Backend-разработчик на PHP

Современные инструменты и лучшие практики для глубокого понимания процесса разработки на PHP

Длительность курса: 154 академических часа

1 General Knowledge

1	**Подготовка к курсу**	познакомиться

2 **Виртуализация, контейнеры и облачные вычисления**

организация рабочего пространства

Домашние задания

1 Docker, виртуализация и облака

Цель: Учимся работать с инфраструктурой

1. Docker
1.1. Установить Docker себе на машину
1.2. С помощью Dockerfile настроить статический сайт (можно использовать nginx образ)

2. Выберите в качестве примера свою текущую компанию (или компанию, в которой хотите работать), коротко опишите ее (количество сотрудников, сфера, приоритеты)
Сравните целесообразность разворачивания своей инфраструктуры или аренды публичного облака (можно выбрать любого провайдера)

3 **Linux** выровнять знания в области Linux-based серверов

Домашние задания

1 Bash-скрипты

Цель: Учимся писать скрипты

1. Написать консольное приложение (bash-скрипт), который принимает два числа и выводит их сумму в стандартный вывод.
Если предоставлены неверные аргументы (для проверки на число можно использовать регулярное выражение) вывести ошибку в консоль.
2. Имеется таблица следующего вида:

id user city phone
1 test Moscow 1234123
2 test2 Saint-P 1232121
3 test3 Tver 4352124
4 test4 Milan 7990923
5 test5 Moscow 908213

Таблица хранится в текстовом файле.

Вывести на экран 3 наиболее популярных города среди пользователей системы, используя утилиты Линукса.

Подсказка: рекомендуется использовать утилиты uniq, awk, sort, head.

4 **Основы PHP**

выровнять знания языка PHP

Домашние задания

1 Готовим окружение

1. Необходимо установить любое расширение через pecl и через make (xdebug, redis)
- прислать скриншот команды pecl list, где должно значиться расширение + вывод функции `php -i | grep "ваше расширение"`
- прислать вывод команды make, т.е. `make > make_output.txt` + вывод функции `php -i | grep "ваше расширение"`
2. Необходимо создать свой пакет, и выложить в git и/или на packagist.org
- прислать команду для клонирования с гита
- прислать команду для установки через composer
3. Создать Docker-образ для работы
Необходимо создать образ, который будет включать:
- образ php, берем с https://hub.docker.com/_/php/
- необходимые утилиты (git, curl, wget, grep...)
- установленный composer
- установленные расширения redis, memcached, pecl_http, pdo_pgsql

---

5 **PHP in CLI**

научиться работать с консольным PHP

Домашние задания

1 Сокеты

Написать два PHP скрипта, который запускаются на одной машине и обмениваются сообщениями через unix-сокеты

---

6 **PHP**

выровнять знания о веб-серверах

WebServers

Домашние задания

1 Простое веб-приложение в docker

1. Используя Docker, вы описали сборку двух контейнеров – один с nginx, второй – с php-fpm и вашим кодом.
Используя docker-compose вы запускаете оба контейнера.
Контейнер с nginx пробрасывает 80 порт на вашу хостовую машину и ожидает соединений.
Клиент соединяется, и шлёт следующий HTTP-запрос:

POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 48

string=(()()()()))((((()()()))(()()()((((()))))))

String - это POST-параметр, который можно проверять:
1.1. [обязательно] На длину и непустоту
1.2. [по желанию] На корректность кол-ва открытых и закрытых скобок

Все запросы с динамическим содержимым (*.php) nginx, используя директиву fastcgi_pass, проксирует в контейнер с php-fpm и вашим кодом.
Nginx должен обрабатывать запросы не обращая внимания на директиву Host. После обработки,
• если строка корректна, то пользователю возвращается ответ 200 OK, с информационным текстом, что всё хорошо;
• если строка некорректна, то пользователю возвращается ответ 400 Bad Request, с информационным текстом, что всё плохо.

2. Развернуть на двухконтейнерном окружении (nginx+php-fpm) базовую установку Laravel. https://laravel.com/docs/5.8/installation

7	**Сети, протоколы. Балансировка**	обеспечить понимание сетевого взаимодействия и отказоустойчивости Домашние задания 1 Работа с Web-серверами 1. Приложение верификации email 1.1. Реализовать приложение (сервис/функцию) для верификации email. 1.2. Реализация будет в будущем встроена в более крупное решение. 1.3. Минимальный функционал - список строк, которые необходимо проверить на наличие валидных email. 1.4. Валидация по регулярным выражения и проверке DNS mx записи, без полноценной отправки письма-подтверждения. 2. Создать как минимум три машины/контейнера 2.1. Балансировщик nginx-upstream 2.2. Балансируемые бэкенды на nginx+php-fpm

8	**Безопасность**	обеспечение безопасности кода и приложения

Домашние задания

1 Сканер уязвимостей (OpenVAS)

Установить сканер уязвимостей OpenVAS (использовать готовые сборки или собрать самостоятельно).
Просканировать свои проекты.
Результат работы сканера в PDF в чат по ДЗ.
Если найдены уязвимости - внести исправления (код, настройки). Повторное сканирование, результат работы сканера в PDF в чат по ДЗ.
Рекомендуем использовать на регулярной основе.

Если своего проекта нет, то выбираем сайт (желательно что-то небольшое - там больше вероятность что-то найти), и строим отчёт с рекомендациями.

9	**Командная разработка**	научиться производить код коллективно

2 Databases

1 **Основные понятия баз данных**

выравниваем знания о БД

Домашние задания

1 Необходимо спроектировать схему базы данных для одного из предложенных проектов.

Спроектировать базу данных, для планировщика задач (каких угодно на ваш выбор).

---

2 **PostgreSQL для администратора**

углубляемся в аспекты работы с СУБД

---

3 **PostgreSQL для разработчика**

научиться организовывать логику на уровне данных

Домашние задания

1 Продолжаем работать с базами данных

1. Вам предоставлена Футбольная база данных группого тура чемпионата Мира по футболу 2018 года.
Придумайте и сформулируйте 10 вопросов по этой базе данных и составьте SQL запросы для нахождения ответов.
Приложите ссылку на файл с вопросами и запросами. Работы будут проверяться с обратной связью.

2. EAV-хранение для базы данных кинотеатра

---

4 **Как устроен PostgreSQL**

заглянуть под капот СУБД

Домашние задания

1 Индексы, XML и PHP.

Создать большую базу данных из XML файлов. Создать PHP скрипт для генерации данных. Создать/удалить индексы, проверить время выполнения запросов

---

5 **Другие SQL-решения**

посмотреть на рынок хранилищ

---

6 **MongoDB**

углубляемся в NoSQL

---

7 **Redis**

изучаем один из самых популярных инструментов кэширования в web

Домашние задания

1 Навыки работы с NoSQL

1. Создать приложение для анализа каналов на Youtube:
1.1. Создать структуру/структуры хранения информации о канале и видео канала в mongoDB, описать в виде JSON с указанием типов полей. Описать какие индексы понадобятся в данной структуре?
1.2. Создать необходимые модели для добавления и удаления данных из коллекций
1.3. Реализовать класс статистики, который может возвращать:
- Суммарное кол-во лайков и дизлайков для канала по всем его видео
- Топ N каналов с лучшим соотношением кол-во лайков/кол-во дизлайков

1.4*. Можно создать паука, который будет ходить по Youtube и наполнять базу данными

2. Аналитик хочет иметь систему со следующими возможностями:
2.1. Система должна хранить события, которые в последующем будут отправляться сервису событий
2.2. События характеризуются важностью (аналитик готов выставлять важность в целых числах)
2.3. События характеризуются критериями возникновения. Событие возникает только если выполнены все критерии его возникновения. Для простоты все критерии заданы так: <критерий>=<значение>

Таким образом предположим, что аналитик заносит в систему следующие события:

```
{
priority: 1000,
conditions: {
param1 = 1
},
event: {
::event::
},
},
{
priority: 2000,
conditions: {
param1 = 2,
param2 = 2
},
event: {
::event::
},
},
{
priority: 3000,
conditions: {
param1 = 1,
```

param2 = 2
},
event: {
::event::
},
},

От пользователя приходит запрос:
{
params: {
param1 = 1,
param2 = 2
}
}

Под этот запрос подходят первая и третья запись, т.к. в них обеих выполнены все условия, но приоритетнее третья, так как имеет больший priority.

Написать систему, которая будет уметь:
1) добавлять новое событие в систему хранения событий
2) очищать все доступные события
3) отвечать на запрос пользователя наиболее подходящим событием
4) использовать для хранения событий redis

объединяем код и данные

Домашние задания

1 Реализация одного из паттернов работы с хранилищем данных

Необходимо реализовать один из паттернов: Table Data Gateway, Raw Data Gateway, Active Record, DataMapper для произвольной таблицы. Паттерн должен содержать метод массового получения информации из таблицы, результат которого возвращается в виде коллекции. Дополнительно можно использовать паттерн Identity Map для устранения дублирования объектов, ссылающихся на одну строку в БД или Lazy Load для отложенной загрузки связанных записей в таблице или коллекции.

3 Developing

1	**Парадигмы программирования**	познакомиться с фундаментальными принципами построения приложений
2	**Архитектура кода**	знакомство с принципами построения взаимодействия сущностей в коде
3	**Design patterns**	узнать или улучшить понимание паттернов проектирования кода
4	**Практики хорошего кода**	чуть ближе подойти к понятию "хорошего кода" Домашние задания 1 Анализ своих проектов Выберите один из своих проектов Проведите анализ на предмет соответствия изученным принципам. Предложите свои варианты исправления.
5	**Введение в тестирование**	узнать, почему тестирование - залог крепкого сна команды разработки Домашние задания 1 None Разработанное ранее мини приложение необходимо покрыть unit-тестами, используя PHPUnit и добиться code coverage в минимум 70%

6	**Unit-тестирование**	чуть глубже рассмотрим один из аспектов автоматического тестирования
7	**Алгоритмы. Начало**	получить фундаментальные знания о классических алгоритмах
8	**Алгоритмы. Продолжение**	узнать о деревьях, графах и алгоритмах их обработки Домашние задания 1 Паттерны и алгоритмы 1. Паттерны 1.1. Спроектируйте систему классов для работы с заказами и скидками 1.2. Заказ может быть разных типов (b2b, b2c, например) 1.3. Заказ привязан к клиенту 1.4. Заказ состоит из 1 и более товаров 1.5. Товары упаковываются в 1 и более посылок 1.6. К финальной стоимости заказа могут применять различные скидки (купоны, бесплатные товары, бесплатная доставка) 1.7. У заказа есть службы доставки со своей стоимостью услуги 2. Алгоритмы 2.1. Реализуйте алгоритм хранения Nested Sets

4 Architecture & HighLoad

1	**Профилирование и логирование**	учимся анализировать работу приложения
2	**Очереди**	изучить основной инструмент работы асинхронных отказоустойчивых приложений - очереди Домашние задания 1 API Используя мини-приложение, разработанное в прошлом модуле, необходимо реализовать Rest API с использованием очередей. Ваши клиенты будут отправлять запросы на обработку, а вы будете складывать их в очередь и возвращать номер запроса. В фоновом режиме вы будете обрабатывать запросы, а ваши клиенты периодически, используя номер запроса, будут проверять статус его обработки.
3	**Проектирование API**	узнать, что такое API и зачем он нужен
4	**Репликация**	изучить важный инструмент обеспечения отказоустойчивости БД
5	**Шардинг**	усиливаем отказоустойчивость хранилищ
6	**Кеширование**	усилить знания о кэшировании в Highload-системах

7	**Deploying**	познакомиться с понятиями деплоя и сборок Домашние задания 1 Скрипт деплоя Используя выбранный инструмент автоматического деплоя, необходимо реализовать автоматическую выкатку написанного ранее мини-приложения на собственный виртуальный сервер.
8	**Site Reliability Engineering**	узнать о best practices обеспечения производительности и отказоустойчивости

5 Проектный модуль

1	**Консультация по проектам**	Обсуждение тем проектов, вопросы и ответы Домашние задания 1Проект
2	**Презентация проектов**	Подведение итогов курса и презентация реализованных проектов