**Задание 1 Профиль**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Профиль Ведьмака</title>

<style>

body {

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

background-color: #2c2c2c; /\* Темный фон \*/

color: #f8f8f2; /\* Светлый текст \*/

margin: 0;

padding: 20px;

}

h1 {

text-align: center;

border-bottom: 2px solid #f8f8f2;

padding-bottom: 10px;

margin-bottom: 20px;

}

.container {

max-width: 600px;

margin: auto;

background-color: #1c1c1c; /\* Дефолтный фон для контейнера \*/

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(255, 255, 255, 0.1);

}

.section {

margin-bottom: 20px;

}

.section-title {

font-size: 1.5em;

border-bottom: 1px solid #f8f8f2;

margin-bottom: 10px;

padding-bottom: 5px;

}

.list {

list-style-type: none;

padding: 0;

}

.list-item {

background-color: #333333; /\* Темная закраска для каждого элемента списка \*/

margin: 5px 0;

padding: 10px;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="container">

<h1>{{ witcher.name }}</h1>

<div class="section">

<div class="section-title">Школа:</div>

<p>{{ witcher.school }}</p>

</div>

<div class="section">

<div class="section-title">Знаки:</div>

<ul class="list">

{% for sign in witcher.signs %}

<li class="list-item">{{ sign }}</li>

{% endfor %}

</ul>

</div>

<div class="section">

<div class="section-title">Статистика:</div>

<ul class="list">

<li class="list-item">Уровень токсичности: {{ witcher.toxicity }}</li>

<li class="list-item">Жизненная сила: {{ witcher.health }}</li>

<li class="list-item">Уровень опыта: {{ witcher.experience }}</li>

</ul>

</div>

</div>

</body>

</html>

Pytchon:

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/witcher\_profile')

def witcher\_profile():

witcher = {

'name': 'Геральт из Ривии',

'school': 'Школа Волка',

'signs': ['Игни', 'Аард', 'Книн', 'Дельи'],

'toxicity': 15,

'health': 100,

'experience': 2000

}

return render\_template('witcher\_profile.html', witcher=witcher)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Задание 2 Инвентарь ведьмака**

from flask import Flask, render\_template, jsonify, request

app = Flask(\_\_name\_\_)

# Мок-данные для алхимических предметов

alchemy\_items = [

{'name': 'Черная кровь', 'type': 'potion', 'toxicity': 10},

{'name': 'Золотая иволга', 'type': 'potion', 'toxicity': 5},

{'name': 'Маска смерти', 'type': 'bomb', 'toxicity': 15},

{'name': 'Бомба удачи', 'type': 'bomb', 'toxicity': 8},

{'name': 'Эликсир здоровья', 'type': 'potion', 'toxicity': 3},

{'name': 'Кровь дракона', 'type': 'potion', 'toxicity': 20},

]

@app.route('/alchemy', methods=['GET'])

def alchemy():

item\_type = request.args.get('type')

max\_toxicity = request.args.get('toxicity', type=int)

filtered\_items = [

item for item in alchemy\_items

if item['type'] == item\_type and (max\_toxicity is None or item['toxicity'] <= max\_toxicity)

]

return jsonify(filtered\_items)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Задание 3 аунтификация по школам.**

Приложение:

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

from flask\_sqlalchemy import SQLAlchemy

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

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key' # Секретный ключ для сессий

db = SQLAlchemy(app)

# Имитация модели пользователя

class User:

def \_\_init\_\_(self, username, school):

self.username = username

self.school = school

# Mock users

users = [

User('Geralt', 'Волк'),

User('Vesemir', 'Волк'),

User('Yennefer', 'Гадюка'),

User('Triss', 'Грифон')

]

# Доступные школы

SCHOOLS = ['Волк', 'Гадюка', 'Грифон']

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form.get('username')

school = request.form.get('school')

# Проверка существования пользователя

user = next((user for user in users if user.username == username), None)

if user and user.school == school:

session['username'] = username

session['school'] = school

flash(f'Вы успешно вошли как {username} из школы {school}.')

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

else:

flash('Некорректное имя или школа!')

return render\_template('login.html', schools=SCHOOLS)

@app.route('/home')

@login\_required

def home():

return f'Добро пожаловать, {session["username"]}! Ваш выбор: {session["school"]}.'

@app.route('/kaermorhen')

@login\_required

def kaermorhen():

if session.get('school') != 'Волк':

flash('У вас нет доступа к этой странице.')

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

return 'Добро пожаловать в Каэр Морхен!'

@app.route('/logout')

@login\_required

def logout():

session.pop('username', None)

session.pop('school', None)

flash('Вы успешно вышли из системы.')

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

Login.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Вход</title>

</head>

<body>

<h1>Вход</h1>

<form method="POST">

<label for="username">Имя:</label>

<input type="text" id="username" name="username" required><br>

<label for="school">Школа:</label>

<select id="school" name="school" required>

<option value="">Выберите школу</option>

{% for school in schools %}

<option value="{{ school }}">{{ school }}</option>

{% endfor %}

</select><br>

<input type="submit" value="Войти">

</form>

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<ul>

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

</body>

</html>

Index.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Главная</title>

</head>

<body>

<h1>Добро пожаловать в мир Ведьмака!</h1>

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

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

<a href="{{ url\_for('kaermorhen') }}">Каэр Морхен</a>

</body>

</html>

**Задание 4. Бестиарий**

import json

class Monster:

def \_\_init\_\_(self, name, monster\_type, weakness):

self.name = name

self.monster\_type = monster\_type

self.weakness = weakness

def to\_dict(self):

return {

'name': self.name,

'type': self.monster\_type,

'weakness': self.weakness

}

class Bestiary:

def \_\_init\_\_(self):

self.monsters = {}

def add\_monster(self, name, monster\_type, weakness):

if name not in self.monsters:

monster = Monster(name, monster\_type, weakness)

self.monsters[name] = monster

print(f'Монстр "{name}" добавлен в бестиарий.')

else:

print(f'Монстр с именем "{name}" уже существует.')

def remove\_monster(self, name):

if name in self.monsters:

del self.monsters[name]

print(f'Монстр "{name}" удален из бестиария.')

else:

print(f'Монстр с именем "{name}" не найден.')

def search\_by\_weakness(self, weakness):

found = [monster.to\_dict() for monster in self.monsters.values() if monster.weakness.lower() == weakness.lower()]

if not found:

print(f'Монстры с уязвимостью к "{weakness}" не найдены.')

return

print(json.dumps(found, indent=4, ensure\_ascii=False))

def display\_monsters(self):

if not self.monsters:

print('Бестиарий пуст.')

else:

for monster in self.monsters.values():

print(f'Имя: {monster.name}, Тип: {monster.monster\_type}, Слабость: {monster.weakness}')

def main():

bestiary = Bestiary()

while True:

print("\nМеню:")

print("1. Добавить монстра")

print("2. Удалить монстра")

print("3. Искать монстров по слабости")

print("4. Показать всех монстров")

print("5. Выйти")

choice = input("Выберите действие (1-5): ")

if choice == '1':

name = input("Введите имя монстра: ")

monster\_type = input("Введите тип монстра: ")

weakness = input("Введите слабость монстра: ")

bestiary.add\_monster(name, monster\_type, weakness)

elif choice == '2':

name = input("Введите имя монстра для удаления: ")

bestiary.remove\_monster(name)

elif choice == '3':

weakness = input("Введите слабость для поиска (например, 'Серебро' или 'Игни'): ")

bestiary.search\_by\_weakness(weakness)

elif choice == '4':

bestiary.display\_monsters()

elif choice == '5':

print("Выход из программы.")

break

else:

print("Неверный выбор. Пожалуйста, выберите число от 1 до 5.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Задание 5.**

5.1

from flask import Flask, render\_template, redirect, url\_for, request, session, flash, jsonify

import csv

from datetime import datetime

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key' # Секретный ключ для сессий

# Имитация модели пользователя

class User:

def \_\_init\_\_(self, username, school, rank):

self.username = username

self.school = school

self.rank = rank

# Mock users

users = [

User('Geralt', 'Волк', 'Master'),

User('Vesemir', 'Волк', 'Master'),

User('Yennefer', 'Гадюка', 'Novice'),

User('Triss', 'Грифон', 'Novice')

]

@app.route('/contracts')

def contracts():

# Проверка прав доступа для пользователя

if session.get('rank') != 'Master':

flash('У вас нет доступа к этому маршруту.')

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

# Здесь может быть логика для отображения контрактов

return 'Список контрактов доступен.'

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form.get('username')

school = request.form.get('school')

# Проверка существования пользователя

user = next((user for user in users if user.username == username), None)

if user:

session['username'] = username

session['school'] = user.school

session['rank'] = user.rank

flash(f'Вы успешно вошли как {username} из школы {user.school}.')

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

else:

flash('Пользователь не найден!')

return render\_template('login.html')

@app.route('/home')

def home():

return 'Добро пожаловать на главную страницу!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

5.2

# Mock data for completed contracts

completed\_contracts = [

{'monster': 'Грифон', 'reward': 150, 'date': '2023-01-10'},

{'monster': 'Вампайр', 'reward': 200, 'date': '2023-02-05'},

{'monster': 'Медведь', 'reward': 100, 'date': '2023-03-15'}

]

@app.route('/generate\_report')

def generate\_report():

if session.get('rank') != 'Master':

return "Доступ запрещен", 403

# Create CSV report

report\_data = "Монстр, Вознаграждение, Дата\n"

for contract in completed\_contracts:

report\_data += f"{contract['monster']}, {contract['reward']}, {contract['date']}\n"

response = app.response\_class(

response=report\_data,

mimetype='text/csv',

headers={"Content-Disposition": "attachment;filename=completed\_contracts.csv"}

)

return response

**Задание 6. Отзывы**

from flask import Flask, render\_template, request, jsonify, session, flash

import json

import os

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Путь к файлу JSON для хранения отзывов

REVIEW\_FILE = 'reviews.json'

# Функция для загрузки отзывов из файла

def load\_reviews():

if os.path.exists(REVIEW\_FILE):

with open(REVIEW\_FILE, 'r', encoding='utf-8') as f:

return json.load(f)

return []

# Функция для сохранения отзывов в файл

def save\_reviews(reviews):

with open(REVIEW\_FILE, 'w', encoding='utf-8') as f:

json.dump(reviews, f, ensure\_ascii=False, indent=4)

@app.route('/reviews', methods=['GET', 'POST'])

def reviews():

if request.method == 'POST':

# Получаем данные из формы

username = session.get('username', 'Гость') # Используем имя пользователя из сессии или Гость

rating = request.form.get('rating')

comment = request.form.get('comment')

# Загружаем существующие отзывы

reviews = load\_reviews()

# Создаем новый отзыв

new\_review = {

'username': username,

'rating': int(rating),

'comment': comment

}

# Добавляем новый отзыв и сохраняем

reviews.append(new\_review)

save\_reviews(reviews)

flash('Ваш отзыв успешно добавлен!')

return redirect('/reviews')

# Отображение всех отзывов

reviews = load\_reviews()

return render\_template('reviews.html', reviews=reviews)

@app.route('/view\_reviews')

def view\_reviews():

reviews = load\_reviews()

return jsonify(reviews)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

reviews.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Отзывы о контрактах</title>

</head>

<body>

<h1>Отзывы о контрактах</h1>

<form action="/reviews" method="POST">

<label for="rating">Оценка (1-5):</label>

<input type="number" id="rating" name="rating" min="1" max="5" required>

<br>

<label for="comment">Комментарий:</label>

<textarea id="comment" name="comment" required></textarea>

<br>

<input type="submit" value="Отправить отзыв">

</form>

<h2>Существующие отзывы:</h2>

<ul>

{% for review in reviews %}

<li>

<strong>{{ review.username }}</strong>: {{ review.rating }} звезд(ы) - {{ review.comment }}

</li>

{% else %}

<li>Нет отзывов.</li>

{% endfor %}

</ul>

</body>

</html>

**Доп задание:**

from flask import Flask, render\_template, session, jsonify

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Примеры данных

completed\_contracts = [

{'monster': 'Грифон', 'reward': 150},

{'monster': 'Вампайр', 'reward': 200},

{'monster': 'Медведь', 'reward': 100},

]

equipment = {

'sword': 'Steel Sword',

'silver\_sword': 'Silver Sword',

'armor': 'Witcher Armor',

}

active\_quests = [

'Уничтожить Грифона',

'Спасти деревню от вампиров',

]

alchemy\_items = [

{'name': 'Черная кровь', 'type': 'potion', 'toxicity': 40},

{'name': 'Золотая иволга', 'type': 'potion', 'toxicity': 5},

{'name': 'Маска смерти', 'type': 'bomb', 'toxicity': 15},

]

@app.route('/initialize\_session')

def initialize\_session():

session['toxicity'] = 20

return "Сессия инициализирована!"

@app.route('/witcher/stats')

def witcher\_stats():

return render\_template('witcher\_stats.html', equipment=equipment, toxicity=session['toxicity'], active\_quests=active\_quests)

def calculate\_total\_gold(contracts):

return sum(contract['reward'] for contract in contracts)

@app.route('/calculate\_gold')

def calculate\_gold():

total\_gold = calculate\_total\_gold(completed\_contracts)

return f'Общее количество золота: {total\_gold}'

@app.route('/alchemy', methods=['GET'])

def alchemy():

max\_toxicity = request.args.get('toxicity', type=int)

filtered\_items = [item for item in alchemy\_items if item['toxicity'] > max\_toxicity]

return jsonify(filtered\_items)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Witcher-stats.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Статистика Ведьмака</title>

</head>

<body>

<h1>Статистика Геральта</h1>

<h2>Снаряжение:</h2>

<ul>

<li>Меч: {{ equipment.sword }}</li>

<li>Серебряный меч: {{ equipment.silver\_sword }}</li>

<li>Броня: {{ equipment.armor }}</li>

</ul>

<h2>Уровень токсичности: {{ toxicity }}</h2>

<h2>Активные квесты:</h2>

<ul>

{% for quest in active\_quests %}

<li>{{ quest }}</li>

{% endfor %}

</ul>

</body>

</html>