Задание на 3 (удовлетворительно)

№	Задание	Комментарий/критерии выполнения
1	По выданной теме сделать минимум 3 модели, сериализаторы, api.	Модели связаны между собой Есть возможность по api получить список объектов, получить 1 конкретный объект, удалить объект, создать объект
2	Валидация полей.	Реализовать все 3 предложенных варианта
3	Использование Q в запросах	См варианты сложных запросов в описании темы. Минимум 2 запроса с Q, включающих одновременно OR и AND и NOT (|, &, ~)
4	Пагинация	https://www.django-rest-framework.org/api-guide/pagination/
5	Фильтрация	Реализовать минимум 5 вариантов, см примеры

6	Поиск по названию или по содержанию	`from rest_framework.filters import SearchFilter`
		
		`class TaskListView(generics.ListAPIView):`
		`  queryset = Task.objects.all()`
		`  serializer_class = TaskSerializer`
		`  filter_backends = [SearchFilter]`
		`  search_fields = ['title']`

7	Дополнительные методы	Обязательно @action(methods=['GET'], detail=False) @action(methods=['POST'], detail=True)
8	Сохранение истории изменений объектов	Подключить Django-simple-history https://django-simple-history.readthedocs.io/en/latest/
9	Экспорт данных из админки в эксель	https://django-import-export.readthedocs.io/en/latest/api_admin.html Реализовать минимум 3 метода для кастомизации, например `get_export_queryset`, `dehydrate_{field_name}`, `get_{field_name}`)
10	Админка должна быть настроена для удобной работы	В admin.py есть list_display, fields, list_filter, есть Гиперссылки на страницы других объектов, short_description, fieldsets, inlines. Можно заменить на другие настройки, должно быть минимум 5.
11	Знать теорию	Ответить на 3 случайных вопроса из списка

Django:

1. Шаблон проектирования Model-View-Controller, его реализация в Джанго (MTV (Model, Template, and View)), особенности и отличия - кратко, можно показывать на примере своего проекта
2. Что такое CSRF? Какие механизмы есть в джанго для защиты (https://docs.djangoproject.com/en/4.1/ref/csrf/)

3. Клиент посылает запрос на сервер, как вернуть ему какие-то данные в ответ? (JsonResponse, Response) Какие бывают статусы ответов? (2xx, 3xx, 4xx, 5xx + знать по 2-3 примера из каждой группы)
4. Что такое Middleware, для чего, как реализуется
5. Запросы на сервер, отличия работы с GET и POST запросами?
6. Что такое select_related и prefetch_related
7. Что такое Meta в классах Django и для чего нужен
8. Административная панель Django - где настраивается, регистрация моделей - показать на примере своего проекта
9. Административная панель Django - list_display, list_filter, fieldsets - показать на примере своего проекта
10. Административная панель Django - ModelAdmin.inlines - показать на примере своего проекта
11. Административная панель Django - raw_id_fields
12. Функции login, logout, get_user_model (из django.contrib.auth)

REST:

1. Что такое REST
2. Отличия viewsets.ViewSet / GenericViewSet /viewsets.ModelViewSet / APIView
3. Дополнительные действия для маршрутизации - @detail_route, @list_route,
4. Дополнительные действия для маршрутизации - @action
5. REST framework - поддержка автоматического роутинга (Метод register() и его аргументы)
6. Сериализаторы в REST framework (что такое, зачем нужно, какие могут быть поля и как они описываются)
7. Сериализаторы - class Meta, extra_kwargs
8. Сериализаторы - class Meta, validators
9. reverse и reverse_lazy - что это за функции и уметь привести пример на основе своего проекта (можно в консоли)
10. Функция get_serializer_context

Пример реализации (будет обновляться) https://github.com/igolubeva/mospol_ip_2022

Задание на оценку 4 (хорошо)

1) Подключаем linter
2) Подключаем swagger
3) Настраиваем docker
4) Celery – реализовать 2 периодические задачи, смотри примеры
5) mailhog – реализовать отправку писем

3) redis(запись результатов в cache) – реализовать кеширование запросов.

При первом запросе данные должны извлекаться из базы данных и сохраняться в кэше Redis. При последующих запросах данные должны извлекаться из кэша.

Например,

```python
def get_user_tasks(user):
    cache_key = f"user_tasks_{user.id}"
    tasks = cache.get(cache_key)
    if tasks is None:
        tasks = get_list_or_404(Task, owner=user)
        cache.set(cache_key, tasks, timeout=60*15)  # Кэш на 15 минут
    return tasks
```

На оценку 5 (отлично) сделать на выбор:

1) Реализовать логирование (запись информации) о посещении пользователей вашего сайта (кто на какую страницу заходил)

2) OAUTH2 – реализовать возможность заходить на ваш сайт через соц сети

Варианты заданий:

1. Система управления задачами

- Фильтрация:

1. Фильтрация задач по статусу (Фильтр по именованным аргументам в URL)
2. Фильтрация по приоритету (DjangoFilterBackend)
3. Фильтрация по дате окончания (Фильтр по GET-параметрам в URL)
4. Фильтрация по диапазону дат (django_filters)
5. Фильтрация по текущему аутентифицированному пользователю

- Валидация:

1. Проверка, что дата начала не позже даты окончания.
2. Валидация приоритета (например, должен быть в пределах 1-5).
3. Проверка уникальности названия задачи для пользователя.

- Сложные запросы:

1. Получение задач, которые должны быть выполнены в ближайшие 7 дней.
2. Поиск задач, содержащих определенное слово в описании.
3. Получение задач, которые не выполнены и имеют высокий приоритет.
4. Найдите все задачи, которые либо имеют высокий приоритет и не завершены, либо задачи, которые должны быть выполнены завтра.
5. Найдите все задачи, которые не принадлежат текущему пользователю, но имеют статус "в процессе" или "отменены".

- Дополнительные методы:

1. @action(methods=['post'], detail=True) для изменения статуса задачи.
2. @action(methods=['get'], detail=False) для получения всех просроченных задач.
3. @action(methods=['get'], detail=True) для получения истории изменений задачи.

- Экспорт данных из админки в Excel:

Задание: Реализуйте экспорт задач из админки в формат Excel. Кастомизируйте следующие методы:

- `get_export_queryset`: Измените стандартный набор данных для экспорта, чтобы включать только задачи с высоким приоритетом.
- `dehydrate_due_date`: Преобразуйте поле `due_date` в формат "DD-MM-YYYY" для удобства чтения.
- `get_status`: Преобразуйте поле `status` в более читабельный формат (например, "Completed" вместо "C").

Пример периодических задач:

`archive_tasks` — Архивирует все задачи, которые завершены более 30 дней назад.

`send_task_reminders` — Отправляет напоминания по email о задачах, которые должны быть выполнены завтра.

`delete_old_tasks` — Удаляет задачи, которые были созданы более года назад и не завершены.

2. Приложение для заметок

- Фильтрация:

1. Фильтрация заметок по дате создания (Фильтр по GET-параметрам в URL)
2. Фильтрация по категории (Фильтр по именованным аргументам в URL)
3. Фильтрация по автору (DjangoFilterBackend).
4. Фильтрации заметок по диапазону дат создания (`django_filters`)
5. Фильтрация по текущему аутентифицированному пользователю

- Валидация:

1. Проверка уникальности заголовка заметки.
2. Ограничение на длину заметки.
3. Валидация категории (например, наличие в списке разрешенных категорий - Валидация должна проверять, что категория заметки принадлежит к списку заранее определенных категорий (например, "Работа", "Личное", "Учеба"). Если категория не соответствует ни одной из разрешенных, приложение должно вернуть ошибку.).

- Сложные запросы:

1. Поиск заметок, созданных в последние 30 дней.
2. Получение заметок, содержащих определенные ключевые слова в заголовке или содержании.
3. Поиск заметок, принадлежащих нескольким категориям.
4. Найдите все заметки, которые не принадлежат текущему пользователю и содержат ключевое слово "важно" в заголовке или содержании
5. Найдите все заметки, которые были обновлены за последние 7 дней, но не содержат ни одного вложения или имеют статус "черновик".

- Дополнительные методы:

1. @action(methods=['post'], detail=True) для добавления заметки в избранное.
2. @action(methods=['get'], detail=False) для получения всех избранных заметок.
3. @action(methods=['post'], detail=True) для изменения категории заметки.

Экспорт данных из админки в Excel

Задание: Реализуйте экспорт заметок из админки в формат Excel. Кастомизируйте следующие методы:

- `get_export_queryset` : Экспортируйте только заметки, созданные за последний месяц.
- `dehydrate_title` : Добавьте к заголовку заметки префикс "Note: ".
- `get_author_name` : Преобразуйте поле `author` в формат "Имя Фамилия".

Пример периодических задач:

`archive_old_notes` — Архивирует все заметки, которые не обновлялись более 6 месяцев.

`send_note_summaries` — Отправляет еженедельные резюме заметок пользователям.

`delete_temp_notes` — Удаляет временные заметки, которые были созданы более недели назад.

3. Приложение для рецептов

- Фильтрация:

1. Фильтрация рецептов по типу кухни (Фильтр по именованным аргументам в URL).
2. Фильтрация по времени приготовления (Фильтр по GET-параметрам в URL).
3. Фильтрация по ингредиентам (DjangoFilterBackend).
4. Фильтрации рецептов по диапазону времени приготовления (django_filters).
5. Фильтрация по текущему аутентифицированному пользователю

- Валидация:

1. Проверка уникальности названия рецепта.
2. Валидация количества ингредиентов (например, не более 20).
3. Проверка времени приготовления (должно быть положительным числом).

- Сложные запросы:

1. Поиск рецептов, содержащих определенный ингредиент.
2. Получение рецептов, которые готовятся за менее чем 30 минут.
3. Поиск рецептов определенной кухни с использованием нескольких ингредиентов.
4. Найдите все рецепты, которые содержат "шоколад" в ингредиентах, но не являются десертами или имеют время приготовления менее 30 минут.
5. Найдите все рецепты, которые не были добавлены текущим пользователем и имеют рейтинг ниже 3, но входят в категорию "веганские".

- Дополнительные методы:

1. @action(methods=['post'], detail=True) для добавления рецепта в избранное.
2. @action(methods=['get'], detail=False) для получения популярных рецептов.
3. @action(methods=['post'], detail=True) для добавления комментария к рецепту.

Экспорт данных из админки в Excel

Задание: Реализуйте экспорт рецептов из админки в формат Excel. Кастомизируйте следующие методы:

- `get_export_queryset`: Экспортируйте только рецепты с рейтингом выше 4.
- `dehydrate_ingredients`: Преобразуйте список ингредиентов в строку, разделенную запятыми.

- `get_cuisine_type`: Преобразуйте поле `cuisine` в формат "Кухня: {тип кухни}".

Пример периодических задач:

`update_recipe_ratings` — Обновляет рейтинг всех рецептов на основе новых отзывов.

`notify_new_recipes` — Уведомляет пользователей о новых рецептах, добавленных за последнюю неделю.

`delete_unpopular_recipes` — Удаляет рецепты, которые имеют рейтинг ниже 2 и были созданы более года назад.