OTUS

Web-разработчик на Python

Полный набор Fullstack навыков, с которыми вы сможете создавать сложные web-сайты и решать задачи уровня Middle+

Длительность курса: 174 академических часа

1 Основы python и модульного тестирования

1 **Знакомство с курсом. Введение в HTTP. Работа с библиотеками requests и BeautifulSoup**

Цели занятия:

понимать работу HTTP;
отправлять запросы с помощью requests;
использовать BeautifulSoup для парсинга html-страниц.

Краткое содержание:

основы взаимодействия на курсе;
основы HTTP;
библиотека requests;
библиотека BeautifulSoup;

Домашние задания

1 Парсер ссылок с сайта

Цель: В этой самостоятельной работе тренируем умения:
1. Отправлять запросы с помощью requests
2. Использовать BeautifulSoup для парсинга страницы

Чтобы:
Лучше понимать как работает HTTP и разогреться для дальнейшего использования python

Задача:
Создать программу парсер (консольную), которая будет получать с сайта все ссылки ведущие на другие сайты

0. Выбрать любой сайт на котором есть хотя бы одна ссылка на другие сайты (html-тэг a)

Написать парсер, который:
1. будет отправлять запрос на этот сайт
2. получать с него все ссылки на другие сайты (которые находятся в href-атрибуте html-тэга a)
3. выводить полученные ссылки в терминал

дополнительно:
4. для каждой полученной ссылки повторять процедуру (отправлять запрос на этот адрес, получать все связанные ссылки и выводить в терминал)
5. добавить возможность выбора либо выводить результат в терминал, либо сохранять его в файл

6. Сдать дз в виде ссылки на репозиторий

---

2 **Основы ООП. Принципы ООП**

Цели занятия:

создавать классы;
создавать объекты классов;
создавать свойства и методы класса;
создавать методы доступа;
объяснить как реализованы принципы ООП в python.

Краткое содержание:

что такое ооп, основные принципы ооп;
создание классов и объектов в python;
как реализованы основные принципы ооп в python;
getter, setter.

практическое задание: написание программы с помощью классов.

---

3 **ООП. Магические методы, утиная типизация, статические методы, методы класса**

Цели занятия:

писать код с применением основных магических методов и утиной типизации;
объяснить, что такое магические методы и зачем они нужны;
объяснить, для чего нужны статические методы и методы класса и как их писать.

Краткое содержание:

зачем нужны магические методы;
основные магические методы и зачем нужен каждый из них;
утиная типизация;
зачем нужны статические методы и методы класса.

практическое задание: рефакторинг кода с магическими методами.

Домашние задания

1 Написать игру "Лото"

Цель: В этой самостоятельной работе тренируем умения:

1. разбивать программу на классы
2. у класса создавать свойства, методы, инициализатор
3. создавать объекты класса
4. создавать методы доступа
5. использовать магические методы

Эти умения нам понадобятся при написании программ на ООП

1. Создать новый проект "Игра лото"

2. Правила игры можно скачать тут: https://gist.github.com/DanteOnline/038d59cb9c5b704d02238f4e11b95c97

3. Написать игру лото

Возможные подходы к решению задачи:
1) Проектирование на основании предметной области. Подумать какие объекты есть в игре и какие из них можно перенести в программу. Для них создать классы с соответствующими свойствами и методами. Проверить каждый класс отдельно. Написать программу с помощью этих классов

2) Метод грубой силы + рефакторинг. Написать программу как получится. После этого с помощью принципа DRY убрать дублирование в коде

3) Процедурное программирование

4. Минимальные требования: 2 игрока - человек играет с компьютером
5. (Дополнительно *) возможность выбирать тип обоих игроков (компьютер или человек) таким образом чтобы можно было играть: компьютер - человек, человек - человек, компьютер - компьютер
6. (Дополнительно *) возможность играть для любого количества игроков от 2 и более

7. Сдать дз в виде ссылки на репозиторий

4 **Введение в автотесты. pytest**

Цели занятия:

писать тесты для функций и классов на pytest;
запускать тесты;
объяснить, что такое автоматизированное тестирование;
осознать зачем нужно автоматизированное тестирование;
запомнить плюсы и минусы библиотеки pytest;
объяснить для чего нужны методы setup, teardown.

Краткое содержание:

введение в автотесты;
библиотека pytest;
демонстрация написания тестов.

практическое задание: написание тестов для готовой функции
Setup, teardown, fixtures.

Домашние задания

1 Покрыть предыдущие дз тестами

Цель: В этой самостоятельной работе тренируем умения:
1. пользоваться библиотекой pytest
2. писать тесты для имеющихся функций и методов
3. запускать тесты
Для того чтобы проверять код программы автоматически, писать чистый код, создавать оптимальную архитектуру программы

1. В проекте "Лото" написать тесты
2. Если написать тесты не удается, попробуйте разбить программу так, чтобы в ней было больше чистых функций
3. В качестве тренировки можно попробовать написать тесты для парсера из 1-го дз
4. Сдать дз в качестве ссылки на репозиторий с проектом

2 Создаем свой блог. База данных и ORM. Flask. MVC. Docker

1 **Введение в docker, docker-compose**

Цели занятия:

объяснить, что такое docker и для чего он нужен;
проанализировать плюсы и минусы docker;
разобраться с примерами настроек docker-контейнера;
собирать docker-контейнер для django проекта
проанализировать основные команды docker, понять как они работают

Краткое содержание:

docker;
плюсы и минусы docker;
примеры настроек докер контейнера;
как собирать докер-контейнер для django проекта;
основные команды docker, показать как они работают

---

2 **ORM, SQLAlchemy**

Цели занятия:

создавать модели данных с помощью SQLAlchemy для заданной предметной области;
объяснить, что такое ORM, для чего он используется;
объяснить как делать основные запросы в базу данных с помощью ORM.

Краткое содержание:

введение в ORM;
SQLAlchemy;
как создавать модели данных;
основные запросы.

практическое задание: создание моделей данных для некоторой предметной области.

Домашние задания

1 Создание моделей данных для сайта "Мой блог" на выбранную тему

Цель: В этой самостоятельной работе тренируем умения:
1. Создавать модели данных

2. Создавать связи между моделями
3. Работать с сессией
4. Делать простые запросы
Смысл:
Для того чтобы работать с SQLAlchemy в проектах с базой данных. Понимать как работать с orm

Создать модели Post, Tag для сайта "Мой блог" на тему (ваша тема). Для пользователя можно использовать стандартную модель User.
Установить связи между моделями.
Добавить некоторые данные.
Выбрать все посты конкретного пользователя с 2-мя любыми тегами

1. Создать новый проект "Мой блог", по нему будет 3 домашних задания. Рекомендуется создать для этого проекта отдельный репозиторий
2. Придумать тему блога. Она может быть любая какая вам более интересна (например экзотические птицы, занятия workout-ом, искусство, ...)
3. С помощью SQLAlchemy создать модели данных для блога, например (Post, User, ...) и все другие, которые вы считаете важными
4. Установить связи между моделями
5. В качестве примера ввести некоторые данные
6. Выбрать все посты конкретного пользователя, попробовать сделать другие запросы (Рекомендуется сделать это в виде тестов pytest, можно просто с помощью print)
7. Сдать дз в виде ссылки на репозиторий

3 **Знакомство с Front-end частью курса. Основы HTML, CSS, методологии верстки. Немного Bootstrap 4**

Цели занятия:

писать css селекторы;
запомнить устройство http, web, rest;
осознать назначение кодов ответа;
объяснить как связаны html, css, js и из чего они состоят;
проанализировать, какие есть способы разработки css;
объяснить как пользоваться bootstrap4.

Краткое содержание:

http, web;
rest;
http коды;
введение в html, js, css;
селекторы css;
упражнение на css;
методологии разработки css;
Bootstrap 4.

Домашние задания

1 Сделать верстку для сайта "Мой блог"

Цель:
В этой самостоятельной работе мы тренируем умения
1. htm
2. css
3. js

1. В проекте "Мой блог" создать папку templates
Пока сделаем статические страницы, позже подключим flask
2. Создать следующие страницы и переходы между ними: главная страница, все посты (они могут быть сразу на главной), 1 пост, контакты.
3. Создать любые другие страницы которые вы считаете нужными
4. В зависимости от выбранной темы создать дизайн для страниц
Можно использовать bootstrap, можно самим написать css, можно использовать любой другой способ

4 **Введение в werkzeug; Flask**

Цели занятия:

запустить тестовый сервер на Flask;
проанализировать как связаны view и шаблоны;
объяснить как работает шаблонизатор, что это такое;
объяснить зачем нужны Blueprint и как их использовать;
создать небольшой проект на Flask.

Краткое содержание:

где используется Flask фреймворк;
запуск проекта на Flask;
передача данных из view в шаблон;
шаблонизатор и рендеринг шаблонов;
работа с Blueprint;
создание небольшого сайта на Flask.

практическое задание: добавление на получившийся сайт еще несколько страниц и переходов.

---

5 **Werkzeug; Flask + SQLAlchemy. Работа с моделями данных**

Цели занятия:

добавлять модели и базу данных в проект на Flask;
проанализировать паттерн MVC и зачем он нужен;
настроить Flask для работы с SQLAlchemy;
объяснить как сохранять и получать данные.

Краткое содержание:

MVC модель;
как подружить Flask и SQL Alchemy;
настройки проекта Flask+Alchemy;
добавление моделей данных к сайту;
как сохранять и получать данные.

практическое задание: добавить еще несколько модель на сайт и вывести данные на странице.

---

6 **Связь контейнеров в docker. Сборка проекта на Flask**

Цели занятия:

понять, как собирать проект из нескольких контейнеров и docker-compose;
собирать проект на flask в докере;
связывать контейнеры друг с другом

Краткое содержание:

принцип "один процесс - один контейнер";
как связывать контейнеры;
как создать отдельные контейнеры для flask, бд, wsgi

Домашние задания

1 Сделать сайт "Мой блог" на Flask + SQLAlchemy

Цель: В этой самостоятельной работе мы тренируем умения:
1) Работать с SQLAlchemy из Flask
2) Использовать MVT паттерн
3) Использовать docker

Для того чтобы:
Создавать небольшые сайты на Flask. Соединять вместе модели, базу данных, view и шаблоны

1) Заканчиваем мини проект "Мой блог"
2) Собираем все вместе: (базу, view, шаблоны и дизайн)
3) Хорошо будет добавить регистрацию и авторизацию пользователя на сайте
4) Можно добавить любой новый полезный функционал
5) Сдать ссылку на репозиторий с проектом
6) Написать небольшой readme как работает система
7) Реализовать запуск проекта в docker
8) Сдать дз в виде ссылки на репозиторий

3 Разработка проекта с Server-Side Rendering на Django. Оптимизация и тестирование

1 **Django settings, orm, админка, миграции, superuser**

Цели занятия:

добавлять модели и базу данных в проект на Django;
делать миграции данных;
сохранять данные в базу;
объяснить как создавать проект на Django;
запомнить из чего состоит проект;
осознать, что такое миграции и зачем они нужны;
посмотреть на стандартную админку.

Краткое содержание:

создание проекта;
как проект django разбит на приложения;
введение в ORM на django;
как создавать модели;
разные типы полей в модели;
миграции;
superuser;
стандартная админка и как ей пользоваться;
выборка данных;
различные запросы в базу.

практическое задание: добавить модель данных, сделать миграции и ввести данные любым способом.

Домашние задания

1 Обучающий сайт на выбранную тему

Цель: В этой самостоятельной работе тренируем умения:
1. Создавать проект django
2. Настраивать его
3. Создавать модели данных
4. Делать миграции
5. Работать со стандартной админкой
Для того чтобы работать с проектами на django

1. Создать проект "Обучающий сайт". По нему будут задания до конца курса, сначала backend потом frontend. Рекомендуется создать для проекта отдельный репозиторий
2. Придумать тему (чему будем обучать на сайте) можно любую какая вам наиболее интересна.
3. Примерное описание работы сайта:
Сайт будет похож на OTUS + дополнительный функционал в зависимости от темы. На нем будут курсы, категории, преподаватели, занятия, расписание, ... + доп. функционал в зависимости от выбранной темы.
Будут как классические страницы (созданные на сервере), так и api в json
4. Необходимо создать модели данных (Преподаватель, студент, курс, расписание, ...). Можно не все сразу, а какую-то часть и постепенно добавлять новые
5. Создать связи между моделями
6. Сделать миграции
7. Настроить стандартную админку
8. Заполнить базу некоторыми реальными или тестовыми данными
9. Сдать сайт в виде ссылки на репозиторий (базу используем пока тестовую sqlite) сам файл с базой в репозиторий рекомендуется не заливать
10. Рекомендуется добавить в репозиторий файл readme с кратким описанием работы системы

---

2 **Django cbv, шаблоны, наследование шаблонов**

Цели занятия:

разобраться как работает шаблонизатор django;
объяснить для чего и как использовать наследование шаблонов;
осознать что такое cbv в django;
объяснить какие классы из cbv используются для crud;
разобраться для чего нужны классы View и TemplateView;
осознать для чего нужны Mixins и как они

позволяют расширять стандартные классы.

Краткое содержание:

шаблонизатор django;
наследование шаблонов и включенный шаблон;
Django CBV;
работа с основными классами для CRUD;
работа с View и TemplateView;
работа с Mixins и расширение стандартных классов CBV.

Домашние задания

1. Страницы для создания, удаления, редактирования, просмотра 1-го курса и списка курсов

   Цель: В этой самостоятельной работе мы тренируем умения:
   Использовать CBV для создания view;
   Использовать шаблонизатор django.
   Для того, чтобы использовать преимущества CBV и понимать механизм рендеринга шаблонов

   1. Продолжаем проект на django
   2. В проекте добавляем страницы (у нас будут как классические страницы с рендерингом на сервере, так и api с рендерингом на клиенте)
   3. Пока добавляем классические страницы с рендерингом на сервере
   4. Добавить страницу для просмотра списка курсов
   5. Добавить страницу для просмотра одного курса
   6. Добавить страницы для создания, удаления, редактирования курса
   7. Дополнительно можно добавить любой полезный функционал
   8. Так же в проект рекомендуется добавить необходимые базовые и включенные шаблоны

   Рекомендуется все view делать с использованием CBV

3	**Django forms. Наследование моделей. Абстрактные классы и proxy в django**	**Цели занятия:** взаимодействовать с пользователем с помощью Django Forms; проанализировать различные варианты форм; объяснить как можно настраивать форму; разобраться с наследованием моделей в Django; проанализировать варианты наследования. **Краткое содержание:** для чего используются Djago Forms; какие виды форм есть; как настроить внешний вид формы; как работать с формой во view и в шаблоне; виды наследования моделей в django.
4	**Django m2m, select_related/prefetch_related, django debug toolbar**	**Цели занятия:** объяснить зачем нужен django-debug-toolbar; установить и настроить; познакомиться с manytomany; добавлять many_to_many записи; объяснить зачем нужны prefetch_related и select_related и в чем их разница. **Краткое содержание:** установка django-debug-toolbar; связи manytomany; как добавлять many_to_many записи; prefetch_related, select_related и в чем их разница.
5	**Django ORM, оптимизация работы с БД**	**Цели занятия:** писать запросы с применением F-объектов; оптимизировать запросы с помощью

exists;
оптимизировать запросы с помощью cashed_property;
объяснить для чего и как использовать bulk update, iterator в queryset, аннотации.

Краткое содержание:

Prefetch - prefetch object, only, ManyToMany through;
update_fields в методе save, bulk update;
ленивые запросы в базу и iterator в queryset;
F-запрос и Аннотации (например агрегацию);
count, exists;
cashed_property.

Домашние задания

1. Оптимизировать работу с базой данных. Написать отчет

   Цель: В этой самостоятельной работе тренируем умения:
   1. Использовать django-debug-toolbar
   2. Использовать инструменты оптимизации

   1. Задача состоит в том чтобы ускорить работу сайта. Критерием будет количество запросов на странице (в большинстве случаев это подходит, иногда с уменьшением количества запросов увеличивается время ответа от страниц, поэтому можно еще обращать внимание на время загрузки страницы)

   2. Пробуем оптимизировать максимальное количество страниц на сайте, чем больше тем лучше для тренировки.
   Если какую то страницу оптимизировать не удается, ничего страшного бывают сложные непонятные ситуации.

   3. Дополнительно приложить отчет в виде таблицы (скрин или документ).

В таблице 4 колонки:

- Адрес страницы
- Кол-во запросов до оптимизации
- Кол-во запросов после оптимизации
- Средство оптимизации (кратко, например select_related, cached_property, with, кэширование, любые другие)
кэширование - желательно использовать только в крайнем случае и аккуратно

---

6 **Тестирование django приложений. Тестирование моделей. mixer для создания фейковых данных**

Цели занятия:

тестировать django-приложения;
запускать тесты;
объяснить для чего setUp и tearDown;
создавать фейковые данные с помощью mixer.

Краткое содержание:

как тестируются web-приложения;
запуск тестов для django-приложения;
проверка методов в модели;
методы setUp, tearDown;
создание тестовых фейковых данных с помощью mixer или аналогичной библиотеки.

практическое задание: проверить другие методы у модели.

---

7 **Django. фабрики: mixer, Factory Boy, Faker**

Цели занятия:

познакомиться поближе с Mixer, Factory Boy и Faker.

Краткое содержание:

для чего и как используются фабрики;
mixer;
factory boy;
faker.

---

8 **Азы работы с очередями задач**

Цели занятия:

разобраться зачем нужны очереди задачи;
настроить rq и redis;
создавать задачи;
запускать задачи по отдельности и по расписанию;
делать в проекте на django.

Краткое содержание:

зачем нужны очереди задач;
установка RQ;
Redis и redis сервер;
создание задачи и запуск в redis;
запуск задачи через delay и с помощью worker;
запуск по расписанию с rq-scheduler;
варианты запуска задач по расписанию;
использование в django django-rq;
мониторинг задач в django;
запуск по расписанию в django.

Домашние задания

1 Добавить страницу с контактами и отправкой сообщения с помощью очереди задач

Цель: В этой самостоятельно работе мы тренируем умения работает с очередями задачи. Для того чтобы использовать их в своей работе

1. Добавить страницу с контактами
2. На странице создать форму для отправки сообщения
3. После отправки формы отправлять письмо на почту администратора (о том, что нам отправили сообщение)
4.И второе письмо на почту указанную в форме (о том, что мы приняли его сообщение)
5 .Отправку писем реализовать через очередь задач (можно использовать rq или любую другую библиотеку)

4 Django REST framework. GraphQl. Создание API

1 **Введение в django-rest-framework**

Цели занятия:

объяснить зачем нужен rest framework;
установить rest framework;
работать с APIView;
объяснить для чего и как используются сериализаторы;
создать CRUD для модели данных.

Краткое содержание:

установка django-rest-framework;
работа с APIView;
сериализаторы;
создание CRUD для модели.

практическое задание: создать crud для новой модели.

---

2 **DRF. Serializers. Renderers. Routers**

Цели занятия:

понимать назначение serializers, renderers, routers в DRF;
выбирать нужный вариант класса для конкретной задачи;
создавать serializers для разных моделей;
описывать разные типы связей;

Краткое содержание:

разновидности Serializers, Renderers, Routers в DRF.

---

3 **DRF. Views. Filtering. Pagination**

Цели занятия:

понимать назначение views;
использовать различные варианты views в DRF;
добавлять фильтры в API;
добавлять постраничный вывод в API.

Краткое содержание:

APIView, GenericViews, ViewSets;
Filtering;
Pagination.

4 **Django-rest-framework авторизация**

Цели занятия:

проанализировать варианты авторизации с django-rest-framework;
объяснить в каком случае какой вариант используется;
реализовать некоторые варианты;
объяснить как происходит авторизация по JWT и Oauth2.

Краткое содержание:

авторизация с помощью django-rest-framework;
варианты авторизации, когда какой используется;
реализация нескольких вариантов авторизации;
JWT, Oauth2.

практическое задание: реализовать какой нибудь из вариантов авторизации.

Домашние задания

1 Создать rest-api для сайта

Цель: В этой самостоятельной работе мы тренируем умения:
1. создавать rest-api для сайта
2. реализовывать систему прав и авторизацию пользователя

1. Продолжаем работать с проектом
2. Подумать для каких частей сайта будет rest api
3. Реализовать rest api (можно использовать django-rest-framework или любые другие средства)
4. Продумать систему прав
5. Какие права будут по умолчанию для всех страниц?
6. Какие права будут для каждой страницы?
7. Дописать свои права, если они требуются
8. Продумать систему аутентификации
9. Реализовать систему аутентификации
10. Рекомендуется в систему аутентификации включить аутентификации по токену (из за большой популярности) и реализовать возможность создания токена для конкретного пользователя (например, в личном кабинете)

5 **DRF. Система версий. Документация**

Цели занятия:

использовать swagger для создания документации;
создавать и поддерживать разные версии API.

Краткое содержание:

варианты создания нескольких версий API в DRF;
генерация документации для API.

---

6 **API. GraphQL и его реализация в Python. GraphQL и Django**

Цели занятия:

разобраться зачем нужен GraphQL;
объяснить как он реализован в python;
объяснить как создавать схему;
проанализировать варианты использования GraphQL;
фильтровать данные с GraphQL;
изменять (мутировать) данные.

Краткое содержание:

зачем нужен GraphQL;
реализация GraphQL на python;
создание схемы;
примеры использования;
примеры фильтрации данных;
примеры изменения (мутации) данных.

Домашние задания

1 С помощью GraphQL создать схему

Цель: В этой самостоятельно работе мы тренируем умения работать с GraphQL

С помощью GraphQL создать схему, которая позволяет получать одновременно курсы, преподавателей и всех студентов записанных на курс

---

7 **Тестирование django приложений. Тестирование views. Тестирование api**

Цели занятия:

использовать тестовый клиент для тестирования view в django;
объяснить, что можно проверять на странице;
писать тесты для api.

Краткое содержание:

тестовый клиент в django;
как с помощью него проводить тестирование view;
что можно проверить с помощь тестового клиента;
тестирование api;
как проверить api.

практическое задание: написать тесты для страницы проекта.

---

8 **Code review бэкенд части приложения**

Цели занятия:

делать code review;
проанализировать слабые места своей работы;
запомнить best practice.

Краткое содержание:

обзор кода на основе работ студентов;
поиск зон роста;
демонстрация best practice.

5 Начинаем создавать frontend часть обучающего сайта, получаем данные с backend. Основы html, css, js, ES6, node.js, webpack, ajax

1 Современный фронтенд

Цели занятия:

различать современный JS синтаксис (JS, ES5/ES6/ES8);
основные типы;
наследование;
работа с функциями.

Краткое содержание:

JS, ES5/ES6/ES8.

---

2 ES6, NodeJS окружение

Цели занятия:

объяснить разницу кода в ES6;
разобраться с деструкцией и распаковкой;
проанализировать объекты в ES6, getters, setters;
разобраться с import, export;
объяснить зачем нужен node.js, npm.

Краткое содержание:

отличия кода в EC6 объекты, функции, переменные;
деструкция, распаковка;
объекты в EC6;
getters, setters;
import, export;
node.js, npm

---

3 **webpack + babel, транспайлинг**

Цели занятия:

babel, webpack;
разобраться с настройкой проекта.

Краткое содержание:

babel, webpack;
настройка проекта;
препроцессоры.

Домашние задания

1 Сборка UI с помощью webpack

Цель: В этой самостоятельной работе мы тренируем умения:
1. Устанавливать node.js
2. Устанавливать webpack
3. Писать код на ES6
4. Использовать babel

В этом домашнем задании можно потренироваться в отдельном проекте. В следующем домашнем задании мы будем собирать всем вместе с backend

1. Установить node js
2. После установки приложить скриншот с версией node.js
3. Для установки библиотек используем npm
4. Установить babel
5. Написать любой код на ES6 и с помощью babel посмотреть как он преобразуется в старый стиль
6. Приложить скриншот
7. Создать проект (nmp init)
8. Установить в проект пакет axios
9. Приложить скриншот package.json
10. Клонировать демонстрационный webpack проект https://github.com/wbkd/webpack-starter
11. Установить зависимости
12. Запустить проект
13. Приложить скриншот запущенного проекта

---

4 **CSS препроцессоры". "fetch || axios || $.ajax для REST запросов, модульность**

Цели занятия:

написать код на less;
объяснить зачем нужны css препроцессоры;
установить less в webpack;

объяснить зачем нужны ajax, axios, fetch и в чем их разница.

Краткое содержание:

чистый css и препроцессоры;
обзор препроцессоров;
упражнение: less;
установка less в webpack;
ajax, axios, fetch. примеры.

Домашние задания

1. Взаимодействие с api с помощью fetch или axios или ajax

   Цель: В этой самостоятельной работе мы тренируем умения:
   1. Взаимодействовать с backend
   2. Использовать fetch или axios или ajax

   1) Настроить сборку webpack-ом в проекте django. Это можно сделать по данной инструкции: https://gist.github.com/DanteOnline/501018d64e2ddbb2324121df9a94b7e5
   2) Создать в django тестовую view и тестовый url. Просто пустую страницу с шаблоном
   3) К шаблону подключить скрипт собранный webpack-ом <script src="{% static 'frontend/index.js' %}"></script>
   Проверить, что всё работает и скрипт выполняется при отгрузке страницы

   4) Установить axios
   5) В скрипте с помощью fetch получить данные по курсам
   6) В этом же скрипте с помощью axios получить данные по студентам (или любые другие для которых у вас есть api)
   7) Вывести полученные данные в консоль (этого будет достаточно) или на страницу использую минимальную разметку

   В этом дз этого будет достаточно, т.к. всё остальное будет удобнее делать с помощью react

   Для тех у кого есть желание ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:

   1. Сделать страницы логина и регистрации (но теперь мы взаимодействуем с сервером с помощью fetch или axios или ajax)
   Дополнительно:

2. Так же можно попробовать переделать другие страницы, например страницу со списком курсов для тренировки (хотя потом это удобнее будет сделать уже на react)
3 Если в api есть авторизация по токену, можно в личном кабинете сделать на ajax возможность обновлять и создавать токен

6 Создаем SPA приложение на Vue3. Vue2, Vuex, vue-router, SPA, тестирование в js

1 **Основы Vue, JSX, компоненты Vue**

Цели занятия:

объяснить, что такое компонентный подходи и зачем он нужен;
разобраться с virtual DOM.
разобраться со структурой Vue
настроить себе окружение IDE, зависимости и библиотеки для создания проектов и работы с Vue;
создавать простейшие приложения используя Vue.

Краткое содержание:

Hello Vue;
MVVM;
Vue templates;
Vue Dev tools.
Vue-CLI

2 **Компоненты, шаблонизатор и формы, Props, data-flow**

Цели занятия:

менять state компонента по событию;
запомнить варианты использования и валидации props;
запомнить какие есть доступные события;
объяснить, что такое state и для чего он используется;
объяснить разницу между state и props;
проанализировать принцип data-flow.

Краткое содержание:

one way data flow;
Vue components;
props & props validation;
refs;
filters;
computed, watchers.

Домашние задания

1 SPA (Single Page Application) игра "Устный счет"

В разделе Vue одна большая самостоятельная работа - SPA (Single Page Application) игра "Устный счет".
Игра состоит из двух экранов - на первом экране пользователь выбирает настройки, которые будут использовать в игре - типы вычислений, сложность, время раунда.
На этой же странице показывается статистика тренировок.
Вторая страница - сама игра.
Пользователь должен решить максимальное количество задач на заданное время.
Мокапы - https://app.moqups.com/korzio@gmail.com/bTYyBLCtpU/edit/page/ad64222d5

Подготовить общую структуру приложения - компоненты контейнеры для страниц приложения.
Сделать первую страницу приложения - форму настроек.

3 **Vue3 и Routing**

Цели занятия:

проанализировать этапы жизненного цикла vue-компонента;
объяснить в каком методе лучше делать загрузку данных с сервера и почему;
разобраться с загрузкой данных через fetch;
vue-router;
Vue3.

Краткое содержание:

этапы жизненного цикла компонента;
в каком методе делать загрузку данных с сервера и почему;
пример загрузки данных с сервера через fetch;
практический пример.

---

4 **Состояние приложения, Vuex**

Цели занятия:

после занятия вы сможете:
создавать плагины;
работать Vuex.

Краткое содержание:

plugins & directives;
vuex.

Домашние задания

1 Routing для приложения "Устный счет"

Цель: В этой самостоятельной работе мы тренируем умение использовать Vue

Реализовать второй экран - игру "калькулятор". Настроить переходы по страницам приложения.

---

5 **Vue, Best Practices**

Цели занятия:

работать с анимацией во Vue.
SSR с Nuxt;
паттерны и антипаттерны.

Краткое содержание:

SSR,
Nuxt,
CSS animations;
Vue codestyle.

---

6 **Тестирование JS приложений**

Цели занятия:

проанализировать инструменты тестирования в JS;
писать тесты;
запускать тесты;
Vue Test Utils.

Краткое содержание:

инструменты тестирования js;
запуск и примеры тестов.
практическое задание: написание теста.

---

7 **Code review frontend части на Vue**

Цели занятия:

проанализировать слабые места своей работы;
best practice.

Краткое содержание:

обзор кода на основе работ студентов;
поиск зон роста;
демонстрация best practice.

Домашние задания

1 Внести правка по итогам ревью кода

типовые ошибки - проверить есть ли они в ваших работах

7 Взаимодействие frontend и backend. Full-stack разработка. Добавляем vue к обучающему сайту

1 Варианты авторизации. JWT, cookies, 3rd party integration

Цели занятия:

различать варианты авторизации;
понимать плюсы и минусы каждого варианта;
понимать принципы авторизации по JWT, cookies, 3rd party integration.

Краткое содержание:

варианты авторизации (связь backend + frontend);
плюсы и минусы каждого из варинатов;
примеры реализации;
JWT;
cookies;
3rd party integration.

2 **Сборка проекта с docker для разработки и production**

Цели занятия:

настраивать среду разработки для работы с backend и frontend;
собирать проект (Django+Vue) с docker и docker-compose для разработки и для production.

Краткое содержание:

структура проекта backend и frontend;
связь nginx, gunicorn, django, vue;
сборка проекта с docker и docker-compose для разработки и production.

Домашние задания

1 Личный кабинет для обучающего сайта на vue

Цель: В этой самостоятельной работе мы тренируем умение использовать django совместно со vue

1. В проект "обучающий сайт" добавить frontend на vue (создать удобную структуру для работы с frontend и backend)
2. На vue у нас будет SPA для личного кабинета пользователя
3. В личном кабинете сделать следующие страницы (используем routing на vue):
- Страницу для списка курсов (на которые записался пользователь, для преподавателя это может быть курсы которые он ведет)
- Страницу для просмотра своего токена авторизации и его изменения
- Возможность отписаться от курса
- Возможность просмотра информации по курсу, на который записан студент (аналог detail в django)
(Итого 4 страницы, так же можно добавить любые другие по желанию)

Дополнительно можно собрать итоговый проект для production с docker и docker-compose, а также перенести проект на vps (при наличии) или виртуальную машину

3	**Контектсные процессоры и middleware в django. Подведение итогов**	**Цели занятия:** писать контекстные процессоры; объяснить строение middleware в django; оценить результаты обучения на курсе. **Краткое содержание:** контекстные процессоры; middleware; результаты обучения.

8 Проект

1 Выбор темы и организация проектной работы

Цели занятия:

выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом.

Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Выбор проекта и дальнейшая работа с ним

Цель: В данной проектной работе мы применим полученные знания и умения для реализации системы на любую свободную тему.
Это позволит
1. Закрепить полученные умения
2. Определить, все ли усвоилось или надо что то повторить
3. Определить свои сильные и слабые стороны
4. Научиться решать поставленную задачу

1. Выбрать тему проекта и утвердить ее в личном кабинете
2. Выделить самое главное в вашей системе
3. Реализовать самое главное
4. Реализовать все остальное, по мере сил
5. Приложить результат в чат с преподавателем

По желанию подготовиться выступление для занятия по защите проекта

2	**Консультация по проектам и домашним заданиям**	**Цели занятия:** получить ответы на вопросы по проекту, ДЗ и по курсу. **Краткое содержание:** вопросы по улучшению и оптимизации работы над проектом; затруднения при выполнении ДЗ; вопросы по программе.
3	**Защита проектных работ**	**Цели занятия:** защитить проект и получить рекомендации экспертов. **Краткое содержание:** презентация проектов перед комиссией; вопросы и комментарии по проектам.