULL, 'Prolog', 'Сценарн', 'IBM'),
    -> (NULL, 'XML', 'Сценарн', 'Borland');
Query OK, 10 rows affected (0.066 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

Получение данных из базы данных

```
SELECT * FROM `Языки` ORDER BY `N` ASC;
```

Результат

```
MariaDB [mydb]> SELECT * FROM `Языки` ORDER BY `N` ASC;
+----+--------------+-----------------+------------+
| N  | Назв         | Тип             | Фирма      |
+----+--------------+-----------------+------------+
|  1 | Pascal       | Процед          | Borland    |
|  2 | C            | Процед          | Borland    |
|  3 | Java         | Процед          | Java inc   |
|  4 | C++          | Объект          | Java inc   |
|  5 | Visual C     | Объект          | Microsoft  |
|  6 | Visual Basic | Объект          | Microsoft  |
|  7 | Delphi       | Объект          | Borland    |
|  8 | Lisp         | Сценарн         | IBM        |
|  9 | Prolog       | Сценарн         | IBM        |
| 10 | XML          | Сценарн         | Borland    |
+----+--------------+-----------------+------------+
10 rows in set (0.001 sec)
```

Получить число строк, не входящих в тип (параметр)

```
SELECT COUNT(*) AS `count` FROM `Языки` WHERE `Тип` != "Сценарн";
```

Результат

```
MariaDB [mydb]> SELECT COUNT(*) AS `count` FROM `Языки` WHERE
`Тип` != "Сценарн";
+-------+
| count |
+-------+
|     7 |
+-------+
1 row in set (0.001 sec)

MariaDB [mydb]>
```

Получить список языков, за исключением относящихся к первому по алфавиту типу

```
SELECT * FROM `Языки` WHERE `Тип` != (SELECT DISTINCT(`Тип`) FROM `Языки` ORDER BY `Тип` ASC LIMIT 1) ORDER BY `N` ASC;
```

Результат

```
MariaDB [mydb]> SELECT * FROM `Языки` WHERE `Тип` != (SELECT D
ISTINCT(`Тип`) FROM `Языки` ORDER BY `Тип` ASC LIMIT 1) ORDER
BY `N` ASC;
+----+--------------+-----------------+------------+
| N  | Назв         | Тип             | Фирма      |
+----+--------------+-----------------+------------+
|  4 | C++          | Объект          | Java inc   |
|  5 | Visual C     | Объект          | Microsoft  |
|  6 | Visual Basic | Объект          | Microsoft  |
|  7 | Delphi       | Объект          | Borland    |
|  8 | Lisp         | Сценарн         | IBM        |
|  9 | Prolog       | Сценарн         | IBM        |
| 10 | XML          | Сценарн         | Borland    |
+----+--------------+-----------------+------------+
7 rows in set (0.001 sec)

MariaDB [mydb]>
```

Получить список языков, за исключением относящихся к последнему по алфавиту типу

```
SELECT * FROM `Языки` WHERE `Тип` != (SELECT DISTINCT(`Тип`) FROM `Языки` ORDER BY `Тип` DESC LIMIT 1) ORDER BY `N` ASC;
```

Результат

```
MariaDB [mydb]> SELECT * FROM `Языки` WHERE `Тип` != (SELECT D
ISTINCT(`Тип`) FROM `Языки` ORDER BY `Тип` DESC LIMIT 1) ORDER
 BY `N` ASC;
+---+--------------+--------------+------------+
| N | Назв         | Тип          | Фирма      |
+---+--------------+--------------+------------+
| 1 | Pascal       | Процед       | Borland    |
| 2 | C            | Процед       | Borland    |
| 3 | Java         | Процед       | Java inc   |
| 4 | C++          | Объект       | Java inc   |
| 5 | Visual C     | Объект       | Microsoft  |
| 6 | Visual Basic | Объект       | Microsoft  |
| 7 | Delphi       | Объект       | Borland    |
+---+--------------+--------------+------------+
7 rows in set (0.001 sec)

MariaDB [mydb]>
```

Получить названия языков, имеющих те же типы, что и типы языков, выпускаемые фирмой, указанной в параметре.

```
SELECT `Назв` FROM `Языки` WHERE `Тип` IN (SELECT DISTINCT(`Тип`) FROM `Языки` WHERE `Фирма` = "IBM") ORDER BY `Назв` ASC;
```

Результат

```
MariaDB [mydb]> SELECT `Назв` FROM `Языки` WHERE `Тип` IN (SEL
ECT DISTINCT(`Тип`) FROM `Языки` WHERE `Фирма` = "IBM") ORDER
BY `Назв` ASC;
+----------+
| Назв     |
+----------+
| Lisp     |
| Prolog   |
| XML      |
+----------+
3 rows in set (0.001 sec)

MariaDB [mydb]>
```

Реализация приложения

Технологический стек реализации:

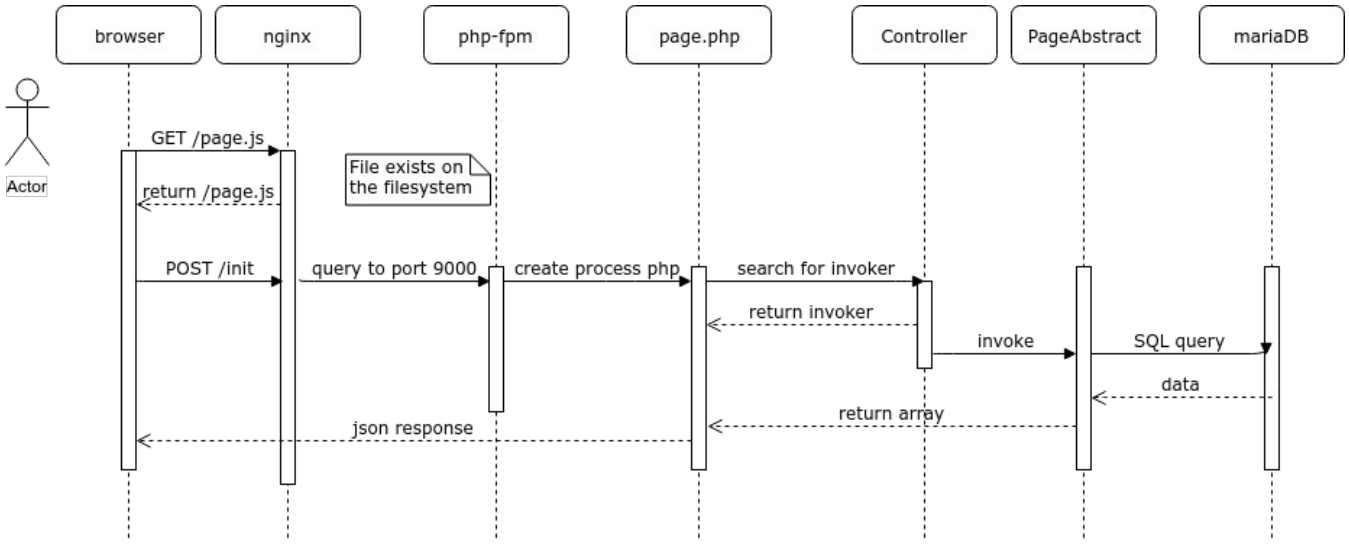
Компонент	Версия	Версия образа docker
Docker CE	18.09.2	
Docker-Compose	1.23.2	
MariaDB	10.3.12	mariadb:latest
Nginx	1.15.8	nginx:latest, расширен Dockerfile
PHP-fpm	7.3.2	php:fpm, расширен Dockerfile
PHPMyAdmin	4.8.5	phpmyadmin/phpmyadmin:latest
php-pdo	5.0.12-dev	
Semantic UI	2.4.2	
jQuery	3.1.1	
HTML	5	
ECMAScript	2015	

Данное приложение работает в связке нескольких сервисов:

1. mariadb
2. nginx
3. fpm
4. phpmyadmin

Приложение работает как одностраничное приложение (SPA – single page application). На яызке ECMAScript (JavaScript) реализована работа пользовательского интерфейса (UI).
На языке PHP реализована работа API для доступа к данным по протоколу REST API.

Диаграмма запросов



Пользовательский интерфейс

Клиент написан на HTML5 при использовании библиотек Semantic UI, jQuery.
В шапке страницы имеются три кнопки:

Кнопка	Действие
Recreate	Пересоздает таблицу
Fill table	Заполняет таблицу данными
Recreate and fill table	Пересоздает и заполняет таблицу данными

Пользовательский интерфейс реализован в виде страницы, разделенной вкладками.

Вкладка	Предназначение
All records	Отображение всех содержащихся в таблице данных
Select only one column	Отображение данных только одной колонки
Excluding type	Отображение данных, исключив определенный тип
Familiar companies	Отображение данных о других языках, типы которых использовались компанией

При переходе между вкладками, происходит переключение режимов отображения, при этом может происходить асинхронное получение требуемых для работы вкладки данных с шины API.

Вкладка	Данные	Асинхронный запрос
All records	нет	нет
Select only one column	Список столбцов	/column
Excluding type	Список типов	/type
Familiar companies	Список компаний	/company

Интерфейс API

Точка входа	Предназначение
/	Получение index.html, SPA
/init	Инициализация базы данных
/company	Получение списка компаний
/type	Получение списка типов
/column	Получение списка колонок
/language	Получение списка языков
/language/column/<column_name>	Получение столбца column_name
/language/exclude_by_type/<type_name>	Получение языков, за исключением типа type_name
/language/familiar/<company_name>	Получение языков, использующих типы компании company_name

Инструкция по запуску приложения

Зависимости проекта:

1. docker-ce (https://docs.docker.com/install/)
2. docker-compose (https://docs.docker.com/compose/install/)
3. node js (https://nodejs.org/en/download/)
4. npm (https://www.npmjs.com/get-npm)

Шаги инициализации проекта:

1. Склонируйте репозиторий проекта
   ```
   git clone https://github.com/johnynsk/sibsutis-labs.git
   ```
2. Перейдите в директорию с проектом
   ```
   cd sibsutis-labs/databases
   ```
3. Установить необходимые проекту пакеты npm:
   ```
   npm install
   ```
4. Запустить сервера mysql, nginx, php-fpm, phpmyadmin:
   ```
   docker-compose up -d
   ```
5. Перейти на http://127.0.0.1:8200

Структура проекта

• README.md	описание проекта;
КОНФИГУРАЦИЯ DOCKER	
• docker-compose.yml	конфигурация сервисов docker;
• images/nginx: ◦ Dockerfile ◦ nginx.conf	конфигурация сервиса nginx;
• system/nginx/server.conf	конфигурация вебсервера;
• images/php/Dockerfile	установка pdo-mysql;
КОНФИГУРАЦИЯ NPM	
• package.json	описание npm-пакета
• package-lock.json	
• semantic.json	конфигурационный файл semantic-ui
• .gitignore	gitignore для директории semantic
ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС (UI)	
• templates/index.html	шаблон HTML-страницы
• www:	
◦ page.js	имплементация клиентской части проекта
◦ page.php	входная точка (entrypoint) для проекта
МИГРАЦИИ (SQL-данные)	
• migrations: ◦ create.sql ◦ data.sql	миграционные файлы проекта; создание таблицы; наполнение данными;
• scripts	
◦ bootstrap.php	конфигурация PSR-0 autoload
◦ Project	
▪ Controller.php	контроллер обработки URL-запросов
▪ Database.php	обертка-конфигуратор PDO
ИСКЛЮЧЕНИЯ (EXCEPTIONS)	
▪ Exception	
• Exception.php	Базовое, наследуемое исключение
• Client.php	Исключения по вине клиента
• BadRequest.php	Неверный формат входных данных
• UnspecifiedParameter.php	Не был указан обязательный параметр
• UnknownUrl.php	Не найден endpoint для обработки запроса
ОБРАБОТКА ЗАПРОСОВ (API ENDPOINTS)	
▪ Page	
• PageAbstract.php	абстрактный endpoint
• Root.php	/
• Company.php	/company
• Init.php	/init
• Types.php	/type
• Column.php	/column
• Languages.php	/language
• LanguagesColumn.php	/language/column/<column_name>
• LanguagesExcludingType.php	/language/exclude_by_type/<type_name>
• LanguagesFamiliar.php	/language/familiar/<company_name>

Приложение 1. Код программы

./README.md

```
# Базы данных - контрольная работа - задание
## Написать последовательность команд:

1. создания таблицы;
2. начального заполнения таблицы данными;
3. выборки из нее итоговой информации.

Таблица должна содержать данные согласно вашему варианту задания, номер варианта соответствует номеру в журнале группы.
Имя таблицы должно быть в точности таким, как написано в задании. Изменять данные в приведенных таблицах НЕ
РАЗРЕШАЕТСЯ!
### Т3 - Языки программирования (ЯП).
Обязательные поля: номер, название языка, тип языка, фирма-разработчик (выбрана условно).

| N  | Назв.        | Тип     | Фирма     |
|----|--------------|---------|-----------|
| 1  | Pascal       | Процед  | Borland   |
| 2  | C            | Процед  | Borland   |
| 3  | Java         | Процед  | Java inc  |
| 4  | C++          | Объект  | Java inc  |
| 5  | Visual C     | Объект  | Microsoft |
| 6  | Visual Basic | Объект  | Microsoft |
| 7  | Delphi       | Объект  | Borland   |
| 8  | Lisp         | Сценарн | IBM       |
| 9  | Prolog       | Сценарн | IBM       |
| 10 | XML          | Сценарн | Borland   |

## Написать три PHP-скрипта:
1. создания таблицы в СУБД MySQL и начального заполнения таблицы данными,
2. создания HTML-формы (указанного вида) для выбора имени столбца таблицы,
3. вывод в браузере содержимого выбранного столбца.

Имя таблицы должно быть в точности таким, как написано в задании.
Отчет по КР должен отвечать всем требованиям к оформлению КР, содержать текст задания, тексты сценариев и результаты
выполнения команд.

Таблица Т3
1. Вывести число строк, не содержащих данные о языках, тип которой задавать с помощью параметра.
2. Вывести список языков, кроме тех , что относятся к первому или последнему (задавать с помощью параметра) по алфавиту
типу.
3. Вывести названия языков, имеющих те же типы, что и типы языков, выпускаемые фирмой, указанной в параметре.
HTML-форма: флажки-переключатели.

## Инструкция по запуску
1. Убедиться, что установлен:
    - [docker-ce](https://docs.docker.com/install/)
    - [docker-compose](https://docs.docker.com/compose/install/)
    - [node js](https://nodejs.org/en/download/)
    - [npm](https://www.npmjs.com/get-npm)
2. Перейти в директорию с проектом

        cd sibsutis-labs/databases
3. Установить необходимые проекту пакеты npm:

        npm install
4. Запустить для инициализации серверов:

        docker-compose up -d
5. Перейти на http://127.0.0.1:8200
```

./images/php/Dockerfile

```
FROM php:fpm

RUN docker-php-ext-configure pdo_mysql --with-pdo-mysql \
        && docker-php-ext-install pdo_mysql
```

./images/nginx/Dockerfile

```
FROM nginx:latest
RUN mkdir /data/project -p
COPY nginx.conf /etc/nginx/
```

./images/nginx/nginx.conf

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /data/config/*.conf;
}
```

./docker-compose.yml

```
version: '2'
services:
    nginx:
        build: ./images/nginx/
        links:
            - fpm
        ports:
            - 8200:80
        volumes:
            - ./system/nginx/:/data/config/
            - .:/data/project

    fpm:
        build: ./images/php/
        links:
            - mariadb
        volumes:
            - .:/data/project

    mariadb:
        image: mariadb:latest
        environment:
            - MYSQL_DATABASE=mydb
            - MYSQL_USER=myuser
            - MYSQL_PASSWORD=secret
            - MYSQL_ROOT_PASSWORD=docker

    phpmyadmin:
        image: phpmyadmin/phpmyadmin
        links:
            - mariadb:mysql
        environment:
            - PMA_HOST=mysql
            - MYSQL_USERNAME=root
            - MYSQL_ROOT_PASSWORD=docker
```

./system/nginx/server.conf

```
server {
    listen 80;

    root /data/project/www;

    location / {
        try_files $uri /page.php$is_args$args;
    }

    location ~ ^/.+\.php(/|$) {
        fastcgi_pass fpm:9000;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
```

./package.json

```
{
  "name": "databases",
  "version": "1.0.0",
  "description": "Sample for Databases course in SibSUTIS",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Evgeniy Vasilev http://evgeniyvasilev.ru",
  "license": "MIT",
  "dependencies": {
    "semantic-ui": "^2.4.2"
  }
}
```

./package-lock.json [BLOB]

Автогенерируемый файл, генерируется при запуске npm install.

./semantic.json

```
{
  "base": "www/semantic",
  "paths": {
    "source": {
      "config": "src/theme.config",
      "definitions": "src/definitions/",
      "site": "src/site/",
      "themes": "src/themes/"
    },
    "output": {
      "packaged": "dist/",
      "uncompressed": "dist/components/",
      "compressed": "dist/components/",
      "themes": "dist/themes/"
    },
    "clean": "dist/"
  },
  "permission": false,
  "autoInstall": true,
  "rtl": false,
  "version": "2.4.2"
}
```

./.gitignore

```
www/semantic/
```

./templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta content="text/html; charset=UTF-8" http-equiv="content-type">
    <title>Databases course work</title>
    <link rel="stylesheet" type="text/css" href="semantic/dist/semantic.min.css">
    <script
            src="https://code.jquery.com/jquery-3.1.1.min.js"
            integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
            crossorigin="anonymous"></script>
    <script src="semantic/dist/semantic.min.js"></script>
    <style type="text/css">
        .ui.checkbox label {
            cursor: pointer;
        }
    </style>

</head>
<body role="document">

<div class="ui text container">
    <h2 class="first">Coursework demonstration</h2>
    <p>Repository url: <a href="https://github.com/johnynsk/sibsutis-labs/"><i class="icon github"></i>johnynsk /
<b>sibsutis-labs</b></a></p>
    <p>
        <button class="ui labeled icon primary button" data-action="recreate"><i class="icon redo"></
i>Recreate</button>
        <button class="ui labeled icon button green" data-action="fill"><i class="icon plus"></i>Fill table</button>
        <button class="ui labeled icon button yellow" data-action="recreate-fill"><i class="icon rocket"></i>Recreate
and fill table</button>
    </p>

    <div class="ui message hidden">
    </div>

    <div class="ui top attached tabular menu">
        <a class="item active" data-tab="all-records">All records</a>
        <a class="item" data-tab="columns">Select only one column</a>
        <a class="item" data-tab="excluding">Excluding type</a>
        <a class="item" data-tab="companies">Familiar companies</a>
    </div>
    <div class="ui bottom attached tab segment active" data-tab="all-records">
        <p>Here you can show all records in the table</p>
        <p><button class="ui labeled icon primary button" data-action="reload-all"><i class="icon search"></i>Show
all</button></p>
        <div class="table-data"></div>
    </div>
    <div class="ui bottom attached tab segment form" data-tab="columns">
        <p>Here you can receive the data about available columns in the table</p>
        <p><button class="ui labeled icon primary button" data-action="columns-refresh"><i class="icon redo"></
i>Refresh</button></p>
        <div class="grouped fields buttons-data"></div>
        <p class="button-placeholder"></p>
        <h3 class="ui header counter"></h3>
        <div class="table-data"></div>
    </div>
    <div class="ui bottom attached tab segment form" data-tab="excluding">
        <p>Here you can filter records in the table by the specific value</p>
        <div class="grouped fields buttons-data"></div>
        <p class="button-placeholder"></p>
        <h3 class="ui header counter"></h3>
        <div class="table-data"></div>
    </div>
    <div class="ui bottom attached tab segment form" data-tab="companies">
        <p>Here you can receive all records familiar by the specified argument</p>
        <div class="grouped fields buttons-data"></div>
        <p class="button-placeholder"></p>
        <h3 class="ui header counter"></h3>
        <div class="table-data"></div>
    </div>
</div>

<script src="/page.js"></script>
</body>
</html>
```

./www/page.js

```
(function () {
    $('.menu .item').tab();

    let store = {
        columns: null,
    };
```

```
    let renderTable = function (data) {
        let table = $('<table class="ui celled table"></table>');

        let thead = $('<thead></thead>');
        let headRow = $("<tr></tr>");
        store.columns.forEach(column => headRow.append($(`<th>${column}</th>`)));
        thead.append(headRow);
        table.append(thead);

        data.forEach(item => {
            let row = $("<tr></tr>");
            store.columns.forEach(column => row.append($(`<td>${column in item ? item[column] : '&mdash;'}</td>`)));
            table.append(row);
        });

        return table;
    };

    let bindRenderBlock = function (tabName, optionsEndpoint, mapper, filteringEndpoint, tableMapper, buttonText) {
        $(`a[data-tab="${tabName}"]`).click(function () {
            let resultTable = $(`div[data-tab="${tabName}"] .table-data`);

            operate.get(optionsEndpoint, function (response) {
                let records = mapper(response);

                let container = $(`div[data-tab="${tabName}"] .buttons-data`);
                container.html("");

                records.forEach((item, order) => {
                    container.append($(`<div class="field">
    <div class="ui checkbox radio">
        <input type="radio" name="${tabName}" value="${item}" id="${tabName}_${order}">
        <label for="${tabName}_${order}">${item}</label>
    </div>
</div>`));
                });

                let button = $(`<button class="ui button green">${buttonText}</button>`)
                button.click(function () {
                    let type = $(`input[name=${tabName}]:checked`).val();

                    operate.get(`${filteringEndpoint}${type}`, function (response) {
                        resultTable.html(renderTable(tableMapper(response)));

                        if ('count' in response) {
                            $(`div[data-tab="${tabName}"] h3.counter`).html(`Found ${response.count} items:`)
                        }
                    });
                });

                container.append(button);
            });
        });
    };

    let operate = function () {
        let notificate = function () {};
        let load = function () {};

        let public = {
            get: function (url, success, fail) {
                load();
                $.get(url, null, function (response) {
                    if ('error' in response) {
                        if (typeof fail != 'undefined') {
                            return fail(response);
                        }
                    }

                    if (typeof success != 'undefined') {
                        return success(response);
                    }
                }, 'json')
            },
            post: function (url, payload, success, fail) {
                load();
                $.post(url, JSON.stringify(payload), function (response) {
                    if ('error' in response) {
                        if (typeof fail != 'undefined') {
                            return fail(response);
                        }
                    }

                    if (typeof success != 'undefined') {
                        success(response);
                    }
                }, 'json')
            }
        };

        return public;
    }();
```

```
    var syncColumns = function (callback) {
        operate.get("/column", function (response) {
            store.columns = response.columns;
            if (typeof callback !== 'undefined') {
                callback(response);
            }
        });
    };
    syncColumns();

    $('button[data-action="columns-refresh"]').click(syncColumns);

    $('button[data-action="recreate"]').click(function () {
        operate.post("/init", {structure: true});
    });

    $('button[data-action="fill"]').click(function () {
        operate.post("/init", {data: true});
    });

    $('button[data-action="recreate-fill"]').click(function () {
        operate.post("/init", {data: true, structure: true});
    });

    $('button[data-action="reload-all"]').click(function () {
        operate.get("/language", function (response) {
            $('div[data-tab="all-records"] .table-data').html(renderTable(response.languages));
        });
    });

    bindRenderBlock("excluding", "/type", response => {
        let result = response.types;
        result.unshift("first");
        result.push("last");
        return result;
    }, "/language/exclude_by_type/", item => item.languages, 'Exclude type');

    bindRenderBlock("companies", "/company", response => response.companies,
        "/language/familiar/", item => item.languages, 'Show familiar languages');

    bindRenderBlock("columns", "/column", response => store.columns = response.columns,
        "/language/column/", item => item.languages, 'Show only one column');
})();
```

./www/page.php

```
<?php

call_user_func(function()
{
    require_once dirname(__DIR__) . '/scripts/bootstrap.php';
    $handler = function (Exception $exception) {
        echo json_encode([
            'error' => [
                'message' => $exception->getMessage(),
                'trace' => $exception->getTrace(),
                'class' => get_class($exception),
            ]
        ], JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
    };

    try {
        if (!isset($_ENV['REQUEST_URI'])) {
            throw new Exception('env.REQUEST_URI was not set');
        }

        $db = new Project\Database('mariadb', 'myuser', 'secret', 'mydb');

        $rawPayload = file_get_contents('php://input');
        $payload = json_decode($rawPayload, true);

        if (!empty($rawPayload) && json_last_error()) {
            throw new Project\Exception\Client("Payload should be in the json format.");
        }

        $controller = new Project\Controller();
        $controller->register('#^/$#', ['GET'], new Project\Page\Root($db));
        $controller->register('#^/init/?$#', ['POST'], new Project\Page\Init($db));
        $controller->register('#^/type/?$#', ['GET'], new Project\Page\Types($db));
        $controller->register('#^/column/?#', ['GET'], new Project\Page\Column($db));
        $controller->register('#^/company/?#', ['GET'], new Project\Page\Company($db));
        $controller->register('#^/language/?$#usi', ['GET'], new Project\Page\Languages($db));
        $controller->register('#^/language/exclude_by_type/(?P<exclude_type>[^/]+)/?$#u', ['GET'], new Project\Page\
LanguagesExcludingType($db));
        $controller->register('#^/language/familiar/(?P<company_name>[^/]+)/?$#u', ['GET'], new Project\Page\
LanguagesFamiliar($db));
        $controller->register('#^/language/column/(?P<column_name>[^/]+)/?$#u', ['GET'], new Project\Page\
LanguagesColumn($db));
```

```
        ob_start();
        /** @var $invoker \Project\Page\PageAbstract */
        list($invoker, $arguments) = $controller->getInvokerAndArguments($_ENV['REQUEST_URI'],
$_ENV['REQUEST_METHOD']);
        $result = $invoker->invoke($payload, $arguments);
        $headers = $invoker->getHeaders();
        foreach ($headers as $header) {
            header($header);
        }

        if (is_object($result) || is_array($result)) {
            header("Content-Type: application/json");
            echo json_encode($result, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
            return;
        }

        $contents = ob_get_clean();
        echo $contents;
    } catch (\Project\Exception\UnknownUrl $exception) {
        header("HTTP/1.0 404 Not Found");
        $handler($exception);
    } catch (\Project\Exception\Client $exception) {
        header("HTTP/1.0 400 Bad Request");
        $handler($exception);
    } catch (Exception $exception) {
        header("HTTP/1.0 500 Internal Server Error");
        $handler($exception);
    }
});
```

./migrations/create.sql

```
DROP TABLE IF EXISTS `Языки`;
CREATE TABLE `Языки` (
  `N` int(2) UNSIGNED NOT NULL,
  `Назв` char(12) NOT NULL,
  `Тип` enum('Процед','Объект','Сценарн','') NOT NULL,
  `Фирма` char(9) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `Языки`
  ADD PRIMARY KEY (`N`);

ALTER TABLE `Языки`
  MODIFY `N` int(2) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;
```

./migrations/data.sql

```
INSERT INTO `Языки` (`N`, `Назв`, `Тип`, `Фирма`) VALUES
(NULL, 'Pascal', 'Процед', 'Borland'),
(NULL, 'C', 'Процед', 'Borland'),
(NULL, 'Java', 'Процед', 'Java inc'),
(NULL, 'C++', 'Объект', 'Java inc'),
(NULL, 'Visual C', 'Объект', 'Microsoft'),
(NULL, 'Visual Basic', 'Объект', 'Microsoft'),
(NULL, 'Delphi', 'Объект', 'Borland'),
(NULL, 'Lisp', 'Сценарн', 'IBM'),
(NULL, 'Prolog', 'Сценарн', 'IBM'),
(NULL, 'XML', 'Сценарн', 'Borland');
```

./scripts/bootstrap.php

```
<?php

define('ROOT_DIR', dirname(__DIR__));
define('SCRIPTS_DIR', __DIR__);

/**
 * PSR-0 autoload
 */
spl_autoload_register(function ($className) {
    $base_dir = __DIR__;

    $file = __DIR__ . '/' . str_replace(['\\', '_'], '/', $className) . '.php';

    if (file_exists($file)) {
        require $file;
    }
});

define('MIGRATIONS_PATH', implode([ROOT_DIR, 'migrations'], DIRECTORY_SEPARATOR));
define('TEMPLATES_PATH', implode([ROOT_DIR, 'templates'], DIRECTORY_SEPARATOR));
```

./scripts/Project/Controller.php

```php
<?php

namespace Project;

use Project\Exception\UnknownUrl;
use Project\Page\PageAbstract;

class Controller {
    private $controllers = [];
    public function __construct() {
    }

    public function register($regexp, $methods, PageAbstract $controller) {
        array_push($this->controllers, [
            'regexp' => $regexp,
            'methods' => $methods,
            'invoker' => $controller
            ]);
    }

    /**
     * @param $path
     * @param $method
     * @return [PageAbstract, string[]]
     * @throws UnknownUrl
     */
    public function getInvokerAndArguments($path, $method)
    {
        foreach ($this->controllers as $controller) {
            if (!preg_match($controller['regexp'], $path, $matches)) {
                continue;
            }

            if (!in_array($method, $controller['methods'])) {
                continue;
            }

            return [$controller['invoker'], $matches];
        }

        throw new UnknownUrl("You have specified the wrong URL. The handler does not exists.");
    }
}
```

./scripts/Project/Database.php

```php
<?php

namespace Project;
use \PDO;

class Database extends PDO {
    public function __construct($hostname, $user, $password, $database) {
        parent::__construct("mysql:host=${hostname};dbname=${database};charset=utf8", $user, $password);
        $this->setAttribute(self::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $this->setAttribute(self::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    }
}
```

./scripts/Project/Exception/Exception.php

```php
<?php

namespace Project\Exception;

class Exception extends \Exception {};
```

./scripts/Project/Exception/Client.php

```php
<?php

namespace Project\Exception;

class Client extends Exception {};
```

./scripts/Project/Exception/BadRequest.php

```php
<?php

namespace Project\Exception;

class BadRequest extends Client {};
```

./scripts/Project/Exception/UnspecifiedParameter.php

```php
<?php

namespace Project\Exception;

class UnspecifiedParameter extends Client
{
    public function __construct($parameterName)
    {
        parent::__construct("You have not specified the mandatory \"${parameterName}\" parameter.");
    }
}
```

./scripts/Project/Exception/UnknownUrl.php

```php
<?php

namespace Project\Exception;

class UnknownUrl extends Client {};
```

./scripts/Project/Page/PageAbstract.php

```php
<?php

namespace Project\Page;
use Project\Database;

abstract class PageAbstract {
    /**
     * @var Database
     */
    protected $db;

    private $headers = [];

    public function __construct(Database $db) {
        $this->db = $db;
    }

    abstract public function invoke($payload, $arguments);

    protected function setHeader($header) {
        $this->headers[] = $header;

        return $this;
    }

    public function getHeaders() {
        return $this->headers;
    }
};
```

./scripts/Project/Page/Root.php

```php
<?php

namespace Project\Page;

class Root extends PageAbstract {
    public function invoke($payload, $arguments) {
        include TEMPLATES_PATH . "/index.html";
    }
}
```

./scripts/Project/Page/Company.php

```php
<?php

namespace Project\Page;

class Company extends PageAbstract {
    public function invoke($payload, $arguments) {
        $query = $this->db->prepare("SELECT DISTINCT(`Фирма`) FROM `Языки` ORDER BY `Фирма` ASC");
        $query->execute();
        $columns = array_map(function($item) {
            return reset(array_values($item));
        }, $query->fetchAll());

        return [
            'count' => count($columns),
            'companies' => $columns
        ];
    }
}
```

./scripts/Project/Page/Init.php

```php
<?php

namespace Project\Page;

use Project\Exception\BadRequest;

class Init extends PageAbstract {
    public function invoke($payload, $arguments) {
        $result = [];

        if (isset($payload["structure"])) {
            $structure = file_get_contents(MIGRATIONS_PATH . "/create.sql");
            $this->db->exec($structure);
            $result["structure"] = true;
        }

        if (isset($payload["data"])) {
            $data = file_get_contents(MIGRATIONS_PATH . "/data.sql");
            $this->db->exec($data);
            $result["data"] = true;
        }

        if (count($result) == 0) {
            throw new BadRequest("You have to specify the target for migrations.");
        }

        $this->setHeader("HTTP/1.0 201 Created");

        return $result;
    }
}
```

./scripts/Project/Page/Types.php

```php
<?php

namespace Project\Page;

class Types extends PageAbstract {
    public function invoke($payload, $arguments) {
        $query = $this->db->prepare("SELECT DISTINCT(`Тип`) FROM `Языки` ORDER BY `Тип` ASC");
        $query->execute();

        $types = array_map(function($type) {
            return reset(array_values($type));
        }, $query->fetchAll());

        return [
            'count' => count($types),
            'types' => $types
        ];
    }
}
```

./scripts/Project/Page/Column.php

```php
<?php

namespace Project\Page;

class Column extends PageAbstract {
    public function invoke($payload, $arguments) {
        $query = $this->db->prepare("SELECT COLUMN_NAME FROM information_schema.columns
WHERE table_schema='mydb' AND table_name='Языки'");
        $query->execute();
        $columns = array_map(function($item) {
            return reset(array_values($item));
        }, $query->fetchAll());

        return [
            'count' => count($columns),
            'columns' => $columns
        ];
    }
}
```

./scripts/Project/Page/Languages.php

```php
<?php

namespace Project\Page;

class Languages extends PageAbstract {
    public function invoke($payload, $arguments) {
        $query = $this->db->prepare("SELECT * FROM `Языки` ORDER BY `N` ASC");
        $query->execute();

        $langs = $query->fetchAll();

        return [
            'count' => count($langs),
            'languages' => $langs
        ];
    }
}
```

./scripts/Project/Page/LanguagesColumn.php

```php
<?php

namespace Project\Page;

use Project\Exception\BadRequest;
use Project\Exception\UnspecifiedParameter;

class LanguagesColumn extends PageAbstract {
    public function invoke($payload, $arguments) {
        if (!isset($arguments["column_name"])) {
            throw new UnspecifiedParameter("column_name");
        }

        $columns = (new Column($this->db))->invoke(null, null);
        $columnName = urldecode($arguments["column_name"]);

        if (!in_array($columnName, $columns["columns"])) {
            throw new BadRequest("You have specified the wrong column name.");
        }


        $query = $this->db->prepare("SELECT `${columnName}` FROM `Языки` ORDER BY `N` ASC");
        $query->execute();
        $langs = $query->fetchAll();

        $query = $this->db->prepare("SELECT COUNT(*) AS `count` FROM `Языки`");
        $query->execute();
        $count = $query->fetch();

        return [
            'count' => (int)$count["count"],
            'languages' => $langs
        ];
    }
}
```

./scripts/Project/Page/LanguagesExcludingType.php

```php
<?php

namespace Project\Page;

use Project\Exception\UnspecifiedParameter;

class LanguagesExcludingType extends PageAbstract {
    public function invoke($payload, $arguments) {
        if (!isset($arguments["exclude_type"])) {
            throw new UnspecifiedParameter("exclude_type");
        }

        $type = urldecode($arguments['exclude_type']);

        if ($type == 'first') {
            $query = $this->db->prepare("SELECT * FROM `Языки` WHERE `Тип` != (SELECT DISTINCT(`Тип`) FROM `Языки`
ORDER BY `Тип` ASC LIMIT 1) ORDER BY `N` ASC");
            $query->execute();
        } else if ($type == 'last') {
            $query = $this->db->prepare("SELECT * FROM `Языки` WHERE `Тип` != (SELECT DISTINCT(`Тип`) FROM `Языки`
ORDER BY `Тип` DESC LIMIT 1) ORDER BY `N` ASC");
            $query->execute();
        } else {
            $query = $this->db->prepare("SELECT * FROM `Языки` WHERE `Тип` != ? ORDER BY `N` ASC");
            $query->execute([$type]);
        }

        $langs = $query->fetchAll();

        return [
            'count' => count($langs),
            'languages' => $langs
        ];
    }
}
```

./scripts/Project/Page/LanguagesFamiliar.php

```php
<?php

namespace Project\Page;

use Project\Exception\UnspecifiedParameter;

class LanguagesFamiliar extends PageAbstract {
    public function invoke($payload, $arguments) {
        if (!isset($arguments["company_name"])) {
            throw new UnspecifiedParameter("company_name");
        }

        $query = $this->db->prepare("SELECT * FROM `Языки` WHERE `Тип` IN (SELECT DISTINCT(`Тип`) FROM `Языки` WHERE
`Фирма` = ?) ORDER BY `N` ASC");
        $query->execute([urldecode($arguments['company_name'])]);
        $langs = $query->fetchAll();

        return [
            'count' => count($langs),
            'languages' => $langs
        ];
    }
}
```