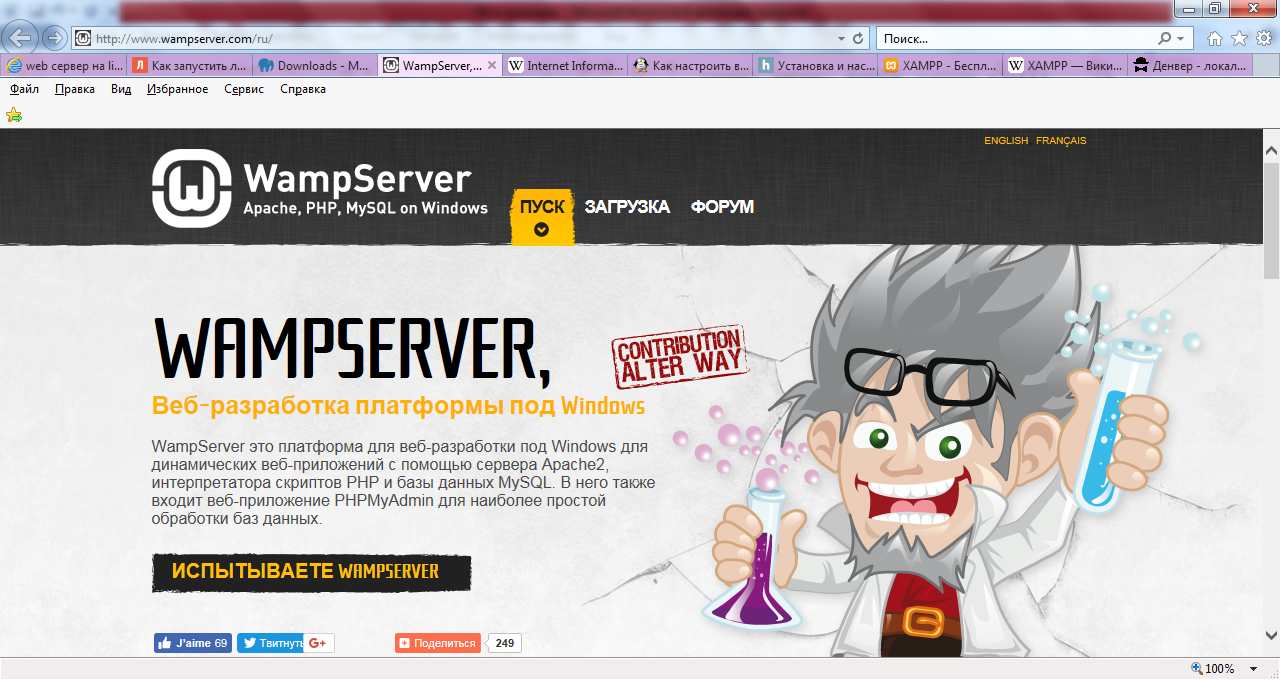
1. Обзор локальных web-серверов.
2. Установка и настройка Open Server
3. Установка и настройка IDE – среды разработки приложений. Бесплатная IDE: NetBeans IDE
4. Установка Laravel с помощью менеджера зависимостей COMPOSER
5. Подключение системы контроля версий GIT
6. Настройка проекта в режим Разработка, добавление CSS фреймворка, настройка OpenServer под новый проект.
7. Структура проекта и отрисовка представлений.
8. Установка и настройка web-сервера.

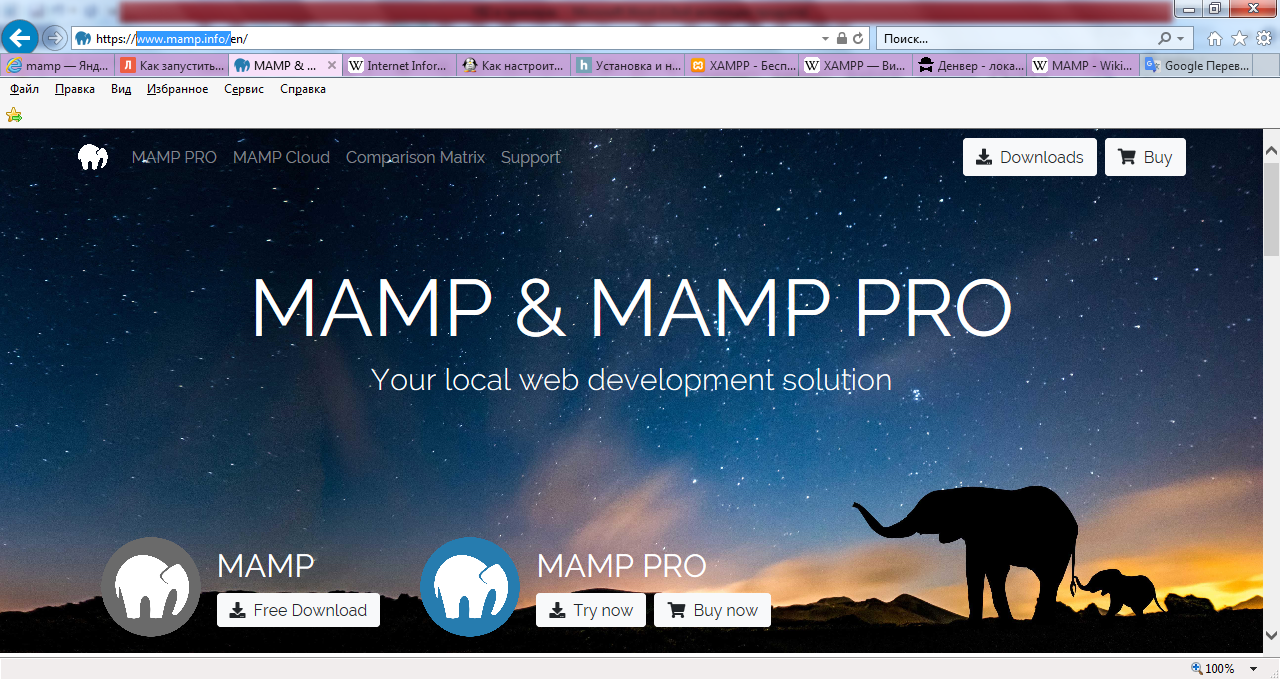
В последнее время выбор web-серверов для локальной web-разработки огромен. В данной книге будет использоваться OpenServer. Подобные решения «все в одном» становятся все более популярными для локальной разработки, так как сводят процесс инсталляции и настройки сервера к минимуму. Его установку и основные настройки я опишу вкарце далее.

Небольшой обзор имеющихся решений для локального web-сервера.

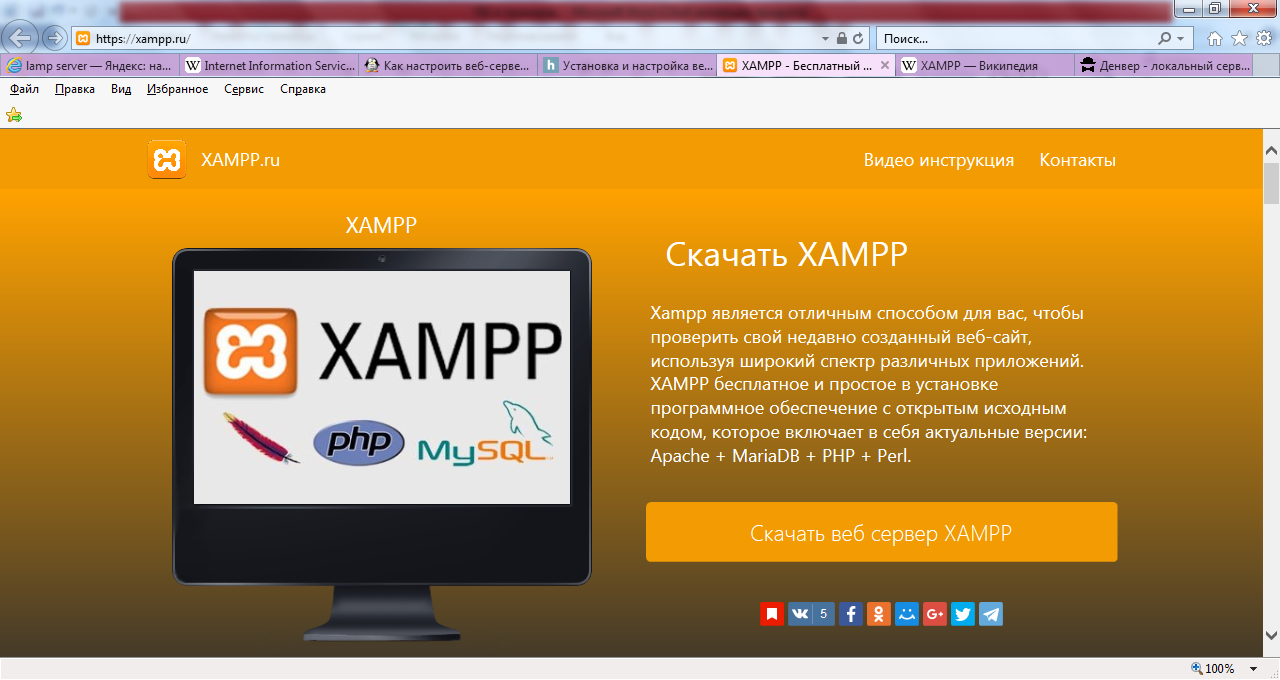
**WAMPServer** – (аббревиатура Windows, Apache, MySQL, PHP) бесплатный продукт для Windows систем из разряда «все в одном». WampServer это платформа для веб-разработки под Windows для динамических веб-приложений с помощью сервера Apache2, интерпретатора скриптов PHP и базы данных MySQL. В него также входит веб-приложение PHPMyAdmin для наиболее простой обработки баз данных. Необходимо только скачать, установить и произвести первичные настройки, которые можно найти на сайте разработчика (www.wampserver.com).



**MAMP** - это пакет решений, состоящий из свободного и открытого исходного кода и запатентованного коммерческого программного обеспечения, используемого для запуска динамических веб-сайтов на компьютерах Mac OS. Из-за популярности MAMP теперь также доступен для Windows 7 и выше. Существует бесплатная версия MAMP и платная версия MAMP PRO (www.mamp.info).



**XAMPP** - кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер (xampp.ru).



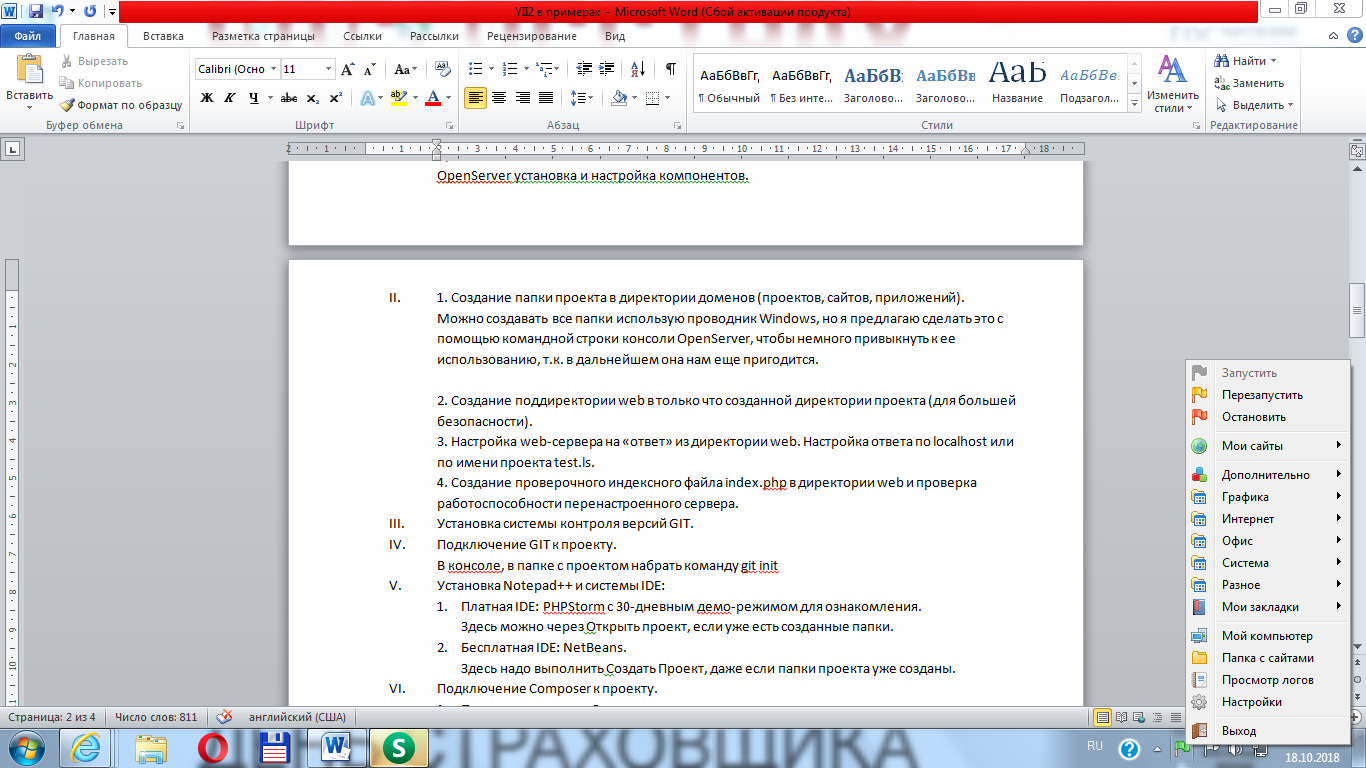
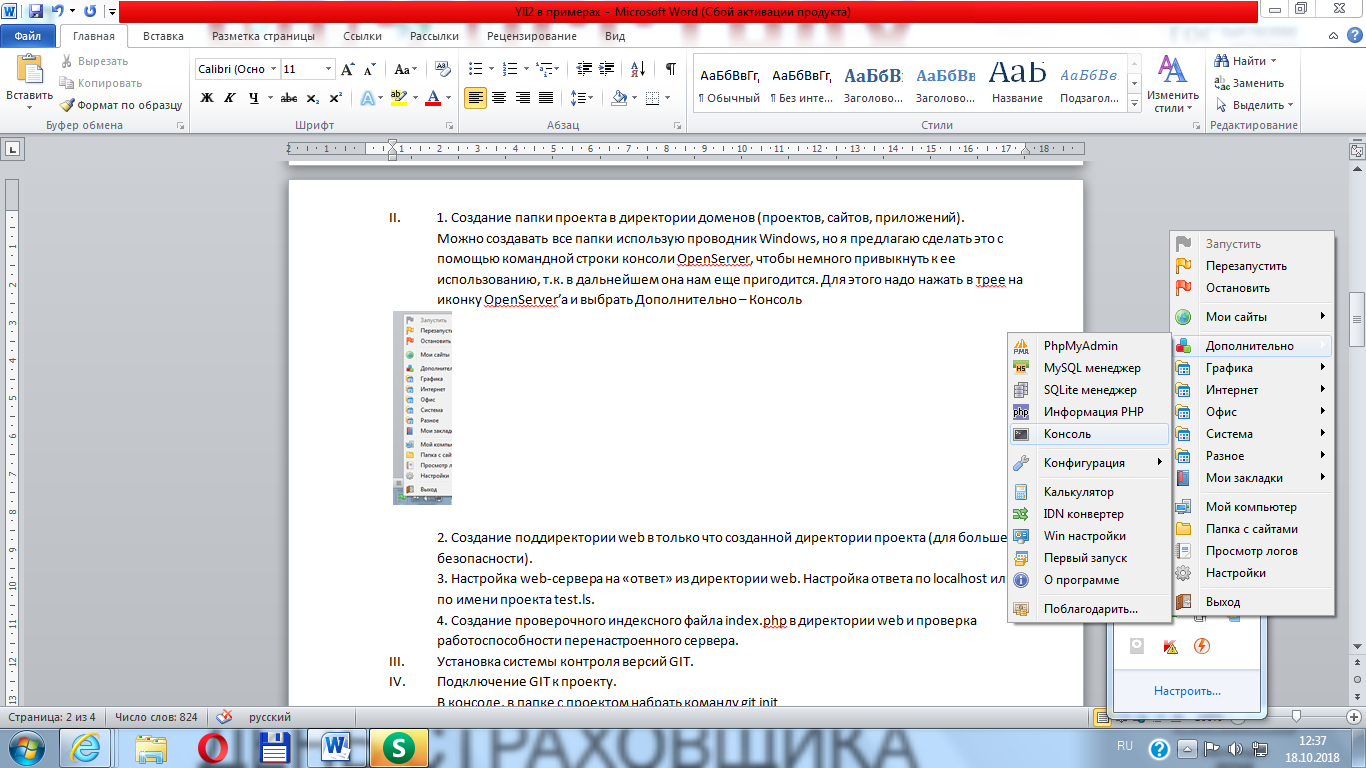
Можно выбрать немного более сложные решения, в том плане, что либо компоненты этих решений необходимо доустанавливать по-отдельности, либо больше настроек, но появляется более точно настроить локальный сервер под себя и под свои проекты. А именно Denwer, web-сервер в IIS представленной Microsoft во всех версиях Windows и т.д.

Установка web-сервера на системы семейства Unix в данной книге не рассматривается. Такую информацию с подробными последовательностями действий можно довольно легко найти в Интернете.

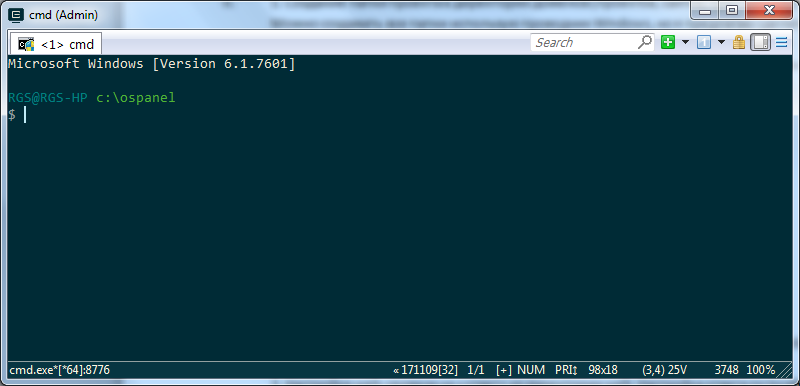
OpenServer установка и настройка компонентов.

1. 1. Создание папки проекта в директории доменов (проектов, сайтов, приложений).

Можно создавать все папки использую проводник Windows, но я предлагаю сделать это с помощью командной строки консоли OpenServer, чтобы немного привыкнуть к ее использованию, т.к. в дальнейшем она нам еще пригодится. Для этого надо нажать в трее на иконку OpenServer’а и выбрать Дополнительно – Консоль

Вот то, что у нас откроется



Мы находимся в папке с развернутым web-сервером (это ospanel или openserver – в зависимости от версии и установки OpenServer)

Набираем в dir, нажимаем Enter и видим список папок в папке установки.

Нас интересует папка domains в которую OpenServer по умолчанию предлагает разместить проекты. Открываем ее командой cd domains и создаем в ней новую папку с нашим будущим проектом. Предлагаю назвать ее first.lc (название проекту можете дать любое, т.к. вы всегда можете его сменить. Префикс .lc использую по старой привычке для обозначения локального расположения).

mkdir first.lc

И в нашей директории domains появилась поддиректория first.lc, в которой и будет располагаться наш проект.

3. Настройка web-сервера на «ответ» из директории web. Настройка ответа по localhost или по имени проекта test.ls.

4. Создание проверочного индексного файла index.php в директории web и проверка работоспособности перенастроенного сервера.

1. Установка системы контроля версий GIT.
2. Подключение GIT к проекту.

В консоле, в папке с проектом набрать команду git init

1. Установка Notepad++ и системы IDE:
2. Платная IDE: PHPStorm с 30-дневным демо-режимом для ознакомления.

Здесь можно через Открыть проект, если уже есть созданные папки.

1. Бесплатная IDE: NetBeans.

Здесь надо выполнить Создать Проект, даже если папки проекта уже созданы.

1. Подключение Composer к проекту.
2. Проверка установки Composer из папки проекта

$ composer

Небольшая вставка об инструменте Composer

1. Если Composer не установлен:

Установка Composer на различных системах.

1. Создание файла .gitignore

Правой клавишей в IDE:

В PHPStorm: New – File

В NetBeans: он его создает сам

PHPStorm: после создания, файл открывается автоматически для внесения изменений. Галочка ставится на те файлы, которые не надо добавлять в GIT.

Первоначально только .gitignore

Чтобы закоммитить надо:

В PHPStorm: правой клавишей по паке проекта – CommitDirectory.

В NetBeans: правой клавишей по папке пректа – GIT – Фиксировать…

Создаем новую таблицу в БД, используя миграции.

Синтаксис:

php artisan make:migration migration\_name --create="table\_name"

Нам надо создать таблицу litters с записями

id автоинкремент, integer, первичный ключ

litter – строковая, 2 символа

php artisan make:migration create\_litters\_table --create="litters"

Дожидаемся подтверждения создания миграции.

Находим файл здесь\_будет\_дата\_и\_штамп\_времени \_create\_litters\_table.php миграции в папке database\migrations и начинаем его редактировать.

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateLittersTable extends Migration

{

public function up()

{

Schema::create('litters', function (Blueprint $table) {

$table->increments('id');

$table->string('litter', 2);

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('litters');

}

}

И запускаем нашу миграцию командой

**php** artisan migrate

Если при применении миграции вы получаете ошибку «class not found» («класс не найден»), попробуйте выполнить команду **composer** dump-autoload и заново запустить команду **migrate**.

**Принудительные миграции в продакшне**

Некоторые операции миграций разрушительны, значит они могут привести к потере ваших данных. Для предотвращения случайного запуска этих команд на вашей боевой БД перед их выполнением запрашивается подтверждение. Для принудительного запуска команд без подтверждения используйте ключ **--force**:

**php** artisan migrate --force

Как обычно для unix систем просмотреть подсказки по доступным опциям команды migrate можно вызвав справку:

php artisan migrate –h

Вот еще полезные опции данной команды:

migrate:install Create the migration repository

migrate:refresh Reset and re-run all migrations

migrate:reset Rollback all database migrations

migrate:rollback Rollback the last database migration

migrate:status Show the status of each migration

[ОТКАТ МИГРАЦИЙ](http://laravel.su/docs/5.0/migrations#rolling-back-migrations)

Отмена изменений последней миграции

php artisan migrate:rollback

Отмена изменений всех миграций

php artisan migrate:reset

Откат всех миграций и их повторное применение

php artisan migrate:refresh

php artisan migrate:refresh --seed

Сразу создадим Модель для нашей таблицы:

php artisan make:model Litter

И отредактируем app\Litter.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Litter extends Model

{

protected $fillable = ["litter"];

}

Мы можем для тестирования использовать механизм наполнения таблиц БД начальными данными с помощью специальных классов. Они создаются и хранятся в директории database\seeds.

Создадим класс для добавления данных в только что созданную таблицу нашей БД.

php artisan make:seeder LitterTableSeeder

Теперь открываем новый класс в database\seeds\LitterTableSeeder.php

И правим класс, занося наши данные:

<?php

use Illuminate\Database\Seeder;

use App\Litter;

class LitterTableSeeder extends Seeder

{

public function run()

{

Litter::create(array('litter' => 'A'));

}

}

Теперь мы можем запустить наполнение нашей таблицы новыми данными, выполнив команду

php artisan db:seed --class=LittersTableSeeder

Но мы не будем так делать. Мы внесем изменения в уже имеющемся во фреймворке классе DatabaseSeeder, чтобы контролировать последовательное наполнение данными наши таблицы. Откроем для этого файл database\seeds\DatabaseSeeder.php и исправим его:

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

$this->call('LitterTableSeeder');

$this->command->info(‘operation completed successfully!’);

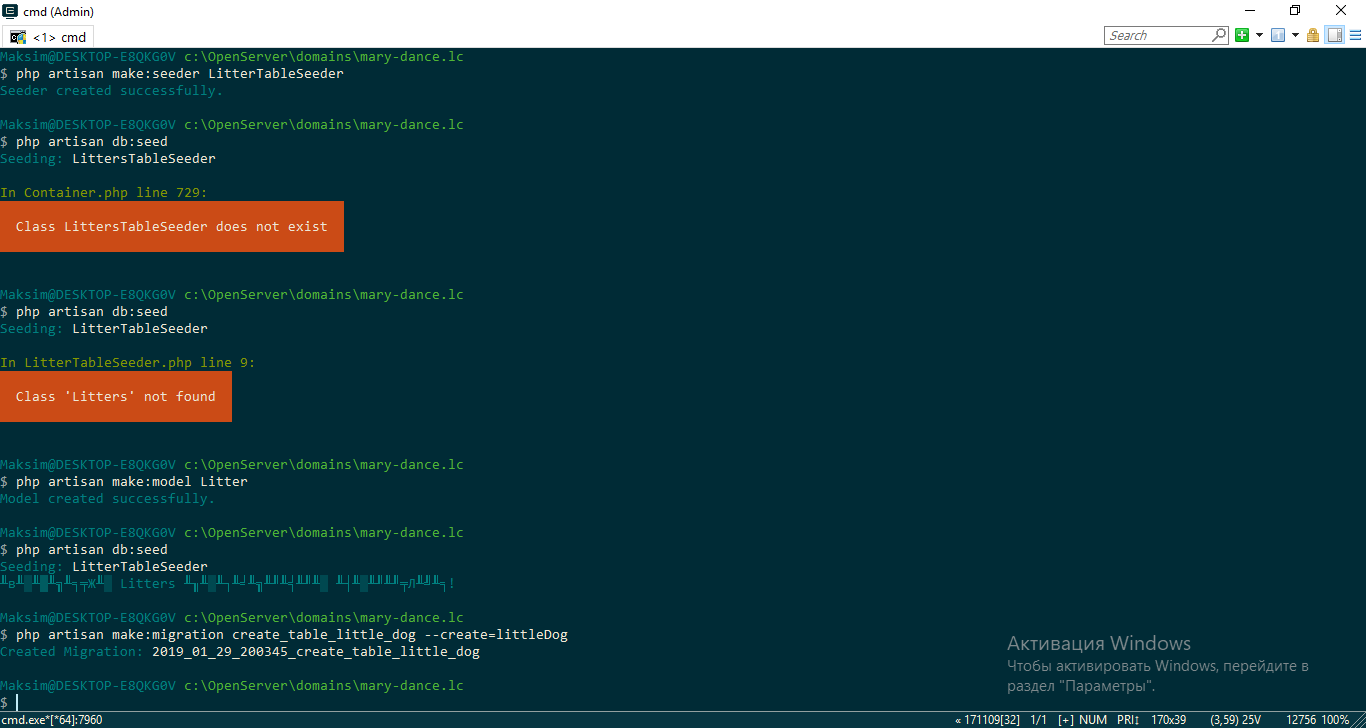
}

}

Сохраним его и запустим в командной строке:

php artisan db:seed

Это действие вызовет на исполнение класс DatabaseSeeder, в котором произойдет вызов класса LittersTableSeeder, произойдет наполнение данными, и выйдет сообщение о заполнении соответствующей таблицы. Если мы будем наполнять несколько таблиц, то необходимо создать отдельные классы для каждого действия по наполнению соответствующей таблицы, а потом указать вызовы этих классов в DatabaseSeeder в той последовательности, в которой мы хотим производить их наполнение.



<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateTableLittleDog extends Migration

{

public function up()

{

Schema::create('littleDogs', function (Blueprint $table) {

$table->increments('id');

$table->integer('idLitter', false); //false указывает, что это не автоинкримент

$table->integer(‘idPhoto,

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('littleDogs');

}

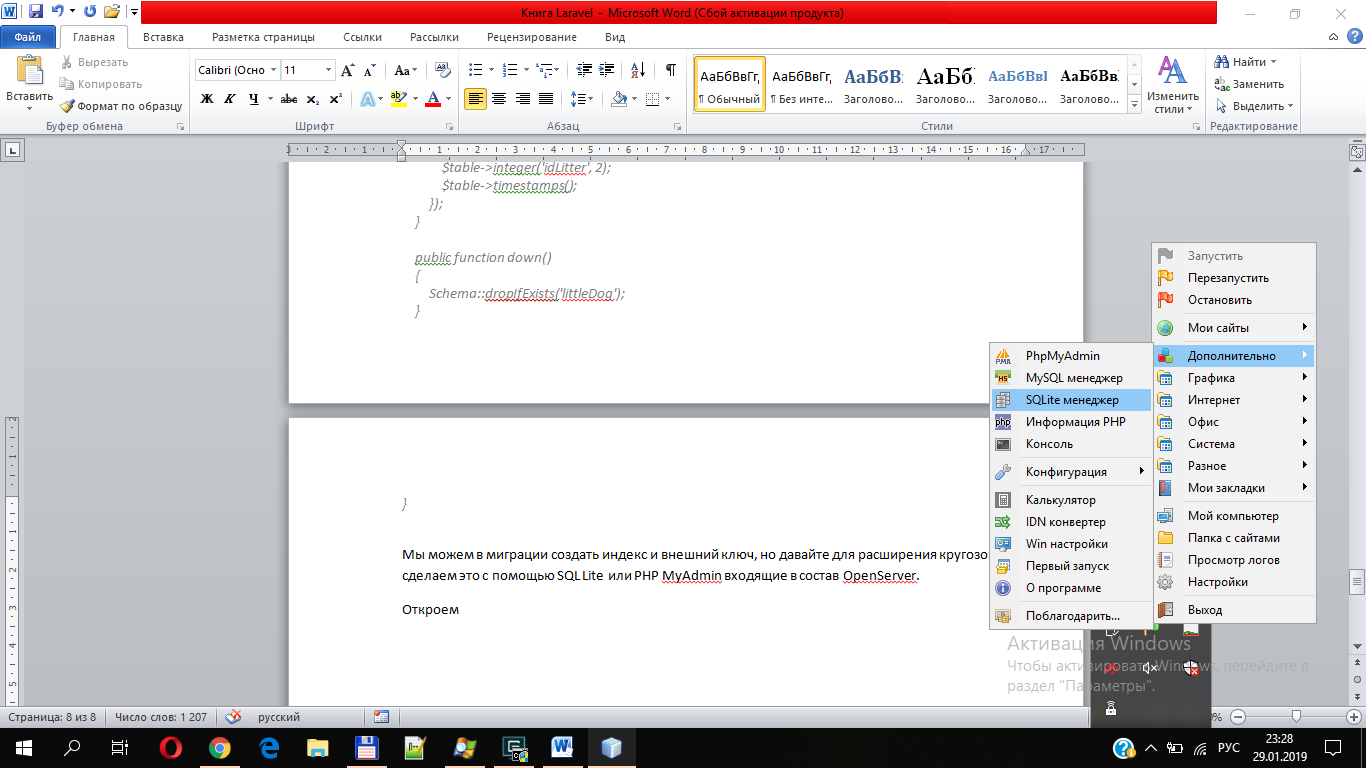
}

Создаем модель для таблицы

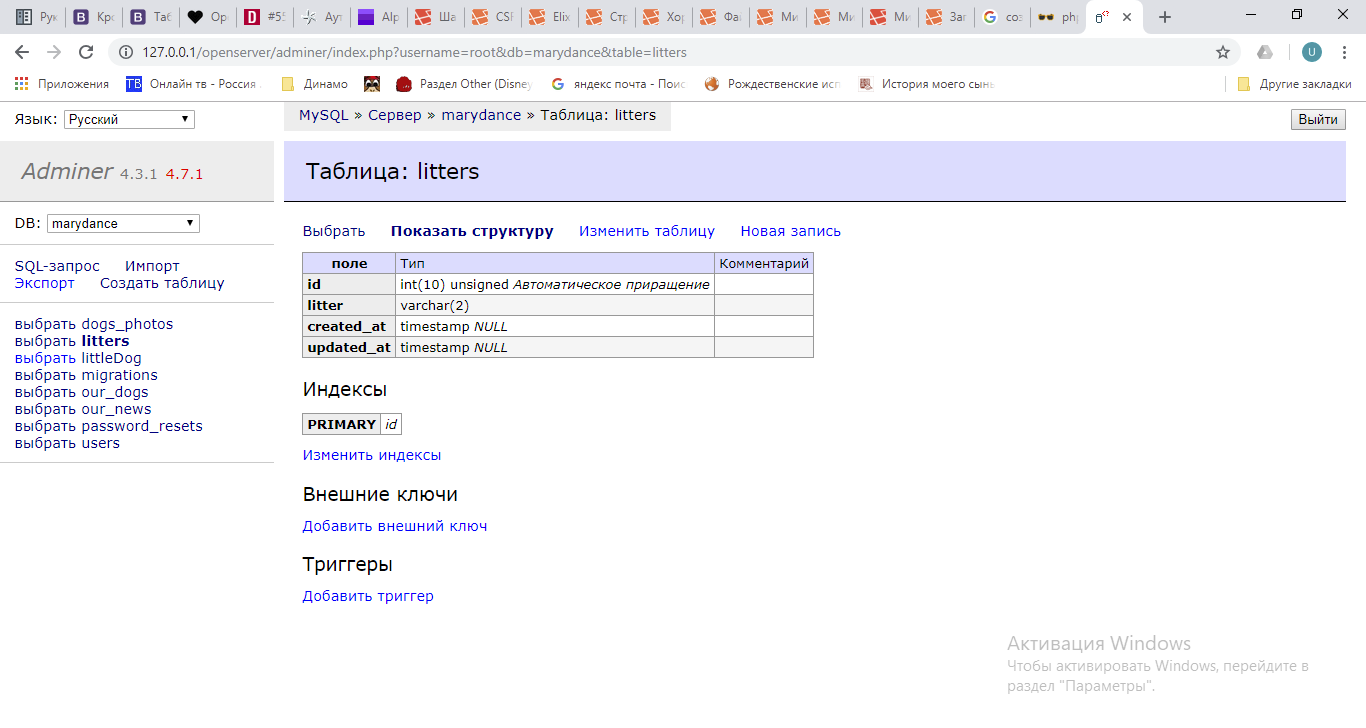
php artisan make:model littleDog

Мы можем в миграции создать индекс и внешний ключ, но давайте для расширения кругозора сделаем это с помощью SQLite или PHP MyAdmin входящие в состав OpenServer.

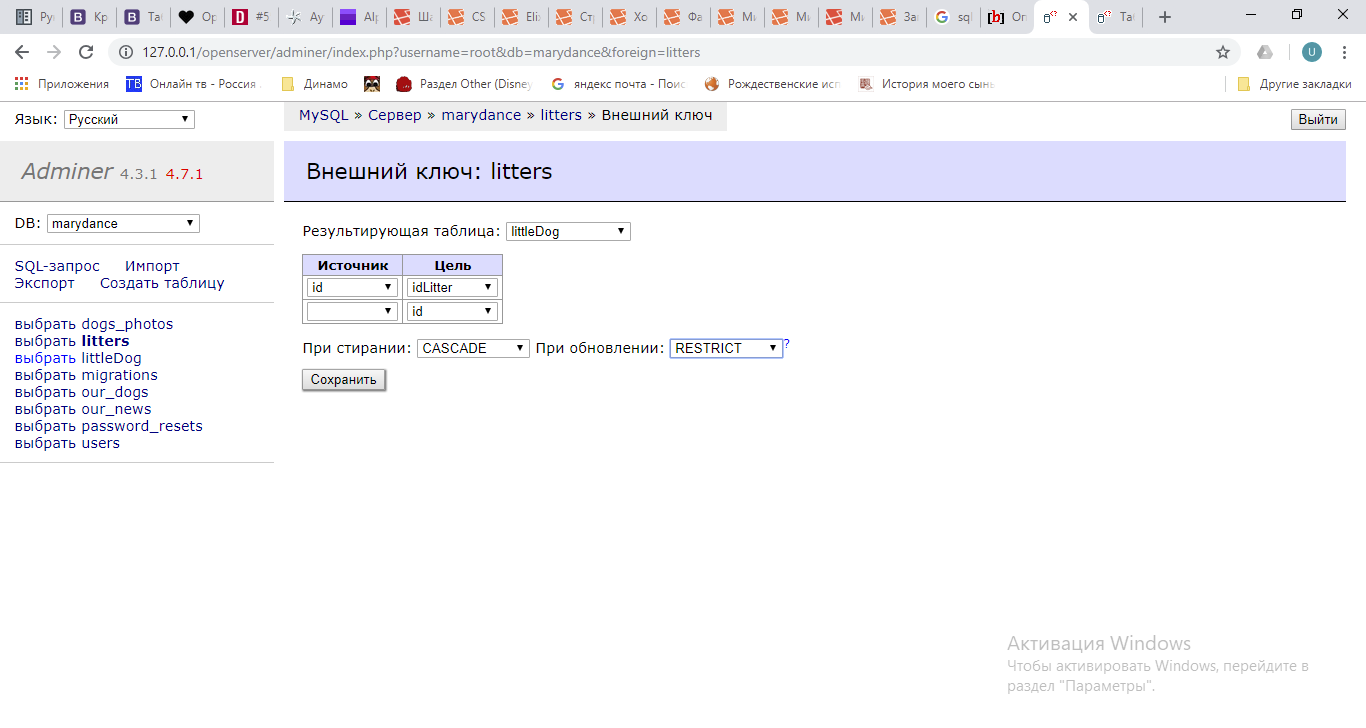
Откроем SQLite



Введя Логин/Пароль (врезка по первичной настройке OpenServer с БД – Логины и пароли по-умолчанию), выбираем нашу БД и нажимаем название таблицы



Нажимаем добавить внешний ключ и заполняем поля как на картинке



Для стирания (ON DELETE) выбрано CASCADE, т.е. удалять все связанные записи из дочерней таблицы, т.к. допустим если нам нужно удалить сам помет, то соответственно будут удалены и щенки, которые были в данном помете. Соответственно при обновлении (ON UPDATE) выбрано изменения и в соответствующих записях, чтобы не потерять целостность.