Требования к отчету

Общие требования:

1) Шрифт – PT Astra Serif, 14 пт.;
2) Интервалы: междустрочный – 1,5 строки, интервал до и после абзаца – 0 пт.;
3) Отступ первой строки – 1,25;
4) Рисунки и подписи к ним выравниваются по центру;
5) Основной текст выравнивается по ширине.

Требования к структуре отчета:

1) Титульный лист;
2) Содержание;
3) Основная часть:
    1.1. Отчет по выполнению работы;
    1.2. Ссылка на открытый git репозиторий.
4) Выводы по работе.

ОГБПОУ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Отчет по практической работе №1

УП.05 Разработка кода информационных систем

Тема: «Реализация тестового проекта LocalLibrary»

Выполнил:

студент 432 группы

Левченко П.Л.

Проверил:

преподаватель, высшей к.к.

Фунтиков М.Н.

г. Томск – 2025 г

Содержание

1. Листинг кода

1.1 Шаблоны в базовой директории

Шаблон регистации

```
{% extends "base_generic.html" %}

{% block content %}

{% if form.errors %}
<p>Your username and password didn't match. Please try again.</p>
{% endif %}

{% if next %}
    {% if user.is_authenticated %}
    <p>Your account doesn't have access to this page. To proceed,
    please login with an account that has access.</p>
    {% else %}
    <p>Please login to see this page.</p>
    {% endif %}
{% endif %}

<form method="post" action="{% url 'login' %}">
{% csrf_token %}
<table>
<tr>
    <td>{{ form.username.label_tag }}</td>
    <td>{{ form.username }}</td>
</tr>
<tr>
```

```
    <td>{{ form.password.label_tag }}</td>
    <td>{{ form.password }}</td>
</tr>
</table>

<input type="submit" value="login">
<input type="hidden" name="next" value="{{ next }}">
</form>

{# Assumes you setup the password_reset view in your URLconf #}
<p><a href="{% url 'password_reset' %}">Lost password?</a></p>

{% endblock %}
```

Шаблон Выхода

```
{% extends "base_generic.html" %}

{% block content %}
<p>Logged out!</p>

<a href="{% url 'login'%}">Click here to login again.</a>
{% endblock %}
```

Шаблон book_form

```
{% extends "base_generic.html" %}

{% block content %}
  <form action="" method="post">
      {% csrf_token %}
```

```
        <table>
        {{ form.as_table }}
        </table>
        <input type="submit" value="Submit" />
      </form>
    {% endblock %}
```

Шаблон author_list

```
    {% extends "base_generic.html" %}

    {% block content %}

    <h1>Author List</h1>

    {% if author_list %}
      <ul>

      {% for author in author_list %}
        <li>
          <a href="{{ author.get_absolute_url }}">
          {{ author }} ({{author.date_of_birth}} - {% if author.date_of_death
%}{{author.date_of_death}}{% endif %})
          </a>
        </li>
      {% endfor %}

     </ul>
    {% else %}
      <p>There are no authors available.</p>
```

{% endif %}

{% endblock %}

Шаблон смены пароля email

Someone asked for password reset for email {{ email }}. Follow the link below:

{{ protocol}}://{{ domain }}{% url 'password_reset_confirm' uidb64=uid token=token %}

1.2 Шаблоны в приложении catalog

Базовый шаблон

```
<!doctype html>
<html lang="en">
  <head>
    {% block title %}<title>Local Library</title>{% endblock %}
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <link
     rel="stylesheet"
     href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
min.css" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script
>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></sc
ript>

    <!-- Добавление дополнительного статического CSS файла -->
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/styles.css' %}" />
  </head>

  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-2">
```

```
                {% block sidebar %}
                <ul class="sidebar-nav">
                  <li><a href="{% url 'index' %}">Home</a></li>
                  <li><a href="{% url 'books' %}">All books</a></li>
                  <li><a href="{% url 'authors' %}">All authors</a></li>
                  <li><a href="{% url 'my-borrowed' %}">My
Borrowed</a></li>
                  {% if perms.catalog.can_mark_returned %}
                    <li><a href="{% url 'all-borrowed' %}">All borrowed
books</a></li>
                    <li><a href="{% url 'author_create' %}">Create
Authors</a></li>
                   <li><a href="{% url 'book_create' %}">Create books</a></li>
                  {% endif %}
                  {% if user.is_authenticated %}
                    <li>User: {{ user.get_username }}</li>
                      <li>
                      <form method="post" action="{% url 'logout' %}?next={{
request.path }}">
                          {% csrf_token %}
                          <input type="submit" value="Logout">
                      </form>
                    </li>
                  {% else %}
                    <li><a href="{% url 'login' %}?next={{ request.path
}}">Login</a></li>
                  {% endif %}
                  </ul>
                </ul>
```

```
            {% endblock %}
        </div>
        <div class="col-sm-10 ">
            {% block content %}{% endblock %}
            <p>
             You have visited this page {{ num_visits }}{% if num_visits ==
1 %} time{%
             else %} times{% endif %}.
            </p>
        {% block pagination %}
        {% if is_paginated %}
            <div class="pagination">
            <span class="page-links">
                {% if page_obj.has_previous %}
                <a href="{{ request.path }}?page={{
page_obj.previous_page_number }}">previous</a>
                {% endif %}
                <span class="page-current">
                Page {{ page_obj.number }} of {{
page_obj.paginator.num_pages }}.
                </span>
                {% if page_obj.has_next %}
                <a href="{{ request.path }}?page={{
page_obj.next_page_number }}">next</a>
                {% endif %}
            </span>
            </div>
        {% endif %}
        {% endblock %}
```

```
      </div>
    </div>

  </div>
 </body>
</html>
```

Шаблон index.html

```
{% extends "base_generic.html" %}
{% block title %}<title>Home</title>{% endblock %}
{% block content %}
<h1>Local Library Home</h1>

 <p>Welcome to <em>LocalLibrary</em>, a very basic Django website developed as a tutorial example on the Mozilla Developer Network.</p>

<h2>Dynamic content</h2>

 <p>The library has the following record counts:</p>
 <ul>
  <li><strong>Books:</strong> {{ num_books }}</li>
  <li><strong>Copies:</strong> {{ num_instances }}</li>
  <li><strong>Copies available:</strong> {{ num_instances_available }}</li>
  <li><strong>Authors:</strong> {{ num_authors }}</li>
 </ul>
 <p><strong>Жанров в базе:</strong> {{ num_genres }}</p>
```

```
  <p><strong>Книг со словом "окак" в названии:</strong> {{
num_books_with_word }}</p>


{% endblock %}
```

1.3 Файлы проекта

settings.py

```
"""
Django settings for locallibrary project.

Generated by 'django-admin startproject' using Django 5.2.7.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.2/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-ip-mu$#v=9bd1!x+(k@ny=eemjf-(^syxkgu@omd57@p(p$38r'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'catalog.apps.CatalogConfig',

]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

```
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'locallibrary.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'locallibrary.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
```

```
    }
}

# Password validation
# https://docs.djangoproject.com/en/5.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/5.2/topics/i18n/
```

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Moscow'

USE_I18N = True

USE_TZ = True

STATIC_URL = 'static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = '/'
```

urls.py

```
"""
URL configuration for locallibrary project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.2/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
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import RedirectView
from django.urls import include
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('catalog/', include('catalog.urls')),
    path( '', RedirectView.as_view(url='/catalog/', permanent=True)),
    path('accounts/', include('django.contrib.auth.urls')),

]

urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
```

1.4 Файлы главного приложения catalog

Admin.py

```
from django.contrib import admin
```

```
from .models import Author, Genre, Book, BookInstance, Language

#admin.site.register(Book)
#admin.site.register(Author)
admin.site.register(Genre)
admin.site.register(Language)
#admin.site.register(BookInstance)

class BooksInstanceInline(admin.TabularInline):
    model = BookInstance

class BooksInline(admin.TabularInline):
    model = Book

@admin.register(Author)
class AuthorAdmin(admin.ModelAdmin):
    list_display = ('last_name', 'first_name', 'date_of_birth', 'date_of_death')
    fields = ['first_name', 'last_name', ('date_of_birth', 'date_of_death')]
    inlines = [BooksInline]

@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    list_display = ('title', 'author', 'display_genre')
    inlines = [BooksInstanceInline]

@admin.register(BookInstance)
class BookInstanceAdmin(admin.ModelAdmin):
    list_display = ('book', 'status', 'borrower', 'due_back', 'id')
```

```
    fieldsets = (
        (None, {
            'fields': ('book','imprint', 'id')
        }),
        ('Availability', {
            'fields': ('status', 'due_back', 'borrower')
        }),
    )
```

Forms.py

```
from django import forms
from django.core.exceptions import ValidationError
from django.utils.translation import gettext_lazy as _
import datetime

class RenewBookForm(forms.Form):
    renewal_date = forms.DateField(
        help_text="Enter a date between now and 4 weeks (default 3)."
    )

    def clean_renewal_date(self):
        data = self.cleaned_data['renewal_date']

        if data < datetime.date.today():
            raise ValidationError(_('Invalid date - renewal in past'))

        if data > datetime.date.today() + datetime.timedelta(weeks=4):
```

```
            raise ValidationError(_('Invalid date - renewal more than 4 weeks ahead'))

        return data
```

Models.py

```
from django.db import models
from django.urls import reverse

from django.db.models import UniqueConstraint
from django.db.models.functions import Lower

from django.contrib.auth.models import User

class Genre(models.Model):
    """
    Model representing a book genre (e.g. Science Fiction, Non Fiction).
    """
    name = models.CharField(max_length=200, help_text="Enter a book genre (e.g. Science Fiction, French Poetry etc.)")

    def __str__(self):
        """
        String for representing the Model object (in Admin site etc.)
        """
        return self.name

class Language(models.Model):
```

```
        name = models.CharField(max_length=200, help_text="Enter the
language of the book")

        def get_absolute_url(self):
            return reverse('language-detail', args=[str(self.id)])

        def __str__(self):
            return self.name

        class Meta:
            constraints = [
                UniqueConstraint(
                    Lower('name'),
                    name='language_name_case_insensitive_unique',
                    violation_error_message = "Language already exists"
                ),
            ]


    class Book(models.Model):
        """
        Model representing a book (but not a specific copy of a book).
        """
        title = models.CharField(max_length=200)
        author = models.ForeignKey('Author', on_delete=models.SET_NULL,
null=True)
        summary = models.TextField(max_length=1000, help_text="Enter a
brief description of the book")
```

```
        isbn = models.CharField('ISBN',max_length=13, help_text='13
Character <a href="https://www.isbn-international.org/content/what-
isbn">ISBN number</a>')
        genre = models.ManyToManyField(Genre, help_text="Select a genre
for this book")
        language = models.ForeignKey('Language',
on_delete=models.SET_NULL, null=True)

        def __str__(self):
            """
            String for representing the Model object.
            """
            return self.title


        def get_absolute_url(self):
            """
            Returns the url to access a particular book instance.
            """
            return reverse('book-detail', args=[str(self.id)])
        def display_genre(self):
            return ', '.join([ genre.name for genre in self.genre.all()[:3] ])


        display_genre.short_description = 'Genre'


    import uuid
    from datetime import date

    from django.conf import settings
```

```
class BookInstance(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, help_text="Unique ID for this particular book across whole library")
    book = models.ForeignKey('Book', on_delete=models.SET_NULL, null=True)
    imprint = models.CharField(max_length=200)
    due_back = models.DateField(null=True, blank=True)
    borrower = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, blank=True)


    LOAN_STATUS = (
        ('m', 'Maintenance'),
        ('o', 'On loan'),
        ('a', 'Available'),
        ('r', 'Reserved'),
    )

    status = models.CharField(max_length=1, choices=LOAN_STATUS, blank=True, default='m', help_text='Book availability')

    class Meta:
        ordering = ["due_back"]
        permissions = (
            ("can_mark_returned", "Set book as returned"),
        )
```

```
    def __str__(self):
        return f"{self.id}, {self.book.title}"

    @property
    def is_overdue(self):
        if self.due_back and date.today() > self.due_back:
            return True
        return False

class Author(models.Model):
    """ Model representing an author."""
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    date_of_birth = models.DateField(null=True, blank=True)
    date_of_death = models.DateField('Died', null=True, blank=True)

    def get_absolute_url(self):
        return reverse('author-detail', args=[str(self.id)])

    def __str__(self):
        """
        String for representing the Model object.
        """
        return f"{self.last_name}, {self.first_name}"
```

Urls.py

```
from django.urls import path
from . import views
```

```
    urlpatterns = [
        path('', views.index, name='index'),
        path('books/', views.BookListView.as_view(), name='books'),
        path('book/<int:pk>', views.BookDetailView.as_view(), name='book-
detail'),
        path('authors/', views.AuthorListView.as_view(), name='authors'),
        path('author/<int:pk>', views.AuthorDetailView.as_view(),
name='author-detail'),
        path('author/<int:pk>/update', views.AuthorUpdateView.as_view(),
name='author-update'),
        path('mybooks/', views.LoanedBooksByUserListView.as_view(),
name='my-borrowed'),
        path('borrowed/', views.AllBorrowedBooksListView.as_view(),
name='all-borrowed'),
        path('book/<uuid:pk>/renew/', views.renew_book_librarian,
name='renew-book-librarian'),
        path('author/create/', views.AuthorCreate.as_view(),
name='author_create'),
        path('author/<int:pk>/update/', views.AuthorUpdate.as_view(),
name='author_update'),
        path('author/<int:pk>/delete/', views.AuthorDelete.as_view(),
name='author_delete'),
        path('book/create/', views.BookCreate.as_view(), name='book_create'),
        path('book/<int:pk>/update/', views.BookUpdate.as_view(),
name='book_update'),
        path('book/<int:pk>/delete/', views.BookDelete.as_view(),
name='book_delete'),
    ]
```

Views.py

```
from django.shortcuts import render
from .models import Book, Author, BookInstance, Genre, Language

from django.contrib.auth.mixins import LoginRequiredMixin

def index(request):

    num_books = Book.objects.all().count()
    num_instances = BookInstance.objects.all().count()
    num_instances_available =
BookInstance.objects.filter(status__exact='a').count()
    num_authors = Author.objects.count()
    num_genres = Genre.objects.count()
    search_word = 'окак'
    num_visits=request.session.get('num_visits', 0)
    request.session['num_visits'] = num_visits+1

    num_books_with_word =
Book.objects.filter(title__icontains=search_word).count()
    num_visits += 1
    request.session['num_visits'] = num_visits

    return render(
        request,
        'index.html',
        context={'num_books': num_books, 'num_instances':
num_instances,
```

```
                'num_instances_available': num_instances_available,
'num_authors': num_authors,
                'num_genres': num_genres,
                'num_books_with_word': num_books_with_word,
                'num_visits': num_visits},
            )

        from django.views import generic

        class BookDetailView(generic.DetailView):
            model = Book

        class BookListView(generic.ListView):
            model = Book
            paginate_by = 2

        class AuthorListView(generic.ListView):
            model = Author

        class AuthorDetailView(generic.DetailView):
            model = Author

        class AuthorUpdateView(generic.UpdateView):
            model = Author


        class
LoanedBooksByUserListView(LoginRequiredMixin,generic.ListView):
            model = BookInstance
```

```
        template_name ='catalog/bookinstance_list_borrowed_user.html'
        paginate_by = 2

        def get_queryset(self):
            return
BookInstance.objects.filter(borrower=self.request.user).filter(status__exact='o').
order_by('due_back')

    from django.contrib.auth.mixins import PermissionRequiredMixin

    class AllBorrowedBooksListView(PermissionRequiredMixin,
generic.ListView):
        model = BookInstance
        template_name = 'catalog/bookinstance_list_borrowed_all.html'
        permission_required = 'catalog.can_mark_returned'
        paginate_by = 10

        def get_queryset(self):
            return
BookInstance.objects.filter(status__exact='o').order_by('due_back')


    from django.contrib.auth.decorators import permission_required

    from django.shortcuts import get_object_or_404
    from django.http import HttpResponseRedirect
    from django.urls import reverse
    import datetime
```

```
from .forms import RenewBookForm

@permission_required('catalog.can_mark_returned')
def renew_book_librarian(request, pk):
    """
    View function for renewing a specific BookInstance by librarian
    """
    book_inst = get_object_or_404(BookInstance, pk=pk)

    # If this is a POST request then process the Form data
    if request.method == 'POST':

        # Create a form instance and populate it with data from the request (binding):
        form = RenewBookForm(request.POST)

        # Check if the form is valid:
        if form.is_valid():
            # process the data in form.cleaned_data as required (here we just write it to the model due_back field)
            book_inst.due_back = form.cleaned_data['renewal_date']
            book_inst.save()

            # redirect to a new URL:
            return HttpResponseRedirect(reverse('all-borrowed') )

    # If this is a GET (or any other method) create the default form.
    else:
```

```
            proposed_renewal_date = datetime.date.today() +
datetime.timedelta(weeks=3)
            form = RenewBookForm(initial={'renewal_date':
proposed_renewal_date,})

        return render(request, 'catalog/book_renew_librarian.html', {'form':
form, 'bookinst':book_inst})

    from django.views.generic.edit import CreateView, UpdateView,
DeleteView
    from django.urls import reverse_lazy
    from .models import Author
    from .models import Book

    class AuthorCreate(CreateView):
        model = Author
        fields = '__all__'
        initial={'date_of_death':'12/10/2016',}

    class AuthorUpdate(UpdateView):
        model = Author
        fields = ['first_name','last_name','date_of_birth','date_of_death']

    class AuthorDelete(DeleteView):
        model = Author
        success_url = reverse_lazy('authors')

    class BookCreate(CreateView):
        model = Book
```

```
    fields = '__all__'

class BookUpdate(UpdateView):
    model = Book

class BookDelete(DeleteView):
    model = Book
    success_url = reverse_lazy('books')
```

1.5 Пользователи

Login: Staff

Password: QWEasd2008

Login: kasper

Password: kasper_2008

Login: TestUser

Password: Kasper_2008

Ссылка на github: https://github.com/kasper-bl/practice