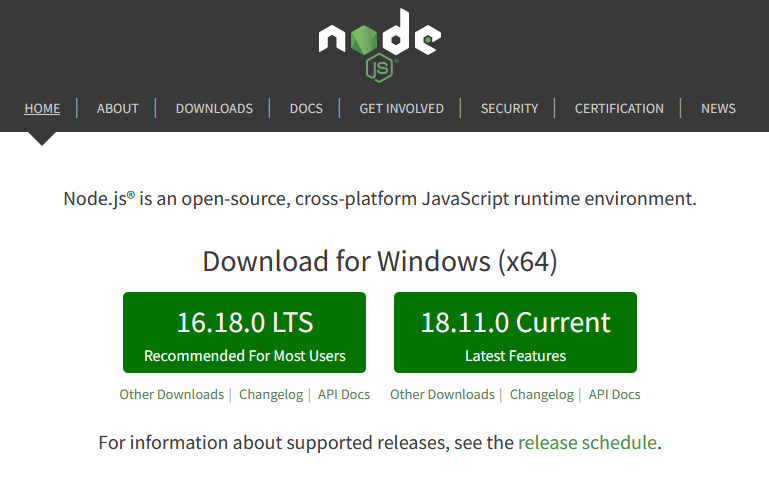
**2. Opis technologii wykorzystywanych w projekcie**

**2.1. Node.js - ekosystem niezbędny do budowy aplikacji w technologiach JavaScript**

Node.js [1] jest dostępnym dla wielu platform środowiskiem uruchomieniowym opartym na otwartym kodzie źródłowym. Stanowi główną bazę pod różnego rodzaju aplikacje wytwarzane w technologiach opartych na języku programowania JavaScript, co oznacza uruchamianie projektu (zarówno warstwy front-end jak i warstwy back-end) z poziomu konsoli właśnie za pomocą wyżej wymienionej technologii. Jej obecność jest niezbędna także wziąwszy pod uwagę potrzebę instalowania nowych, zewnętrznych komponentów, a także ich przechowywania w celu późniejszego użycia bez potrzeby każdorazowego pobierania z sieci.



Rys 1. Strona startowa strony Node.js.

*Źródło:* [*https://nodejs.org/en/*](https://nodejs.org/en/)

Skupienie się na roli Node.js w omawianym projekcie umożliwia bardziej zrozumieć sposób dodawania zewnętrznych bibliotek do projektu oraz opis implementacji systemu.

**1. Express.js – budowa serwera**

Express.js jest jednym z wyżej przytoczonych komponentów wymagających instalacji do systemu. W niniejszym projektu gra bardzo ważną rolę, ponieważ jego działanie stanowi fundament pracy warstwy serwerowej, która obsługuje funkcje, których działanie nie jest widoczne podczas obsługi aplikacji. Powyższa biblioteka dostarcza możliwości implementacji funkcji, które powinny się wykonać po otrzymaniu odpowiedniego żądania (ang. *request*) od warstwy front-end, czyli wykonać odpowiednie operacje oraz zwracać rezultat, który można wykorzystać do innych operacji w różnym stopniu. Względnie prosta struktura, według której inicjujemy właściwości nie wymusza na twórcy oprogramowania dokonywania wielu żmudnych zadań. Zapis poszczególnych funkcji opiera się wyłącznie na:

* określeniu polecenia CRUD (ang. *Create, Read, Update, Delete*), czyli jak warstwa back-end powinna zachować się podczas wywołania danej części kodu
* implementacji wyrażenia funkcyjnego zawierającego parametry do obsługi żądania (ang. *request*) oraz odpowiedzi (ang. *response*).
* Zakodowaniu ciała wyżej wymienionego bloku odpowiednim algorytmem, który winien zostać wykonaniu przy jego wywołaniu wraz z zapisami odnoszącymi się, co docelowo zostanie pobrane jako dane wejściowe (użycie z parametrem żądania) oraz co będzie stanowić informację zwrotną (użycie z parametrem odpowiedzi).

Nie istnieje także konieczność tworzenia rozległej sieci omawianej warstwy aplikacji, o ile programiście nie zależy na jej skrupulatnym uporządkowaniu, dzięki czemu mniej wymagający system nie będzie wymagał poświęcenia czasu oraz zasobów w nadmiarze.

**2. Body-Parser – operacje na przesyłanych obiektach po stronie serwera**

Body-parser jest zewnętrzną biblioteką, bez której warstwa back-end nie może przetworzyć danych wprowadzanej z warstwy od strony klienta. Sprawdza się w przypadku wierszy w formacie JSON pobieranych z warstwy front-end.

**3. JWT – autoryzacja**

JWT (pełna nazwa: JsonWebToken) jest biblioteką przeznaczoną do tworzenia szyfrogramów przydatnych przy sekcji logowania czy też sekcji zmiany hasła, które następnie podlegają weryfikacji przy wykonywaniu odpowiednich operacji. Wspomniany wcześniej zabieg ma na celu zabezpieczenie działania systemu przed potencjalną ingerencją ze stron osób trzecich, czyli włamaniem.

W praktyce praca JWT opiera się na wytwarzaniu szyfrogramów, czyli tokenów dla każdej operacji (np. podczas udanej próby zalogowania do systemu, czy podejścia do odzyskiwania hasła), które są z reguły przeznaczone do identyfikacji dokonywanej operacji. Częstym takim przypadkiem jest użycie wygenerowanego ciągu znaków w plikach przechowujących dane niezbędne do obsługi zalogowanego użytkownika, czyli ciasteczkach (ang. *cookies*).

**4. Bcrypt.js – zabezpieczenie hasła przez szyfrowanie**

Poważnym potencjalnym zagrożeniem ze strony osób niepożądanych jest dostęp do haseł użytkowników. Z pomocą przychodzi biblioteka Bcrypt.js, dzięki której w bazie danych można zapisać nie ciąg znaków podany w procesie rejestracji przez użytkownika końcowego, a wcześniej w warstwie back-end zaszyfrować dany ciąg z użyciem kosztu wytwarzania (ang. *cost factory*).

Szyfrowanie zachodzi z użyciem metody „hashSync()”, gdzie jako parametry wchodzą: hasło, które powinno być poddane zaszyfrowaniu, a także liczba wskazująca, ile razy powinniśmy zaszyfrować dany ciąg znaków.

W ramach omawianej technologii, dokonujemy też porównania dwóch ciągów znaków, czy są identyczne (na podstawie ich szyfrogramów). Sprawdza się to przy identyfikacji użytkownika podczas próby logowania do systemu.

**5. Socket.io – zabezpieczenie hasła przez szyfrowanie**

Socket.io jest biblioteką umożliwiającą dwukierunkową komunikację z minimalnym opóźnieniem dla każdej platformy między użytkownikami w czasie rzeczywistym [2].

Fundamentem działania funkcjonalności jest wywołanie instancji serwera, na bazie którego można wykonywać dowolne operacje związane z powyżej wymienioną komunikacją. Wewnątrz umieszczamy kod lub zmienną reprezentującą utworzenie serwera dla warstwy backendowej wykreowanej w technologii Express.js oraz dane pozwalające obejść zabezpieczenia CORS dla podanego adresu URL warstwy front-end oraz metody, które biblioteka powinna obsługiwać (najważniejsze z nich to „GET” oraz „POST”).

**6. Nodemailer – komunikacja za pośrednictwem poczty e-mail**

Nodemailer [3] jest modułem dla aplikacji pisanych w technologii Node.js, którego zadaniem jest nadawanie wiadomości e-mail do użytkowników danego systemu.

Jego podstawowa obsługa nie wydaję się być skomplikowana, o ile zrozumiała jest struktura typowej wiadomości e-mail. Początkowo należy ustalić transport, a dokładnie serwis nadawcy oferujący usługi związane z pocztą internetową (ang. *e-mail*) oraz adres i hasło nadawcy, od którego oficjalnie powinna wychodzić wyżej opisana korespondencja.

Sama procedura wysyłania pojedynczej wiadomości polega na wywołaniu metody dla wywołanego wcześniej transportu, „sendMail()”, w miejsce parametru (wewnątrz nawiasu) winien się znaleźć obiekt zawierający właściwości: adres poczty elektronicznej nadawcy, adres odbiorcy, tytuł, jaki zostanie wyświetlony na poczcie, oraz treść w formie tekstu bądź kodu w języku znaczników HTML (ang. *Hyper-Text Markup Language*).

Jeżeli wszystkie wymagane dane zostały zaimplementowane prawidłowo oraz algorytm wykonał się bez problemów, odbiorca powinien otrzymać wiadomość.

**2.2.React - programowanie warstwy aplikacji od strony klienta**

React jest biblioteką przeznaczoną do budowania interfejsów użytkownika [4] opierającą się na fundamentach języka JavaScript. Została ona wypuszczona do użytku w roku 2013 przez firmę Facebook oraz funkcjonuje na zasadach wolnego oprogramowania (ang. *Open Source*), dzięki czemu dana biblioteka może być na bieżąco udoskonalana przez szerokie grono programistów zainteresowanych działaniem na jej rzecz. Jej kod źródłowy jest dostępny do wglądu.

Głównym i najważniejszym elementem projektu wykreowanego w powyższej technologii jest komponent, który dzięki nieskomplikowanej i względnie liberalnej strukturze tworzenia aplikacji można budować według własnych preferencji, w postaci klasy bądź funkcji.

Wykonywanie bardziej zaawansowanych czynności w powyższej technologii ułatwiają dedykowane dla React biblioteki, które można zainstalować za pomocą środowiska „npm” dostępnego razem z ekosystemem „Node.js”. W omawianej aplikacji niezbędne okażą się:

* react-router-dom: z jej pomocą zaimplementowane zostaje przełączanie komponentów oraz relacje między sobą. W niektórych przypadkach owa biblioteka umożliwia przenoszenie konkretnych danych z poprzedniego komponentu do następnego. Dostarcza ona także narzędzi do nawigowania między komponentami między sobą oraz przenoszenia danych w postaci obiektu podczas zmiany podstrony, jeśli nie zachodzi ona na zasadzie relacji „rodzic – dziecko”, czyli nie następuje otwarcie komponentu nadrzędnego.
* react-cookies: głównym zadaniem tego elementu jest tworzenie oraz odtwarzanie tzw. ciasteczek (ang. *cookies*), czyli plików, które są wytwarzane na potrzeby danej sesji (w przypadku omawianego systemu logowania na poszczególne konto), w przeglądarce internetowej można je odnaleźć w narzędziach deweloperskich w zakładce „Aplikacja” lub „Dane” w rekordzie „cookies” lub „ciasteczka”. Ważną funkcjonalnością jest także pobieranie konkretnych danych do wykorzystania w kodzie aplikacji.
* react-hook-form: biblioteka ułatwiająca pracę z formularzami wykorzystywanymi w projekcie do przetwarzania danych wejściowych podawanych bezpośrednio przez użytkownika końcowego. Wykorzystanie jej ułatwia ich zapis wyników z poszczególnych za pomocą wbudowanej komendy „register” do obiektu w formacie JSON, więc nie istnieje potrzeba tworzenia oddzielnej zmiennej do przechowywania pojedynczych wpisów. Ponadto ostateczny obiekt zostaje wygenerowany po zatwierdzeniu zmian (ang. *submitting*) w formularzu.
* react-icons: biblioteka dostarczająca szeroką gamę ikon gotowych do wykorzystania w projekcie.
* react-finnhub: Wersja dostosowana do struktur React do przetwarzania danych z interfejsu API od Finnhub.io.
* react-chartjs2: Zewnętrzna biblioteka do zwracania grupy danych na bazie wykresów

Stylizowanie treści w sposób bezkolizyjny dla różnych komponentów jest możliwe przy użyciu biblioteki „styled-components”, którą należy zainstalować za pośrednictwem środowiska „npm” zawartego w ekosystemie Node.js.

**2.3.PostgreSQL – baza danych**

„PostgreSQL jest potężnym, opartym na otwartym kodzie źródłowym systemem dla obiektowo-relacyjnych baz danych”[5]. Jak sama nazwa wskazuje, powyższa technologia opiera się na języku SQL - przeznaczonym do budowy baz danych opartych na relacjach.

Jej wykorzystanie jest fundamentem do implementacji bazy danych dla systemu w przejrzysty sposób. Tworzenie przestrzeni do przechowywania danych polega na zasadach działania języka SQL.

**2.4. Stripe - System płatności**

**„**Stripe jest kompletem interfejsów zewnętrznych zasilających procesy płatności internetowych lub komercyjne rozwiązania dla biznesów internetowych wszelkich rozmiarów” [6]. Głównym zadaniem niniejszego narzędzia jest umożliwienie integracji dowolnego systemu informatycznego z algorytmami przetwarzającymi transakcje on-line, przykładowo aplikację internetową, czy mobilną.

Technologia została wydana w 2010 roku przez firmę Stripe Inc., irlandzko-amerykańskie przedsiębiorstwo zajmujące się serwisami oraz oprogramowaniem do spraw finansowych [7].

Wykorzystanie tego systemu w owej pracy umożliwia operowanie danymi karty płatniczej wprowadzonymi przez użytkownika (tworzenie tokenu płatności oraz metody płatności za nabyty instrument). Następnie w podobny sposób podlega zapisowy klient, który jeszcze nie był dodany do bazy Stripe dla jednego konta podpiętego do warstwy back-end.

Dodatkowo, niezbędnym jest pobranie danych karty płatniczej, z której środki powinny pokryć koszty transakcji. Po uzyskaniu wszystkich potrzebnych pozycji dodajemy pojedynczą transakcję, w której zawieramy dane tj. ilość (kwota) pieniędzy będąca należnością za nabyty przedmiot transakcji, walutę, w jakiej dokonujemy zakupu oraz, w razie konieczności, funkcji dotyczących sposobu płatności bądź automatycznego doboru środka płatniczego.

W przypadku implementacji algorytmu dla pojedynczej transakcji (zakup jednego rodzaju instrumentów finansowych od jednej firmy), gdy transakcja zakończy się pomyślnie, zostaje wygenerowany przez platformę Stripe dokument w rodzaju dowodu zakupu.

**2.5. Zewnętrzne interfejsy API**

Różne podmioty specjalizujące się w utrzymywaniu danych odnośnie niszowych rozwiązań udostępniają odpłatnie lub nieodpłatnie interfejsy, które są możliwe do wykorzystywania w dowolnych celach, bardzo często występuje to w przypadków wytwarzania różnych aplikacji informatycznych.

Na potrzeby niniejszego systemu niezbędny jest użytek API od dostawców: Finnhub.io (https://finnhub.io/) oraz TwelveData (https://twelvedata.com/).

Pierwszy z podmiotów udostępnia obszerny interfejs z okazałą ilością danych, które można wykorzystać do przetwarzania informacji w aplikacji. Skupiwszy się na istotnych zasobach, pobieramy API z sekcji wiadomości, symboli wymiany walut wirtualnych oraz danych firm należących do amerykańskiej giełdy.

W poniższej pracy interfejs od firmy TwelveData dostarcza cen wybranego instrumentu zarówno dla sekcji giełdy jak i sekcji kryptowalut. Wynik zawiera 30 pozycji ostatnich odczytów.