

✓ Wykonano

Ponieważ na najbliższych zajęciach nie będzie już zadania ćwiczeniowego, dlatego do samodzielnego wykonania jest pięć zadań przygotowawczych (ikona 🏠).

## Materiały pomocnicze

### Deno

1.

Deno - następca Node.js? Backend dla JavaScript developerów



2.

Getting Started with Deno & MongoDB



### Bezpieczeństwo aplikacji Node.js

1. Production Best Practices: Security
2. Node.js Security Checklist

## Narzędzia do badania bezpieczeństwa aplikacji WWW

1. [Różnice między SAST, DAST, IAST i RASP](#)
2. [Application Security Testing - Top Questions Answered](#)
- 3.

Starting With Snyk: an overview of the CLI onboarding flow



4. [Wprowadzenie do narzędzia Zed Attack Proxy \(ZAP\).](#)
5. [Samouczek OWASP ZAP: Kompleksowy przegląd narzędzia OWASP ZAP](#)
- 6.

OWASP ZAP - Instalacja



### 1. Deno

1. [Zainstaluj](#) Deno
2. Zainstaluj [wtyczkę](#) Deno.
3. Wykonaj następujące komendy:

```
1. deno init cw7
2. cd cw7
3. mkdir views
```

4. Obejrzyj zawartość wszystkich wygenerowanych plików — `more * | cat`
5. Sprawdź, w praktyce, działanie poniższych komend:

```
$ deno run main.ts
$ deno task dev
$ deno test
```

6. Utwórz plik 'app1.ts' o następującej zawartości:

```
1.  /**
2.   * @author Stanisław Polak <polak@agh.edu.pl>
3.   */
4.
5.  // @deno-types="npm:@types/express@^4"
6.  import express, { Express, Request, Response } from "npm:express@^4";
7.  import morgan from "npm:morgan@^1";
8.  import "npm:pug@^3";
9.
10. const app: Express = express();
11. deno_logo =
12.   'https://upload.wikimedia.org/wikipedia/commons/thumb/e/e8/Deno_2021.svg/120px-Deno_2021.svg.png';
13.
14. app.set('view engine', 'pug');
15. app.locals.pretty = app.get('env') === 'development';
16.
17. app.use(morgan('dev'));
18. app.use(express.urlencoded({ extended: false }));
19.
20. app.get('/', function (_req: Request, res: Response) {
21.   res.render('index', {deno_logo});
22. });
23.
24. app.post('/', function (req: Request, res: Response) {
25.   res.send(`Hello ${req.body.name}`);
26. });
27.
28. app.listen(8000, function () {
29.   console.log('The application is available on port 8000');
```

7. Utwórz plik 'views/index.pug' o następującej zawartości:

```
1.  doctype html
2.  html(lang='en')
3.    head
4.      meta(charset='utf-8')
5.      meta(name='viewport' content='width=device-width, initial-scale=1')
6.      title First Express application in Deno
7.    body
8.      main
9.        h1 First Express application in Deno
10.       img(src=deno_logo alt='[Deno logo]')
11.       form(method='POST' action='/')
12.         label(for='name') Give your name
13.         input(name='name')
14.         br
15.         input(type='submit')
16.         input(type='reset')
```

8. Uruchom aplikację — `deno run app1.ts` i sprawdź, czy *Deno*, domyślnie:

- Używa [trybu ścisłego](#)?
- Uwzględnia typowanie?

9. Popraw błędną linię i ponownie uruchom aplikację.

10. Wpisz w przeglądarce adres <http://localhost:8000/> i obejrzyj stronę wynikową.

11. Zatrzymaj serwer.

12. Sprawdź poprawność typowania za pomocą komendy `deno check app1.ts`, a następnie popraw znalezione błędy typowania.

13. Przeczytaj następujące sekcje manuala: [Command Line Interface](#), [Configuration File](#) oraz [Permissions](#)

14. Uruchom aplikację w taki sposób (z takimi opcjami linii komend), aby *Deno*:

- Wykonywał sprawdzanie typowania przed jej uruchomieniem.
- Restartował ją, gdy treść kodu źródłowego uległa zmianie.
- Nie pytał o uprawnienia do wykonywania operacji wrażliwych (dostęp do sieci, itp.), tzn. należy zezwolić na wykonywanie tych operacji za pomocą opcji linii komend.

15. Spowoduj, aby wykonanie komendy `deno task express` uruchomiło aplikację z wyżej opisanymi opcjami linii komend.

## Dla ciekawskich

- Zarządzanie zależnościami
- Odpowiedniki komend NodeJS



## 2. Język TypeScript

- Poniższe zadanie proszę wykonać w oparciu o slajdy z wykładu lub podręcznik języka TypeScript
- Możesz skorzystać z pakietu *Express* — patrz 'app1.ts' lub użyć wbudowanego serwera WWW — patrz przykład poniżej.

```

1.  /**
2.   * @author Stanisław Polak <polak@agh.edu.pl>
3.   */
4.
5.  Deno.serve(async (req) => {
6.    const url = new URL(req.url);
7.
8.    switch ([req.method, url.pathname].join(" ")) {
9.      case "GET /":
10.        return new Response(
11.          `<!DOCTYPE html>
12.          <html lang="en">
13.            <head>
14.              <meta charset="utf-8">
15.              <meta name="viewport" content="width=device-width, initial-scale=1">
16.              <title>Vanilla Deno application</title>
17.            </head>
18.            <body>
19.              <main>
20.                <h1>Vanilla Deno application</h1>
21.                <form method="POST" action="/">
22.                  <label for="name">Give your name</label>
23.                  <input name="name">
24.                  <br>
25.                  <input type="submit">
26.                  <input type="reset">
27.                </form>
28.              </main>
29.            </body>
30.          </html>`,
31.          {
32.            headers: {
33.              "content-type": "text/html; charset=utf-8",
34.            },
35.          }
36.        );
37.      case "POST /": {
38.        const data = await req.formData();
39.        return new Response(`Hello '${data.get("name")}'`);
40.      }
41.      default:
42.        return new Response("Error 501: Not implemented", { status: 501 });
43.    }
44.  });

```

Zmodyfikuj aplikację "Księga gości", którą utworzyłeś/aś na zajęciach "Środowisko uruchomieniowe NodeJS":

- Implementacja ma bazować na języku *TypeScript*, a nie *JavaScript*:
  - Obowiązkowo**, wszystkie zmienne oraz argumenty funkcji/metod mają posiadać [deklarację typu](#). W przypadku ostatnich z wymienionych należy również określić typ zwracanej wartości.  
O ile to możliwe, proszę unikać następujących typów: *any* oraz *unknown*
  - Dla ambitnych**, użyj również innych składników *TypeScript*, na przykład: [klasy](#), [moduły](#) / [przestrzenie nazewnicze](#) ...
- Wpisy mają być przechowywane w bazie danych MongoDB, a nie w pliku
- Trasy:
  - GET(/) — wyświetlenie aktualnych wpisów oraz formularza dodawania nowego wpisu
  - POST(/) — przetwarzanie zawartości formularza — dodawanie nowego wpisu do bazy danych



### 3. Oak

- Przeczytaj [Web Frameworks](#)
- Utwórz plik 'app2.ts' o następującej zawartości:

```

1. // Original source: https://medium.com/recoding/rendering-html-css-in-deno-using-view-
   engine-e07469613598
2. // Modifications: Stanisław Polak <polak@agh.edu.pl>
3.
4. // Requiring modules
5. import {
6.   Application,
7.   Router,
8.   Context,
9.   // send
10. } from "https://deno.land/x/oak/mod.ts";
11. import {
12.   dejsEngine,
13.   oakAdapter,
14.   viewEngine,
15. } from "https://deno.land/x/view_engine/mod.ts";
16. import logger from "https://deno.land/x/oak_logger/mod.ts";
17.
18. // Initiate app
19. const app: Application = new Application();
20. const router: Router = new Router({
21.   // prefix: "/admin",
22. });
23.
24. // Allowing Static file to fetch from server
25. /*
26. app.use(async (ctx, next) => {
27.   await send(ctx, ctx.request.url.pathname, {
28.     root: `${Deno.cwd()}/public`,
29.   });
30.   next();
31. });
32. */
33.
34. app.use(logger.logger);
35. app.use(logger.responseTime);
36.
37. // Passing view-engine as middleware
38. app.use(viewEngine(oakAdapter, dejsEngine, { viewRoot: "./views" }));
39.
40. // Creating Routes
41. router.get("/", async (ctx: Context) => {
42.   await ctx.render("index.ejs", {
43.     data: { title: "First Oak application in Deno" },
44.   });
45. });
46.
47. router.post("/", async (ctx: Context) => {
48.   const reqBodyForm = await ctx.request.body.form();
49.   // ctx.response.type = 'text/html'
50.   ctx.response.body = `Hello '${reqBodyForm.get("name")}'`;
51. });
52.
53. // Adding middleware to require our router
54. app.use(router.routes());
55. app.use(router.allowedMethods());
56.
57. // Making app to listen to port
58. console.log("App is listening to port: 8000");
59. await app.listen({ port: 8000 });

```

3. Utwórz plik 'views/index.ejs' o następującej zawartości:

```

1. <html lang="en">
2.
3. <head>
4.   <meta charset="utf-8">
5.   <meta name="viewport" content="width=device-width, initial-scale=1">
6.   <title><%= data.title %></title>
7. </head>
8.
9. <body>
10.   <main>
11.     <h1><%= data.title %></h1>
12.     
14.     <form method="POST" action="/">
15.       <label for="name">Give your name</label>
16.       <input name="name"><br>
17.       <input type="submit">
18.       <input type="reset">
19.     </form>
20.   </main>
21. </body>
22.
23. </html>

```

4. Uruchom aplikację, a następnie wpisz w przeglądarce adres <http://localhost:8000/> i obejrzyj stronę wynikową.
5. Bazując na powyższym przykładzie, utwórz alternatywną wersję aplikacji "Księga gości":
  - Zamiast z *Express* ma korzystać z [Oak](#)
  - Zamiast *Pug* ma używać systemu szablonów wspieranego przez [View Engine](#)
  - Obsługa bazy danych *MongoDB* — [deno mongo](#), [npm:mongodb](#) lub dowolny inny sterownik

## Uwagi

1. Program *Deno* należy uruchamiać bez opcji `--check`



## 4. Narzędzie Snyk

1. Zainstaluj wtyczkę [Snyk Security - Code, Open Source Dependencies, IaC Configurations](#)

The Big Fix 2022 - Getting started with VS Code IDE security fixes



2. Korzystając z tej wtyczki, znajdź luki w kodzie źródłowym aplikacji, którą tworzymy przez cały cykl ćwiczeń z JavaScript. Usuń znalezione luki.

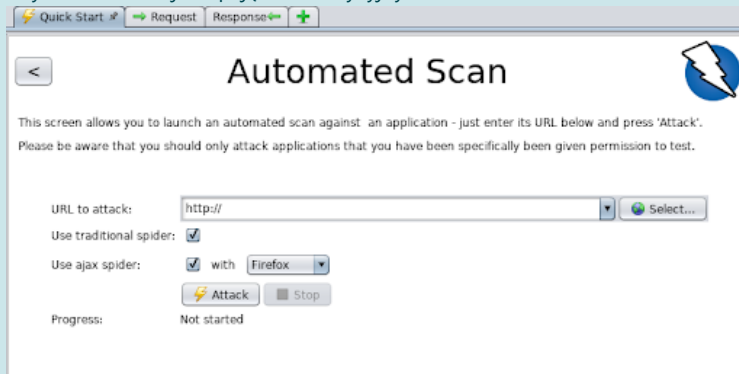
- W przypadku osób, które nie dysponują własną implementacją aplikacji z ćwiczeń, proszę użyć Snyk-a do znalezienia luk w kodzie źródłowym aplikacji "**Księga gości**".
- Nie musisz instalować programu *Snyk*, gdyż jest on automatycznie instalowany wraz ze wtyczką. Jeżeli chcesz go używać z linii komend, to skorzystaj z komendy 'npx', np. `npx snyk --help`.



## 5. Narzędzie ZAP

1. Zainstaluj narzędzie [ZAP](#)
2. Uruchom aplikację z ćwiczeń
3. Za pomocą programu *ZAP* — opcja "Automated Scan" — przetestuj bezpieczeństwo swojej aplikacji. Popraw ją — usuń, znalezione w niej, luki **poziomu ryzyka średniego oraz wysokiego**

- Ponieważ aplikacja z ćwiczeń używa również JavaScript po stronie klienta, zachęcam do przetestowania jej bezpieczeństwa obydwoma rodzajami pajaków: tradycyjnym oraz AJAX.




- W przypadku osób, które nie dysponują własną implementacją aplikacji z ćwiczeń, proszę przetestować aplikację "**Księga gości**".

Edytuj zadanie

Usuń zadanie

## Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Nieocenione
Ostatnio modyfikowane	środa, 5 czerwca 2024, 19:35
Przesyłane pliki	<div>  <a href="#">lab7.zip</a> <span>5 czerwca 2024, 19:35</span> </div>
Komentarz do przesłanego zadania	<a href="#">Komentarze (0)</a>

## Informacja zwrotna

Ocena	5,00 / 5,00
Ocenione dnia	piątek, 14 czerwca 2024, 08:51



---

Platforma obsługiwana przez:  
[Centrum e-Learningu i Innowacyjnej Dydaktyki AGH](#)  
[Centrum Rozwiązań Informatycznych AGH](#)

---

Pobierz aplikację mobilną



---

Wybierz język

