Jan Tomczak 4TR

Sprawozdanie

Node – komunikacja z bazą danych i wysyłanie wiadomości e-mail z poziomu aplikacji Node.js

Co to jest Node.js?

Node – specjalne wieloplatformowe i open-source'owe środowisko, służące do wykonywania kodu JavaScript po stronie serwera. Jest to jedno z najpopularniejszych rozwiązań do budowania skalowanych oraz wydajnych aplikacji. Node łączy się z wieloma bazami danych, takich jak np.: MySQL, PostgresSQL czy MongoDB.

Node – Bazy Danych

1. MySQL.

Do połączenia się z bazą danych w MySQL jest potrzebne specjalne rozszerzenie do node.js → *mysql* lub *mysql2*, które pozwalają w łatwy i przyjemny sposób na połączenie się z istniejącą bazą. Instaluje je się za pomocą narzędzia *npm* (Node Package Manager), które jest instalowane wraz z naszym node.js'em.

npm install mysql2

Aby node.js użył tej biblioteki należy załączyć ją w naszym kodzie.

```
const mysql = require('mysql2')

//const mysql = require('mysql') -> Jeżeli korzystamy
```

Z zainstalowanym i złączonym pakietem można przejść do łączenia się z bazą. Potrzebne są do tego: *nazwa użytkownika*, *hasło użytkownika* oraz sama *nazwa bazy danych*.

```
let link = mysql.createConnection({
    host:'localhost',
    user:'root',
    password:'',
    database:'exampleDB'
})
```

Następujący kod pokazuje połączenie z bazą. Jeżeli nie posiadamy hasła do naszego użytkownika to należy pole *password* zostawić puste bez spacji!

2. PostgresSQL.

Aby połączyć się z bazą danych w PostgresSQL należy, podobnie jak ze wszystkimi innymi bazami, pobrać specjalną bibliotekę do naszego Node.js'a → pg.

npm install pg

Następnie należy załączyć naszą nowo zainstalowaną bibliotekę do kodu:

```
const mysql = require('pg');
```

Po tym kroku można już korzystać z bazy PostgresSQL.

Metody do wysyłania poczty w Node'zie.

1.SMTP (Simple Mail Transfer Protocol).

Służy do wysyłania wychodzących wiadomości e-mail przez sieci i jest jedną z najpopularniejszych rozwiązań, jeśli chodzi o przekazywanie wiadomości e-mail z jednego serwera na drugi. Podczas wysyłania wiadomości do jakiejkolwiek osoby, za pomocą klienta poczty e-mail, serwer poczty wychodzącej (SMTP) odpiera taką wiadomość i łączy się z systemem odbiorczym określonej osoby. Następnie obydwa serwery komunikują się przy użyciu specjalnych wytycznych określonych właśnie przez protokół SMTP.

Zalety:

Szczególną zaletą tego rozwiązania jest jego prostota oraz jego powszechne stosowanie. Dzięki SMTP zyskujemy szczegółową kontrolę nad każdym aspektem wysyłania wiadomości e-mail.

Wady:

Największą wadą SMTP jest jego podatność na np.: ataki DDoS, jakiekolwiek naruszenia bezpieczeństwa i integralności danych oraz phishing. Posiadanie własnego serwera poczty e-mail SMTP wiążę się z szczególną odpowiedzialnością za utrzymanie bezpieczeństwa. Jedną z mniejszych, ale również istotnych wad, może być powolne wysyłanie wiadomości. Wysyłanie takiej jednej wiadomości protokół SMTP wymaga szeroko zakrojonych przepływów zwrotnych między serwerami pocztowymi SMTP.

2.Interfejs API:

Jednym z powszechnych rozwiązań jest też interfejs API, używa się go do obsługi tworzenia wiadomości, wysyłania ich i ich dostarczania, zamiast samodzielnego zarządzania serwerem poczty e-mail. Oferuje on bogatą funkcjonalność, podczas szybkiego wysyłania dużej ilości wiadomości oraz możliwość jego szybkiej integralności. Przykładami takich interfejsów może być chociażby: *Postmark* czy *Amazon SES*.

Zalety:

Większość interfejsów posiada obszerną dokumentacje, a same są proste w konfigurowaniu i wykonywania na nich operacji. Posiadają one wysoką skalowalność, i są znacznie bezpieczniejsze w użyciu niż *SMTP*. Oszczędza się również na tym rozwiązaniu znaczne ilości czasu oraz koszta.

Wady:

Ogromną wadą korzystania z hostowanej usługi e-mail jest to, że polegamy na osobie trzeciej, która zajmuje się naszymi e-mailami. Zmusza nas to do poświęcenia czasu na zapoznaniem się z dostawcami i wybraniem dla siebie odpowiedniej usługi, która odpowiada naszym wymaganiom. Jest również problem z jeżeli chcemy, aby aplikacja powiadamiała użytkowników na innych kanałach to trzeba integrować każdy kanał osobno, co prowadzi do dodatkowego wysiłku, bądź straty zasobów.

3.Notification services (usługi powiadamiania):

Jest to rozwiązanie, które oferuje zaawansowane prymitywne interfejsy API obsługujące całą logikę związaną z typowymi powiadomieniami i umożliwiają dotarcie do użytkowników wieloma różnymi kanałami przy użyciu jednego jednolitego interfejsu API. Zazwyczaj pozwalają one na przyprowadzenie własnego dostawcy dla każdego kanału, jednak w przypadku poczty elektronicznej może to być nasz własny serwer SMTP, bądź też hostowany interfejs API poczty e-mail. Pozwala również na awaryjne przełączanie do różnych dostawców takiego samego kanału oraz na dodawanie wielu kanałów. Jednym z popularnych usług powiadamiania jest *Courier*.

Zalety:

Zaletą jest posiadanie całej wbudowanej logiki i systemu potrzebnego do obsługi powiadomień, można tu wymienić np.: wielokanałowe rejestrowanie, routing wielokanałowy czy grupowanie i analizowanie treści użytkowników. Dzięki usługi takiej jak *Courier* użytkownicy nietechniczni posiadają możliwość edytowania stylu treści, ją samą jak i również oznakowania marki wychodzących wiadomości e-mail

bez angażowania jakichkolwiek programistów, bądź też wdrażania kodu. W programie *Courier* da się łatwo przeglądać takie wiadomości.

Wady:

Wady są bardzo podobne co do poprzedniego punktu, ponieważ dalej korzystamy z interfejsu API – czyli polegamy na osobie trzeciej. Jest jednak ona znacznie mniejsza, dlatego że jesteśmy w stanie konfigurować przełączanie awaryjne dostawcy lub przełączanie innego dostawcy poczty e-mail. Przed podjęciem jakichkolwiek decyzji należy się zaznajomić z produktem oraz dostosowaniem go do naszych potrzeb.

Drugie Sprawozdanie

Sprawozdanie z wdrożenia aplikacji Node (5 zapytań).

1.7 sposobów na wdrożenie aplikacji Node'a:

- Kubernetes.
- Docker.
- App engine flex.
- Cloud – run.
- VPS.
- App engine standard (STD).
- Nasz własny serwer.

2.Rozwiązania, które nie wchodzą w gre:

Stosowanie jakichkolwiek innych języków niż *JavaScript* lub *TypeScript*, ponieważ aplikacja *Node.js* opiera się na środowisku uruchamiającym *JavaScript* co uniemożliwia zastosowania innych języków bez konwersji kodu. Nie powinno również wdrażać się aplikacji Node.js do innych środowisk.

3. Z jakich etapów składa się ogólny proces Wdrożenia?

1. Znalezienie odpowiadającego nam serwera.
2. Przygotowanie środowiska.
3. Konfiguracja domeny, aby wskazywała na nasz serwer.
4. Tworzenie nowego użytkownika (niewymagane – najbezpieczniejsze rozwiązanie).
5. Instalacja *Node'a*, *pm2* oraz *nginx*.
6. Implementacja kodu na nasz serwer.

7. Wystartowanie aplikacji przy użyciu narzędzia *pm2*, konfiguracja *nginx*, aby proxował ruch na aplikację *Node'a*.
8. Restartowanie *nginx*.

4.Jakie składowe środowiska uruchamiającego są wymagane?

Wymagane są: *Node.js* - centralna składniowa, baza danych – w zależności od potrzeby aplikacji oraz serwer *HTTP* – do obsługi żądań i przekierowywania ich do aplikacji.

5.Od czego będzie zależeć wybór rozwiązania Wdrożeniowego?

Będzie zależał on od wielu czynników, takich jak:

- Dostępność i budżet.
- Zapewnienie bezpieczeństwa.
- Skompilowanie samej aplikacji.
- Zapewnienia efektywnej pracy i wdrożenia.
- Własnych potrzeb.