

Backend разработчик на PHP

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

Продолжительность

5 месяцев, 4 часа в неделю

Начало занятий

27 мая

Длительность курса: 186 академических часов

1 General Knowledge

1 Подготовка к курсу

* обсудить цели курса, коротко обсудить рассматриваемые темы

* обсудить историю развития веб-приложений, состояние на текущий день

* ввести общую терминологию

* обсудить организационные вопросы, порядок сдачи и проверки домашних работ

2 **Виртуализация, контейнеры и облачные вычисления**

- * поговорить про виртуализацию и контейнеры
- * рассмотреть облачных провайдеров
- * рассмотреть экономику работы с публичным облаком, своим датацентром и т.д.
- * научиться работать с докером

Домашние задания

1 Провести экономическое сравнение Приватного и Публичного облаков

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)

2 Основы работы с Docker

- * Установить Docker себе на машину
 - * С помощью Dockerfile настроить статический сайт (можно использовать nginx образ)
-

3 Linux

- * объяснить актуальность Linux систем (почему сервера в основном * работает на этой ОС)
 - * рассмотреть процессы, потоки
 - * обсудить пользователей, привилегии
 - * научиться использовать ряд полезных утилит (top, grep etc)
- научиться использовать bash скрипты

Домашние задания

1 Консольная команда Sum

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

2 Анализ таблицы пользователей

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`.

- * рассмотреть историю языка
- * изучить организацию памяти и garbage collector
- * обсудить актуальность экосистемы
- * рассмотреть менеджеры пакетов

Домашние задания

1 Установить расширение двумя способами

Необходимо установить любое расширение через `pecl` и через `make`.

- 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 Создание собственного расширения

Необходимо создать расширение, и выложить в `git` и 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`
-

Цель занятия:

- * обсудить лучшие практики создания скриптов
- * научиться демонизировать скрипты
- * изучить способы запускать скрипты по расписанию (cron, supervisord и т.д.)
- * обсудить IPC

Домашние задания

1 [Всеm] Калькулятор по шаблону. Strategy.
Покрyть его PHPUnit тестами.

2 [Вариант 1] Сокеты

Два PHP-шных скрипта, запущенных на одной машине обмениваются сообщениями через сокеты

3 [Вариант 2] Утилита для исправления опечаток

Существует утилита The fuck
(<https://github.com/nvbn/thefuck>)

Написать аналог на php для одной команды (например git commit)

- * обсудить принципы построения веб-серверов
- * рассмотреть FastCGI
- * рассмотреть php-fpm
- * изучить nginx
- * популярные фреймворки

Домашние задания

1 Простое веб-приложение в docker

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=(((())()))((((())())())())()
```

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 ОК, с информационным текстом, что всё хорошо;
 - если строка некорректна, то пользователю возвращается ответ 400 Bad Request, с информационным текстом, что всё плохо.
-

7 Сети, протоколы. Балансировка

- * рассмотреть сетевой стек
- * поговорить о HTTP и HTTPS протоколах
- * обсудить балансировку, рассмотреть решения на разных уровнях сетевого стека (L4, DNS и т.д.)

Домашние задания

1 приложение верификации email

Реализовать приложение (сервис/функцию) для верификации email.

Реализация будет в будущем встроена в более крупное решение.

Минимальный функционал - список строк, которые необходимо проверить на наличие валидных email.

Валидация по регулярным выражения и проверке DNS mx записи, без полноценной отправки письма-подтверждения.

8 Безопасность

* рассмотреть симметричные и ассиметричные алгоритмы шифрования AES, RSA, Blowfish. Digest-алгоритмы sha и md5.

* детально рассмотреть TOP10 видов уязвимостей web-приложений по OWASP (в частности SQL-injections, XSS, CSRF).

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.

9 Командна разработка

* обсудить методологии разработки ПО

* обсудить роли в команде

* подробно изучить SCRUM

Домашние задания

1 Клиент для вашей библиотеки должен работать по протоколу HTTP.

Для этого, используя docker compose, вы создадите два контейнера, один с nginx, а второй с php-fpm и приложением, использующим библиотеку.

1 Основные понятия баз данных

Обсудим модели данных и ранние подходы к организации данных, в частности, иерархические и сетевые базы данных. Научимся описывать концептуальные схемы предметной области при помощи ER-модели. Остановимся на реляционной модели и погрузимся в реляционную алгебру. Поговорим о SQL, его истории, стандартах и совместимости.

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

Полное погружение в PostgreSQL. Поговорим об администрации кластера, ролях, атрибутах, привилегиях, схемах, табличных пространствах и системном каталоге. Для всего этого изучим DDL.

Домашние задания

- 1 Практическая отработка навыков с вебинаров
-

3 PostgreSQL для разработчика

Продолжим изучение PostgreSQL, но уже в качестве клиентского разработчика. Изучим DML, поговорим о типах данных, функциях и операторах. Узнаем как устроены индексы и работают транзакции. Обсудим ACID, MVCC и уровни изоляции.

Домашние задания

1 Футбольная база данных

Вам предоставлена Футбольная база данных группового 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.

4 Как устроен PostgreSQL

Перестанем бояться чудодейственной магии PostgreSQL и детально разберём как база данных работает “под капотом”. В этом занятии будет буферный кеш, журнал упреждающей записи, контрольная точка, страницы и версии строк, LRU, снимки и блокировки, а также Vacuum. Используя EXPLAIN, посмотрим как PostgreSQL выполняет запрос и попытаемся оптимизировать его выполнение.

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

Посмотрим на другие SQL-решения, в частности, сделаем детальный обзор возможностей MySQL и SQLite. Поговорим о колоночных базах данных на примере ClickHouse.

6 MongoDB

Рассмотрим not only SQL-решения на примере MongoDB. Познакомимся с CRUD операциями. Поговорим о Aggregation Pipeline и MapReduce. Поработаем с MongoDB из кода на PHP.

Домашние задания

1 Приложение для анализа каналов на Youtube

Создать приложение для анализа каналов на Youtube:

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?
 2. Создать необходимые модели для добавления и удаления данных из коллекций
 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
 - 4*. Можно создать паука, который будет ходить по Youtube и наполнять базу данными
-

7 Redis

Поговорим о Redis как о базе данных. Изучим типы данных и способы работы с ними из кода на PHP. Сравним Redis с Memcached.

Домашние задания

1 Система хранения событий для аналитики

Аналитик хочет иметь систему со следующими возможностями:

1) Система должна хранить события, которые в последующем будут отправляться сервису событий

2) События характеризуются важностью (аналитик готов выставлять важность в целых числах)

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
-

Изучим все способы работы кода на PHP с изученными базами данных. Как дань истории будут показаны устаревшие драйверы, но остановимся на PDO. Рассмотрим ООП-подход для работы с базами данных. Научимся реализовать и применять такие паттерны как DAO, ActiveRecord, ORM, ODM. Поговорим об их плюсах и минусах.

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.

- 1 Парадигмы программирования**

Когда мы пишем код - мы придерживаемся какой-то парадигмы. В этом занятии мы обсудим различные парадигмы программирования и увидим, что не ООП-единым, на примере использования функциональной парадигмы. Посмотрим на функции высшего порядка, каррирование, замыкания и монады. После чего детально остановимся на ООП.

- 2 Архитектура кода**

Погрузимся в архитектуру кода. UML, SOLID, SoC, DRY, KISS, YAGNI, DI и DI-контейнеры.

- 3 Design patterns**

Рассмотрим часто встречающиеся проблемы при проектировании ООП-программ и, как способ их решения, шаблоны проектирования. Обсудим порождающие, структурные, поведенческие шаблоны, а также коснёмся шаблонов GRASP.

- 4 Практики хорошего кода**

Поговорим о том, как писать хороший код, о принципах CQRS и Fluent interface. Обсудим coding styles и необходимость документирования кода. Рассмотрим PHP the Right way и стандарты из PHP-FIG.

5 Введение в тестирование

Поговорим о тестировании - его видах и какие проблемы призван решить каждый вид. В частности, обсудим acceptance, integration и unit тестирование. Узнаем, что такое test case и как он должен выглядеть. Научимся писать интеграционные тесты на codeception.

Домашние задания

1 None

Разработанное ранее мини приложение необходимо покрыть unit-тестами, используя PHPUnit и добиться code coverage в минимум 70%

6 Unit-тестирование

Поговорим о том, что такое тестируемый код и как его писать. Научимся писать Unit-тесты с использованием PHPUnit. Поговорим об A-TRIP, TDD и Red-Green-Refactor. Рассмотрим идеологии CI/CD и запустим автоматический прогон наших тестов в Travis.

7 Алгоритмы. Начало

Поговорим об алгоритмах и структурах данных. Детально рассмотрим асимптотический анализ. Рассмотрим алгоритмы сортировки, в частности: сортировка Шелла, быстрая сортировка и сортировка слиянием. Изучим стек и очередь на примере реализаций из SPL. Детально рассмотрим связанные списки и способы их обхода.

Домашние задания

1 Необходимо реализовать один из предложенных алгоритмов на деревьях.

8 **Алгоритмы.** **Продолжение**

Продолжим говорить об алгоритмах. Рассмотрим такие структуры данных как двоичные и сбалансированные деревья поиска. Обсудим хеш-таблицы и способы борьбы с коллизиями. Закончим алгоритмами на графах - поиск в ширину и алгоритм Дейкстры.

1 Очереди

Рассмотрим асинхронный подход обработки данных на основе очередей. Разберём несколько стандартных сценариев использования очередей (отправка уведомлений, инвалидация кеша). Реализуем работу с очередями, используя различные инструменты (очереди на базе, Redis Pub/Sub, Gearman, Beanstalkd). Детально изучим протокол AMQP и одну из его прикладных реализаций - RabbitMQ.

Домашние задания

1 None

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.

2 Проектирование API

Научимся проектировать API для web и mobile используя Rest и RPC-протоколы. Обсудим JSON, XML, Protocol Buffers. Детально остановимся на Rest и способе его описания, используя RAML. Получим представление о WebSockets.

- 3 **Профилирование и логирование** Скрипт тормозит? Научимся находить узкие места, используя инструменты профилирования. Также детально обсудим логирование, чтобы понимать что делает наш скрипт. Затронем уровни логирования (по PSR-3), библиотеку Monolog и сбор логов в ELK.
-
- 4 **Репликация** Получим полное представление о репликации: о её видах (Master-Slave, Master-Master), о способе синхронизации изменений (sync, async), о формате изменений (SBR, RBR), о модели передачи изменений (push, pull) и о том, как с этим работать на уровне PHP кода.
-
- 5 **Шардинг** Поймём, что такое шардинг и когда его стоит применять. Обсудим виды шардинга (горизонтальный и вертикальный). Поговорим о ребалансировке и решардинге. Затронем партиционирование.
-
- 6 **Кеширование** Поговорим зачем приложению нужен кеш. Рассмотрим Redis и Memcached в качестве кеш-серверов. Поговорим о кеш-тегах и инвалидации кеша.
-

7 Deploying

Обсудим возможные способы доставки вашего приложения в production-окружение - от ручного git pull до инструментов автоматизации этого процесса в лице (mina и capistrano).

Домашние задания

1 None

Используя выбранный инструмент автоматического деплоя, необходимо реализовать автоматическую выкатку написанного ранее мини-приложения на собственный виртуальный сервер.

8 Site Reliability Engineering

Цель занятия:

- * принципы построения отказоустойчивых систем
- * SLAs, SIs, SLOs
- * лучшие практики

5 Проектный модуль

1 Консультация по проектам

Обсуждение тем проектов, вопросы и ответы

Домашние задания

1 Проект

2 Презентация проектов

Подведение итогов курса и презентация реализованных проектов