Московский государственный технический университет им. Н.Э. Баумана

Факультет информатика и системы управления

Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»

Отчет по ДЗ

Выполнил:
студент группы ИУ5-32Б:
Опарина А.А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г

Текст программы

models.py

```
from django.db import models
from django.utils.text import slugify
from django.contrib.auth.models import AbstractUser
from django.db.models import Sum


class DefaultModel(models.Model):
    class Meta:
        abstract = True

    is_active = models.BooleanField(default=True, verbose_name="Активен?")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Время
создания", editable=False, null=True)
    updated_at = models.DateTimeField(auto_now=True, verbose_name="Время
обновления", editable=False, null=True)


class User(AbstractUser):
    avatar = models.ImageField(upload_to='avatars', null=True, blank=True)

    class Meta:
        verbose_name = 'Пользователь'
        verbose_name_plural = 'Пользователи'


from django.db.models import Sum

class QuestionManager(models.Manager):
    def new(self):
        return self.all().order_by('-created_at')

    from django.db.models import Sum

    def hot(self):
        return self.annotate(score_total=Sum('likes__value')).order_by('-
score_total', '-created_at')

    def by_tag(self, tag):
        return self.filter(tags=tag).order_by('-created_at')



class Question(DefaultModel):
    class Meta:
        verbose_name = 'Вопрос'
        verbose_name_plural = 'Вопросы'

    slug = models.SlugField(max_length=200, unique=True, null=True, blank=True)
    title = models.CharField(max_length=200)
    text = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    tags = models.ManyToManyField('Tag', blank=True, verbose_name="Теги")
    objects = QuestionManager()

    def __str__(self):
        return str(self.title)

    def save(self, *args, **kwargs):
        self.slug = slugify(self.title, allow_unicode=True)
        return super(Question, self).save(*args, **kwargs)

    @property
    def score(self):
        result = self.likes.aggregate(score=Sum('value'))['score'] or 0
        return result


class Answer(DefaultModel):
    class Meta:
        verbose_name = 'Ответ'
        verbose_name_plural = 'Ответы'

    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    answer_text = models.TextField()

    is_correct = models.BooleanField(default=False)

    def __str__(self):
        return "Ответ на вопрос ID=" + str(self.question_id)

    @property
    def score(self):
        result = self.likes.aggregate(score=Sum('value'))['score'] or 0
        return result


class Tag(models.Model):
    class Meta:
        verbose_name = 'Тег'
        verbose_name_plural = 'Теги'

    title = models.CharField(max_length=200, verbose_name="Название тега")

    def __str__(self):
        return self.title


class AnswerLike(models.Model):
    class Meta:
        verbose_name = 'Лайк ответа'
        verbose_name_plural = 'Лайки ответов'
        unique_together=('answer', 'user')

    LIKE = 1
    DISLIKE = -1
    VALUE_CHOICES = [
        (LIKE, 'лайк'),
        (DISLIKE, 'дизлайк'),
    ]

    answer = models.ForeignKey(Answer, on_delete=models.CASCADE,
related_name="likes")
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    value = models.SmallIntegerField(
        choices=VALUE_CHOICES,
        default=LIKE,
        verbose_name="Лайк/дизлайк"
    )

    def __str__(self):
        return f"Был поставлен {dict(self.VALUE_CHOICES).get(self.value)}
пользователем {self.user} на ответ {self.answer}"
# HACK: сделать так, чтобы в админке можно было менять лайк на дизлайк

class QuestionLike(models.Model):
    class Meta:
        verbose_name = 'Лайк вопроса'
        verbose_name_plural = 'Лайки вопросов'
        unique_together=('question', 'user')

    LIKE = 1
    DISLIKE = -1
    VALUE_CHOICES = [
        (LIKE, 'Like'),
        (DISLIKE, 'Dislike'),
    ]

    question = models.ForeignKey(Question, on_delete=models.CASCADE,
related_name="likes")
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    value = models.SmallIntegerField(
        choices=VALUE_CHOICES,
        default=LIKE,
        verbose_name="Лайк/дизлайк"
    )

    def __str__(self):
        return f"Был поставлен {dict(self.VALUE_CHOICES).get(self.value)}
пользователем {self.user}"

```

views.py

```
from django.shortcuts import render
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from django.views.generic import TemplateView, View
from django.shortcuts import get_object_or_404
from django.utils.decorators import method_decorator

from django.views.decorators.cache import never_cache
from django.contrib.auth.decorators import login_required

from app.models import User, Question, Answer, Tag, AnswerLike, QuestionLike

from app.forms import LoginForm, QuestionForm, RegisterForm, AnswerForm

from django.http import JsonResponse

from django.shortcuts import redirect

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

from datetime import datetime

def get_page_range(page, paginator):
    current = page.number
    last = paginator.num_pages

    start_pages = [1, 2, 3]
    end_pages = [last - 2, last - 1, last]

    middle_pages = [
        current - 1,
        current,
        current + 1
    ]

    pages = set()

    for p in start_pages + middle_pages + end_pages:
        if 1 <= p <= last:
            pages.add(p)

    pages = sorted(pages)
```

```python
    final = []
    for i, p in enumerate(pages):
        final.append(p)

        if i < len(pages) - 1:
            next_p = pages[i+1]
            if next_p - p > 1:
                final.append("...")

    return final


def paginate(objects_list, page_number, per_page=10):
    paginator = Paginator(objects_list, per_page)
    try:
        page = paginator.page(page_number)
    except PageNotAnInteger:
        page = paginator.page(1)
    except EmptyPage:
        page = paginator.page(paginator.num_pages)
    return page

@method_decorator(never_cache, name='dispatch')
class QuestionListView(TemplateView):
    template_name = "app/index.html"
    QUESTIONS_PER_PAGE = 4

    def get_queryset_and_meta(self):
        """
        Возвращает:
        queryset, page_type, page_title, tag (или None)
        """
        filter_type = self.request.GET.get('filter')
        tag_name = self.request.GET.get('tag_name')

        if filter_type == 'hot':
            return (
                Question.objects.hot(),
                'hot',
                'Hot Questions',
                None
            )

        if filter_type == 'tag' and tag_name:
            tag = get_object_or_404(Tag, title=tag_name)
            return (
                Question.objects.by_tag(tag),
                'tag',
                f'Tag: {tag.title}',
                tag
            )
        return (
            Question.objects.new(),
            'new',
            'New Questions',
            None
        )

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        page_number = self.kwargs.get('page') or self.request.GET.get('page', 1)
        try:
            page_number = int(page_number)
        except ValueError:
            page_number = 1

        queryset, page_type, page_title, tag = self.get_queryset_and_meta()

        paginator = Paginator(queryset, self.QUESTIONS_PER_PAGE)

        try:
            page_obj = paginator.page(page_number)
        except PageNotAnInteger:
            page_obj = paginator.page(1)
        except EmptyPage:
            page_obj = paginator.page(paginator.num_pages)

        context.update({
            "questions": page_obj.object_list,
            "page": page_obj,
            "pages": get_page_range(page_obj, paginator),
            "max_page": paginator.num_pages,

            "page_type": page_type,
            "page_title": page_title,
            "tag": tag,

            "filter": self.request.GET.get('filter'),
            "tag_name": self.request.GET.get('tag_name'),

            "tags": Tag.objects.all(),
        })

        return context

#TODO: добавить дату
@method_decorator(login_required, name='dispatch')
class QuestionView(TemplateView):
    http_method_names = ['get', 'post']
    template_name = 'app/question.html'
    ANSWERS_PER_PAGE = 4

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        question_id = kwargs.get('question_id')
        page_number = kwargs.get('page') or int(self.request.GET.get('page', 1))

        if 'form' not in context:
            context['form'] = AnswerForm()

        question = Question.objects.prefetch_related('tags').get(id=question_id)

        answers_list = Answer.objects.filter(question=question).order_by('-
created_at')

        paginator = Paginator(answers_list, self.ANSWERS_PER_PAGE)
        try:
            page_obj = paginator.page(page_number)
        except PageNotAnInteger:
            page_obj = paginator.page(1)
        except EmptyPage:
            page_obj = paginator.page(paginator.num_pages)

        page_range = get_page_range(page_obj, paginator)

        context['question'] = question
        context['answers'] = page_obj.object_list
        context['page'] = page_obj
        context['page_range'] = page_range
        context['answers_per_page'] = self.ANSWERS_PER_PAGE
        context['max_page'] = paginator.num_pages

        return context

    def dispatch(self, request, *args, **kwargs):
        print(request)
        return super(QuestionView, self).dispatch(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        question_id = kwargs.get('question_id')

        question = get_object_or_404(Question, id=question_id)

        form = AnswerForm(request.POST)

        if form.is_valid():
            answer = form.save(commit=False)

            answer.author = request.user
            answer.question = question
            answer.save()

            messages.add_message(request, messages.SUCCESS, "Ваш ответ
опубликован!")
            return redirect("question", question_id=question_id)

        return render(request, "app/question.html", {"form": form})

def ask(request):
    return render(request, "app/ask.html")

def settings(request):
    if request.method == 'POST':
        if 'avatar' in request.FILES:
            user = request.user
            user.avatar = request.FILES['avatar']
            user.save()
            return redirect('settings')
    return render(request, "app/settings.html")

def tag(request, tag_name, page):
    questions = []
    for i in range(1, 30):
        questions.append({
            'id': i,
            'title': f'What is Frutiger Aero? {i}',
            'text': 'Guys, tell me more about the aesthetics of the early
internet!',
            'votes': 42,
            'count': 3,
            'tags': ['frutiger_aero', '2000s'],
        })

    paginator = Paginator(questions, 5)
    try:
        page_obj = paginator.page(page)
    except PageNotAnInteger:
        page_obj = paginator.page(1)
    except EmptyPage:
        page_obj = paginator.page(paginator.num_pages)

    page_range = get_page_range(page_obj, paginator)

    return render(request, "tag.html", {
        'tag_name': tag_name,
        'questions': page_obj.object_list,
        'page': page_obj,
        'page_range': page_range
    })
```

```
class AuthView(TemplateView):
    http_method_names = ['get', 'post']
    template_name = 'app/login.html'

    def get_context_data(self, **kwargs):
        form = LoginForm()
        context = super(AuthView, self).get_context_data(**kwargs)
        context['form'] = form
        return context

    def post(self, request, *args, **kwargs):
        form = LoginForm(request.POST, request.FILES)

        if form.is_valid():
            login(request, form.user)
            messages.add_message(request, messages.SUCCESS, "Вы успешно 
авторизованы в вашем аккаунте")
            return redirect("/")
        return render(request, "app/login.html", {"form": form})

@login_required
def logout_view(request):
    logout(request)
    return redirect("/login")

@method_decorator(login_required, name='dispatch')
class CreateQuestionView(TemplateView):
    http_method_names = ['get', 'post']
    template_name = 'app/ask.html'

    def get_context_data(self, **kwargs):
        context = super(CreateQuestionView, self).get_context_data(**kwargs)
        context['form'] = QuestionForm()
        return context

    def post(self, request, *args, **kwargs):
        form = QuestionForm(request.POST)
        if form.is_valid():
            question = form.save(commit=False)
            question.author = request.user
            #question.cover = request.FILES['image']
            question.save()
            print(request)
            return redirect('index')

        return render(request, 'app/ask.html', {'form': form})

class RegisterView(TemplateView):
    http_method_names = ['get', 'post']
    template_name = 'app/signup.html'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['form'] = RegisterForm()
        return context

    def post(self, request, *args, **kwargs):
        form = RegisterForm(request.POST, request.FILES)

        if form.is_valid():
            user = User.objects.create_user(
                username=form.cleaned_data['username'],
                email=form.cleaned_data['email'],
                password=form.cleaned_data['password'],
            )

            if form.cleaned_data.get('avatar'):
                user.avatar = form.cleaned_data['avatar']
                user.save()

            login(request, user)
            messages.success(request, "Вы успешно зарегистрированы!")
            return redirect("/")

@method_decorator(login_required, name='dispatch')
class QuestionLikeAPI(View):
    http_method_names = ["post"]

    def apply_vote(self, user, question, value):
        like = QuestionLike.objects.filter(user=user, question=question).first()
        old_value = like.value if like else 0
        new_value = value

        # если голос не меняется
        if old_value == new_value:
            return 0

        # снять голос
        if new_value == 0 and like:
            like.delete()
            return -old_value

        # поставить голос впервые
        if not like:
            QuestionLike.objects.create(user=user, question=question,
value=new_value)
            return new_value

        # сменить лайк на дизлайк
        like.value = new_value
        like.save(update_fields=['value'])
        return new_value - old_value
    def post(self, request, *args, **kwargs):
        user = request.user
        question_id = kwargs.get('question_id')
        question = get_object_or_404(Question, id=question_id)

        if question.author == user:
            return JsonResponse({
                'success': False,
                'error': 'Вы являетесь автором вопроса.'
            }, status=400)

        if 'dislike' in request.path:
            value = -1
        elif 'like' in request.path:
            value = 1
        else:
            return JsonResponse({
                'success': False,
                'error': 'Некорректное действие.'
            }, status=400)

        delta = self.apply_vote(user=user, question=question, value=value)

        question.refresh_from_db()
        return JsonResponse({
            'success': True,
            'delta': delta,
            'score': question.score
        })




'''пока что коды классов QuestionLikeAPI и AnswerLikeAPI совпадают, но работают -
позже я избавлюсь от дублирования'''


@method_decorator(login_required, name='dispatch')
class AnswerLikeAPI(View):
    http_method_names = ["post"]

    def apply_vote(self, user, answer, value):
        like = AnswerLike.objects.filter(user=user, answer=answer).first()
        old_value = like.value if like else 0
        new_value = value

        if old_value == new_value:
            return 0

        if new_value == 0 and like:
```

```python
            like.delete()
            return -old_value

        if not like:
            AnswerLike.objects.create(user=user, answer=answer, value=new_value)
            return new_value

        like.value = new_value
        like.save(update_fields=['value'])
        return new_value - old_value

    def post(self, request, *args, **kwargs):
        user = request.user
        answer_id = kwargs.get('answer_id')
        answer = get_object_or_404(Answer, id=answer_id)

        if answer.author == user:
            return JsonResponse({
                'success': False,
                'error': 'Вы являетесь автором вопроса.'
            }, status=400)

        if 'dislike' in request.path:
            value = -1
        elif 'like' in request.path:
            value = 1
        else:
            return JsonResponse({
                'success': False,
                'error': 'Некорректное действие.'
            }, status=400)

        delta = self.apply_vote(user=user, answer=answer, value=value)

        answer.refresh_from_db()
        return JsonResponse({
            'success': True,
            'delta': delta,
            'score': answer.score
        })


@method_decorator(login_required, name='dispatch')
class MarkCorrectAnswerAPI(View):
    http_method_names = ["post"]

    def post(self, request, *args, **kwargs):
        answer_id = kwargs.get('answer_id')
        answer = get_object_or_404(Answer, id=answer_id)
        question = answer.question
        if question.author != request.user:
            return JsonResponse({
                'success': False,
                'error': 'Только автор вопроса может выбирать правильный ответ.'
            }, status=403)
        if answer.is_correct:
            answer.is_correct = False
        else:
            Answer.objects.filter(question=question,
is_correct=True).update(is_correct=False)
            answer.is_correct = True

        answer.save()

        return JsonResponse({
            'success': True,
            'is_correct': answer.is_correct
        })
```

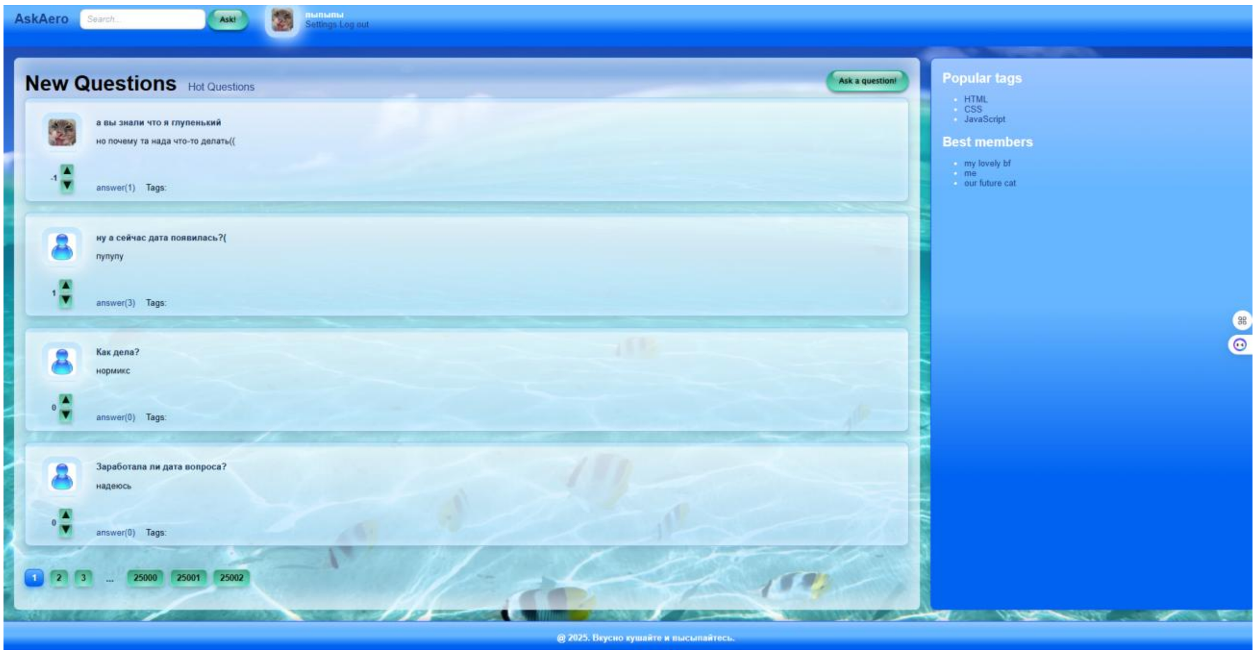
forms.py

```
from django import forms
from django.contrib.auth import authenticate
from django.forms.widgets import PasswordInput
from django.forms.widgets import FileInput

from app.models import Question, User, Answer

class LoginForm(forms.Form):
    username = forms.CharField(
        max_length=150,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Имя пользователя',
            'autocomplete': 'username'
        })
    )
    password = forms.CharField(
        max_length=128,
        widget=PasswordInput(attrs={
            'class': 'form-control',
            'placeholder': 'Пароль',
            'autocomplete': 'current-password'
        }),
    )

    def clean(self):
        cleaned_data = super().clean()
        username = cleaned_data.get('username')
        password = cleaned_data.get('password')

        if username and password:
            self.user = authenticate(username=username, password=password)
            if self.user is None:
                raise forms.ValidationError(
                    'Неверное имя пользователя или пароль'
                )

        return cleaned_data

class QuestionForm(forms.ModelForm):
    image = forms.ImageField(widget=FileInput, required=False)

    class Meta:
        model = Question
        fields = ('title', 'text') #'tags'

class RegisterForm(forms.Form):
    username = forms.CharField(
        max_length=150,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Имя пользователя',
            'autocomplete': 'username'
        })
    )
    email = forms.EmailField(
        max_length=150,
        widget=forms.TextInput(attrs={
            'class': 'form-control',
            'placeholder': 'Email',
            'autocomplete': 'email'
        })
    )
    password = forms.CharField(
        max_length=128,
        widget=forms.PasswordInput(attrs={
            'class': 'form-control',
            'placeholder': 'Пароль',
            'autocomplete': 'new-password'
        }),
    )
    password_repeat = forms.CharField(
        max_length=128,
        widget=forms.PasswordInput(attrs={
            'class': 'form-control',
            'placeholder': 'Повторите пароль',
            'autocomplete': 'new-password'
        }),
    )
    avatar = forms.ImageField(required=False)

    def clean_username(self):
        username = self.cleaned_data['username']
        if User.objects.filter(username=username).exists():
            raise forms.ValidationError("Пользователь с таким именем уже 
существует")
        return username

    def clean(self):
        cleaned_data = super().clean()
        password = cleaned_data.get('password')
        password_repeat = cleaned_data.get('password_repeat')

        if password and password_repeat and password != password_repeat:
            raise forms.ValidationError("Пароли не совпадают")

        return cleaned_data

class AnswerForm(forms.ModelForm):
    widgets = {
        'answer_text': forms.Textarea(attrs={
            'class': 'form-control',
            'placeholder': 'Введите ваш ответ здесь...',
            'rows': 5
        })
    }
    class Meta:
        model = Answer
        fields = ('answer_text',)
```

urls.py

```
from django.urls import path
from .views import QuestionListView, QuestionView, AuthView, CreateQuestionView, 
RegisterView, ask, settings, logout_view, QuestionLikeAPI, AnswerLikeAPI, 
MarkCorrectAnswerAPI


urlpatterns = [
    path('', QuestionListView.as_view(), name='index'),
    path('page/<int:page>/', QuestionListView.as_view(), name='index_page'),
    path('ask/', CreateQuestionView.as_view(), name='ask'),
    path('login/', AuthView.as_view(), name='login'),
    path('signup/', RegisterView.as_view(), name='signup'),
    path('settings/', settings, name='settings'),
    path('question/<int:question_id>/', QuestionView.as_view(), name='question'),
    path('logout/', logout_view, name="logout"),
    path('question/<int:question_id>/like', QuestionLikeAPI.as_view(),
name='question_like'),
    path('question/<int:question_id>/dislike', QuestionLikeAPI.as_view(),
name='question_dislike'),
    path('answer/<int:answer_id>/like', AnswerLikeAPI.as_view(),
name='answer_like'),
    path('answer/<int:answer_id>/dislike', AnswerLikeAPI.as_view(),
name='answer_dislike'),
    path('answer/<int:answer_id>/correct/', MarkCorrectAnswerAPI.as_view(),
name='mark_correct'),
]
```

Скриншот работы программы



New Questions Hot Questions

а вы знали что я глупенький
но почему та нада что-то делать((
-1
answer(1) Tags:

ну а сейчас дата появилась?(
пупупу
1
answer(3) Tags:

Как дела?
нормикс
0
answer(0) Tags:

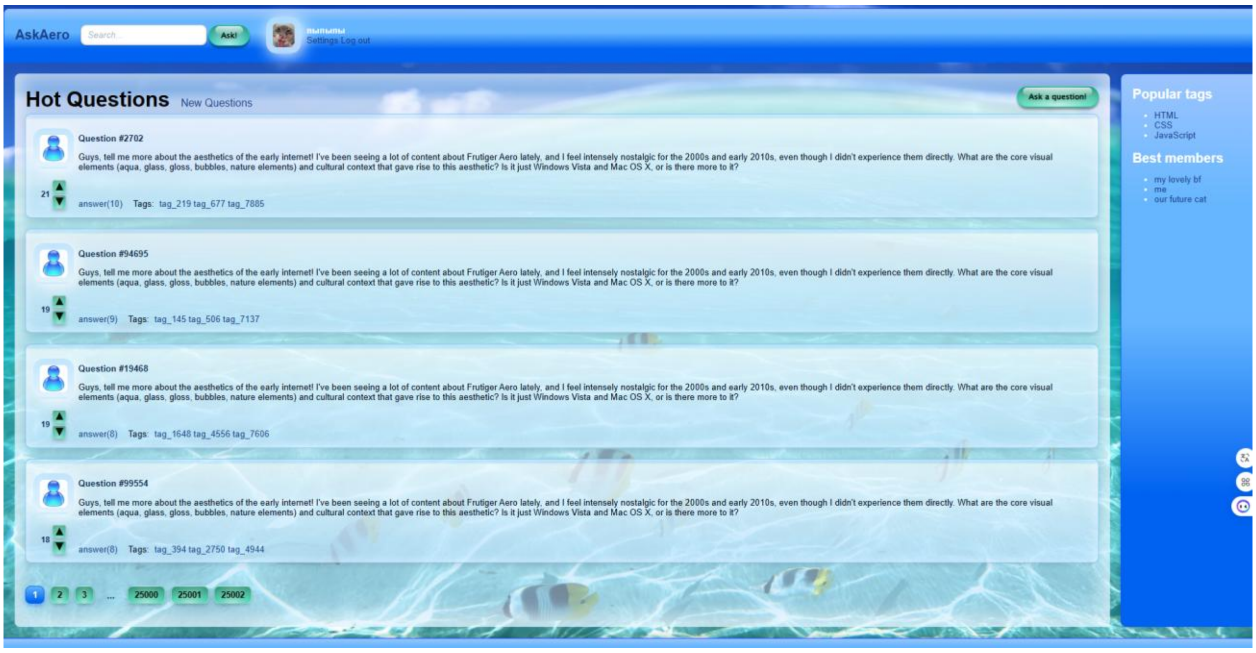
Заработала ли дата вопроса?
надеюсь
0
answer(0) Tags:

Popular tags
- HTML
- CSS
- JavaScript

Best members
- my lovely bf
- me
- our future cat

@ 2025. Вкусно кушайте и высыпайтесь.

Рис. 1. Главная страница



Hot Questions New Questions

Question #2702

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

21 answer(10) Tags: tag_219 tag_677 tag_7885

Question #94695

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

19 answer(9) Tags: tag_145 tag_506 tag_7137

Question #19468

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

19 answer(8) Tags: tag_1648 tag_4556 tag_7606

Question #99554

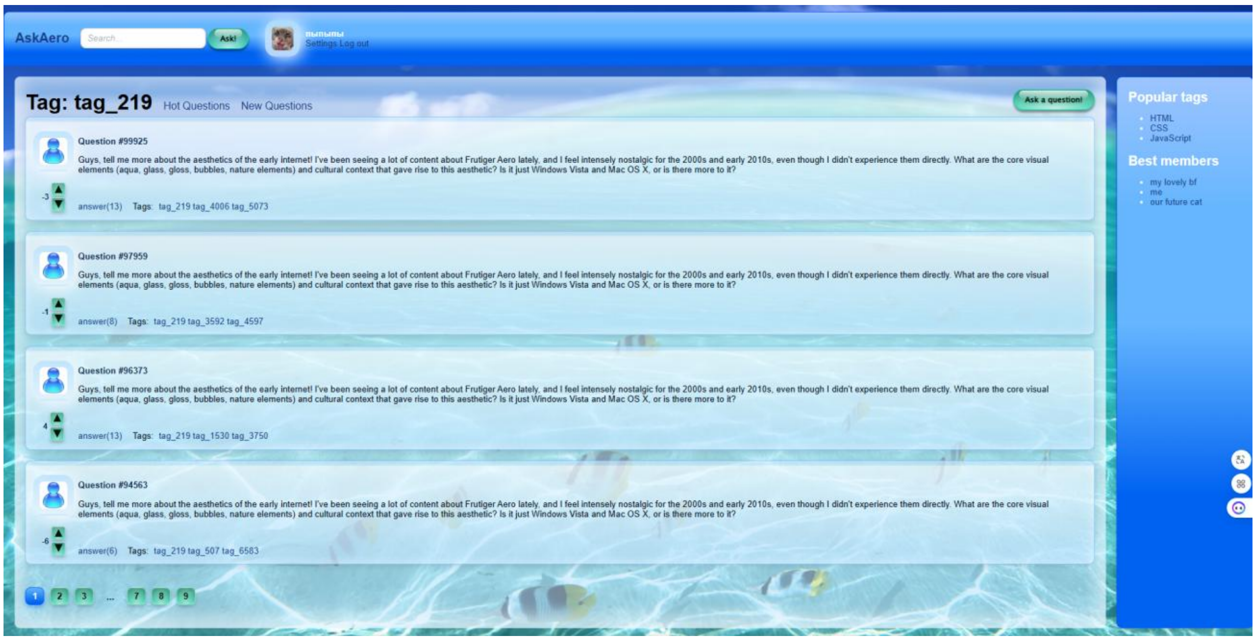
Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

18 answer(8) Tags: tag_394 tag_2750 tag_4944

Popular tags
- HTML
- CSS
- JavaScript

Best members
- my lovely bf
- me
- our future cat

Рис. 2. Страница вопросов отсортированных по рейтингу



Tag: tag_219 Hot Questions New Questions

Question #99925

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

-3 answer(13) Tags: tag_219 tag_4006 tag_5073

Question #97959

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

-1 answer(8) Tags: tag_219 tag_3592 tag_4597

Question #96373

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

4 answer(13) Tags: tag_219 tag_1530 tag_3750

Question #94563

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

-6 answer(6) Tags: tag_219 tag_507 tag_6583

Popular tags
- HTML
- CSS
- JavaScript

Best members
- my lovely bf
- me
- our future cat

Рис. 3. Страница вопросов, содержащих исходный тег

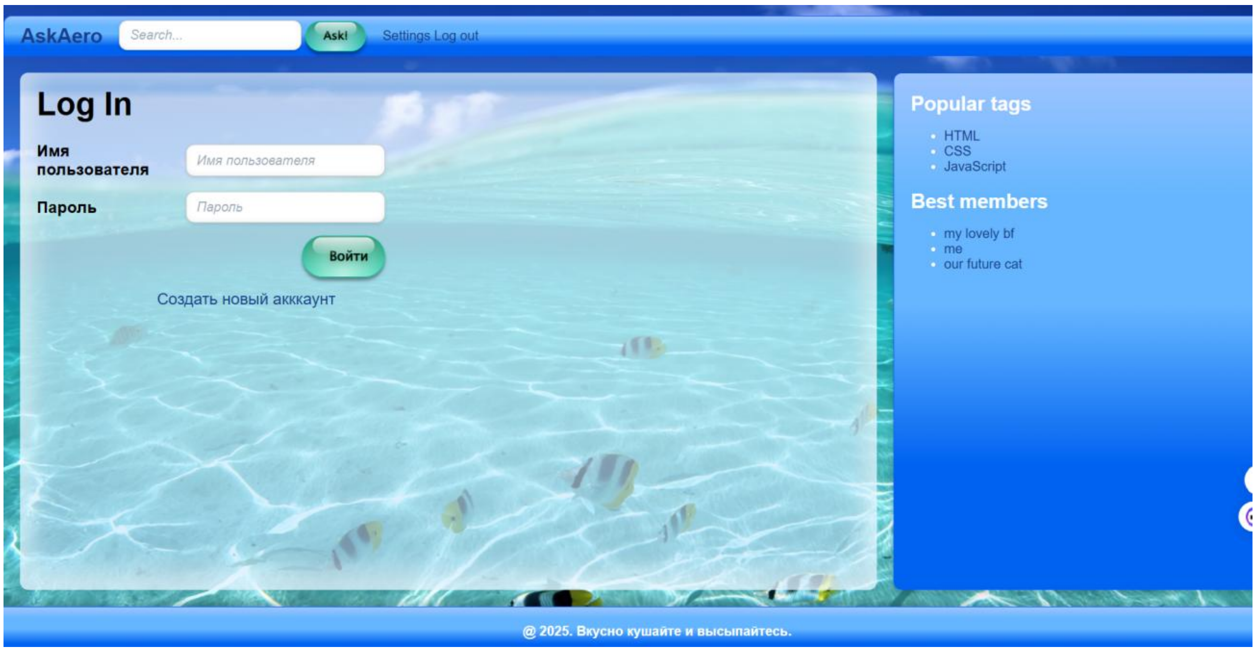


Рис. 4. Страница входа

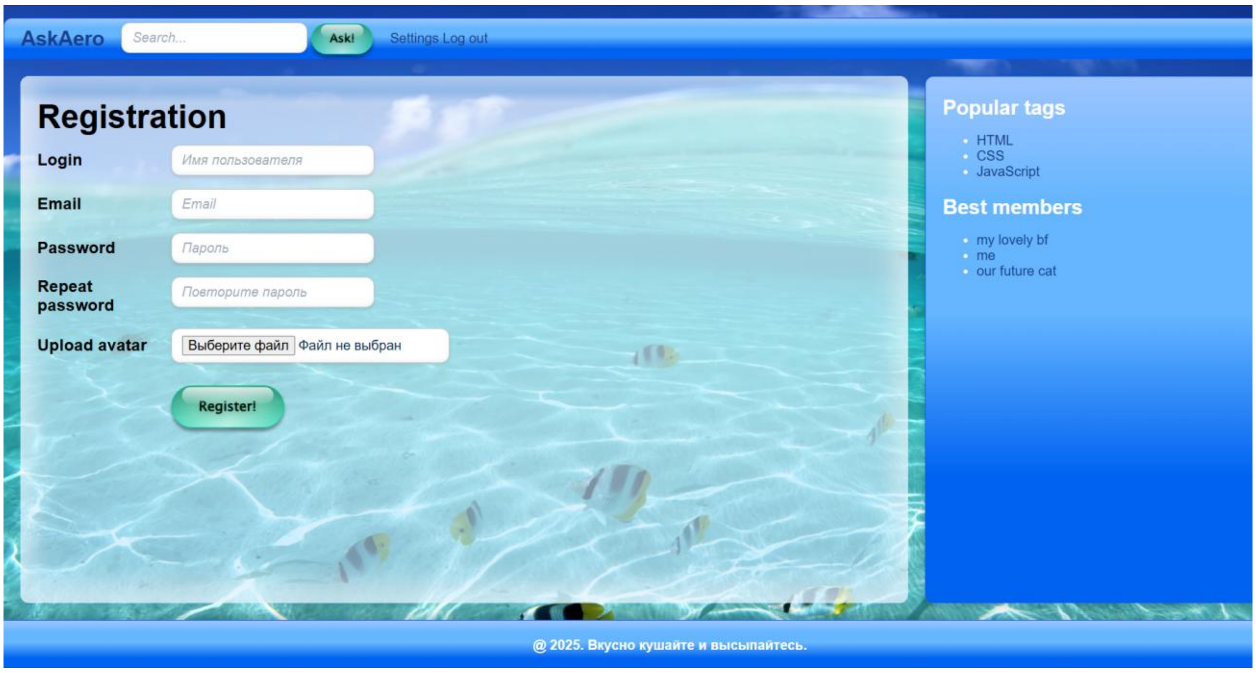
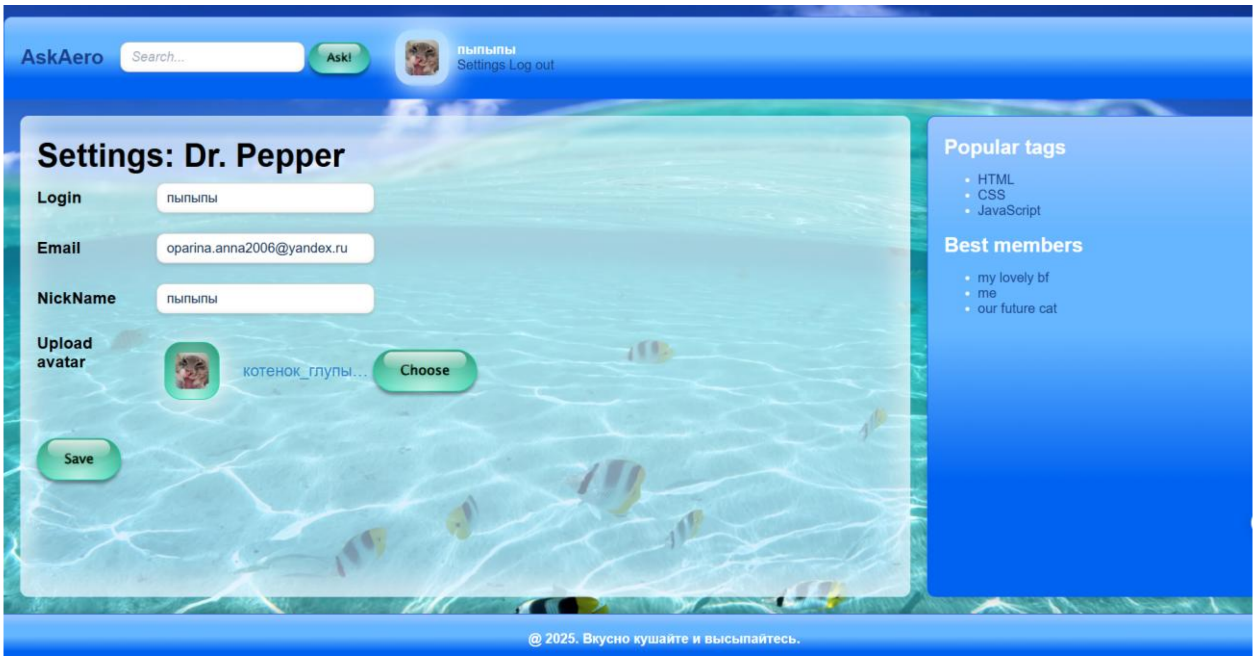


Рис. 5. Страница регистрации


AskAero

Search...

Ask!

пыпыпы

Settings Log out

Settings: Dr. Pepper

Login	пыпыпы
Email	oparina.anna2006@yandex.ru
NickName	пыпыпы
Upload avatar	котенок_глупы... Choose

Save

Popular tags

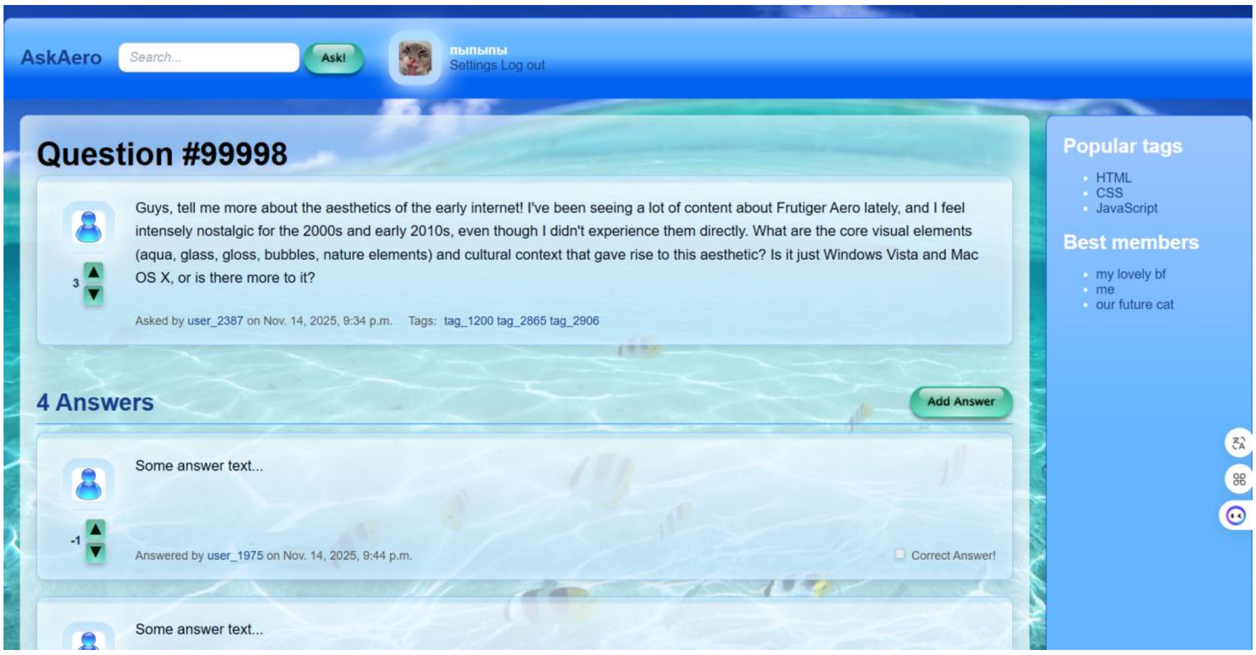
- HTML
- CSS
- JavaScript

Best members

- my lovely bf
- me
- our future cat

@ 2025. Вкусно кушайте и высыпайтесь.

Рис. 6. Страница настроек пользователя


AskAero

Search...

Ask!

пыпыпы

Settings Log out

Question #99998

Guys, tell me more about the aesthetics of the early internet! I've been seeing a lot of content about Frutiger Aero lately, and I feel intensely nostalgic for the 2000s and early 2010s, even though I didn't experience them directly. What are the core visual elements (aqua, glass, gloss, bubbles, nature elements) and cultural context that gave rise to this aesthetic? Is it just Windows Vista and Mac OS X, or is there more to it?

3

Asked by user_2387 on Nov. 14, 2025, 9:34 p.m. Tags: tag_1200 tag_2865 tag_2906

4 Answers

Add Answer

Some answer text...

-1

Answered by user_1975 on Nov. 14, 2025, 9:44 p.m. Correct Answer!

Some answer text...

Popular tags

- HTML
- CSS
- JavaScript

Best members

- my lovely bf
- me
- our future cat

Рис. 7. Страница вопроса