Sprawozdanie z laboratorium 1

Mikołaj Kubś 272662

30 marca 2025

1 Cel zadania

Celem zadania było zapoznanie się z procesem tworzenia serwerów i stron internetowych za pomocą Node.js oraz Express.js. W późniejszych etapach należało utworzyć obraz Dockerowy, aby zbudować kontener z aplikacją. Ostatecznie należało przekształcić aplikację, aby działała w trybie serverless i wdrożyć ją na jednej z platform chmurowych.

2 Wykorzystane technologie

- Node.js
- Express.js (wraz z Expressjs-layouts)
- serverless, serverless-http
- Docker - do tworzenia kontenerów
- Vercel - do hostowania aplikacji

Dodatkowo wszystkie pliki źródłowe zostały napisane w TypeScript, a proces kompilacji został zautomatyzowany przed każdym uruchomieniem aplikacji.

3 Opis aplikacji

Aplikacja jest stroną internetową typu Portfolio, która może zawierać informacje o użytkowniku, jego projektach oraz galerię zdjęć. Dotyczy konkretnie autora sprawozdania, jego doświadczenia profesjonalnego i projektach w formie galerii. Dodatkowo aplikacja zawiera formularz kontaktowy, który w przyszłości umożliwi kontakt z autorem.

4 Opis procesu

4.1 Tworzenie aplikacji

W celu utworzenia aplikacji został stworzony plik o nazwie `server.ts`, który zawiera kod odpowiedzialny za uruchomienie serwera, konfigurację ścieżek oraz przekierowywanie za pomocą middleware.

Formularz kontaktowy jest obsługiwany przez middleware `body-parser`, który przekształca dane z formularza na format JSON.

Pliki statyczne, takie jak CSS, skrypty TypeScript czy grafiki, są przechowywane w katalogu `public`, a ich ścieżka jest przekazywana do middleware `express.static()`.

4.2 Dockeryzacja

Aby utworzyć obraz Dockera, został stworzony plik `Dockerfile`, który zawiera instrukcje dotyczące budowy obrazu. Następnie za pomocą polecenia:

```
docker build -t node-web-app .
```

został utworzony obraz o nazwie `node-web-app`. Po zbudowaniu obrazu, można uruchomić kontener za pomocą polecenia:

```
docker run -p port:port -d node-web-app
```

4.3 Serverless

Aby przekształcić aplikację w tryb serverless, zostały wykorzystane dwie biblioteki: `serverless` oraz `serverless-http`. Następnie zmodyfikowano kod w pliku `server.ts` oraz dodano plik konfiguracyjny `serverless.yml`.

Za pomocą polecenia:

```
serverless start-offline
```

można uruchomić aplikację w trybie offline, co pozwala na testowanie aplikacji bez potrzeby wdrażania jej na platformę chmurową. Problemem, który napotkałem podczas uruchamiania aplikacji w trybie offline, był brak dostępu do plików statycznych, tj. obrazów. Pozostałe pliki typu `js` oraz `css` w folderze `public` były dostępne. Mimo że pliki z obrazami były poprawnie przekazywane, to nie były wyświetlane. Żeby naprawić ten problem, trzeba było poprawić plik `serverless.yml`, dodając odpowiednie include'y i akceptowalne typy binarne mediów.

4.4 Wdrożenie aplikacji na Vercel

Aby wdrożyć aplikację na Vercel, należy zainstalować CLI Vercel oraz utworzyć konto na platformie. Następnie należy zalogować się do swojego konta za pomocą polecenia:

```
vercel login
```

Po zalogowaniu się można wdrożyć aplikację za pomocą polecenia:

```
vercel
```

Oraz deployować na produkcję:

```
vercel --prod
```

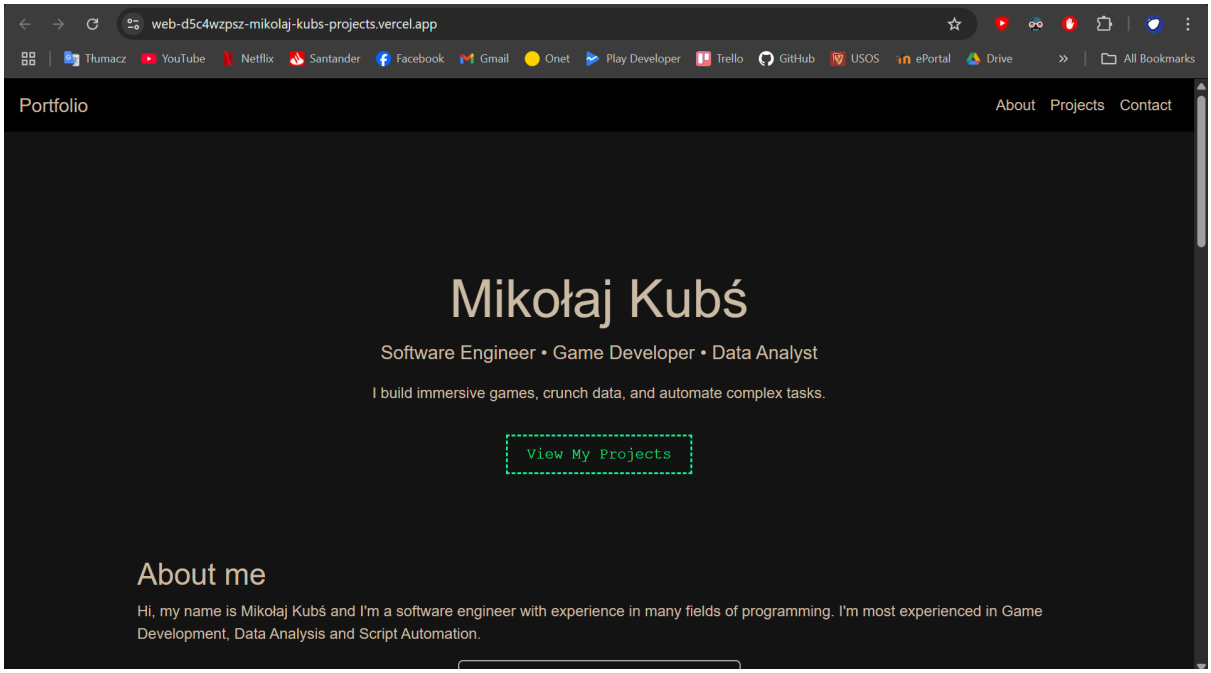
Oraz testować vercel lokalnie:

```
vercel dev
```

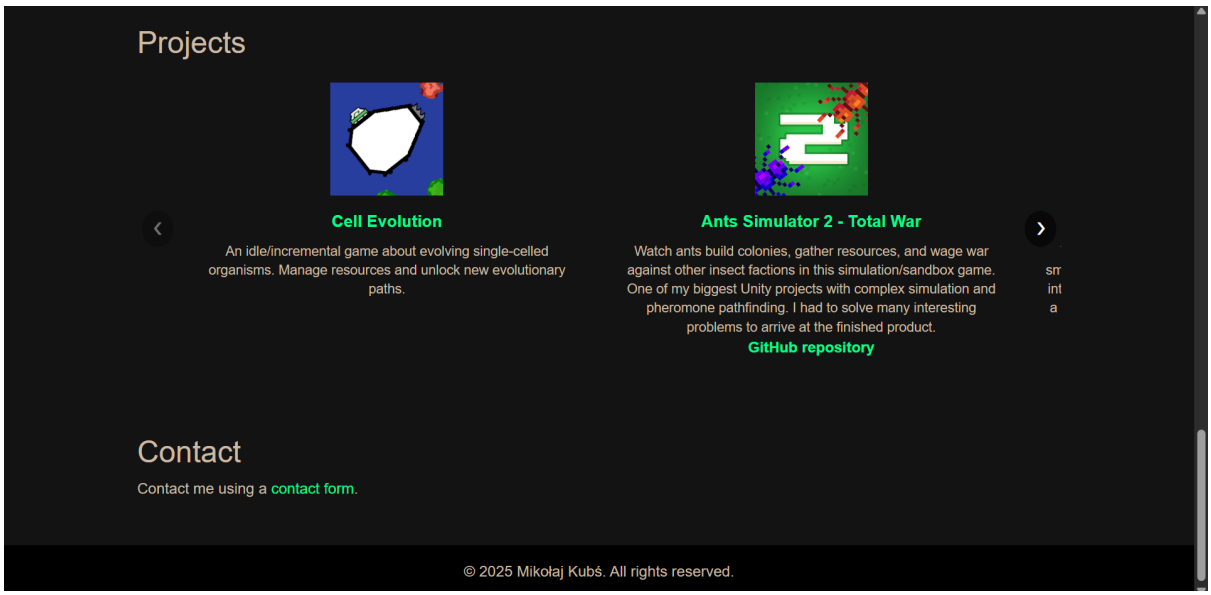
Strona znajduje się pod linkiem: Portfolio Mikołaj Kubś.

Wdrożenie aplikacji przebiegło względnie bezproblemowo. Wystąpiły problemy z ładowaniem treści strony, a potem obrazów czy plików JavaScript. Należało poprawić `vercel.json`, `tsconfig.json` oraz `server.ts`. Część błędów powstała przez używanie TypeScript i kompilacji plików do folderu `dist`.

5 Wygląd aplikacji



Rysunek 1: Wygląd strony



Rysunek 2: Wygląd galerii z projektami

Rysunek 3: Wygląd formularza z kontaktem