# Список файлов проекта

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\docker-compose.yml

services:  
 web:  
 build: .  
 ports:  
 - "9502:9502"  
 volumes:  
 - .:/app  
 environment:  
 - FLASK\_ENV=development  
 - SECRET\_KEY=a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6  
 - FLASK\_RUN\_PORT=9502

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\Dockerfile

FROM python:3.9-slim  
WORKDIR /app  
COPY requirements.txt .  
RUN pip install --no-cache-dir -r requirements.txt  
COPY . .  
CMD ["python", "main.py"]

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\main.py

from flask import Flask  
from pyngrok import ngrok  
import os  
from flask\_sqlalchemy import SQLAlchemy  
from app.routes import register\_routes  
from datetime import datetime, timedelta  
  
app = Flask(\_\_name\_\_, template\_folder='templates') # Явно указываем путь к templates относительно app/  
  
app.config['SECRET\_KEY'] = os.getenv('SECRET\_KEY', 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6')  
base\_dir = os.path.abspath(os.path.dirname(\_\_file\_\_))  
instance\_path = os.path.join(base\_dir, 'instance')  
os.makedirs(instance\_path, exist\_ok=True)  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + os.path.join(instance\_path, 'medical\_db.sqlite')  
app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  
  
db = SQLAlchemy(app)  
  
# Определяем модели  
PREFLIGHT\_CONDITIONS = ['Допущен', 'Отстранен']  
EXAM\_TYPES = ['ВЛК', 'КМО', 'УМО', 'КМО2']  
  
class Employee(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 fio = db.Column(db.String(100), nullable=False, unique=True)  
 birth\_date = db.Column(db.Date, nullable=False)  
 position = db.Column(db.String(100), nullable=False)  
 order\_no = db.Column(db.String(50), nullable=False)  
 preflight\_condition = db.Column(db.String(100), default='Допущен')  
 note = db.Column(db.String(200))  
 examinations = db.relationship('Examination', backref='employee', lazy=True, cascade='all, delete')  
  
class Examination(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 employee\_id = db.Column(db.Integer, db.ForeignKey('employee.id'), nullable=False)  
 exam\_type = db.Column(db.String(50), nullable=False)  
 exam\_date = db.Column(db.Date, nullable=False)  
 diagnosis = db.Column(db.String(200))  
  
register\_routes(app)  
  
with app.app\_context():  
 db.create\_all()  
  
port = int(os.getenv('FLASK\_RUN\_PORT', 5000)) # По умолчанию порт 5000  
if port == 5000 and os.getenv('USE\_NGROK', '0') == '1': # Используем ngrok только если явно указано  
 public\_url = ngrok.connect(port).public\_url  
 print(f"Ngrok tunnel: {public\_url}")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True, host='0.0.0.0', port=port)

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\ngrok.yml

version: "3"  
agent:  
 authtoken: 2uQzRigttgjHqKzLtfrRlLn9yAs\_6Y2yEAK2h7kESLXANCasR

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\export\_to\_word.py

from docx import Document  
from docx.oxml.ns import qn  
from docx.enum.style import WD\_STYLE\_TYPE  
from docx.shared import Pt  
import os  
from datetime import datetime  
  
# Путь к папке с вашими файлами  
folder\_path = "C:/Users/Nurba/PycharmProjects/militarybase2" # на работе папка  
# folder\_path = "C:/Users/User/PycharmProjects/militarybase2" # домашняя папка  
  
# Получаем текущую дату в формате дд.мм  
current\_date = datetime.now().strftime("%d.%m")  
output\_file = f"all\_project\_files\_{current\_date}.docx" # Имя выходного файла с датой  
  
# Список допустимых расширений файлов  
ALLOWED\_EXTENSIONS = {'.html', '.css', '.py', '.yaml', '.yml'}  
ALLOWED\_FILENAMES = {'Dockerfile'} # Точное имя файла без расширения  
  
# Создаём новый документ Word  
doc = Document()  
  
# Добавляем стиль "Code", если его нет  
styles = doc.styles  
if "Code" not in styles:  
 style = styles.add\_style("Code", WD\_STYLE\_TYPE.PARAGRAPH)  
 style.font.name = "Courier New" # Моноширинный шрифт для кода  
 style.font.size = Pt(10) # Размер шрифта 10pt  
 style.paragraph\_format.space\_after = Pt(6) # Отступ после абзаца 6pt  
  
# Добавляем заголовок документа  
doc.add\_heading("Список файлов проекта", level=1)  
  
# Проходим по всем файлам в папке и её подпапках  
for root, dirs, files in os.walk(folder\_path):  
 # Пропускаем папки \_\_pycache\_\_ и venv  
 if "\_\_pycache\_\_" in dirs:  
 dirs.remove("\_\_pycache\_\_")  
 if "venv" in dirs:  
 dirs.remove("venv")  
  
 for file in files:  
 # Проверяем расширение файла или точное имя  
 file\_extension = os.path.splitext(file)[1].lower() # Получаем расширение файла  
 file\_name = file # Полное имя файла  
  
 if file\_extension in ALLOWED\_EXTENSIONS or file\_name in ALLOWED\_FILENAMES:  
 file\_path = os.path.join(root, file)  
 doc.add\_heading(f"Файл: {file\_path}", level=2)  
  
 # Читаем содержимое файла  
 try:  
 with open(file\_path, "r", encoding="utf-8") as f:  
 content = f.read()  
 doc.add\_paragraph(content, style="Code") # Добавляем содержимое с форматированием  
 except Exception as e:  
 doc.add\_paragraph(f"Ошибка чтения файла {file\_path}: {e}", style="Normal")  
  
 doc.add\_page\_break() # Разделяем файлы новой страницей  
  
# Сохраняем документ  
doc.save(output\_file)  
print(f"Файлы сохранены в {output\_file}")

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\migrate.py

import sqlite3  
import os  
  
# Путь к базе данных в папке instance относительно корня проекта  
base\_dir = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..'))  
db\_path = os.path.join(base\_dir, 'instance', 'medical\_db.sqlite')  
  
# Подключаемся к базе данных  
conn = sqlite3.connect(db\_path)  
cursor = conn.cursor()  
  
# Добавляем столбец note  
cursor.execute("ALTER TABLE employee ADD COLUMN note VARCHAR(200)")  
  
# Сохраняем изменения  
conn.commit()  
conn.close()  
  
print("Столбец 'note' успешно добавлен в таблицу 'employee'!")

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\models.py

from datetime import datetime, timedelta  
  
def process\_employee\_form(form):  
 from main import PREFLIGHT\_CONDITIONS, EXAM\_TYPES  
 preflight = form['preflight\_condition'] if form['preflight\_condition'] in PREFLIGHT\_CONDITIONS else 'Допущен'  
 employee\_data = {  
 'fio': form['fio'],  
 'birth\_date': datetime.strptime(form['birth\_date'], '%Y-%m-%d') if form['birth\_date'] else None,  
 'position': form['position'],  
 'order\_no': form['order\_no'],  
 'preflight\_condition': preflight,  
 'note': form['note'] if form['note'] else None  
 }  
 examinations = []  
 for exam\_type in EXAM\_TYPES:  
 date\_key = f"{exam\_type.lower()}\_date"  
 diag\_key = f"{exam\_type.lower()}\_diagnosis"  
 if form.get(date\_key):  
 try:  
 exam\_date = datetime.strptime(form[date\_key], '%Y-%m-%d')  
 examinations.append({  
 'exam\_type': exam\_type,  
 'exam\_date': exam\_date,  
 'diagnosis': form.get(diag\_key)  
 })  
 except ValueError:  
 print(f"Ошибка формата даты для {exam\_type}: {form[date\_key]}")  
 return employee\_data, examinations  
  
def calculate\_expiry(employee):  
 from main import EXAM\_TYPES  
 latest\_exam\_dates = {exam\_type: None for exam\_type in EXAM\_TYPES}  
 expiry\_dates = {exam\_type: None for exam\_type in EXAM\_TYPES}  
 days\_left = {exam\_type: float('inf') for exam\_type in EXAM\_TYPES}  
 nearest\_exam = None  
 min\_days\_left = float('inf')  
  
 # Находим последнюю дату для каждого типа осмотра  
 for exam in employee.examinations:  
 if exam.exam\_type in EXAM\_TYPES:  
 if not latest\_exam\_dates[exam.exam\_type] or exam.exam\_date > latest\_exam\_dates[exam.exam\_type]:  
 latest\_exam\_dates[exam.exam\_type] = exam.exam\_date  
  
 current\_date = datetime.now().date()  
 vlk\_date = latest\_exam\_dates['ВЛК']  
  
 # Базовые сроки от ВЛК, если ВЛК есть  
 if vlk\_date:  
 expiry\_dates['ВЛК'] = vlk\_date + timedelta(days=365) # 365 дней от ВЛК  
 expiry\_dates['КМО'] = vlk\_date + timedelta(days=90) # 90 дней от ВЛК  
 expiry\_dates['УМО'] = vlk\_date + timedelta(days=180) # 180 дней от ВЛК  
 expiry\_dates['КМО2'] = vlk\_date + timedelta(days=270) # 270 дней от ВЛК  
  
 # Переопределяем сроки, если есть более поздние даты осмотров  
 for exam\_type in EXAM\_TYPES:  
 if latest\_exam\_dates[exam\_type] and (not vlk\_date or latest\_exam\_dates[exam\_type] > vlk\_date):  
 expiry\_dates[exam\_type] = latest\_exam\_dates[exam\_type] + timedelta(days=365) # 365 дней от последнего осмотра  
  
 # Рассчитываем дни до окончания для каждого типа осмотра  
 for exam\_type in EXAM\_TYPES:  
 if expiry\_dates[exam\_type]:  
 days\_left[exam\_type] = (expiry\_dates[exam\_type] - current\_date).days  
 if days\_left[exam\_type] < min\_days\_left:  
 min\_days\_left = days\_left[exam\_type]  
 nearest\_exam = exam\_type  
  
 # Обновляем состояние сотрудника  
 if min\_days\_left < 0 or not employee.examinations:  
 employee.preflight\_condition = 'Отстранен'  
 else:  
 employee.preflight\_condition = 'Допущен'  
  
 return {  
 'employee': employee,  
 'vlk\_expiry': expiry\_dates['ВЛК'],  
 'kmo\_expiry': expiry\_dates['КМО'],  
 'umo\_expiry': expiry\_dates['УМО'],  
 'kmo2\_expiry': expiry\_dates['КМО2'],  
 'vlk\_days\_left': days\_left['ВЛК'] if days\_left['ВЛК'] != float('inf') else None,  
 'kmo\_days\_left': days\_left['КМО'] if days\_left['КМО'] != float('inf') else None,  
 'umo\_days\_left': days\_left['УМО'] if days\_left['УМО'] != float('inf') else None,  
 'kmo2\_days\_left': days\_left['КМО2'] if days\_left['КМО2'] != float('inf') else None,  
 'min\_days\_left': min\_days\_left if min\_days\_left != float('inf') else None,  
 'nearest\_exam': nearest\_exam  
 }  
  
def recalculate\_all\_employees(db\_session):  
 """Пересчитывает сроки для всех сотрудников в базе."""  
 from main import Employee  
 employees = Employee.query.all()  
 for emp in employees:  
 db\_session.refresh(emp) # Синхронизируем объект с базой  
 expiry\_data = calculate\_expiry(emp)  
 emp.preflight\_condition = expiry\_data['employee'].preflight\_condition  
 db\_session.commit()

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\process\_employee\_form.py

from datetime import datetime  
def process\_employee\_form(form):  
 return {  
 'fio': form['fio'],  
 'birth\_date': datetime.strptime(form['birth\_date'], '%Y-%m-%d'),  
 'position': form['position'],  
 'order\_no': form['order\_no'],  
 'vlk\_date': datetime.strptime(form['vlk\_date'], '%Y-%m-%d') if form['vlk\_date'] else None,  
 'vlk\_diagnosis': form['vlk\_diagnosis'],  
 'kmo\_date': datetime.strptime(form['kmo\_date'], '%Y-%m-%d') if form['kmo\_date'] else None,  
 'kmo\_diagnosis': form['kmo\_diagnosis'],  
 'umo\_date': datetime.strptime(form['umo\_date'], '%Y-%m-%d') if form['umo\_date'] else None,  
 'umo\_diagnosis': form['umo\_diagnosis'],  
 'kmo2\_date': datetime.strptime(form['kmo2\_date'], '%Y-%m-%d') if form['kmo2\_date'] else None,  
 'kmo2\_diagnosis': form['kmo2\_diagnosis'],  
 'preflight\_condition': form['preflight\_condition']  
 }

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\\_\_init\_\_.py

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\routes\employee.py

from flask import render\_template, request, redirect, url\_for, flash  
from app.models import process\_employee\_form, recalculate\_all\_employees  
  
def add():  
 from main import db, Employee, Examination, PREFLIGHT\_CONDITIONS  
  
 if request.method == 'POST':  
 try:  
 employee\_data, examinations = process\_employee\_form(request.form)  
 if not all([employee\_data['fio'], employee\_data['birth\_date'], employee\_data['position'],  
 employee\_data['order\_no']]):  
 flash('Поля ФИО, дата рождения, должность и "По приказу № 721" обязательны!', 'danger')  
 return redirect(url\_for('add'))  
  
 existing\_employee = Employee.query.filter\_by(fio=employee\_data['fio']).first()  
 if existing\_employee:  
 flash(f'Сотрудник с ФИО "{employee\_data["fio"]}" уже существует!', 'danger')  
 return redirect(url\_for('add'))  
  
 employee = Employee(\*\*employee\_data)  
 db.session.add(employee)  
 db.session.flush()  
  
 for exam in examinations:  
 db.session.add(Examination(employee\_id=employee.id, \*\*exam))  
  
 recalculate\_all\_employees(db.session) # Пересчитываем всех сотрудников  
 flash('Сотрудник успешно добавлен!', 'success')  
 return redirect(url\_for('index'))  
 except Exception as e:  
 db.session.rollback()  
 flash(f"Ошибка добавления сотрудника: {str(e)}", 'danger')  
 return redirect(url\_for('add'))  
  
 return render\_template('add.html', preflight\_conditions=PREFLIGHT\_CONDITIONS)  
  
def edit(id):  
 from main import db, Employee, Examination, PREFLIGHT\_CONDITIONS  
  
 employee = Employee.query.get\_or\_404(id)  
 if request.method == 'POST':  
 try:  
 employee\_data, examinations = process\_employee\_form(request.form)  
 if not all([employee\_data['fio'], employee\_data['birth\_date'], employee\_data['position'],  
 employee\_data['order\_no']]):  
 flash('Поля ФИО, дата рождения, должность и "По приказу № 721" обязательны!', 'danger')  
 return redirect(url\_for('edit', id=id))  
  
 existing\_employee = Employee.query.filter(Employee.fio == employee\_data['fio'],  
 Employee.id != id).first()  
 if existing\_employee:  
 flash(f'Сотрудник с ФИО "{employee\_data["fio"]}" уже существует!', 'danger')  
 return redirect(url\_for('edit', id=id))  
  
 for key, value in employee\_data.items():  
 setattr(employee, key, value)  
 Examination.query.filter\_by(employee\_id=id).delete()  
 for exam in examinations:  
 db.session.add(Examination(employee\_id=id, \*\*exam))  
  
 recalculate\_all\_employees(db.session) # Пересчитываем всех сотрудников  
 flash('Данные сотрудника успешно обновлены!', 'success')  
 return redirect(url\_for('index'))  
 except Exception as e:  
 db.session.rollback()  
 flash(f"Ошибка обновления данных: {str(e)}", 'danger')  
 return redirect(url\_for('edit', id=id))  
  
 examinations = Examination.query.filter\_by(employee\_id=id).all()  
 return render\_template('edit.html', employee=employee, examinations=examinations,  
 preflight\_conditions=PREFLIGHT\_CONDITIONS)  
  
def history(id):  
 from main import Employee, Examination  
  
 employee = Employee.query.get\_or\_404(id)  
 examinations = Examination.query.filter\_by(employee\_id=id).order\_by(Examination.exam\_date.asc()).all()  
 return render\_template('history.html', employee=employee, examinations=examinations)  
  
def delete(id):  
 from main import db, Employee  
  
 employee = Employee.query.get\_or\_404(id)  
 try:  
 db.session.delete(employee)  
 recalculate\_all\_employees(db.session) # Пересчитываем всех сотрудников  
 flash('Сотрудник успешно удален!', 'success')  
 except Exception as e:  
 flash(f"Ошибка удаления сотрудника: {str(e)}", 'danger')  
 return redirect(url\_for('index'))

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\routes\excel.py

from flask import render\_template, request, redirect, url\_for, flash, send\_file  
from io import BytesIO  
from openpyxl import Workbook, load\_workbook  
from openpyxl.utils import get\_column\_letter  
import xlwt  
import xlrd  
from datetime import datetime  
  
def export\_excel\_xlsx():  
 from main import Employee # Импорт внутри функции  
  
 employees = Employee.query.all()  
 wb = Workbook()  
 ws = wb.active  
 ws.title = "Сотрудники"  
  
 headers = [  
 'ID', 'ФИО', 'Дата рождения', 'Должность', 'По приказу № 721', 'Состояние', 'Примечание',  
 'ВЛК дата', 'ВЛК диагноз', 'КМО дата', 'КМО диагноз', 'УМО дата', 'УМО диагноз',  
 'КМО2 дата', 'КМО2 диагноз'  
 ]  
 ws.append(headers)  
  
 for emp in employees:  
 vlk\_date = vlk\_diagnosis = kmo\_date = kmo\_diagnosis = umo\_date = umo\_diagnosis = kmo2\_date = kmo2\_diagnosis = ''  
 for exam in emp.examinations:  
 if exam.exam\_type == 'ВЛК':  
 vlk\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 vlk\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'КМО':  
 kmo\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 kmo\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'УМО':  
 umo\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 umo\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'КМО2':  
 kmo2\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 kmo2\_diagnosis = exam.diagnosis or ''  
  
 row = [  
 emp.id, emp.fio,  
 emp.birth\_date.strftime('%Y-%m-%d') if emp.birth\_date else '',  
 emp.position, emp.order\_no, emp.preflight\_condition, emp.note or '',  
 vlk\_date, vlk\_diagnosis, kmo\_date, kmo\_diagnosis, umo\_date, umo\_diagnosis,  
 kmo2\_date, kmo2\_diagnosis  
 ]  
 ws.append(row)  
  
 for col in range(1, len(headers) + 1):  
 ws.column\_dimensions[get\_column\_letter(col)].width = 15  
  
 output = BytesIO()  
 wb.save(output)  
 output.seek(0)  
  
 return send\_file(  
 output,  
 mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',  
 as\_attachment=True,  
 download\_name='employees.xlsx'  
 )  
  
def export\_excel\_xls():  
 from main import Employee # Импорт внутри функции  
  
 employees = Employee.query.all()  
 wb = xlwt.Workbook(encoding='utf-8')  
 ws = wb.add\_sheet('Сотрудники')  
  
 headers = [  
 'ID', 'ФИО', 'Дата рождения', 'Должность', 'По приказу № 721', 'Состояние', 'Примечание',  
 'ВЛК дата', 'ВЛК диагноз', 'КМО дата', 'КМО диагноз', 'УМО дата', 'УМО диагноз',  
 'КМО2 дата', 'КМО2 диагноз'  
 ]  
 for col, header in enumerate(headers):  
 ws.write(0, col, header)  
  
 row\_num = 1  
 for emp in employees:  
 vlk\_date = vlk\_diagnosis = kmo\_date = kmo\_diagnosis = umo\_date = umo\_diagnosis = kmo2\_date = kmo2\_diagnosis = ''  
 for exam in emp.examinations:  
 if exam.exam\_type == 'ВЛК':  
 vlk\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 vlk\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'КМО':  
 kmo\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 kmo\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'УМО':  
 umo\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 umo\_diagnosis = exam.diagnosis or ''  
 elif exam.exam\_type == 'КМО2':  
 kmo2\_date = exam.exam\_date.strftime('%Y-%m-%d')  
 kmo2\_diagnosis = exam.diagnosis or ''  
  
 row = [  
 emp.id, emp.fio,  
 emp.birth\_date.strftime('%Y-%m-%d') if emp.birth\_date else '',  
 emp.position, emp.order\_no, emp.preflight\_condition, emp.note or '',  
 vlk\_date, vlk\_diagnosis, kmo\_date, kmo\_diagnosis, umo\_date, umo\_diagnosis,  
 kmo2\_date, kmo2\_diagnosis  
 ]  
 for col, value in enumerate(row):  
 ws.write(row\_num, col, value)  
 row\_num += 1  
  
 output = BytesIO()  
 wb.save(output)  
 output.seek(0)  
  
 return send\_file(  
 output,  
 mimetype='application/vnd.ms-excel',  
 as\_attachment=True,  
 download\_name='employees.xls'  
 )  
  
def import\_excel():  
 from main import db, Employee, Examination, PREFLIGHT\_CONDITIONS  
 from app.models import recalculate\_all\_employees  
  
 if request.method == 'POST':  
 if 'file' not in request.files:  
 flash('Файл не выбран!', 'danger')  
 return redirect(url\_for('import\_excel'))  
 file = request.files['file']  
 if file.filename == '':  
 flash('Файл не выбран!', 'danger')  
 return redirect(url\_for('import\_excel'))  
 if file and (file.filename.endswith('.xlsx') or file.filename.endswith('.xls')):  
 try:  
 if file.filename.endswith('.xlsx'):  
 wb = load\_workbook(file)  
 ws = wb.active  
 rows = list(ws.rows)  
 headers = [cell.value for cell in rows[0]]  
 data\_rows = rows[1:]  
 else:  
 wb = xlrd.open\_workbook(file\_contents=file.read())  
 ws = wb.sheet\_by\_index(0)  
 headers = ws.row\_values(0)  
 data\_rows = [ws.row\_values(i) for i in range(1, ws.nrows)]  
  
 skipped\_count = 0  
 for row in data\_rows:  
 if file.filename.endswith('.xlsx'):  
 data = [cell.value if cell.value is not None else '' for cell in row]  
 else:  
 data = [str(cell) if cell else '' for cell in row]  
  
 fio = data[1]  
 existing\_employee = Employee.query.filter\_by(fio=fio).first()  
 if existing\_employee:  
 skipped\_count += 1  
 continue  
  
 birth\_date\_str = data[2]  
 if birth\_date\_str:  
 try:  
 birth\_date = datetime.strptime(str(birth\_date\_str), '%Y-%m-%d')  
 except ValueError:  
 flash(f"Неверный формат даты рождения для {fio}: {birth\_date\_str}", 'danger')  
 return redirect(url\_for('import\_excel'))  
 else:  
 birth\_date = None  
  
 employee = Employee(  
 fio=fio,  
 birth\_date=birth\_date,  
 position=data[3],  
 order\_no=data[4],  
 preflight\_condition=data[5] if data[5] in PREFLIGHT\_CONDITIONS else 'Допущен',  
 note=data[6] if data[6] else None  
 )  
 db.session.add(employee)  
 db.session.flush()  
  
 for i, exam\_type in enumerate(['ВЛК', 'КМО', 'УМО', 'КМО2']):  
 date\_idx = 7 + i \* 2  
 diag\_idx = 8 + i \* 2  
 exam\_date\_str = data[date\_idx]  
 if exam\_date\_str:  
 try:  
 exam\_date = datetime.strptime(str(exam\_date\_str), '%Y-%m-%d')  
 exam = Examination(  
 employee\_id=employee.id,  
 exam\_type=exam\_type,  
 exam\_date=exam\_date,  
 diagnosis=data[diag\_idx] if data[diag\_idx] else None  
 )  
 db.session.add(exam)  
 except ValueError:  
 flash(f"Неверный формат даты осмотра {exam\_type} для {fio}: {exam\_date\_str}",  
 'danger')  
 return redirect(url\_for('import\_excel'))  
  
 db.session.commit()  
 recalculate\_all\_employees(db.session) # Пересчитываем всех сотрудников  
 flash(f'Сотрудники успешно импортированы из Excel! Пропущено дубликатов: {skipped\_count}',  
 'success')  
 except Exception as e:  
 db.session.rollback()  
 flash(f"Ошибка импорта Excel: {str(e)}", 'danger')  
 return redirect(url\_for('index'))  
 return render\_template('import\_excel.html')

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\routes\index.py

from flask import render\_template, request  
from app.models import calculate\_expiry  
from datetime import datetime  
  
  
def index():  
 from main import db, Employee, PREFLIGHT\_CONDITIONS # Импорт внутри функции  
  
 page = request.args.get('page', 1, type=int)  
 per\_page = request.args.get('per\_page', 20, type=int)  
 sort = request.args.get('sort', 'deadline\_asc')  
 search = request.args.get('search', '')  
  
 query = Employee.query  
 if search:  
 query = query.filter(Employee.fio.ilike(f'%{search}%'))  
  
 if sort == 'fio\_asc':  
 query = query.order\_by(Employee.fio.asc())  
 elif sort == 'fio\_desc':  
 query = query.order\_by(Employee.fio.desc())  
 elif sort == 'suspended':  
 query = query.filter(Employee.preflight\_condition == 'Отстранен')  
 elif sort == 'deadline\_asc':  
 pass  
  
 pagination = query.paginate(page=page, per\_page=per\_page, error\_out=False)  
 employees = pagination.items  
 total\_pages = pagination.pages  
  
 employees\_with\_expiry = []  
 for emp in employees:  
 db.session.refresh(emp) # Синхронизируем объект с базой  
 expiry\_data = calculate\_expiry(emp) # Пересчитываем для каждого сотрудника  
 employees\_with\_expiry.append(expiry\_data)  
  
 # Сохраняем изменения в базе (например, preflight\_condition)  
 for emp in employees\_with\_expiry:  
 db.session.add(emp['employee'])  
 db.session.commit()  
  
 if sort == 'deadline\_asc':  
 employees\_with\_expiry.sort(  
 key=lambda x: x['min\_days\_left'] if x['min\_days\_left'] is not None else float('inf'),  
 reverse=False) # Сортировка по возрастанию, без reverse=True  
  
 return render\_template(  
 'index.html',  
 employees\_with\_expiry=employees\_with\_expiry,  
 page=page,  
 per\_page=per\_page,  
 total\_pages=total\_pages,  
 datetime=datetime,  
 preflight\_conditions=PREFLIGHT\_CONDITIONS  
 )

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\app\routes\\_\_init\_\_.py

from flask import Flask  
from app.routes.index import index  
from app.routes.employee import add, edit, history, delete  
from app.routes.excel import export\_excel\_xlsx, export\_excel\_xls, import\_excel  
  
def register\_routes(app: Flask):  
 app.route('/')(index)  
 app.route('/add', methods=['GET', 'POST'])(add)  
 app.route('/edit/<int:id>', methods=['GET', 'POST'])(edit)  
 app.route('/history/<int:id>')(history)  
 app.route('/delete/<int:id>')(delete)  
 app.route('/export\_excel\_xlsx')(export\_excel\_xlsx)  
 app.route('/export\_excel\_xls')(export\_excel\_xls)  
 app.route('/import\_excel', methods=['GET', 'POST'])(import\_excel)

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\static\style.css

/\* /static/style.css \*/  
body {  
 font-family: 'Roboto', sans-serif;  
 background-color: #f4f7fa;  
 margin: 0; /\* Убираем лишние отступы \*/  
}  
  
/\* Стили для шапки \*/  
.header {  
 background-color: #343a40;  
 color: white;  
 padding: 15px; /\* Уменьшаем padding для мобильных \*/  
 border-radius: 8px;  
 text-align: center;  
}  
  
/\* Таблица \*/  
.table th, .table td {  
 vertical-align: middle;  
 padding: 8px; /\* Уменьшаем padding для компактности \*/  
 font-size: 14px; /\* Уменьшаем шрифт по умолчанию \*/  
}  
  
.table-warning {  
 background-color: #ffff99;  
}  
  
.table-danger {  
 background-color: #ffcccc;  
}  
  
/\* Кнопки \*/  
.btn-primary, .btn-success, .btn-info, .btn-warning, .btn-danger, .btn-secondary {  
 padding: 6px 12px; /\* Уменьшаем размер кнопок \*/  
 font-size: 14px;  
}  
  
/\* Формы \*/  
.compact-form .form-control, .compact-form .form-select {  
 padding: 0.25rem 0.5rem;  
 font-size: 14px;  
}  
  
.compact-form .row {  
 margin-bottom: 0.5rem;  
}  
  
.form-label {  
 font-weight: bold;  
 margin-bottom: 0;  
}  
  
/\* Основные стили таблицы \*/  
.table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-bottom: 1rem;  
}  
  
.table th, .table td {  
 padding: 8px;  
 vertical-align: middle;  
 font-size: 14px;  
 border: 1px solid #dee2e6;  
 box-sizing: border-box;  
}  
  
.table th {  
 background-color: #343a40;  
 color: white;  
 font-weight: bold;  
}  
  
.table-striped tbody tr:nth-of-type(odd) {  
 background-color: #f8f9fa;  
}  
  
.table-warning {  
 background-color: #fff3cd;  
}  
  
.table-danger {  
 background-color: #f8d7da;  
}  
  
/\* Стили для кнопок и сортировки \*/  
.btn-group {  
 margin-bottom: 10px;  
}  
  
.btn-group .btn {  
 margin-right: 5px;  
}  
  
.sorting-controls {  
 display: flex;  
 flex-wrap: nowrap;  
 gap: 10px;  
 margin-bottom: 10px;  
}  
  
/\* Стили для заголовков \*/  
.header h1 {  
 font-size: 1.5rem;  
 margin-bottom: 0.5rem;  
}  
  
.header h2 {  
 font-size: 1.25rem;  
 margin-bottom: 0.5rem;  
}  
  
.header h3 {  
 font-size: 1rem;  
 margin-bottom: 1rem;  
}  
  
/\* Адаптивность для мобильных \*/  
@media (max-width: 768px) {  
 .table {  
 display: block;  
 overflow-x: auto;  
 white-space: nowrap;  
 -webkit-overflow-scrolling: touch;  
 }  
  
 .table thead, .table tbody, .table tr {  
 display: table;  
 width: auto;  
 }  
  
 .table th, .table td {  
 padding: 4px;  
 font-size: 11px;  
 text-overflow: ellipsis;  
 overflow: hidden;  
 white-space: nowrap;  
 }  
  
 .table th {  
 position: sticky;  
 top: 0;  
 z-index: 1;  
 }  
  
 /\* Кнопки в одну строку (для index.html) \*/  
 .table .actions {  
 display: flex;  
 flex-wrap: nowrap;  
 gap: 5px;  
 justify-content: center;  
 }  
  
 .table .btn {  
 padding: 2px 6px;  
 font-size: 10px;  
 margin: 0;  
 }  
  
 /\* Сортировка (для index.html) \*/  
 .sorting-controls {  
 flex-direction: row;  
 justify-content: space-between;  
 align-items: center;  
 }  
  
 .sorting-controls select, .sorting-controls input, .sorting-controls button {  
 font-size: 12px;  
 padding: 4px;  
 }  
  
 /\* Уменьшаем заголовки на мобильных \*/  
 .header h1 {  
 font-size: 1.2rem;  
 }  
  
 .header h2 {  
 font-size: 1rem;  
 }  
  
 .header h3 {  
 font-size: 0.9rem;  
 }  
}

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\templates\add.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Добавить сотрудника</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container mt-3">  
 <h2 class="mb-2">Добавить сотрудника</h2>  
 <form method="POST" id="employeeForm" class="compact-form">  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">ФИО:</label>  
 <div class="col">  
 <input type="text" name="fio" class="form-control form-control-sm" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Дата рождения:</label>  
 <div class="col">  
 <input type="date" name="birth\_date" class="form-control form-control-sm" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Должность:</label>  
 <div class="col">  
 <input type="text" name="position" class="form-control form-control-sm" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">По приказу № 721:</label>  
 <div class="col">  
 <input type="text" name="order\_no" class="form-control form-control-sm" required>  
 </div>  
 </div>  
  
 <!-- Осмотры -->  
 {% for exam\_type in ['ВЛК', 'КМО', 'УМО', 'КМО2'] %}  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">{{ exam\_type }} дата:</label>  
 <div class="col">  
 <input type="date" name="{{ exam\_type.lower() }}\_date" class="form-control form-control-sm">  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">{{ exam\_type }} диагноз:</label>  
 <div class="col">  
 <input type="text" name="{{ exam\_type.lower() }}\_diagnosis" class="form-control form-control-sm">  
 </div>  
 </div>  
 {% endfor %}  
  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Состояние:</label>  
 <div class="col">  
 <select name="preflight\_condition" class="form-select form-select-sm">  
 {% for condition in preflight\_conditions %}  
 <option value="{{ condition }}" {% if condition == 'Допущен' %}selected{% endif %}>{{ condition }}</option>  
 {% endfor %}  
 </select>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Примечание:</label>  
 <div class="col">  
 <input type="text" name="note" class="form-control form-control-sm">  
 </div>  
 </div>  
 <div class="row g-2">  
 <div class="col">  
 <button type="submit" class="btn btn-primary btn-sm">Добавить</button>  
 <a href="{{ url\_for('index') }}" class="btn btn-secondary btn-sm">Назад</a>  
 </div>  
 </div>  
 </form>  
 </div>  
  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
 <script>  
 document.getElementById('employeeForm').addEventListener('submit', function(event) {  
 const requiredFields = ['fio', 'birth\_date', 'position', 'order\_no'];  
 let valid = true;  
 let errorMessage = '';  
  
 requiredFields.forEach(field => {  
 const input = document.querySelector(`input[name="${field}"]`);  
 if (!input.value) {  
 valid = false;  
 errorMessage += `Поле "${input.previousElementSibling.textContent}" обязательно для заполнения.\n`;  
 }  
 });  
  
 if (!valid) {  
 event.preventDefault();  
 alert(errorMessage);  
 }  
 });  
 </script>  
</body>  
</html>

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\templates\edit.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Редактировать профиль</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container mt-3">  
 <h2 class="mb-2">Редактировать профиль: {{ employee.fio }}</h2>  
 <form method="POST" id="employeeForm" class="compact-form">  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">ФИО:</label>  
 <div class="col">  
 <input type="text" name="fio" class="form-control form-control-sm" value="{{ employee.fio }}" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Дата рождения:</label>  
 <div class="col">  
 <input type="date" name="birth\_date" class="form-control form-control-sm" value="{{ employee.birth\_date.strftime('%Y-%m-%d') if employee.birth\_date else '' }}" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Должность:</label>  
 <div class="col">  
 <input type="text" name="position" class="form-control form-control-sm" value="{{ employee.position }}" required>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">По приказу № 721:</label>  
 <div class="col">  
 <input type="text" name="order\_no" class="form-control form-control-sm" value="{{ employee.order\_no }}" required>  
 </div>  
 </div>  
  
 <!-- Осмотры -->  
 {% for exam\_type in ['ВЛК', 'КМО', 'УМО', 'КМО2'] %}  
 {% set exam = examinations | selectattr('exam\_type', 'equalto', exam\_type) | first %}  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">{{ exam\_type }} дата:</label>  
 <div class="col">  
 <input type="date" name="{{ exam\_type.lower() }}\_date" class="form-control form-control-sm" value="{{ exam.exam\_date.strftime('%Y-%m-%d') if exam else '' }}">  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">{{ exam\_type }} диагноз:</label>  
 <div class="col">  
 <input type="text" name="{{ exam\_type.lower() }}\_diagnosis" class="form-control form-control-sm" value="{{ exam.diagnosis if exam else '' }}">  
 </div>  
 </div>  
 {% endfor %}  
  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Состояние:</label>  
 <div class="col">  
 <select name="preflight\_condition" class="form-select form-select-sm">  
 {% for condition in preflight\_conditions %}  
 <option value="{{ condition }}" {% if employee.preflight\_condition == condition %}selected{% endif %}>{{ condition }}</option>  
 {% endfor %}  
 </select>  
 </div>  
 </div>  
 <div class="row g-2 align-items-center mb-2">  
 <label class="col-auto col-form-label pe-1">Примечание:</label>  
 <div class="col">  
 <input type="text" name="note" class="form-control form-control-sm" value="{{ employee.note or '' }}">  
 </div>  
 </div>  
 <div class="row g-2">  
 <div class="col">  
 <button type="submit" class="btn btn-primary btn-sm">Сохранить</button>  
 <a href="{{ url\_for('index') }}" class="btn btn-secondary btn-sm">Назад</a>  
 </div>  
 </div>  
 </form>  
 </div>  
  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
 <script>  
 document.getElementById('employeeForm').addEventListener('submit', function(event) {  
 const requiredFields = ['fio', 'birth\_date', 'position', 'order\_no'];  
 let valid = true;  
 let errorMessage = '';  
  
 requiredFields.forEach(field => {  
 const input = document.querySelector(`input[name="${field}"]`);  
 if (!input.value) {  
 valid = false;  
 errorMessage += `Поле "${input.previousElementSibling.textContent}" обязательно для заполнения.\n`;  
 }  
 });  
  
 if (!valid) {  
 event.preventDefault();  
 alert(errorMessage);  
 }  
 });  
 </script>  
</body>  
</html>

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\templates\history.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>История осмотров</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container mt-4">  
 <h1>История осмотров: {{ employee.fio }}</h1>  
 <a href="{{ url\_for('index') }}" class="btn btn-secondary mb-3">Назад</a>  
  
 <table class="table table-striped table-bordered">  
 <thead class="table-dark">  
 <tr>  
 <th>№</th>  
 <th>Тип осмотра</th>  
 <th>Дата осмотра</th>  
 <th>Срок действия</th>  
 <th>Примечание</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for exam in examinations %}  
 <tr>  
 <td>{{ loop.index }}</td>  
 <td>{{ exam.exam\_type }}</td>  
 <td>{{ exam.exam\_date.strftime('%Y-%m-%d') if exam.exam\_date else '' }}</td>  
 <td>{{ exam.expiry\_date.strftime('%Y-%m-%d') if exam.expiry\_date else '' }}</td>  
 <td>{{ exam.note or '' }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
 <script>  
 // Вычисляем ширину каждого столбца на основе самого длинного текста в этом столбце  
 function adjustColumnWidths() {  
 if (window.innerWidth <= 768) { // Только для мобильных  
 console.log("Запуск adjustColumnWidths для мобильных устройств (history.html)");  
  
 const table = document.querySelector('.table');  
 if (!table) {  
 console.log("Таблица не найдена, повторная попытка через 500мс");  
 setTimeout(adjustColumnWidths, 500);  
 return;  
 }  
  
 const rows = table.querySelectorAll('tr');  
 if (!rows.length) {  
 console.log("Таблица пуста, повторная попытка через 500мс");  
 setTimeout(adjustColumnWidths, 500);  
 return;  
 }  
  
 const columnCount = rows[0].children.length; // Количество столбцов  
 const maxWidths = Array(columnCount).fill(0); // Массив для хранения максимальных ширин  
  
 // Создаём временный элемент для измерения ширины текста  
 const tempDiv = document.createElement('div');  
 tempDiv.style.position = 'absolute';  
 tempDiv.style.visibility = 'hidden';  
 tempDiv.style.whiteSpace = 'nowrap';  
 tempDiv.style.fontSize = '11px'; // Соответствует шрифту таблицы  
 tempDiv.style.padding = '4px'; // Учитываем padding  
 document.body.appendChild(tempDiv);  
  
 // Проходим по всем строкам и ячейкам  
 rows.forEach(row => {  
 Array.from(row.children).forEach((cell, index) => {  
 tempDiv.textContent = cell.textContent.trim();  
 const textWidth = tempDiv.offsetWidth;  
 maxWidths[index] = Math.max(maxWidths[index], textWidth);  
 });  
 });  
  
 // Удаляем временный элемент  
 document.body.removeChild(tempDiv);  
  
 // Логируем вычисленные ширины  
 console.log("Максимальные ширины столбцов (в пикселях):", maxWidths);  
  
 // Применяем максимальную ширину к каждому столбцу  
 for (let colIndex = 0; colIndex < columnCount; colIndex++) {  
 const cellsInColumn = table.querySelectorAll(`th:nth-child(${colIndex + 1}), td:nth-child(${colIndex + 1})`);  
 cellsInColumn.forEach(cell => {  
 cell.style.width = `${maxWidths[colIndex]}px`;  
 cell.style.minWidth = `${maxWidths[colIndex]}px`;  
 });  
 }  
 }  
 }  
  
 // Выполняем при загрузке DOM и при изменении размера окна  
 document.addEventListener('DOMContentLoaded', adjustColumnWidths);  
 window.addEventListener('resize', adjustColumnWidths);  
 </script>  
</body>  
</html>

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\templates\import\_excel.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Импорт сотрудников из Excel</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container mt-4">  
 <h2>Импорт сотрудников из Excel</h2>  
 {% with messages = get\_flashed\_messages(with\_categories=true) %}  
 {% if messages %}  
 {% for category, message in messages %}  
 <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">  
 {{ message }}  
 <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>  
 </div>  
 {% endfor %}  
 {% endif %}  
 {% endwith %}  
 <form method="POST" enctype="multipart/form-data">  
 <div class="mb-3">  
 <label for="file" class="form-label">Выберите Excel-файл (.xls или .xlsx):</label>  
 <input type="file" name="file" id="file" class="form-control" accept=".xls,.xlsx" required>  
 </div>  
 <button type="submit" class="btn btn-primary">Загрузить</button>  
 <a href="{{ url\_for('index') }}" class="btn btn-secondary">Назад</a>  
 </form>  
 </div>  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
</body>  
</html>

## Файл: C:/Users/Nurba/PycharmProjects/militarybase2\templates\index.html

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Медицинский контроль</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">  
</head>  
<body>  
 <div class="container mt-4">  
 <div class="header text-center mb-4">  
 <h1>Ұлттық Ұлан № 3656 әскери бөлімі</h1>  
 <h2>Авиациялық қызметкерлердің денсаулық жағдайын медициналық бақылау</h2>  
 <h3>№ 699 бұйрық, 05.09.2019ж МВД РК</h3>  
 </div>  
  
 {% with messages = get\_flashed\_messages(with\_categories=true) %}  
 {% if messages %}  
 {% for category, message in messages %}  
 <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">  
 {{ message }}  
 <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>  
 </div>  
 {% endfor %}  
 {% endif %}  
 {% endwith %}  
  
 <!-- Кнопки экспорта/импорта -->  
 <div class="btn-group mb-3">  
 <a href="{{ url\_for('add') }}" class="btn btn-primary">Добавить сотрудника</a>  
 <a href="{{ url\_for('export\_excel\_xlsx') }}" class="btn btn-success">Экспорт .xlsx</a>  
 <a href="{{ url\_for('export\_excel\_xls') }}" class="btn btn-success">Экспорт .xls</a>  
 <a href="{{ url\_for('import\_excel') }}" class="btn btn-info">Импорт Excel</a>  
 </div>  
  
 <!-- Сортировка и поиск -->  
 <div class="sorting-controls mb-3">  
 <select name="sort" onchange="location = this.value;" class="form-select me-2" style="width: auto;">  
 <option value="{{ url\_for('index', page=page, per\_page=per\_page, sort='fio\_asc', search=request.args.get('search', '')) }}" {% if request.args.get('sort') == 'fio\_asc' %}selected{% endif %}>По алфавиту (А-Я)</option>  
 <option value="{{ url\_for('index', page=page, per\_page=per\_page, sort='fio\_desc', search=request.args.get('search', '')) }}" {% if request.args.get('sort') == 'fio\_desc' %}selected{% endif %}>По алфавиту (Я-А)</option>  
 <option value="{{ url\_for('index', page=page, per\_page=per\_page, sort='deadline\_asc', search=request.args.get('search', '')) }}" {% if request.args.get('sort', 'deadline\_asc') == 'deadline\_asc' %}selected{% endif %}>По ближайшему сроку</option>  
 <option value="{{ url\_for('index', page=page, per\_page=per\_page, sort='suspended', search=request.args.get('search', '')) }}" {% if request.args.get('sort') == 'suspended' %}selected{% endif %}>Только отстраненные</option>  
 </select>  
 <form method="GET" action="{{ url\_for('index') }}" class="d-flex flex-grow-1">  
 <input type="text" name="search" class="form-control me-2" placeholder="Поиск по ФИО" value="{{ request.args.get('search', '') }}">  
 <input type="hidden" name="page" value="{{ page }}">  
 <input type="hidden" name="per\_page" value="{{ per\_page }}">  
 <input type="hidden" name="sort" value="{{ request.args.get('sort', 'deadline\_asc') }}">  
 <button type="submit" class="btn btn-secondary">Применить</button>  
 </form>  
 </div>  
  
 <!-- Таблица -->  
 <table class="table table-striped table-bordered">  
 <thead class="table-dark">  
 <tr>  
 <th>№</th>  
 <th>ФИО</th>  
 <th>Дата рождения</th>  
 <th>Должность</th>  
 <th>По приказу № 721</th>  
 <th>ВЛК срок</th>  
 <th>КМО срок</th>  
 <th>УМО срок</th>  
 <th>КМО2 срок</th>  
 <th>Дней до осмотра</th>  
 <th>Состояние</th>  
 <th>Примечание</th>  
 <th>Действия</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for item in employees\_with\_expiry %}  
 {% set employee = item.employee %}  
 {% set has\_warning = item.min\_days\_left is not none and 0 <= item.min\_days\_left <= 30 %}  
 {% set has\_expired = item.min\_days\_left is not none and item.min\_days\_left < 0 %}  
 <tr class="{% if has\_expired %}table-danger{% elif has\_warning %}table-warning{% endif %}">  
 <td>{{ loop.index + (page - 1) \* per\_page }}</td>  
 <td>{{ employee.fio }}</td>  
 <td>{{ employee.birth\_date.strftime('%Y-%m-%d') if employee.birth\_date else '' }}</td>  
 <td>{{ employee.position }}</td>  
 <td>{{ employee.order\_no }}</td>  
 <td>  
 {% set vlk\_exam = employee.examinations | selectattr('exam\_type', 'equalto', 'ВЛК') | list | last %}  
 {{ vlk\_exam.exam\_date.strftime('%Y-%m-%d') if vlk\_exam else '' }}  
 </td>  
 <td>  
 {% set kmo\_exam = employee.examinations | selectattr('exam\_type', 'equalto', 'КМО') | list | last %}  
 {{ kmo\_exam.exam\_date.strftime('%Y-%m-%d') if kmo\_exam else '' }}  
 </td>  
 <td>  
 {% set umo\_exam = employee.examinations | selectattr('exam\_type', 'equalto', 'УМО') | list | last %}  
 {{ umo\_exam.exam\_date.strftime('%Y-%m-%d') if umo\_exam else '' }}  
 </td>  
 <td>  
 {% set kmo2\_exam = employee.examinations | selectattr('exam\_type', 'equalto', 'КМО2') | list | last %}  
 {{ kmo2\_exam.exam\_date.strftime('%Y-%m-%d') if kmo2\_exam else '' }}  
 </td>  
 <td>  
 {% if item.min\_days\_left is not none %}  
 {{ item.nearest\_exam }}:  
 {% if item.min\_days\_left < 0 %}  
 Истёк срок ({{-item.min\_days\_left}} дней)  
 {% elif item.min\_days\_left <= 30 %}  
 Менее месяца ({{ item.min\_days\_left }} дней)  
 {% else %}  
 {{ item.min\_days\_left }} дней  
 {% endif %}  
 {% else %}  
 Нет данных  
 {% endif %}  
 </td>  
 <td>{{ employee.preflight\_condition }}</td>  
 <td>{{ employee.note or '' }}</td>  
 <td class="actions">  
 <a href="{{ url\_for('history', id=employee.id) }}" class="btn btn-sm btn-info">История</a>  
 <a href="{{ url\_for('edit', id=employee.id) }}" class="btn btn-sm btn-warning">Редактировать</a>  
 <a href="{{ url\_for('delete', id=employee.id) }}" class="btn btn-sm btn-danger" onclick="return confirm('Вы уверены?')">Удалить</a>  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
  
 <div class="d-flex justify-content-center mt-3">  
 <nav>  
 <ul class="pagination">  
 {% if page > 1 %}  
 <li class="page-item">  
 <a class="page-link" href="{{ url\_for('index', page=page-1, per\_page=per\_page, sort=request.args.get('sort'), search=request.args.get('search')) }}">Назад</a>  
 </li>  
 {% endif %}  
 <li class="page-item disabled"><span class="page-link">Страница {{ page }} из {{ total\_pages }}</span></li>  
 {% if page < total\_pages %}  
 <li class="page-item">  
 <a class="page-link" href="{{ url\_for('index', page=page+1, per\_page=per\_page, sort=request.args.get('sort'), search=request.args.get('search')) }}">Вперед</a>  
 </li>  
 {% endif %}  
 </ul>  
 </nav>  
 </div>  
 </div>  
  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>  
 <script>  
 // Вычисляем ширину каждого столбца на основе самого длинного текста в этом столбце  
 function adjustColumnWidths() {  
 if (window.innerWidth <= 768) { // Только для мобильных  
 console.log("Запуск adjustColumnWidths для мобильных устройств");  
  
 const table = document.querySelector('.table');  
 const rows = table.querySelectorAll('tr');  
 if (!rows.length) {  
 console.log("Таблица пуста или не найдена");  
 return;  
 }  
  
 const columnCount = rows[0].children.length; // Количество столбцов  
 const maxWidths = Array(columnCount).fill(0); // Массив для хранения максимальных ширин  
  
 // Создаём временный элемент для измерения ширины текста  
 const tempDiv = document.createElement('div');  
 tempDiv.style.position = 'absolute';  
 tempDiv.style.visibility = 'hidden';  
 tempDiv.style.whiteSpace = 'nowrap';  
 tempDiv.style.fontSize = '11px'; // Соответствует шрифту таблицы  
 tempDiv.style.padding = '4px'; // Учитываем padding  
 document.body.appendChild(tempDiv);  
  
 // Проходим по всем строкам и ячейкам  
 rows.forEach(row => {  
 Array.from(row.children).forEach((cell, index) => {  
 if (cell.classList.contains('actions')) {  
 // Для столбца "Действия" измеряем ширину блока с кнопками  
 const buttonsWidth = cell.scrollWidth; // Учитываем всю ширину содержимого  
 maxWidths[index] = Math.max(maxWidths[index], buttonsWidth);  
 } else {  
 // Для остальных ячеек измеряем ширину текста  
 tempDiv.textContent = cell.textContent.trim();  
 const textWidth = tempDiv.offsetWidth;  
 maxWidths[index] = Math.max(maxWidths[index], textWidth);  
 }  
 });  
 });  
  
 // Удаляем временный элемент  
 document.body.removeChild(tempDiv);  
  
 // Логируем вычисленные ширины  
 console.log("Максимальные ширины столбцов (в пикселях):", maxWidths);  
  
 // Применяем максимальную ширину к каждому столбцу  
 for (let colIndex = 0; colIndex < columnCount; colIndex++) {  
 const cellsInColumn = table.querySelectorAll(`th:nth-child(${colIndex + 1}), td:nth-child(${colIndex + 1})`);  
 cellsInColumn.forEach(cell => {  
 cell.style.width = `${maxWidths[colIndex]}px`;  
 cell.style.minWidth = `${maxWidths[colIndex]}px`; // Устанавливаем минимальную ширину  
 });  
 }  
 }  
 }  
  
 // Выполняем при загрузке страницы и при изменении размера окна  
 window.addEventListener('load', adjustColumnWidths);  
 window.addEventListener('resize', adjustColumnWidths);  
 </script>  
</body>  
</html>