МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

Изображение выглядит как Графика, Красочность, снимок экрана, графический дизайн

Автоматически созданное описание

**ОТЧЁТ**

**по** расчетно-графическому заданию

**на тему:** Разработка веб-сайта на фреймворке django

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

Выполнили работу:

Студенты гр. АП-226, АВТФ

Ищенко С. Г. Александрова Лилия Александровна

«10» июня 2025г

Проверил работу:

Старший преподаватель каф. АСУ

Эстрайх И.В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г

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

(подпись)

**Оглавление**

[Введение 3](#_Toc209044878)

[Описание приложения и деталей его реализации 5](#_Toc209044879)

[Заключение 26](#_Toc209044880)

[Cпискок литературы 27](#_Toc209044881)

[Приложение А. Листинг программы 28](#_Toc209044882)

# **Введение**

Данное веб-приложение представляет собой сервис по управлению компьютерными науками, разработанный на фреймворке Django.

Приложение включает в себя комплексное управление пользователями, начиная с регистрации и аутентификации (с возможностью входа по логину или email) до полного контроля над профилем, включая изменение личных данных и смену пароля. Для обеспечения удобства и безопасности реализована система восстановления доступа через электронную почту, интегрированная с SMTP Яндекса для отправки реальных писем со ссылкой на форму сброса пароля.

Основное приложение, посвященное компьютерным наукам, содержит всю логику для создания, отображения и обсуждения различных концепций. В его основе лежит модель данных "Concept", представляющая собой центральную сущность с названием, описанием, изображением и временными метками создания/обновления. Каждая концепция уникально связана с пользователем-автором и имеет статус публикации (is\_published), что позволяет отображать в публичном доступе только одобренные записи. Для интерактивности на страницах деталей концепций предусмотрены комментарии (модель "Comment").

В целях унификации навигации и повторного использования кода, общее меню и логика постраничной навигации вынесены в миксин DataMixin. Основные пользовательские страницы, такие как главная страница с пагинацией, статичная страница "О нас", общий список концепций и детальное представление каждой концепции, реализованы с использованием классов ListView, DetailView и TemplateView. На странице детального представления, помимо информации о самой концепции, отображается блок комментариев.

Добавление и редактирование концепций может осуществляться двумя способами: через каркасную форму (FormView) и с помощью ModelForm (CreateView/UpdateView). При этом система автоматически назначает автора и сразу же публикует новую запись. Возможность редактировать и удалять концепции строго ограничена собственными записями пользователя, что контролируется методом get\_queryset в классах UpdateView и DeleteView.

Административная панель приложения предоставляет широкие возможности для управления концепциями. Она включает в себя список всех записей с вычисляемыми полями ("краткая информация", "цена с налогом"), мощные фильтры по статусу, производителю и ценовому диапазону, а также удобный поиск по названию и автору. Для массового управления предусмотрены действия "Опубликовать" и "Снять с публикации". Гибкая система прав доступа реализована с помощью пользовательского разрешения can\_publish\_concept и групп Editors/Viewers на уровне Django-shell.

Архитектура приложения построена на модульном принципе, где каждый компонент выполняет свою специализированную функцию: models.py описывает структуру базы данных, forms.py отвечает за валидацию ввода, views.py содержит логику обработки запросов, templates/ обеспечивает отображение данных, а urls.py управляет маршрутизацией. Использование классов-представлений (class-based views), миксинов и стандартных компонентов Django гарантирует чистоту, расширяемость и устойчивость кода к изменениям, а продуманная структура шаблонов обеспечивает консистентный и удобный интерфейс как для конечных пользователей, так и для разработчиков.

# **Описание приложения и деталей его реализации**

1. Проектирование базы данных

Задача: определить, какие сущности понадобятся и как они связаны.

Основная модель ComputerScienceConcept. В файле cs/models.py мы описали класс ComputerScienceConcept, в котором хранится вся информация об концепции:

*from* django.db *import* models

*from* django.urls *import* reverse

*from* django.utils.text *import* slugify

*from* django.contrib.auth *import* get\_user\_model

User = get\_user\_model()

class ComputerScienceConcept(models.Model):

    class Status(models.IntegerChoices):

        DRAFT = 0, 'Черновик'

        PUBLISHED = 1, 'Опубликовано'

    title        = models.CharField(*max\_length*=255, *verbose\_name*="Название концепции")

    slug         = models.SlugField(*max\_length*=255, *unique*=True, *db\_index*=True, *verbose\_name*="URL")

    description  = models.TextField(*blank*=True, *verbose\_name*="Описание")

    difficulty   = models.IntegerField(default=1, *verbose\_name*="Сложность (от 1 до 5)")

    image        = models.ImageField(*upload\_to*='concept\_images/%Y/%m/%d/', *blank*=True, *null*=True, *verbose\_name*='Изображение')

    time\_create  = models.DateTimeField(*auto\_now\_add*=True, *verbose\_name*="Время создания")

    time\_update  = models.DateTimeField(*auto\_now*=True, *verbose\_name*="Время обновления")

    is\_published = models.BooleanField(*choices*=Status.choices, default=Status.DRAFT, *verbose\_name*="Публикация")

    class Meta:

        permissions = [

            ('can\_publish\_concept', 'Может публиковать концепцию'),

        ]

        ordering = ['-time\_create']

        verbose\_name = "Концепция компьютерных наук"

        verbose\_name\_plural = "Концепции компьютерных наук"

    def \_\_str\_\_(*self*):

*return* *self*.title

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.title)

            base\_slug = *self*.slug

            num = 1

*while* ComputerScienceConcept.objects.filter(*slug*=*self*.slug).exists():

*self*.slug = f"{base\_slug}-{num}"

                num += 1

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('cs:concept\_detail', *kwargs*={'concept\_slug': *self*.slug})

- title и description предоставляют основную информацию о концепции.

- is\_published — логическое поле, определяющее, видна ли концепция в публичной части сайта.

- difficulty — поле для оценки сложности концепции по шкале от 1 до 5.

- Присутствует поле image для загрузки изображений, связанных с концепцией.

- time\_create и time\_update автоматически фиксируют время создания и последнего обновления записи.

- Разрешение can\_publish\_concept позволяет гибко управлять правами на публикацию концепций.

- Концепции могут быть связаны с FieldOfStudy (Область науки) через внешний ключ.

Справочники: FieldOfStudy и Tag

Для классификации и организации концепций используются следующие справочные модели:

class FieldOfStudy(models.Model):

    name = models.CharField(*max\_length*=255, *verbose\_name*="Область науки")

    slug = models.SlugField(*max\_length*=255, *unique*=True, *db\_index*=True, *verbose\_name*="URL")

    description = models.TextField(*blank*=True, *verbose\_name*="Описание области")

    class Meta:

        verbose\_name = "Область науки"

        verbose\_name\_plural = "Области науки"

    def \_\_str\_\_(*self*):

*return* *self*.name

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.name)

            base\_slug = *self*.slug

            num = 1

*while* FieldOfStudy.objects.filter(*slug*=*self*.slug).exists():

*self*.slug = f"{base\_slug}-{num}"

                num += 1

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('cs:field\_of\_study\_detail', *kwargs*={'field\_of\_study\_slug': *self*.slug})

class Tag(models.Model):

    name = models.CharField(*max\_length*=50, *unique*=True, *verbose\_name*="Тег")

    slug = models.SlugField(*max\_length*=50, *unique*=True, *db\_index*=True, *verbose\_name*="URL тега")

    class Meta:

        verbose\_name = "Тег"

        verbose\_name\_plural = "Теги"

    def \_\_str\_\_(*self*):

*return* *self*.name

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.name)

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('concepts\_by\_tag', *kwargs*={'tag\_slug': *self*.slug})

*# Связь ManyToManyField между ComputerScienceConcept и Tag*

ComputerScienceConcept.add\_to\_class('tags', models.ManyToManyField(

    Tag,

*related\_name*='concepts',

*verbose\_name*="Теги",

*blank*=True

))

Модель Comment  
Для обеспечения интерактивности и возможности обратной связи пользователей, в проекте предусмотрена модель Comment:

*from* django.contrib.auth *import* get\_user\_model

User = get\_user\_model()

class Comment(models.Model):

    concept   = models.ForeignKey(ComputerScienceConcept, *on\_delete*=models.CASCADE, *related\_name*='comments', *verbose\_name*='Концепция')

    author  = models.ForeignKey(User, *on\_delete*=models.CASCADE, *related\_name*='comments', *verbose\_name*='Автор')

    text    = models.TextField(*verbose\_name*='Комментарий')

    created = models.DateTimeField(*auto\_now\_add*=True)

    class Meta:

        ordering = ['created']

        verbose\_name = "Комментарий"

        verbose\_name\_plural = "Комментарии"

    def \_\_str\_\_(*self*):

*return* f"Комментарий от {*self*.author} к {*self*.concept.title}"

- concept связывает комментарий с конкретной концепцией ComputerScienceConcept.

- author указывает на пользователя, оставившего комментарий.

- text содержит содержимое комментария.

- created автоматически фиксирует время создания комментария.

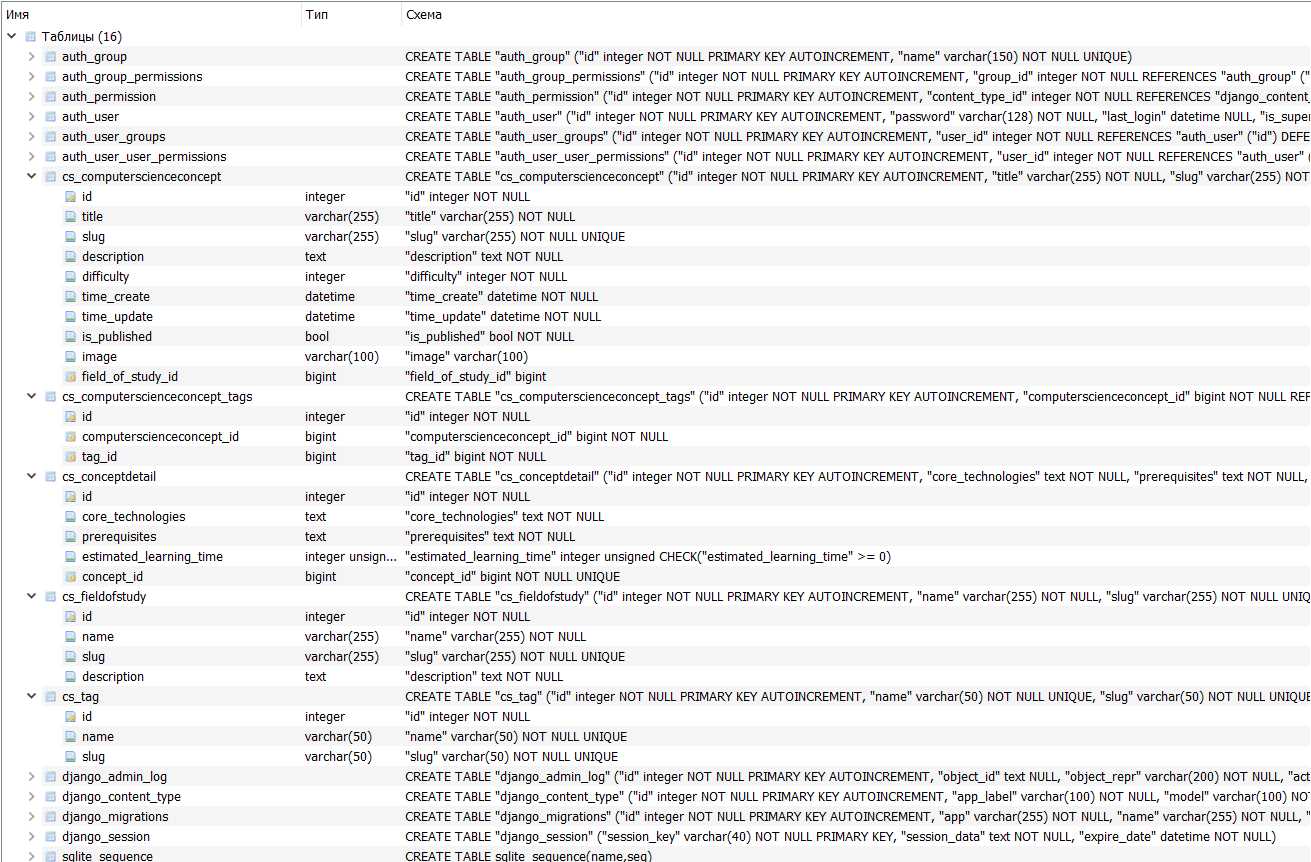


Рисунок 1 – Структура базы данных.

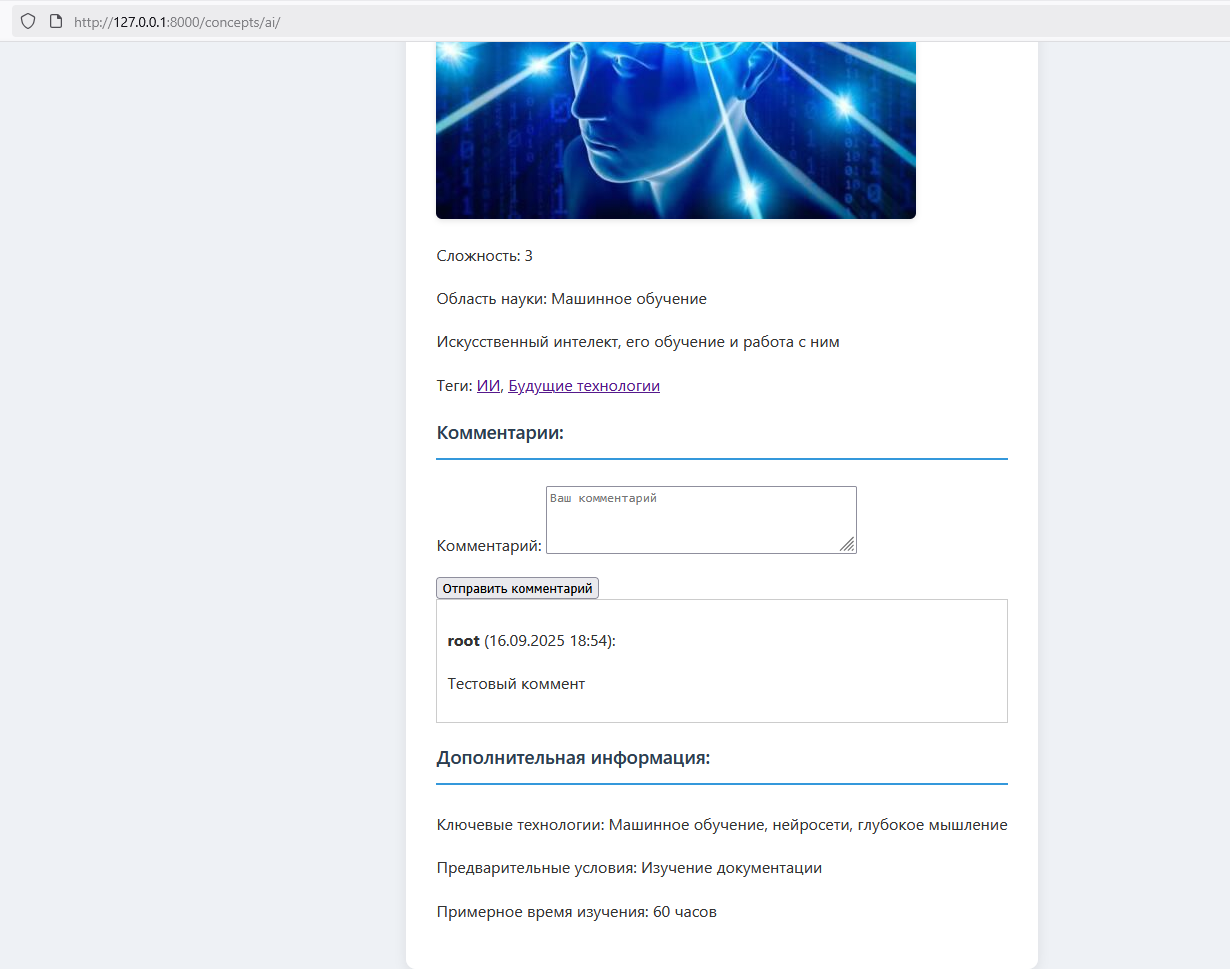


Рисунок 2 – Комментарии.

Маршрутизация

Задача: связать URL-адреса с представлениями приложения.

Корневой urls.py  
*Файл: cs\_ty/urls.py*

*from* django.contrib *import* admin

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

*from* django.conf *import* settings

*from* django.conf.urls.static *import* static

urlpatterns = [

    path('', include('cs.urls')),

    path('admin/', admin.site.urls),

    path('users/', include('users.urls', *namespace*='users')),

]

*if* settings.DEBUG:

    urlpatterns += static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT)

Маршруты приложения cs  
*Файл: cs/urls.py:*

*from* django.urls *import* path

*from* .views *import* (

    HomeView, AboutView, ConceptDetailView,

    AddConceptCustomView, ConceptCreateView,

    ConceptUpdateView, ConceptDeleteView, UploadFileView, FieldOfStudyDetailView, ConceptByTagListView

)

app\_name = 'cs'

urlpatterns = [

    path('',              HomeView.as\_view(),       *name*='home'),

    path('about/',        AboutView.as\_view(),      *name*='about'),

    path('concepts/<slug:concept\_slug>/', ConceptDetailView.as\_view(), *name*='concept\_detail'),

    path('add-custom/',   AddConceptCustomView.as\_view(), *name*='add\_concept\_custom'),

    path('add-model/',    ConceptCreateView.as\_view(),   *name*='add\_concept\_model'),

    path('edit/<slug:concept\_slug>/',   ConceptUpdateView.as\_view(), *name*='edit\_concept'),

    path('delete/<slug:concept\_slug>/', ConceptDeleteView.as\_view(), *name*='delete\_concept'),

    path('upload/',       UploadFileView.as\_view(),  *name*='upload\_file'),

    path('field/<slug:field\_of\_study\_slug>/', FieldOfStudyDetailView.as\_view(), *name*='field\_of\_study\_detail'),

    path('tag/<slug:tag\_slug>/', ConceptByTagListView.as\_view(), *name*='concepts\_by\_tag'),

]

* Статические пути (about/, add/, edit/, delete/, upload/) расположены **до** детального <slug>/, чтобы не было конфликтов.
* Порядок важен: детальный просмотр всегда последним.

Маршруты приложения users  
*Файл: users/urls.py*

*from* django.urls *import* path

*from* . *import* views

app\_name = 'users'

urlpatterns = [

    path('login/',     views.login\_user,  *name*='login'),

    path('logout/',    views.logout\_user, *name*='logout'),

*# Сброс пароля:*

    path('password-reset/',

         views.CustomPasswordResetView.as\_view(),

*name*='password\_reset'),

    path('password-reset/done/',

         views.CustomPasswordResetDoneView.as\_view(),

*name*='password\_reset\_done'),

    path('reset/<uidb64>/<token>/',

         views.CustomPasswordResetConfirmView.as\_view(),

*name*='password\_reset\_confirm'),

    path('reset/done/',

         views.CustomPasswordResetCompleteView.as\_view(),

*name*='password\_reset\_complete'),

*# Профиль и смена пароля:*

    path('profile/',

         views.ProfileView.as\_view(),

*name*='profile'),

    path('password-change/',

         views.CustomPasswordChangeView.as\_view(),

*name*='password\_change'),

    path('password-change/done/',

         views.CustomPasswordChangeDoneView.as\_view(),

*name*='password\_change\_done'),

]

Представления и шаблоны

Задача: реализовать логику обработки запросов и отрисовку данных.

Общий миксин

*Файл: cs/utils.py*

*from* django.views.generic.base *import* ContextMixin

MENU = [

    {'title': 'Главная',              'url\_name': 'cs:home'},

    {'title': 'О сайте',              'url\_name': 'cs:about'},

    {'title': 'Добавить (Form)',      'url\_name': 'cs:add\_concept\_custom'},

    {'title': 'Добавить (ModelForm)', 'url\_name': 'cs:add\_concept\_model'},

    {'title': 'Загрузка файла',       'url\_name': 'cs:upload\_file'},

]

class DataMixin(ContextMixin):

    """

    Добавляет в контекст:

      - global menu

      - title, если задано

      - page\_range для ListView с пагинацией

    """

    title = None

    def get\_context\_data(*self*, \*, *object\_list*=None, \*\**kwargs*):

*# Сначала получаем весь контекст от родительских классов,*

*# в том числе paginator, page\_obj, object\_list*

        context = super().get\_context\_data(*object\_list*=*object\_list*, \*\**kwargs*)

*# Базовый контекст*

        context['menu'] = MENU

*if* *self*.title:

            context['title'] = *self*.title

*# Если есть пагинация — вычисляем ограниченный диапазон страниц*

        paginator = context.get('paginator')

        page\_obj = context.get('page\_obj')

*if* paginator and page\_obj:

            total = paginator.num\_pages

            current = page\_obj.number

            window = 2

            start = max(current - window, 1)

            end   = min(current + window, total)

            context['page\_range'] = range(start, end + 1)

*return* context

Внедряет в любой шаблон пункты меню и заголовок страницы, чтобы избежать копирования меню во все views и templates.

CRUD для cs

HomeView (список с пагинацией) в файле cs/views.py

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

*from* .forms *import* ConceptForm, ConceptModelForm, UploadForm, CommentForm

*from* .utils *import* DataMixin

class HomeView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/index.html'

    context\_object\_name = 'concepts'

    paginate\_by = 5

    queryset = ComputerScienceConcept.published.all()

    title = 'Главная'

* Выводит только концепции со статусом “Опубликовано”.
* paginate\_by=5 ограничивает число концепций на странице.

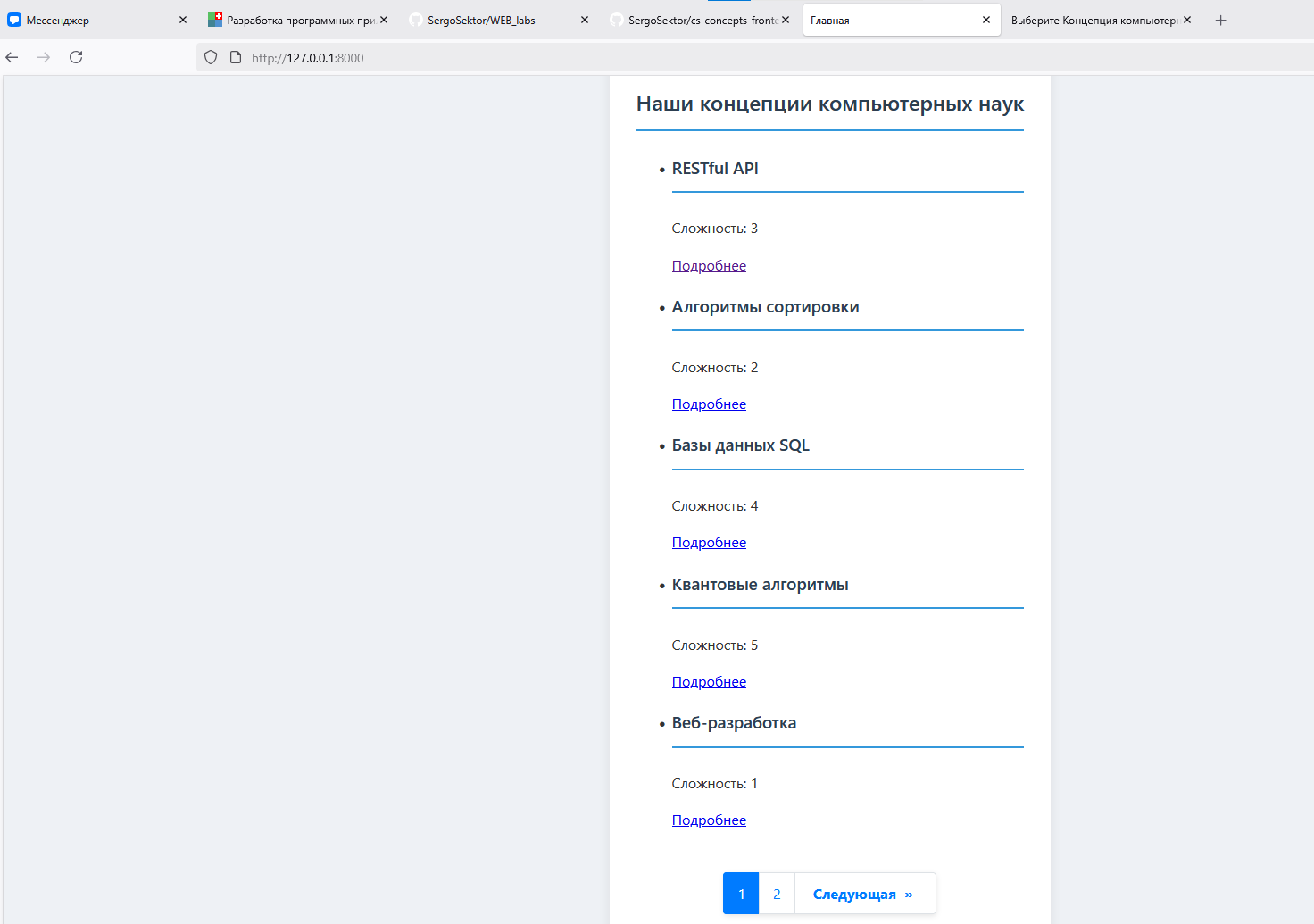


Рисунок 3 – Пагинация.

ConceptDetailView (просмотр + добавление комментариев)

class ConceptDetailView(DataMixin, DetailView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_detail.html'

    context\_object\_name = 'concept'

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

        concept = *self*.get\_object()

        context['title'] = concept.title

        context['comments'] = concept.comments.all()

        context['comment\_form'] = CommentForm()

*return* context

    def post(*self*, *request*, \**args*, \*\**kwargs*):

*self*.object = *self*.get\_object()

        form = CommentForm(*request*.POST)

*if* form.is\_valid():

            comment = form.save(*commit*=False)

            comment.concept = *self*.object

            comment.author = *request*.user

            comment.save()

*return* *self*.get(*request*, \**args*, \*\**kwargs*) *# Перенаправляем на GET запрос после успешной отправки*

*else*:

*# Если форма недействительна, возвращаем ее с ошибками*

            context = *self*.get\_context\_data()

            context['comment\_form'] = form

*return* *self*.render\_to\_response(context)

* GET: собирает концепцию, связанные комментарии и форму.
* POST: сохраняет новый комментарий и делает PRG-redirect.

AddConceptCustomView (ручная форма):

class AddConceptCustomView(LoginRequiredMixin, DataMixin, FormView):

    form\_class = ConceptForm

    template\_name = 'cs/add\_concept\_custom.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (Form)'

    login\_url = reverse\_lazy('users:login')

    def form\_valid(*self*, *form*):

        cd = *form*.cleaned\_data

        concept = ComputerScienceConcept.objects.create(

*title*=cd['title'],

*slug*=cd['title'].lower().replace(' ', '-'),

*description*=cd['description'],

*difficulty*=cd['difficulty'],

*is\_published*=ComputerScienceConcept.Status.DRAFT

        )

        image = cd.get('image')

*if* image:

            concept.image = image

            concept.save()

*return* super().form\_valid(*form*)

Редактирование и удаление

* ConceptUpdateView/ConceptDeleteView наследуются от UpdateView/DeleteView
* Переопределяют get\_queryset, чтобы включить фильтр author=self.request.user.
* LoginRequiredMixin защищает от анонимного доступа.
*  **LoginRequiredMixin** требует входа пользователя.
* **form\_valid** автоматически устанавливает автора и публикует новую запись.
* **get\_queryset** гарантирует, что редактировать/удалять можно только свои записи.

Теги и категории

Дают возможность выбирать производителя и теги при создании/редактировании концепции.

Форма включила поля:

class ConceptModelForm(forms.ModelForm):

class Meta:

        model = ComputerScienceConcept

        fields = [

            'title',

            'slug',

            'description',

            'difficulty',

            'field\_of\_study',

            'tags',

            'image',

        ]

        widgets = {

            'description': forms.Textarea,

        }

Шаблон add\_concept\_model.html:

<p>

      {{ form.field\_of\_study.label\_tag }}<br>

      {{ form.field\_of\_study }}<br>

      {{ form.field\_of\_study.errors }}

    </p>

    <p>

      {{ form.tags.label\_tag }}<br>

      {{ form.tags }}<br>

      {{ form.tags.errors }}

</p>

Настройка админки ComputerScienceConceptAdmin:

@admin.register(ComputerScienceConcept)

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Поля для формы добавления/редактирования*

    fields = [

        'title', 'slug', 'description', 'difficulty',

        'field\_of\_study', 'tags', 'image', 'image\_preview'

    ]

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

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

*# Список записей*

    list\_display = (

        'id', 'title', 'field\_of\_study', 'time\_create',

        'is\_published', brief\_info, display\_difficulty,

    )

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

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

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

    list\_per\_page = 5

    search\_fields = ['title', 'field\_of\_study\_\_name']

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

list\_filter: Это свойство настроено на [PublishedFilter, 'field\_of\_study', DifficultyRangeFilter]. Оно определяет, какие фильтры будут доступны в боковой панели административного интерфейса для модели ComputerScienceConcept.

PublishedFilter позволяет фильтровать записи по статусу публикации.

'field\_of\_study' позволяет фильтровать записи по связанной области исследования.

DifficultyRangeFilter позволяет фильтровать записи по диапазону сложности.

Эти фильтры обеспечивают удобный способ сужения списка отображаемых концепций.

Горизонтальный фильтр для tags: В списке полей (fields) присутствует 'tags'. Поскольку tags является полем Many-to-Many в модели ComputerScienceConcept, Django Admin по умолчанию предоставляет удобный виджет для выбора нескольких тегов, который визуально представляет собой горизонтальный фильтр (filter\_horizontal), даже если filter\_horizontal = ['tags'] явно не указан в классе ModelAdmin. Это упрощает добавление и удаление тегов для каждой концепции.

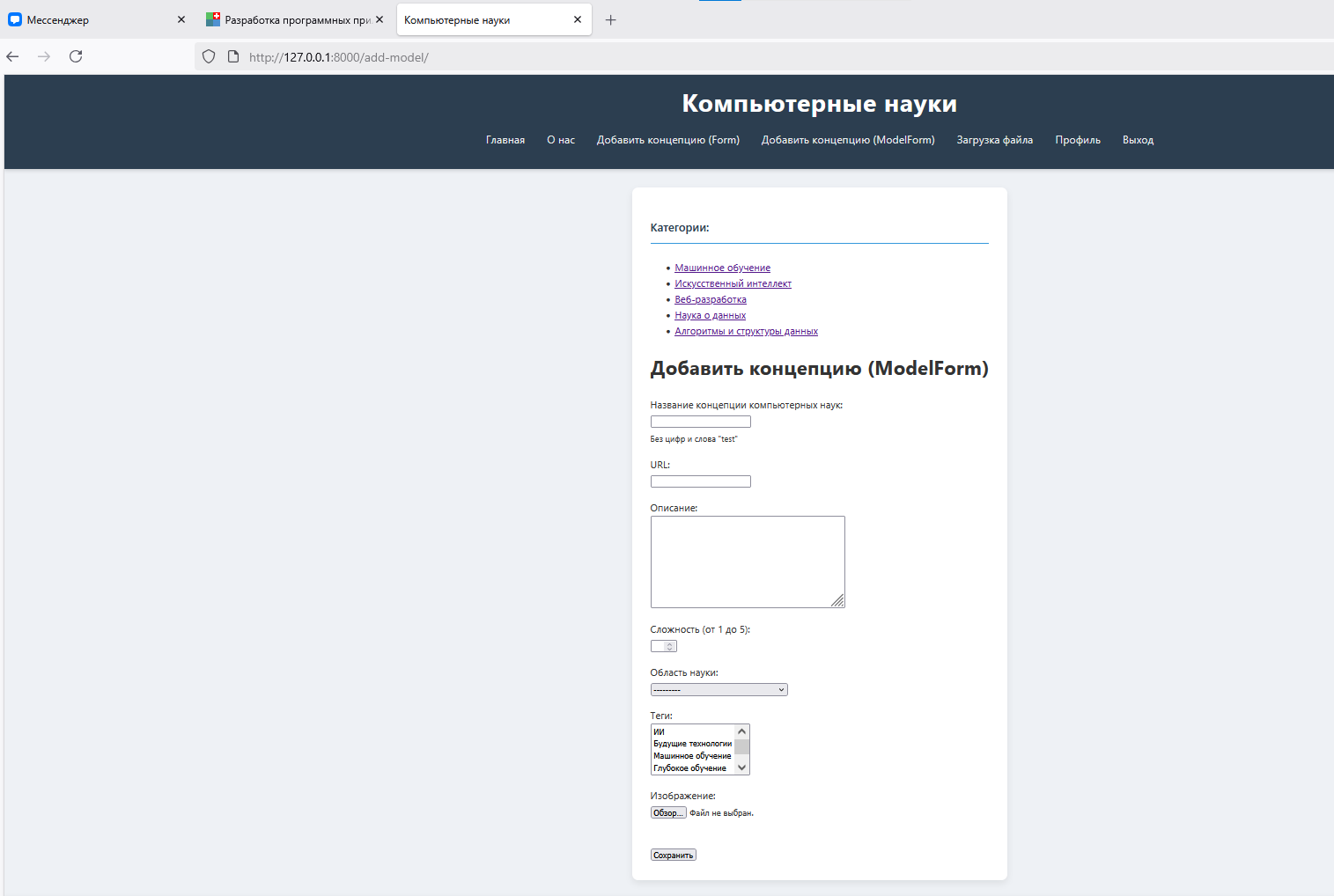


Рисунок 4 – Форма добавления концепции.

Профиль пользователя и управление паролем

Задача: реализовать профиль, смену и сброс пароля по e-mail.

ProfileView (users/views.py):

class ProfileView(LoginRequiredMixin, TemplateView):

    """

    Страница профиля текущего пользователя.

    """

    login\_url = reverse\_lazy('users:login')

    template\_name = 'users/profile.html'

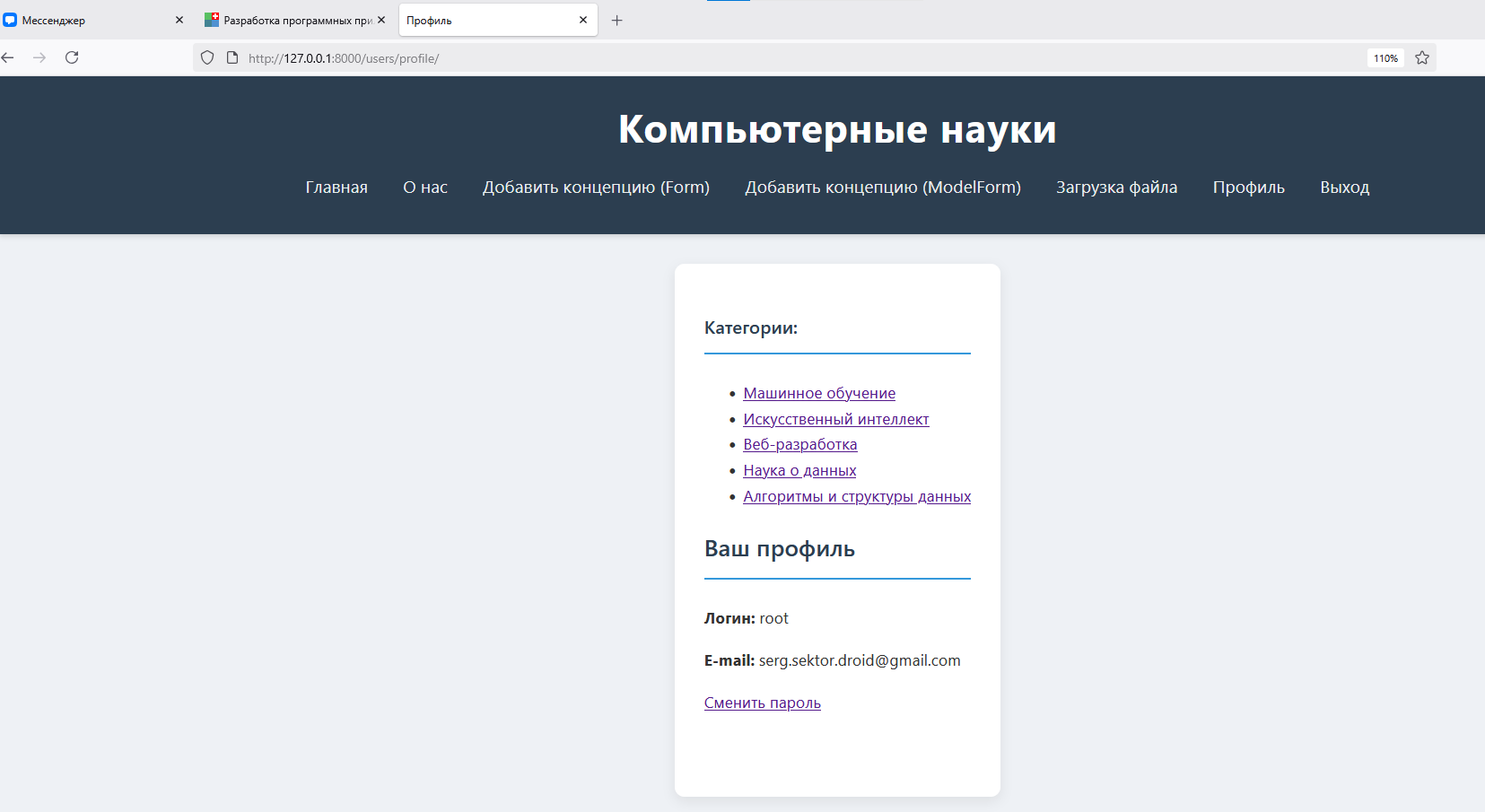
    def get\_context\_data(*self*, \*\**kwargs*):

        ctx = super().get\_context\_data(\*\**kwargs*)

        ctx['user'] = *self*.request.user

*return* ctx

В шаблоне profile.html выводятся {{ user.username }}, {{ user.email }} и форма смены пароля.

  
Рисунок 5 – Профиль пользователя.

Смена пароля:

class CustomPasswordChangeView(LoginRequiredMixin, PasswordChangeView):

    """

    Форма смены пароля.

    """

    login\_url = reverse\_lazy('users:login')

    template\_name = 'users/password\_change\_form.html'

    success\_url = reverse\_lazy('users:password\_change\_done')

Шаблон password\_change\_form.html содержит поля old\_password, new\_password1, new\_password2

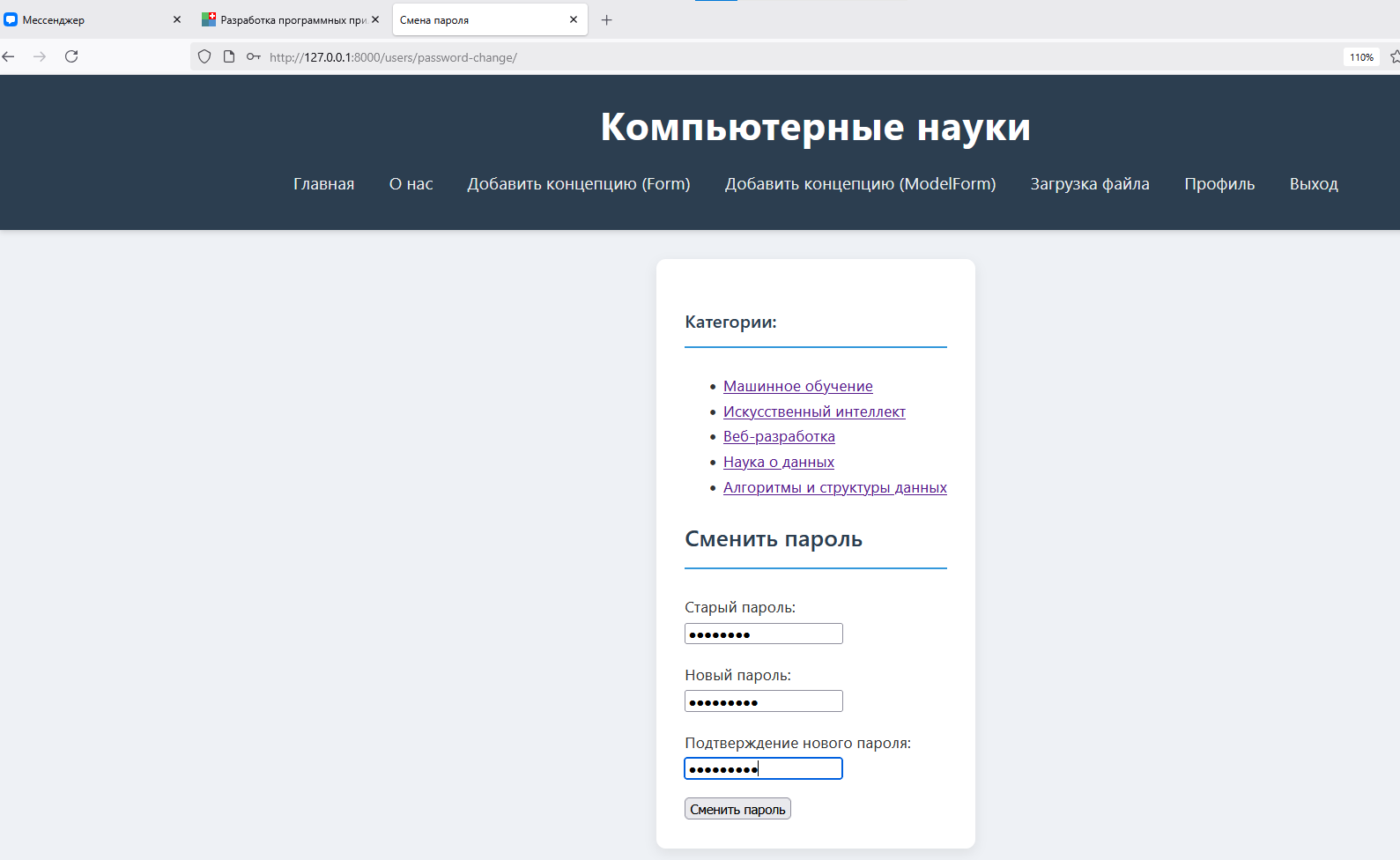


Рисунок 6 – Смена пароля.

Восстановление пароля (PasswordResetView и др.):

class CustomPasswordResetView(PasswordResetView):

    """

    Шаг 1: форма запроса сброса пароля (ввод e-mail, отправка письма).

    """

    template\_name = 'users/password\_reset\_form.html'

    email\_template\_name = 'users/password\_reset\_email.html'

    subject\_template\_name = 'users/password\_reset\_subject.txt'

    success\_url = reverse\_lazy('users:password\_reset\_done')

Настройки SMTP Яндекса в settings.py:

EMAIL\_BACKEND       = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST          = 'smtp.yandex.ru'

EMAIL\_PORT          = 587

EMAIL\_USE\_TLS       = True

EMAIL\_HOST\_USER     = 'sergeipython884@yandex.ru'

EMAIL\_HOST\_PASSWORD = 'pass'

DEFAULT\_FROM\_EMAIL  = 'Computer Science Concept <sergeipython884@yandex.ru>'

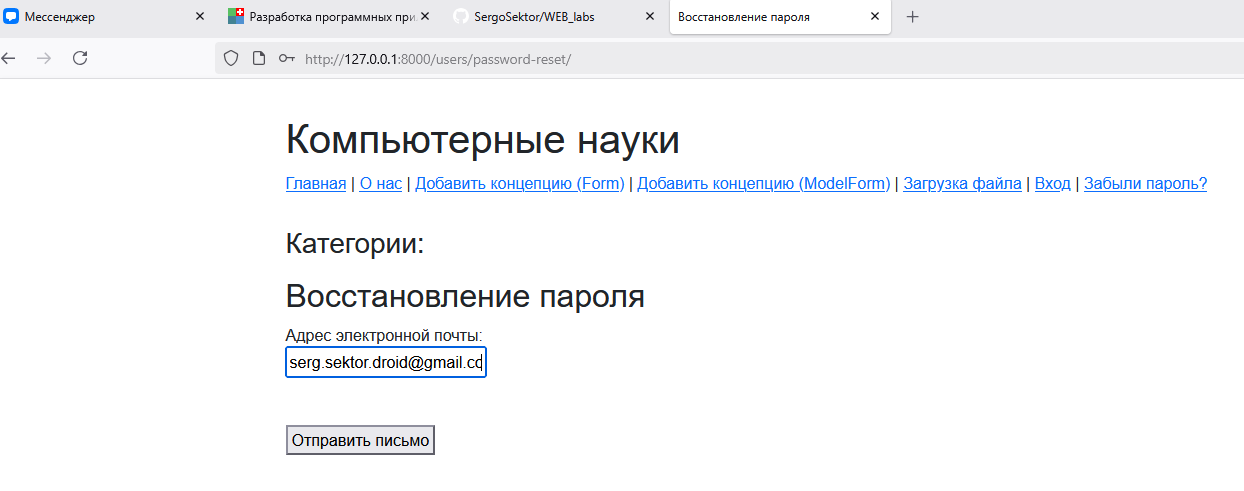


Рисунок 7 – Восстановление пароля.

После выбора адреса на настоящий почтовый ящик приходит письмо с ссылкой для восстановления пароля.

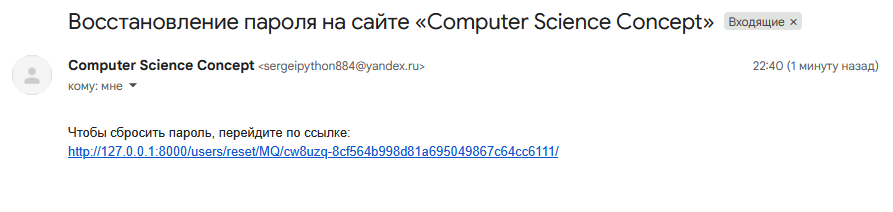


Рисунок 8 – Письмо на настоящий email.

При переходе по ссылке открывается форма восстановления пароля с вводом нового пароля и его подтверждением.

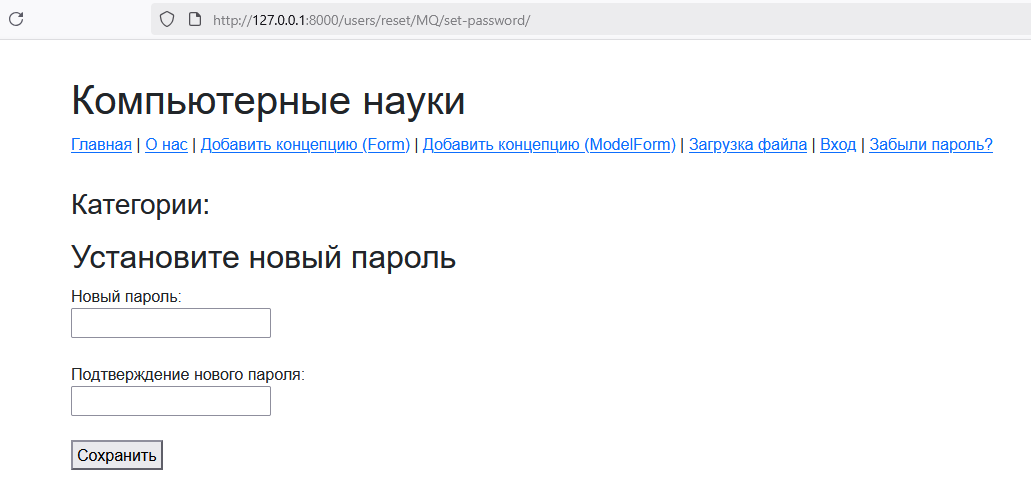


Рисунок 9 – Смена пароля.

После успешного ввода одинаковых паролей пользователю приходит уведомление о смене.

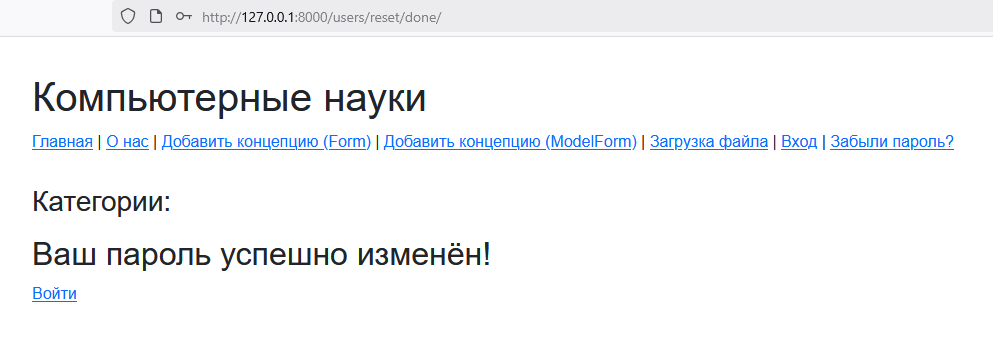


Рисунок 10 – Уведомление о смене пароля.

Админ-панель и разрешения

Задача: дать администраторам удобные инструменты и создать группы прав.

1. ComputerScienceConceptAdmin (cs/admin.py):

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Поля для формы добавления/редактирования*

    fields = [

        'title', 'slug', 'description', 'difficulty',

        'field\_of\_study', 'tags', 'image', 'image\_preview' *# Updated fields*

    ]

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

    readonly\_fields = ['time\_create', 'time\_update', 'image\_preview'] *# Added image\_preview*

*# Список записей*

    list\_display = (

        'id', 'title', 'field\_of\_study', 'time\_create',

        'is\_published', brief\_info, display\_difficulty, *# Updated display\_difficulty*

    )

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

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

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

    list\_per\_page = 5

    search\_fields = ['title', 'field\_of\_study\_\_name'] *# Updated search fields*

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

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

Пользовательские фильтры PublishedFilter, PriceRangeFilter позволяют быстро отфильтровать записи в админке.

Группы и права (через shell):

*from* django.contrib.auth.models *import* Group, Permission

*from* django.contrib.contenttypes.models *import* ContentType

*from* cs.models *import* ComputerScienceConcept *# Убедитесь, что этот импорт правильный*

*# Создаем или получаем группы*

editors, \_ = Group.objects.get\_or\_create(*name*='Editors')

viewers, \_ = Group.objects.get\_or\_create(*name*='Viewers')

*# Получаем ContentType для модели ComputerScienceConcept*

content\_type = ContentType.objects.get\_for\_model(ComputerScienceConcept)

*# Получаем необходимые разрешения*

change\_concept\_perm = Permission.objects.get(*content\_type*=content\_type, *codename*='change\_computerscienceconcept')

view\_concept\_perm = Permission.objects.get(*content\_type*=content\_type, *codename*='view\_computerscienceconcept')

*# Назначаем разрешения группе "Редакторы"*

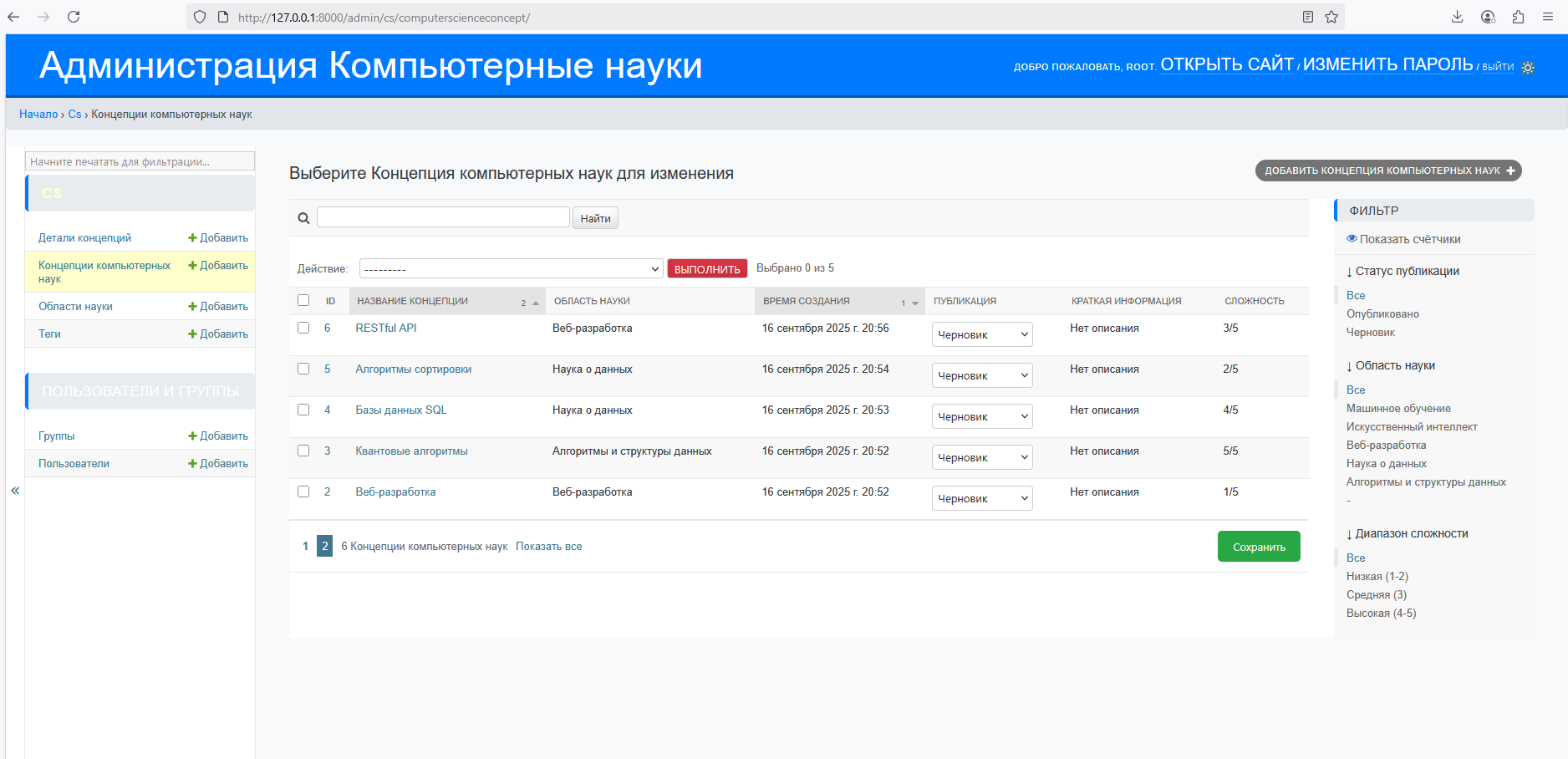
editors.permissions.add(change\_concept\_perm, view\_concept\_perm)

*# Назначаем разрешения группе "Просмотрщики"*

viewers.permissions.add(view\_concept\_perm)

Создали 2 группы и выдали права для каждой группы.

Во вкладке списка концепций в админ панели удобный интерфейс позволяющий изменять статус публикации, добавлять и удалять концепции. Также реализована возможность массовых действий, при выборе нескольких концепций можно массово опубликовывать, удалить или переместить в черновик.

Рисунок 11 – Список концепций в админ панели.

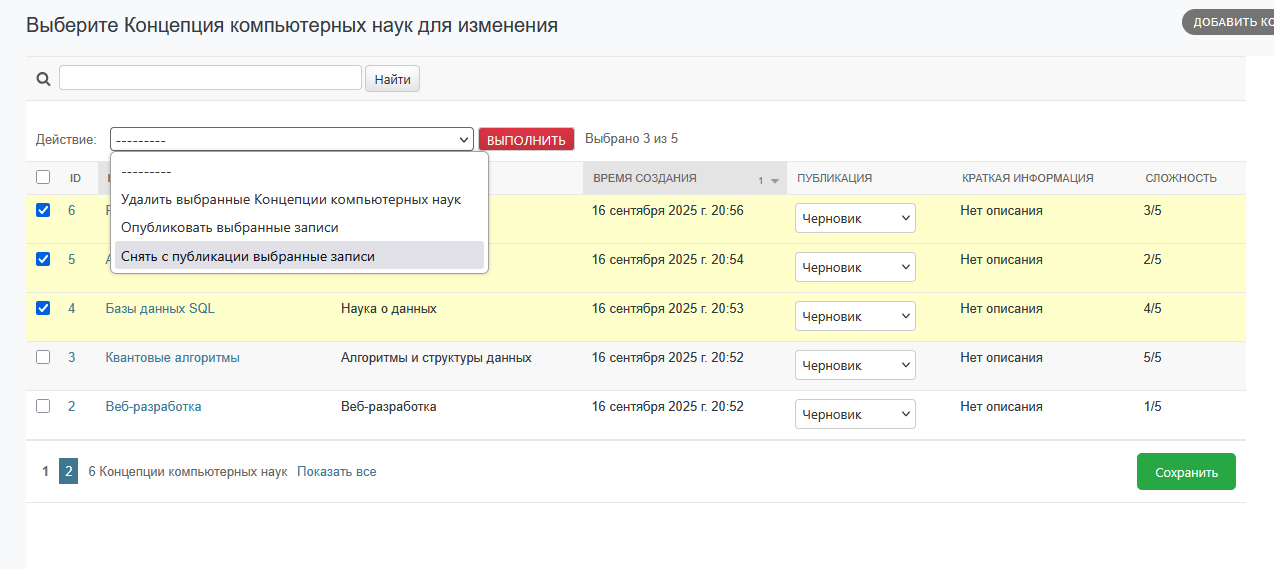


Рисунок 12 – Массовые действия в админ панели.

Во вкладе Области науки можно посмотреть, добавить, изменить список областей и их описания.

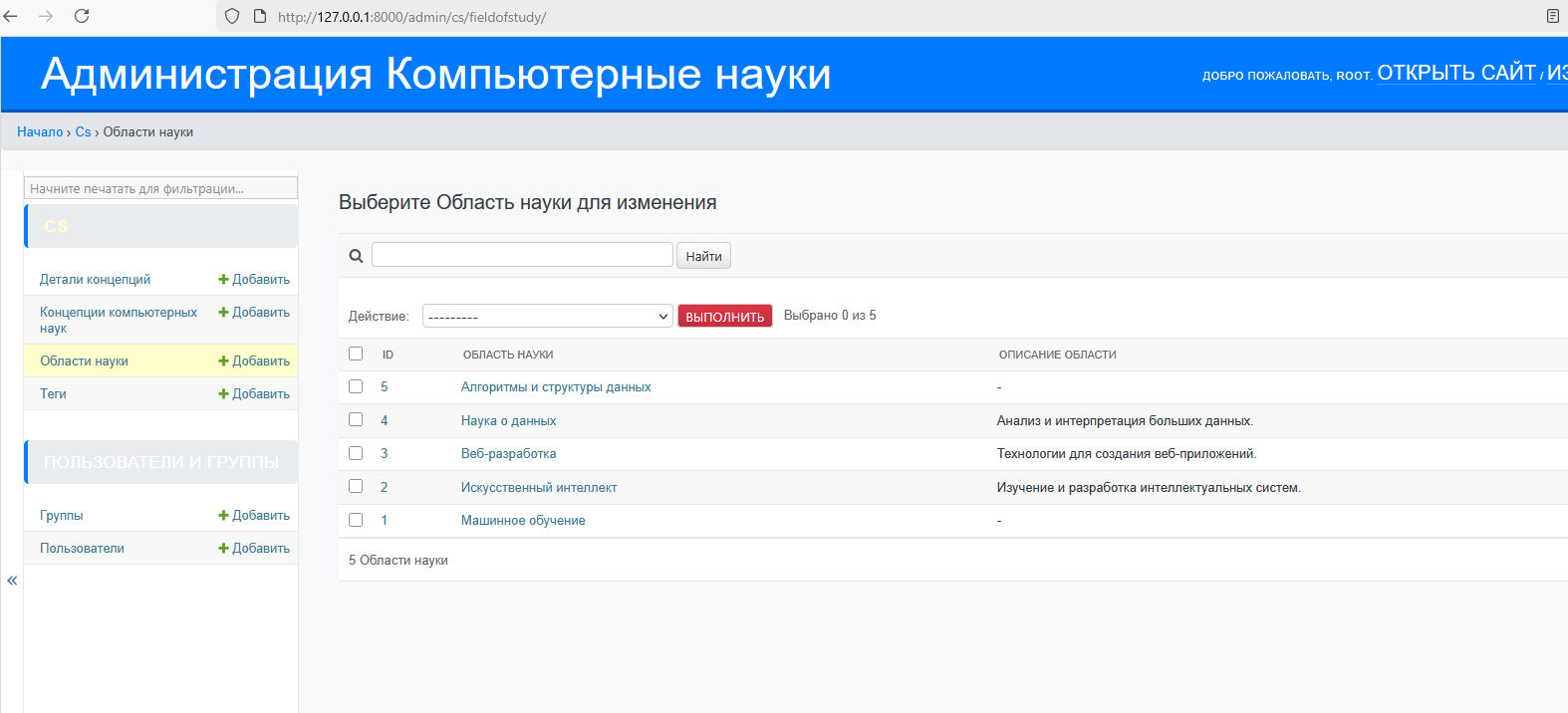


Рисунок 13 – Список компаний.

Во кладке tags можно посмотреть, добавить, изменить список категорий концепций.

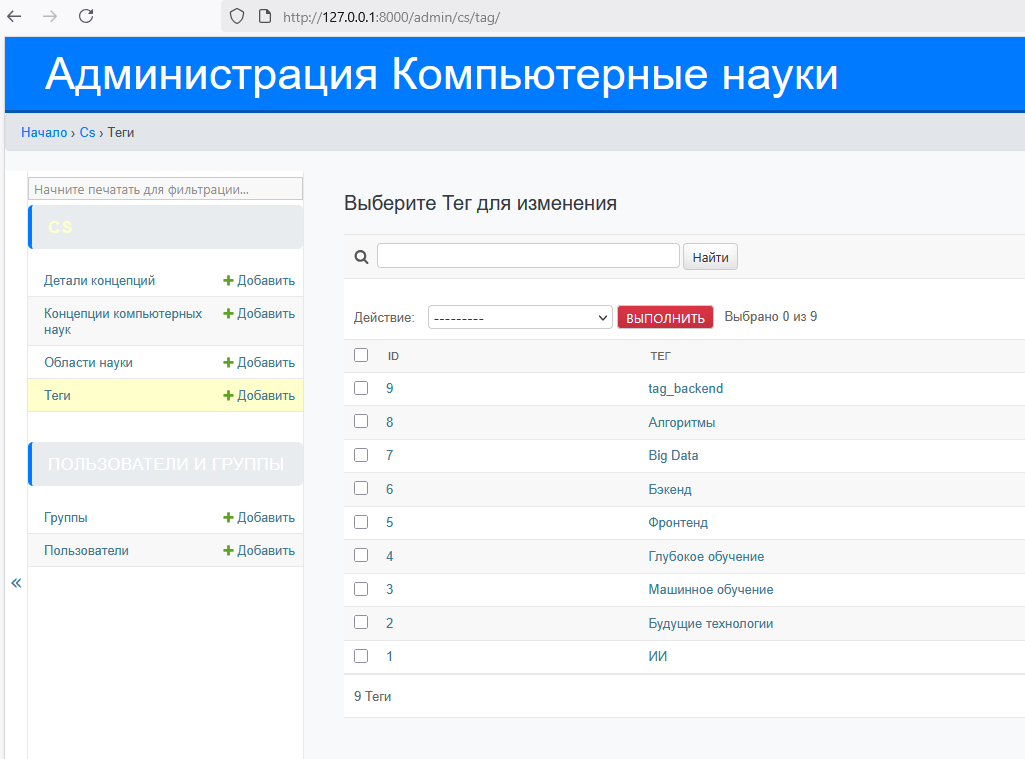


Рисунок 14 – Список категорий концепций.

Во вкладке детали концепций посмотреть, добавить, изменить названия концепций, ключевые технологии, предварительные условия, примерное время изучения.

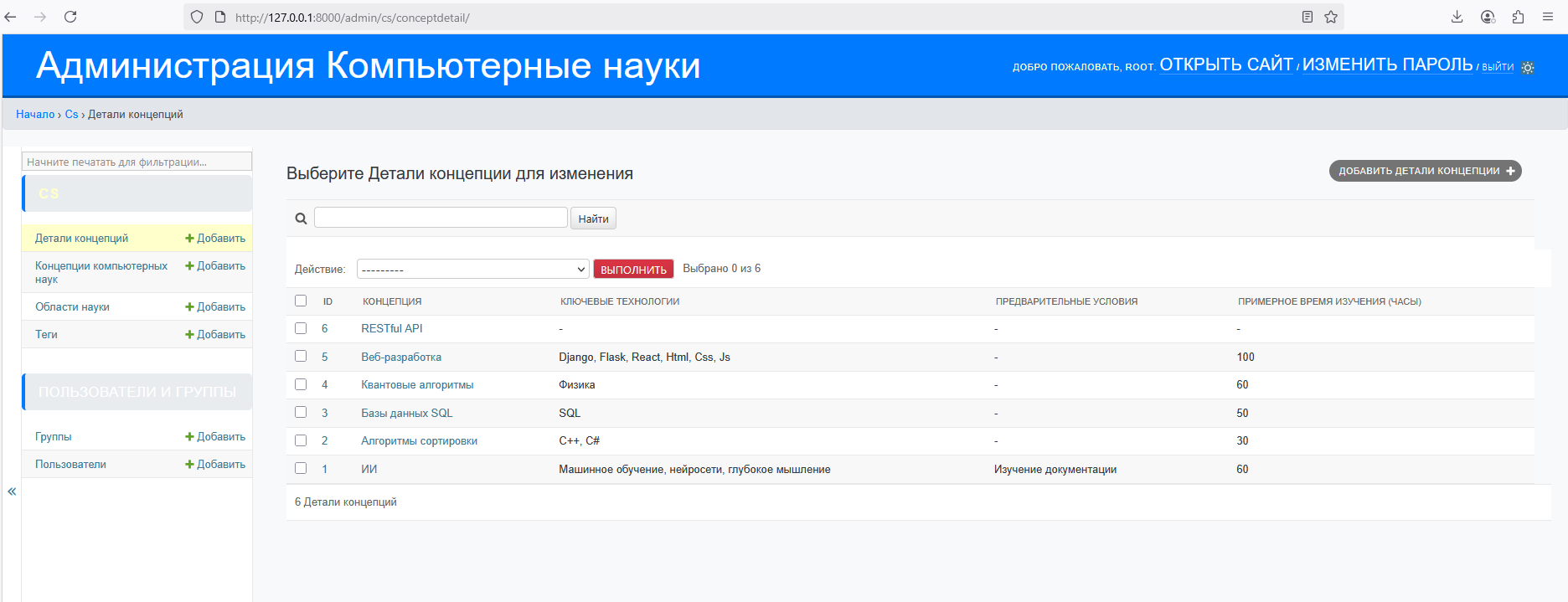


Рисунок 15 – Детали концепций.

Во вкладке группы можно посмотреть, добавить, изменить список групп пользователей.

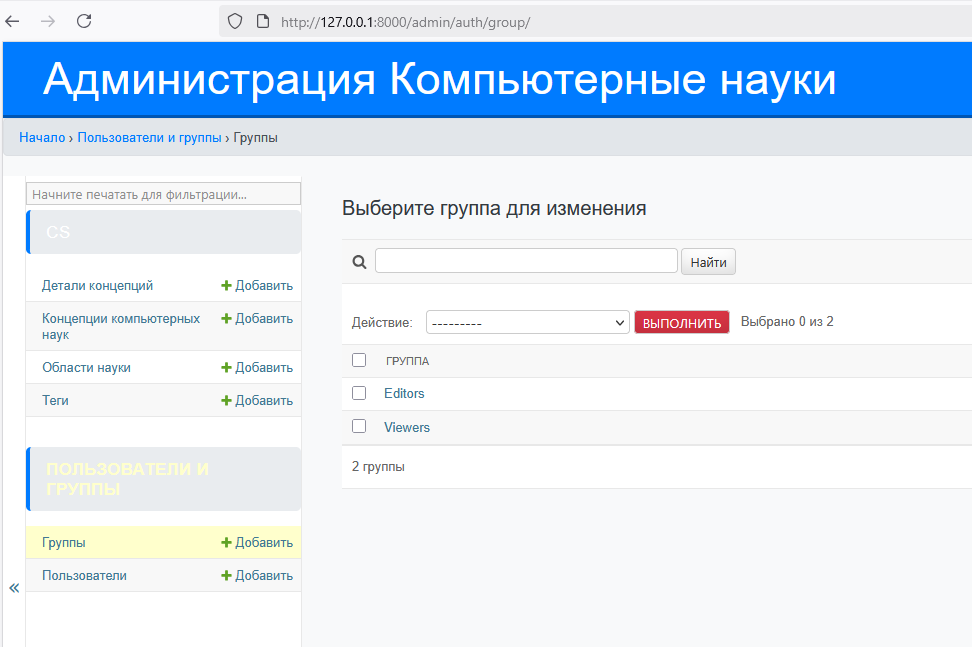


Рисунок 16 – Список групп.

Во вкладке пользователи можно посмотреть, добавить, изменить список пользователей.

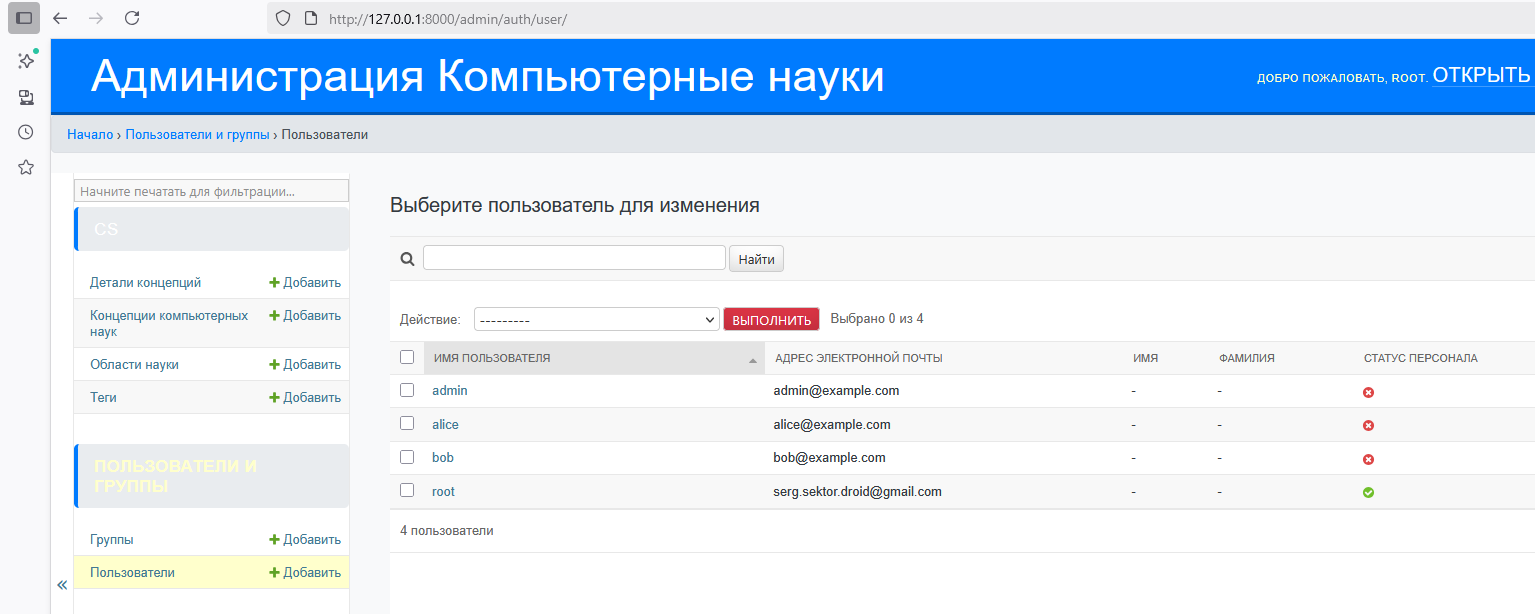


Рисунок 17 – Список пользователей.

Каждому пользователю можно настроить группу, права, фамилию, имя, посмотреть дату последнего входа и регистрации.

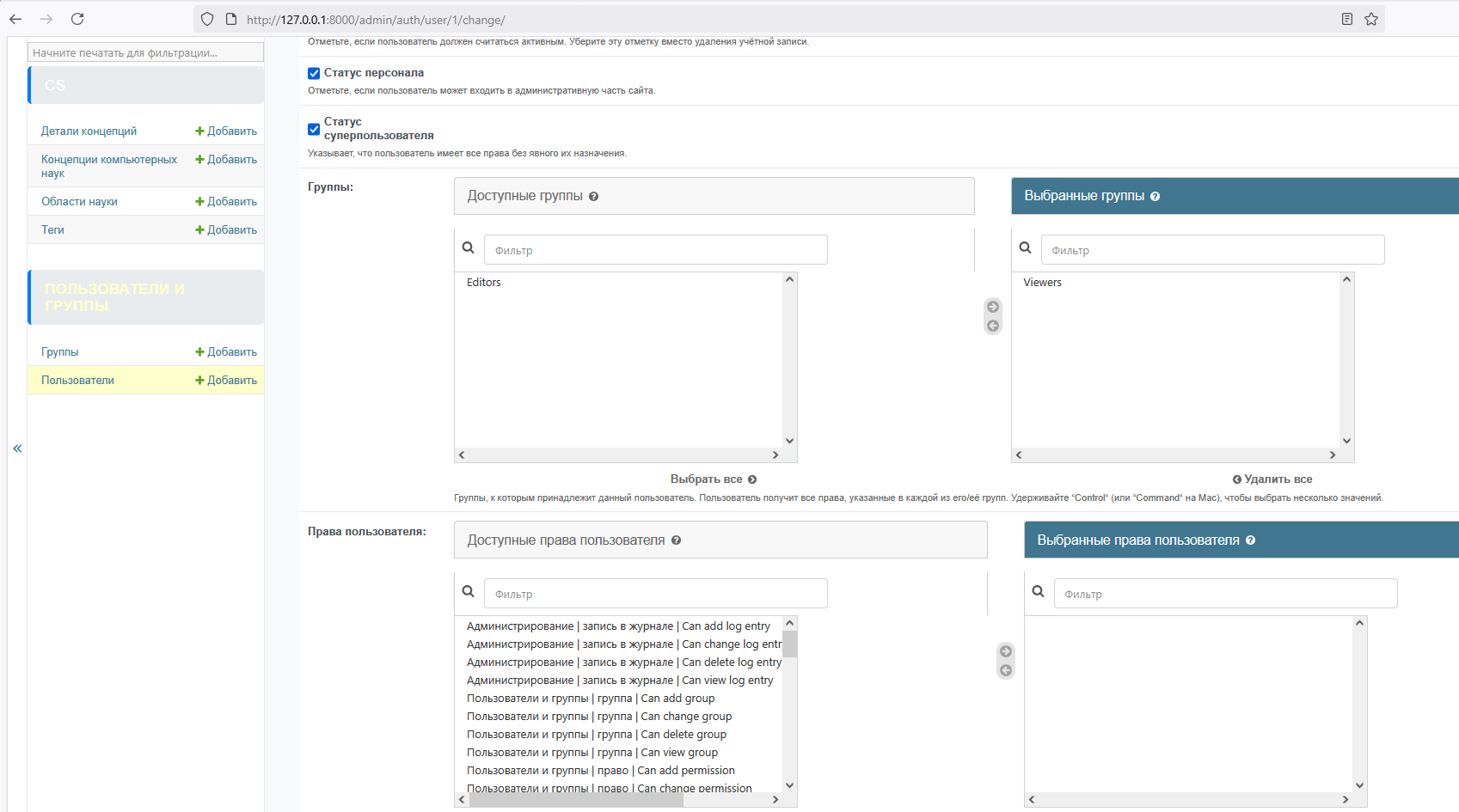


Рисунок 18 – Настройка пользователей.

# **Заключение**

В процессе выполнения расчетно графического задания было разработано и поэтапно расширено веб-приложение «Мир компьютерных наук» на Django. К началу работы проект представлял собой простой список статических концепций, а сегодня — это:

* Модели, отражающие авторство, статус публикации, отношения «концепции», «теги» и «комментарии», обеспечивающие гибкую структуру данных и масштабируемость.
* Формы и валидация, от каркасной FormView до ModelForm, гарантируют корректный ввод и удобный интерфейс создания/редактирования.
* Класс-based views с миксином общего контекста позволяют легко добавлять новые страницы, не дублируя код меню и пагинации.
* Интерактивность: система комментариев и «лайк в одно нажатие» без лишних перезагрузок делает работу пользователя более комфортной.
* Безопасность и права: авторизованные пользователи работают только со своей информацией, админ-панель расширена фильтрами, массовыми действиями и пользовательскими разрешениями.
* Управление аккаунтом: регистрация, профиль, смена и восстановление пароля через SMTP-сервер Яндекса обеспечивают полный цикл работы с учётными записями.

Таким образом, на каждом шаге мы не просто добавляли новую функциональность, а выстраивали архитектуру так, чтобы код оставался модульным, читаемым и легко расширяемым.

# **Cпискок литературы**

1. Официальная документация Django <https://docs.djangoproject.com/en/5.1/>
2. Думанский В.Н. Изучаем Python: введение в программирование на языке Python / В.Н. Думанский. — 2-е изд. — СПб. : Питер, 2018. — 608 с.
3. Greenfeld A.R., Greenfeld D.R. Two Scoops of Django 3.x: лучшие практики для Django / пер. с англ. — Leanpub, 2020. — 290 с.
4. Дакетт Дж. HTML и CSS: разработка и дизайн веб-сайтов / Джон Дакетт; пер. с англ. — СПб. : Питер, 2011. — 512 с. : ил.
5. Дакетт Дж. JavaScript и jQuery: интерактивная веб-разработка / Джон Дакетт; пер. с англ. — СПб. : Питер, 2015. — 624 с.

# **Приложение А. Листинг программы**

Cs/

Admin.css:

body {

    background-color: #f8f9fa;

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

    color: #343a40;

}

#header {

    background-color: #007bff !important; */\* Синий цвет \*/*

    border-bottom: 3px solid #0056b3; */\* Темно-синяя полоса \*/*

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

#header h1, #header a {

    color: #ffffff !important;

    font-size: 1.8em;

    font-weight: 500;

}

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

    background-color: #e9ecef; */\* Светло-серый фон \*/*

    color: #343a40; */\* Темный текст \*/*

    padding: 12px 15px;

    border-radius: 5px;

    margin-bottom: 15px;

    font-size: 1.2em;

    border-left: 4px solid #007bff; */\* Синяя полоса слева \*/*

}

div.breadcrumbs {

    background-color: #e2e6ea;

    padding: 10px 15px;

    border-radius: 5px;

    margin-bottom: 20px;

    color: #495057;

    font-size: 0.95em;

    border: 1px solid #dcdfe3;

}

div.breadcrumbs a {

    color: #007bff;

    text-decoration: none;

}

div.breadcrumbs a:hover {

    text-decoration: underline;

}

.actions button {

    background-color: #dc3545 !important; */\* Красный для удаления \*/*

    color: #ffffff !important;

    border: none;

    padding: 10px 18px;

    margin: 5px;

    border-radius: 5px;

    font-size: 0.95em;

    text-transform: uppercase;

    font-weight: 500;

    transition: background-color 0.3s ease;

}

.actions button:hover {

    background-color: #c82333 !important;

}

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

    background: #28a745 !important; */\* Зеленый для сохранения \*/*

    color: #ffffff !important;

    border-radius: 5px;

    padding: 10px 18px;

    font-weight: 500;

    transition: background-color 0.3s ease;

}

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

    background: #218838 !important;

}

.object-tools a {

    color: #007bff;

    text-decoration: none;

    font-weight: 500;

    margin-right: 15px;

    transition: color 0.3s ease;

}

.object-tools a:hover {

    color: #0056b3;

    text-decoration: underline;

}

*/\* Стили для форм \*/*

.form-row {

    padding: 10px 0;

    border-bottom: 1px solid #eee;

}

.form-row label {

    font-weight: 600;

    color: #495057;

}

.form-row input[type="text"], .form-row input[type="number"], .form-row textarea, .form-row select {

    border: 1px solid #ced4da;

    border-radius: 4px;

    padding: 8px 12px;

    width: 100%;

    max-width: 400px; */\* Ограничить ширину полей \*/*

    box-sizing: border-box;

}

.form-row textarea {

    min-height: 100px;

}

.field-box {

    margin-bottom: 15px;

}

.aligned label {

    width: 120px; */\* Выравнивание лейблов \*/*

    padding-right: 10px;

}

.aligned p.help, .aligned div.help {

    margin-left: 130px;

}

*/\* Media Queries для адаптивности \*/*

@media (max-width: 768px) {

    #header h1 {

        font-size: 1.5em;

    }

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

        font-size: 1em;

        padding: 8px 10px;

    }

    div.breadcrumbs {

        font-size: 0.9em;

        padding: 8px 10px;

    }

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

        padding: 8px 12px;

        font-size: 0.85em;

    }

    .object-tools a {

        margin-right: 10px;

        font-size: 0.9em;

    }

    .form-row input[type="text"], .form-row input[type="number"], .form-row textarea, .form-row select {

        max-width: 100%;

    }

    .aligned label {

        width: 100%;

        padding-right: 0;

        margin-bottom: 5px;

    }

    .aligned p.help, .aligned div.help {

        margin-left: 0;

    }

}

styles.css:

html, body {

    height: 100%;

    margin: 0;

}

body {

    font-family: 'Roboto', 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; */\* Изменен шрифт \*/*

    line-height: 1.6;

    color: #333;

    background-color: #eef1f5; */\* Более светлый фон \*/*

    display: flex;

    flex-direction: column;

}

.container {

    max-width: 1200px;

    margin: 30px auto; */\* Увеличен отступ сверху/снизу \*/*

    padding: 30px; */\* Увеличен внутренний отступ \*/*

    background-color: #fff;

    border-radius: 10px; */\* Более мягкие углы \*/*

    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.08); */\* Более выраженная тень \*/*

    flex-grow: 1;

}

header {

    background-color: #2c3e50; */\* Темно-синий цвет для шапки \*/*

    color: #fff;

    padding: 1.2rem 0; */\* Увеличен padding \*/*

    text-align: center;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.15);

}

header h1 {

    margin: 0;

    font-size: 2.5em; */\* Увеличен размер заголовка \*/*

    letter-spacing: 1px;

}

header nav ul {

    list-style-type: none;

    padding: 0;

    margin-top: 15px; */\* Отступ между заголовком и меню \*/*

    display: flex;

    justify-content: center;

    flex-wrap: wrap; */\* Для адаптивности \*/*

}

header nav ul li {

    margin: 0 18px; */\* Увеличен горизонтальный отступ \*/*

}

header nav ul li a {

    color: #ecf0f1; */\* Более светлый цвет ссылок \*/*

    text-decoration: none;

    font-weight: 500; */\* Среднее начертание \*/*

    font-size: 1.1em; */\* Увеличен размер шрифта \*/*

    transition: color 0.3s ease, transform 0.2s ease; */\* Добавлен transform \*/*

}

header nav ul li a:hover {

    color: #3498db; */\* Синий цвет при наведении \*/*

    transform: translateY(-2px); */\* Небольшое поднятие при наведении \*/*

}

h2, h3 {

    color: #2c3e50; */\* Темно-синий для заголовков \*/*

    border-bottom: 2px solid #3498db; */\* Более выраженная синяя полоса \*/*

    padding-bottom: 12px;

    margin-bottom: 25px;

    font-weight: 600;

}

p {

    margin-bottom: 15px;

    line-height: 1.7;

}

img {

    max-width: 100%;

    height: auto;

    border-radius: 6px; */\* Более мягкие углы \*/*

    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);

}

.concept-list-item {

    background-color: #ffffff;

    border: 1px solid #e0e0e0;

    padding: 20px;

    margin-bottom: 20px;

    border-radius: 8px;

    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);

    transition: transform 0.2s ease, box-shadow 0.2s ease;

}

.concept-list-item:hover {

    transform: translateY(-3px);

    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);

}

.concept-list-item h3 a {

    color: #3498db;

    text-decoration: none;

    font-size: 1.6em;

}

.concept-list-item h3 a:hover {

    text-decoration: underline;

    color: #2980b9;

}

.comment-section {

    margin-top: 40px;

    border-top: 2px solid #3498db; */\* Синяя полоса \*/*

    padding-top: 30px;

}

.comment-form textarea {

    width: 100%;

    padding: 12px;

    border: 1px solid #c0c0c0;

    border-radius: 6px;

    resize: vertical;

    font-size: 1em;

    margin-bottom: 10px;

}

.comment-form button {

    background-color: #28a745;

    color: #fff;

    padding: 12px 20px;

    border: none;

    border-radius: 6px;

    cursor: pointer;

    font-size: 1.1em;

    transition: background-color 0.3s ease, transform 0.2s ease;

}

.comment-form button:hover {

    background-color: #218838;

    transform: translateY(-1px);

}

.comment-item {

    background-color: #ecf0f1; */\* Светло-серый фон для комментариев \*/*

    border: 1px solid #d0d0d0;

    padding: 18px;

    margin-bottom: 15px;

    border-radius: 6px;

    box-shadow: 0 1px 4px rgba(0, 0, 0, 0.05);

}

.comment-item strong {

    color: #2c3e50;

    font-size: 1.1em;

}

.comment-item p {

    margin-top: 8px;

    margin-bottom: 0;

    font-size: 0.95em;

}

footer {

    margin-top: auto; */\* Важно для sticky footer \*/*

    text-align: center;

    color: #7f8c8d; */\* Темно-серый цвет текста \*/*

    padding: 25px;

    background-color: #34495e; */\* Темно-синий для футера \*/*

    border-top: 1px solid #2c3e50;

    border-radius: 0;

    color: #ecf0f1;

    box-shadow: 0 -2px 5px rgba(0, 0, 0, 0.15);

}

*/\* Адаптивность \*/*

@media (max-width: 768px) {

    header h1 {

        font-size: 2em;

    }

    header nav ul {

        flex-direction: column;

        align-items: center;

    }

    header nav ul li {

        margin: 5px 0;

    }

    .container {

        margin: 15px auto;

        padding: 15px;

    }

}

*/\* Стили для пагинации \*/*

.pagination-wrapper {

    margin-top: 30px;

    text-align: center;

}

.pagination {

    display: inline-flex;

    list-style: none;

    padding: 0;

    border-radius: 0.25rem;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

}

.page-item {

    margin: 0;

}

.page-link {

    display: block;

    padding: 10px 15px;

    color: #007bff;

    text-decoration: none;

    border: 1px solid #dee2e6;

    margin-left: -1px;

    transition: all 0.3s ease;

}

.page-link:hover {

    background-color: #e9ecef;

    color: #0056b3;

    border-color: #dee2e6;

}

.page-item:first-child .page-link {

    border-top-left-radius: 0.25rem;

    border-bottom-left-radius: 0.25rem;

}

.page-item:last-child .page-link {

    border-top-right-radius: 0.25rem;

    border-bottom-right-radius: 0.25rem;

}

.page-item.active .page-link {

    z-index: 1;

    color: #fff;

    background-color: #007bff;

    border-color: #007bff;

}

.page-link.prev-next {

    font-weight: bold;

    padding: 10px 20px;

}

.page-link.prev-next span {

    margin: 0 5px;

}

Admin/Base.html

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

{% load static %}

{% block extrastyle %}

    {{ block.super }}

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

{% endblock %}

{% block extrahead %}

    {{ block.super }}

    <meta name="viewport" content="user-scalable=no, width=device-width, initial-scale=1.0, maximum-scale=1.0">

    <link rel="stylesheet" type="text/css" href="{% static "admin/css/responsive.css" %}" />

{% endblock %}

base\_site.html:

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

{% load static %}

{% block title %}{{ title }} | Администрация Компьютерные науки{% endblock %}

{% block branding %}

<h1 id="site-name"><a href="{% url 'admin:index' %}">Администрация Компьютерные науки</a></h1>

{% endblock %}

{% block nav-global %}{% endblock %}

{% block extrastyle %}

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

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

{% endblock %}

About.html:

{% extends 'cs/base.html' %}

{% block title %}О нас - Компьютерные науки{% endblock %}

{% block heading %}О нас{% endblock %}

{% block content %}

  <p>Мы — команда энтузиастов, собравшая информацию о различных областях компьютерных наук для создания образовательных материалов и обзоров.</p>

{% endblock %}

Add\_concept\_custom.html:

{% extends 'cs/base.html' %}

{% block content %}

  <h1>Добавить концепцию (каркасная форма)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      {{ form.title.help\_text }}<br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.help\_text }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.help\_text }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Add\_concept\_model.html:

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

{% block content %}

  <h1>Добавить концепцию (ModelForm)</h1>

  <form method="post" enctype="multipart/form-data">{% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.title.label\_tag }}<br>

      {{ form.title }}<br>

      <small>{{ form.title.help\_text }}</small><br>

      {{ form.title.errors }}

    </p>

    <p>

      {{ form.slug.label\_tag }}<br>

      {{ form.slug }}<br>

      {{ form.slug.errors }}

    </p>

    <p>

      {{ form.description.label\_tag }}<br>

      {{ form.description }}<br>

      {{ form.description.errors }}

    </p>

    <p>

      {{ form.difficulty.label\_tag }}<br>

      {{ form.difficulty }}<br>

      {{ form.difficulty.errors }}

    </p>

    <p>

      {{ form.field\_of\_study.label\_tag }}<br>

      {{ form.field\_of\_study }}<br>

      {{ form.field\_of\_study.errors }}

    </p>

    <p>

      {{ form.tags.label\_tag }}<br>

      {{ form.tags }}<br>

      {{ form.tags.errors }}

    </p>

    <p>

      {{ form.image.label\_tag }}<br>

      {{ form.image }}<br>

      {{ form.image.help\_text }}<br>

      {{ form.image.errors }}

    </p>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Base.html:

{% load static %}

{% load cs\_tags %}

<!DOCTYPE html>

<html lang="ru">

<head>

  <meta charset="UTF-8">

  <title>{% block title %}Компьютерные науки{% endblock %}</title>

*<!-- Подключение собственного CSS -->*

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

</head>

<body>

  <header>

    <h1>{% block heading %}Компьютерные науки{% endblock %}</h1>

    <nav>

      <ul>

        <li><a href="{% url 'cs:home' %}">Главная</a></li>

        <li><a href="{% url 'cs:about' %}">О нас</a></li>

        <li><a href="{% url 'cs:add\_concept\_custom' %}">Добавить концепцию (Form)</a></li>

        <li><a href="{% url 'cs:add\_concept\_model' %}">Добавить концепцию (ModelForm)</a></li>

        <li><a href="{% url 'cs:upload\_file' %}">Загрузка файла</a></li>

        {% if user.is\_authenticated %}

          <li><a href="{% url 'users:profile' %}">Профиль</a></li>

          <li><a href="{% url 'users:logout' %}">Выход</a></li>

        {% else %}

          <li><a href="{% url 'users:login' %}">Вход</a></li>

          <li><a href="{% url 'users:password\_reset' %}">Забыли пароль?</a></li>

        {% endif %}

      </ul>

    </nav>

  </header>

  <div class="container">

    <div class="categories my-3">

      <h3>Категории:</h3>

      {% get\_categories as cats %}

      <ul>

        {% for cat in cats %}

          <li><a href="{% url 'cs:field\_of\_study\_detail' field\_of\_study\_slug=cat.slug %}">{{ cat.name }}</a></li>

        {% endfor %}

      </ul>

    </div>

    <main>

      {% block content %}{% endblock %}

    </main>

  </div>

  <footer>

    <p>&copy; 2025 Компьютерные науки. Все права защищены.</p>

  </footer>

</body>

</html>

concept\_detail.html:

{% extends 'cs/base.html' %}

{% block title %}{{ concept.title }}{% endblock %}

{% block content %}

  <h2>{{ concept.title }}</h2>

*{# Если для этой концепции загружено изображение, покажем его #}*

  {% if concept.image %}

    <div style="margin-bottom: 1em;">

      <img src="{{ concept.image.url }}" alt="{{ concept.title }}" style="max-width:100%; height:auto;">

    </div>

  {% endif %}

  <p>Сложность: {{ concept.difficulty }}</p>

  {% if concept.field\_of\_study %}

    <p>Область науки: {{ concept.field\_of\_study.name }}</p>

  {% endif %}

  <div>

    {{ concept.description|linebreaks }}

  </div>

  {% if concept.tags.all %}

    <p>Теги:

      {% for tag in concept.tags.all %}

        <a href="{{ tag.get\_absolute\_url }}">{{ tag.name }}</a>{% if not forloop.last %}, {% endif %}

      {% endfor %}

    </p>

  {% endif %}

  <h3>Комментарии:</h3>

  {% if user.is\_authenticated %}

    <form action="" method="post">

      {% csrf\_token %}

      {{ comment\_form.as\_p }}

      <button type="submit">Отправить комментарий</button>

    </form>

  {% else %}

    <p>Чтобы оставить комментарий, пожалуйста, <a href="{% url 'users:login' %}">войдите</a>.</p>

  {% endif %}

  {% if comments %}

    {% for comment in comments %}

      <div style="border: 1px solid #ccc; padding: 10px; margin-bottom: 10px;">

        <p><strong>{{ comment.author.username }}</strong> ({{ comment.created|date:"d.m.Y H:i" }}):</p>

        <p>{{ comment.text|linebreaksbr }}</p>

      </div>

    {% endfor %}

  {% else %}

    <p>Пока нет комментариев.</p>

  {% endif %}

  {% if concept.detail %}

    <h3>Дополнительная информация:</h3>

    <p>Ключевые технологии: {{ concept.detail.core\_technologies }}</p>

    <p>Предварительные условия: {{ concept.detail.prerequisites }}</p>

    <p>Примерное время изучения: {{ concept.detail.estimated\_learning\_time }} часов</p>

  {% endif %}

{% endblock %}

cs\_list.html:

{% extends 'cs/base.html' %}

{% block title %}Темы - Компьютерные науки{% endblock %}

{% block heading %}Список тем{% endblock %}

{% block content %}

  <p>Ниже представлен список тем в области компьютерных наук.</p>

  {% if cs\_topics %}

    <ul>

      {% for topic in cs\_topics %}

        <li>

          <strong>{{ topic.name }}</strong> – {{ topic.description|truncatewords:20 }}

          <a href="{% url 'cs\_detail' topic.id %}">Подробнее</a>

        </li>

      {% endfor %}

    </ul>

  {% else %}

    <p>Темы не найдены.</p>

  {% endif %}

{% endblock %}

Index.html:  
{% extends 'cs/base.html' %}

{% block title %}{{ title }}{% endblock %}

{% block heading %}Добро пожаловать в мир компьютерных наук!{% endblock %}

{% block content %}

  <h2>Наши концепции компьютерных наук</h2>

  <ul class="list-articles">

    {% for concept in concepts %}

      <li>

        <h3>{{ concept.title }}</h3>

        <p>{{ concept.description|linebreaks|truncatewords:30 }}</p>

        <p>Сложность: {{ concept.difficulty }}</p>

        <a href="{{ concept.get\_absolute\_url }}">Подробнее</a>

      </li>

    {% endfor %}

  </ul>

  {% if page\_obj.has\_other\_pages %}

    <div class="pagination-wrapper">

        <nav aria-label="Постраничная навигация">

            <ul class="pagination">

                {% if page\_obj.has\_previous %}

                    <li class="page-item">

                        <a class="page-link prev-next" href="?page={{ page\_obj.previous\_page\_number }}" aria-label="Предыдущая">

                            <span aria-hidden="true">&laquo;</span> Предыдущая

                        </a>

                    </li>

                {% endif %}

                {% for num in page\_range %}

                    {% if num == page\_obj.number %}

                        <li class="page-item active" aria-current="page">

                            <span class="page-link">{{ num }}</span>

                        </li>

                    {% else %}

                        <li class="page-item">

                            <a class="page-link" href="?page={{ num }}">{{ num }}</a>

                        </li>

                    {% endif %}

                {% endfor %}

                {% if page\_obj.has\_next %}

                    <li class="page-item">

                        <a class="page-link prev-next" href="?page={{ page\_obj.next\_page\_number }}" aria-label="Следующая">

                            Следующая <span aria-hidden="true">&raquo;</span>

                        </a>

                    </li>

                {% endif %}

            </ul>

        </nav>

    </div>

    {% endif %}

{% endblock %}

Tags\_list.html:

{% extends 'cs/base.html' %}

{% block title %}{{ title }}{% endblock %}

{% block content %}

  <h2>Список тегов</h2>

  <ul>

    {% for tag in tags %}

      <li><a href="{{ tag.get\_absolute\_url }}">{{ tag.name }}</a></li>

    {% endfor %}

  </ul>

{% endblock %}

Upload.html:

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

{% block content %}

  <h1>Загрузка файла</h1>

  <form method="post" enctype="multipart/form-data">

    {% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.file.label\_tag }}<br>

      {{ form.file }}<br>

      <small>{{ form.file.help\_text }}</small><br>

      {{ form.file.errors }}

    </p>

    <button type="submit">Загрузить</button>

  </form>

  {% if link %}

    <p>Файл успешно загружен: <a href="{{ link }}" target="\_blank">{{ link }}</a></p>

  {% endif %}

{% endblock %}

Admin.py:

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

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

*from* django.utils.html *import* mark\_safe *# New import*

*from* django.db.models *import* ExpressionWrapper, F, DecimalField

*from* .models *import* ComputerScienceConcept, FieldOfStudy, ConceptDetail, Tag *# Updated model imports*

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

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

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

*# Пользовательское вычисляемое поле: краткая информация (п.7)*

@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: цена с налогом (п.7)*

@admin.display(*description*="Сложность") *# Changed description*

def display\_difficulty(*obj*):

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

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

*return* "N/A"

*# Кастомный фильтр для статуса публикации (п.9)*

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 (п.9)*

class DifficultyRangeFilter(SimpleListFilter):

    title = "Диапазон сложности"

    parameter\_name = "difficulty\_range"

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

*return* [

            ('low', 'Низкая (1-2)'),

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

            ('high', 'Высокая (4-5)'),

        ]

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

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

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

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

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

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

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

*return* *queryset*

@admin.register(ComputerScienceConcept) *# Updated model*

class ComputerScienceConceptAdmin(admin.ModelAdmin):

*# Поля для формы добавления/редактирования*

    fields = [

        'title', 'slug', 'description', 'difficulty',

        'field\_of\_study', 'tags', 'image', 'image\_preview' *# Updated fields*

    ]

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

    readonly\_fields = ['time\_create', 'time\_update', 'image\_preview'] *# Added image\_preview*

*# Список записей*

    list\_display = (

        'id', 'title', 'field\_of\_study', 'time\_create',

        'is\_published', brief\_info, display\_difficulty, *# Updated display\_difficulty*

    )

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

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

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

    list\_per\_page = 5

    search\_fields = ['title', 'field\_of\_study\_\_name'] *# Updated search fields*

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

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

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

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

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

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

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

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

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

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

    @admin.display(*description*='Превью изображения') *# New method*

    def image\_preview(*self*, *obj*):

*if* *obj*.image:

*return* mark\_safe(f"<img src='{*obj*.image.url}' style='max-height:200px;' />")

*return* "(нет изображения)"

@admin.register(FieldOfStudy) *# Updated model*

class FieldOfStudyAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'description') *# Updated fields*

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

    search\_fields = ('name', 'description') *# Updated search fields*

@admin.register(ConceptDetail) *# Updated model*

class ConceptDetailAdmin(admin.ModelAdmin):

    list\_display = ('id', 'concept', 'core\_technologies', 'prerequisites', 'estimated\_learning\_time') *# Updated fields*

    list\_display\_links = ('id', 'concept') *# Updated link*

    search\_fields = ('concept\_\_title', 'core\_technologies', 'prerequisites') *# Updated search fields*

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name')

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

    search\_fields = ('name',)

apps.py:

*from* django.apps *import* AppConfig

class CsConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'cs'

forms.py:

*from* django *import* forms

*from* django.conf *import* settings

*from* django.core.files.storage *import* default\_storage

*import* os

*import* uuid

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

*# 1. Собственный валидатор: запрет цифр в названии*

def validate\_title\_no\_digits(*value*):

*if* any(ch.isdigit() *for* ch *in* *value*):

*raise* forms.ValidationError(

            'Название не должно содержать цифр',

*code*='no\_digits'

        )

*# 2. Собственный валидатор: запрет слова "test"*

def validate\_no\_test(*value*):

*if* 'test' in *value*.lower():

*raise* forms.ValidationError(

            'Слово "test" запрещено в названии',

*code*='no\_test'

        )

*# Несвязанная с моделью форма*

class ConceptForm(forms.Form):

    title = forms.CharField(

*max\_length*=255,

*label*='Название концепции компьютерных наук',

*validators*=[validate\_title\_no\_digits],

*help\_text*='До 255 символов, без цифр'

    )

    description = forms.CharField(

        widget=forms.Textarea,

*required*=False,

*label*='Описание',

*help\_text*='Дополнительная информация (необязательно)'

    )

    difficulty = forms.IntegerField(

*label*='Сложность (от 1 до 5)',

*min\_value*=1,

*max\_value*=5,

*help\_text*='Число от 1 до 5'

    )

    image = forms.ImageField(

*required*=False,

*label*='Изображение',

*help\_text*='JPG/PNG до 5 МБ'

    )

*# Форма, связанная с моделью*

class ConceptModelForm(forms.ModelForm):

*# Переопределяем title, чтобы применить оба валидатора*

    title = forms.CharField(

*max\_length*=255,

*label*='Название концепции компьютерных наук',

*validators*=[validate\_no\_test, validate\_title\_no\_digits],

*help\_text*='Без цифр и слова "test"'

    )

    difficulty = forms.IntegerField(

*label*='Сложность (от 1 до 5)',

*min\_value*=1,

*max\_value*=5,

*help\_text*='Число от 1 до 5'

    )

    class Meta:

        model = ComputerScienceConcept

        fields = [

            'title',

            'slug',

            'description',

            'difficulty',

            'field\_of\_study',

            'tags',

            'image',

        ]

        widgets = {

            'description': forms.Textarea,

        }

class UploadForm(forms.Form):

    file = forms.FileField(

*label*='Выберите файл',

*help\_text*='Максимум 10 МБ. Любой тип.'

    )

    def clean\_file(*self*):

        f = *self*.cleaned\_data['file']

*if* f.size > 10 \* 1024 \* 1024:

*raise* forms.ValidationError('Файл слишком большой (больше 10 МБ)')

*return* f

    def save\_file(*self*):

        f = *self*.cleaned\_data['file']

        ext = os.path.splitext(f.name)[1]

        new\_name = f"{uuid.uuid4().hex}{ext}"

        path = os.path.join('uploads', new\_name)

        saved\_path = default\_storage.save(path, f)

*return* settings.MEDIA\_URL + saved\_path

class CommentForm(forms.ModelForm):

    class Meta:

        model = Comment

        fields = ['text']

        widgets = {

            'text': forms.Textarea(*attrs*={'rows': 4, 'placeholder': 'Ваш комментарий'}),

        }

        labels = {

            'text': 'Комментарий',

        }

models.py:

*from* django.db *import* models

*from* django.urls *import* reverse

*from* django.utils.text *import* slugify *# Импортируем slugify*

*from* .utils *import* resize\_image *# Импортируем функцию изменения размера изображения*

*# Модель для областей компьютерных наук*

class FieldOfStudy(models.Model):

    name = models.CharField(*max\_length*=255, *verbose\_name*="Область науки")

    slug = models.SlugField(*max\_length*=255, *unique*=True, *db\_index*=True, *verbose\_name*="URL")

    description = models.TextField(*blank*=True, *verbose\_name*="Описание области")

    class Meta:

        verbose\_name = "Область науки"

        verbose\_name\_plural = "Области науки"

    def \_\_str\_\_(*self*):

*return* *self*.name

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.name)

*# Проверяем уникальность слага*

            base\_slug = *self*.slug

            num = 1

*while* FieldOfStudy.objects.filter(*slug*=*self*.slug).exists():

*self*.slug = f"{base\_slug}-{num}"

                num += 1

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('cs:field\_of\_study\_detail', *kwargs*={'field\_of\_study\_slug': *self*.slug})

*# Пользовательский менеджер для выборки только опубликованных записей*

class PublishedManager(models.Manager):

    def get\_queryset(*self*):

*# Фильтруем записи по полю публикации с использованием перечисления*

*return* super().get\_queryset().filter(*is\_published*=ComputerScienceConcept.Status.PUBLISHED)

class ComputerScienceConcept(models.Model):

*# Класс-перечисление для статуса публикации*

    class Status(models.IntegerChoices):

        DRAFT = 0, 'Черновик'

        PUBLISHED = 1, 'Опубликовано'

    title = models.CharField(*max\_length*=255, *verbose\_name*="Название концепции")

    slug = models.SlugField(*max\_length*=255, *unique*=True, *db\_index*=True, *verbose\_name*="URL")

    description = models.TextField(*blank*=True, *verbose\_name*="Описание")

    difficulty = models.IntegerField(default=1, *verbose\_name*="Сложность (от 1 до 5)") *# Новое поле для сложности*

    time\_create = models.DateTimeField(*auto\_now\_add*=True, *verbose\_name*="Время создания")

    time\_update = models.DateTimeField(*auto\_now*=True, *verbose\_name*="Время обновления")

    is\_published = models.BooleanField(*choices*=Status.choices, default=Status.DRAFT, *verbose\_name*="Публикация")

    image = models.ImageField(*upload\_to*='concept\_images/%Y/%m/%d/', *blank*=True, *null*=True, *verbose\_name*='Изображение') *# New field*

*# Связь один-ко-многим с FieldOfStudy*

    field\_of\_study = models.ForeignKey(

        FieldOfStudy,

*on\_delete*=models.CASCADE,

*related\_name*='concepts',

*verbose\_name*="Область науки",

*null*=True,

*blank*=True

    )

    objects = models.Manager()         *# Стандартный менеджер*

    published = PublishedManager()       *# Пользовательский менеджер для опубликованных записей*

    class Meta:

        permissions = [

            ('can\_publish\_concept', 'Может публиковать концепцию'),

        ]

        ordering = ['-time\_create']

        indexes = [models.Index(*fields*=['-time\_create'])]

        verbose\_name = "Концепция компьютерных наук"

        verbose\_name\_plural = "Концепции компьютерных наук"

    def \_\_str\_\_(*self*):

*return* *self*.title

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.title)

*# Проверяем уникальность слага*

            base\_slug = *self*.slug

            num = 1

*while* ComputerScienceConcept.objects.filter(*slug*=*self*.slug).exists():

*self*.slug = f"{base\_slug}-{num}"

                num += 1

*# Обработка изменения размера изображения, если изображение присутствует*

*if* *self*.image and hasattr(*self*.image, 'file'):

*self*.image = resize\_image(*self*.image)

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('cs:concept\_detail', *kwargs*={'concept\_slug': *self*.slug})

*# Модель для расширенной информации о концепции (OneToOne)*

class ConceptDetail(models.Model):

    concept = models.OneToOneField(

        ComputerScienceConcept,

*on\_delete*=models.CASCADE,

*related\_name*='detail',

*verbose\_name*="Концепция"

    )

    core\_technologies = models.TextField(*blank*=True, *verbose\_name*="Ключевые технологии")

    prerequisites = models.TextField(*blank*=True, *verbose\_name*="Предварительные условия")

    estimated\_learning\_time = models.PositiveIntegerField(*verbose\_name*="Примерное время изучения (часы)", *null*=True, *blank*=True)

    class Meta:

        verbose\_name = "Детали концепции"

        verbose\_name\_plural = "Детали концепций"

    def \_\_str\_\_(*self*):

*return* f"Детали {*self*.concept.title}"

*# Модель для тегов (Many-to-Many)*

class Tag(models.Model):

    name = models.CharField(*max\_length*=50, *unique*=True, *verbose\_name*="Тег")

    slug = models.SlugField(*max\_length*=50, *unique*=True, *db\_index*=True, *verbose\_name*="URL тега") *# Убрал default='default-slug'*

    class Meta:

        verbose\_name = "Тег"

        verbose\_name\_plural = "Теги"

    def \_\_str\_\_(*self*):

*return* *self*.name

    def save(*self*, \**args*, \*\**kwargs*):

*if* not *self*.slug:

*self*.slug = slugify(*self*.name)

        super().save(\**args*, \*\**kwargs*)

    def get\_absolute\_url(*self*):

*return* reverse('cs:concepts\_by\_tag', *kwargs*={'tag\_slug': *self*.slug})

*# Добавляем ManyToManyField в ComputerScienceConcept после определения Tag*

ComputerScienceConcept.add\_to\_class('tags', models.ManyToManyField(

    Tag,

*related\_name*='concepts',

*verbose\_name*="Теги",

*blank*=True

))

*# Модель для комментариев*

*from* django.contrib.auth *import* get\_user\_model

User = get\_user\_model()

class Comment(models.Model):

    concept   = models.ForeignKey(ComputerScienceConcept, *on\_delete*=models.CASCADE, *related\_name*='comments', *verbose\_name*='Концепция')

    author  = models.ForeignKey(User, *on\_delete*=models.CASCADE, *related\_name*='comments', *verbose\_name*='Автор')

    text    = models.TextField(*verbose\_name*='Комментарий')

    created = models.DateTimeField(*auto\_now\_add*=True)

    class Meta:

        ordering = ['created']

        verbose\_name = "Комментарий"

        verbose\_name\_plural = "Комментарии"

    def \_\_str\_\_(*self*):

*return* f"Комментарий от {*self*.author} к {*self*.concept.title}"

tests.py:

*from* django.test *import* TestCase

urls.py:  
*from* django.urls *import* path

*from* .views *import* (

    HomeView, AboutView, ConceptDetailView,

    AddConceptCustomView, ConceptCreateView,

    ConceptUpdateView, ConceptDeleteView, UploadFileView, FieldOfStudyDetailView, ConceptByTagListView

)

app\_name = 'cs'

urlpatterns = [

    path('',              HomeView.as\_view(),       *name*='home'),

    path('about/',        AboutView.as\_view(),      *name*='about'),

    path('concepts/<slug:concept\_slug>/', ConceptDetailView.as\_view(), *name*='concept\_detail'),

    path('add-custom/',   AddConceptCustomView.as\_view(), *name*='add\_concept\_custom'),

    path('add-model/',    ConceptCreateView.as\_view(),   *name*='add\_concept\_model'),

    path('edit/<slug:concept\_slug>/',   ConceptUpdateView.as\_view(), *name*='edit\_concept'),

    path('delete/<slug:concept\_slug>/', ConceptDeleteView.as\_view(), *name*='delete\_concept'),

    path('upload/',       UploadFileView.as\_view(),  *name*='upload\_file'),

    path('field/<slug:field\_of\_study\_slug>/', FieldOfStudyDetailView.as\_view(), *name*='field\_of\_study\_detail'),

    path('tag/<slug:tag\_slug>/', ConceptByTagListView.as\_view(), *name*='concepts\_by\_tag'),

]

utils.py:

*from* django.views.generic.base *import* ContextMixin

MENU = [

    {'title': 'Главная',              'url\_name': 'cs:home'},

    {'title': 'О сайте',              'url\_name': 'cs:about'},

    {'title': 'Добавить (Form)',      'url\_name': 'cs:add\_concept\_custom'},

    {'title': 'Добавить (ModelForm)', 'url\_name': 'cs:add\_concept\_model'},

    {'title': 'Загрузка файла',       'url\_name': 'cs:upload\_file'},

]

class DataMixin(ContextMixin):

    """

    Добавляет в контекст:

      - global menu

      - title, если задано

      - page\_range для ListView с пагинацией

    """

    title = None

    def get\_context\_data(*self*, \*, *object\_list*=None, \*\**kwargs*):

*# Сначала получаем весь контекст от родительских классов,*

*# в том числе paginator, page\_obj, object\_list*

        context = super().get\_context\_data(*object\_list*=*object\_list*, \*\**kwargs*)

*# Базовый контекст*

        context['menu'] = MENU

*if* *self*.title:

            context['title'] = *self*.title

*# Если есть пагинация — вычисляем ограниченный диапазон страниц*

        paginator = context.get('paginator')

        page\_obj = context.get('page\_obj')

*if* paginator and page\_obj:

            total = paginator.num\_pages

            current = page\_obj.number

            window = 2

            start = max(current - window, 1)

            end   = min(current + window, total)

            context['page\_range'] = range(start, end + 1)

*return* context

*from* PIL *import* Image

*from* io *import* BytesIO

*from* django.core.files.uploadedfile *import* InMemoryUploadedFile

def resize\_image(*image\_field*, *size*=(512, 512), *quality*=80):

*if* not *image\_field*:

*return* *image\_field*

    img = Image.open(*image\_field*)

    img.thumbnail(*size*, Image.LANCZOS)

*# Создаем буфер для сохранения изображения*

    thumb\_io = BytesIO()

    img\_format = img.format *if* img.format *else* 'JPEG' *# Определяем формат или по умолчанию JPEG*

    img.save(thumb\_io, *format*=img\_format, *quality*=*quality*)

    thumb\_io.seek(0)

*# Создаем новый InMemoryUploadedFile, чтобы его можно было сохранить*

    new\_image = InMemoryUploadedFile(

        thumb\_io,

        'ImageField',

*image\_field*.name, *# Сохраняем оригинальное имя файла*

        f'image/{img\_format.lower()}', *# Определяем content\_type на основе формата Pillow*

        thumb\_io.tell(),

        None

    )

*return* new\_image

views.py:

*from* django.urls *import* reverse\_lazy

*from* django.views.generic *import* (

    TemplateView, ListView, DetailView,

    FormView, CreateView, UpdateView, DeleteView

)

*from* django.contrib.auth.mixins *import* LoginRequiredMixin

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

*from* .forms *import* ConceptForm, ConceptModelForm, UploadForm, CommentForm

*from* .utils *import* DataMixin

class HomeView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/index.html'

    context\_object\_name = 'concepts'

    paginate\_by = 5

    queryset = ComputerScienceConcept.published.all()

    title = 'Главная'

class AboutView(DataMixin, TemplateView):

    template\_name = 'cs/about.html'

    title = 'О сайте'

class ConceptDetailView(DataMixin, DetailView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_detail.html'

    context\_object\_name = 'concept'

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

        concept = *self*.get\_object()

        context['title'] = concept.title

        context['comments'] = concept.comments.all()

        context['comment\_form'] = CommentForm()

*return* context

    def post(*self*, *request*, \**args*, \*\**kwargs*):

*self*.object = *self*.get\_object()

        form = CommentForm(*request*.POST)

*if* form.is\_valid():

            comment = form.save(*commit*=False)

            comment.concept = *self*.object

            comment.author = *request*.user

            comment.save()

*return* *self*.get(*request*, \**args*, \*\**kwargs*) *# Перенаправляем на GET запрос после успешной отправки*

*else*:

*# Если форма недействительна, возвращаем ее с ошибками*

            context = *self*.get\_context\_data()

            context['comment\_form'] = form

*return* *self*.render\_to\_response(context)

class AddConceptCustomView(LoginRequiredMixin, DataMixin, FormView):

    form\_class = ConceptForm

    template\_name = 'cs/add\_concept\_custom.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (Form)'

    login\_url = reverse\_lazy('users:login')

    def form\_valid(*self*, *form*):

        cd = *form*.cleaned\_data

        concept = ComputerScienceConcept.objects.create(

*title*=cd['title'],

*slug*=cd['title'].lower().replace(' ', '-'),

*description*=cd['description'],

*difficulty*=cd['difficulty'],

*is\_published*=ComputerScienceConcept.Status.DRAFT

        )

        image = cd.get('image')

*if* image:

            concept.image = image

            concept.save()

*return* super().form\_valid(*form*)

class ConceptCreateView(LoginRequiredMixin, DataMixin, CreateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    title = 'Добавить (ModelForm)'

    login\_url = reverse\_lazy('users:login')

class ConceptUpdateView(LoginRequiredMixin, DataMixin, UpdateView):

    model = ComputerScienceConcept

    form\_class = ConceptModelForm

    template\_name = 'cs/add\_concept\_model.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Редактировать концепцию'

    login\_url = reverse\_lazy('users:login')

class ConceptDeleteView(LoginRequiredMixin, DataMixin, DeleteView):

    model = ComputerScienceConcept

    template\_name = 'cs/concept\_confirm\_delete.html'

    success\_url = reverse\_lazy('cs:home')

    slug\_field = 'slug'

    slug\_url\_kwarg = 'concept\_slug'

    title = 'Удалить концепцию'

    login\_url = reverse\_lazy('users:login')

class UploadFileView(LoginRequiredMixin, DataMixin, FormView):

    form\_class = UploadForm

    template\_name = 'cs/upload.html'

    success\_url = reverse\_lazy('cs:upload\_file')

    title = 'Загрузка файла'

    login\_url = reverse\_lazy('users:login')

    def form\_valid(*self*, *form*):

*form*.save\_file()

*return* super().form\_valid(*form*)

class FieldOfStudyDetailView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/field\_of\_study\_detail.html' *# Шаблон для отображения концепций по области*

    context\_object\_name = 'concepts'

    paginate\_by = 5

    def get\_queryset(*self*):

*# Получаем слаг области науки из URL*

        field\_of\_study\_slug = *self*.kwargs['field\_of\_study\_slug']

*# Фильтруем концепции по этой области*

*return* ComputerScienceConcept.published.filter(*field\_of\_study\_\_slug*=field\_of\_study\_slug)

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

*# Получаем объект FieldOfStudy, чтобы отобразить его название*

        field\_of\_study\_slug = *self*.kwargs['field\_of\_study\_slug']

        field\_of\_study = FieldOfStudy.objects.get(*slug*=field\_of\_study\_slug)

        context['title'] = f"Концепции в области: {field\_of\_study.name}"

        context['field\_of\_study'] = field\_of\_study

*return* context

class ConceptByTagListView(DataMixin, ListView):

    model = ComputerScienceConcept

    template\_name = 'cs/concepts\_by\_tag.html'  *# Предполагается, что этот шаблон существует*

    context\_object\_name = 'concepts'

    paginate\_by = 5

    def get\_queryset(*self*):

        tag\_slug = *self*.kwargs['tag\_slug']

*return* ComputerScienceConcept.published.filter(*tags\_\_slug*=tag\_slug)

    def get\_context\_data(*self*, \*\**kwargs*):

        context = super().get\_context\_data(\*\**kwargs*)

        tag\_slug = *self*.kwargs['tag\_slug']

        tag = Tag.objects.get(*slug*=tag\_slug)

        context['title'] = f"Концепции по тегу: {tag.name}"

        context['tag'] = tag

*return* context

cs\_ty/

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

*from* django.conf *import* settings

*from* django.conf.urls.static *import* static

urlpatterns = [

    path('', include('cs.urls')),

    path('admin/', admin.site.urls),

    path('users/', include('users.urls', *namespace*='users')),

]

*if* settings.DEBUG:

    urlpatterns += static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT)

users/

Login.html:

{% extends 'cs/base.html' %}

{% block title %}Вход{% endblock %}

{% block content %}

  <h2>Вход на сайт</h2>

  <form method="post">

    {% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>

      {{ form.username.label\_tag }}<br>

      {{ form.username }}<br>

      {{ form.username.errors }}

    </p>

    <p>

      {{ form.password.label\_tag }}<br>

      {{ form.password }}<br>

      {{ form.password.errors }}

    </p>

*{# Скрытое поле next сохраняет URL запроса #}*

    <input type="hidden" name="next" value="{{ next }}">

    <button type="submit">Войти</button>

  </form>

{% endblock %}

Password\_change\_done.html:

{% extends 'cs/base.html' %}

{% block title %}Пароль изменён{% endblock %}

{% block content %}

  <h2>Пароль успешно обновлён!</h2>

  <p><a href="{% url 'users:profile' %}">Вернуться в профиль</a></p>

{% endblock %}

Password\_change\_form.html:

{% extends 'cs/base.html' %}

{% block title %}Смена пароля{% endblock %}

{% block content %}

  <h2>Сменить пароль</h2>

  <form method="post">

    {% csrf\_token %}

    {{ form.non\_field\_errors }}

    <p>{{ form.old\_password.label\_tag }}<br>{{ form.old\_password }}{{ form.old\_password.errors }}</p>

    <p>{{ form.new\_password1.label\_tag }}<br>{{ form.new\_password1 }}{{ form.new\_password1.errors }}</p>

    <p>{{ form.new\_password2.label\_tag }}<br>{{ form.new\_password2 }}{{ form.new\_password2.errors }}</p>

    <button type="submit">Сменить пароль</button>

  </form>

{% endblock %}

Password\_reset\_complete.html:

{% extends 'cs/base.html' %}

{% block title %}Пароль сброшен{% endblock %}

{% block content %}

  <h2>Ваш пароль успешно изменён!</h2>

  <p><a href="{% url 'users:login' %}">Войти</a></p>

{% endblock %}

Password\_reset\_confirm.html:

{% extends 'cs/base.html' %}

{% block title %}Установка нового пароля{% endblock %}

{% block content %}

  <h2>Установите новый пароль</h2>

  <form method="post">{% csrf\_token %}

    {{ form.new\_password1.label\_tag }}<br>

    {{ form.new\_password1 }}<br><br>

    {{ form.new\_password2.label\_tag }}<br>

    {{ form.new\_password2 }}<br><br>

    <button type="submit">Сохранить</button>

  </form>

{% endblock %}

Password\_reset\_done.html:

{% extends 'cs/base.html' %}

{% block title %}Письмо отправлено{% endblock %}

{% block content %}

  <h2>Письмо с инструкциями отправлено!</h2>

  <p>Проверьте вашу почту.</p>

{% endblock %}

Password\_reset\_email.html:

{% autoescape off %}

Чтобы сбросить пароль, перейдите по ссылке:

{{ protocol }}://{{ domain }}{% url 'users:password\_reset\_confirm' uidb64=uid token=token %}

{% endautoescape %}

Password\_reset\_form.html:

{% extends 'cs/base.html' %}

{% block title %}Восстановление пароля{% endblock %}

{% block content %}

  <h2>Восстановление пароля</h2>

  <form method="post">{% csrf\_token %}

    {{ form.email.label\_tag }}<br>

    {{ form.email }}<br>

    {{ form.email.errors }}<br><br>

    <button type="submit">Отправить письмо</button>

  </form>

{% endblock %}

Profile.html:

{% extends 'cs/base.html' %}

{% block title %}Профиль{% endblock %}

{% block content %}

  <h2>Ваш профиль</h2>

  <p><strong>Логин:</strong> {{ user.username }}</p>

  <p><strong>E-mail:</strong> {{ user.email }}</p>

  <p><a href="{% url 'users:password\_change' %}">Сменить пароль</a></p>

{% endblock %}

Backends.py:

*from* django.contrib.auth.backends *import* ModelBackend

*from* django.contrib.auth *import* get\_user\_model

User = get\_user\_model()

class EmailOrUsernameBackend(ModelBackend):

    """

    Аутентификация: сначала пробуем по email, если не нашли — по username.

    """

    def authenticate(*self*, *request*, *username*=None, *password*=None, \*\**kwargs*):

*if* *username* is None or *password* is None:

*return* None

        user = None

*# попробуем e-mail*

*if* '@' in *username*:

            user = User.objects.filter(*email\_\_iexact*=*username*, *is\_active*=True).first()

*# если не нашли по e-mail, попробуем username*

*if* user is None:

            user = User.objects.filter(*username\_\_iexact*=*username*, *is\_active*=True).first()

*if* user and user.check\_password(*password*):

*return* user

*return* None

Forms.py:

*from* django *import* forms

*from* django.contrib.auth.forms *import* AuthenticationForm

class EmailOrUsernameAuthenticationForm(AuthenticationForm):

    """

    Форма логина: в одном поле пользователь вводит либо e-mail, либо username.

    """

    username = forms.CharField(

*label*='E-mail или логин',

*widget*=forms.TextInput(*attrs*={'autofocus': True})

    )

Urls.py:

*from* django.urls *import* path

*from* . *import* views

app\_name = 'users'

urlpatterns = [

    path('login/',     views.login\_user,  *name*='login'),

    path('logout/',    views.logout\_user, *name*='logout'),

*# Сброс пароля:*

    path('password-reset/',

         views.CustomPasswordResetView.as\_view(),

*name*='password\_reset'),

    path('password-reset/done/',

         views.CustomPasswordResetDoneView.as\_view(),

*name*='password\_reset\_done'),

    path('reset/<uidb64>/<token>/',

         views.CustomPasswordResetConfirmView.as\_view(),

*name*='password\_reset\_confirm'),

    path('reset/done/',

         views.CustomPasswordResetCompleteView.as\_view(),

*name*='password\_reset\_complete'),

*# Профиль и смена пароля:*

    path('profile/',

         views.ProfileView.as\_view(),

*name*='profile'),

    path('password-change/',

         views.CustomPasswordChangeView.as\_view(),

*name*='password\_change'),

    path('password-change/done/',

         views.CustomPasswordChangeDoneView.as\_view(),

*name*='password\_change\_done'),

]

views.py:

*from* django.contrib.auth.views *import* (

    PasswordResetView, PasswordResetDoneView,

    PasswordResetConfirmView, PasswordResetCompleteView

)

*from* django.urls *import* reverse\_lazy

*from* django.contrib.auth *import* login, logout

*from* django.shortcuts *import* render, redirect

*from* django.contrib.auth.mixins *import* LoginRequiredMixin

*from* django.views.generic *import* TemplateView

*from* django.contrib.auth.views *import* PasswordChangeView, PasswordChangeDoneView

*from* .forms *import* EmailOrUsernameAuthenticationForm

def login\_user(*request*):

    """

    Логин по e-mail или username с учётом next-параметра.

    """

    next\_url = *request*.GET.get('next') or *request*.POST.get('next') or ''

*if* *request*.method == 'POST':

        form = EmailOrUsernameAuthenticationForm(*request*, *data*=*request*.POST)

*if* form.is\_valid():

            login(*request*, form.get\_user())

*return* redirect(next\_url *or* 'cs:home')

*else*:

        form = EmailOrUsernameAuthenticationForm(*request*)

*return* render(*request*, 'users/login.html', {'form': form, 'next': next\_url})

def logout\_user(*request*):

    """

    Выход и редирект на главную.

    """

    logout(*request*)

*return* redirect('cs:home')

class ProfileView(LoginRequiredMixin, TemplateView):

    """

    Страница профиля текущего пользователя.

    """

    login\_url = reverse\_lazy('users:login')

    template\_name = 'users/profile.html'

    def get\_context\_data(*self*, \*\**kwargs*):

        ctx = super().get\_context\_data(\*\**kwargs*)

        ctx['user'] = *self*.request.user

*return* ctx

class CustomPasswordChangeView(LoginRequiredMixin, PasswordChangeView):

    """

    Форма смены пароля.

    """

    login\_url = reverse\_lazy('users:login')

    template\_name = 'users/password\_change\_form.html'

    success\_url = reverse\_lazy('users:password\_change\_done')

class CustomPasswordChangeDoneView(LoginRequiredMixin, PasswordChangeDoneView):

    """

    Страница подтверждения смены пароля.

    """

    login\_url = reverse\_lazy('users:login')

    template\_name = 'users/password\_change\_done.html'

class CustomPasswordResetView(PasswordResetView):

    """

    Шаг 1: форма запроса сброса пароля (ввод e-mail, отправка письма).

    """

    template\_name = 'users/password\_reset\_form.html'

    email\_template\_name = 'users/password\_reset\_email.html'

    subject\_template\_name = 'users/password\_reset\_subject.txt'

    success\_url = reverse\_lazy('users:password\_reset\_done')

class CustomPasswordResetDoneView(PasswordResetDoneView):

    """

    Шаг 2: страница, показываемая после отправки письма.

    """

    template\_name = 'users/password\_reset\_done.html'

class CustomPasswordResetConfirmView(PasswordResetConfirmView):

    """

    Шаг 3: форма установки нового пароля (переход по ссылке из письма).

    """

    template\_name = 'users/password\_reset\_confirm.html'

    success\_url = reverse\_lazy('users:password\_reset\_complete')

class CustomPasswordResetCompleteView(PasswordResetCompleteView):

    """

    Шаг 4: страница, показываемая после успешного сброса пароля.

    """

    template\_name = 'users/password\_reset\_complete.html'