

**Python Flask   
Aplikacje Webowe**

**Podręcznik kursowy - kod**

Mobilo © 2021

Spis treści

[Wstęp - Dlaczego Flask? 5](#_Toc69417954)

[Środowisko wirtualne, instalacja Flask, pierwszy program 6](#_Toc69417955)

[Terminal, zmienne środowiskowe i typowe problemy 7](#_Toc69417956)

[Dynamiczny routing 8](#_Toc69417957)

[Przekazywanie parametrów przez Query String 9](#_Toc69417958)

[Przyjmowanie danych z formularzy 10](#_Toc69417959)

[Obsługa GET i POST w jednej funkcji. Obiekt request 11](#_Toc69417960)

[Funkcja url\_for – Twój przyjaciel w budowaniu linków 12](#_Toc69417961)

[Funkcja redirect, czyli “zapraszamy do kasy obok” 13](#_Toc69417962)

[Statyczne elementy projetku 14](#_Toc69417963)

[Definiowanie środowiska Flask 15](#_Toc69417964)

[Szablony Jinja 16](#_Toc69417965)

[Warunkowe wyświetlanie części szablonu Jinja 17](#_Toc69417966)

[Pętle w szablonach Jinja 18](#_Toc69417967)

[Funkcja flash() – czyli krótkie info o statusie 20](#_Toc69417968)

[Makra Jinja 21](#_Toc69417969)

[Dziedziczenie w szablonach i budowanie własnych standardów 22](#_Toc69417970)

[Dołączanie szablonów Jinja - include 23](#_Toc69417971)

[Flask Bootstrap 24](#_Toc69417972)

[Korzystanie z Bootstrap bez Flask Bootstrap 25](#_Toc69417973)

[Połącznie kodu z szablonami 26](#_Toc69417974)

[Stosowanie formatów bootsrap 28](#_Toc69417975)

[Projekt 02 – Formularz z pomysłami wycieczek 30](#_Toc69417976)

[Instalacja SQLite 3 i podstawy pracy z danymi 32](#_Toc69417977)

[Zapis danych z rekordu do bazy danych 33](#_Toc69417978)

[Pobieranie rekordów z bazy danych 34](#_Toc69417979)

[Kasowanie rekordu z pytaniem o potwierdzenie 36](#_Toc69417980)

[Modyfikacja danych 37](#_Toc69417981)

[Użytkownicy aplikacji - przygotowanie 39](#_Toc69417982)

[Sesja w akcji – logowanie i wylogowanie użytkownika 41](#_Toc69417983)

[Dodawanie użytkowników – zadbaj o wygodę użytkownika 43](#_Toc69417984)

[Lista użytkowników 45](#_Toc69417985)

[Edycja użytkownika 47](#_Toc69417986)

[Edycja uprawnień 49](#_Toc69417987)

[Implementacja uprawnień w aplikacji 50](#_Toc69417988)

[Flask SQLAlchemy 51](#_Toc69417989)

[Dodawanie I pobieranie danych z bazy danych 52](#_Toc69417990)

[Filtrowanie danych 53](#_Toc69417991)

[Sortowanie,zliczanie i ograniczanie liczby rekordów 54](#_Toc69417992)

[Dodawanie i usuwanie danych 55](#_Toc69417993)

[Relacyjne bazy danych 56](#_Toc69417994)

[Porównanie metod pracy z bazą danych 57](#_Toc69417995)

[Flask-WTF – instalacja i pierwszy formularz 59](#_Toc69417996)

[Sprawdzanie poprawności danych (validators) 60](#_Toc69417997)

[Zapisywanie danych z formularzy w obiektach 61](#_Toc69417998)

[Dziedziczenie z klasy formularza 62](#_Toc69417999)

[Korzystanie z kontrolek WTForms 63](#_Toc69418000)

[Kontrolki HTML5 64](#_Toc69418001)

[Flask Login – testowa aplikacja 65](#_Toc69418002)

[Flask-Login instalacja i wykorzystanie 67](#_Toc69418003)

[Przekierowanie użytkownika do strony logowania 68](#_Toc69418004)

[Wymuszenie ponownego logowania 69](#_Toc69418005)

[Spróbuj też! 70](#_Toc69418006)

Wstęp - Dlaczego Flask?

Tutaj znajdziesz fragmenty kodu, które będzie ewentualnie łatwiej kopiować i wklejać do swoich programów w razie potrzeby.

Środowisko wirtualne, instalacja Flask, pierwszy program

Propozycja rozwiązania (tylko kod aplikacji)

from flask import Flask

from datetime import datetime

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

@app.route('/')

def index():

time\_now = datetime.now().strftime('%H:%M:%S')

return '<h1>Hello World: {}</h1>'.format(time\_now)

@app.route('/links')

def links():

body = '''<a href="http://www.google.com" target="\_blank">Google</a> <br />

<a href="http://www.bing.com" target="\_blank">Default search engine to find Google</a>'''

return body

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

app.run()

Terminal, zmienne środowiskowe i typowe problemy

Propozycja rozwiązania

set FLASK\_APP=time\_app.py

set FLASK\_DEBUG=1

Na Linux

export FLASK\_APP=time\_app.py

export FLASK\_DEBUG=1

Dynamiczny routing

Propozycja rozwiązania

@app.route('/cook/<string:receipt>/<int:step>')

def cook(receipt, step):

    body = f'''<H1>In the receipt {receipt} you are on step {step}</H1>'''

    return  body

Przekazywanie parametrów przez Query String

Propozycja rozwiązania

@app.route('/cook/<string:receipt>/<int:step>')

def cook(receipt, step):

print(request.query\_string)

print('----')

print(request.args)

font\_size='100%'

if 'font-size' in request.args:

font\_size = request.args['font-size']

body = f'''<H1 style="font-size: { font\_size }">In the receipt {receipt} you are on step {step}</H1>'''

return body

Przyjmowanie danych z formularzy

Propozycja rozwiązania

from flask import Flask, request

# …

@app.route('/rate\_receipt')

def rate\_receipt():

body = '''

<form id="rating" action="/rate\_receipt\_save" method="POST">

<label for=note>What is your note for the receipt?</label><br>

<select id="nore" name="note">

<option value="5">It is great!</option>

<option value="4">It is very good</option>

<option value="3" selected>It is just good</option>

<option value="2">It was poor</option>

<option value="1">It was horrible!</option>

</select><br>

<label for=comment>Write down your comments:</label><br>

<textarea id="comment" name="comment" rows="3" cols="50">

</textarea><br>

<label for="decision">Would you cook it for your family?</label><br>

<input type="checkbox" id="decision" name="decision"><br>

<input type="submit" value="Share my feedback">

</form>

'''

return body

@app.route('/rate\_receipt\_save', methods=['POST'])

def rate\_receipt\_save():

note = 3

if 'note' in request.form:

note = request.form['note']

comment=''

if 'comment' in request.form:

comment = request.form['comment']

decision = False

if 'decision' in request.form:

decision = True

message = f'''Your rating was: {note}<br>

Your comment was: {comment}<br>

Your decision was {decision}

'''

return message

Obsługa GET i POST w jednej funkcji. Obiekt request

Propozycja rozwiązania

from flask import Flask, request

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

@app.route('/rate\_receipt', methods=['GET', 'POST'])

def rate\_receipt():

if request.method == 'GET':

body = '''

<form id="rating" action="/rate\_receipt" method="POST">

<label for=note>What is your note for the receipt?</label><br>

<select id="nore" name="note">

<option value="5">It is great!</option>

<option value="4">It is very good</option>

<option value="3" selected>It is just good</option>

<option value="2">It was poor</option>

<option value="1">It was horrible!</option>

</select><br>

<label for=comment>Write down your comments:</label><br>

<textarea id="comment" name="comment" rows="3" cols="50">

</textarea><br>

<label for="decision">Would you cook it for your family?</label><br>

<input type="checkbox" id="decision" name="decision"><br>

<input type="submit" value="Share my feedback">

</form>

'''

return body

else:

note = 3

if 'note' in request.form:

note = request.form['note']

comment=''

if 'comment' in request.form:

comment = request.form['comment']

decision = False

if 'decision' in request.form:

decision = True

message = f'''Your rating was: {note}<br>

Your comment was: {comment}<br>

Your decision was {decision}

'''

return message

# …

Funkcja url\_for – Twój przyjaciel w budowaniu linków

Propozycja rozwiązania

from flask import Flask, request, url\_for

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

@app.route('/rate\_receipt', methods=['GET', 'POST'])

def rate\_receipt():

if request.method == 'GET':

body = f'''

<form id="rating" action="{ url\_for('rate\_receipt') }" method="POST">

#... etc

Funkcja redirect, czyli “zapraszamy do kasy obok”

Propozycja rozwiązania

from flask import Flask, url\_for, redirect

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

@app.route('/not\_implemented/<message>')

def not\_implemented(message):

return '<h1 style="color:red">{}</h1>'.format(message)

@app.route('/new\_receipt')

def new\_receipt():

return redirect(url\_for('not\_implemented', message="Function new\_receipt is not ready yet"))

@app.route('/delete\_receipt/<name>')

def delete\_receipt(name):

return redirect(url\_for('not\_implemented', message="Function new\_receipt is not ready yet"))

Statyczne elementy projetku

Propozycja rozwiązania

from flask import Flask, url\_for, redirect

import os

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

@app.route('/download/<file>')

def download(file):

subpath = 'download/'+file

local\_file\_path = os.path.join(app.static\_folder, subpath)

if(os.path.isfile(local\_file\_path)):

print('file found')

return redirect(url\_for('static', filename=subpath))

else:

print('sorry file not found')

return 'File not found!'

Definiowanie środowiska Flask

Propozycja rozwiązania

W pliku activate dodaj: SET FLASK\_ENV=development lub EXPORT FLASK\_ENV an Linuxie

flask routes

Szablony Jinja

Propozycja rozwiązania

# templates/notification.html

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

<form id="notification\_form" method="POST" action="{{ url\_for('notification') }}">

<label for="room\_number">Room number</label>

<input type="text" name="room\_number" id="room\_number"><br>

<label for="guest\_name">Guest name</label>

<input type="text" name="guest\_name" id="guest\_name"><br>

<label for="notification\_text">Notification</label>

<textarea rows="4" cols="50" name="notification\_text"></textarea><br>

<input type="submit" value="Send">

</form>

</body>

</html>

# templates/notification\_content.html

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

<h1>Notification content</h1>

<ul>

<li style="font-weight: bold;">Room number</li>

<li>{{ room\_number }}</li>

<li style="font-weight: bold;">Guest name</li>

<li>{{ guest\_name }}</li>

<li style="font-weight: bold;">Notification</li>

<li>{{ notification\_text }}</li>

</ul>

</body>

</html>

# app.py

from flask import Flask, url\_for, request, render\_template

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

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

def notification():

if request.method == 'GET':

return render\_template('notification.html')

else:

room\_number = request.form['room\_number'] if 'room\_number' in request.form else ''

guest\_name = request.form['guest\_name'] if 'guest\_name' in request.form else ''

notification\_text = request.form['notification\_text'] if 'notification\_text' in request.form else ''

return render\_template('notification\_content.html',

room\_number=room\_number, guest\_name=guest\_name, notification\_text=notification\_text)

Warunkowe wyświetlanie części szablonu Jinja

Propozycja rozwiązania

# template notification.html (only added part)

<label for="priority">Priority</label>

<select name="priority" id="priority">

<option value="high">HIGH PRIORITY</option>

<option value="medium">MEDIUM</option>

<option value="normal" selected>NOT URGENT</option>

</select><br>

# ----- getting new field value from the request (only modified part)

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

def notification():

if request.method == 'GET':

return render\_template('notification.html')

else:

room\_number = request.form['room\_number'] if 'room\_number' in request.form else ''

guest\_name = request.form['guest\_name'] if 'guest\_name' in request.form else ''

notification\_text = request.form['notification\_text'] if 'notification\_text' in request.form else ''

priority = request.form['priority'] if 'priority' in request.form else 'normal'

return render\_template('notification\_content.html',

room\_number=room\_number, guest\_name=guest\_name,

notification\_text=notification\_text, priority=priority)

# template notification\_confirmation.html. Only added part

<body>

{% if priority=='high' %}

<h1>Critical notification content</h1>

{% elif priority=='medium' %}

<h1>Important notification content</h1>

{% else %}

<h1>Notification content</h1>

{% endif %}

Pętle w szablonach Jinja

Propozycja rozwiązania

app.py:

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

import os

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

class PriorityType:

def \_\_init\_\_(self, code, description, selected):

self.code = code

self.description = description

self.selected = selected

class NotificationPriorities:

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

self.list\_of\_priorities = []

def load\_priorities(self):

self.list\_of\_priorities.append(PriorityType('high','HIGH PRIORITY', False))

self.list\_of\_priorities.append(PriorityType('medium','MEDIUM', False))

self.list\_of\_priorities.append(PriorityType('normal', 'NOT URGENT', True))

self.list\_of\_priorities.append(PriorityType('low', 'REMARK', False))

def get\_priority\_by\_code(self, code):

for p in self.list\_of\_priorities:

if p.code == code:

return p

return PriorityType('normal', 'NOT URGENT', True)

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

def notification():

notification\_priorities = NotificationPriorities()

notification\_priorities.load\_priorities()

if request.method == 'GET':

return render\_template('notification.html',

list\_of\_priorities=notification\_priorities.list\_of\_priorities)

else:

room\_number = request.form['room\_number'] if 'room\_number' in request.form else ''

guest\_name = request.form['guest\_name'] if 'guest\_name' in request.form else ''

notification\_text = request.form['notification\_text'] if 'notification\_text' in request.form else ''

priority = request.form['priority'] if 'priority' in request.form else 'normal'

priority\_type = notification\_priorities.get\_priority\_by\_code(priority)

print('found', priority\_type.code)

return render\_template('notification\_content.html',

room\_number=room\_number, guest\_name=guest\_name,

notification\_text=notification\_text, priority\_type=priority\_type)

notification.html:

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

<form id="notification\_form" method="POST" action="{{ url\_for('notification') }}">

<label for="room\_number">Room number</label>

<input type="text" name="room\_number" id="room\_number"><br>

<label for="guest\_name">Guest name</label>

<input type="text" name="guest\_name" id="guest\_name"><br>

<label for="notification\_text">Notification</label>

<textarea rows="4" cols="50" name="notification\_text"></textarea><br>

<label for="priority">Priority</label>

<select name="priority" id="priority">

{% for priority in list\_of\_priorities %}

<option value="{{ priority.code }}" {{ 'selected' if priority.selected}}>{{ priority.description }}</option>

{% endfor %}

</select><br>

<input type="submit" value="Send">

</form>

</body>

</html>

notification\_confirmation.html:

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

{% if priority\_type.code=='high' %}

<h1>Critical notification content</h1>

{% elif priority\_type.code=='medium' %}

<h1>Important notification content</h1>

{% else %}

<h1>Notification content</h1>

{% endif %}

<ul>

<li style="font-weight: bold;">Room number</li>

<li>{{ room\_number }}</li>

<li style="font-weight: bold;">Guest name</li>

<li>{{ guest\_name }}</li>

<li style="font-weight: bold;">Notification</li>

<li>{{ notification\_text }}</li>

</ul>

</body>

</html>

Funkcja flash() – czyli krótkie info o statusie

Propozycja rozwiązania

# in the app.py

from flask import flash

from datetime import datetime

# …

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

app.config['SECRET\_KEY'] = '123GoniszTy!'

# …

# somewhere in the function notification()

flash('Notification has been sent')

the\_hour = datetime.now().hour

raise\_priority = (the\_hour >= 20 or the\_hour < 6) and priority == 'medium'

if raise\_priority:

priority = 'high'

flash('Rising priority from medium to high')

# in the nothification\_content.html

{% for message in get\_flashed\_messages() %}

<div>{{ message }}</dev>

{% endfor %}

Makra Jinja

Propozycja rozwiązania

# file macros.html

{% macro show\_flash() %}

<ul>

{% for message in get\_flashed\_messages() %}

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

{% endfor %}

</ul>

{% endmacro %}

# file notification\_content.html

{% from "macros.html" import show\_flash %}

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

{{ show\_flash() }}

{% if priority\_type.code=='high' %}

<h1>Critical notification content</h1>

{% elif priority\_type.code=='medium' %}

<h1>Important notification content</h1>

{% else %}

<h1>Notification content</h1>

{% endif %}

<ul>

<li style="font-weight: bold;">Room number</li>

<li>{{ room\_number }}</li>

<li style="font-weight: bold;">Guest name</li>

<li>{{ guest\_name }}</li>

<li style="font-weight: bold;">Notification</li>

<li>{{ notification\_text }}</li>

</ul>

</body>

</html>

Dziedziczenie w szablonach i budowanie własnych standardów

Propozycja rozwiązania

# base.html :

{% from "macros.html" import show\_flash %}

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

{{ show\_flash() }}

{% block page\_body %}

{% endblock %}

</body>

</html>

# notification.html :

{% extends "base.html" %}

{% block page\_body %}

<form id="notification\_form" method="POST" action="{{ url\_for('notification') }}">

<label for="room\_number">Room number</label>

<input type="text" name="room\_number" id="room\_number"><br>

<label for="guest\_name">Guest name</label>

<input type="text" name="guest\_name" id="guest\_name"><br>

<label for="notification\_text">Notification</label>

<textarea rows="4" cols="50" name="notification\_text"></textarea><br>

<label for="priority">Priority</label>

<select name="priority" id="priority">

{% for priority in list\_of\_priorities %}

<option value="{{ priority.code }}" {{ 'selected' if priority.selected}}>{{ priority.description }}</option>

{% endfor %}

</select><br>

<input type="submit" value="Send">

</form>

{% endblock %}

# notification\_content.html

{% extends "base.html" %}

{% block page\_body %}

{% if priority\_type.code=='high' %}

<h1>Critical notification content</h1>

{% elif priority\_type.code=='medium' %}

<h1>Important notification content</h1>

{% else %}

<h1>Notification content</h1>

{% endif %}

<ul>

<li style="font-weight: bold;">Room number</li>

<li>{{ room\_number }}</li>

<li style="font-weight: bold;">Guest name</li>

<li>{{ guest\_name }}</li>

<li style="font-weight: bold;">Notification</li>

<li>{{ notification\_text }}</li>

</ul>

{% endblock %}

Dołączanie szablonów Jinja - include

Propozycja rozwiązania

# file footer.html

<div style="text-align: center; border-style: dotted; font-weight: bold;">  
 Py-Hotel

</div>

# file base.html

{% from "macros.html" import show\_flash %}

<!DOCTYPE html>

<html>

<head>

<title>Hotel 101</title>

</head>

<body>

{{ show\_flash() }}

{% block page\_body %}

{% endblock %}

{% include 'footer.html' %}

</body>

</html>

Flask Bootstrap

Propozycja rozwiązania

# Przygotowanie środowiska:

Python -m venv venv

.\venv\scripts\activate.bat

Pip install flask

Pip install flask-bootstrap

SET FLASK\_ENV=development

# Kod aplikacji

from flask import Flask, render\_template

from flask\_bootstrap import Bootstrap

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

bootstrap = Bootstrap(app)

@app.route('/')

def index():

return render\_template('index.html')

# Kod szablonu index.html

{% extends "bootstrap/base.html" %}

{%block content %}

<h1>Example heading <span class="badge bg-secondary">New</span></h1>

<h2>Example heading <span class="badge bg-secondary">New</span></h2>

<h3>Example heading <span class="badge bg-secondary">New</span></h3>

<h4>Example heading <span class="badge bg-secondary">New</span></h4>

<h5>Example heading <span class="badge bg-secondary">New</span></h5>

<h6>Example heading <span class="badge bg-secondary">New</span></h6>

{% endblock %}

Korzystanie z Bootstrap bez Flask Bootstrap

Propozycja rozwiązania

# File base.html

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">

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

</head>

<body>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous"></script>

</body>

</html>

# file index.html

{% extends "base.html" %}

{% block title %} Demo page for bootstrap in flask {% endblock %}

{%block content %}

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#staticBackdrop">

Launch static backdrop modal

</button>

<!-- Modal -->

<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

...

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary">Understood</button>

</div>

</div>

</div>

</div>

{% endblock %}

Połącznie kodu z szablonami

Propozycja rozwiązania

# menu.html

<ul class="nav nav-pills">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{{ url\_for('index') }}">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Notifications</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{{ url\_for('notification') }}">New notification</a>

</li>

<li class="nav-item">

<a class="nav-link" href="{{ url\_for('about') }}" tabindex="-1">About this app</a>

</li>

</ul>

# base.html

…

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

…

{% include 'menu.html' %}

{% block content %} {% endblock %}

…

# index.html

{% extends "base.html" %}

{% block title %} Hotel notifications {% endblock %}

{%block content %}

<div id="carouselExampleIndicators" class="carousel slide" data-bs-ride="carousel">

<div class="carousel-indicators">

<button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>

<button type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide-to="1" aria-label="Slide 2"></button>

</div>

<div class="carousel-inner">

<div class="carousel-item active">

<div class="d-flex justify-content-center">

<img src="{{ url\_for('static', filename='carousel/1.jpg') }}" class="d-block" alt="..." style="width: 90%;">

</div>

</div>

<div class="carousel-item">

<div class="d-flex justify-content-center">

<img src="{{ url\_for('static', filename='carousel/2.jpg') }}" class="d-block" alt="..." style="width: 90%;">

</div>

</div>

</div>

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Previous</span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleIndicators" data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Next</span>

</button>

</div>

{% endblock %}

# app.py

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

from datetime import datetime

import os

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

app.config['SECRET\_KEY'] = '123GoniszTy!'

class PriorityType:

…

class NotificationPriorities:

…

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

def notification():

…

@app.route('/about')

def about():

return render\_template('about.html')

@app.route('/')

def index():

return render\_template('index.html')

Stosowanie formatów bootsrap

Propozycja rozwiązania

# form.html – only the modified part

<form id="notification\_form" method="POST" action="{{ url\_for('notification') }}">

<div class="container px-4">

<div class="row mb-3">

<div class="col-12"><h2>Notify shift manager:</h2></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="room\_number" class="form-label">Room number</label></div>

<div class="col-6"><input type="text" name="room\_number" id="room\_number" class="form-control"></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="guest\_name" class="form-label">Guest name</label></div>

<div class="col-6"><input type="text" name="guest\_name" id="guest\_name" class="form-control"></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="notification\_text" class="form-label">Notification</label></div>

<div class="col-6"><textarea rows="4" cols="50" name="notification\_text" class="form-control"></textarea></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="priority" class="form-label">Priority</label></div>

<div class="col-6">

<select name="priority" id="priority" class="form-select">

{% for priority in list\_of\_priorities %}

<option value="{{ priority.code }}" {{ 'selected' if priority.selected}}>{{ priority.description }}</option>

{% endfor %}

</select>

</div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"></div>

<div class="col-6"><input type="submit" value="Send" class="btn btn-primary"></div>

<div class="col-3"></div>

</div>

</div>

</form>

# notification\_confirmation.html

<div class="container px-4">

<div class="row mb-3">

<div class="col-12">

<h2>

{% if priority\_type.code=='high' %}

Critical notification content

{% elif priority\_type.code=='medium' %}

Important notification content

{% else %}

Notification content

{% endif %}

</h2>

</div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="room\_number" class="form-label">Room number</label></div>

<div class="col-6"><input type="text" name="room\_number" id="room\_number" class="form-control" value="{{ room\_number }}" readonly></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="guest\_name" class="form-label">Guest name</label></div>

<div class="col-6"><input type="text" name="guest\_name" id="guest\_name" class="form-control" value="{{ guest\_name }}" readonly></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="notification\_text" class="form-label">Notification</label></div>

<div class="col-6">

<textarea rows="4" cols="50" name="notification\_text" class="form-control">

{{ notification\_text }}

</textarea>

</div>

<div class="col-3"></div>

</div>

</div>

Projekt 02 – Formularz z pomysłami wycieczek

def read\_csv\_data():

    full\_file\_path = os.path.join(app.static\_folder, 'trips.txt')

    fieldnames = ['trip\_name', 'email', 'description', 'completness', 'contact\_ok']

    entries = []

    with open(full\_file\_path, mode='r', encoding="utf-8") as f:

        csv\_reader = csv.DictReader(f, fieldnames=fieldnames)

        line\_count = 0

        for row in csv\_reader:

            if line\_count == 0:

                line\_count += 1

            else:

                entries.append(row)

            line\_count += 1

    return entries

def append\_csv\_data(data):

    full\_file\_path = os.path.join(app.static\_folder, 'trips.txt')

    fieldnames = ['trip\_name', 'email', 'description', 'completness', 'contact\_ok']

    if not os.path.exists(full\_file\_path):

        with open(full\_file\_path, 'w+', newline='',encoding="utf-8") as f:

                writer = csv.DictWriter(f, fieldnames=fieldnames)

                writer.writeheader()

    with open(full\_file\_path, 'a', newline='',encoding="utf-8") as f:

        writer = csv.DictWriter(f, fieldnames=fieldnames)

        writer.writerow(data)

Propozycja rozwiązania (fragmenty)

Przyjmowanie parametrów formularza:

    if request.method == 'GET':

        return render\_template('new\_trip.html',

            trip\_name=trip\_name, email=email, description=description, completness=completness, conctact\_ok=contact\_ok)

    else:

        trip\_name = '' if 'trip\_name' not in request.form else request.form['trip\_name']

        email = '' if 'email' not in request.form else request.form['email']

        description = '' if 'description' not in request.form else request.form['description']

        completness = False if 'completness' not in request.form else request.form['completness']=='yes'

        contact\_ok = False if 'contact\_ok' not in request.form else True

Kod formularza przyjmującego pomysły:

<form id="trip\_form" method="POST" action="{{ url\_for('new\_trip') }}"

<div class="container">

    <div class="row mb-3">

      <div class="col-1">

      </div>

      <div class="col-10">

        <div class="form-group row mb-3">

            <label for="trip\_name" class="col-sm-4 col-form-label">Trip name</label>

            <div class="col-sm-8">

              <input type="text" class="form-control" id="trip\_name" name="trip\_name" value="{{trip\_name}}">

            </div>

          </div>

          <div class="form-group row mb-3">

            <label for="email" class="col-sm-4 col-form-label">Your email</label>

            <div class="col-sm-8">

              <input type="email" class="form-control" id="email" name="email" value="{{email}}">

            </div>

          </div>

          <div class="form-group row mb-3">

            <label for="description" class="col-sm-4 col-form-label">Short description</label>

            <div class="col-sm-8">

              <textarea rows="4" cols="50" class="form-control" id="description" name="description">{{description}}</textarea><br>

            </div>

          </div>

          <fieldset class="form-group mb-3">

            <div class="row">

              <legend class="col-form-label col-sm-4 pt-0">Completness</legend>

              <div class="col-sm-8">

                <div class="form-check">

                  <input class="form-check-input" type="radio" name="completness" id="completness\_1" value="yes" {{'checked' if completness else ''}}>

                  <label class="form-check-label" for="completness\_1">

                    Yes - the idea is complete including price proposal

                  </label>

                </div>

                <div class="form-check">

                  <input class="form-check-input" type="radio" name="completness" id="completness\_2" value="no" {{'checked' if not completness}}>

                  <label class="form-check-label" for="completness\_2">

                    No - this is just pure idea

                  </label>

                </div>

              </div>

            </div>

          </fieldset>

          <div class="form-group row mb-3">

            <div class="col-sm-4">May we contact you for details?</div>

            <div class="col-sm-8">

              <div class="form-check">

                <input class="form-check-input" type="checkbox" id="contact\_ok" name="contact\_ok" value="yes" {{'checked' if contact\_ok}}>

                <label class="form-check-label" for="contact\_ok">

                  Yes, I agree

                </label>

              </div>

            </div>

          </div>

          <div class="form-group row mb-3">

            <div class="col-sm-12">

              <button type="submit" class="btn btn-success btn-block">Send proposal</button>

            </div>

          </div>

        </div>

        <div class="col-1">

        </div>

      </div>

    </div>

</form>

Instalacja SQLite 3 i podstawy pracy z danymi

Propozycja rozwiązania

sqlite3 notification.db

sqlite> create table notifications(id integer primary key autoincrement, room\_number varchar(10), guest\_name varchar(30), notification text, priority varchar(20));

sqlite> insert into notifications(room\_number, guest\_name, notification, priority) values('10A', 'Frederico Romane', 'The window cannnot be open', 'MEDIUM');

sqlite>.help

sqlite> .header on

sqlite> select \* from notifications;

id|room\_number|guest\_name|notification|priority

1|10A|Frederico Romane|The window cannnot be open|MEDIUM

.quit

Zapis danych z rekordu do bazy danych

Propozycja rozwiązania

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

from datetime import datetime

import sqlite3

import os

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

app.config['SECRET\_KEY'] = '123GoniszTy!'

app\_info = { ‘db\_file' : r"D:\FLASK\hotel\data\notification.db" }

def get\_db():

if not hasattr(g, 'sqlite\_db'):

conn = sqlite3.connect(app\_info['db\_file'])

conn.row\_factory = sqlite3.Row

g.sqlite\_db = conn

return g.sqlite\_db

@app.teardown\_appcontext

def close\_db(error):

if hasattr(g, 'sqlite\_db'):

g.sqlite\_db.close()

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

def notification():

notification\_priorities = NotificationPriorities()

notification\_priorities.load\_priorities()

if request.method == 'GET':

return render\_template('notification.html',

list\_of\_priorities=notification\_priorities.list\_of\_priorities)

else:

room\_number = request.form['room\_number'] if 'room\_number' in request.form else ''

guest\_name = request.form['guest\_name'] if 'guest\_name' in request.form else ''

notification\_text = request.form['notification\_text'] if 'notification\_text' in request.form else ''

priority = request.form['priority'] if 'priority' in request.form else 'normal'

priority\_type = notification\_priorities.get\_priority\_by\_code(priority)

print('found', priority\_type.code)

flash('Notification has been sent')

the\_hour = datetime.now().hour

raise\_priority = (the\_hour >= 20 or the\_hour < 10) and priority == 'medium'

if raise\_priority:

priority = 'high'

flash('Rising priority from medium to high')

db = get\_db()

sql\_command = 'insert into notifications(room\_number, guest\_name, notification, priority) values(?, ?, ?, ?)'

db.execute(sql\_command, [room\_number, guest\_name, notification\_text, priority])

db.commit()

return render\_template('notification\_content.html',

room\_number=room\_number, guest\_name=guest\_name,

notification\_text=notification\_text, priority\_type=priority\_type)

Pobieranie rekordów z bazy danych

Propozycja rozwiązania

# app.py – only the new notification function (add active\_menu param to each render template):

@app.route('/notifications')

def notifications():

db = get\_db()

sql\_command = 'select id, room\_number, guest\_name, notification, priority from notifications;'

cur = db.execute(sql\_command)

notifications = cur.fetchall()

return render\_template('notifications.html', active\_menu='notifications', notifications=notifications)

# notifications.html

{% extends "base.html" %}

{% block title %}

Current notifications

{% endblock %}

{% block content %}

<div class="container">

<table class="table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">Room Number</th>

<th scope="col">Guest Name</th>

<th scope="col">Notification</th>

<th scope="col">Priority</th>

<th scope="col">Operations</th>

</tr>

</thead>

<tbody>

{% for n in notifications %}

<tr>

<th scope="row">{{ n.id }}</th>

<td>{{ n.room\_number }}</td>

<td>{{ n.guest\_name }}</td>

<td>{{ n.notification }}</td>

<td>{{ n.priority }}</td>

<td>

<a href="#" class="btn btn-primary btn-sm" role="button">Actions...</a>

<a href="#" class="btn btn-success btn-sm" role="button">Edit...</a>

<a href="#" class="btn btn-danger btn-sm" role="button">Delete...</a>

</td>

</tr>

{%endfor%}

</tbody>

</table>

</div>

{% endblock %}

# menu.html

<ul class="nav nav-pills">

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='index' }}" aria-current="page" href="{{ url\_for('index') }}">Home</a>

</li>

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='notifications' }}" href="{{ url\_for('notifications') }}">Notifications</a>

</li>

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='notification' }}" href="{{ url\_for('notification') }}">New notification</a>

</li>

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='about' }}" href="{{ url\_for('about') }}" tabindex="-1">About this app</a>

</li>

</ul>

Kasowanie rekordu z pytaniem o potwierdzenie

Propozycja rozwiązania

# file notifications.html – only fragment with buttons:

<a href="#" class="btn btn-primary btn-sm" role="button">Actions...</a>

<a href="#" class="btn btn-success btn-sm" role="button">Edit...</a>

<a type="button" class="btn btn-danger btn-sm delete-confirm"

data-bs-toggle="modal" data-bs-target="#confirmDeleteModal"

data-desc="{{ 'Delete notification for {} {}?'.format(n.room\_number, n.guest\_name) }}"

data-url="{{ url\_for('delete\_notification’, notification\_id=n.id) }}">

Delete

</a>

# file app.py – only method deleting a row

@app.route('/delete\_notification/<int:notification\_id>')

def delete\_notification(notification\_id):

db = get\_db()

sql\_statement = 'delete from notifications where id = ?;'

db.execute(sql\_statement, [notification\_id])

db.commit()

return redirect(url\_for(‘notifications’))

Modyfikacja danych

Propozycja rozwiązania

# app.py – only the edit\_notification function:

@app.route('/edit\_notification/<int:notification\_id>', methods=['GET', 'POST'])

def edit\_notification(notification\_id):

db = get\_db()

notification\_priorities = NotificationPriorities()

notification\_priorities.load\_priorities()

if request.method == 'GET':

sql\_statement = 'select id, room\_number, guest\_name, notification, priority from notifications where id = ?'

cur = db.execute(sql\_statement, [notification\_id])

notif\_obj = cur.fetchone()

if notif\_obj == None:

flash('No such notification...')

return redirect('notifications')

else:

return render\_template('edit\_notification.html', active\_menu='notifications', notif\_obj=notif\_obj,

list\_of\_priorities=notification\_priorities.list\_of\_priorities)

else:

room\_number = request.form['room\_number'] if 'room\_number' in request.form else ''

guest\_name = request.form['guest\_name'] if 'guest\_name' in request.form else ''

notification\_text = request.form['notification\_text'] if 'notification\_text' in request.form else ''

priority = request.form['priority'] if 'priority' in request.form else 'normal'

priority\_type = notification\_priorities.get\_priority\_by\_code(priority)

sql\_command = '''update notifications set room\_number = ?, guest\_name = ?,

notification = ?, priority = ? where id = ? '''

db.execute(sql\_command, [room\_number, guest\_name, notification\_text, priority, notification\_id])

db.commit()

flash('Notification has been updated')

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

# notifications.html – how to start edit action:

<a href="{{ url\_for('edit\_notification', notification\_id=n.id) }}" class="btn btn-success btn-sm" role="button">Edit...</a>

# edit\_notification.html

{% extends "base.html" %}

{% block title %} Hotel notifications {% endblock %}

{%block content %}

<form id="notification\_form" method="POST" action="{{ url\_for('edit\_notification', notification\_id=notif\_obj.id) }}">

<div class="container px-4">

<div class="row mb-3">

<div class="col-12"><h2>Edit notification details:</h2></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="room\_number" class="form-label">Room number</label></div>

<div class="col-6"><input type="text" name="room\_number" id="room\_number" class="form-control"

value="{{ notif\_obj.room\_number }}"></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="guest\_name" class="form-label">Guest name</label></div>

<div class="col-6"><input type="text" name="guest\_name" id="guest\_name" class="form-control"

value="{{ notif\_obj.guest\_name }}"></div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="notification\_text" class="form-label">Notification</label></div>

<div class="col-6">

<textarea rows="4" cols="50" name="notification\_text" class="form-control">{{ notif\_obj.notification }}</textarea>

</div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"><label for="priority" class="form-label">Priority</label></div>

<div class="col-6">

<select name="priority" id="priority" class="form-select">

{% for priority in list\_of\_priorities %}

<option value="{{ priority.code }}" {{ 'selected' if priority.code==notif\_obj.priority }}>{{ priority.description }}</option>

{% endfor %}

</select>

</div>

<div class="col-3"></div>

</div>

<div class="row mb-3">

<div class="col-3"></div>

<div class="col-6"><input type="submit" value="Send" class="btn btn-primary"></div>

<div class="col-3"></div>

</div>

</div>

</form>

{% endblock %}

Użytkownicy aplikacji - przygotowanie

Propozycja rozwiązania

# SQL to create a table

CREATE TABLE users(

id integer primary key autoincrement,

name varchar(100) not null unique,

email varchar(100) not null unique,

password text,

is\_active boolean not null default 0,

is\_admin boolean not null default 0

);

# app.py - import statements

import random

import string

import hashlib

import binascii

# app.py – helper UserPass class

class UserPass:

def \_\_init\_\_(self, user='', password=''):

self.user = user

self.password = password

def hash\_password(self):

"""Hash a password for storing."""

# the value generated using os.urandom(60)

os\_urandom\_static = b"ID\_\x12p:\x8d\xe7&\xcb\xf0=H1\xc1\x16\xac\xe5BX\xd7\xd6j\xe3i\x11\xbe\xaa\x05\xccc\xc2\xe8K\xcf\xf1\xac\x9bFy(\xfbn.`\xe9\xcd\xdd'\xdf`~vm\xae\xf2\x93WD\x04"

salt = hashlib.sha256(os\_urandom\_static).hexdigest().encode('ascii')

pwdhash = hashlib.pbkdf2\_hmac('sha512', self.password.encode('utf-8'), salt, 100000)

pwdhash = binascii.hexlify(pwdhash)

return (salt + pwdhash).decode('ascii')

def verify\_password(self, stored\_password, provided\_password):

"""Verify a stored password against one provided by user"""

salt = stored\_password[:64]

stored\_password = stored\_password[64:]

pwdhash = hashlib.pbkdf2\_hmac('sha512', provided\_password.encode('utf-8'), salt.encode('ascii'), 100000)

pwdhash = binascii.hexlify(pwdhash).decode('ascii')

return pwdhash == stored\_password

def get\_random\_user\_pasword(self):

random\_user = ''.join(random.choice(string.ascii\_lowercase)for i in range(3))

self.user = random\_user

password\_characters = string.ascii\_letters #+ string.digits + string.punctuation

random\_password = ''.join(random.choice(password\_characters)for i in range(3))

self.password = random\_password

# app.py – init route and function

@app.route('/init\_app')

def init\_app():

# check if there are users defined (at least one active admin required)

db = get\_db()

sql\_statement = 'select count(\*) as cnt from users where is\_active and is\_admin;'

cur = db.execute(sql\_statement)

active\_admins = cur.fetchone()

if active\_admins!=None and active\_admins['cnt']>0:

flash('Application is already set-up. Nothing to do')

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

# if not - create/update admin account with a new password and admin privileges, display random username

user\_pass = UserPass()

user\_pass.get\_random\_user\_pasword()

sql\_statement = '''insert into users(name, email, password, is\_active, is\_admin)

values(?,?,?,True, True);'''

db.execute(sql\_statement, [user\_pass.user, 'noone@nowhere.no', user\_pass.hash\_password()])

db.commit()

flash('User {} with password {} has been created'.format(user\_pass.user, user\_pass.password))

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

Sesja w akcji – logowanie i wylogowanie użytkownika

Propozycja rozwiązania

# app.py

from flask import session

class UserPass:

# … - the code as in the previous version

def login\_user(self):

db = get\_db()

sql\_statement = 'select id, name, email, password, is\_active, is\_admin from users where name=?'

cur = db.execute(sql\_statement, [self.user])

user\_record = cur.fetchone()

if user\_record != None and self.verify\_password(user\_record['password'], self.password):

return user\_record

else:

self.user = None

self.password = None

return None

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

def login():

if request.method == 'GET':

return render\_template('login.html', active\_menu='login')

else:

user\_name = '' if 'user\_name' not in request.form else request.form['user\_name']

user\_pass = '' if 'user\_pass' not in request.form else request.form['user\_pass']

login = UserPass(user\_name, user\_pass)

login\_record = login.login\_user()

if login\_record != None:

session['user'] = user\_name

flash('Logon succesfull, welcome {}'.format(user\_name))

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

else:

flash('Logon failed, try again')

return render\_template('login.html')

@app.route('/logout')

def logout():

if 'user' in session:

session.pop('user', None)

flash('You are logged out')

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

# login.html

{% extends "base.html" %}

{% block title %}Logon{% endblock %}

{% block content %}

<div class="container">

<form method="POST" action={{url\_for('login')}}>

<div class="row mt-3">

<div class="col-6 h1">Login</div>

</div>

<div class="row mt-3">

<div class="col-2 col-form-label"><label for="user\_name">User name</label></div>

<div class="col-4">

<input type="text" id="user\_name" name="user\_name" class="form-control">

</div>

</div>

<div class="row mt-3">

<div class="col-2 col-form-label"><label for="user\_pas">Password</label></div>

<div class="col-4">

<input type="password" id="user\_pass" name="user\_pass" class="form-control">

</div>

</div>

<div class="row mt-3">

<div class="col-2"></div>

<div class="col-4"><input type="submit" value="Login" class="btn btn-primary"></div>

</div>

</form>

</div>

{% endblock %}

# menu.html

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='login' }}" href="{{ url\_for('login') }}" tabindex="-1">Login</a>

</li>

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='logout' }}" href="{{ url\_for('logout') }}" tabindex="-1">

Logout {{ session['user'] if 'user' in session }}</a>

</li>

Dodawanie użytkowników – zadbaj o wygodę użytkownika

Propozycja rozwiązania

# to add to app.py

@app.route('/users')

def users():

return 'not implemented'

@app.route('/user\_status\_change/<action>/<user\_name>')

def user\_status\_change(action, user\_name):

return 'not implemented'

@app.route('/edit\_user/<user\_name>', methods=['GET', 'POST'])

def edit\_user(user\_name):

return 'not implemented'

@app.route('/user\_delete/<user\_name>')

def delete\_user(user\_name):

return 'not implemented'

@app.route('/new\_user', methods=['GET', 'POST'])

def new\_user():

if not 'user' in session:

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

login = session['user']

db = get\_db()

message = None

user = {}

if request.method =='GET':

return render\_template('new\_user.html', active\_menu='users', user=user)

else:

user['user\_name'] = '' if not 'user\_name' in request.form else request.form['user\_name']

user['email'] = '' if not 'email' in request.form else request.form['email']

user['user\_pass'] = '' if not 'user\_pass' in request.form else request.form['user\_pass']

cursor = db.execute('select count(\*) as cnt from users where name = ?', [user['user\_name']])

record = cursor.fetchone()

is\_user\_name\_unique = (record['cnt'] == 0)

cursor = db.execute('select count(\*) as cnt from users where email = ?', [user['email']])

record = cursor.fetchone()

is\_user\_email\_unique = (record['cnt'] == 0)

if user['user\_name'] == '':

message = 'Name cannot be empty'

elif user['email'] == '':

message = 'email cannot be empty'

elif user['user\_pass'] == '':

message = 'Password cannot be empty'

elif not is\_user\_name\_unique:

message = 'User with the name {} already exists'.format(user['user\_name'])

elif not is\_user\_email\_unique:

message = 'User with the email {} alresdy exists'.format(user['email'])

if not message:

user\_pass = UserPass(user['user\_name'], user['user\_pass'])

password\_hash = user\_pass.hash\_password()

sql\_statement = '''insert into users(name, email, password, is\_active, is\_admin)

values(?,?,?, True, False);'''

db.execute(sql\_statement, [user['user\_name'], user['email'], password\_hash])

db.commit()

flash('User {} created'.format(user['user\_name']))

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

else:

flash('Correct error: {}'.format(message))

return render\_template('new\_user.html', active\_menu='users', user=user)

# to add to menu.html

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"

data-bs-toggle="dropdown" aria-expanded="false">

Users

</a>

<ul class="dropdown-menu" aria-labelledby="navbarDropdown">

<li><a class="dropdown-item" href="{{ url\_for('users') }}">Users</a></li>

<li><a class="dropdown-item" href="{{ url\_for('new\_user') }}">New user</a></li>

</ul>

</li>

# new\_user.html

{% extends "base.html" %}

{% block content %}

<div class="container">

<div class="row mb-3">

<div class="col-5 h1">New user</div>

</div>

<form id="user\_form" action="{{ url\_for('new\_user') }}" method="POST">

<div class="row mb-3">

<div class="col-1 col-form-label"><label for="user\_name">User name</label></div>

<div class="col-4">

<input type="text" id="user\_name" name="user\_name" value="{{ user.user\_name }}" class="form-control"><br>

</div>

</div>

<div class="row mb-3">

<div class="col-1 col-form-label"><label for="email">Email</label></div>

<div class="col-4">

<input type="text" id="email" name="email" value="{{ user.email }}" class="form-control"><br>

</div>

</div>

<div class="row mb-3">

<div class="col-1 col-form-label"><label for="user\_pass">Password</label></div>

<div class="col-4">

<input type="password" id="user\_pass" name="user\_pass" class="form-control"><br>

</div>

</div>

<div class="row mb-3">

<div class="col-1"></div>

<div class="col-2"><input type="submit" value="Send" class="btn btn-primary"></div>

</div>

</form>

</div>

{% endblock %}

Lista użytkowników

Propozycja rozwiązania

# users.html – the most important points only

{% block content %}

<!-- Modal -->

<div class="modal fade" id="confirmDeleteModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">This entry will be deleted:</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body" id="idDeleteModalBody">

...

</div>

<div class="modal-footer">

<form action="" method="GET" id="confirmDeleteForm">

<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

<button type="submit" class="btn btn-danger">Delete</button>

</form>

</div>

</div>

</div>

</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script>

$(document).ready(function () {

// For A Delete Record Popup

// This function is applied to all elements with class ending with ".delete-confirm"

$('.delete-confirm').click(function () {

// get attributes of the found element

var desc = $(this).attr('data-desc');

var url = $(this).attr('data-url');

// the #... designates id of an element - change the text in the modal window

$('#idDeleteModalBody').text(desc);

$("#confirmDeleteForm").attr("action", url);

});

});

</script>

<div class="container">

<table class="table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">User name</th>

<th scope="col">Email</th>

<th scope="col">Is active</th>

<th scope="col">Is admin</th>

<th scope="col">Actions</th>

</tr>

</thead>

<tbody>

{% for user in users %}

<tr>

<th scope="row">{{ user.id }}</th>

<td>{{ user.name }}</td>

<td>{{ user.email }}</td>

<td> </td>

<td> </td>

<td>

<a href="{{ url\_for('edit\_user', user\_name=user.name) }}"

class="btn btn-success btn-sm" role="button">Edit...</a>

<a type="button" class="btn btn-danger btn-sm delete-confirm"

data-bs-toggle="modal" data-bs-target="#confirmDeleteModal"

data-desc="{{ 'Delete user {}?'.format(user.name) }}"

data-url="{{ url\_for('delete\_user', user\_name=user.name) }}">

Delete

</a>

</td>

</tr>

{%endfor%}

</tbody>

</table>

</div>

{% endblock %}

# app.py – only the most important functions:

@app.route('/users')

def users():

db = get\_db()

sql\_command = 'select id, name, email, is\_admin, is\_active from users;'

cur = db.execute(sql\_command)

users = cur.fetchall()

return render\_template('users.html', active\_menu='users', users=users)

@app.route('/user\_delete/<user\_name>')

def delete\_user(user\_name):

if not 'user' in session:

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

login = session['user']

db = get\_db()

sql\_statement = "delete from users where name = ? and name <> ?"

db.execute(sql\_statement, [user\_name, login])

db.commit()

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

Edycja użytkownika

Propozycja rozwiązania

# app.py – only modified function

@app.route('/edit\_user/<user\_name>', methods=['GET', 'POST'])

def edit\_user(user\_name):

db = get\_db()

cur = db.execute('select name, email from users where name = ?', [user\_name])

user = cur.fetchone()

message = None

if user == None:

flash('No such user')

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

if request.method == 'GET':

return render\_template('edit\_user.html', active\_menu='users', user=user)

else:

new\_email = '' if 'email' not in request.form else request.form["email"]

new\_password = '' if 'user\_pass' not in request.form else request.form['user\_pass']

if new\_email != user['email']:

sql\_statement = "update users set email = ? where name = ?"

db.execute(sql\_statement, [new\_email, user\_name])

db.commit()

flash('Email was changed')

if new\_password != '':

user\_pass = UserPass(user\_name, new\_password)

sql\_statement = "update users set password = ? where name = ?"

db.execute(sql\_statement, [user\_pass.hash\_password(), user\_name])

db.commit()

flash('Password was changed')

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

# edit\_user.html

{% extends "base.html" %}

{% block title %}

Edit user

{% endblock %}

{% block content %}

<div class="container">

<div class="row mb-3">

<div class="col-5 h1">Edit user {{ user.name }}</div>

</div>

<form id="user\_form" action="{{ url\_for('edit\_user', user\_name=user.name) }}" method="POST">

<div class="row mb-3">

<div class="col-1 col-form-label"><label for="email">Email</label></div>

<div class="col-4">

<input type="text" id="email" name="email" value="{{ user.email }}" class="form-control"><br>

</div>

</div>

<div class="row mb-3">

<div class="col-1 col-form-label"><label for="user\_pass">Password</label></div>

<div class="col-4">

<input type="password" id="user\_pass" name="user\_pass" class="form-control"><br>

</div>

</div>

<div class="row mb-3">

<div class="col-1"></div>

<div class="col-2"><input type="submit" value="Send" class="btn btn-primary"></div>

</div>

</form>

</div>

{% endblock %}

Edycja uprawnień

Propozycja rozwiązania

# app.py – only one function:

@app.route('/user\_status\_change/<action>/<user\_name>')

def user\_status\_change(action, user\_name):

if not 'user' in session:

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

login = session['user']

db = get\_db()

if action == 'active':

db.execute("""update users set is\_active = (is\_active + 1) % 2

where name = ? and name <> ?""",

[user\_name, login])

db.commit()

elif action == 'admin':

db.execute("""update users set is\_admin = (is\_admin + 1) % 2

where name = ? and name <> ?""",

[user\_name, login])

db.commit()

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

# users.html – missing part of the table

<td>

<a href="{{ url\_for('user\_status\_change', action='active', user\_name=user.name) }}">

{% if user.is\_active %}

&check;

{% else %}

&#x25a2;

{% endif %}

</a>

</td>

<td>

<a href="{{ url\_for('user\_status\_change', action='admin', user\_name=user.name) }}">

{% if user.is\_admin %}

&check;

{% else %}

&#x25a2;

{% endif %}

</a>

</td>

Implementacja uprawnień w aplikacji

Propozycja rozwiązania

# app.py – properties and method for UserPass class:

class UserPass:

def \_\_init\_\_(self, user='', password=''):

self.user = user

self.password = password

self.email = ''

self.is\_valid = False

self.is\_admin = False

def get\_user\_info(self):

db = get\_db()

sql\_statement = 'select name, email, is\_active, is\_admin from users where name=?'

cur = db.execute(sql\_statement, [self.user])

db\_user = cur.fetchone()

if db\_user == None:

self.is\_valid = False

self.is\_admin = False

self.email = ''

elif db\_user['is\_active']!=1:

self.is\_valid = False

self.is\_admin = False

self.email = db\_user['email']

else:

self.is\_valid = True

self.is\_admin = db\_user['is\_admin']

self.email = db\_user['email']

# app.py – code to add to functions – non admin access:

login = UserPass(session.get('user'))

login.get\_user\_info()

if not login.is\_valid:

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

# app.py – code to add to functions – admin access

login = UserPass(session.get('user'))

login.get\_user\_info()

if not login.is\_valid or not login.is\_admin:

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

# menu.html – code to show/hide different menu options (only 2 examples):

{% if login.is\_valid: %}

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='notification' }}" href="{{ url\_for('notification') }}">New notification</a>

</li>

{% endif %}

<li class="nav-item">

<a class="nav-link {{ 'active' if active\_menu=='about' }}" href="{{ url\_for('about') }}" tabindex="-1">About this app</a>

</li>

{% if login.is\_valid and login.is\_admin: %}

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"

data-bs-toggle="dropdown" aria-expanded="false">

Users

</a>

<ul class="dropdown-menu" aria-labelledby="navbarDropdown">

<li><a class="dropdown-item" href="{{ url\_for('users') }}">Users</a></li>

<li><a class="dropdown-item" href="{{ url\_for('new\_user') }}">New user</a></li>

</ul>

</li>

{% endif %}

Flask SQLAlchemy

Propozycja rozwiązania

#app.py

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

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

app.config.from\_pyfile('config.cfg')

db = SQLAlchemy(app)

class Author(db.Model):

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

name = db.Column(db.String(50))

special = db.Column(db.Boolean)

@app.route('/')

def index():

db.create\_all()

return '''Hello Flask-SQLAlchemy

'''

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

app.run()

# config.cfg

SQLALCHEMY\_DATABASE\_URI='sqlite:///c:\\data\\museum.db'

SQLALCHEMY\_TRACK\_MODIFICATIONS=False

Dodawanie I pobieranie danych z bazy danych

Propozycja rozwiązania

# app.py – class definition

class Author(db.Model):

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

name = db.Column(db.String(50))

special = db.Column(db.Boolean)

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

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

# in python session:

>>> from app import app, db, Author

>>> dali = Author(id=1, name='Salvador Dali', special=False)

>>> db.session.add(dali)

>>> picasso = Author(id=2, name='Pablo Picasso', special=False)

>>> db.session.add(picasso)

>>> cezane = Author(id=3, name='Paul Cezane', special=True)

>>> db.session.add(cezane)

>>> db.session.commit()

>>> Author.query.all()

[<id: 1, name: Salvador Dali, special: False>, <id: 2, name: Pablo Picasso, special: False>, <id: 3, name: Paul Cezane, special: True>]

>>>

Filtrowanie danych

Propozycja rozwiązania

# in python session:

>>> from app import app, db, Authors

>>> monet = Author(id=4, name='Cloud Monet', special=False)

>>> db.session.add(monet)

>>> warhol = Authors(id=5, name='Andy Warhol')

>>> db.session.add(warhol)

>>> kahlo = Author(id=6, name='Frida Kahlo')

>>> db.session.add(kahlo)

>>> db.session.commit()

>>> Author.query.all()

>>> Author.query.filter(Author.id.in\_([1,3,5])).all()

>>> Author.query.filter(Author.special == None).all()

>>> Author.query.filter(Author.special != None).all()

>>> Author.query.filter(Author.name.like('%u%')).all()

>>> Author.query.filter(db.and\_(Author.name.like('%w%'), Author.special==None)).all()

>>> Author.query.filter(db.or\_(Author.name.like('%w%'), Author.special==None)).all()

>>> Author.query.filter(db.and\_(~Author.name.like('%w%'), Author.special==None)).all()

>>> Author.query.filter(db.and\_(~Author.name.like('%w%'), Author.special!=None)).all()

>>> Author.query.filter(db.or\_(Author.name.like('%w%'), Author.name.like('%u%'))).all()

Sortowanie,zliczanie i ograniczanie liczby rekordów

Propozycja rozwiązania

Author.query.order\_by(Author.name).all()

Author.query.order\_by(Author.name.desc()).all()

Author.query.filter(Author.special != None).order\_by(Author.name).all()

Author.query.order\_by(Author.id).limit(5).all()

Author.query.order\_by(Author.id).offset(5).limit(5).all()

Author.query.count()

Author.query.filter(Author.special == None).count()

Author.query.filter(Author.special != None).count()

Dodawanie i usuwanie danych

Propozycja rozwiązania

Author.query.all()

picasso = Author.query.filter(Author.name == 'Pablo Picasso').first()

picasso.special = True

cezane = Author.query.filter(Author.name == 'Paul Cezane').first()

cezane.special = False

db.session.commit()

Author.query.all()

warhol = Author.query.filter(Author.name == 'Andy Warhol').first()

db.session.delete(warhol)

db.session.commit()

Author.query.all()

Relacyjne bazy danych

Propozycja rozwiązania

Solution placeholder

from app import db, app, Author, ArtWork

db.create\_all()

monet = Author.query.filter(Author.name=='Cloud Monet').first()

nature = ArtWork(id=11, name='The Truth of Nature', author=monet)

sunflowers = ArtWork(id=12, name='Bouquet of Sunflowers', author=monet)

dali = Author.query.filter(Author.name=='Salvador Dali').first()

girafe = ArtWork(id=21, name='Girafe En Feu', author=dali)

anthony = ArtWork(id=22, name='Sant Anthony', author=dali)

db.session.add(nature)

db.session.add(sunflowers)

db.session.add(girafe)

db.session.add(anthony)

db.session.commit()

dali.artwork.all()

ArtWork.query.filter(ArtWork.id == 21).first().author

ArtWork.query.filter(ArtWork.id == 12).first().author = dali

db.session.commit()

Porównanie metod pracy z bazą danych

Propozycja rozwiązania

# some snippets to reuse in your solution

# old – get user information -----------------------------------------------------------------

db = get\_db()

sql\_statement = 'select id, name, email, password, is\_active, is\_admin from users where name=?'

cur = db.execute(sql\_statement, [self.user])

user\_record = cur.fetchone()

# new

user\_record = User.query.filter(User.name == self.user).first()

# old – get all users ------------------------------------------------------------------------

db = get\_db()

sql\_command = 'select id, name, email, is\_admin, is\_active from users;'

cur = db.execute(sql\_command)

users = cur.fetchall()

# new

users = User.query.all()

# old – update is\_active for a user, but not for a current user -------------------------------

db.execute("""update users set is\_active = (is\_active + 1) % 2

where name = ? and name <> ?""",

[user\_name, login.user])

# new

user = User.query.filter(User.name == user\_name, User.name != login.user).first()

if user:

user.is\_active = (user.is\_active + 1) % 2

db.session.commit()

# old – update email --------------------------------------------------------------------------

sql\_statement = "update users set email = ? where name = ?"

db.execute(sql\_statement, [new\_email, user\_name])

db.commit()

# new

user.email = new\_email

db.session.commit()

# old – delete a user -------------------------------------------------------------------------

db = get\_db()

sql\_statement = "delete from users where name = ? and name <> ?"

db.execute(sql\_statement, [user\_name, login.user])

db.commit()

# new

user = User.query.filter(User.name==user\_name, User.name != login.user).first()

if user:

db.session.delete(user)

db.session.commit()

# old – add a new user ------------------------------------------------------------------------

sql\_statement = '''insert into users(name, email, password, is\_active, is\_admin)

values(?,?,?, True, False);'''

db.execute(sql\_statement, [user['user\_name'], user['email'], password\_hash])

db.commit()

# new

new\_user = User(name=user['user\_name'], email=user['email'], password=password\_hash,

is\_active=True, is\_admin=False)

db.session.add(new\_user)

db.session.commit()

Flask-WTF – instalacja i pierwszy formularz

Propozycja rozwiązania

# app.py

from flask import Flask, render\_template, url\_for

from flask\_wtf import FlaskForm

from wtforms import StringField, IntegerField, BooleanField, SelectField

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

app.config['SECRET\_KEY'] = 'ACompl1cat3dText.'

class TrainInfo(FlaskForm):

train\_number = StringField('Train number')

is\_delayed = BooleanField('Is delayed')

delay\_minutes = IntegerField('Delay in minutes')

delay\_reason = SelectField('Delay reason', choices=['None', 'Weather', 'Failure', 'Other'])

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

def index():

form = TrainInfo()

if form.validate\_on\_submit():

return f'''<H1>Hello</H1>

<ul>

<li>{form.train\_number.label}: {form.train\_number.data}</li>

<li>{form.is\_delayed.label}: {form.is\_delayed.data}</li>

<li>{form.delay\_minutes.label}: {form.delay\_minutes.data}</li>

<li>{form.delay\_reason.label}: {form.delay\_reason.data}</li>

</ul>'''

return render\_template('index.html', form=form)

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

app.run()

# index.html

<form method="POST" action="{{ url\_for('index') }}">

{{ form.csrf\_token }}

{{ form.train\_number.label }} {{ form.train\_number }} <br/>

{{ form.is\_delayed.label }} {{ form.is\_delayed }} <br/>

{{ form.delay\_minutes.label }} {{ form.delay\_minutes }} <br/>

{{ form.delay\_reason.label }} {{ form.delay\_reason }} <br/>

<input type="submit" value="GO!">

</form>

Sprawdzanie poprawności danych (validators)

Propozycja rozwiązania

# app.py – only fragments:

from wtforms.validators import DataRequired, NumberRange, ValidationError

…

class TrainInfo(FlaskForm):

def start\_with2letters(form, field):

if not (len(field.data) > 2 and field.data[0:2].isalpha()):

raise ValidationError('Train number must start with 2 letters')

train\_number = StringField('Train number',

validators=[DataRequired('Enter train number'), start\_with2letters])

is\_delayed = BooleanField('Is delayed')

delay\_minutes = IntegerField('Delay in minutes', validators=[DataRequired('Enter delay'),

NumberRange(min=0, message='The delay must be a number >= 0')])

delay\_reason = SelectField('Delay reason', choices=['None', 'Weather', 'Failure', 'Other'])

# macros.html

{% macro show\_validation\_results(field) %}

<ul>

{% for error in field.errors %}

<li>{{ error }}</li>

{% endfor %}

</ul>

{% endmacro %}

# index.html

{% from "macros.html" import show\_validation\_results %}

<form method="POST" action="{{ url\_for('index') }}">

{{ form.csrf\_token }}

{{ form.train\_number.label }} {{ form.train\_number }} <br/>

{{ show\_validation\_results(form.train\_number) }}

{{ form.is\_delayed.label }} {{ form.is\_delayed }} <br/>

{{ form.delay\_minutes.label }} {{ form.delay\_minutes }} <br/>

{{ show\_validation\_results(form.delay\_minutes) }}

{{ form.delay\_reason.label }} {{ form.delay\_reason }} <br/>

<input type="submit" value="GO!">

</form>

Zapisywanie danych z formularzy w obiektach

Propozycja rozwiązania

# app.py – only selected fragments:

from flask\_sqlalchemy import SQLAlchemy

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

app.config['SECRET\_KEY'] = 'ACompl1cat3dText.'

app.config['SQLALCHEMY\_DATABASE\_URI']='sqlite:///d:\\Flask\\trains.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS']=False

db = SQLAlchemy(app)

class TrainDelay(db.Model):

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

train\_number = db.Column(db.String(30))

is\_delayed = db.Column(db.Boolean)

delay\_minutes = db.Column(db.Integer)

delay\_reason = db.Column(db.String(100))

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

return f"{self.train\_number} - {self.delay\_minutes}"

db.create\_all()

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

def index():

form = TrainInfo()

if form.validate\_on\_submit():

train\_delay = TrainDelay()

form.populate\_obj(train\_delay)

db.session.add(train\_delay)

db.session.commit()

return f'''<H1>Hello</H1>

<ul>

<li>{form.train\_number.label}: {form.train\_number.data}</li>

<li>{form.is\_delayed.label}: {form.is\_delayed.data}</li>

<li>{form.delay\_minutes.label}: {form.delay\_minutes.data}</li>

<li>{form.delay\_reason.label}: {form.delay\_reason.data}</li>

</ul>

Following record was added to the table in database: {train\_delay}

'''

return render\_template('index.html', form=form)

Dziedziczenie z klasy formularza

Propozycja rozwiązania

# app.py – only important fragments:

class TrainDelayOnStation(db.Model):

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

train\_number = db.Column(db.String(30))

is\_delayed = db.Column(db.Boolean)

delay\_minutes = db.Column(db.Integer)

delay\_reason = db.Column(db.String(100))

station = db.Column(db.String(50))

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

return f"{self.station} - {self.train\_number} - {self.delay\_minutes}"

class TrainInfoOnStation(TrainInfo):

station = StringField('Station', validators=[DataRequired('Enter station name')])

@app.route('/delay\_on\_station', methods=['POST', 'GET'])

def delay\_on\_station():

form = TrainInfoOnStation()

if form.validate\_on\_submit():

train\_delay\_on\_station = TrainDelayOnStation()

form.populate\_obj(train\_delay\_on\_station)

db.session.add(train\_delay\_on\_station)

db.session.commit()

return f'''<H1>Hello</H1>

<ul>

<li>{form.train\_number.label}: {form.train\_number.data}</li>

<li>{form.is\_delayed.label}: {form.is\_delayed.data}</li>

<li>{form.delay\_minutes.label}: {form.delay\_minutes.data}</li>

<li>{form.delay\_reason.label}: {form.delay\_reason.data}</li>

<li>{form.station.label}: {form.station.data}</li>

</ul>

Following record was added to the table in database: {train\_delay\_on\_station}

'''

return render\_template('index.html', form=form)

# index.html

{% from "macros.html" import show\_validation\_results %}

<form method="POST" }}">

{{ form.csrf\_token }}

{% if form.station %}

{{ form.station.label }} {{ form.station }} <br/>

{{ show\_validation\_results(form.station) }}

{% endif %}

{{ form.train\_number.label }} {{ form.train\_number }} <br/>

{{ show\_validation\_results(form.train\_number) }}

{{ form.is\_delayed.label }} {{ form.is\_delayed }} <br/>

{{ form.delay\_minutes.label }} {{ form.delay\_minutes }} <br/>

{{ show\_validation\_results(form.delay\_minutes) }}

{{ form.delay\_reason.label }} {{ form.delay\_reason }} <br/>

<input type="submit" value="GO!">

</form>

Korzystanie z kontrolek WTForms

{% for option in form.delay\_reason %}

<tr>

<td>{{ option }}</td>

<td>{{ option.label }}</td>

</tr>

{% endfor %}

Propozycja rozwiązania

# app.py – only changed fragments:

from wtforms import StringField, IntegerField, BooleanField, SelectField, RadioField

class TrainInfo(FlaskForm):

def start\_with2letters(form, field):

if not (len(field.data) > 2 and field.data[0:2].isalpha()):

raise ValidationError('Train number must start with 2 letters')

train\_number = StringField('Train number',

validators=[DataRequired('Enter train number'), start\_with2letters])

is\_delayed = BooleanField('Is delayed')

delay\_minutes = IntegerField('Delay in minutes',

validators=[DataRequired('Enter delay'),

NumberRange(min=0, message='The delay must be a number >= 0')])

#delay\_reason = SelectField('Delay reason', choices=['None', 'Weather', 'Failure', 'Other'])

delay\_reason = RadioField('Delay reason', choices=['None', 'Weather', 'Failure', 'Other'],

default='None')

# index.html – full content

{% from "macros.html" import show\_validation\_results %}

<form method="POST" }}">

{{ form.csrf\_token }}

{% if form.station %}

{{ form.station.label }} {{ form.station }} <br/>

{{ show\_validation\_results(form.station) }}

{% endif %}

{{ form.train\_number.label }} {{ form.train\_number }} <br/>

{{ show\_validation\_results(form.train\_number) }}

{{ form.is\_delayed.label }} {{ form.is\_delayed }} <br/>

{{ form.delay\_minutes.label }} {{ form.delay\_minutes }} <br/>

{{ show\_validation\_results(form.delay\_minutes) }}

<!--{{ form.delay\_reason.label }} {{ form.delay\_reason }} <br/> -->

{% for option in form.delay\_reason %}

<tr>

<td>{{ option }}</td>

<td>{{ option.label }}</td>

</tr>

{% endfor %}</br>

<input type="submit" value="GO!">

</form>

Kontrolki HTML5

Propozycja rozwiązania

#app.py – only fragments:

class TrainDelayOnStationDay(db.Model):

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

train\_number = db.Column(db.String(30))

is\_delayed = db.Column(db.Boolean)

delay\_minutes = db.Column(db.Integer)

delay\_reason = db.Column(db.String(100))

station = db.Column(db.String(50))

day = db.Column(db.Date)

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

return f"{self.station} - {self.train\_number} - {self.delay\_minutes}"

class TrainInfo(FlaskForm):

def start\_with2letters(form, field):

if not (len(field.data) > 2 and field.data[0:2].isalpha()):

raise ValidationError('Train number must start with 2 letters')

train\_number = StringField('Train number',

validators=[DataRequired('Enter train number'), start\_with2letters])

is\_delayed = BooleanField('Is delayed')

delay\_minutes = IntegerField('Delay in minutes', validators=[DataRequired('Enter delay'),

NumberRange(min=0, message='The delay must be a number >= 0')])

delay\_reason = RadioField('Delay reason', choices=['None', 'Weather', 'Failure', 'Other'],

default='None')

class TrainInfoOnStation(TrainInfo):

station = StringField('Station', validators=[DataRequired('Enter station name')])

day = DateField('Day', validators=[DataRequired('Enter date')], default=date.today())

@app.route('/delay\_on\_station', methods=['POST', 'GET'])

def delay\_on\_station():

form = TrainInfoOnStation()

if form.validate\_on\_submit():

train\_delay\_on\_station\_day = TrainDelayOnStationDay()

form.populate\_obj(train\_delay\_on\_station\_day)

db.session.add(train\_delay\_on\_station\_day)

db.session.commit()

return f'''<H1>Hello</H1>

<ul>

<li>{form.train\_number.label}: {form.train\_number.data}</li>

<li>{form.is\_delayed.label}: {form.is\_delayed.data}</li>

<li>{form.delay\_minutes.label}: {form.delay\_minutes.data}</li>

<li>{form.delay\_reason.label}: {form.delay\_reason.data}</li>

<li>{form.station.label}: {form.station.data}</li>

<li>{form.day.label}: {form.day.data}</li>

</ul>

Following record was added to the db: {train\_delay\_on\_station\_day}

'''

return render\_template('index.html', form=form)

#index.html – only added part:

{% if form.day %}

{{ form.day.label }} {{ form.day }} <br/>

{{ show\_validation\_results(form.day) }}

{% endif %}

Flask Login – testowa aplikacja

Propozycja rozwiązania

# app.py – only changed/added fragments

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

from flask\_wtf import FlaskForm

from wtforms import BooleanField, StringField, SelectField, RadioField, PasswordField

from wtforms.fields.html5 import IntegerField, DateField

from wtforms.validators import DataRequired, NumberRange, ValidationError

from flask\_sqlalchemy import SQLAlchemy

from datetime import date

import hashlib

import binascii

class User(db.Model):

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

name = db.Column(db.String(50), unique=True)

password = db.Column(db.String(100))

first\_name = db.Column(db.String(50))

last\_name = db.Column(db.String(50))

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

return ('User: {},{}'.format(self.name))

def get\_hashed\_password(password):

"""Hash a password for storing."""

# the value generated using os.urandom(60)

os\_urandom\_static = b"ID\_\x12p:\x8d\xe7&\xcb\xf0=H1\xc1\x16\xac\xe5BX\xd7\xd6j\xe3i\x11\xbe\xaa\x05\xccc\xc2\xe8K\xcf\xf1\xac\x9bFy(\xfbn.`\xe9\xcd\xdd'\xdf`~vm\xae\xf2\x93WD\x04"

salt = hashlib.sha256(os\_urandom\_static).hexdigest().encode('ascii')

pwdhash = hashlib.pbkdf2\_hmac('sha512', password.encode('utf-8'), salt, 100000)

pwdhash = binascii.hexlify(pwdhash)

return (salt + pwdhash).decode('ascii')

def verify\_password(stored\_password\_hash, provided\_password):

"""Verify a stored password against one provided by user"""

salt = stored\_password\_hash[:64]

stored\_password = stored\_password\_hash[64:]

pwdhash = hashlib.pbkdf2\_hmac('sha512', provided\_password.encode('utf-8'), salt.encode('ascii'), 100000)

pwdhash = binascii.hexlify(pwdhash).decode('ascii')

return pwdhash == stored\_password

class LoginForm(FlaskForm):

name = StringField('User name')

password = PasswordField('Password')

remember = BooleanField('Remember me')

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

def login():

form = LoginForm()

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

@app.route('/logout')

def logout():

return '<h1>You are logged out</h1>'

# login.html – full

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">

<title>Logon</title>

</head>

<body>

{% for message in get\_flashed\_messages() %}

<div class="alert alert-warning alert-dismissible fade show" role="alert">

<strong>{{ message }}</strong>

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

</div>

{% endfor %}

<!--Grid row-->

<div class="row d-flex justify-content-center">

<!--Grid column-->

<div class="col-md-4">

<form method="POST">

{{ form.csrf\_token }}

<div class="form-row align-items-center">

<div class="col-auto my-1">

<label class="sr-only" for="inlineFormInputName">{{ form.name.label }}</label>

{{ form.name(class\_="form-control") }}

</div>

<div class="col-auto my-1">

<label class="sr-only" for="inlineFormInputGroupUsername">{{ form.password.label }}</label>

{{ form.password(class\_="form-control") }}

</div>

<div class="col-auto my-1">

<div class="form-check">

{{ form.remember(class\_="form-check-input") }}

<label class="form-check-label" for="autoSizingCheck2">

{{ form.remember.label }}

</label>

</div>

</div>

<div class="col-auto my-1">

<button type="submit" class="btn btn-primary">Login</button>

</div>

</div>

</form>

</div>

<!--Grid column-->

</div>

<!--Grid row-->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous"></script>

</body>

</html>

Flask-Login instalacja i wykorzystanie

Propozycja rozwiązania

#app.py – only fragments

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

login\_manager = LoginManager(app)

class User(db.Model, UserMixin):

# […]

@login\_manager.user\_loader

def load\_user(id):

return User.query.filter(User.id == id).first()

@app.route('/init')

def init():

db.create\_all()

admin = User.query.filter(User.name=='admin').first()

if admin == None:

admin = User(id=1, name='admin', password=User.get\_hashed\_password('Passw0rd'),

first\_name='King', last\_name='Kong')

db.session.add(admin)

db.session.commit()

return '<h1>Initial configuration done!</h1>'

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

def login():

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter(User.name == form.name.data).first()

if user != None and User.verify\_password(user.password, form.password.data):

login\_user(user)

next = request.args.get('next')

if next and is\_safe\_url(next):

return redirect(next)

else:

return '<h1>You are authenticated!</h1>'

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

@app.route('/logout')

def logout():

logout\_user()

return '<h1>You are logged out</h1>'

@app.route('/delay\_on\_station', methods=['POST', 'GET'])

@login\_required

def delay\_on\_station():

# […]

Przekierowanie użytkownika do strony logowania

Propozycja rozwiązania

from urllib.parse import urlparse, urljoin

login\_manager.login\_view = 'login'

login\_manager.login\_message = 'First, please log in using this form:'

def is\_safe\_url(target):

ref\_url = urlparse(request.host\_url)

test\_url = urlparse(urljoin(request.host\_url, target))

return test\_url.scheme in ('http', 'https') and \

ref\_url.netloc == test\_url.netloc

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

def login():

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter(User.name == form.name.data).first()

if user != None and User.verify\_password(user.password, form.password.data):

login\_user(user)

next = request.args.get('next')

if next and is\_safe\_url(next):

return redirect(next)

else:

return '<h1>You are authenticated!</h1>'

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

Wymuszenie ponownego logowania

Propozycja rozwiązania

# app.py

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

def login():

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter(User.name == form.name.data).first()

if user != None and User.verify\_password(user.password, form.password.data):

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

next = request.args.get('next')

if next and is\_safe\_url(next):

return redirect(next)

else:

return '<h1>You are authenticated!</h1>'

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

Spróbuj też!

