

Лабораторная работа №2

Laravel Framework

Теоретическая часть

Конфигурационный файл nginx для Laravel

```
server {  
    listen 80;  
    server_name uXXXXXX-lab2.local;  
    root /home/miet/apps/uXXXXXX-lab2/public;  
  
    location / {  
        try_files $uri /index.php?$query_string;  
    }  
  
    location = /favicon.ico { access_log off; log_not_found off; }  
  
    client_max_body_size 100m;  
  
    location ~ \.php$ {  
        fastcgi_split_path_info ^(.+\.php)(/.+)$;  
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;  
        fastcgi_index index.php;  
        include fastcgi_params;  
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
  
        fastcgi_intercept_errors off;  
        fastcgi_buffer_size 16k;  
        fastcgi_buffers 4 16k;  
        fastcgi_connect_timeout 300;
```

```
fastcgi_send_timeout 300;  
fastcgi_read_timeout 300;  
}  
}
```

Работа с Postgres в Ubuntu через консоль

Стандартный способ работы с Postgres через консоль - утилита psql.

Её необходимо запускать из-под пользователя postgres

```
sudo -u postgres psql
```

Для выхода из интерактивной сессии Postgres необходимо напечатать \q

Создание нового пользователя

```
sudo -u postgres createuser <user_name>
```

Добавление пользователю пароля:

```
sudo -u postgres psql  
psql=# alter user <user_name> with encrypted password '<password>';
```

Создание новой базы данных

```
sudo -u postgres createdb <database_name>
```

Добавление пользователю прав на доступ к базе данных:

```
sudo -u postgres psql  
psql=# grant all privileges on database <database_name> to <user_name>;
```

Практическая часть

Общие задания

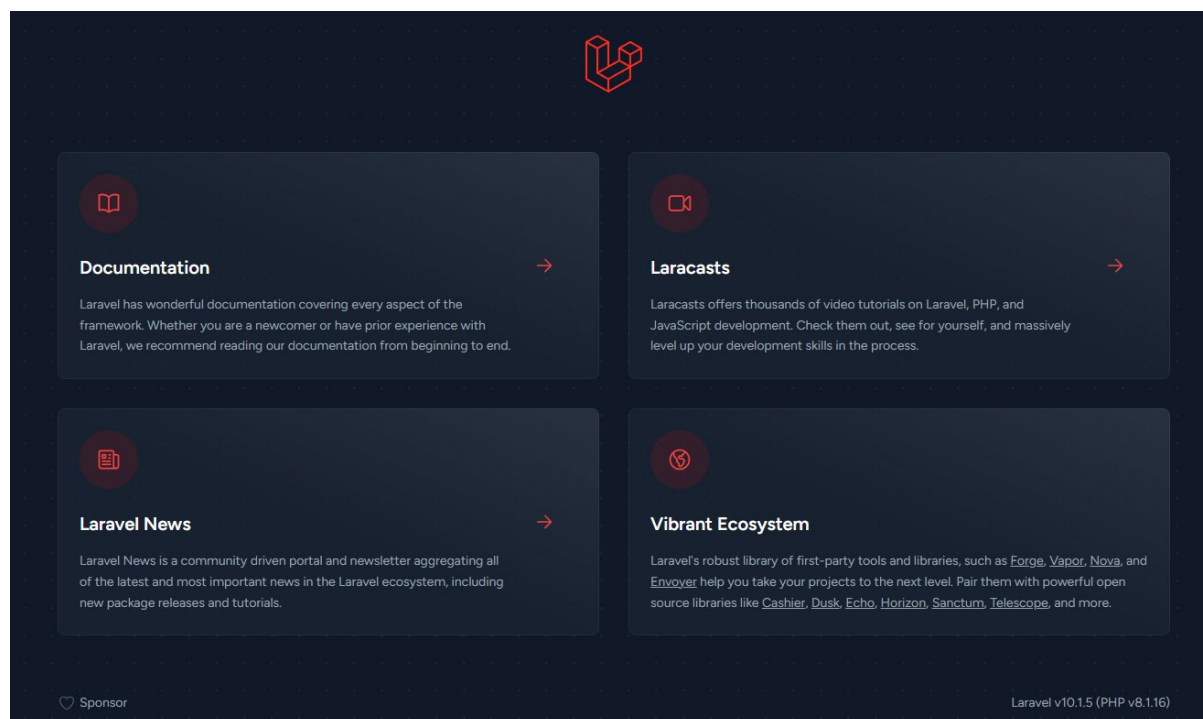
Задание 1.1 Установка Laravel

Повторите шаги “Настройка система именования DNS” и “Настройка nginx и php-fpm” из лабораторной работы №1 изменив в них uXXXXXX-lab1 на uXXXXXX-lab2 и взяв приведенный выше конфигурационный файл для nginx.

Перейдите в директорию /home/miet/apps/ и создайте новое приложение при помощи команды:

```
laravel new uXXXXXX-lab2
```

В браузере вы должны увидеть страницу Laravel по-умолчанию



Создайте новый (пустой) проект в PhpStorm / VSCode, а также создайте публичный репозиторий на gitlab.com или github.com

Загрузите код приложения в ветку master

Задание 1.2 Создание БД

Создайте пользователя с именем uXXXXXX.

Создайте базу данных по формату uXXXXXX_lab2.

Дайте созданному пользователю права на доступ к данной базе данных.

Задание 1.3 Конфигурация подключения к БД в Laravel

Укажите в файле .env корректные значения для переменных окружения DB_*.

Запустите консольную команду:

```
php artisan migrate
```

и убедитесь что в базе данных произошли изменения.

Задания по вариантам

Для выполнения дальнейших заданий вам необходимо определить ваш вариант.

№ варианта = логин % 4 + 1.

Например, если номер вашего студенческого 8123456, то получаем **8123456 % 4 + 1 = 0 + 1 = 1 вариант.**

Логин – номер учетной записи для входа в систему (*может отличаться от номера студенческого билета*).

Если вы сделали не свой вариант, то работа не принимается.

Если вы сомневаетесь какой вариант выбрать, то можете уточнить у преподавателя.

Вариант №1

Задание 2

Ваша предметная область – покупатели и их адреса. У одного покупателя может быть много адресов. Каждый адрес принадлежит лишь одному покупателю.

Поля покупателя:

- Идентификатор
- Имя
- Заблокирован (true/false)
- Фамилия
- Телефон
- Email
- Дата и время регистрации

Поля адреса:

- Идентификатор
- Название данное покупателем
- Город
- Улица/Микрорайон
- Дом
- Этаж
- Квартира
- Код домофона
- Дата и время добавления

Наличие (в БД) указанных полей является обязательным условием, дополнять список полей не запрещается.

В задании необходимо:

- Написать миграции для добавления указанных сущностей в БД
<https://laravel.com/docs/master/migrations>
- Создать модели, прописать в них связи между сущностями
<https://laravel.com/docs/master/eloquent#introduction>
- Наполнить каждую из таблиц 100 записями через Database Seeders
<https://laravel.com/docs/master/seeding>
- Проверить что данные появились в БД

Задание 3

В задании необходимо реализовать две страницы и консольную команду.

3.1 Список покупателей `uXXXXXX-lab2.local/customers`

В списке вывести все поля покупателей.

Реализовать постраничную навигацию и фильтры по полям “Заблокирован”, “Email”, “Телефон” и фильтр по имени который ищет и по имени и по фамилии.

Постраничная навигация – <https://laravel.com/docs/master/pagination>

3.2 Страница покупателя `uXXXXXX-lab2.local/customers/{id}`

На странице вывести все поля покупателя. Если пользователя с данным `{id}` не существует, отобразить 404 ошибку.

Под полями покупателя вывести блок со всеми адресами покупателя. Адреса покупателя должны быть отсортированы от более новых к более старым.

3.3 Консольная команда `php artisan customer:count-addresses {id}`

Консольная команда должна возвращать количество адресов у пользователя с идентификатором `{id}`. Если пользователя с данным `{id}` не существует, то необходимо выбросить соответствующее исключение.

Вариант №2

Задание 2

Ваша предметная область – товары и категории каталога.

Товар может принадлежать только одной категории.

В категории может лежать сколько угодно товаров.

Поля товара:

- Идентификатор
- Название
- [Символьный код](#)
- Подробное описание
- Дата и время создания
- Цена
- Изображение
- Количество единиц на складе

Поля категории:

- Идентификатор
- Название
- [Символьный код](#)
- Активность (true/false)
- Дата создания
- Родительская категория

Наличие (в БД) указанных полей является обязательным условием, дополнять список полей не запрещается.

В задании необходимо:

- Написать миграции для добавления указанных сущностей в БД
<https://laravel.com/docs/master/migrations>
- Создать модели, прописать в них связи между сущностями
<https://laravel.com/docs/master/eloquent#introduction>
- Наполнить каждую из таблиц 100 записями через Database Seeders
<https://laravel.com/docs/master/seeding>
- Проверить что данные появились в БД

Задание 3

В задании необходимо реализовать две страницы и консольную команду

3.1 Страница товара `uXXXXXX-lab2.local/products/{code}`

В списке вывести все поля товар включая изображение как ``

Если товара с **{code}** (символьным кодом) не существует, отобразить 404 ошибку.

3.2 Страница категории каталога `uXXXXXX-lab2.local/category/{code}`

Если категория не активна, то приложение должно отдавать 404 код ошибки. На странице вывести список всех товаров в данной категории.

Реализовать постраничную навигацию и фильтр по цене товара.

Постраничная навигация – <https://laravel.com/docs/master/pagination>

3.3 Консольная команда `php artisan product:category {id}`

Консольная команда должна возвращать символьный код категории у товара с идентификатором равным `{id}`. Если товара с данным `{id}` не существует, то необходимо выбросить соответствующее исключение.

Вариант №3

Задание 2

Ваша предметная область – статьи и теги. У статьи может быть несколько тэгов. Один тег может быть прикреплен к нескольким статьям.

Поля статьи:

- Идентификатор
- Название
- [Символьный код](#)
- Содержание
- Дата и время создания
- Автор

Поля тэга:

- Идентификатор
- Название
- [Символьный код](#)

Наличие (в БД) указанных полей является обязательным условием, дополнять список полей не запрещается.

В задании необходимо:

- Написать миграции для добавления указанных сущностей в БД
<https://laravel.com/docs/master/migrations>
- Создать модели, прописать в них связи между сущностями
<https://laravel.com/docs/master/eloquent#introduction>
- Наполнить каждую из таблиц 100 записями через Database Seeders
<https://laravel.com/docs/master/seeding>
- Проверить что данные появились в БД

Задание 3

В задании необходимо реализовать две страницы и консольную команду

3.1 Список статей uXXXXXX-lab2.local/posts

В списке вывести все поля статей.

Реализовать постраничную навигацию и фильтры по полям “Символьный код”, “Название”, а также фильтр, позволяющий отфильтровать статьи по тегу.

Постраничная навигация – <https://laravel.com/docs/master/pagination>

3.2 Страница статьи uXXXXXX-lab2.local/posts/{code}

На странице вывести все поля статьи. Если статьи с **{code}** (символьным кодом) не существует, отобразить 404 ошибку. Над полями статьи вывести блок с названиями всех тэгов статьи. Теги должны быть отсортированы в алфавитном порядке.

3.3 Консольная команда php artisan tag:count {id}

Консольная команда должна возвращать количество статей привязанных к тегу с идентификатором {id}. Если тега с данным {id} не существует, то необходимо выбросить соответствующее исключение.

Вариант №4

Задание 2

Ваша предметная область - категории каталога и баннеры. В каждой категории каталога может быть размещено несколько баннеров. Один и тот же баннер может быть показан в нескольких разделах каталога

Поля категории:

- Идентификатор
- Название
- Активна (true/false)
- [Символьный код](#)
- Дата создания
- Родительская категория

Поля баннера:

- Идентификатор
- Название
- Активен (true/false)
- Активен с “Дата и время”
- Активен до “Дата и время”
- Ссылка с баннера
- Файл-картинка

Наличие (в БД) указанных полей является обязательным условием, дополнять список полей не запрещается.

В задании необходимо:

- Написать миграции для добавления указанных сущностей в БД <https://laravel.com/docs/master/migrations>
- Создать модели, прописать в них связи между сущностями

<https://laravel.com/docs/master/eloquent#introduction>

- Наполнить каждую из таблиц 100 записями через Database Seeders
<https://laravel.com/docs/master/seeding>
- Проверить что данные появились в БД

Задание 3

В задании необходимо реализовать страницу и консольную команду

3.1 Страница категории каталога `uXXXXXX-lab2.local/category/{code}`

Если категория не активна, то приложение должно отдавать 404 код ошибки. На странице вывести 5 первых баннеров привязанных к категории. Баннеры должны быть отсортированы от более новых к более старых и отфильтрованы по всем трем полям активности.

Картинки баннеров должны быть показаны через `` и являться ссылками на указанный в базе данных URL

3.2 Консольная команда `php artisan banner:deactivate {id}`

Консольная команда должна деактивировать баннер с указанным `id`, т.е поменять у него в базе данных значения поля Активен с `true` на `false`. Если баннера с данным `{id}` не существует, то необходимо выбросить соответствующее исключение.

Список литературы

1. <https://postgrespro.ru/docs/postgrespro/10/app-psql>
2. <https://laravel.com/docs/master/configuration>
3. <https://laravel.com/docs/master/migrations>
4. <https://laravel.com/docs/master/seeding>
5. <https://laravel.com/docs/master/helpers#method-str-slug>
6. <https://laravel.com/docs/master/pagination>