

# Содержание

---

01 Знакомство	2
02.1 Массивы и циклы	3
02.2 Массивы и циклы	4
03.1 Шаблонизация и организация кода	5
03.2 Шаблонизация и организация кода	7
04.1 HTTP и формы	8
04.2 HTTP и формы	9
05.1 Хранение состояния	10
05.2 Хранение состояния	11
06.1 Базы данных_ теория	12
06.2 Базы данных_ теория	13
07.1 MySQL и PHP	14
07.2 MySQL и PHP	15
08.1 Продвинутые возможно	16
08.2 Продвинутые возможно	17
09.1 Чужой код	18
09.2 Чужой код	19

## Наименование и описание системы

YetiCave – это интернет аукцион для продажи сноубордического и горнолыжного снаряжения. Зарегистрированный пользователь может создать свои лоты на продажу вещей на условиях аукциона. Покупатель, предложивший максимальную цену в установленные сроки, получает эту вещь.

### Предыстория

YetiCave позволяет войти на сайт под своим именем. После входа пользователь видит своё имя и аватар в шапке сайта. Формы аутентификации и регистрации мы пока делать не умеем, поэтому данные пользователя уже будут находиться в сценарии. Вам остается лишь вставить их в нужное место.

### Подготовка

1. Вся работа происходит в файле `index.php`
2. Откройте файл `index.php` и напишите условие на показ имени и аватара пользователя

### Как должно работать условие

1. В начале скрипта определена переменная `$is_auth`. Её значение может быть `true` (истина) или `false` (ложь)
2. В HTML коде найти тег `<nav class="user-menu">` внутри которого поместить условие
3. Условие должно проверять истинность значения переменной `$is_auth` (сравнивать с истиной)
4. Если условие истинно, необходимо показать следующий html код:

```
<div class="user-menu__image">
    <img src="" width="40" height="40" alt="Пользователь">
</div>
<div class="user-menu__logged">
    <p></p>
</div>
```

где в теге `<img>` в атрибут `src` подставить значение переменной `$user_avatar`, а в тег `<p>` поместить значение переменной `$user_name`

5. Если условие ложно, то вместо информации о пользователе показать ссылки на вход и регистрацию, использовав этот код:

```
<ul class="user-menu__list">
    <li class="user-menu__item">
        <a href="#">Регистрация</a>
    </li>
    <li class="user-menu__item">
        <a href="#">Вход</a>
    </li>
</ul>
```

### Как проверить результат

Открыть в браузере сайт и несколько раз обновить страницу. При каждом обновлении переменная `$is_auth` будет случайным образом иметь значение `true` или `false`, а значит в шапке сайта должен отображаться либо аватар пользователя с его именем, либо ссылки на вход и регистрацию.

## Предыстория

На площадке YetiCave пользователи публикуют объявления о продаже самых разных видов сноубордического снаряжения: крепления, одежда, доски и многое другое. Чтобы облегчить покупателю поиск нужной ему вещи, каждое объявление назначается одной из существующих категорий.

Вам следует вывести в HTML из подготовленных массивов список категорий, а также список из шести объявлений.

## Список шагов

- Добавьте в код файла `index.php` простой массив категорий со следующими значениями: «Доски и лыжи», «Крепления», «Ботинки», «Одежда», «Инструменты», «Разное»
- Добавьте в код файла `index.php` двумерный массив, каждый элемент которого будет ассоциативным массивом с данными одного объявления. Информацию для этих массивов взять из следующей таблицы:

Название	Категория	Цена	URL картинки
2014 Rossignol District Snowboard	Доски и лыжи	10999	img/lot-1.jpg
DC Ply Mens 2016/2017 Snowboard	Доски и лыжи	159999	img/lot-2.jpg
Крепления Union Contact Pro 2015 года размер L/XL	Крепления	8000	img/lot-3.jpg
Ботинки для сноуборда DC Mutiny Charocal	Ботинки	10999	img/lot-4.jpg
Куртка для сноуборда DC Mutiny Charocal	Одежда	7500	img/lot-5.jpg
Маска Oakley Canopy	Разное	5400	img/lot-6.jpg

- Найдите в HTML-коде список `ul.nav_list`. Используйте цикл, чтобы заменить содержимое списка данными из массива категорий./li>
- Объявления выводятся как элементы списка `ul.lots_list`. Используя цикл для прохода по массиву с лотами, заполните этот список объявлениями. Каждый элемент списка (объявление) должно подставлять информацию в теги из соответствующего ключа массива.

## Как проверить результат

Откройте в браузере сайт. Главная страница должна показывать шесть объявлений с данными из таблицы, а список категорий в футере будет заполнен данными из задания.

## Предыстория

Все товары на сайте YetiCave продаются по принципу аукциона. Владелец вещи только назначает минимальную цену, а затем потенциальные покупатели назначают свои ставки. Автор самой последней ставки на момент окончания аукциона становится победителем и получает право купить товар из лота по этой цене.

В нижней части карточки объявления указана начальная цена лота, заданная его автором. Для удобства пользователя эта цена выводится в формате с делением на разряды и добавлением знака рубля.

Карточка объявления на сайте показывается в двух местах: список объявлений (на главной странице) и на странице просмотра объявления.

### Ваша задача

Напишите функцию для форматирования суммы и добавления к ней знака рубля. Затем используйте эту функцию для показа цены в карточке объявлений на главной странице.

### Список шагов

1. Напишите функцию форматирования суммы, используя спецификацию ниже
2. Используйте функцию, чтобы показать цену каждого из объявлений

### Спецификация функции

Функция должна принимать один аргумент – целое число.

Функция должна возвращать результат – отформатированную сумму вместе со знаком рубля.

Как должна работать функция:

1. Округлить число до целого используя функцию `ceil()`
2. Если переданное число меньше `1000`, то оставить его как есть
3. Если число больше `1000`, то отделить пробелом три последних цифры от остальной части суммы  
Пример: заменить `54999` на `54 999`
4. Добавить к получившейся строке пробел и знак рубля – ₽

### Как отформатировать число

У вас есть два способа как отформатировать число, чтобы добавить деление на разряды.

1. Используйте функцию `number_format`
2. Либо вы можете самостоятельно перевести число в строку, разделить её на две части и склеить в новую строку, добавив пробел между двумя частями. В этом вам помогут:
  - Функция `substr`
  - Произвольный доступ к символу в строке

В верстке карточки объявления замените содержимое тега `span.lot__cost` вызовом вашей функции.

## Предыстория

Шаблонизация – это преобразование готовой HTML-верстки страниц сайта (статика) в специально подготовленные шаблоны. Шаблоны – это PHP веб-сценарии, которые содержат только HTML-код с включениями PHP-переменных и простые условные конструкции.

Шаблоны нужны для двух целей:

- избавление от дублирования HTML-кода;
- отделение бизнес-логики веб-сайта от его вида (представления).

Страницы любых многостраничных веб-сайтов состоят из повторяющихся частей: шапка, подвал, боковое меню и т. д. Чтобы не дублировать HTML-код этих элементов на каждой новой странице, имеет смысл выделить каждый такой элемент в отдельный файл-шаблон. Это позволит редактировать содержимое шаблона в одном месте, а его обновленный контент будет показан на всех страницах, которые его включают.

Бизнес-логикой в веб-разработке называют программный код, который ответственен за обработку данных, но не за их представление. Работа с данными массивов, подсчет дат, обработка форм – это примеры бизнес-логики сайта. Включение PHP-переменных в HTML-коде, форматирование строк – это примеры представления.

Использование шаблонов позволяет отделить бизнес-логику от представления, что упростит поддержку кода и сделает его более масштабируемым. Работа с данными будет происходить в одном месте, а их показ – в другом. Для выполнения этого задания вам потребуется сделать две вещи: перенести HTML-код в PHP-шаблоны и написать функцию, которая будет подключать эти шаблоны и передавать в них данные (опционально).

## Из чего состоит задание

1. Создать лейаут
2. Создать шаблон для главной страницы
3. Написать функцию-шаблонизатор
4. Убрать из index.php весь HTML код, заменив вызовами шаблонизатора и показа результата

## Список шагов

1. Создайте в корне проекта папку `templates`.
2. Создайте там два php-файла: `layout.php`, `index.php`
3. В `layout.php` поместите весь HTML код из текущего index.php со следующими правками:
  - в теге `title` должна быть переменная, в которой будет имя страницы
  - на место содержимого тега `main` поставьте вывод переменной, в которой будет контент страницы
4. В файл `templates/index.php` поместите бывшее содержимое тега `main` из `./index.php`
5. В файле `index.php` удалите весь HTML-код, который вы перенесли в файлы шаблонов.
6. Создайте в корне файл `functions.php` и подключите его в `index.php`.
7. В файле `functions.php` напишите функцию-шаблонизатор по спецификации.
8. Используйте эту функцию для включения всех шаблонов в `index.php`:
  - в начале включите шаблон страницы, передав туда необходимые данные (списки категорий/лотов/ задач)
  - затем включите лейаут и передайте туда содержимое шаблона страницы, полученное из пред. шага, а также имя пользователя, имя страницы
  - содержимое лейаута нужно показать на странице (`print`)
9. считайте, что содержимое главной страницы (список лотов или список задач) получено от пользователя, поэтому выводите их соответствующим образом (с фильтрацией)

## Спецификация функции

Функция должна принимать два аргумента: путь к файлу шаблона и массив с данными для этого шаблона. Функция должна возвращать строку – итоговый HTML-код с подставленными данными.

Как должна работать функция:

1. Проверять, что файл шаблона, переданный в аргументе, существует. Если не существует, то функция

- должна вернуть пустую строку.
2. Функция должна подключить файл шаблона и использовать буферизацию вывода для захвата его содержимого.
  3. Итоговое содержимое шаблона вернуть как результат работы функции.

#### **Как проверить результат**

После выполнения этого задания во внешнем виде сайта ничего не должно поменяться. То есть должна сохраниться вся функциональность, которую вы создавали в прошлых заданиях.

## Общие сведения

В этом задании мы будем с помощью арифметики и стандартных функций работать с датой и временем.

## Подготовка

Для вычисления разницы между двумя датами (сколько осталось дней до даты, сколько прошло с определенной даты, и т.д.) очень удобно работать с «временными метками».

Временная метка (далее просто «TS» для краткости) – это количество секунд, прошедших с 1 января 1970 по текущее время.

### Пример как посчитать количество дней до нового года (2019)

Чтобы выполнить этот расчет, следует знать два значения: TS от 01 января 2019 года и TS настоящего дня:

TS Нового Года: 1514754000

TS Текущий: 1491843137

Зная оба значения, остается вычесть одно из другого. Результатом будет количество секунд до нового года.  
Осталось перевести секунды в дни.

Полезно знать:

- 1 минута = 60 секунд
- 1 час = 60 минут = 3600 секунд
- 1 сутки = 24 часа = 1440 минут = 86400 секунд

Формула для перевода секунд в дни: секунды / количество секунд в сутках

Итоговая формула для получения дней до НГ: (TS НГ - TS текущий) / 86400

Конечное решение:  $(1514754000 - 1491843137) / 86400 = 265$  дней

## Стандартные функции для работы с временными метками в PHP

Стандартные функции в PHP – это такие маленькие помощники, предоставляемые языком для выполнения разнообразных задач.

В предыдущем примере было необходимо получить временные метки для двух дат: будущей и текущей.

Самостоятельный расчет временной метки для даты – это очень сложная задача. К счастью, эту заботу можно полностью переложить на PHP.

- Функция для получения временной метки для даты из будущего или прошлого: `strtotime($format)`  
Эта функция возвращает метку для переданной ей в произвольном формате даты.  
Пример: `$ny_date = strtotime("01.01.2018")`
- Функция для округления числа: `floor($number)`  
Эта функция округляет переданное ей число в меньшую сторону до целого.  
Пример: `$number = floor(2.54); // 2`

YetiCave

Дела в Порядке

## Наименование и описание системы

Все лоты, публикуемые на сайте YetiCave «живут» на сайте ограниченное время. После истечения этого времени лот скрывается с сайта. Количество часов и минут, оставшихся до истечения, указаны в карточке лота под ценой.

## Постановка задачи

Пока, для простоты, будем считать, что каждый лот живет только до полуночи (как в сказке про золушку :). Необходимо написать код, который посчитает сколько часов и минут осталось до новых суток.

## Задача

1. Установите часовую зону в московское время
2. Требуется написать выражение, которое по уже знакомой формуле посчитает количество часов и минут до новых суток
3. Полученные часы и минуты в формате "Ч:М" показать в блоке под ценой лота `div.lot_timer`
4. Обновить страницу и убедиться, что под лотами теперь указано корректное время до начала суток

## Общая информация

Строка запроса – это адрес текущей страницы в браузере, который состоит из домена сайта, пути к странице и параметров запроса.

Параметры запроса – это как бы набор переменных, которые будут доступны на странице. Благодаря параметрам запроса можно показывать на странице разную информацию, в зависимости от переданных параметров.

YetiCave

Дела в Порядке

## Предыстория

Страница `lot.php` содержит подробную информацию о лоте: полное описание, текущую цену, историю ставок. Попасть сюда можно со списка последних лотов на главной странице. Ссылка на страницу лота состоит из имени страницы – `lot.php`, а также специального параметра запроса – уникального идентификатора лота.

Существование этого параметра во время загрузки скрипта `lot.php` позволяет понять, о каком из лотов надо показать информацию.

Вам предстоит сформировать правильные ссылки на лоты на главной странице, а также показывать на странице `lot.php` информацию только по лоту, чей идентификатор был передан в параметрах запроса.

## Необходимые действия

- Перенесите массив с лотами в отдельный PHP-сценарий и подключите его на страницах `index.php` и `lot.php`.
- На страницу `index.php` у вас выводится список лотов из массива. Необходимо добавить каждому лоту правильную ссылку на страницу просмотра. Сформируйте ссылку, которая будет состоять из имени файла (`lot.php`) и параметра запроса, равного идентификатору лота. Идентификатором лота будем считать его индекс в массиве.
- На странице `lot.php` определять переданный в параметрах идентификатор и получать по нему соответствующий лот из массива.
- Заменить в блоке `section class="lot-item"` всю информацию о лоте на данные из массива.
- Если переданный идентификатор не присутствует в массиве (соответствующий индекс отсутствует), то вместо содержимого страницы возвращать код ответа 404.

## Как проверить результат

Кликая по объявлениям на главной странице, вы должны переходить на страницу просмотра объявления, где данные лота будут соответствовать выбранному на главной странице. Если в ссылке на объявление подставить несуществующий идентификатор, то должна вернуться ошибка 404.

## Предыстория

Чтобы добавить на сайт новое объявление пользователь должен заполнить форму. В этой форме он описывает свой товар, указывает начальную цену и добавляет фотографию. После отправки формы проверяется корректность её заполнения: все обязательные поля должны быть заполнены, а форматы данных соответствовать заданным. В этом задании вам предстоит работать с данными из формы: получать данные из полей и проверять их корректность, а затем показывать отправленную информацию на странице.

**Пока информация из формы будет только показана на странице после отправки. Сохранять данные мы научимся на следующих вебинарах**

## Необходимые действия

1. Создать новый PHP-сценарий: `add.php`.
2. Создать шаблон для этого сценария, куда перенести верстку формы из файла `pages/add-lot.html`.
3. Все поля в форме должны иметь имена (атрибут `name`), а action формы указывать на `add.php`.
4. После отправки формы проверять, что были заполнены все поля, а также значения полей «начальная цена» и «шаг ставки»: туда можно вводить только цифры.
5. Если проверка формы выявила ошибки, то сделать следующее:
  - в тег формы добавить класс `form--invalid`;
  - для всех полей формы, где найдены ошибки:
    - добавить контейнеру с этим полем класс `form__item--invalid`;
    - в тег `span.form__error` этого контейнера записать текст ошибки. Например: «Заполните это поле».
6. Загруженный файл изображения переместите в папку `img`.
7. Если проверка формы прошла успешно, то на странице `add.php` вместо формы должен быть показан шаблон с карточкой товара, который будет содержать данные из формы, включая изображение товара

## Как проверить результат

Попробуйте отправить пустую форму: после отправки вы должны увидеть все поля формы, подсвеченные красным с описанием ошибок. Введенные в поля формы данные должны там остаться. Введите вместо цифр слова в поля для цены и ставки: это тоже должно считаться, как ошибка. Если форма заполнена без ошибок, то после её отправки вы увидите содержимое страницы: объявления с данными, полученными из формы.

## Описание процесса аутентификации

Процесс аутентификации состоит в сравнении переданных пользователем данных (email, пароль) с хранящимися на сервере.

Алгоритм сравнения:

- Среди существующих пользователей найти юзера с переданным email.
- Если такой пользователь есть, то сравнить пароли, а точнее, их хеши. Пароли не хранятся в открытом виде, вместо них хранятся отпечатки – хеши. Функция `password_verify` умеет сравнивать хеш и пароль. Используйте её, чтобы убедиться в корректности пароля.

Если пользователь передал верные данные (такой email существует и пароль корректный), то необходимо его идентифицировать: открыть новую сессию, в которой будут данные об этом пользователе.

В новом файле `userdata.php` есть массив с пользователями. Используйте его для выполнения аутентификации.

### Таблица паролей для пользователей

Email	Пароль
ignat.v@gmail.com	ugOGdVMI
kitty_93@li.ru	daecNazD
warrior07@mail.ru	oixb3aL8

YetiCave

Дела в Порядке

### Предыстория

Сайт YetiCave делит всех посетителей на две группы: анонимные и прошедшие аутентификацию.

Анонимные пользователи могут просматривать списки объявлений и сами объявления, но не могут добавлять новые лоты и участвовать в торгах. Чтобы получить такую возможность они должны пройти аутентификацию.

Аутентификация – это проверка подлинности предъявленных пользователем данных. В нашем случае такие данные – это логин и пароль, которые пользователь должен ввести в форме, чтобы получить доступ к дополнительным возможностям.

В этом задании вы реализуете весь процесс аутентификации: от показа формы до проверки пароля и разрешения доступа пользователю к закрытым страницам.

### Общий план работы над заданием

- Сделать страницу с формой входа.
- Написать код для валидации данных формы.
- Написать код для аутентификации пользователя.
- Показывать имя пользователя в шапке.
- Сделать страницу, которая разлогинит пользователя.
- Разделить показ информации на сайте для анонимных и залогиненных пользователей.

### Памятка по созданию новых страниц

Помните, что теперь весь HTML-код в ваших сценариях должен быть оформлен в виде PHP-шаблонов.

Каждая новая страница сайта будет состоять из двух PHP-файлов:

- файл сценария (например, `add.php`): содержит только бизнес-логику и подключение шаблонов;
- файл шаблона (например, `templates/add.php`): содержит только HTML-код с PHP-переменными и условными конструкциями.

Файл сценария производит действия по подготовке информации, подключает все шаблоны и передаёт им нужные данные.

### Список шагов

- Создайте новую страницу с формой входа. Вёрстку для формы нужно взять из файла `pages/login.html`.
- При отправке формы проверяйте заполненность двух полей. Если поле не было заполнено, то следует показать ошибку (см. прошлое задание).
- Если форма прошла валидацию, то выполнить процесс аутентификации и переадресовать пользователя на главную страницу. Если пользователь ввёл неверный пароль, то показать ошибку внутри контейнера с полем «Пароль» и текстом «Вы ввели неверный пароль».
- В шапке сайта показывать имя пользователя, если есть открытая сессия, иначе показывать ссылки «Регистрация» и «Вход». Ссылка «Вход» должна вести на страницу входа.
- На странице `lot.php` скрывать блок добавления ставки (`body > main > section > div > div.lot-item_right > div.lot-item_state`) для анонимных пользователей.
- Добавить новый сценарий: `logout.php`. Сценарий должен обнулять пользовательскую сессию и переадресовывать на главную страницу. Добавить адрес этой страницы в ссылку «Выход» под именем пользователя.
- Закрыть доступ к странице `add.php` для анонимных пользователей. При попытке обращения к этой странице анонимному пользователю должен возвращаться HTTP-код ответа 403.

### Как проверить

- Открыть главную страницу сайта.
- Перейти на страницу любого лота. Блок добавления ставки должен отсутствовать.
- Попробовать открыть страницу `/add.php`. Вы должны увидеть ошибку «403».
- Справа от формы поиска кликнуть на ссылку «Вход». Должна открыться форма аутентификации.
- Отправить пустую форму. Ожидаем увидеть выделение полей и текст ошибки о неверном пароле.
- Отправить форму, введя случайные значения. Ожидаем увидеть выделение полей и текст ошибки о неверном пароле.
- Указать в форме существующий email и подходящий пароль. Ожидаем перехода на главную страницу после отправки.
- В шапке сайта имя аутентифицированного пользователя и ссылка «Выход».
- Перейти на страницу любого лота. Блок добавления ставки должен присутствовать.
- Кликнуть по ссылке «Выход». Должны перейти на главную страницу, вместо аватара и имени пользователя в шапке сайта видим ссылки «Регистрация» и «Вход».

## Как хранить сложную информацию в cookies

Файлы cookies могут хранить только простые текстовые значения, поэтому для хранения композитных типов данных (массивов) их следует предварительно сериализовать.

Сериализация – это преобразование массива в строку с возможностью выполнения обратной операции (десериализации). В PHP для этого есть стандартные функции: `json_encode($arr)` – для сериализации и `json_decode($str)` – для десериализации.

Мы рекомендуем использовать их при необходимости сохранить в cookie массив.

YetiCave

Дела в Порядке

### Предыстория

Для удобства пользователя, все просмотренные им лоты можно увидеть на отдельной странице.

Ваша задача сохранять каждый просмотренный лот в cookie, чтобы потом, на отдельной странице, показать пользователю все просмотренные им лоты одним списком.

### Список шагов

1. Создайте новый сценарий и шаблон для страницы «История просмотров». Назовите его `history.php`.  
Вёрстка для шаблона в `pages/all-lots.html`.
2. В верстке этого шаблона замените текст тега `h2` на «История просмотров»
3. В сценарии `lot.php` добавьте код, который будет записывать в cookie индекс открытого лота

#### Важно!

Вы должны хранить в куке список из просмотренных индексов лотов. Используйте сериализацию, чтобы поместить в cookie массив. Также следите за тем, чтобы в этом массиве не было повторяющихся элементов.

4. В сценарии `history.php` из cookie получите список просмотренных лотов
5. По индексам просмотренных лотов получите сами лоты из существующего массива и покажите их на странице

### Как проверить

1. Войти на сайт
2. Перейти на страницу просмотра любого лота
3. Открыть адрес `/history.php`
4. Увидеть карточку одного лота, просмотренного в п.2

## Как проектируется схема БД

Разработка нового проекта начинается с проектирования схемы базы данных. Схема определяет список таблиц для хранения данных вашего проекта и связи между ними.

Понять какие таблицы нужны вашему проекту поможет его представление в виде набора сущностей.

Сущность – это неделимый объект предметной области, имеющий атрибуты. За редкими исключениями каждой сущности будет соответствовать одна таблица. Примеры сущностей: пользователь, проект, задача, категория.

Выделение сущностей из предметной области вашего проекта – это самый важный этап всего проектирования, потому что он определяет насколько удачной получится итоговая схема БД.

Ваш личный проект имеет [техническое задание](#), которое уже содержит описание всех сущностей, их атрибутов и отношений. Поэтому вам остаётся лишь перенести их на язык SQL, но в реальной работе это придётся делать самостоятельно.

Определение состава самих таблиц – следующий этап проектирования.

Все поля таблицы можно поделить на две группы: смысловые и логические. Смысловые поля хранят непосредственно сами данные, а логические поля нужны для организации связей между таблицами: внешние и первичные ключи. Нельзя забывать и о выборе правильного типа данных для каждого поля: оно должно соответствовать сути хранимой информации, а размерность поля должна быть достаточной и не быть избыточной.

Последним шагом назначьте полям индексы. Поля для хранения уникальных значений должны иметь индекс типа «**UNIQUE**». Поля, по которым часто будут проходить выборки также должны быть с индексом.

## Подготовка

1. Создайте новый файл в корне проекта с именем `schema.sql`.
2. В этом файле напишите весь SQL-код для создания схемы БД вашего проекта.

## Требования

- весь SQL должен без ошибок выполняться на MySQL 5.7;
- в каждой таблице должен быть первичный ключ.

## План работы

1. Сперва напишите SQL для создания всех необходимых таблиц.
2. Добавьте уникальные индексы полям, где должны быть только уникальные значения.
3. Добавьте обычные индексы полям, по которым будет происходить поиск.

## Как проверить

1. Запустите HeidiSQL (для Windows) или командную строку MySQL.
2. Создайте новую базу данных с названием вашего проекта.
3. Попробуйте выполнить ваш файл `schema.sql`.
4. Файл должен выполниться без ошибок.

## Описание

Задание состоит из двух частей:

1. заполнить БД данными, взяв их из существующих массивов
2. написать запросы для манипуляции этими данными

Сегодня вы напишите SQL-запросы для выполнения основных действий: вставка данных, чтение и поиск, обновление. В дальнейшем вы используете эти запросы в своём PHP-коде для интеграции с базой данных. Запросы должны работать с таблицами, которые вы сделали в прошлом задании.

1. Создайте новый файл в корне проекта с именем `queries.sql`.
2. В начале файла напишите запросы типа `INSERT` для добавления в БД всех необходимых данных
3. Ниже этого файла напишите SQL-код всех запросов на выборку данных, каждый с новой строчки.
4. Каждый запрос предваряйте комментарием с названием действия, для которого он предназначен.

YetiCave

Дела в Порядке

**Напишите запросы для добавления информации в БД:**

- Существующий список категорий
- Существующий список пользователей
- Список объявлений
- Добавьте пару ставок для любого объявления

**Не забудьте про связи между этими таблицами. Например, в объявлении должна быть ссылка на пользователя и категорию**

**Напишите запросы для этих действий:**

- получить все категории;
- получить самые новые, **открытые** лоты. Каждый лот должен включать название, стартовую цену, ссылку на изображение, цену, количество ставок, название категории;
- показать лот по его `id`. Получите также название категории, к которой принадлежит лот
- обновить название лота по его идентификатору;
- получить список самых свежих ставок для лота по его идентификатору;

## Описание

В этом задании вы добавите на сайт форму регистрации и оживите её. С помощью этой формы пользователи будут заводить на сайте себе аккаунт. В число необходимых действий в этом задании входит валидация формы и сохранение полученной из неё информации в базе данных. Работать с формами вы уже умеете, поэтому новым здесь будет только взаимодействие с MySQL.

YetiCave

Дела в Порядке

## Предыстория

Добавление новых лотов доступно только зарегистрированным пользователям. Регистрация нужна, чтобы у каждого лота была информация о его хозяине – имя, контактные данные, а авторы объявлений получали уведомления о ходе торгов.

## Необходимые действия

Вся необходимые детали по выполнению этого задания есть в спецификации:

- [3.1 Процесс "Регистрация аккаунта"](#)
- [5.1 Страница "Регистрация аккаунта"](#)
- [4.1 Процесс "Авторизация на сайте"](#)

Также вам потребуется внести изменения в сценарий для аутентификации: искать пользователей теперь нужно не в массиве, а по записям из таблицы пользователей.

Обобщённо список шагов для выполнения выглядит так:

1. Создать новый сценарий и шаблон для страницы регистрации
2. Написать код валидации формы и показа ошибок
3. Настроить подключение к MySQL
4. В сценарии регистрации написать код, который сохранит в БД данные из формы
5. В сценарии с аутентификацией производите поиск пользователей не по массиву, а по таблице из БД

## Как проверить результат

В шапке сайта есть ссылка "Регистрация". Эта ссылка должна работать и вести на форму регистрации нового аккаунта. Если корректно заполнить форму, то пользователя переадресует на главную страницу, а в БД появится новая запись. Далее, используя форму авторизации, можно будет войти на сайт, используя данные только что созданного пользователя: email и пароль.

## Описание

На сайтах часто встречаются различные списки. Учебный проект [Giftube](#) на своих страницах показывает следующие виды списков:

- перечень категорий в левом меню
- список гифок на главной странице
- список комментариев на странице просмотра гифки

Ваша задача в этом задании – научиться получать из MySQL необходимые записи и показывать их на странице. Вы уже умеете выполнять итерацию по двумерным массивам и показать их содержимое в виде отдельных элементов. Но до настоящего времени вы работали уже с готовыми массивами, а теперь вам придется наполнять их содержимым из БД.

YetiCave

Дела в Порядке

## Предыстория

На главной странице показаны карточки девяти новых лотов. Это лоты, у которых не истек срок их публикации, отсортированные от самых новых к старым. В прошлом задании вы уже написали необходимый SQL-запрос для получения таких записей. Вам необходимо лишь заменить существующий массив с лотами на данные, полученные из MySQL по этому запросу.

## Необходимые действия

- В сценарии главной странице выполните подключение к MySQL
- Отправьте SQL-запрос для получения всей информации по новым лотам
- Используйте эти данные для показа списка карточек лотов на главной странице

## Как проверить результат

В прошлом задании вы перенесли массив со списком лотов в БД. Т.к. там теперь точно такая же информация, то внешний вид главной страницы никак не должен измениться. Но чтобы проверить, что все работает корректно, добавьте руками в таблицу лотов новую запись, указав в качестве даты публикации текущее время. Затем обновите главную страницу и убедитесь, что в списке появилась новая карточка и она первая.

## Описание

Полнотекстовый поиск – это возможность MySQL, которая позволяет искать записи в таблице по нестрогому совпадению текста. Используйте его, когда необходимо добавить на сайт поиск по отдельным сущностям с текстовым наполнением.

YetiCave

Дела в Порядке

## Задание

Оживите форму поиска в шапке сайта. Эта форма будет искать лоты по их названию или описанию. Найденные лоты будут показаны на отдельной странице поиска.

### Список действий:

1. В schema.sql добавьте SQL инструкцию на создание полнотекстового индекса для полей "название" и "описание" в таблице лотов
2. Форма поиска должна вести на файл search.php и работать по методу GET
3. Сделайте новый шаблон для страницы поиска (верстка в pages/search.html)
4. Добавьте новый сценарий search.php, который будет искать лоты и показывать результат в шаблоне
5. Если по запросу нет результатов, то на их месте должна быть надпись "Ничего не найдено по вашему запросу"

## Проверка

1. введите название одного из существующих лотов в форму поиска и нажмите enter
2. должны перейти на страницу search.php, где будет показан лот с этим именем
3. в поле формы поиска должен быть текст заданного поискового запроса

Это последнее обязательное задание на нашем интенсиве, поэтому придется постараться.

Вы получили все знания, необходимые для выполнения обязательных пунктов технического задания. А значит осталась самая малость – добавить в личный проект еще пару-тройку возможностей. Вы не должны столкнуться со сложностями, потому что подобные задачи уже выполнялись вами в прошлых заданиях. Нужно будет просто добавить еще одну страницу, обработать еще одну форму, написать еще пару SQL-запросов и так далее.

**При работе над заданием руководствуйтесь в первую очередь спецификацией!**

YetiCave

Дела в Порядке

## Описание

Доведем нашу доску объявлений до готового к работе сайта. Для этого выполните следующие части из ТЗ:

- **3.2 Публикация нового лота**

Форма у вас уже работает. Осталось добавить сохранение информации в базе данных.

- **4.2 Добавление ставки**

Нужно оживить форму и сохранять информацию в базе данных.

- **5.6 Просмотр лота** Вся информация на этой странице должна выводиться из базы данных

## Как проверить

Попробуйте выполнить следующий список действий:

1. Завести новый аккаунт
2. Войти на сайт под новым аккаунтом
3. Добавить новое объявление
4. Добавить ставку к чужому объявлению
5. Перейти на страницу просмотра своего объявления

## Что такое Composer

Composer – это пакетный менеджер для PHP. Composer умеет самостоятельно скачивать и устанавливать библиотеки вместе с их зависимостями. В мире JavaScript и верстки прямым аналогом является NPM - Node Package Manager.

Работа с Composer происходит через командную строку, вызовами команд

```
composer имя_действия
```

## Что такое composer.json

**composer.json** – это конфигурационный файл, в котором composer хранит информацию о вашем проекте и списке установленных библиотек. Каждый раз, когда вы устанавливаете новую библиотеку, её имя и версия записывается в этот файл.

В дальнейшем, при переносе проекта на хостинг, composer сможет автоматически установить на новом месте все библиотеки, взяв их список из composer.json.

## Задание

В этом задании вы установите Composer и выполните его инициализацию в вашем проекте. По итогу работы у вас появится файл composer.json с заполненной информацией и новая папка – vendor.

## Список шагов

1. Установите [Composer](#) у себя на компьютере.
2. Инициализируйте Composer в вашем проекте. Должен появиться файл `composer.json` с информацией о вашем проекте.
3. Подключите во всех сценариях страниц файл `vendor/autoload.php`.
4. Добавьте директорию `vendor` в исключения гита: `.gitignore`

## Как отправить e-mail из сценария

В PHP очень просто отправлять e-mail сообщения, для этого есть даже специальная функция - `mail()`. Но использовать её не надо, т.к. `mail()` справляется с отправкой почты крайне плохо. Не поддерживает SMTP, вложенные файлы и многое другое.

Чтобы отправить электронное письмо по всем правилам, лучше прибегнуть к помощи сторонних библиотек.

## Что такое SwiftMailer

**SwiftMailer** – это самая популярная и качественная библиотека для отправки электронной почты из PHP. SwiftMailer хорошо документирована и проста в использовании. Для начала работы с ней вам потребуется знать параметры подключения по SMTP к почтовому серверу (они предоставлены в задании).

## Параметры подключения к SMTP

Используйте SMTP Transport для отправки сообщений и следующие параметры подключения к почтовому серверу:

Имя пользователя: doingsdone@gmail.com

Пароль: rds7BgcL

SMTP сервер: smtp.mail.ru

Port: 465

Шифрование: ssl

YetiCave

Дела в Порядке

## Задание

В ТЗ на ваш проект есть описание процесса "Определение победителя". Вам следует его реализовать и использовать SwiftMailer для отправки победителю email-сообщения с поздравлением.

## Список шагов

- Установите через Composer библиотеку SwiftMailer
- Создайте новый сценарий – `getwinner.php` и подключите его в `index.php`
- Напишите в этом сценарии всю логику процесса "Определение победителя" из ТЗ
- Создайте новый шаблон – `email.php` и заполните его следующим контентом:

```
<h1>Поздравляем с победой!</h1>
<p>Здравствуйте, [имя пользователя]</p>
<p>Ваша ставка для лота <a href="[ссылка на лот]">[имя лота]</a> победила.</p>
<p>Перейдите по ссылке <a href="[ссылка на страницу мои ставки]">мои ставки</a>, чтобы связаться с автором объявления</p>

<small>Интернет Аукцион "YetiCave"</small>
```

- Отправьте сообщение победителю. Используйте в качестве содержимого письма результат работы шаблона.

Остальные параметры перечислены ниже:

### Имя параметра

Тема письма

Отправитель

Получатель

Content-type тела письма

### Значение

Ваша ставка победила

doingsdone@gmail.com

E-mail пользователя-победителя

text/html