Zaawansowane Techniki WWW
(HTML, CSS i JavaScript)

Dr inż. Marcin Zieliński

Środa 15:30 - 17:00 sala: A-1-04

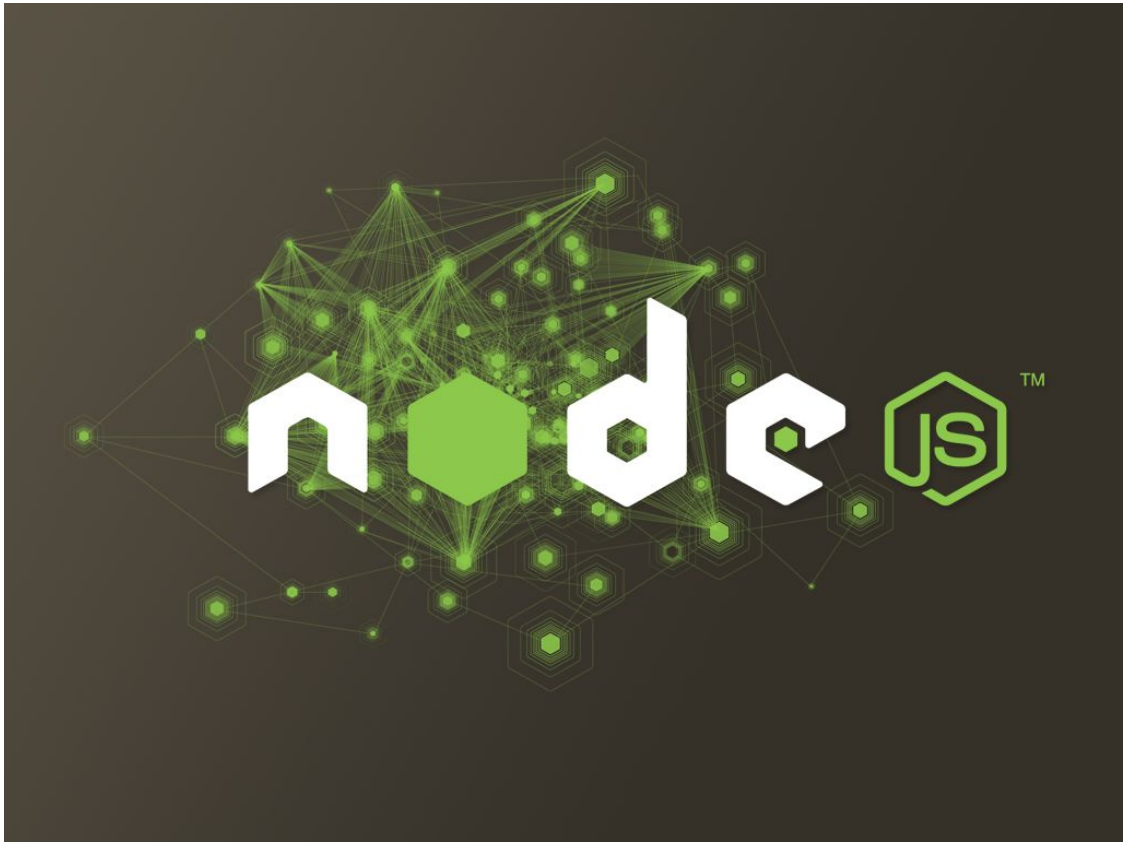
WYKŁAD 11

Wykład dla kierunku: **Informatyka Stosowana II rok**

Rok akademicki: **2015/2016 - semestr zimowy**

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Node.js



http://nodejs.org/

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

JavaScript po stronie serwera

SERVER SIDE

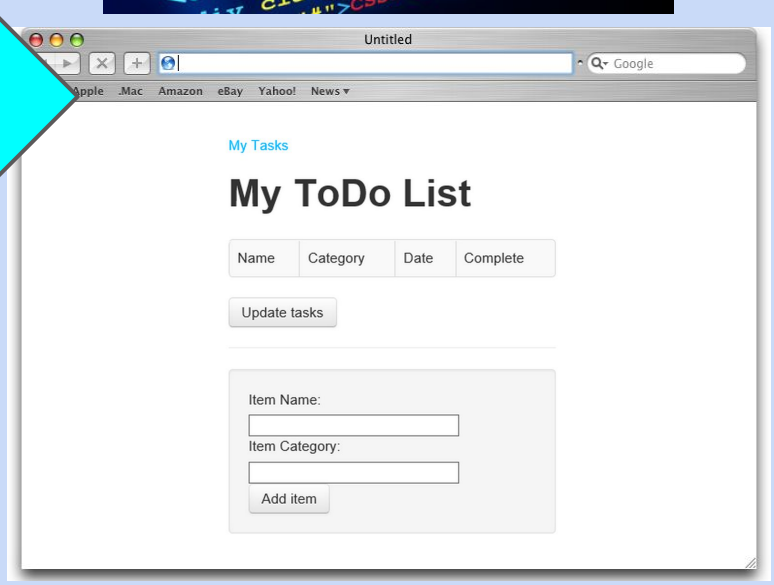
```
var edge = require('edge');

var hello = edge.func(function () {/*
    async (input) =>
    {
        return ".NET welcomes " + input.ToString();
    }
*/});

hello('Node.js', function (error, result) {
    if (error) throw error;
    console.log(result);
});
```

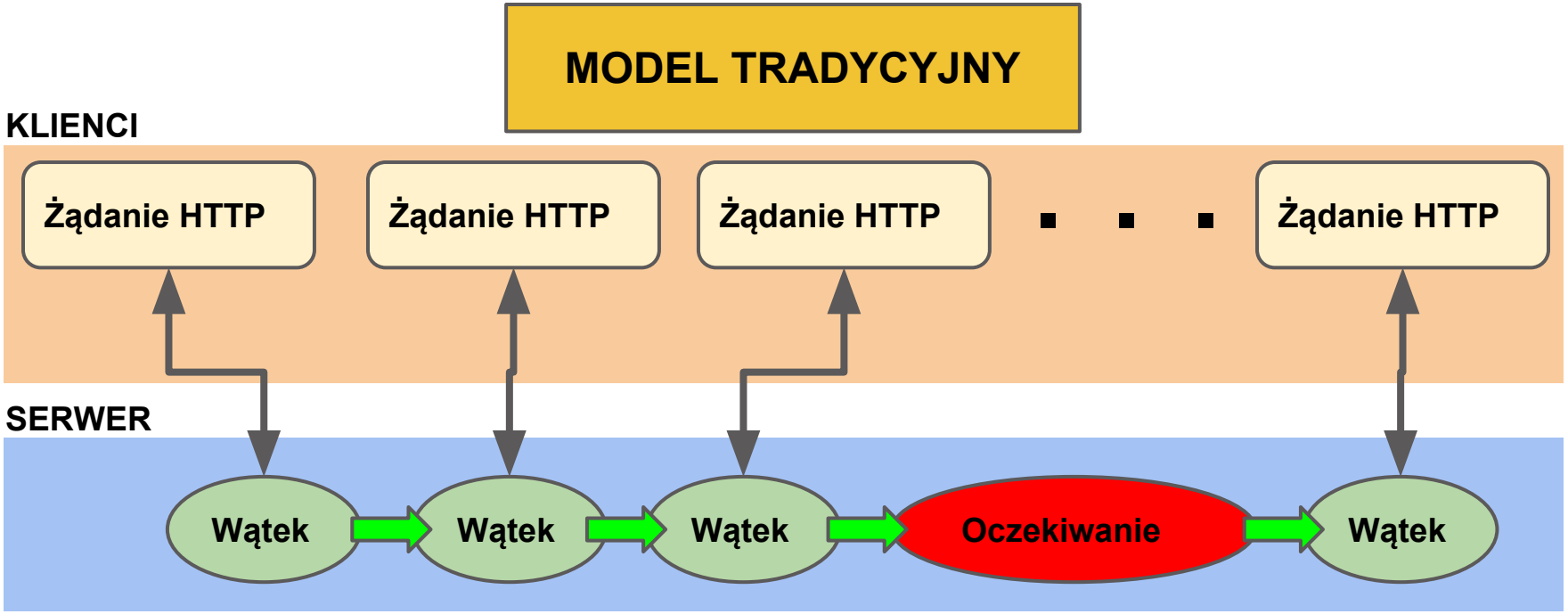
HTTP

CLIENT SIDE



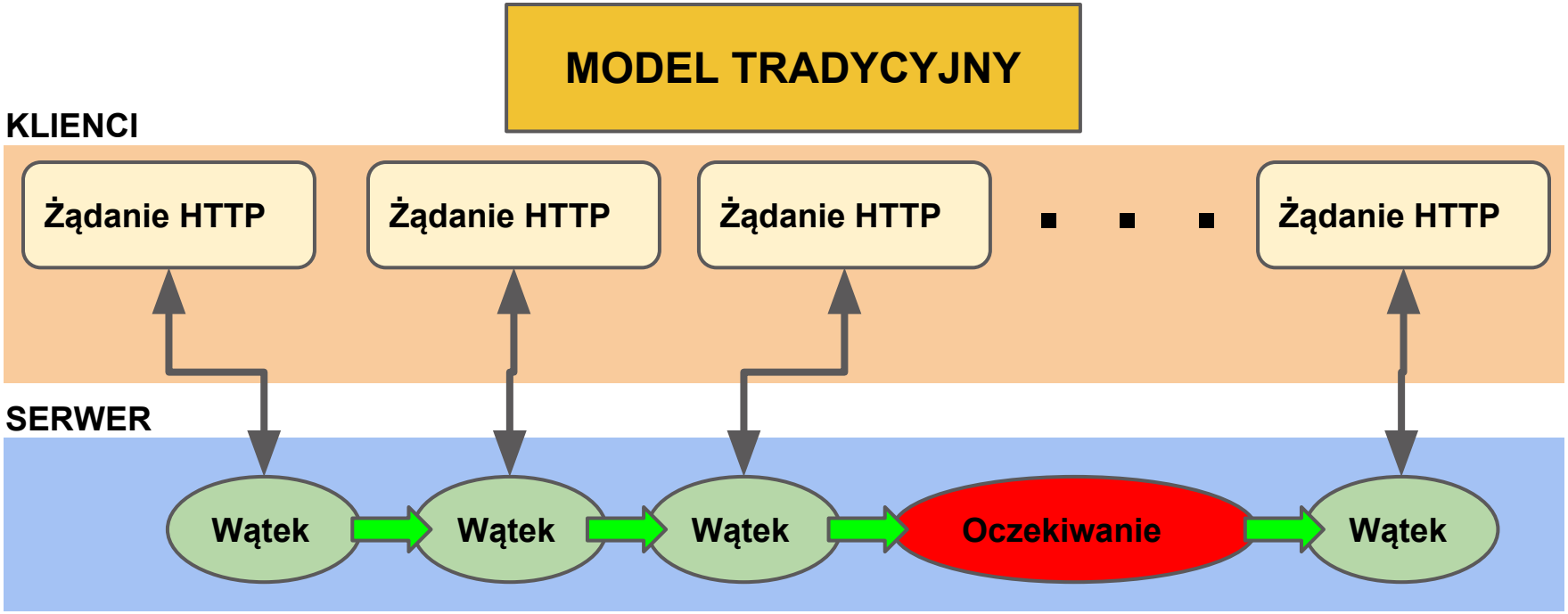
UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Obsługa żądań



W tym modelu serwer do obsługi każdego żadania musi stworzyć osobny wątek z ograniczonej puli jaką może obsłużyć CPU. Powoduje to że przy dużej ilości żądań niektóre z nich muszą czekać w "kolejce" na zrealizowanie. W wyniku czego serwer zaczyna zaczyna wolniej działać co wydłuża czas oczekiwania na odpowiedź.

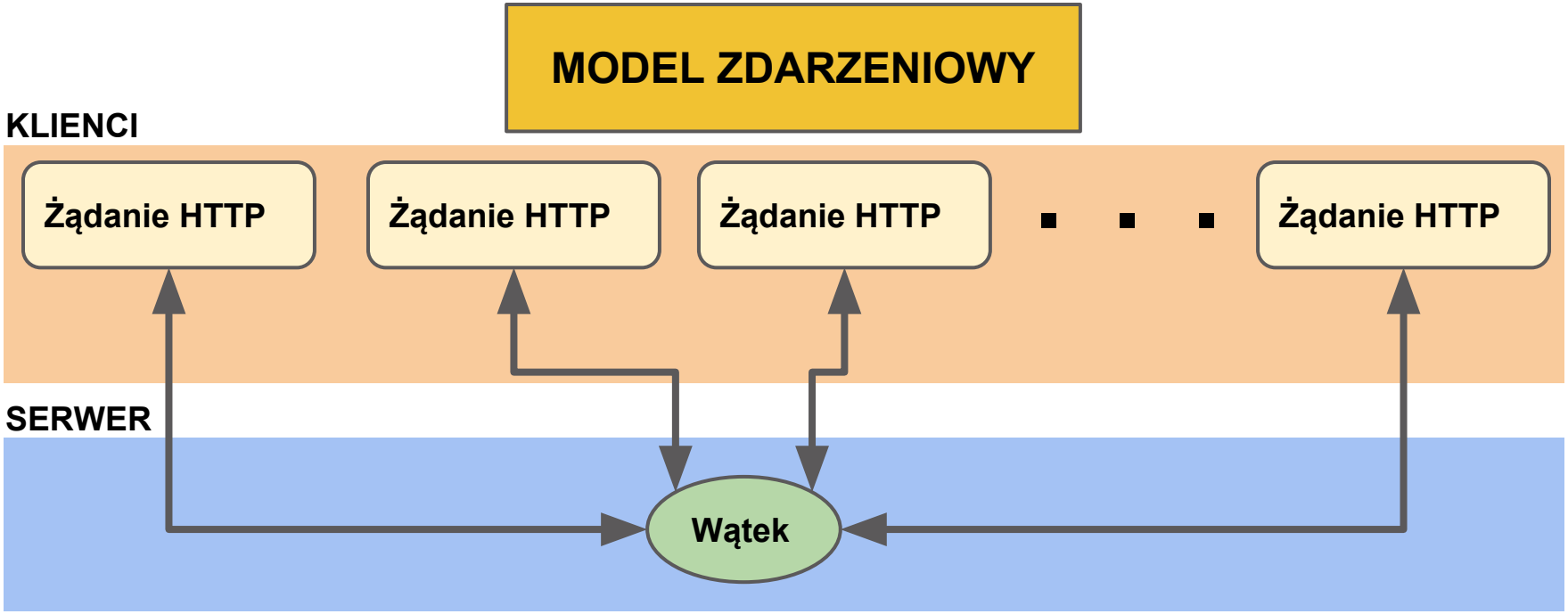
Obsługa żądań



Przykład:
Dla systemu z 8GB pamięci RAM przydzialającego 2MB pamięci na wątek możemy obsłużyć maksymalnie w tym samym czasie 4000 żądań (w rzeczywistości jest to mniej ponieważ zużywamy jeszcze pamięć na inne operacje).

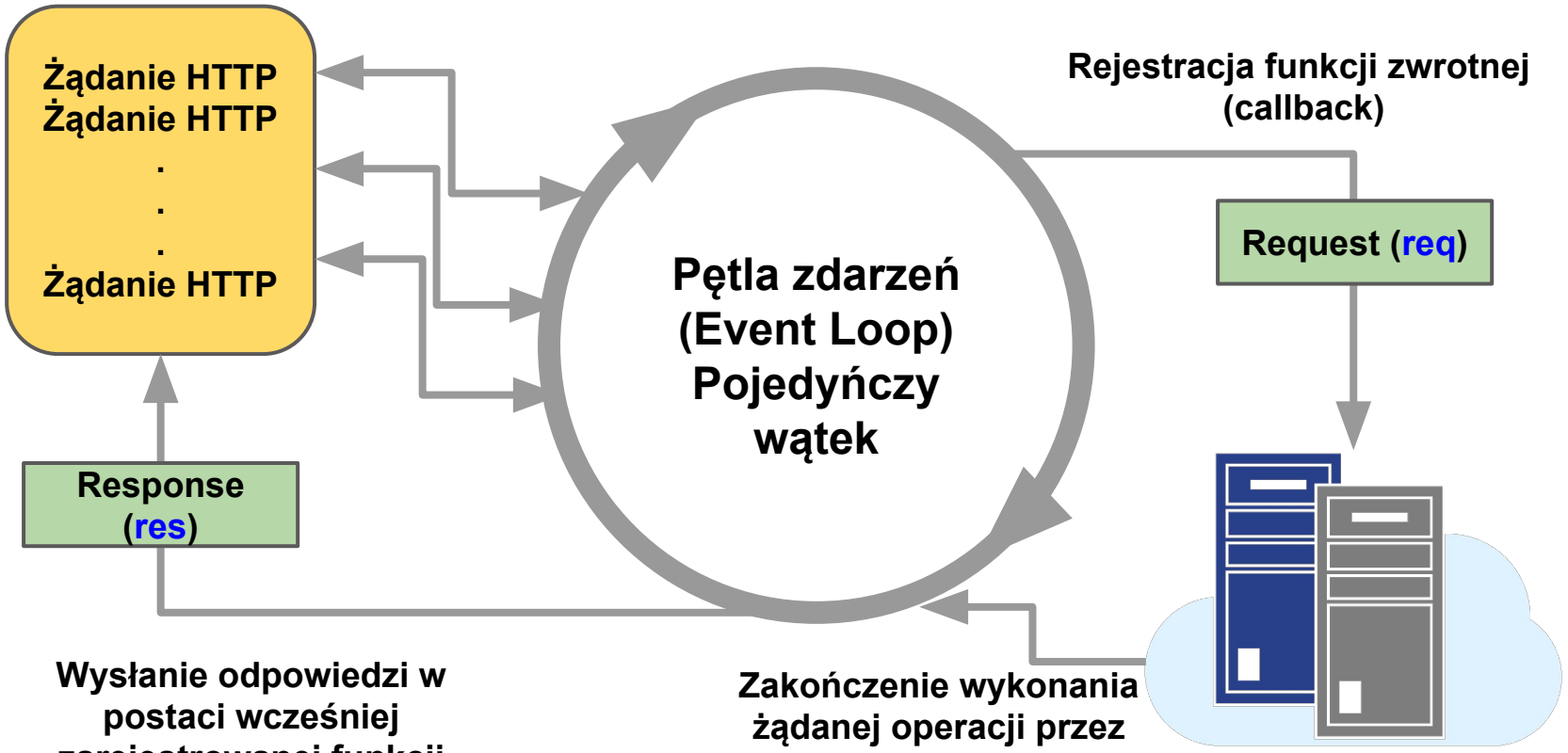
UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Obsługa żądań



W modelu zdarzeniowym Node.js wykorzystuje tylko jeden wątek do obsługi wielu żadąń, oraz “pętlę zdarzeń” co powoduje że aplikacja taka jest bardzo wydajna i skalowalna. W praktyce przy żadaniach które nie wymagają złożonych operacji obliczeniowych można obsłużyć nawet do 1 miliona żądań jednocześnie.

Pętla zdarzeń (Event loop)



Przykład prostego kodu (z strony nodejs.org)

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Prosty serwer przyjmujący żądania http

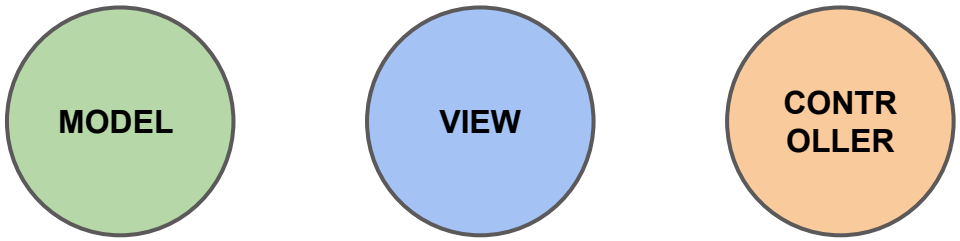
Załóżmy że powyższy kod jest zapisany w pliku “serwer.js”

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Model-View-Controller

Model-View-Controller (MVC) *[Model-Widok-Kontroler]* - jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:

- **Model (dane / logika)**
- **Widok (prezentacja danych)**
- **Kontroler (interakcja z użytkownikiem + sterowanie aplikacją)**

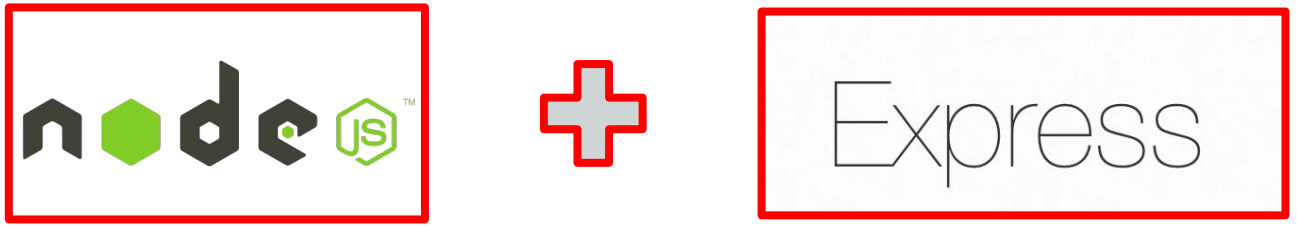


Można go zaimplementować bez użycia bibliotek czy specjalistycznych platform programistycznych, stosując jasne reguły podziału na konkretne komponenty w kodzie źródłowym. W ten sposób każdy komponent aplikacji można niezlaeżnie od siebie rozwijać, implementować i testować.

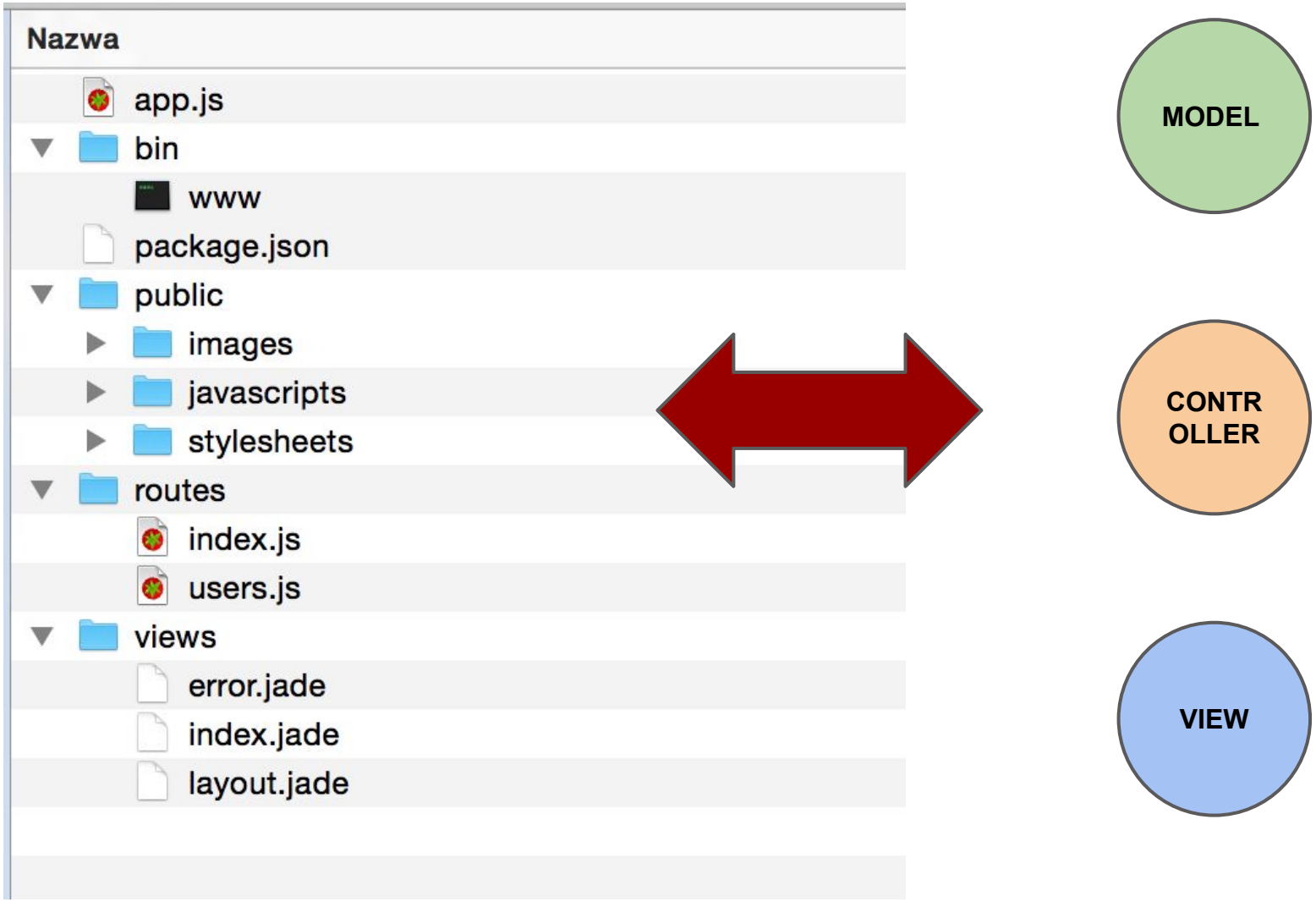
UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Express JS

Express.js jest środowiskiem które pozwala na tworzenie aplikacji internetowych w formie jednostronicowych oraz wielostroniowych witryn dostosowanych do wyświetlania na urządzeniach mobilnych oraz normalnych komputerach.



Express JS - struktura aplikacji



UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Express JS - Uruchamianie

Strona widoczna po uruchomieniu aplikacji “0”

Struktura aplikacji

Funkcje kontrolera odpowiedzialne za realizację konkretnych funkcjonalności znajdują się w kartotece routes. W wersji “0” plik zawiera tylko funkcję obsługującą wyświetlanie strony głownej aplikacji:

Plik: routes/index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

Funkcje routera

```
router.post('/formularz', function(req, res) {
    res.render('index', { title: req.body.fname });
});
```

W podobny sposób możemy korzystać z metody “POST” do przekazywania danych wpisanych w formularzu. Adres URI stanowiący pierwszy argument tej metody jest tzw. akcją która ma zostać wykonane po przesłaniu formularza.

Wszystkie dane zebrane z pól przesyłanego formularza są przekazane przez obiekt “req. body.NAZWA_POLA”. Aby pobrać dane posługujemy się notacją obiektową.

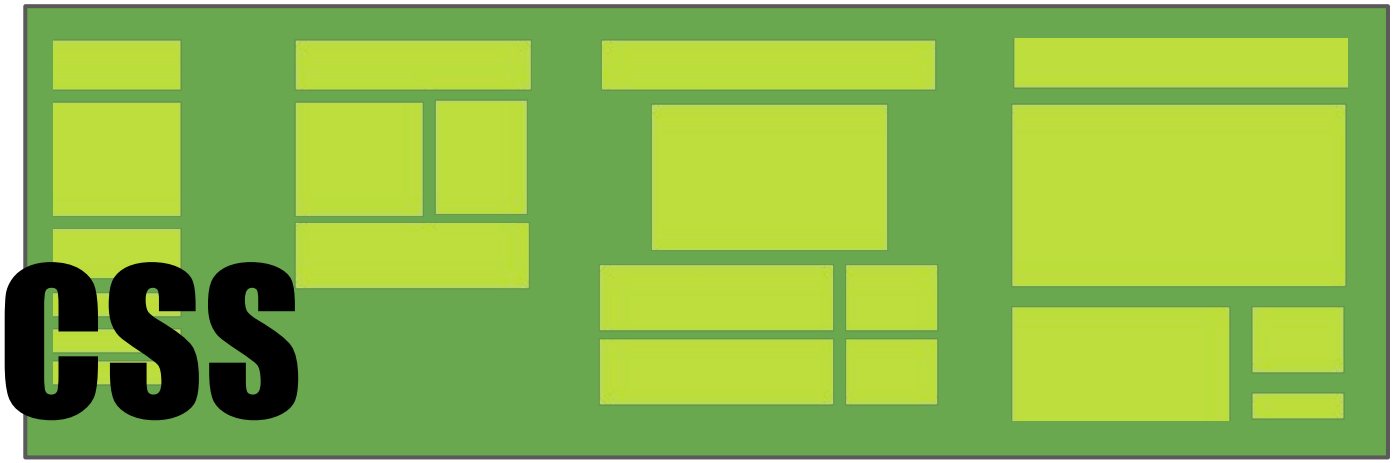
Formularz którego funkcja sterująca podana jest wyżej mógłby wyglądać następująco:

```
<form method='POST' action='/formularz'>
  <input type='text' name='fname'>
  <input type=submit name='Wyslij'>
</form>
```

Struktura modułowa HTML

ZASADA:

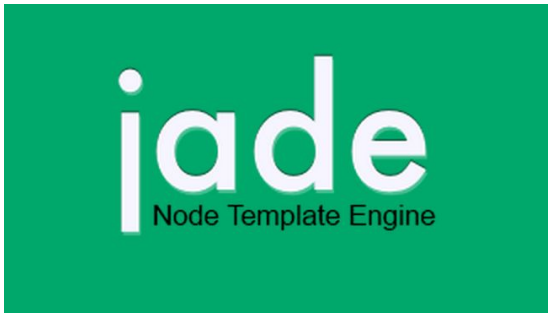
W ogólności zasada jaka powinna przyświecać tworzeniu stron zgodnie z metodologią RWD jest tworzenie jednego pliku HTML, a dla formatowania jego wygądu wiele “layautów” w CSS które będą zawierać odpowiednie reguły wyświetlania strony w zależności od rozdzielczości ekranu.



n x CSS

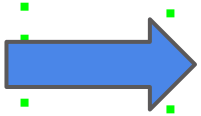
JADE

W środowisku Express.js w celu uproszczenia formy oraz usprawnienia tworzenia widoków wprowadzono nowy język pisania szablonów tzw. JADE (node template engine). Jest bardzo blisko językowi html, jednak w JADE nie występują znaczniki, a jedynie nazwy określające dane znacznik.



http://jade-lang.com/

```
<div>
    Element blokowy
</div>
```



```
div
  | Element blokowy
```

JADE

Dziedziczenie widoków:

```
<!doctype html>
<html>
  <head>
    <title>Article
Title</title>
  </head>
  <body>
    <h1>My Article</h1>
  </body>
</html>
```

```
//- layout.jade
doctype html
html
  head
    block title
      title Default title
  body
    block content
```

```
//- index.jade
extends ./layout.jade
block title
  title Article Title
block content
  h1 My Article
```

Taka organizacja widoków pozwala na stworzenie modularnej i hierarchicznej struktury strony.

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

Jednym z istotnych aspektów tworzenia nowoczesnych aplikacji internetowych jest przechowywania i wymiana danych. Najcześciej dane które są używane w aplikacjach są przechowywane w bazach danych.

Node.js (oraz wszystkie środowiska programistyczne takie jak Express. js) posiadają interfejsy do najpopularniejszych typów baz danych:

- **MySQL**

- **PostrgeSQL**

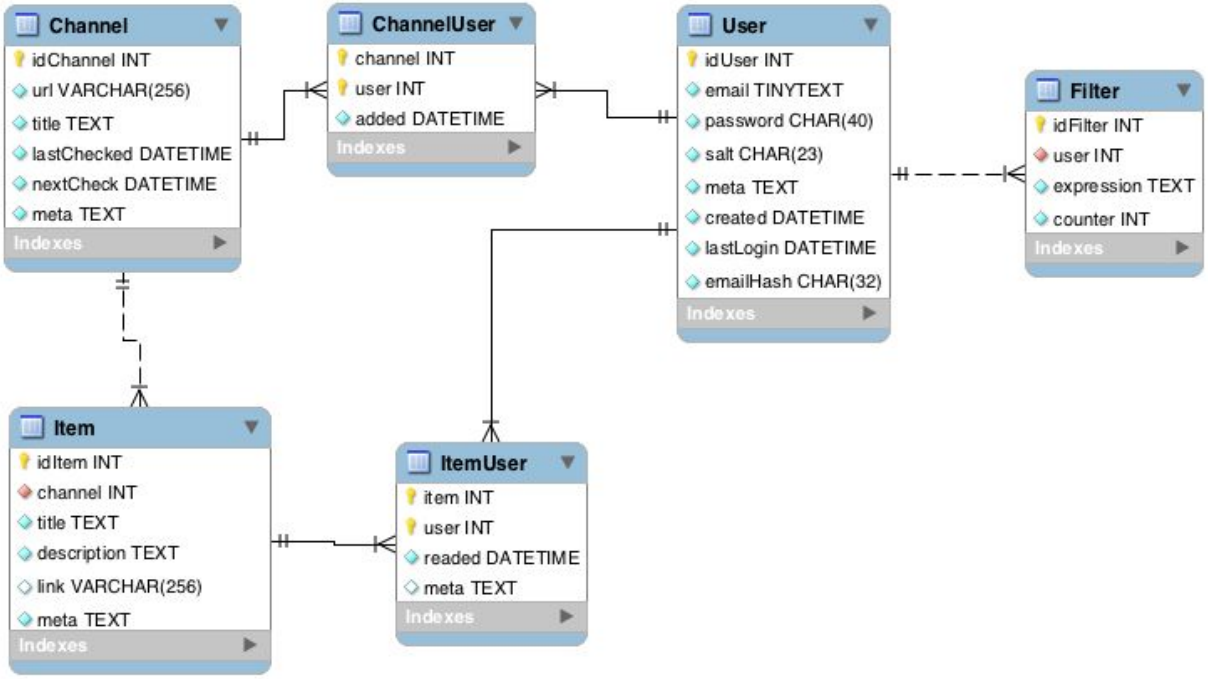
- **Oracle**

- **Moongo (noSQL)**

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

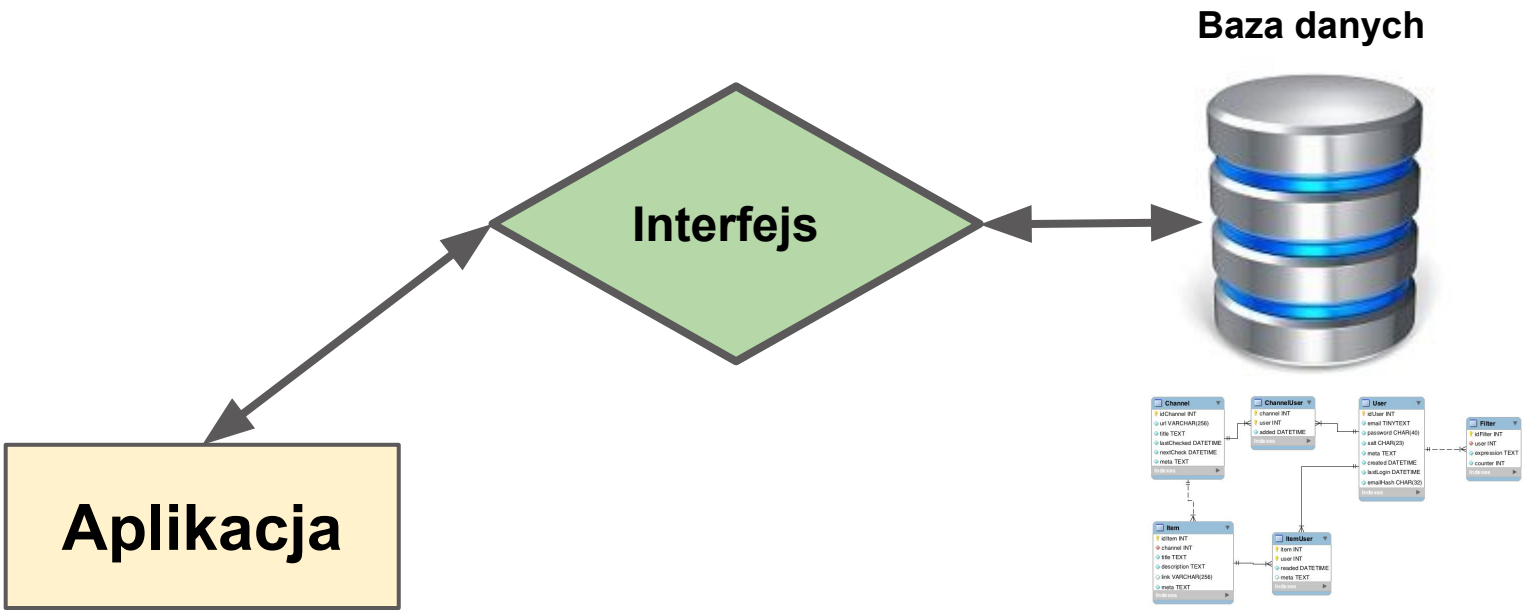
Baza danych “mySQL” należy do rozwiązań OpenSource z których można korzystać bezpłatne. Typowe bazy danych posiadają wiele encji które są ze sobą połączone za pomocą relacji (np. jeden-do-jednego, jeden-do-wielu, etc.).



UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

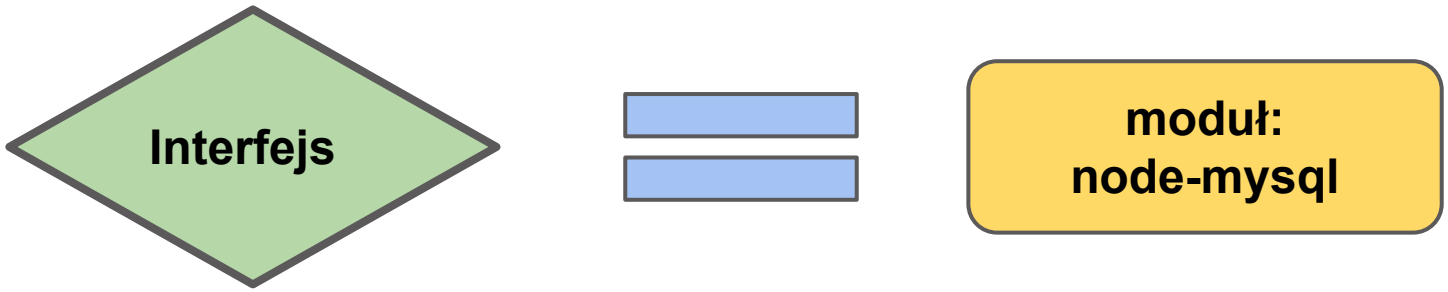
W środowisku Node.js aby korzystać z baz danych konieczne jest użycie interfejsu umożliwiającego połączenie z serwerem baz danych i obsługujących komunikację pomiędzy aplikacją a serwerem baz danych.



UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

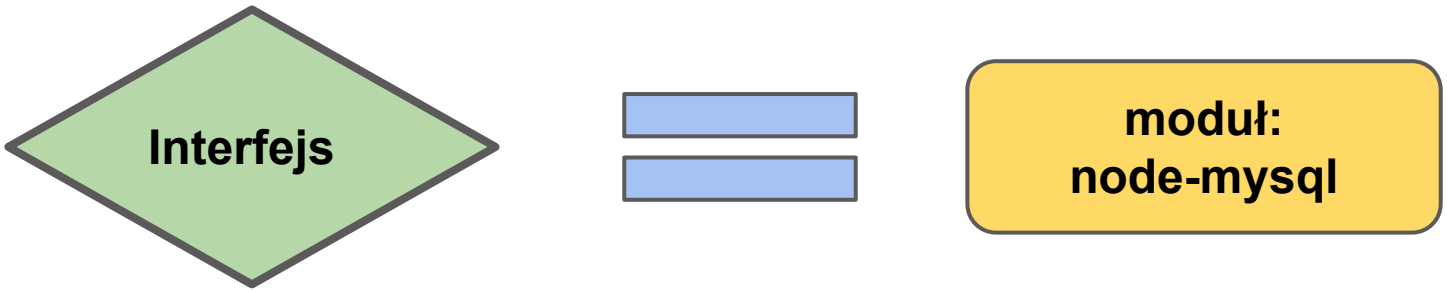
W środowisku Node.js aby korzystać z baz danych konieczne jest użycie interfejsu umożliwiającego połączenie z serwerem baz danych i obsługujących komunikację pomiędzy aplikacją a serwerem baz danych.



UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

W środowisku Node.js aby korzystać z baz danych konieczne jest użycie interfejsu umożliwiającego połączenie z serwerem baz danych i obsługujących komunikację pomiędzy aplikacją a serwerem baz danych.

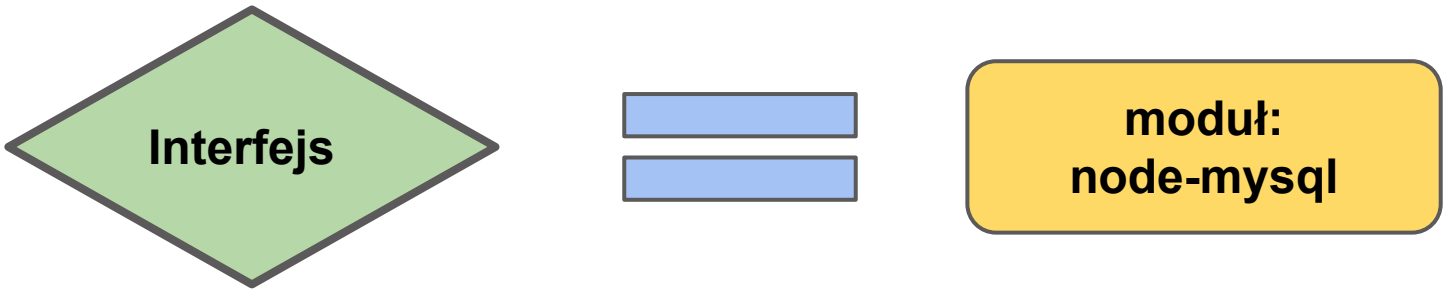


Instalacja pakietu za pomocą repozytorium NPM:

```
~/node> npm install mysql
```

Bazy danych

W środowisku Node.js aby korzystać z baz danych konieczne jest użycie interfejsu umożliwiającego połączenie z serwerem baz danych i obsługujących komunikację pomiędzy aplikacją a serwerem baz danych.



Instalacja pakietu za pomocą repozytorium NPM:

```
~/node> npm install mysql
```

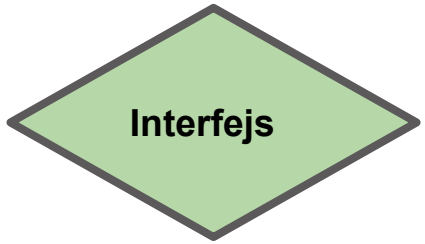
Pakiet dodajemy za pomoca metody require:

```
var mysql = require('mysql');
```

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

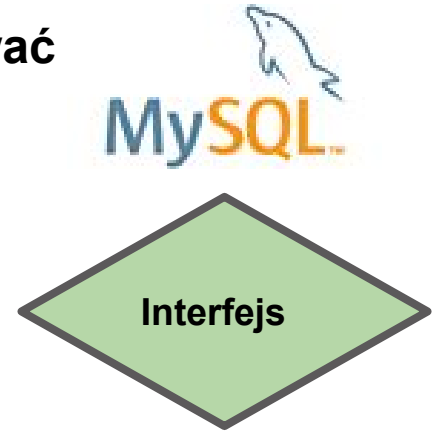
Bazy danych

Aby połączyć się z bazą danych należy najpierw zdefiniować parametry połączennia: nazwa serwera, port na którym baza nasłuchuje żądań, nazwa użytkownika, hasło, oraz nazwa bazy danych z którą chcemy się połączyć. W tym celu interfejs “node-mysql” udostępnia metodę .createConnection(), która nawiązuje połączenie:



Bazy danych

Aby połączyć się z bazą danych należy najpierw zdefiniować parametry połączennia: nazwa serwera, port na którym baza nasłuchuje żądań, nazwa użytkownika, hasło, oraz nazwa bazy danych z którą chcemy się połączyć. W tym celu interfejs “node-mysql” udostępnia metodę .createConnection(), która nawiązuje połączenie:

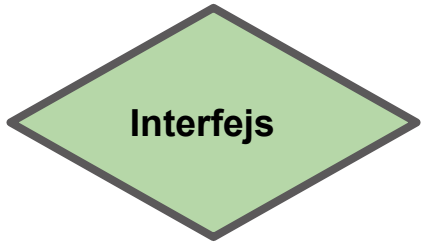



```
var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : 'test',
        database : 'meeting2015'
        });
connection.connect();
```

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

Aby połączyć się z bazą danych należy najpierw zdefiniować parametry połączennia: nazwa serwera, port na którym baza nasłuchuje żądań, nazwa użytkownika, hasło, oraz nazwa bazy danych z którą chcemy się połączyć. W tym celu interfejs "node-mysql" udostępnia metodę .createConnection(), która nawiązuje połączenie:

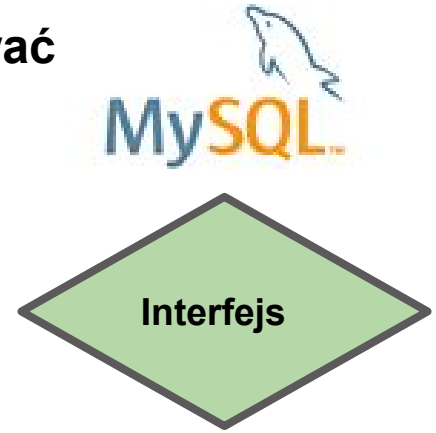



```
var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : 'test',
        database : 'meeting2015'
        });
connection.connect();
```

Metoda przyjmuje jako argument obiekt JSON w którym zapisane są parametry połączenia do serwera.

Bazy danych

Aby połączyć się z bazą danych należy najpierw zdefiniować parametry połączennia: nazwa serwera, port na którym baza nasłuchuje żądań, nazwa użytkownika, hasło, oraz nazwa bazy danych z którą chcemy się połączyć. W tym celu interfejs “node-mysql” udostępnia metodę .createConnection(), która nawiązuje połączenie:




```
var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : 'test',
        database : 'meeting2015'
        });
connection.connect();
```

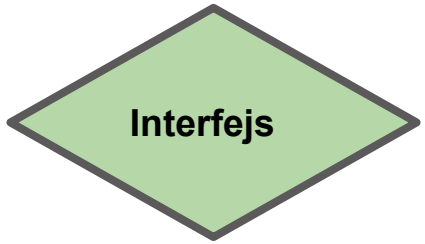
Inicjowanie połączenia

Metoda przyjmuje jako argument obiekt JSON w którym zapisane są parametry połączenia do serwera.

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

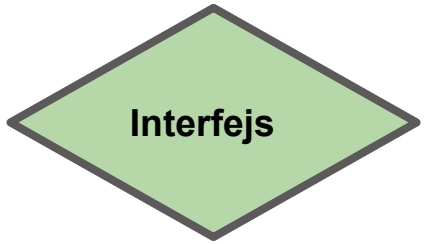
Od tego momentu mamy pełną możliwość wysyłania kwerend do serwera SQL w celu pobrania odpowiednich danych. Zapytanie formułujemy w postaci obiektu string:



Bazy danych

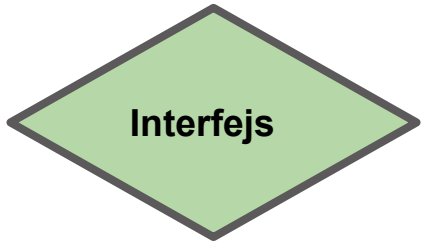
Od tego momentu mamy pełną możliwość wysyłania kwerend do serwera SQL w celu pobrania odpowiednich danych. Zapytanie formułujemy w postaci obiektu string:

```
var zapytanie = 'select * from users';
```



Bazy danych

Od tego momentu mamy pełną możliwość wysyłania kwerend do serwera SQL w celu pobrania odpowiednich danych. Zapytanie formułujemy w postaci obiektu string:



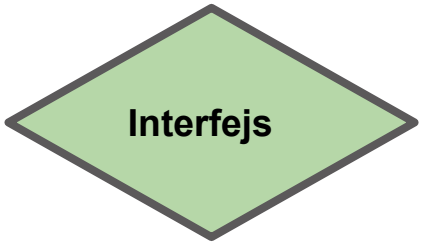

```
var zapytanie = 'select * from users';
```

Następnie zapytanie musi zostać wysłane do serwera SQL, za pomocą metody .query(), która przyjmuje dwa argumenty: pierwszy zapytanie, drugi to wywołanie zwrotne w którym będziemy odbierać dane:

```
connection.query(zapytanie, function(err, rows) {
            res.render('participants',{dane: rows});
        });
```

Bazy danych

Od tego momentu mamy pełną możliwość wysyłania kwerend do serwera SQL w celu pobrania odpowiednich danych. Zapytanie formułujemy w postaci obiektu string:



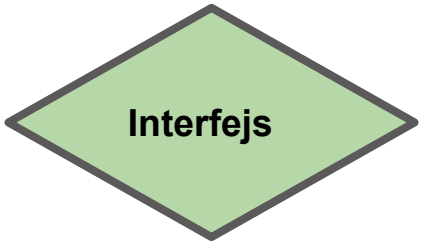

```
var zapytanie = 'select * from users';
```

Następnie zapytanie musi zostać wysłane do serwera SQL, za pomocą metody .query(), która przyjmuje dwa argumenty: pierwszy zapytanie, drugi to wywołanie zwrotne w którym będziemy odbierać dane:

```
connection.query(zapytanie, function(err, rows) {
            res.render('participants',{dane: rows});
        });
```

Bazy danych

Od tego momentu mamy pełną możliwość wysyłania kwerend do serwera SQL w celu pobrania odpowiednich danych. Zapytanie formułujemy w postaci obiektu string:




```
var zapytanie = 'select * from users';
```

Następnie zapytanie musi zostać wysłane do serwera SQL, za pomocą metody .query(), która przyjmuje dwa argumenty: pierwszy zapytanie, drugi to wywołanie zwrotne w którym będziemy odbierać dane:

```
connection.query(zapytanie, function(err, rows) {
            res.render('participants',{dane: rows});
        });
```

Serwer SQL zwraca dane w postaci tablicy wyników który jest argumentem w funkcji zwrotnej: “rows” (wiersze).

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

Tablica “rows” zawiera tyle wierszy ile zostało zwróconych w wyniku zapytania SQL do bazy. Tablica ta może zostać w całości przekazana do widoku:


```
connection.query(zapytanie, function(err, rows) {
            res.render('participants',{dane: rows});
        });
```

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

Bazy danych

Tablica “rows” zawiera tyle wierszy ile zostało zwróconych w wyniku zapytania SQL do bazy. Tablica ta może zostać w całości przekazana do widoku:


```
connection.query(zapytanie, function(err, rows) {
            res.render('participants',{dane: rows});
        });
```

W widoku możemy iterować po przekazanej tablicy i wyświetlać odpowiednia pola korzystając z notacji obiektowej:

```
#shortDescription
      ol
        each item in dane
          li
            b #{item.name} #{item.sname}
             | , #{item.affiliation}, #{item.country}
```

Bazy danych

Baza: na serwerze lokalnym (127.0.0.1)
Nazwa bazy: Personel
Tabela w bazie mySQL: dane

id	imie	nazwisko
1	Maciej	Adamczyk
2	Jan	Kowalski
3	Anna	Nowak

```
var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : 'test',
        database : 'Personel'
        });
connection.connect();
```

```
var zapytanie = 'select * from dane;
```

```
connection.query(zapytanie, function(err, rows) {
          console.log(rows[0].id + rows[0].imie + rows[0].nazwisko);
          console.log(rows[1].id + rows[1].imie + rows[1].nazwisko);
          console.log(rows[2].id + rows[2].imie + rows[2].nazwisko);
        });
```

UNIWERSYTET
JAGIELLOŃSKI
W KRAKOWIE

KONIEC WYKŁADU 11