Express - cechy

- egzystuje w warstwie middleware (można go zastosować do front-endu / back-endu)
- prosty/minimalistyczny (posiada „podstawowe” funkcjonalności)
- lekki/szybki (z w/w powodu oraz braku „przesłaniających” funkcjonalności)
- rozszerzalny (współpracuje z innymi modułami Node)
- elastyczny framework serwera HTTP (nie narzuca jedynego właściwego sposobu)
- zastosowanie w aplikacjach webowych i mobilnych
- posiada API

Uruchomienie serwera (dla aplikacji)

1. załącz bibliotekę Express
2. „zainstancjonuj” serwer Express
3. ustal dodatkowe parametry dla serwera use()
4. określ ścieżki (routes) dostępu do funkcjonalności
5. ustaw port nasłuchujący nadchodzących żądań

```
const http = require('express')
const app = http()
const router = require('./routes/api')
app.use('/',router)
app.listen(conf.port)
```

Obsługa żądań

Serwer standardowo obsługuje wszystkie metody żądań HTTP: GET, POST, PUT, DELETE

`app.get()` - pozwala odebrać adres URL żądania (wraz z parametrami) i na nie zareagować

`app.post()` - pozwala odebrać aplikacji parametry żądania przesyłane w nagłówku zamiast w adresie URL (odpowiednik tablicy **$_POST** odbierającej dane z formularza)

W obu przypadkach metody posiadają 2 parametry:

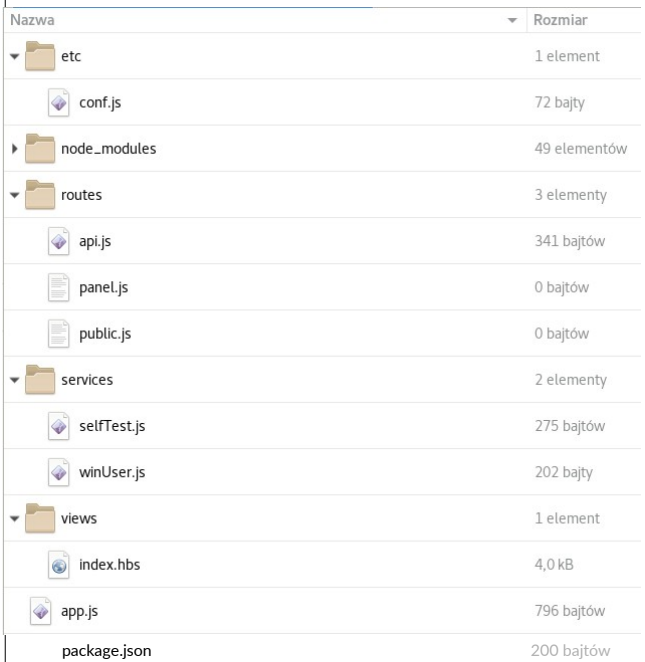
request – żądanie	**response – odpowiedź**
`req.app`	**`res.app`**
`req.baseUrl`	**`res.headersSent`**
`req.body`	**`res.locals`**
oraz 17 innych metod *oprócz właściwości jest też 8 metod*	*oprócz właściwości są też metody odpowiedzi (następna tabela)*

Wybrane metody odpowiedzi

`res.download()`	Wysyła plik do klienta
`res.end()`	Kończy proces odpowiedzi
`res.json()`	Wysyła dane w formacie JSON
`res.jsonp()`	j/w lecz dodatkowo obsługuje obiekt callback
`res.redirect()`	Obsługuje mechanizmy przekierowania
`res.render()`	Renderuje widok – odsyła klientowi string dokumentu HTML
`res.send()`	Wysyła klientowi odpowiedź w wybranym formacie
`res.sendFile()`	Wysyła klientowi plik jako strumień (octet stream)
`res.sendStatus()`	Ustawia kod statusu odpowiedzi oraz tekstową reprezentację jako body odpowiedzi

Struktura projektu

- zależy od programisty (Node i Express są elastyczne)
- musi zawierać katalog z modułami Node'a (node_modules)
- powinna być sensowna i logiczna z punktu widzenia programisty oraz aplikacji
- często realizowana struktura odpowiada wzorcowi MVC lub architekturze serwisów Web z Akcjami lub z Usługami pisanymi w Java

Przykładowa struktura	***Znaczenie/Zastosowanie***
	Katalog z plikami konfiguracyjnymi dla aplikacji i modułów
	Moduły używane przez aplikację
	Ścieżki pozwalające wywołać czynności w aplikacji często nazywane API. Ścieżki mogą być też podzielone na grupy, np.: czynności dostępne publicznie oraz czynności w obrębie panelu zarządzania aplikacji
	Katalog z głównymi funkcjonalnościami, które można nazwać: Akcje (actions), Kontrolery (controlers), Usługi (services)
	Katalog z widokami, czyli wyświetlanymi stronami witryny/aplikacji. Te mogą być pogrupowane na kategorie odpowiadające kontrolerom.
	Główny plik aplikacji oraz plik metadanych opisujących aplikację oraz wymagane pakiety