OTUS

Web-разработчик на Python

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

Длительность курса: 176 академических часов

1 Основы python и модульного тестирования

1	**Знакомство с курсом. Проектирование "хорошей" системы. Написание "чистого" кода**	правильно декомпозировать код; именовать переменные и функции; оценивать сложность кода и использовать статические анализаторы.
2	**Модули, библиотеки, пакеты**	собирать пакет; объяснить, что такое области видимости, локальные и глобальные переменные; импортировать данные из модулей и пакетов; пользоваться конструкцией if __name__ == '__main__'; объяснять разницу между модулем и пакетом запомнить варианты импортов. Домашние задания 1 Создать программу-поисковик Цель: В этой самостоятельной работе тренируем умения: 1. Писать "чистый код" 2. Собирать пакеты Чтобы: Применять принципы написания чистого кода и сборку пакетов Задача: Создать программу поисковик (консольную)

		Пользователь вводит текст запроса, поисковую систему (google.com, yandex.ru, ...), количество результатов, рекурсивный поиск или нет, формат вывода (в консоль, в файл json, в csv) Программа находит в интернете начиная от стартовой точки все ссылки на веб-странице в заданном количестве (название ссылки и саму ссылку) Если поиск не рекурсивный, то берем ссылки только из поисковика, если рекурсивный, то берем первую ссылку, переходим, находим там ссылки, переходим, ... В зависимости от выбранного формата вывода сохраняем результат (текст ссылки: ссылка) либо в консоль либо в файл выбранного формата 0. Создать репозиторий для нового проекта (gitlab, github, ...) 1. Решить задачу 2. Обратить внимание на следующие принципы: 1. декомпозиция сверху вниз 2. srp - принцип единственной ответственности 3. термины предметной области 4. уменьшение зависимости 5. чистые функции 6. цикломатическая сложность python3 -m mccabe --min 5 module.py flake8 --max-complexity 5 7. понятные названия у переменных, функций, классов, модулей 8. контекст ближе к коду (привязка к комитам, тикетам, комменты, документация, вики) 9. разумное использование фишек python 10. код на английском а не на python 11. фичеризм - не слишком гибко, не слишком жестко, обобщать когда используется 2 раза 12. тесты демонстрирующие не очевидное поведение 13. статический анализ кода pycodestyle, flake8, ast 4. Добавить setup.py для сборки программы в пакет 5. Сдать дз в виде ссылки на репозиторий
3	**Введение в docker, docker-compose**	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
4	**Генераторы, тернарные операторы, исключения, декораторы**	писать генераторы; писать тернарные операторы; объяснить назначение декораторов; объяснить как писать декоратор; осознать назначение исключений; объяснить как обрабатывать исключения.

5	**Основы ООП. Принципы ООП**	создавать классы; создавать объекты классов; создавать свойства и методы класса; создавать методы доступа; объяснить как реализованы принципы ООП в python.
6	**ООП. Магические методы, утиная типизация, статические методы, методы класса**	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. 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 6. использовать docker Эти умения нам понадобятся при написании программ на ООП 1. Создать новый проект "Игра лото" 2. Правила игры можно скачать тут: https://drive.google.com/open?id=1bDMcmVfhYaTUw3CB-MV8WtIIXdYEutvX 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) таки образом чтобы можно было играть: компьютер - человек, человек - человек, компьютер - компьютер 6. (Дополнительно *) возможность играть для любого количества игроков от 2 и более

7. Создать Dockerfile для запуска проекта с помощью docker

8. Сдать дз в виде ссылки на репозиторий

Для тренировки предлагаю пользоваться принципами "чистого" когда, которые разбирали на первом занятии

---

7 **Введение в автотесты. pytest**

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.

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 и начинаем создавать обучающий сайт. База данных и ORM, web-фреймворки Flask и Django. MVC, MVT

1 **ORM, SQLAlchemy**

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.

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

---

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

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.

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

---

3 **Введение в werkzeug; Flask**

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.

---

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

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.

---

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

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

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

---

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

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.

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

---

7 **Django cbv, шаблоны,**

разобраться как работает шаблонизатор django;
объяснить для чего и как использовать наследование шаблонов;

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.

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 Создаем backend для обучающего сайта. REST API, django-rest-framework, GraphQL, оптимизация работы с базой данных

1	**Django forms. Наследование моделей. Абстрактные классы и proxy в django**	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.
2	**Тестирование django приложений. Тестирование моделей. mixer для создания фейковых данных**	тестировать django-приложения; запускать тесты; объяснить для чего setUp и tearDown; создавать фейковые данные с помощью mixer.
3	**Django. фабрики: mixer, Factory Boy, Faker**	познакомиться поближе с Mixer, Factory Boy и Faker7
4	**Азы работы с очередями задач**	разобраться зачем нужны очереди задачи; настроить rq и redis; создавать задачи; запускать задачи по отдельности и по расписанию; делать в проекте на 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)
5	**Введение в django-rest-framework**	объяснить зачем нужен rest framework; установить rest framework; работать с APIView; объяснить для чего и как используются

сериализаторы;
создать CRUD для модели данных.

---

6 **Django-rest-framework авторизация**

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.

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. Подумать систему авторизации
9. Реализовать систему авторизации
10. Рекомендуется в систему авторизации включить авторизацию по токену (из за большой популярности) и реализовать возможность создания токена для конкретного пользователя (например в личном кабинете)

---

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

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.

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
8	**Тестирование django приложений. Тестирование views. Тестирование api**	использовать тестовый клиент для тестирования view в django; объяснить, что можно проверять на странице; писать тесты для api.
9	**Django m2m, select_related/prefetch_related, django debug toolbar**	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.
10	**Django ORM, оптимизация работы с БД**	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. 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 бывает 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
11	**Code review бэкенд части приложения**	делать code review; проанализировать слабые места своей работы; запомнить best practice.

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

1	**Продвинутый JS: ООП в JS, прототипирование, асинхронность**	разобраться с ООП в js; разобраться с объектами и функциями; объяснить разницу конструкторов в es5 и es6; разобраться с прототипами; объяснить разницу методов в es5 и es6; объяснить разницу в наследовании в es5 и es6; разобраться с асинхронностью и моделью памяти.
2	**ES6, NodeJS окружение**	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
3	**webpack + babel, транспайлинг**	babel, webpack; разобраться с настройкой проекта. 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 Дополнительно можно почитать материалы как frontend будет взаимодействовать с backend

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

написать код на less;
объяснить зачем нужны css препроцессоры;
установить less в webpack;
объяснить зачем нужны ajax, axios, fetch и в чем их разница.

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

5 Создаем SPA приложение на React, собираем все воедино. React, Redux, SPA, тестирование в js, docker

1	**Основы React, JSX, компоненты React**	создавать приложение на react; создавать react-компоненты; объяснить, что такое компонентный подходи и зачем он нужен; разобраться со структурой react-приложения; проанализировать как работает render; разобраться с virtual DOM. Домашние задания 1 Сделать главную страницу на react Цель: В этой самостоятельной работе мы тренируем умения: 1) Запускать приложение на React 2) Использовать компоненты 3) Использовать render В предыдущем дз мы создали структуру проекта и использовали axios Почти все готово для использования react. В этом дз мы будем делать на React главную страницу (если в вашем проекте на ней есть данные). Если данных нет, то можно сделать любую другую страницу где есть данные и api для их получения (например страницу курсов) 1) Установить npm install react react-dom 2) Тестовый файл который мы создавали в предыдущем дз можно удалить. 3) Вместо него мы создаем точку входа для react (тот же самый файл, только там будет код с импортами react-компонентов) - В папке src (где находятся исходные js файлы) создаем папку components, в ней будут находиться компоненты react приложения 4) В папке components создать компонент для главной страницы. 5) В нем определить следующие части (конструктор, метод render, метод componentDidMount) 6) В методе componentDidMount получить данные с backend (как мы делали в предыдущем дз) 7) В методе render отрисовать страницу с этими данными 8) Сделать экспорт компонента 9) В файле index.js сделать import компонента 10) npm run dev - собираем 11) Выходной файл подключаем к главной странице сайта, а саму страницу оставляем пустую (с пустым контейнером куда будем рендерить) Пример с минимальной настройкой django, webpack, react и одной страницей: https://yadi.sk/d/-XqePypXWTmJUQ
2	**State и props, data-flow в React-**	менять state компонента по событию; запомнить варианты использования props;

	компонентах	запомнить какие есть доступные события; объяснить, что такое обертка на event и зачем она нужна; объяснить, что такое state и для чего он используется; объяснить разницу между state и props; проанализировать принцип data-flow.
3	**Варианты авторизации. JWT, cookies, 3rd party integration**	различать варианты авторизации; понимать плюсы и минусы каждого варианта; понимать принципы авторизации по JWT, cookies, 3rd party integration;
4	**Жизненный цикл React-компонент**	проанализировать этапы жизненного цикла react-компонента; объяснить в каком методе лучше делать загрузку данных с сервера и почему; разобраться с загрузкой данных через fetch. Домашние задания 1 Сделать страницу курсов, одного курса и записи на курс на react Цель: В этой самостоятельной работе мы тренируем умения работать с react 1) С помощью react сделать следующие страницы: - Список курсов - Страница просмотра одного курса - Страница с возможностью записи на курс - Так же можно добавить любые другие полезные для данной системы страницы 2) До занятия по роутингу в react можно создавать новый js-файл и подключать его к соответствующей странице (переходами будет заниматься backend)
5	**Состояние приложения. Flux & Redux**	объяснить зачем нужны Flux и Redux и в чем их разница; запомнить роли во Flux; запомнить реализации Flux; разобраться с работой Redux и его основными принципами. Домашние задания 1 Перевести все страницы сайта на react Цель: В этой самостоятельной работе мы тренируем умение использовать react Заканчиваем наполнять проект функционалом. На react добавляем следующие страницы: 1. Личный кабинет студента, где он может смотреть свои курсы 2. Страницу для генерации и изменения токена авторизации (если ее еще нет), тоже в личном кабинете 3. Любой другой функционал (доделываем сайт) После этого у нас останется попробовать перевести всё в spa (используя react маршрутизацию) и добавить

тестирование

Примерное описание итоговой работы сайта (минимально):
- Список курсов (возможно с другими связанными моделями) - CRUD
- Регистрация/авторизация (в сессии, по токену)
- Запись на курс
- Личный кабинет со списком курсов
- Возможность создавать/пересоздавать токен авторизации
- Страница контактов с возможностью отправить сообщение

---

6 **React hooks**

понять механизм react hooks;
осознать в каких случаях они используются;

---

7 **Routing в React. SPA**

объяснить, что такое SPA и для чего он используется;
рассмотреть Routing в react;
разобраться как создать SPA приложение.

Домашние задания

1 Организация всего приложения в виде SPA

Цель: В этой самостоятельно работе мы тренируем умения:
1. работать со spa приложениями
2. использовать react-router

1. Сохранить предыдущую версию проекта (в гите или локально), она может пригодиться для сравнения и отката.
2. Организовать приложение в виде spa. Для этого
3. Установить react-router
4. Прописать с помощью нее маршрутизацию
5. Сделать одну точку входа на главной странице (теперь пользователь будет переходить на нее, а маршрутизацией заниматься сам react)
6. Ссылки при рендеринге на react рекомендуется заменить на Link
Дополнительно:
Можно попробовать в некоторых местах использовать hooks

---

8 **Тестирование JS приложений**

проанализировать инструменты тестирования в js;
писать тесты;
запускать тесты.

Домашние задания

1 Написание unit-тестов для UI и back-end

Цель: В этой самостоятельной работе мы тренируем умения:
1. Писать тесты для backend
2. Писать тесты для frontend

1. Чем больше будет покрытие тестированием, тем лучше
2. Минимально рекомендуется попробовать написать

		несколько новых тестов для models, view, api и несколько тестов для frontend
9	**Code review frontend части на react**	проанализировать слабые места своей работы; best practice.
10	**Контектсные процессоры и middleware в django. Подведение итогов**	писать контекстные процессоры; объяснить строение middleware в django оценить результаты обучения на курсе.

6 Проект

1	**Выбор темы и организация проектной работы**	выбрать и обсудить тему проектной работы; спланировать работу над проектом; ознакомиться с регламентом работы над проектом. 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**	получить ответы на вопросы по проекту, ДЗ и по курсу.
3	**Защита проектных работ**	защитить проект и получить рекомендации экспертов.