МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по лабораторной работе №9**

**на тему:**

**Работа с админ-панелью**

**по дисциплине:** Разработка программных приложений и WEB-программирование

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г.

«10» апреля 2025 г.

Проверил работу:

Эстрайх И. В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Цель**

Изучить работу с админ-панелью фреймворка Django.

**Задание**

1. Запустить админ-панель, переключиться на русскую локализацию.

2. Создать суперпользователя.

3. Зарегистрировать в админ-панели свое приложение (модель).

4. Войти в админ-панель как суперпользователь. Изучить интерфейс панели. 5. Настроить панель в соответствии с содержанием сайта (заголовки и др.).

6. Настроить отображение списка записей (отображаемые поля, порядок сортировки, редактирование полей и др.).

7. Добавить пользовательские поля (не менее двух).

8. Добавить пользовательские действия (не менее двух). Продемонстрировать их использование. Действия должны сопровождаться выводом сообщений.

9. Добавить и настроить панели поиска и фильтрации. Добавить собственный фильтр. Продемонстрировать работу панелей.

10. Настроить формы добавления и редактирования записей в админ панели. 11. Настроить внешний вид админ-панели в соответствии с оформлением сайта.

**Описание хода выполнения работы**

1. Запустить админ-панель, переключиться на русскую локализацию.

Для начала добавим path для админ панели в файл

cs\_ty/cs\_ty/urls.py:

path('admin/', admin.site.urls),

Затем в файле settings.py изменим локализацию:

LANGUAGE\_CODE = 'ru-RU'

В итоге можно открыть админ панель сайта:

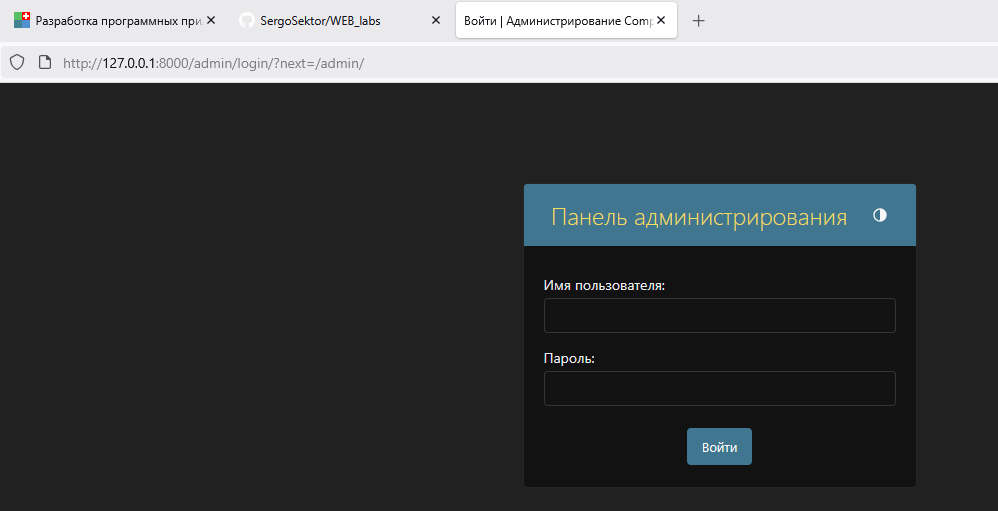


Рисунок 1 – Админ панель.

2. Создать суперпользователя.

С помощью терминал создадим суперпользователя командой

py manage.py createsuperuser, затем введем в поля: Имя пользователя, пароль, подтверждение пароля.

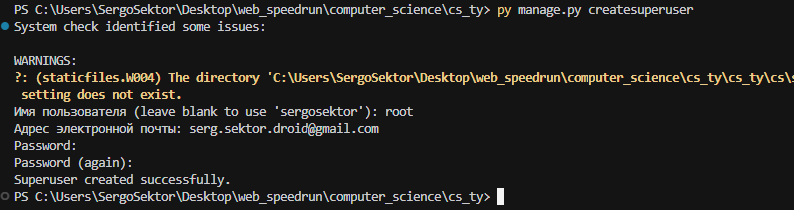


Рисунок 2 – Создание суперпользователя.

3. Зарегистрировать в админ-панели свое приложение (модель).

В файле admin.py импортируем модуль admin и регистрируем все модели с помощью @admin.register.

@admin.register(ComputerScienceConcept)

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Настройка формы добавления/редактирования записей:*

*# Поля, отображаемые в форме, перечисляются в указанном порядке.*

    fields = ('title', 'slug', 'description', 'difficulty', 'field\_of\_study', 'tags', 'is\_published')

*# Автоматическая генерация поля slug на основе поля title.*

    prepopulated\_fields = {"slug": ("title",)}

*# Поля, которые доступны только для чтения*

    readonly\_fields = ['time\_create', 'time\_update']

    list\_display = (

        'id',

        'title',

        'field\_of\_study',

        'time\_create',

        'is\_published',

        brief\_info,

        difficulty\_level,

        estimated\_reading\_time,

    )

    list\_display\_links = ('id', 'title')

    list\_editable = ('is\_published',)

    ordering = ['-time\_create', 'title']

    search\_fields = ('title', 'field\_of\_study\_\_name')

    list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]

    list\_per\_page = 5

*# Пользовательское действие для установки статуса "Опубликовано"*

    @admin.action(*description*="Опубликовать выбранные концепции")

    def set\_published(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*self*.message\_user(*request*, f"Статус 'Опубликовано' обновлён для {count} концепций.", messages.SUCCESS)

*# Пользовательское действие для установки статуса "Черновик"*

    @admin.action(*description*="Снять с публикации выбранные концепции")

    def set\_draft(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*self*.message\_user(*request*, f"{count} концепций сняты с публикации.", messages.WARNING)

    actions = ['set\_published', 'set\_draft']

@admin.register(FieldOfStudy)

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

@admin.register(ConceptDetail)

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'estimated\_learning\_time')

    list\_display\_links = ('id', 'concept')

    search\_fields = ('concept\_\_title', 'core\_technologies')

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

4. Войти в админ-панель как суперпользователь. Изучить интерфейс панели.

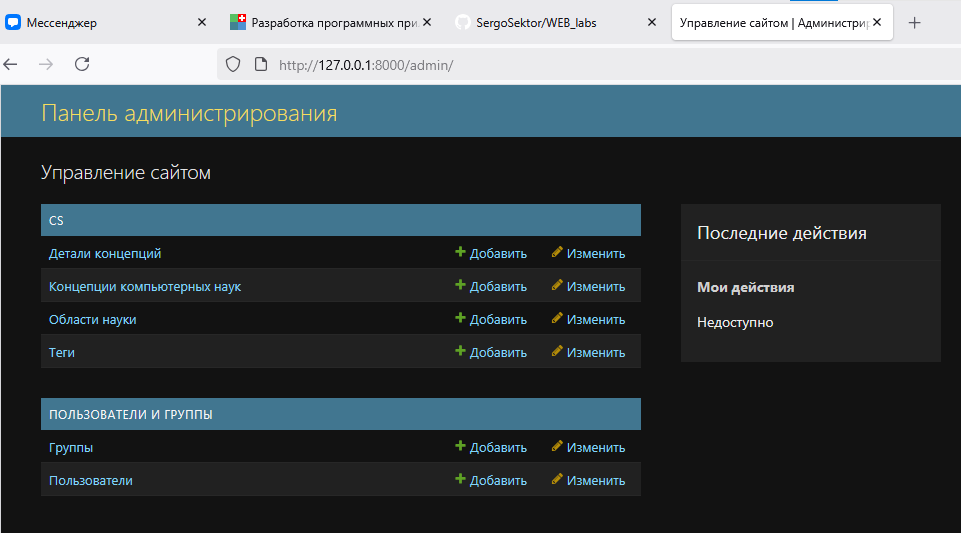
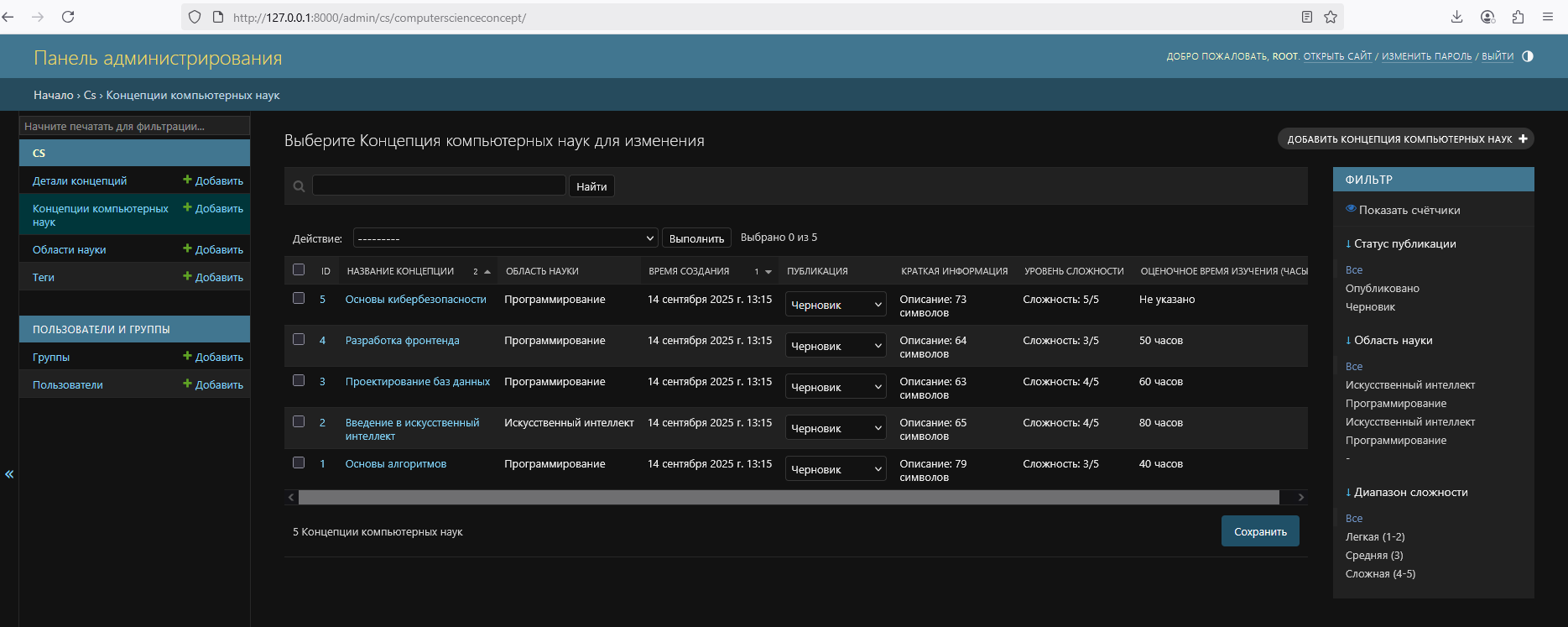


Рисунок 3 – Начальная страница админ панели при входе под суперпользователем.

  
Рисунок 4 – Интерфейс модели ComputerScienceConcept в админ панели.

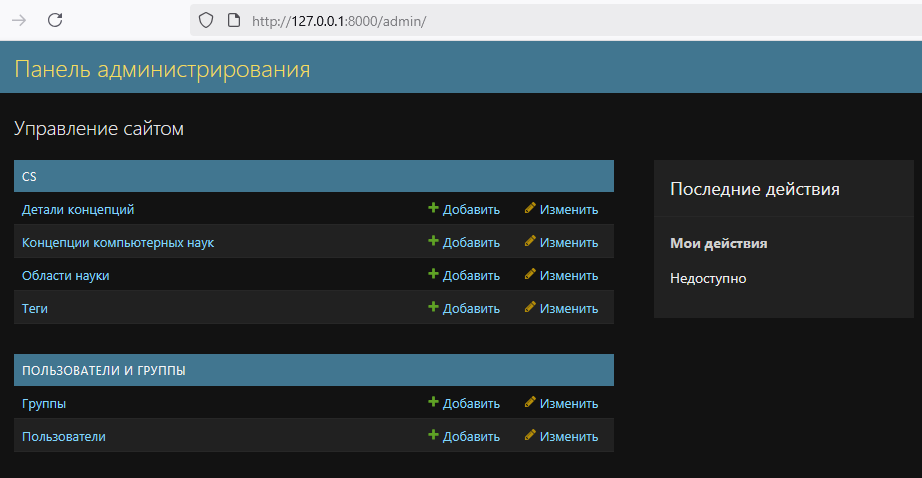
5. Настроить панель в соответствии с содержанием сайта (заголовки и др.).

Добавим в файл cs\_ty/cs/admin.py изменения header и title:

admin.site.site\_header = "Панель администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование Computer Science Project"

  
Рисунок 6 – Измененные заголовки.

6. Настроить отображение списка записей (отображаемые поля, порядок сортировки, редактирование полей и др.).

Изменим файл admin.py:

*from* django.contrib *import* admin, messages

*from* django.contrib.admin *import* SimpleListFilter

*from* .models *import* ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag

admin.site.site\_header = "Панель администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование Computer Science Project"

*# Функция для вычисляемого поля "Краткая информация"*

@admin.display(*description*="Краткая информация")

def brief\_info(*obj*):

*return* f"Описание: {len(*obj*.description)} символов" *if* *obj*.description *else* "Нет описания"

*# Пользовательское вычисляемое поле: уровень сложности*

@admin.display(*description*="Уровень сложности")

def difficulty\_level(*obj*):

*if* *obj*.difficulty is not None:

*return* f"Сложность: {*obj*.difficulty}/5"

*return* "Не указана"

*# Пользовательское вычисляемое поле: оценочное время изучения (часы)*

@admin.display(*description*="Оценочное время изучения (часы)")

def estimated\_reading\_time(*obj*):

*if* hasattr(*obj*, 'detail') and *obj*.detail.estimated\_learning\_time is not None:

*return* f"{*obj*.detail.estimated\_learning\_time} часов"

*return* "Не указано"

*# Кастомный фильтр для статуса публикации*

class PublishedFilter(SimpleListFilter):

    title = "Статус публикации"

    parameter\_name = "pub\_status"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ("published", "Опубликовано"),

            ("draft", "Черновик"),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == "published":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*elif* *self*.value() == "draft":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*return* *queryset*

*# Дополнительный кастомный фильтр по диапазону сложности*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"  *# Заголовок фильтра в админ-панели*

    parameter\_name = "difficulty\_range"  *# Параметр в URL*

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ('easy', 'Легкая (1-2)'),

            ('medium', 'Средняя (3)'),

            ('hard', 'Сложная (4-5)'),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == 'easy':

*return* *queryset*.filter(*difficulty\_\_lte*=2)

*elif* *self*.value() == 'medium':

*return* *queryset*.filter(*difficulty*=3)

*elif* *self*.value() == 'hard':

*return* *queryset*.filter(*difficulty\_\_gte*=4)

*return* *queryset*

@admin.register(ComputerScienceConcept)

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Настройка формы добавления/редактирования записей:*

*# Поля, отображаемые в форме, перечисляются в указанном порядке.*

    fields = ('title', 'slug', 'description', 'difficulty', 'field\_of\_study', 'tags', 'is\_published')

*# Автоматическая генерация поля slug на основе поля title.*

    prepopulated\_fields = {"slug": ("title",)}

*# Поля, которые доступны только для чтения*

    readonly\_fields = ['time\_create', 'time\_update']

    list\_display = (

        'id',

        'title',

        'field\_of\_study',

        'time\_create',

        'is\_published',

        brief\_info,

        difficulty\_level,

        estimated\_reading\_time,

    )

    list\_display\_links = ('id', 'title')

    list\_editable = ('is\_published',)

    ordering = ['-time\_create', 'title']

    search\_fields = ('title', 'field\_of\_study\_\_name')

    list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]

    list\_per\_page = 5

*# Пользовательское действие для установки статуса "Опубликовано"*

    @admin.action(*description*="Опубликовать выбранные концепции")

    def set\_published(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*self*.message\_user(*request*, f"Статус 'Опубликовано' обновлён для {count} концепций.", messages.SUCCESS)

*# Пользовательское действие для установки статуса "Черновик"*

    @admin.action(*description*="Снять с публикации выбранные концепции")

    def set\_draft(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*self*.message\_user(*request*, f"{count} концепций сняты с публикации.", messages.WARNING)

    actions = ['set\_published', 'set\_draft']

@admin.register(FieldOfStudy)

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

@admin.register(ConceptDetail)

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'estimated\_learning\_time')

    list\_display\_links = ('id', 'concept')

    search\_fields = ('concept\_\_title', 'core\_technologies')

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

Для модели ComputerScienceConcept в ComputerScienceConceptAdmin настроены:

* list\_display – задаёт поля, отображаемые в списке (включая вычисляемые поля brief\_info, difficulty\_level и estimated\_reading\_time);
* list\_display\_links – определяет кликабельные поля (id и title);
* list\_editable – поле is\_published можно редактировать прямо из списка;
* ordering – сортировка по времени создания (по убыванию) и по title;
* list\_per\_page – количество записей на странице (5 записей);
* prepopulated\_fields – автоматическая генерация слага на основе поля title;
* search\_fields – поиск по полям title и по связанному полю field\_of\_study\_\_name;
* list\_filter – фильтрация по статусу публикации (PublishedFilter), области науки (field\_of\_study) и диапазону сложности (DifficultyRangeFilter).

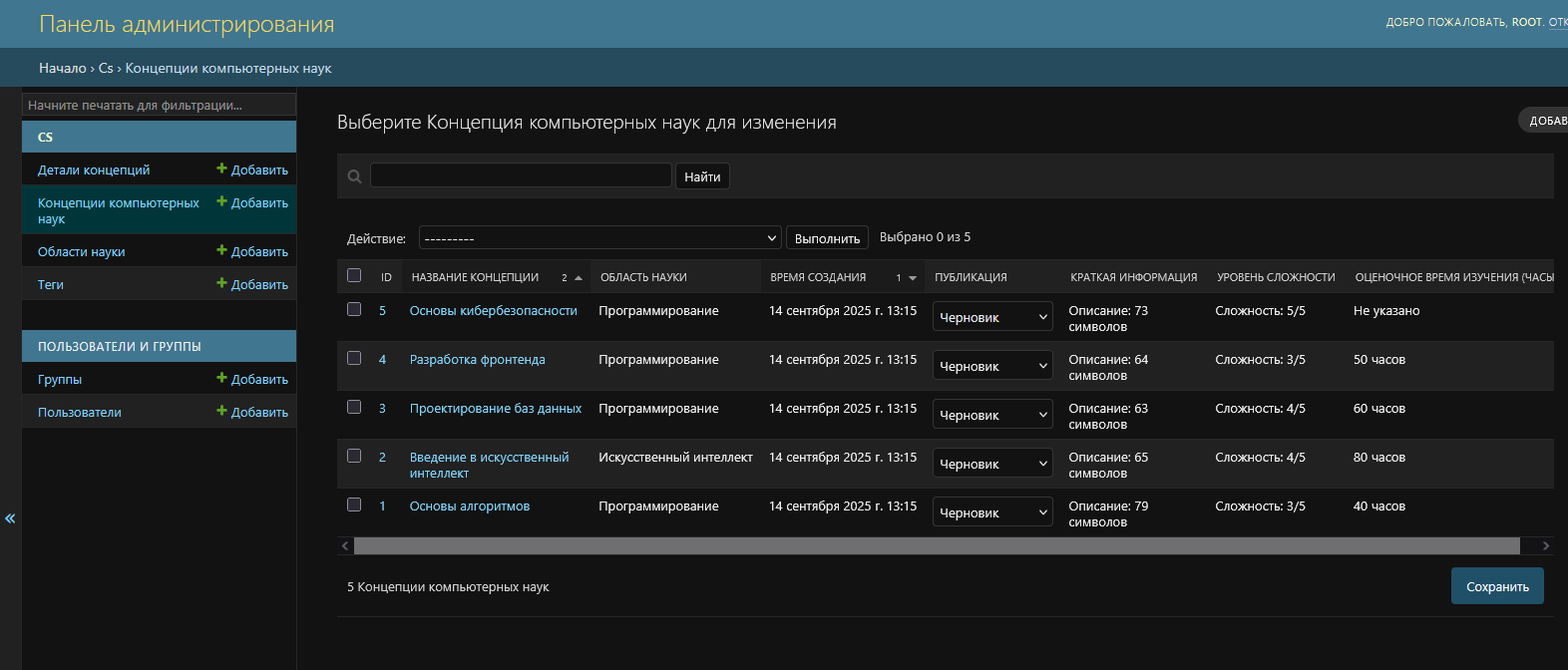
Для моделей FieldOfStudy, ConceptDetail, Tag заданы аналогичные настройки для простоты управления.

Рисунок 7 – Настроенное отображение списка записей.

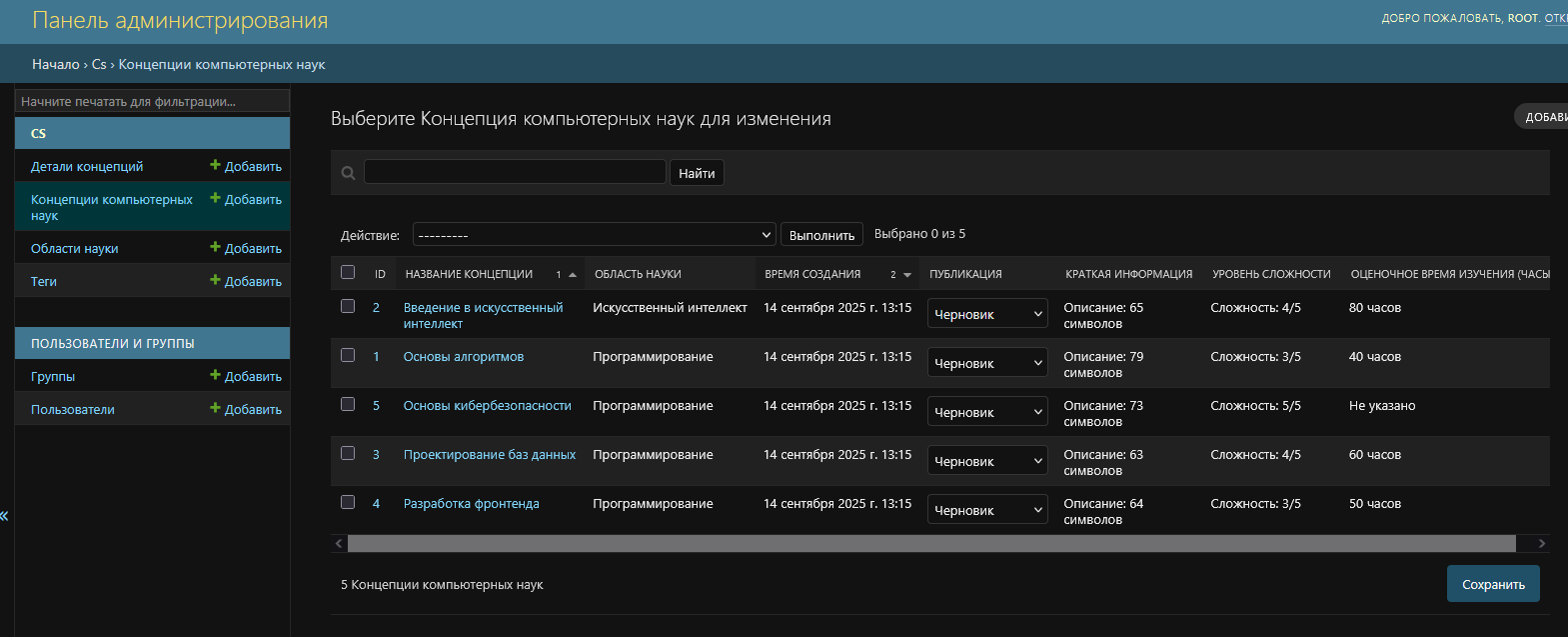


Рисунок 8 – Пример при нажатии на фильтрация по названию.

7. Добавить пользовательские поля (не менее двух).

Файл admin.py:

*# Функция для вычисляемого поля "Краткая информация"*

@admin.display(*description*="Краткая информация")

def brief\_info(*obj*):

*return* f"Описание: {len(*obj*.description)} символов" *if* *obj*.description *else* "Нет описания"

*# Пользовательское вычисляемое поле: уровень сложности*

@admin.display(*description*="Уровень сложности")

def difficulty\_level(*obj*):

*if* *obj*.difficulty is not None:

*return* f"Сложность: {*obj*.difficulty}/5"

*return* "Не указана"

*# Пользовательское вычисляемое поле: оценочное время изучения (часы)*

@admin.display(*description*="Оценочное время изучения (часы)")

def estimated\_reading\_time(*obj*):

*if* hasattr(*obj*, 'detail') and *obj*.detail.estimated\_learning\_time is not None:

*return* f"{*obj*.detail.estimated\_learning\_time} часов"

*return* "Не указано"

@admin.register(ComputerScienceConcept)

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Настройка формы добавления/редактирования записей:*

*# Поля, отображаемые в форме, перечисляются в указанном порядке.*

    fields = ('title', 'slug', 'description', 'difficulty', 'field\_of\_study', 'tags', 'is\_published')

*# Автоматическая генерация поля slug на основе поля title.*

    prepopulated\_fields = {"slug": ("title",)}

*# Поля, которые доступны только для чтения*

    readonly\_fields = ['time\_create', 'time\_update']

    list\_display = (

        'id',

        'title',

        'field\_of\_study',

        'time\_create',

        'is\_published',

        brief\_info,

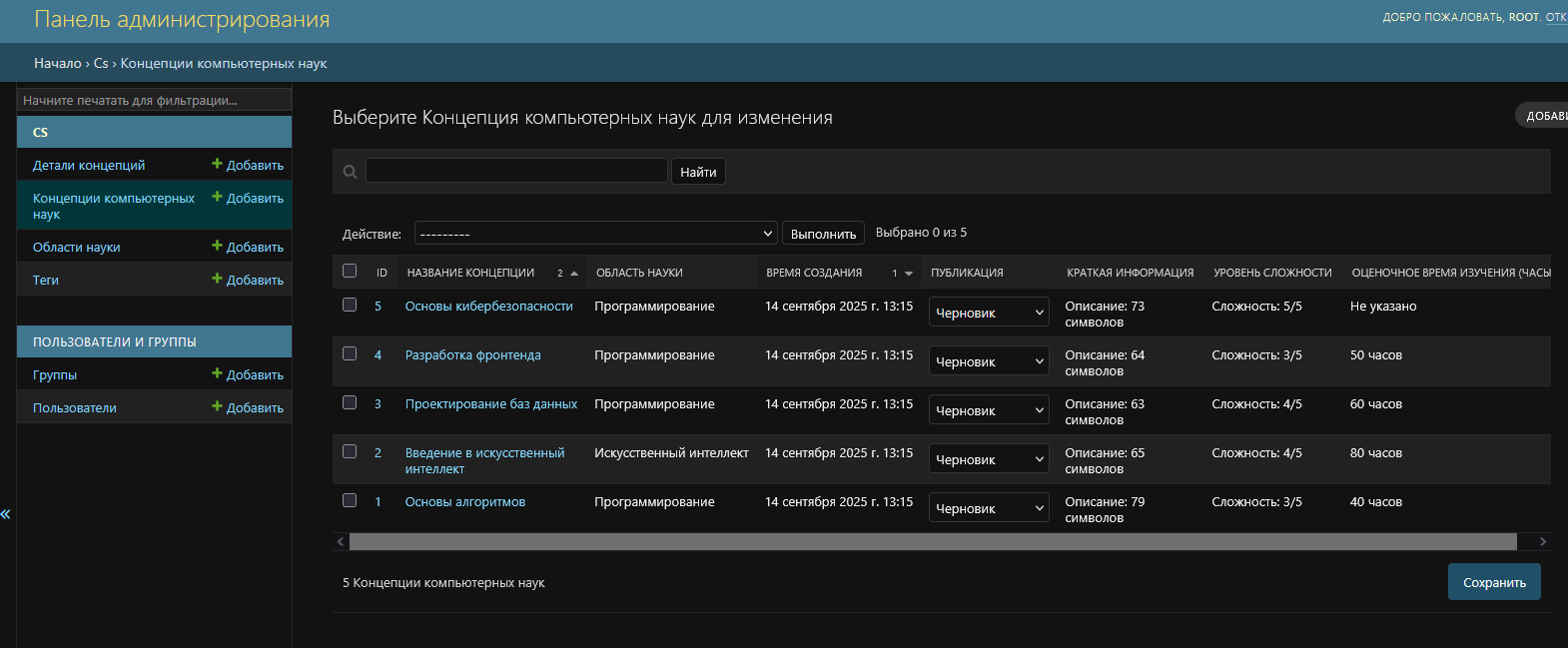
        difficulty\_level,

        estimated\_reading\_time,

    )

Функция difficulty\_level: Проверяет, задан ли уровень сложности концепции (obj.difficulty). Если значение присутствует, она форматирует его как строку "Сложность: X/5" (где X – это число от 1 до 5). В противном случае возвращает "Не указана". Декоратор @admin.display также задаёт подпись "Уровень сложности" для этого столбца.

Функция estimated\_reading\_time: Проверяет, есть ли у концепции связанные детали (obj.detail) и указано ли в них estimated\_learning\_time (примерное время изучения). Если эти данные доступны, функция возвращает строку, указывающую это время в часах. В противном случае возвращает "Не указано". Для этого столбца декоратор @admin.display устанавливает подпись "Оценочное время изучения (часы)".

  
Рисунок 9 – Пользовательские поля.

8. Добавить пользовательские действия (не менее двух). Продемонстрировать их использование. Действия должны сопровождаться выводом сообщений.

Файл admin.py:

*# Пользовательское действие для установки статуса "Опубликовано"*

    @admin.action(*description*="Опубликовать выбранные концепции")

    def set\_published(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*self*.message\_user(*request*, f"Статус 'Опубликовано' обновлён для {count} концепций.", messages.SUCCESS)

*# Пользовательское действие для установки статуса "Черновик"*

    @admin.action(*description*="Снять с публикации выбранные концепции")

    def set\_draft(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

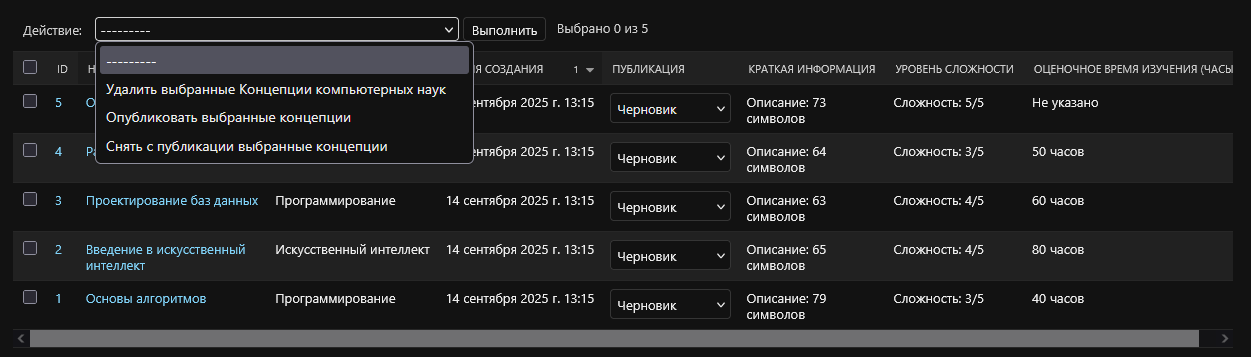
*self*.message\_user(*request*, f"{count} концепций сняты с публикации.", messages.WARNING)

    actions = ['set\_published', 'set\_draft']

В классе ComputerScienceConceptAdmin мы добавили два метода (пользовательских действия):

* set\_published – обновляет поле is\_published для выбранных концепций до значения ComputerScienceConcept.Status.PUBLISHED. Затем через self.message\_user выводится сообщение об успешном обновлении, указывающее, сколько концепций было опубликовано.
* set\_draft – изменяет поле is\_published для выбранных концепций на ComputerScienceConcept.Status.DRAFT. После этого выводится соответствующее сообщение с предупреждением (messages.WARNING) о том, сколько концепций было снято с публикации.

Атрибут actions включает обе функции, чтобы они отображались в выпадающем списке действий в админ-панели.

 Рисунок 10 – Пользовательские действия.

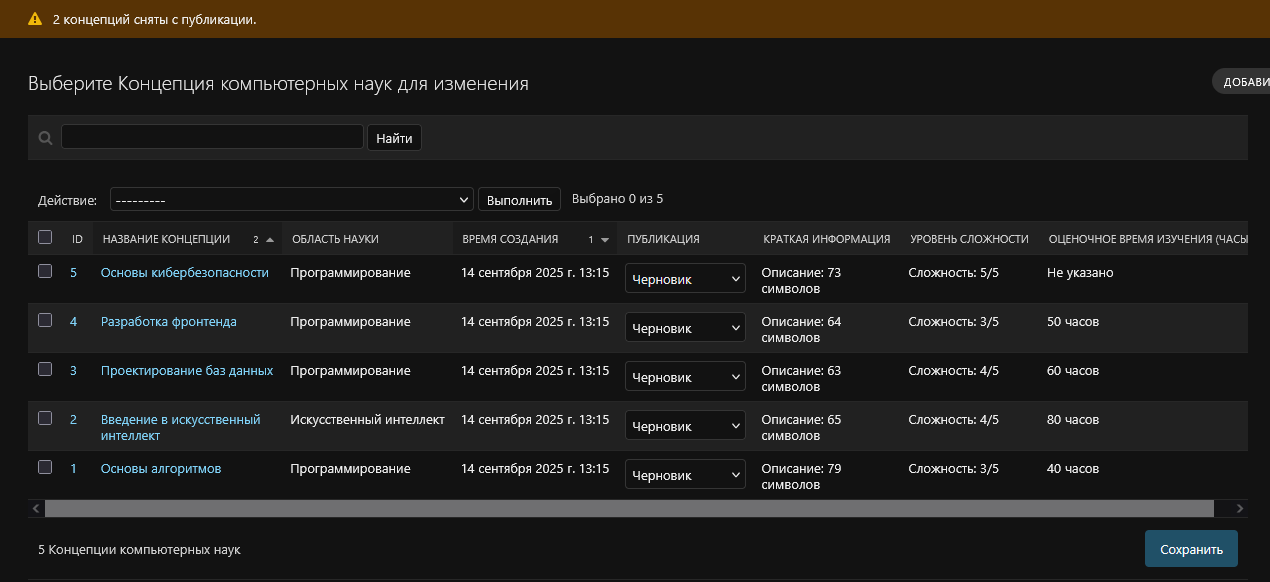


Рисунок 11 – Снятие публикаций.

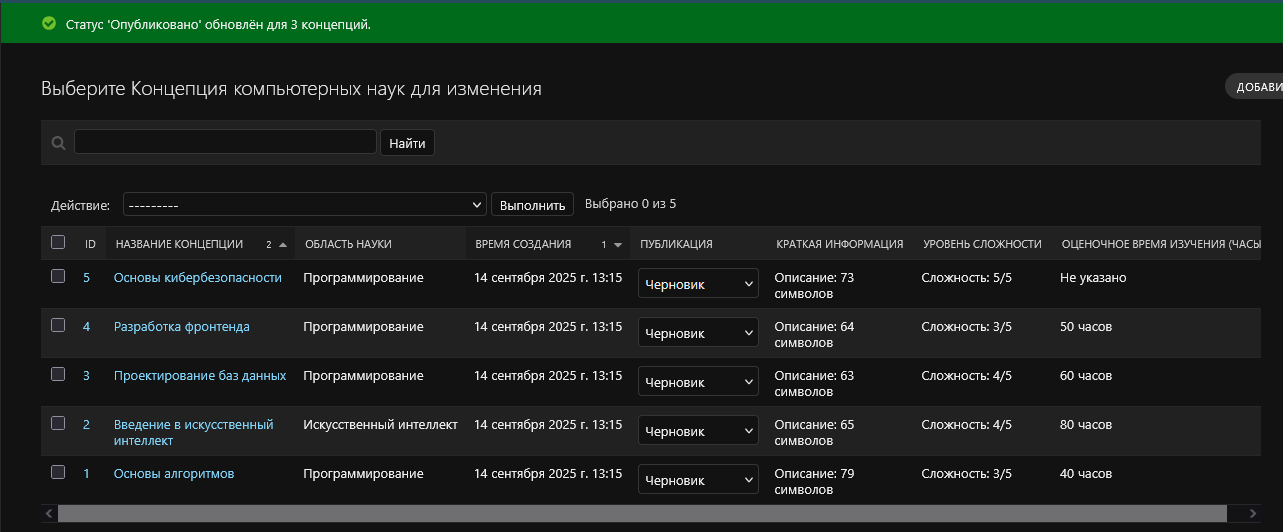


Рисунок 12 – Опубликовать запись.

9. Добавить и настроить панели поиска и фильтрации. Добавить собственный фильтр. Продемонстрировать работу панелей.

Файл admin.py:

*# Кастомный фильтр для статуса публикации*

class PublishedFilter(SimpleListFilter):

    title = "Статус публикации"

    parameter\_name = "pub\_status"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ("published", "Опубликовано"),

            ("draft", "Черновик"),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == "published":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*elif* *self*.value() == "draft":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*return* *queryset*

*# Дополнительный кастомный фильтр по диапазону сложности*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"  *# Заголовок фильтра в админ-панели*

    parameter\_name = "difficulty\_range"  *# Параметр в URL*

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ('easy', 'Легкая (1-2)'),

            ('medium', 'Средняя (3)'),

            ('hard', 'Сложная (4-5)'),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == 'easy':

*return* *queryset*.filter(*difficulty\_\_lte*=2)

*elif* *self*.value() == 'medium':

*return* *queryset*.filter(*difficulty*=3)

*elif* *self*.value() == 'hard':

*return* *queryset*.filter(*difficulty\_\_gte*=4)

*return* *queryset*

search\_fields = ('title', 'field\_of\_study\_\_name')

list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]

PublishedFilter наследуется от SimpleListFilter. Он добавляет дополнительную панель фильтрации в боковой части админ-панели для управления статусом публикации концепций.

В методе lookups возвращается кортеж с двумя вариантами:

Ключ 'published' соответствует концепциям со статусом "Опубликовано".

Ключ 'draft' соответствует концепциям со статусом "Черновик".

Метод queryset проверяет, какой ключ указан в URL-параметре:

Если выбран 'published', возвращается QuerySet с концепциями, у которых is\_published равно ComputerScienceConcept.Status.PUBLISHED.

Если выбран 'draft', возвращается QuerySet с концепциями, у которых is\_published равно ComputerScienceConcept.Status.DRAFT.

Если фильтр не выбран, возвращается исходный QuerySet без изменений.

DifficultyRangeFilter также наследуется от SimpleListFilter. Он добавляет дополнительную панель фильтрации в боковой части админ-панели, позволяя сортировать концепции по диапазону сложности.

В методе lookups возвращается кортеж с тремя вариантами:

Ключ 'easy' соответствует концепциям с уровнем сложности 1-2.

Ключ 'medium' соответствует концепциям с уровнем сложности 3.

Ключ 'hard' соответствует концепциям с уровнем сложности 4-5.

Метод queryset проверяет, какой ключ указан в URL-параметре:

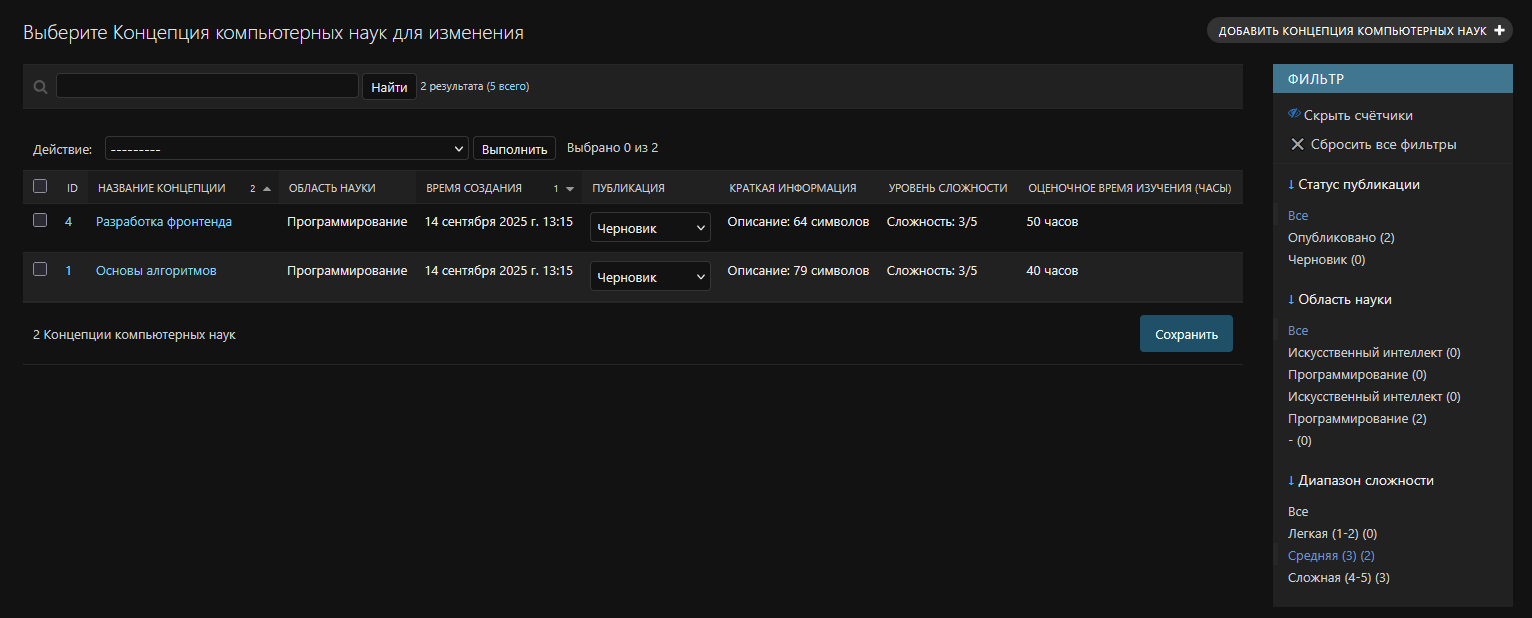
Если выбран 'easy', возвращается QuerySet с концепциями, у которых difficulty меньше или равно 2.

Если выбран 'medium', возвращается QuerySet с концепциями, у которых difficulty равно 3.

Если выбран 'hard', возвращается QuerySet с концепциями, у которых difficulty больше или равно 4.

Если фильтр не выбран, возвращается исходный QuerySet без изменений.

Для подключения этих фильтров в админ-панели мы добавляем их в список list\_filter класса ComputerScienceConceptAdmin: list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]

  
Рисунок 11 – Пользовательский фильтр выбранный в положение Диапазон сложности - средняя.

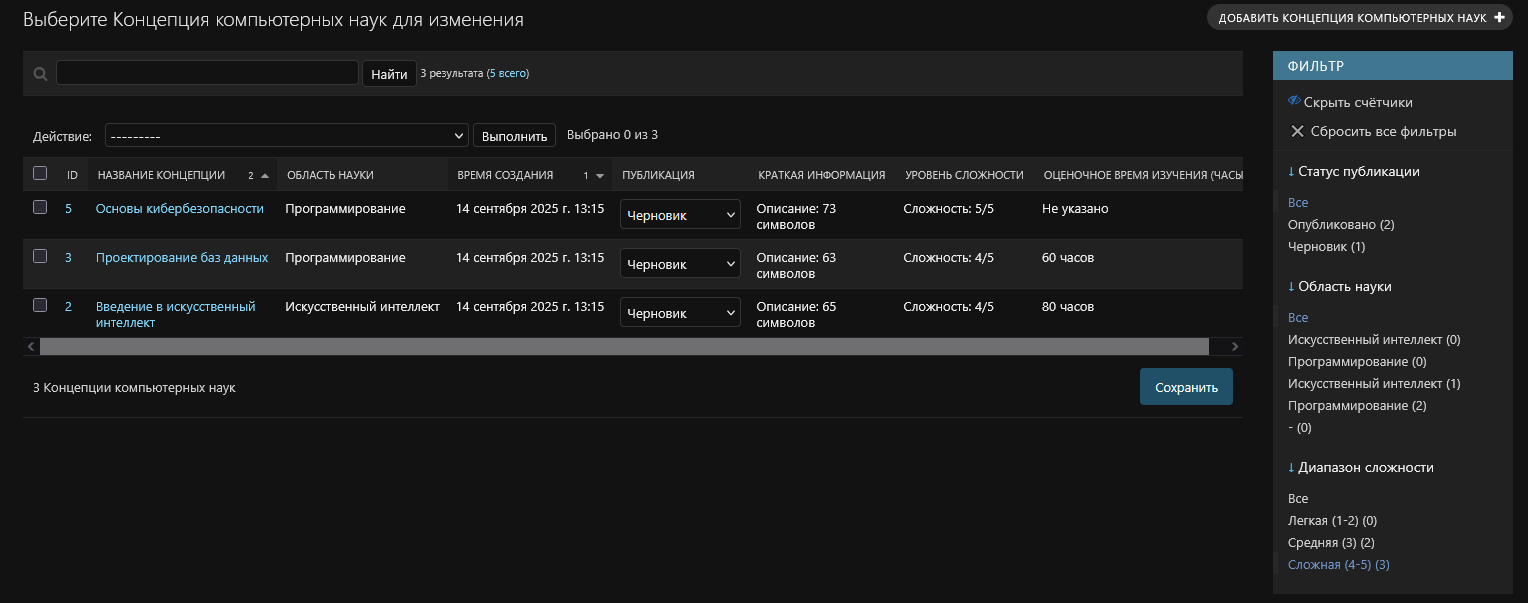


Рисунок 12 – Пользовательский фильтр выбранный в положение Диапазон сложности - сложная.

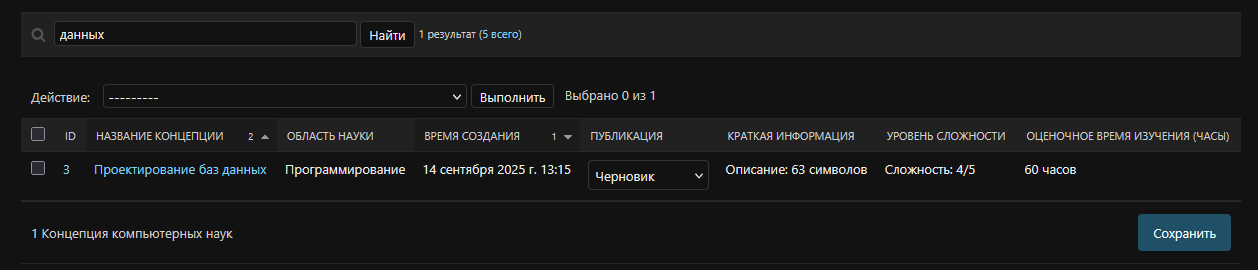


Рисунок 13 – Панель поиска.

10. Настроить формы добавления и редактирования записей в админ панели.

Файл admin.py:

actions = ['set\_published', 'set\_draft']

*# Настройка формы добавления/редактирования записей:*

*# Поля, отображаемые в форме, перечисляются в указанном порядке.*

    fields = ('title', 'slug', 'description', 'difficulty', 'field\_of\_study', 'tags', 'is\_published')

*# Автоматическая генерация поля slug на основе поля title.*

    prepopulated\_fields = {"slug": ("title",)}

*# Поля, которые доступны только для чтения*

    readonly\_fields = ['time\_create', 'time\_update']

Форма добавления/редактирования записей для модели ComputerScienceConcept настраивается с помощью атрибута fields.

Поля перечисляются в том порядке, в котором они будут выводиться на форме. В нашем примере в форме будут отображаться поля: title, slug, description, difficulty, field\_of\_study, tags и is\_published. Атрибут prepopulated\_fields автоматически генерирует значение для поля slug на основе поля title. Например, если вы введёте "Искусственный интеллект", slug сгенерируется как "iskusstvennyj-intellekt" (с учетом настроек).

Атрибут readonly\_fields задаёт поля, которые нельзя изменять в форме. Здесь поля time\_create и time\_update (время создания и обновления) доступны только для чтения и не могут быть изменены пользователем.

Остальные настройки (list\_display, list\_editable, search\_fields, list\_filter и actions) относятся к отображению списка записей, что мы уже настроили ранее и сохраняем для целостности интерфейса админ-панели.

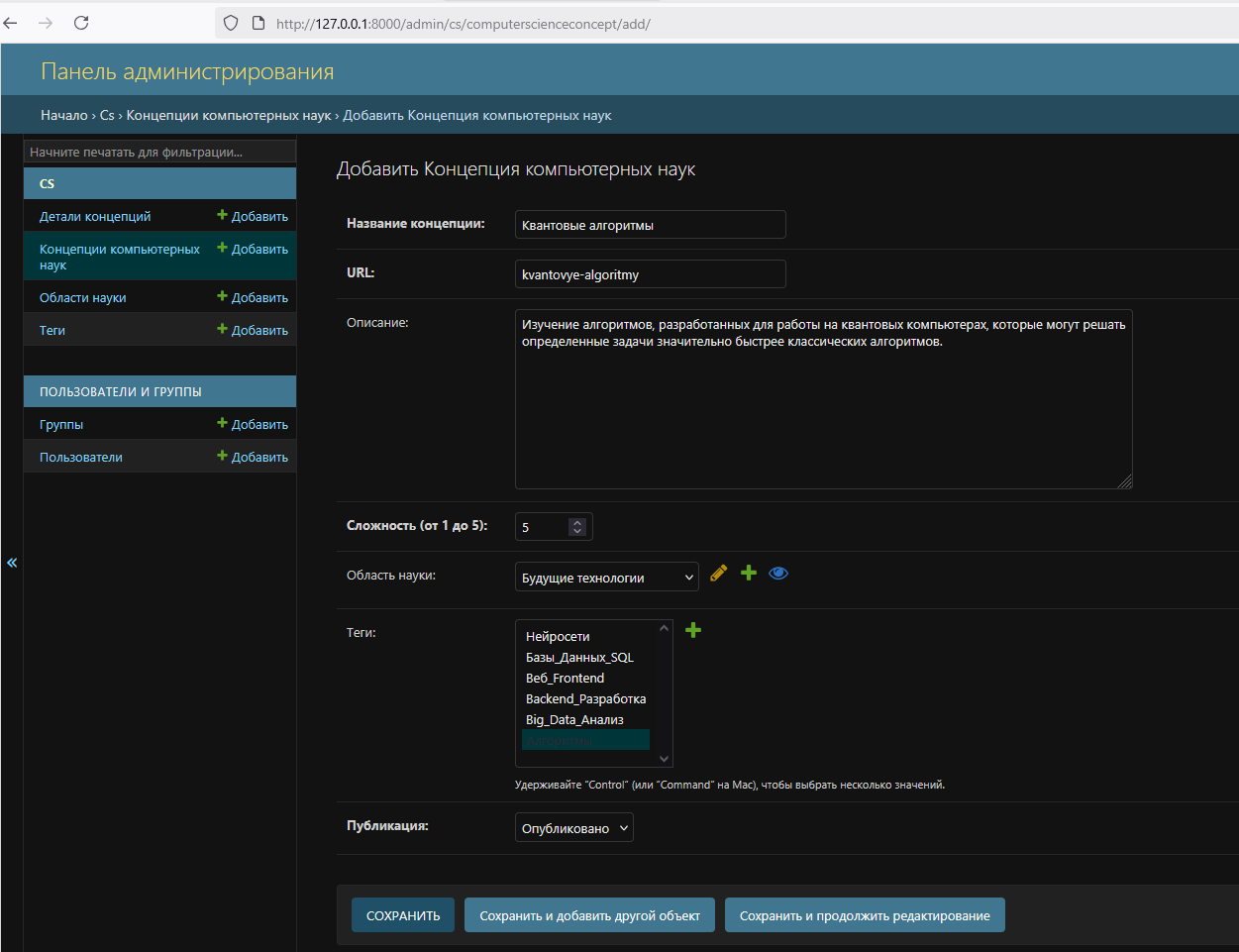
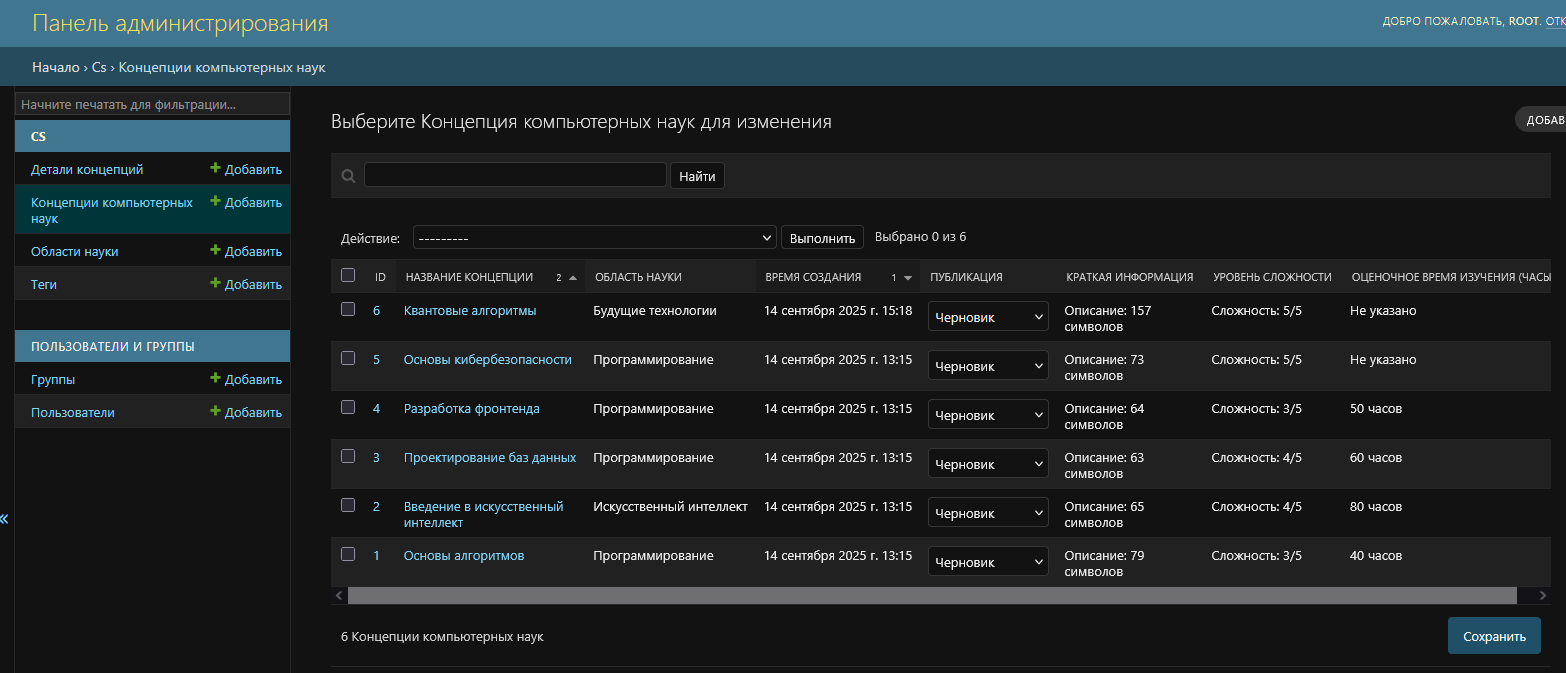
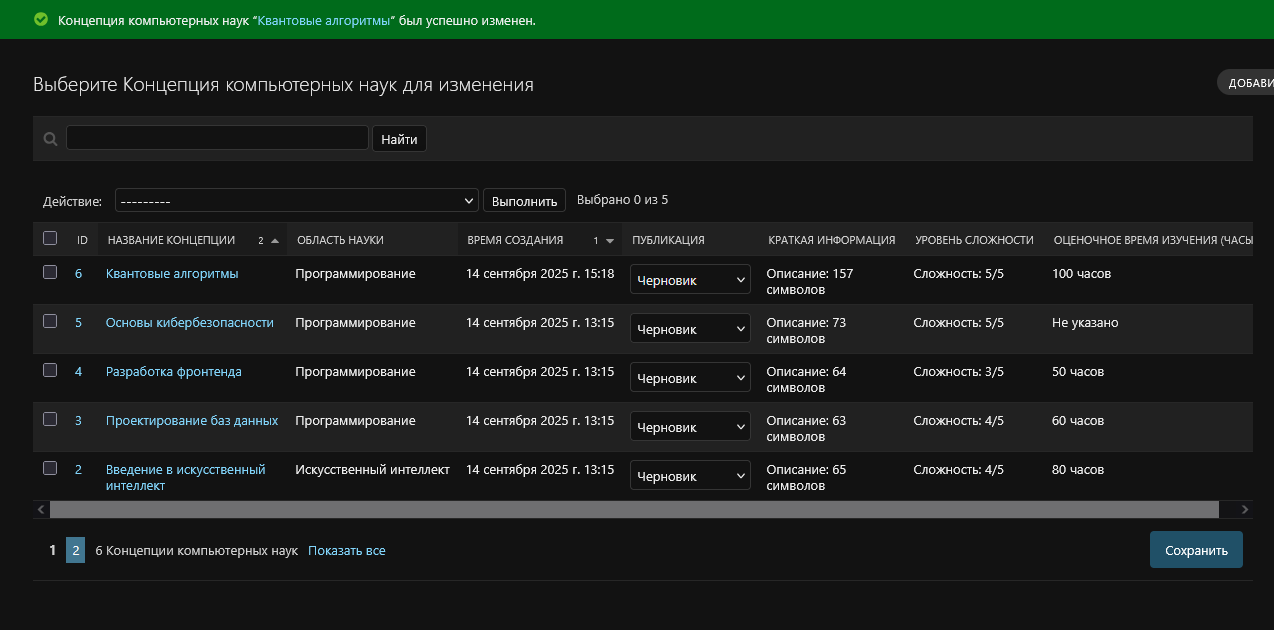
  
Рисунок 14 -Добавление новой концепции. 

Рисунок 15 – Обновленный список концепций.

  
Рисунок 16 – Изменение области науки.

11. Настроить внешний вид админ-панели в соответствии с оформлением сайта.

Добавим шаблон base\_site.html

{% extends "admin/base.html" %}

{% load static %}

{% block extrastyle %}

    <link rel="stylesheet" href="{% static 'admin/css/base.css' %}" />

    <link rel="stylesheet" href="{% static 'cs/css/admin.css' %}">

{% endblock %}

Cформируем файл admin.css в нашем приложении в каталоге static/css/admin относительно каталога нашего проекта.

body {

    background-color: #e0f2f7;

    font-family: 'Roboto', sans-serif;

}

#header {

    background-color: #3f51b5 !important;

    border-bottom: 3px solid #1a237e;

    box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

#header h1, #header a {

    color: #ffffff !important;

    font-size: 2.2em;

    font-weight: 700;

    text-shadow: 1px 1px 2px rgba(0,0,0,0.3);

}

.module h2, .module caption, .inline-group h2 {

    background-color: #5c6bc0;

    color: #ffffff;

    padding: 10px 15px;

    border-radius: 6px;

    margin-bottom: 12px;

    font-size: 1.3em;

    border-left: 5px solid #1a237e;

}

div.breadcrumbs {

    background-color: #7986cb;

    padding: 10px 15px;

    border-radius: 6px;

    margin-bottom: 20px;

    color: #ffffff;

    font-size: 1.05em;

}

.actions button {

    background-color: #e91e63 !important;

    color: #ffffff !important;

    border: none;

    padding: 8px 15px;

    margin: 4px;

    border-radius: 4px;

    font-size: 1em;

    text-transform: uppercase;

    font-weight: 500;

    transition: background-color 0.3s ease;

}

.actions button:hover {

    background-color: #c2185b !important;

}

.button.default, input[type=submit].default {

    background: #4caf50 !important;

    color: #ffffff !important;

}

.button.default:hover, input[type=submit].default:hover {

    background: #388e3c !important;

}

Чтобы фреймворк Django находил нестандартный путь к этой папке, ее нужно указать в настройках settings.py с помощью специальной коллекции:

STATIC\_URL = 'static/'

STATICFILES\_DIRS = [BASE\_DIR / 'cs' / 'static']

STATIC\_ROOT = BASE\_DIR / 'staticfiles'

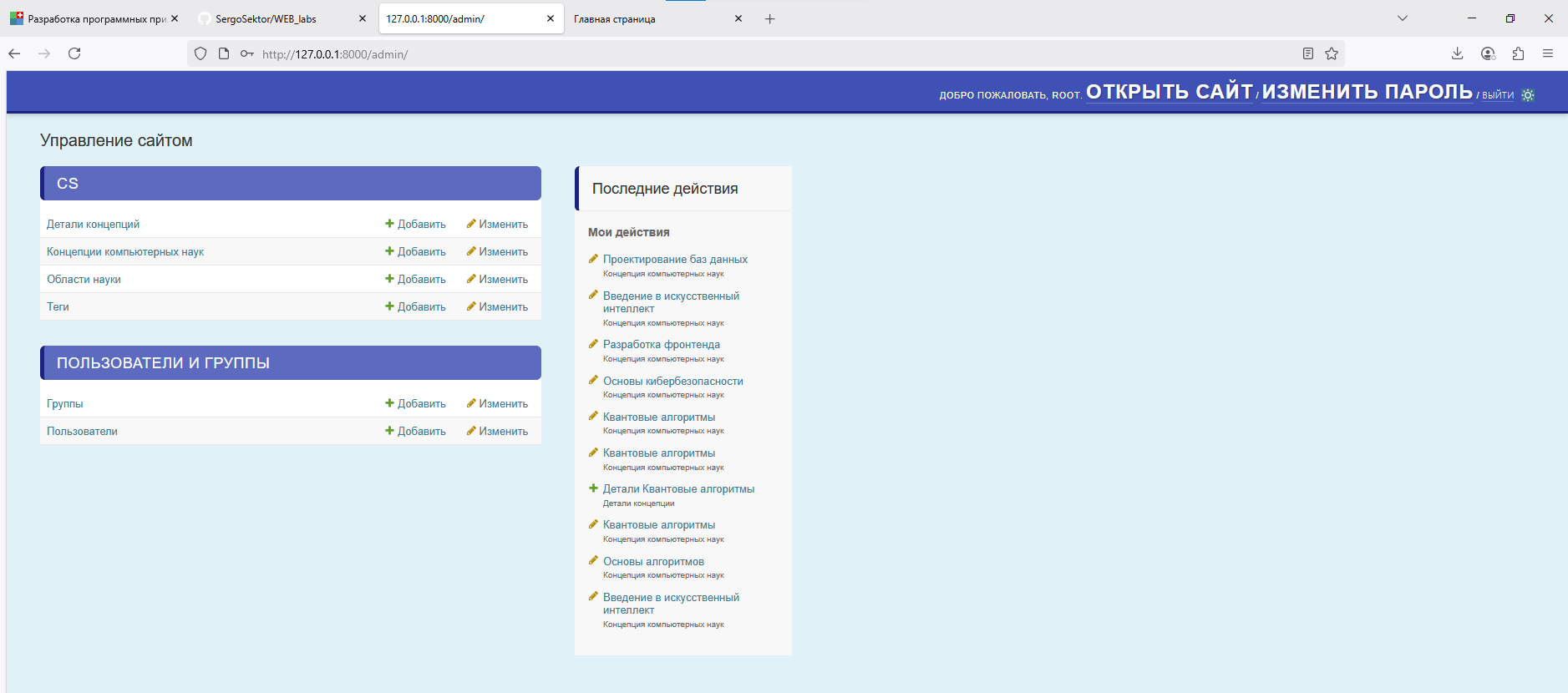


Рисунок 17 – Измененное оформление.

**Листинг программ**

**base\_site.html**

{% extends "admin/base.html" %}

{% load static %}

{% block extrastyle %}

    <link rel="stylesheet" href="{% static 'admin/css/base.css' %}" />

    <link rel="stylesheet" href="{% static 'cs/css/admin.css' %}">

{% endblock %}

**admin.css:**

body {

    background-color: #e0f2f7;

    font-family: 'Roboto', sans-serif;

}

#header {

    background-color: #3f51b5 !important;

    border-bottom: 3px solid #1a237e;

    box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

#header h1, #header a {

    color: #ffffff !important;

    font-size: 2.2em;

    font-weight: 700;

    text-shadow: 1px 1px 2px rgba(0,0,0,0.3);

}

.module h2, .module caption, .inline-group h2 {

    background-color: #5c6bc0;

    color: #ffffff;

    padding: 10px 15px;

    border-radius: 6px;

    margin-bottom: 12px;

    font-size: 1.3em;

    border-left: 5px solid #1a237e;

}

div.breadcrumbs {

    background-color: #7986cb;

    padding: 10px 15px;

    border-radius: 6px;

    margin-bottom: 20px;

    color: #ffffff;

    font-size: 1.05em;

}

.actions button {

    background-color: #e91e63 !important;

    color: #ffffff !important;

    border: none;

    padding: 8px 15px;

    margin: 4px;

    border-radius: 4px;

    font-size: 1em;

    text-transform: uppercase;

    font-weight: 500;

    transition: background-color 0.3s ease;

}

.actions button:hover {

    background-color: #c2185b !important;

}

.button.default, input[type=submit].default {

    background: #4caf50 !important;

    color: #ffffff !important;

}

.button.default:hover, input[type=submit].default:hover {

    background: #388e3c !important;

}

**admin.py:**

*from* django.contrib *import* admin, messages

*from* django.contrib.admin *import* SimpleListFilter

*from* .models *import* ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag

admin.site.site\_header = "Панель администрирования"

admin.site.index\_title = "Управление сайтом"

admin.site.site\_title = "Администрирование Computer Science Project"

*# Функция для вычисляемого поля "Краткая информация"*

@admin.display(*description*="Краткая информация")

def brief\_info(*obj*):

*return* f"Описание: {len(*obj*.description)} символов" *if* *obj*.description *else* "Нет описания"

*# Пользовательское вычисляемое поле: уровень сложности*

@admin.display(*description*="Уровень сложности")

def difficulty\_level(*obj*):

*if* *obj*.difficulty is not None:

*return* f"Сложность: {*obj*.difficulty}/5"

*return* "Не указана"

*# Пользовательское вычисляемое поле: оценочное время изучения (часы)*

@admin.display(*description*="Оценочное время изучения (часы)")

def estimated\_reading\_time(*obj*):

*if* hasattr(*obj*, 'detail') and *obj*.detail.estimated\_learning\_time is not None:

*return* f"{*obj*.detail.estimated\_learning\_time} часов"

*return* "Не указано"

*# Кастомный фильтр для статуса публикации*

class PublishedFilter(SimpleListFilter):

    title = "Статус публикации"

    parameter\_name = "pub\_status"

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ("published", "Опубликовано"),

            ("draft", "Черновик"),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == "published":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*elif* *self*.value() == "draft":

*return* *queryset*.filter(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*return* *queryset*

*# Дополнительный кастомный фильтр по диапазону сложности*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"  *# Заголовок фильтра в админ-панели*

    parameter\_name = "difficulty\_range"  *# Параметр в URL*

    def lookups(*self*, *request*, *model\_admin*):

*return* [

            ('easy', 'Легкая (1-2)'),

            ('medium', 'Средняя (3)'),

            ('hard', 'Сложная (4-5)'),

        ]

    def queryset(*self*, *request*, *queryset*):

*if* *self*.value() == 'easy':

*return* *queryset*.filter(*difficulty\_\_lte*=2)

*elif* *self*.value() == 'medium':

*return* *queryset*.filter(*difficulty*=3)

*elif* *self*.value() == 'hard':

*return* *queryset*.filter(*difficulty\_\_gte*=4)

*return* *queryset*

@admin.register(ComputerScienceConcept)

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Настройка формы добавления/редактирования записей:*

*# Поля, отображаемые в форме, перечисляются в указанном порядке.*

    fields = ('title', 'slug', 'description', 'difficulty', 'field\_of\_study', 'tags', 'is\_published')

*# Автоматическая генерация поля slug на основе поля title.*

    prepopulated\_fields = {"slug": ("title",)}

*# Поля, которые доступны только для чтения*

    readonly\_fields = ['time\_create', 'time\_update']

    list\_display = (

        'id',

        'title',

        'field\_of\_study',

        'time\_create',

        'is\_published',

        brief\_info,

        difficulty\_level,

        estimated\_reading\_time,

    )

    list\_display\_links = ('id', 'title')

    list\_editable = ('is\_published',)

    ordering = ['-time\_create', 'title']

    search\_fields = ('title', 'field\_of\_study\_\_name')

    list\_filter = [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]

    list\_per\_page = 5

*# Пользовательское действие для установки статуса "Опубликовано"*

    @admin.action(*description*="Опубликовать выбранные концепции")

    def set\_published(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

*self*.message\_user(*request*, f"Статус 'Опубликовано' обновлён для {count} концепций.", messages.SUCCESS)

*# Пользовательское действие для установки статуса "Черновик"*

    @admin.action(*description*="Снять с публикации выбранные концепции")

    def set\_draft(*self*, *request*, *queryset*):

        count = *queryset*.update(*is\_published*=ComputerScienceConcept.Status.DRAFT)

*self*.message\_user(*request*, f"{count} концепций сняты с публикации.", messages.WARNING)

    actions = ['set\_published', 'set\_draft']

@admin.register(FieldOfStudy)

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

@admin.register(ConceptDetail)

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'estimated\_learning\_time')

    list\_display\_links = ('id', 'concept')

    search\_fields = ('concept\_\_title', 'core\_technologies')

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

    list\_display\_links = ('id', 'name')

    search\_fields = ('name',)

**urls.py:**

"""

URL configuration for cs\_ty project.

The `urlpatterns` list routes URLs to views. For more information please see:

    https://docs.djangoproject.com/en/4.2/topics/http/urls/

Examples:

Function views

    1. Add an import:  from my\_app import views

    2. Add a URL to urlpatterns:  path('', views.home, name='home')

Class-based views

    1. Add an import:  from other\_app.views import Home

    2. Add a URL to urlpatterns:  path('', Home.as\_view(), name='home')

Including another URLconf

    1. Import the include() function: from django.urls import include, path

    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))

"""

*from* django.contrib *import* admin

*from* django.urls *import* path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('cs.urls')),

]

**Вывод**

В ходе работы была детально изучена административная панель Django и продемонстрированы возможности её настройки для управления контентом проекта. Сначала мы переключили язык интерфейса админки на русский, изменив значение константы LANGUAGE\_CODE в файле настроек, что позволило отобразить все элементы панели на русском языке. Затем был создан суперпользователь с помощью команды createsuperuser, после чего мы успешно вошли в админ-панель и убедились в наличии стандартных разделов (пользователи, группы) и зарегистрированных нами моделей.

Далее мы зарегистрировали модели приложения (ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag) в файле admin.py. Для модели ComputerScienceConcept были настроены отображаемые столбцы (list\_display) с указанием таких полей, как id, название, область науки, время создания, статус публикации, а также добавлены три вычисляемых поля – краткая информация (на основе длины описания), уровень сложности и оценочное время изучения. Кроме того, была настроена возможность редактирования некоторых полей прямо из списка, задан порядок сортировки и установлена пагинация.

После этого мы добавили пользовательские действия, позволяющие массово изменять статус публикации выбранных концепций (переключать между «Черновик» и «Опубликовано») с выводом соответствующих сообщений. Мы внедрили эти действия с использованием декораторов в администрировании модели ComputerScienceConcept, что позволило выбрать несколько записей и изменить их статус, при этом система сообщает, сколько концепций было обновлено.

Далее в работе настраивались панели поиска и фильтрации. В админ-панели добавлены стандартные фильтры (например, по внешнему ключу «Область науки») и собственный фильтр для статуса публикации, а также дополнительный пользовательский фильтр для диапазонов сложности. Эти фильтры позволяют быстро отсеивать записи по нужным критериям, а панель поиска даёт возможность фильтровать записи по заголовкам и связанным полям. Таким образом, мы продемонстрировали возможность комбинирования методов фильтрации в Django.

Наконец, для удобства работы с формами добавления и редактирования записей в админке мы настроили поля, отображаемые в форме, их порядок и автозаполнение слага на основе названия. При этом поля для автоматической регистрации времени создания и обновления были объявлены только для чтения, что предотвращает их случайное изменение.

В результате лабораторная работа продемонстрировала, как с помощью встроенных инструментов Django можно не только управлять базовыми данными, но и глубоко настраивать административный интерфейс. Все изменения – от регистрации моделей до добавления кастомных действий и фильтров, а также настройки форм – позволяют администратору эффективно управлять контентом, проводить массовые операции и адаптировать внешний вид панели под визуальный стиль основного сайта. Эти изменения подтверждаются визуально через обновлённые заголовки, новый интерфейс списка записей, удобные элементы поиска и фильтрации, а также через корректное отображение форм в админ-панели.