МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**КОНТРОЛЬНАЯ РАБОТА**

по дисциплине «Программирование на языке Python»

на тему: «Разработка web-приложения «Ветеринарная лечебница»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Панин М.С.

Шифр 170586

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Группа 71ПГ

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Захарова О.В.

Оценка: «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Орел 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc28386962)

[1 АНАЛИЗ ПОСТАВЛЕННОЙ ЦЕЛИ И ФОРМУЛИРОВКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 4](#_Toc28386963)

[2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 5](#_Toc28386964)

[3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 8](#_Toc28386965)

[ЗАКЛЮЧЕНИЕ 12](#_Toc28386966)

[ПРИЛОЖЕНИЕ А (обязательное). 14](#_Toc28386967)

[ПРИЛОЖЕНИЕ Б (обязательное) 18](#_Toc28386968)

ВВЕДЕНИЕ

Целью контрольной работы является разработка web-приложение «Ветеринарная лечебница» на языке Python. Необходимо спроектировать базу данных «Ветеринарная лечебница», а также реализовать добавление, редактирование и отображение информации. Отображать информации необходимо в читабельном виде, то есть не отображать внешних ключей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Выполнить анализ поставленной цели и сформулировать функциональные требования;
2. Спроектировать данное приложение;
3. Реализовать данное приложение.
4. АНАЛИЗ ПОСТАВЛЕННОЙ ЦЕЛИ И ФОРМУЛИРОВКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

По заданию на контрольную работу необходимо реализовать web-приложение. Веб приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Таким образом, необходимо разработать сайт для ветеринарной лечебницы.

Предположим, что для некоторой лечебницы, необходим сайт, для размещения в интернете услуг, оказываемые в данном учреждении, а также их цену. И к тому же лечебнице необходимо, чтобы клиенты могли записываться на приём к ветеринару прямо на сайте.

Таким образом, можно сформировать такие функциональные требования, как:

* Возможность регистрации;
* Возможность авторизации;
* Возможность изменения профиля пользователем;
* Возможность записаться к ветеринару по сети;
* Возможность просматривать список услуг;
* Возможность фильтрации при просмотре услуг.

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Для данного приложения необходимо разработать базу данных. База данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ). Рассмотрим несколько необходимых таблиц для нашей системы:

В таблице «пользователи» будут храниться данные о пользователей, а именно:

* Идентификатор пользователя – первичный ключ;
* Фамилия;
* Имя;
* Отчество;
* Телефон;
* Электронная почта;
* Уникальное имя пользователя;
* Дата регистрации пользователя;
* Дата последнего обновления пользователя;
* Хеш пароля пользователя.

Таблица «запись к ветеринару» будет хранить заявки пользователей на запись к врачу. В ней будет храниться:

* Идентификатор заявки – первичный ключ
* Желаемая дата приёма;
* Назначенная дата приёма;
* Идентификатор пользователя;
* Идентификатор статуса;
* Идентификатор ветеринара.

В таблице статус будут храниться все возможные состояния заявки. Она будет иметь такие поля, как:

* Идентификатор статуса – первичный ключ;
* Название статуса.

В таблице «ветеринар» будет храниться информация о ветеринарах, а именно:

* Идентификатор ветеринара – первичный ключ;
* Фамилия;
* Имя;
* Отчество;
* Описания ветеринара;
* Идентификатор должности.

В таблице «должность» будет список всех должностей в лечебнице, и она будет иметь поля:

* Идентификатор должности – первичный ключ;
* Название должности.

В таблице «услуги» будет храниться список всех услуг предоставляемой данной лечебницей. Она будет иметь такие поля, как:

* Идентификатор услуги – первичный ключ;
* Название услуги;
* Цена услуги;
* Идентификатор ветеринара.

Физическая схема базы данных представлена на рисунке 1.

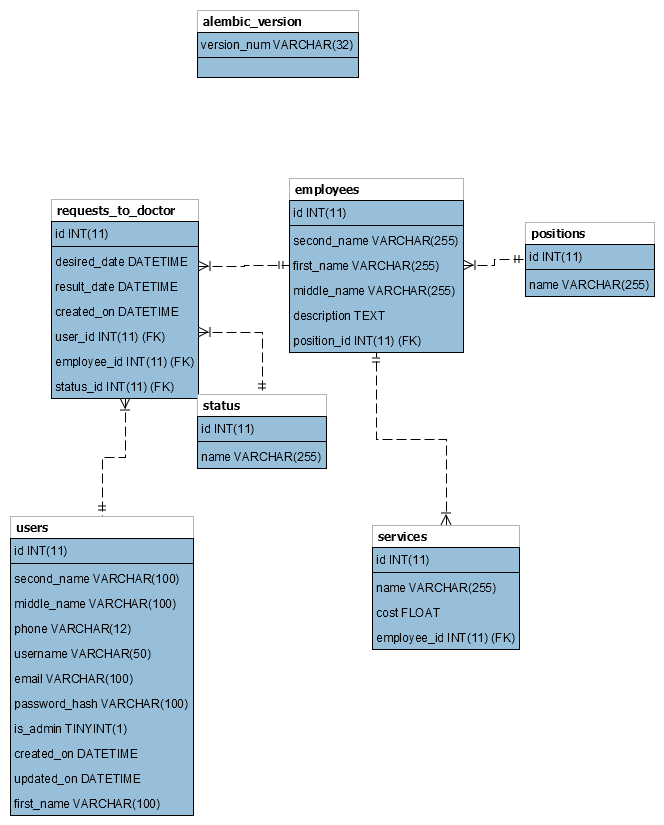


Рисунок 1 – Физическая схема базы данных «ветеринарная лечебница»

1. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

По заданию на контрольную работу необходимо разработать web-приложение на языке Python. На данном языке написано множество библиотек и фреймворков для работы с вебом. Но разрабатывать наше приложения будем с помощью фреймворка «Flask».

Flask — фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2. К данному фреймворку можно подключить множество библиотек, и одна из них – это библиотека «SQLAlchemy».

SQLAlchemy — это программная библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. Служит для синхронизации объектов Python и записей реляционной базы данных. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

Рассмотрим ту часть приложения, которая отвечает за маршрутизацию по сайту. Во фреймворке Flask к каждой функции можно привязать определённый url. Это делается с помощью декоратора @app.route(‘url’). На рисунке 2 представлен пример функции, отвечающая за главную страницу сайта.

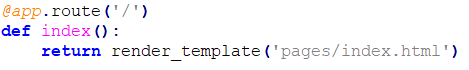


Рисунок 2 – пример функции для главной страницы.

Flask в своей работе использует шаблонизатор Jinja2. Шаблонизатор (в web) — программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц. Основная цель использования шаблонизаторов — это отделение представления данных от исполняемого кода. Одним из преимуществ шаблонизатора является то, что можно релизовать шаблон html-страницы, которым будут пользоваться все страницы сайта, изменяя его под свои нужды. Базовый шаблон сайта, в котором реализованы шапка и подвал сайта, которые на всех страницах будут одинаковыми представлен в приложении А.

На рисунке 3 представлен index.html, в котором реализована главная страница нашего сайта.

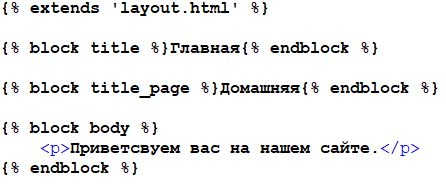


Рисунок 3 – index.html

На рисунке 4 и 5 показаны две страницы: главная и страница с контактами.

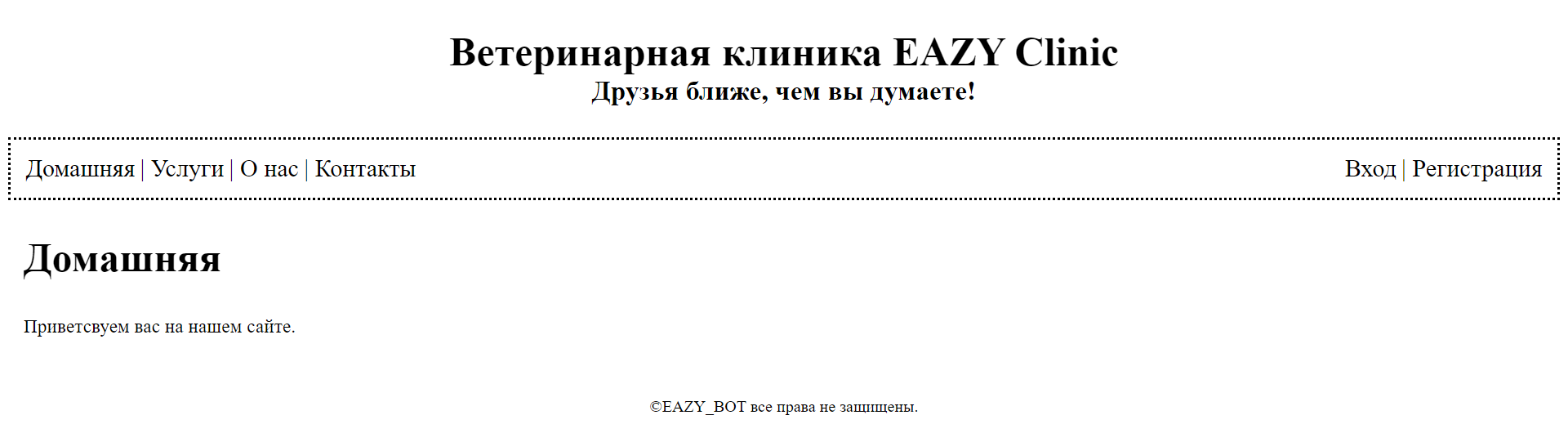


Рисунок 4 – Главная страница

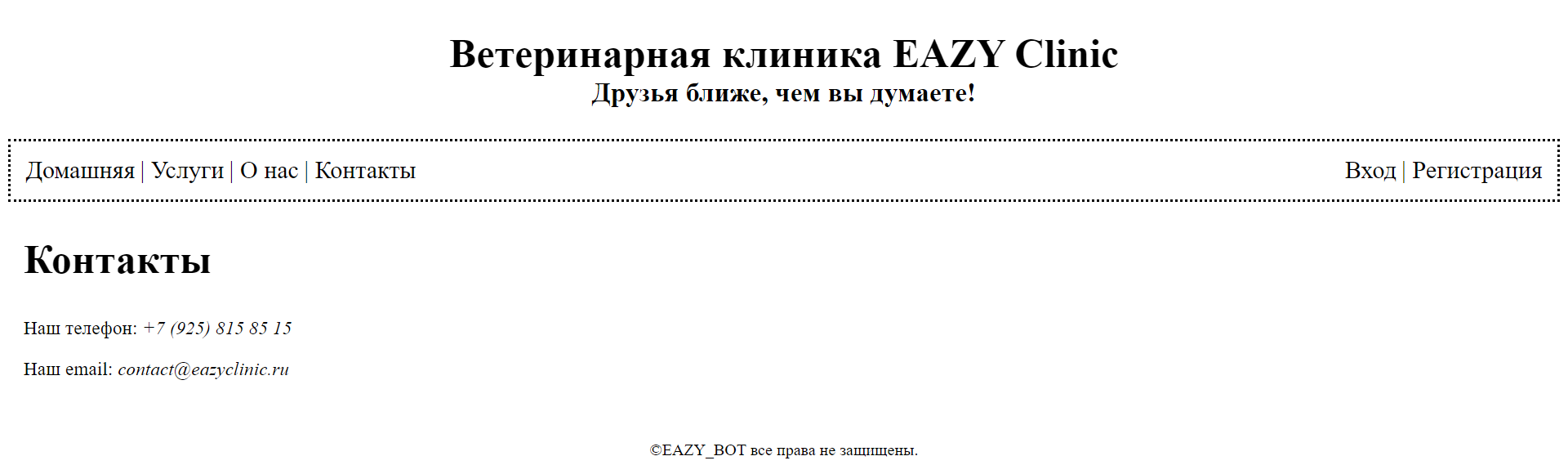


Рисунок 5 – Страница с контактами

Также для регистрации и авторизации необходимы формы. Для этого существует библиотека Flask-form. Она позволяет создавать готовые формы в коде. На рисунке 6 представлен класс, в котором реализована форма для авторизации пользователя.

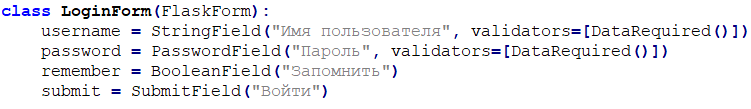


Рисунок 6 – класс LoginForm.

Также на рисунке 7 представлена функции, обрабатывающая страницу авторизации.

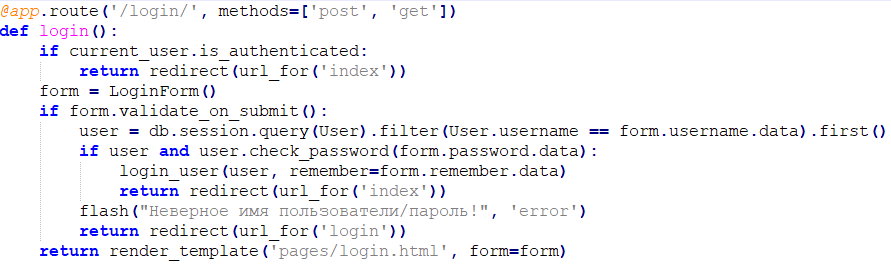


Рисунок 7 – пример функции для страницы авторизации.

Для авторизации также используются библиотека flask-login. С помощью неё можно легко реализовать авторизацию пользователя. На функции, которая представлена выше, можно увидеть, как можно авторизоваться, используя данную библиотеку

Для реализации базы данных была использована СУБД MySQL. Для работы с ней используется библиотека SQLAlchemy. На рисунке 8 представлен класс, в котором хранится информация о пользователях.

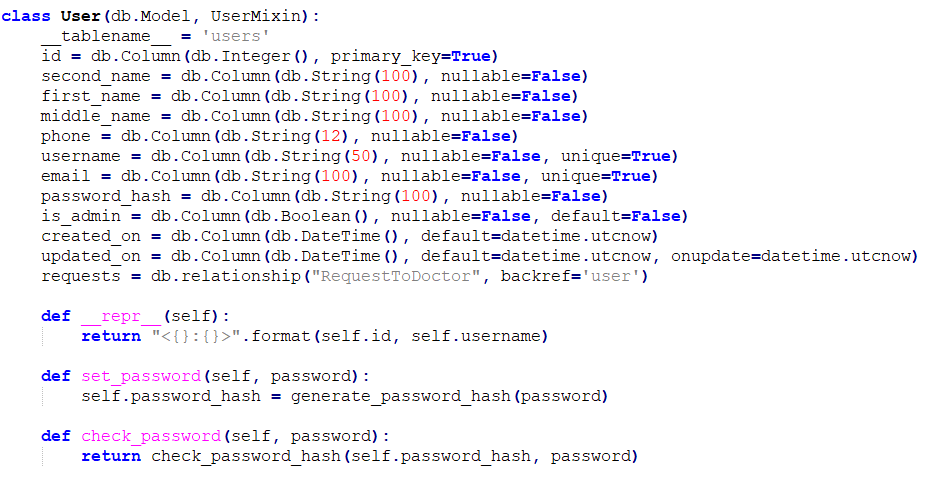


Рисунок 8 – класс User.

ЗАКЛЮЧЕНИЕ

В ходе выполнения контрольной работы была спроектирована и реализована база данных, а также были спроектировано и реализовано web-приложение на языке Python для работы с ней. Приложение позволяет работать с информацией удобным для пользователя образом. Исходя из этого, можно утверждать, что поставленные задачи были выполнены, а значит, и контрольная работа была выполнена.

# **СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Свободная энциклопедия Википедия, статья “Веб-приложение” [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Веб-приложение, свободный (дата обращения: 22.11.2019).
2. Свободная энциклопедия Википедия, статья “Flask (веб-фреймворк)” [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/Flask\_(веб-фреймворк) свободный;
3. Свободная энциклопедия Википедия, статья “SQLAlchemy” [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/SQLAlchemy свободный

ПРИЛОЖЕНИЕ А.

**(обязательное)**

**ЛИСТИНГ ОСНОВНЫХ ШАБЛОНОВ САЙТА**

Файл layout.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

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

<link href="{{ url\_for('static', filename='css/style.css')}}" rel="stylesheet" type="text/css" >

{% block head %}{% endblock %}

</head>

<body>

<header id="header">

<hgroup>

<h2>{% block wet\_title %}<a href="{{ url\_for('index') }}">Ветеринарная клиника EAZY Clinic</a>{% endblock %}</h2>

<h3>{% block wet\_tagline %}<a href="{{ url\_for('index') }}">Друзья ближе, чем вы думаете!</a>{% endblock %}</h3>

</hgroup>

<div class="top">

{% block navigation %}

<nav>

<ul class="navigation">

<li><a href="{{ url\_for('index') }}">Домашняя</a></li>

<li><a href="{{ url\_for('services') }}">Услуги</a></li>

<li><a href="{{ url\_for('about') }}">О нас</a></li>

<li><a href="{{ url\_for('contacts') }}">Контакты</a></li>

</ul>

{% block nav\_user %}

<ul class="nav\_user">

{% if current\_user.is\_authenticated %}

<li><a href="{{ url\_for('settings') }}">{{ current\_user.username }}</a></li>

<li><a href="{{ url\_for('send\_ticket') }}">Отправить заявку</a></li>

<li><a href="{{ url\_for('my\_tickets') }}">Мои заявки</a></li>

<li><a href="{{ url\_for('logout') }}">Выход</a></li>

{% else %}

<li><a href="{{ url\_for('login') }}">Вход</a></li>

<li><a href="{{ url\_for('registration') }}">Регистрация</a></li>

{% endif %}

</ul>

{% endblock %}

</nav>

{% endblock %}

</div>

</header>

<section class="wrapper">

<div class="content">

<h1>{% block title\_page %}{% endblock %}</h1>

{% block body %}{% endblock %}

</div>

</section>

<div id="footer">

{% block footer %}©EAZY\_BOT все права не защищены.{% endblock %}

</div>

</body>

</html>

Файл login.html

{% extends 'layout.html' %}

{% block head %}

<link href="{{ url\_for('static', filename='css/form.css')}}" rel="stylesheet" type="text/css" >

{% endblock %}

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

{% block title\_page %}Авторизация{% endblock %}

{% block nav\_user %}{% endblock %}

{% block body %}

{% for category, message in get\_flashed\_messages(with\_categories=true) %}

<span class="{{ category }}">{{ message }}</span>

{% endfor %}

<form action="" method="post">

{{ form.csrf\_token() }}

{% for field in form if field.name != 'csrf\_token' %}

{% if field.type != 'SubmitField' %}

{{ field.label() }}

{% endif %}

{{ field }}

{% for error in field.errors %}

{{ error }}

{% endfor %}

{% endfor %}

</form>

{% endblock %}

Файл my\_tickets.html

{% extends 'layout.html' %}

{% block title %}Мои заявки{% endblock %}

{% block title\_page %}Мои заявки{% endblock %}

{% block body %}

<table>

<tr>

<th>Номер заявки</th>

<th>Ветеринар</th>

<th>Желаемая дата</th>

<th>Назначенаая дата</th>

<th>Статус</th>

<th>Создано</th>

</tr>

{% for tick in tickets %}

<tr>

<td>{{ tick.id }}</td>

<td>

{{ tick.employee.second\_name }} {{tick.employee.first\_name}} {{ tick.employee.middle\_name }}

</td>

<td>{{ tick.desired\_date }}</td>

<td>

{% if tick.result\_date %}

{{ tick.result\_date }}

{% else %}

-

{% endif %}

</td>

<td>{{ tick.status.name }}</td>

<td>{{ tick.created\_on }}</td>

</tr>

{% endfor %}

</table>

{% endblock %}

ПРИЛОЖЕНИЕ Б

**(обязательное)**

**ЛИСТИНГ ОСНОВНЫХ ФАЙЛОВ ПРОГРАММЫ**

Файл views.py

from app import app

from flask import render\_template, request, redirect, url\_for, flash, make\_response, session

from flask\_login import login\_required, login\_user, current\_user, logout\_user

# from .models import User, Post, Category, Feedback, db

from .models import User, Employee, RequestToDoctor, Position, Status, Service, db

from .forms import RegistrationForm, LoginForm, EditProfileForm, SearchServicesForm, SendTicketForm

import datetime

@app.route('/')

def index():

return render\_template('pages/index.html')

@app.route('/about')

def about():

return render\_template('pages/about.html')

@app.route('/contacts')

def contacts():

return render\_template('pages/contacts.html')

@app.route("/services", methods=['get'])

def services():

list\_of\_services = []

form = SearchServicesForm()

if request.values:

min\_cost = request.values['min\_cost'] if request.values['min\_cost'] else 0

max\_cost = request.values['max\_cost'] if request.values['max\_cost'] else 10000000

name = request.values['name\_service'] if request.values['name\_service'] else ""

try:

min\_cost = float(min\_cost)

max\_cost = float(max\_cost)

except ValueError:

flash("Минимальная/Максимальная цена должны быть числом", "error")

redirect(url\_for('services'))

list\_of\_services = db.session.query(Service).filter(

(Service.name.like('%{}%'.format(name))) &

(Service.cost > min\_cost) & (Service.cost < max\_cost)

).all()

else:

list\_of\_services = db.session.query(Service).all()

return render\_template("pages/services.html", services=list\_of\_services, form=form)

@app.route('/login/', methods=['post', 'get'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = LoginForm()

if form.validate\_on\_submit():

user = db.session.query(User).filter(User.username == form.username.data).first()

if user and user.check\_password(form.password.data):

login\_user(user, remember=form.remember.data)

return redirect(url\_for('index'))

flash("Неверное имя пользователи/пароль!", 'error')

return redirect(url\_for('login'))

return render\_template('pages/login.html', form=form)

@app.route('/registration/', methods=['post', 'get'])

def registration():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = RegistrationForm()

if form.validate\_on\_submit():

user = db.session.query(User).filter(User.username == form.username.data.lower()).first()

if user:

flash("Такое имя пользователя занято!")

return redirect(url\_for('registration'))

user = db.session.query(User).filter(User.email == form.email.data.lower()).first()

if user:

flash("Такой email уже занят!")

return redirect(url\_for('registration'))

user = User()

user.is\_admin = False

user.username = form.username.data.lower()

user.email = form.email.data.lower()

user.first\_name = form.first\_name.data.title()

user.second\_name = form.second\_name.data.title()

user.middle\_name = form.middle\_name.data.title()

user.phone = form.phone.data

user.set\_password(form.password.data)

db.session.add(user)

db.session.commit()

flash("Вы успешно зарегистрировались! Теперь войдите под своими данными.", "success")

return redirect(url\_for('login'))

return render\_template('pages/registration.html', form=form)

@app.route('/logout/')

@login\_required

def logout():

logout\_user()

flash("Вы вышли из аккаунта!", 'success')

return redirect(url\_for('login'))

@app.route('/settings/', methods=['post', 'get'])

@login\_required

def settings():

form = EditProfileForm()

if form.validate\_on\_submit():

if current\_user.username != form.username.data:

user = db.session.query(User).filter(User.username == form.username.data.lower()).first()

if user:

flash("Такое имя пользователя занято!")

return redirect(url\_for('settings'))

current\_user.username = form.username.data

if current\_user.email != form.email.data:

user = db.session.query(User).filter(User.email == form.email.data.lower()).first()

if user:

flash("Такой email уже занят!")

return redirect(url\_for('settings'))

current\_user.email = form.email.data

if len(form.password.data) != 0:

if current\_user.check\_password(form.old\_password.data):

current\_user.set\_password(form.password.data)

else:

flash("Неверный старый пароль", "error")

return redirect(url\_for('settings'))

if current\_user.first\_name != form.first\_name.data:

current\_user.first\_name = form.first\_name.data

if current\_user.second\_name != form.second\_name.data:

current\_user.second\_name = form.second\_name.data

if current\_user.middle\_name != form.middle\_name.data:

current\_user.middle\_name = form.middle\_name.data

if str(current\_user.phone) != str(form.phone.data):

current\_user.phone = form.phone.data

print("Old: ", current\_user.phone, " New: ", form.phone.data)

db.session.add(current\_user)

db.session.commit()

flash("Успешно изменено", "success")

return redirect(url\_for('settings'))

form.username.data = current\_user.username

form.email.data = current\_user.email

form.first\_name.data = current\_user.first\_name

form.second\_name.data = current\_user.second\_name

form.middle\_name.data = current\_user.middle\_name

form.phone.data = current\_user.phone

return render\_template('pages/settings.html', form=form)

@app.route('/my\_tickets/')

@login\_required

def my\_tickets():

my\_ticks = db.session.query(RequestToDoctor).filter(RequestToDoctor.user\_id == current\_user.id).all()

return render\_template('pages/my\_tickets.html', tickets=my\_ticks)

@app.route('/send\_ticket/', methods=['post', 'get'])

@login\_required

def send\_ticket():

form = SendTicketForm()

empl = db.session.query(Employee).all()

if form.validate\_on\_submit():

ticket = RequestToDoctor()

dt\_str = form.desired\_date.data

dt = 0

try:

dt = datetime.datetime(int(dt\_str[:4]), int(dt\_str[6:7]), int(dt\_str[9:10]), int(dt\_str[12:13]),

int(dt\_str[15:]))

except ValueError:

flash("Формат YYYY-MM-DD HH:MM", "error")

redirect(url\_for('send\_ticket'))

if dt <= datetime.datetime.utcnow():

flash("Дата не может быть раньше чем сегодня", 'error')

redirect(url\_for('send\_ticket'))

ticket.desired\_date = form.desired\_date.data

ticket.user = current\_user

ticket.employee = db.session.query(Employee).filter(Employee.id == form.id\_employee.data).first()

str = "Ожидание"

ticket.status = db.session.query(Status).filter(Status.name == str).first()

db.session.add(ticket)

db.session.commit()

return redirect(url\_for('my\_tickets'))

return render\_template('pages/send\_ticket.html', employees=empl, form=form)

Файл models.py

from app import db, login\_manager

from datetime import datetime

from flask\_login import (LoginManager, UserMixin, login\_required,

login\_user, current\_user, logout\_user)

from werkzeug.security import generate\_password\_hash, check\_password\_hash

class Service(db.Model):

\_\_tablename\_\_ = 'services'

id = db.Column(db.Integer(), primary\_key=True)

name = db.Column(db.String(255), nullable=False)

cost = db.Column(db.Float(), nullable=False)

employee\_id = db.Column(db.Integer(), db.ForeignKey('employees.id'))

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.name)

class Employee(db.Model):

\_\_tablename\_\_ = 'employees'

id = db.Column(db.Integer(), primary\_key=True)

second\_name = db.Column(db.String(255), nullable=False)

first\_name = db.Column(db.String(255), nullable=False)

middle\_name = db.Column(db.String(255), nullable=False)

description = db.Column(db.Text(), nullable=False)

position\_id = db.Column(db.Integer(), db.ForeignKey('positions.id'))

requests = db.relationship("RequestToDoctor", backref='employee')

services = db.relationship("Service", backref='employee')

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.first\_name)

class Position(db.Model):

\_\_tablename\_\_ = 'positions'

id = db.Column(db.Integer(), primary\_key=True)

name = db.Column(db.String(255), nullable=False)

employees = db.relationship('Employee', backref='position')

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.name)

class Status(db.Model):

\_\_tablename\_\_ = 'status'

id = db.Column(db.Integer(), primary\_key=True)

name = db.Column(db.String(255), nullable=False)

requests = db.relationship("RequestToDoctor", backref='status')

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.name)

class RequestToDoctor(db.Model):

\_\_tablename\_\_ = 'requests\_to\_doctor'

id = db.Column(db.Integer(), primary\_key=True)

desired\_date = db.Column(db.DateTime(), nullable=False)

result\_date = db.Column(db.DateTime())

created\_on = db.Column(db.DateTime(), nullable=False, default=datetime.utcnow())

user\_id = db.Column(db.Integer(), db.ForeignKey('users.id'))

employee\_id = db.Column(db.Integer(), db.ForeignKey('employees.id'))

status\_id = db.Column(db.Integer(), db.ForeignKey('status.id'))

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.user\_id)

@login\_manager.user\_loader

def load\_user(user\_id):

return db.session.query(User).get(user\_id)

class User(db.Model, UserMixin):

\_\_tablename\_\_ = 'users'

id = db.Column(db.Integer(), primary\_key=True)

second\_name = db.Column(db.String(100), nullable=False)

first\_name = db.Column(db.String(100), nullable=False)

middle\_name = db.Column(db.String(100), nullable=False)

phone = db.Column(db.String(12), nullable=False)

username = db.Column(db.String(50), nullable=False, unique=True)

email = db.Column(db.String(100), nullable=False, unique=True)

password\_hash = db.Column(db.String(100), nullable=False)

is\_admin = db.Column(db.Boolean(), nullable=False, default=False)

created\_on = db.Column(db.DateTime(), default=datetime.utcnow)

updated\_on = db.Column(db.DateTime(), default=datetime.utcnow, onupdate=datetime.utcnow)

requests = db.relationship("RequestToDoctor", backref='user')

def \_\_repr\_\_(self):

return "<{}:{}>".format(self.id, self.username)

def set\_password(self, password):

self.password\_hash = generate\_password\_hash(password)

def check\_password(self, password):

return check\_password\_hash(self.password\_hash, password)

Файл forms.py

from flask\_wtf import FlaskForm

from wtforms import Form, ValidationError

from wtforms import StringField, SubmitField, TextAreaField, BooleanField, PasswordField, FloatField, RadioField, \

IntegerField

from wtforms.fields.html5 import DateTimeLocalField

from wtforms.validators import DataRequired, Email, Regexp, Length, EqualTo, NumberRange, Required

import re

# Custom validator

def password\_edit\_length(min\_length=-1):

message = "Пароль должен содержать не менее {} символов".format(min\_length)

def \_password\_edit\_length(form, field):

if len(field.data) != 0 and len(field.data) < min\_length:

raise ValidationError(message=message)

return \_password\_edit\_length

# Forms

class SearchServicesForm(FlaskForm):

name\_service = StringField("Название")

min\_cost = FloatField("Мин.цена", render\_kw={'placeholder': 'Мин'})

max\_cost = FloatField("Макс.цена", render\_kw={'placeholder': 'Макс'})

accept = SubmitField("Применить")

class EditProfileForm(FlaskForm):

username = StringField("Никнейм", validators=[DataRequired(), Regexp(r"[a-zA-Z\_0-9]+",

message="Используйте только латинские буквы, "

"цифры и '\_'")])

old\_password = PasswordField("Старый пароль")

password = PasswordField("Пароль", validators=[password\_edit\_length(min\_length=4)])

confirm\_password = PasswordField("Потвердите пароль",

validators=[EqualTo('password', message='Разные пароли!')])

email = StringField("Email", validators=[Email(message="Неправильный Email"), DataRequired()])

second\_name = StringField("Фамилия", validators=[DataRequired()])

first\_name = StringField("Имя", validators=[DataRequired()])

middle\_name = StringField("Отчество", validators=[DataRequired()])

phone = StringField("Телефон", validators=[DataRequired(), Regexp(r"\+?\d{11}", message="Неверный формат")])

submit = SubmitField("Изменить")

class RegistrationForm(FlaskForm):

username = StringField("Никнейм", validators=[DataRequired(), Regexp(r"[a-zA-Z\_0-9]+",

message="Используйте только латинские буквы, "

"цифры и '\_'")])

password = PasswordField("Пароль", validators=[DataRequired(), Length(min=4,

message="Пароль должен содержать "

"не менее 4 символов")])

confirm\_password = PasswordField("Потвердите пароль",

validators=[DataRequired(), EqualTo('password', message='Разные пароли!')])

email = StringField("Email", validators=[Email(message="Неправильный Email"), DataRequired()])

second\_name = StringField("Фамилия", validators=[DataRequired()])

first\_name = StringField("Имя", validators=[DataRequired()])

middle\_name = StringField("Отчество", validators=[DataRequired()])

phone = StringField("Телефон", validators=[DataRequired(), Regexp(r"\+?\d{11}", message="Неверный формат")])

submit = SubmitField("Зарегистрироваться")

class LoginForm(FlaskForm):

username = StringField("Имя пользователя", validators=[DataRequired()])

password = PasswordField("Пароль", validators=[DataRequired()])

remember = BooleanField("Запомнить")

submit = SubmitField("Войти")

class SendTicketForm(FlaskForm):

id\_employee = IntegerField("ID Ветеринара", validators=[DataRequired()])

desired\_date = StringField("Желаемая дата", validators=[Regexp(r"\d{4}-\d{2}-\d{2} \d{2}:\d{2}",

message="Формат YYYY-MM-DD HH:MM")])

submit = SubmitField("Отправить")