FAKTURKI

Data Wydania : 27.11.2023

Fakturki Pro Dokumentacja Techniczna

Program do Fakturowania i Zarządzania Finansami dla Małych i Średnich Przedsiębiorstw

Wersja: 2.0

Autorzy :

Marcin Sikora
Jakub Kurzacz
Łukasz Biś
Damian Kostek
Tymoteusz Palak

Wprowadzenie

Cel dokumentacji:

Niniejsza dokumentacja ma na celu zapewnienie użytkownikom i administratorom programu do wystawiania faktur "Fakturki" kompleksowego źródła informacji niezbędnych do efektywnego korzystania z systemu. Została ona przygotowana w taki sposób, aby służyć jako krok po kroku przewodnik zarówno dla nowych użytkowników, którzy dopiero rozpoczynają pracę z "Fakturki"", jak i doświadczonych użytkowników, którzy szukają informacji na temat zaawansowanych funkcji programu. Dokumentacja ta jest również zasobem dla osób odpowiedzialnych za administrację i konfigurację systemu, zapewniającym wskazówki dotyczące zarządzania kontami, bezpieczeństwem oraz integracją z innymi systemami.

Opis programu

"Fakturki"" to nowoczesne narzędzie do zarządzania fakturami, zaprojektowane z myślą o małych i średnich przedsiębiorstwach. Umożliwia ono użytkownikom szybkie i łatwe tworzenie, wysyłanie oraz śledzenie faktur. Dzięki intuicyjnemu interfejsowi użytkownika oraz bogatemu zestawowi funkcji, takich jak zarządzanie danymi klientów, automatyzacja powtarzalnych faktur, obsługa wielu walut i języków, a także możliwość generowania raportów finansowych, program "Fakturki"" jest kompleksowym rozwiązaniem dla firm pragnących usprawnić swoje procesy fakturacji.

Publiczność

Dokumentacja ta jest przeznaczona dla:

- Nowych użytkowników, którzy nie mają wcześniejszego doświadczenia z "Fakturki"" i potrzebują podstawowych informacji na temat funkcjonalności programu.
- Doświadczonych użytkowników, którzy poszukują bardziej zaawansowanych instrukcji lub chcą poszerzyć swoją wiedzę na temat konkretnych funkcji.
- Administratorów systemu, którzy są odpowiedzialni za konfigurację i zarządzanie programem w środowisku pracy, w tym za zarządzanie uprawnieniami użytkowników i zabezpieczeniami.
- Programistów i integratorów systemów, którzy chcą dostosować lub zintegrować "Fakturki" z innymi aplikacjami za pomocą dostępnego API.

Instalacja i Konfiguracja

Wymagania wstępne

Zanim rozpoczniesz proces instalacji, upewnij się, że twoje środowisko pracy spełnia następujące wymagania:

- System operacyjny obsługujący Docker (np. Windows, macOS, Linux)
- Zainstalowany i skonfigurowany Docker
- Zainstalowany Git Bash (dla systemów Windows) lub terminal z zainstalowanym Git (dla macOS lub Linux)

Krok 1: Instalacja Dockera

Docker jest niezbędny do utworzenia izolowanego środowiska, w którym będzie działał program "FakturaPro". Jeśli nie masz jeszcze zainstalowanego Dockera, odwiedź oficjalną stronę Docker i postępuj zgodnie z instrukcjami instalacji dla swojego systemu operacyjnego.

Krok 2: Instalacja Git Bash (dla Windows)

Git Bash jest potrzebny do klonowania repozytorium i zarządzania wersjami kodu. Użytkownicy systemów Windows mogą pobrać Git Bash z oficjalnej strony Git. Użytkownicy macOS i Linux powinni już mieć zainstalowany Git w swoich systemach.

Krok 3: Tworzenie użytkownika

Zalecane jest uruchamianie Dockera jako nieprzywilejowany użytkownik. Aby to zrobić, możesz utworzyć nowego użytkownika w systemie lub dodać swoje istniejące konto do grupy docker (w systemach Unixowych). Instrukcje dotyczące dodawania użytkownika do grupy docker można znaleźć w dokumentacji Dockera.

Krok 4: Klonowanie repozytorium

Otwórz Git Bash lub terminal i przejdź do katalogu, w którym chcesz umieścić projekt. Użyj następującej komendy, aby sklonować repozytorium:**git clone https://github.com/LukaszBis/Fakturki-Project.git**

Krok 5: Otwieranie repozytorium

Po sklonowaniu repozytorium przejdź do nowo utworzonego katalogu projektu:**cd Fakturki-Project**

Krok 6: Uruchamianie Dockera

Zanim uruchomisz Docker, upewnij się, że plik Dockerfile oraz wszystkie konfiguracje środowiskowe są na swoim miejscu i poprawnie skonfigurowane. Następnie wykonaj poniższą komendę, aby zbudować i uruchomić kontener: **docker-compose up --build**

Ta komenda skompiluje obraz i uruchomi kontenery zdefiniowane w pliku docker-compose.yml.

Krok 7: Ładowanie aplikacji

Po zakończeniu procesu budowy i uruchomienia kontenerów, aplikacja "Fakturki" powinna być dostępna lokalnie. Możesz teraz otworzyć przeglądarkę i wpisać adres:**http://localhost:5713**

Pamiętaj, aby zastąpić 5713 numerem portu określonym w konfiguracji Dockera, jeśli jest inny.

Krok 8: Sprawdzenie działania aplikacji

Po załadowaniu aplikacji w przeglądarce, zaleca się przetestowanie podstawowych funkcji, takich jak logowanie, tworzenie nowej faktury i przeglądanie listy faktur, aby upewnić się, że wszystko działa prawidłowo.

Przewodnik Użytkownika: Rejestracja

Krok 1: Wprowadzenie podstawowych informacji

- **Adres e-mail:** Wpisz swój adres e-mail, który będzie używany jako Twój login oraz do komunikacji z systemem.
- **Kod pocztowy:** Podaj kod pocztowy związany z adresem Twojej firmy lub adresu domowego.
- **Hasło:** Stwórz silne hasło, które będzie używane do zabezpieczenia Twojego konta. Powinno składać się z wielkich i małych liter, cyfr oraz znaków specjalnych.
- **Powtórz hasło:** Wpisz ponownie swoje hasło, aby upewnić się, że nie ma błędów.

Krok 2: Wprowadzenie informacji adresowych

- **Miasto:** Wpisz nazwę miasta, w którym znajduje się Twoja firma lub dom.
- **Ulica:** Podaj ulicę, na której znajduje się Twoja firma lub miejsce zamieszkania.
- **Nr budynku:** Wpisz numer budynku.
- **Nr lokalu:** Jeśli dotyczy, podaj numer lokalu.

Krok 3: Wprowadzenie danych osobowych

- **Imię:** Wpisz swoje imię.
- **Nazwisko:** Wpisz swoje nazwisko.

Krok 4: Informacje dodatkowe

- **NIP:** Jeśli rejestrujesz konto firmowe, wprowadź swój NIP (Numer Identyfikacji Podatkowej).
- **Nr telefonu:** Podaj numer telefonu kontaktowego.
- **Nr konta:** Wpisz numer konta bankowego, który może być wykorzystany do transakcji finansowych.

Krok 5: Finalizacja rejestracji

- Po wypełnieniu wszystkich pól, sprawdź podane informacje pod kątem poprawności.
- Kliknij przycisk „Zarejestruj się”, aby utworzyć konto.

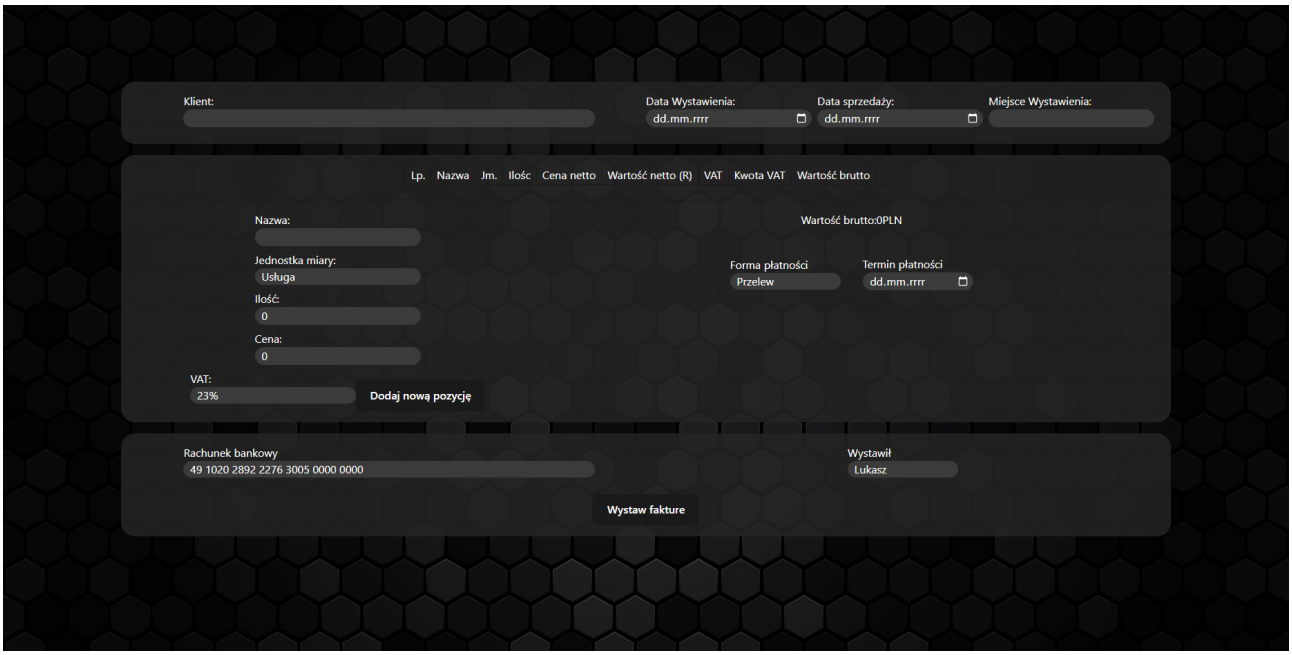
Krok 6: Weryfikacja konta

- Po zarejestrowaniu się, sprawdź swój e-mail w poszukiwaniu wiadomości z systemu "FAKTURKI", która zawiera link weryfikacyjny.
- Kliknij w link weryfikacyjny, aby aktywować swoje konto.

Krok 7: Logowanie

- Po aktywacji konta, możesz się zalogować, klikając na „Masz konto? Zaloguj się” i wprowadzając swoje dane logowania.

Przewodnik Użytkownika: Tworzenie Nowej Faktury



Krok 1: Wybór klienta

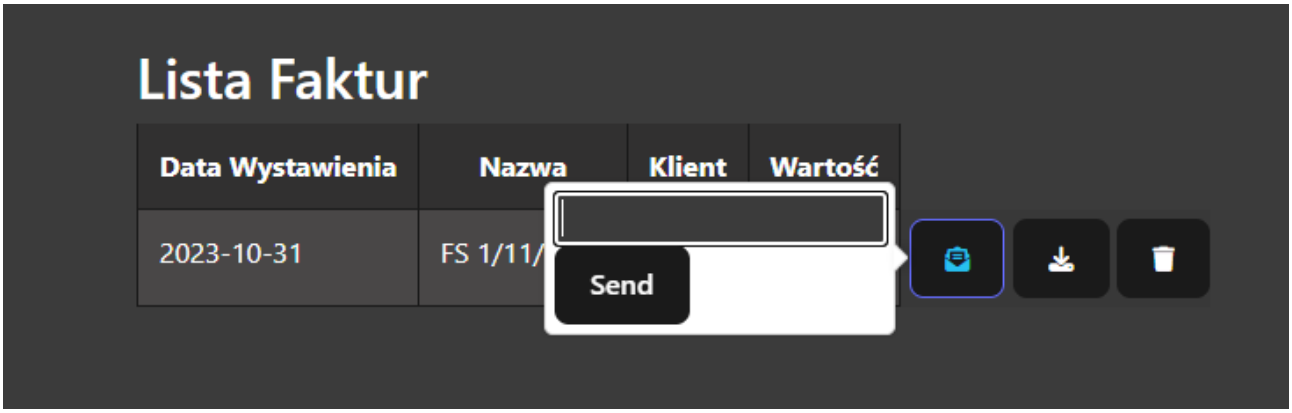
- W sekcji "Klient", wprowadź nazwę klienta, dla którego chcesz wystawić fakturę. Może być to pole wyszukiwania lub rozwijane menu, z którego możesz wybrać istniejącego klienta z bazy danych.

Krok 2: Wprowadzenie danych faktury

- W polach "Data Wystawienia", "Data sprzedaży" i "Miejsce Wystawienia", wprowadź odpowiednie daty oraz miejsce wystawienia faktury. Data sprzedaży może być taka sama jak data wystawienia lub różna, zależnie od wykonanej usługi czy sprzedanego towaru.

Krok 3: Dodawanie pozycji na fakturze

- W sekcji z listą pozycji, wprowadź nazwę towaru lub usługi w polu "Nazwa".
- Wybierz jednostkę miary (np. sztuki, godziny, metry) z rozwijanego menu "Jm.".



- Wpisz ilość sprzedawanego towaru lub zakres usługi w polu "Ilość".
- W polu "Cena netto" wprowadź cenę jednostkową bez VAT.
- "VAT" - wybierz stawkę podatku VAT z rozwijanego menu. W Polsce typowe stawki to 0%, 5%, 8% lub 23%.
- "Wartość netto", "Kwota VAT" i "Wartość brutto" to pola, które zwykle są automatycznie wyliczane na podstawie wprowadzonej ilości, ceny netto i stawki VAT.

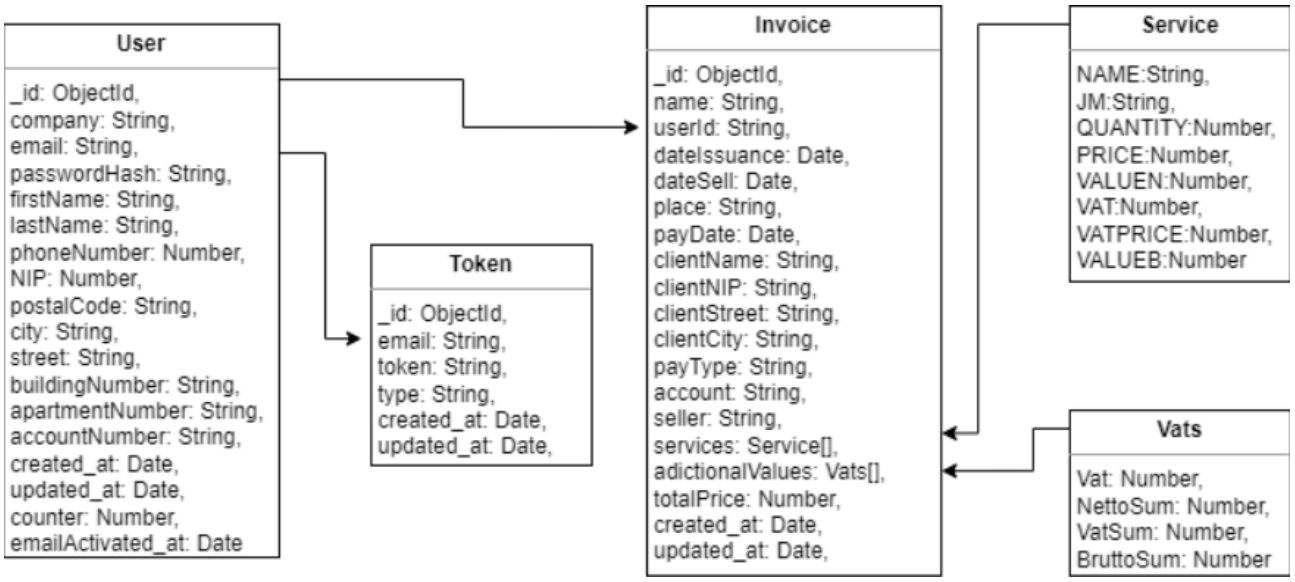
Krok 4: Szczegóły płatności

- W sekcji "Forma płatności" wybierz metodę, którą klient ma zastosować do zapłaty faktury, np. przelew bankowy.
- W polu "Termin płatności" ustaw datę, do której należy dokonać płatności.

Krok 5: Finalizacja i wystawienie faktury

- Sprawdź wszystkie wprowadzone informacje, aby upewnić się, że są poprawne.
- Kliknij przycisk "Wystaw fakturę" lub podobny, aby zakończyć proces tworzenia faktury.
- Po wystawieniu faktury, powinieneś mieć możliwość wydruku, zapisania do pliku (np. PDF) lub bezpośredniego wysłania do klienta drogą elektroniczną.

Diagram Klas



- **Użytkownik (User)**: Ta encja wydaje się reprezentować użytkowników systemu z atrybutami takimi jak identyfikator (id), firma (company), email, hasło (passwordHash), imię (firstName), nazwisko (lastName), numer telefonu (phoneNumber), numer identyfikacji podatkowej (NIP), szczegóły adresowe (kod pocztowy, miasto, ulica, numer budynku, numer mieszkania), numer konta (accountNumber), daty utworzenia i aktualizacji (created_at, updated_at) oraz flaga aktywacji email (emailActivated).
- **Token**: Ta encja może być związana z uwierzytelnianiem lub zarządzaniem sesją z atrybutami takimi jak identyfikator (id), email, token, typ (type), daty utworzenia i aktualizacji (created_at, updated_at).
- **Faktura (Invoice)**: Ta encja najprawdopodobniej reprezentuje szczegóły faktur generowanych w systemie. Posiada atrybuty takie jak identyfikator (id), nazwa (name), identyfikator użytkownika (userId), data faktury (dateIssuance), data sprzedaży (dateSale), termin płatności (dueDate), miejsce wystawienia (place), szczegóły klienta (imię, NIP, adres), typ płatności (payType), konto (account), informacje o sprzedawcy (seller), usługi (services), dodatkowe wartości (additionalValues), całkowita cena (totalPrice) oraz daty utworzenia i aktualizacji (created_at, updated_at).

- **Usługi (Service)**: Jest to prawdopodobnie tablica w obrębie encji Faktura, co sugeruje, że każda faktura może obejmować wiele usług. Posiada atrybuty takie jak nazwa (NAME), stanowisko pracy (JM), ilość (QUANTITY), cena (PRICE), wartość (VALUE), numer VAT (VATNumber), cena VAT (VATPRICE) i wartość całkowita (VALUEB).
- **Podatki VAT (Vats)**: Ta encja wydaje się reprezentować informacje o podatku od wartości dodanej (VAT), z atrybutami takimi jak numer VAT (Vat), suma netto (NettoSum), suma VAT (VatSum) i suma brutto (BruttoSum).

Dokumentacja API Serwera "FakturaPro"

Ogólne Informacje

Serwer "FakturaPro" obsługuje zarządzanie użytkownikami i fakturami dla aplikacji fakturowej. Poniżej przedstawiono dokumentację dostępnych endpointów.

Konfiguracja Middleware

- Body Parser JSON: Do parsowania ciała żądania JSON.
- Body Parser URL Encoded: Do parsowania ciała żądania z danymi formularza.
- CORS: Do obsługi Cross-Origin Resource Sharing, co pozwala na dostęp do API z różnych domen.

Endpointy

Rejestracja Nowego Użytkownika

- **URL: /register**
- **Metoda: POST**
  - **Body:**
  - email (String) - Adres email nowego użytkownika.
  - password (String) - Hasło dla nowego użytkownika.
  - confirmPassword (String) - Potwierdzenie hasła.
  - firstName (String) - Imię.
  - lastName (String) - Nazwisko.

- phoneNumber (String) - Numer telefonu.
- postalCode (String) - Kod pocztowy.
- city (String) - Miasto.
- street (String) - Ulica.
- buildingNumber (String) - Numer budynku.
- apartmentNumber (String) - Numer lokalu.
- NIP (String) - Numer identyfikacji podatkowej.
- accountNumber (String) - Numer konta bankowego.
- **Odpowiedzi:**
- 200 OK - Użytkownik został pomyślnie zarejestrowany.
- 400 Bad Request - Błąd walidacji danych.
- 409 Conflict - Użytkownik już istnieje.

Logowanie

- **URL: /login**
- **Metoda: POST**
  - **Body:**
  - email (String) - Adres email użytkownika.
  - password (String) - Hasło użytkownika.
  - **Odpowiedzi:**
  - 200 OK - Zalogowano pomyślnie, zwraca identyfikator użytkownika.
  - 401 Unauthorized - Błędne dane logowania.

Dodawanie Faktury

- **URL: /invoice**
- **Metoda: POST**
  - **Body:**
  - userId (String) - ID użytkownika wystawiającego fakturę.
  - client (Object) - Obiekt z danymi klienta.
  - items (Array) - Lista pozycji na fakturze.
  - **Odpowiedzi:**
  - 200 OK - Faktura została dodana pomyślnie.
  - 400 Bad Request - Błąd w danych faktury.

Pobieranie PDF Faktury

- **URL: /downloadPdf**
- **Metoda: GET**
  - **Parametry zapytania:**
  - id (String) - ID faktury do pobrania.
  - **Odpowiedzi:**
  - 200 OK - Zwraca plik PDF.
  - 404 Not Found - Faktura nie znaleziona.

Wysyłanie PDF Faktury

- **URL: /sendPdf**
- **Metoda: POST**
  - **Body:**
  - email (String) - Adres email, na który ma zostać wysłana faktura.
  - id (String) - ID faktury do wysłania.
  - **Odpowiedzi:**
  - 200 OK - Email z fakturą został wysłany.
  - 400 Bad Request - Błąd podczas wysyłania emaila.

Usuwanie Faktury

- **URL: /invoiceDelete**
- **Metoda: POST**
  - **Parametry zapytania:**
  - id (String) - ID faktury do usunięcia.
  - **Odpowiedzi:**
  - 200 OK - Faktura została usunięta.
  - 404 Not Found - Faktura nie znaleziona.

Uruchomienie Serwera

Serwer nasłuchuje na porcie 8080 i może być uruchomiony za pomocą komendy npm start.

Dokumentacja Modułów Mongoose dla „FakturaPro"

Konfiguracja Połączenia z Bazą Danych
Używa Mongoose do połączenia z MongoDB, używając nazwy usługi database z docker-compose.yml.

```
mongoose.connect(`mongodb://${dbHost}:27017/faktury`, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
```

Schematy Mongoose

User Schema
Przechowuje dane użytkownika, w tym informacje kontaktowe i adresowe oraz dane do logowania.

Client Schema
Zawiera informacje o klientach, takie jak REGON, NIP, nazwa firmy i adres.

Invoice Schema
Reprezentuje faktury z informacjami o płatnościach, usługach i szczegółach klienta.

Token Schema
Używany do przechowywania tokenów (np. aktywacyjnych, resetowania hasła) z powiązanym adresem email.

Modele
Każdy schemat jest kompilowany do modelu, który może być używany do interakcji z bazą danych:

const User = mongoose.model('User', userSchema);
const Client = mongoose.model('Client', clientSchema);
const Invoice = mongoose.model('Invoice', invoiceSchema);
const Token = mongoose.model('Token', tokenSchema);

Eksport

Modele są eksportowane, aby mogły być używane w innych częściach aplikacji:

module.exports = { User, Client, Invoice, Token };

Obsługa Błędów

Obsługa błędów połączenia z bazą danych jest zaimplementowana poprzez nasłuchiwanie na zdarzenia error i open.

Dokumentacja Modułu Zarządzania Fakturami dla „FakturaPro"

Przegląd

Moduł zarządzania fakturami **invoice** służy do obsługi operacji na fakturach w systemie "FakturaPro". Umożliwia tworzenie, usuwanie, wyszukiwanie i zliczanie faktur, jak również zarządzanie numerami NIP powiązanymi z fakturami użytkownika.
Funkcje Modułu
add(invoice)
Opis: Dodaje nową fakturę do bazy danych.
Parametry:

- **invoice** (Object) - Obiekt reprezentujący fakturę z wymaganymi polami.
- Zwraca:
- **newInvoice** (Object) - Obiekt dokumentu faktury po zapisaniu do bazy danych.

- Przykład użycia:

```
const invoiceData = { /* dane faktury */ };
const savedInvoice = await add(invoiceData);
```

remove(id)

Opis: Usuwa fakturę z bazy danych na podstawie jej identyfikatora.

Parametry:

- **id** (String) - ID faktury do usunięcia.
- Zwraca:
- **true** - Jeśli usunięcie się powiedzie.
- **false** - Jeśli wystąpi błąd.
- Przykład użycia:

```
const success = await remove(,5f5b8abcd123ef1234567890');
```

findAll(id)

Opis: Wyszukuje wszystkie faktury powiązane z danym użytkownikiem.

Parametry:

- **id** (String) - ID użytkownika, którego faktury mają być wyszukane.
- Zwraca:
- **invoices** (Array) - Tablica dokumentów faktur.
- Przykład użycia:

```
const userInvoices = await
findAll(,5f5b8abcd123ef1234567890');
```

count(userid, month, year)

Opis: Zlicza faktury użytkownika wydane w określonym miesiącu i roku.

Parametry:

- **userid** (String) - ID użytkownika.

- **month** (Number) - Miesiąc (0-11).
- **year** (Number) - Rok.
- Zwraca:
- **count** (Number) - Liczba faktur.
- Przykład użycia:

```
const invoiceCount = await
count('5f5b8abcd123ef1234567890', 7, 2023);
```

getById(id)
Opis: Pobiera pojedynczą fakturę na podstawie jej ID.
Parametry:

- **id** (String) - ID faktury do pobrania.
- Zwraca:
- **invoice** (Object) - Dokument faktury lub **null** jeśli nie znaleziono.
- Przykład użycia:

```
const invoice = await getById(,5f5b8abcd123ef1234567890');
```

getNIPArray(id)
Opis: Pobiera unikalne numery NIP dla wszystkich faktur użytkownika.
Parametry:

- **id** (String) - ID użytkownika.
- Zwraca:
- **nips** (Array) - Tablica unikalnych numerów NIP.
- Przykład użycia:

```
const nips = await getNIPArray(,5f5b8abcd123ef1234567890');
```

Obsługa Błędów

Wszystkie funkcje modułu są asynchroniczne i obsługują błędy poprzez wypisywanie ich do konsoli. Zaleca się dodatkowe

zabezpieczenia na wyższych poziomach aplikacji, aby obsłużyć te błędy w sposób, który jest bezpieczny dla użytkownika.

Eksportowane Funkcje

Moduł eksportuje wszystkie funkcje, aby mogły być używane przez inne części aplikacji.

```
module.exports = { add, remove, findAll, count, getById, getNIPArray };
```

Dokumentacja Modułu Mailowego dla „FakturaPro"

Przegląd

Moduł mailowy wykorzystuje **nodemailer** do wysyłania e-maili z aplikacji "FakturaPro". Pozwala na wysyłanie różnego rodzaju e-maili, w tym faktur w formacie PDF, linków do resetowania hasła i linków aktywacyjnych.

Konfiguracja nodemailer

Zdefiniowano konfigurację **transporter** używającą serwisu Gmail do wysyłania e-maili.

```
const transporter = nodemailer.createTransport({
  service: 'Gmail',
  auth: {
    user: 'TwojeFakturki@gmail.com',
    pass: 'hnfpxunonbpivtsg'
  }
```

```
});
```

Uwaga: Wartości **user** i **pass** powinny zostać przechowane w bezpiecznym miejscu, np. zmiennych środowiskowych.

Funkcje Modułu

sendInvoice(email, pdfBuffer, name)

Opis: Wysyła fakturę w formacie PDF na określony adres e-mail.

Parametry:

- **email** (String): Adres e-mail odbiorcy.
- **pdfBuffer** (Buffer): Bufor zawierający zawartość PDF faktury.
- **name** (String): Nazwa pliku PDF.

sendPasswordResetLink(email, token)

Opis: *Wysyła link do resetowania hasła na określony adres e-mail.*

Parametry:

- **email** (String): Adres e-mail odbiorcy.
- **token** (String): Token wykorzystywany do resetowania hasła.

sendActivationLink(email, token)

Opis: *Wysyła link aktywacyjny na określony adres e-mail.*

Parametry:

- **email** (String): Adres e-mail odbiorcy.
- **token** (String): Token wykorzystywany do aktywacji konta.

Wywołanie Funkcji

Każda z funkcji jest asynchroniczna i zwraca **Promise**.

Przykład wywołania funkcji:

```
await sendInvoice('klient@example.com', pdfBuffer, ‚Faktura123');
```

Obsługa Błędów

Podczas wysyłania e-maila, każda funkcja loguje ewentualne błędy do konsoli. W przyszłych wersjach zaleca się implementację lepszej obsługi błędów.

Bezpieczeństwo

Zaleca się zastosowanie dodatkowych środków bezpieczeństwa, takich jak ochrona hasła oraz tokenów.

Eksportowane Funkcje

Moduł eksportuje trzy funkcje:

module.exports = { sendInvoice, sendPasswordResetLink, sendActivationLink };

Dokumentacja Modułu Generowania i Wysyłki PDF dla „FakturaPro"

Przegląd:

Moduł ten jest odpowiedzialny za generowanie dokumentów PDF z faktur i wysyłanie ich do klientów. Wykorzystuje bibliotekę **html-pdf** do konwersji kodu HTML na PDF oraz moduł **mail** do wysyłania gotowych faktur jako załączników e-mail.

Zależności

- **html-pdf**: Biblioteka do tworzenia plików PDF z HTML.
- **mail**: Moduł do wysyłania e-maili z załącznikami.
- **PriceToPolishWords**: Konwerter kwot na słowa w języku polskim.
- **user**: Moduł zarządzania użytkownikami.
- **invoice**: Moduł zarządzania fakturami.

- **bir1**: Biblioteka do pobierania danych z rejestrów publicznych.

Funkcje:

generateHtml(id)

Opis: Generuje kod HTML dla faktury o określonym ID.
Parametry:

- **id** (String): ID faktury do wygenerowania.
- Zwraca:
- **html** (String): Kod HTML reprezentujący fakturę.

pdfBuffer(htmlCode)

Opis: Tworzy bufor PDF z podanego kodu HTML.
Parametry:

- **htmlCode** (String): Kod HTML do konwersji na PDF.
- Zwraca:
- **Promise<Buffer>**: Obietnica, która rozwiązuje się do bufora PDF.

downloadPdf(res, id, name)

Opis: Wysyła plik PDF jako odpowiedź HTTP, umożliwiając pobranie przez klienta.
Parametry:

- **res** (Response): Obiekt odpowiedzi Express.js.
- **id** (String): ID faktury do pobrania.
- **name** (String): Nazwa pliku PDF.
- Zwraca:
- Nic. Odpowiedź HTTP jest wysyłana bezpośrednio do klienta.

sendPdf(email, id, name)

Opis: Generuje PDF z faktury i wysyła go na podany adres e-mail.
Parametry:

- **email** (String): Adres e-mail odbiorcy.
- **id** (String): ID faktury do wysłania.
- **name** (String): Nazwa pliku PDF do wysłania.
- Zwraca:
- Nic. E-mail jest wysyłany bezpośrednio do odbiorcy.

Obsługa Błędów

Wszystkie funkcje asynchroniczne używają **try-catch** do obsługi wyjątków. Błędy są logowane do konsoli.

Eksportowane Funkcje

Moduł eksportuje dwie funkcje:

```
module.exports = { downloadPdf, sendPdf };
```

Przykłady Użycia

-Pobieranie PDF

```
app.get('/download-invoice/:id', async (req, res) => {
  const { id } = req.params;
  await downloadPdf(res, id, 'Faktura');
});
```

-Wysyłanie PDF e-mailem

```
app.post('/send-invoice', async (req, res) => {
  const { email, id } = req.body;
  await sendPdf(email, id, 'Faktura');
});
```

Dokumentacja Modułu Zarządzania Tokenami dla „FakturaPro"

Przegląd

Moduł **token** służy do zarządzania tokenami w aplikacji "FakturaPro". Oferuje funkcje do generowania, przechowywania, weryfikowania oraz usuwania tokenów związanych z różnymi działaniami użytkownika, takimi jak reset hasła czy aktywacja adresu e-mail.

Funkcje Modułu

addToken(type, email)

Opis: Dodaje nowy token do bazy danych dla określonego typu działania i adresu e-mail.
Parametry:

- **type** (String): Typ tokenu (np. 'password', 'email').
- **email** (String): Adres e-mail, dla którego token jest tworzony.
- Zwraca:
- **true**: Jeśli token został pomyślnie dodany.
- **false**: W przypadku wystąpienia błędu.

removeToken(type, email)

Opis: Usuwa token z bazy danych na podstawie typu działania i adresu e-mail.
Parametry:

- **type** (String): Typ tokenu.
- **email** (String): Adres e-mail, dla którego token jest usuwany.

generateToken()

Opis: Generuje losowy token.
Zwraca:

- **token** (String): Wygenerowany token.

getTokenByEmail(type, email)
Opis: Pobiera token na podstawie typu działania i adresu e-mail.
Parametry:

- **type** (String): Typ tokenu.
- **email** (String): Adres e-mail, dla którego token jest wyszukiwany.
- Zwraca:
- **token** (String): Znaleziony token.

getEmailByToken(type, token)
Opis: Pobiera adres e-mail na podstawie typu działania i tokenu.
Parametry:

- **type** (String): Typ tokenu.
- **token** (String): Token, dla którego adres e-mail jest wyszukiwany.
- Zwraca:
- **email** (String): Znaleziony adres e-mail.

checkToken(type, token)
Opis: Sprawdza, czy token istnieje w bazie danych.
Parametry:

- **type** (String): Typ tokenu.
- **token** (String): Token do sprawdzenia.
- Zwraca:
- **true**: Jeśli token istnieje.
- **false**: Jeśli token nie istnieje.

removeOld(type, min)

Opis: Usuwa stare tokeny, które zostały utworzone przed określoną liczbą minut.
Parametry:

- **type** (String): Typ tokenu.
- **min** (Number): Liczba minut, po których tokeny są usuwane.

Użycie

Moduł jest używany do zarządzania tokenami w różnych procesach aplikacji, takich jak weryfikacja e-maili i resetowanie haseł.

Eksportowane Funkcje

Moduł eksportuje następujące funkcje, aby mogły być używane w innych częściach aplikacji:

module.exports = { addToken, getTokenByEmail, getEmailByToken, checkToken, removeToken, removeOld };

Bezpieczeństwo

Zaleca się odpowiednie zabezpieczenie wszystkich tokenów i danych poufnych używanych w aplikacji, w szczególności przy ich przechowywaniu i przesyłaniu.

Dokumentacja Modułu Zarządzania Użytkownikami dla „FakturaPro"

Przegląd:

Moduł **user** jest odpowiedzialny za obsługę operacji związanych z użytkownikami w systemie "FakturaPro". Umożliwia tworzenie nowych kont użytkowników, autoryzację, zarządzanie hasłami, a także zarządzanie danymi osobowymi i firmowymi użytkowników.

Funkcje Modułu

add(...)

Opis: Tworzy nowego użytkownika w systemie.
Parametry:

- **company**, **firstName**, **email**, **password**, **postalCode**, **street**, **lastName**, **phoneNumber**, **city**, **buildingNumber**, **apartmentNumber**, **NIP**, **accountNumber**: Dane użytkownika.
- Zwraca:
- **user**: Utworzony obiekt użytkownika.

auth(id)

Opis: Pobiera dane użytkownika na podstawie jego ID.
Parametry:

- **id** (String): ID użytkownika.
- Zwraca:
- Obiekt użytkownika.

resetCounter(user)

Opis: Resetuje licznik wystawionych faktur użytkownika.
Parametry:

- **user**: Obiekt użytkownika.

increseCounter(user)

Opis: Zwiększa licznik wystawionych faktur użytkownika.
Parametry:

- **user**: Obiekt użytkownika.

changePassword(user, password)

Opis: Zmienia hasło użytkownika.

Parametry:

- **user**: Obiekt użytkownika.
- **password** (String): Nowe hasło.
- Zwraca:
- **true**: Jeśli zmiana hasła się powiedzie.

passwordCompare(passwordHash, password)

Opis: Porównuje podane hasło z zahashowanym hasłem w bazie danych.

Parametry:

- **passwordHash** (String): Hash hasła zapisanego w bazie danych.
- **password** (String): Hasło do sprawdzenia.
- Zwraca:
- **true**: Jeśli hasła się zgadzają.

checkEmail(email)

Opis: Sprawdza, czy istnieje użytkownik o podanym adresie e-mail.

Parametry:

- **email** (String): Adres e-mail do sprawdzenia.
- Zwraca:
- Obiekt użytkownika, jeśli istnieje.

NIPUnique(...)

Opis: Sprawdza, czy NIP jest unikalny w systemie.

Parametry:

- **arr**: Tablica błędów.
- **NIP**: Numer NIP do sprawdzenia.
- Zwraca:
- **true**: Jeśli NIP jest unikalny.

accountNumberUnique(...)

Opis: Sprawdza, czy numer konta jest unikalny w systemie.

Parametry:

- **arr**: Tablica błędów.

- **accountNumber**: Numer konta do sprawdzenia.
- Zwraca:
- **true**: Jeśli numer konta jest unikalny.

emailUnique(...)

Opis: Sprawdza, czy adres e-mail jest unikalny w systemie.

Parametry:

- **arr**: Tablica błędów.
- **email**: Adres e-mail do sprawdzenia.
- Zwraca:
- **true**: Jeśli adres e-mail jest unikalny.

displayAll()

Opis: Pobiera listę wszystkich użytkowników.

Zwraca:

- Tablica obiektów wszystkich użytkowników.

active(email)

Opis: Aktywuje adres e-mail użytkownika.

Parametry:

- **email** (String): Adres e-mail do aktywacji.
- Zwraca:
- **true**: Jeśli aktywacja się powiedzie.

Bezpieczeństwo

Wszystkie hasła są hashowane przy użyciu **bcrypt**. Nigdy nie przechowuj i nie przesyłaj haseł w postaci jawnej.

Eksportowane Funkcje

Moduł eksportuje funkcje do zarządzania użytkownikami:

```
module.exports = {
  increseCounter, resetCounter, add, auth, changePassword,
  passwordCompare, checkEmail, NIPUnique,
accountNumberUnique,
  emailUnique, displayAll, active
};
```

Dokumentacja Modułu Walidacji dla „FakturaPro"

Przegląd

Moduł walidacji oferuje zestaw funkcji do sprawdzania poprawności danych wejściowych w aplikacji "FakturaPro". Obejmuje walidacje tekstu, adresu e-mail, hasła, daty, liczby oraz specyficzne dla NIP.

Funkcje Walidacyjne:

check(arr, value)

Opis: Sprawdza, czy wartość jest pusta.
Parametry:

- **arr** (Array): Tablica, do której dodawane są komunikaty błędów.
- **value** (String): Wartość do sprawdzenia.
- Zwraca:
- **true**: Jeśli wartość jest pusta.
- **false**: W przeciwnym przypadku.

text(arr, value)

Opis: Sprawdza, czy wartość składa się tylko z liter.
Parametry:

- **arr** (Array), **value** (String).
- Zwraca:
- **true**: Jeśli wartość zawiera nieprawidłowe znaki.
- **false**: W przeciwnym przypadku.

email(arr, value)

Opis: Sprawdza poprawność składni adresu e-mail.
Parametry:

- **arr** (Array), **value** (String).
- Zwraca:
- **true**: Jeśli składnia jest nieprawidłowa.
- **false**: W przeciwnym przypadku.

password(arr, value)

Opis: Sprawdza, czy hasło spełnia określone wymagania.

Parametry:

- **arr** (Array), **value** (String).
- Zwraca:
- **true**: Jeśli hasło jest nieprawidłowe.
- **false**: W przeciwnym przypadku.

date(arr, value)

Opis: Sprawdza, czy wartość jest prawidłową datą.

Parametry:

- **arr** (Array), **value** (Date).
- Zwraca:
- **true**: Jeśli wartość nie jest datą.
- **false**: W przeciwnym przypadku.

number(arr, value)

Opis: Sprawdza, czy wartość jest liczbą.

Parametry:

- **arr** (Array), **value** (String).
- Zwraca:
- **true**: Jeśli wartość nie jest liczbą.
- **false**: W przeciwnym przypadku.

equal(arr, value, number)

Opis: Sprawdza, czy długość wartości jest równa określonej liczbie.

Parametry:

- **arr** (Array), **value** (String), **number** (Number).
- Zwraca:
- **true**: Jeśli długość jest nieprawidłowa.
- **false**: W przeciwnym przypadku.

min(arr, value, number)

Opis: Sprawdza, czy długość wartości jest co najmniej określonej liczby.

Parametry:

- **arr** (Array), **value** (String), **number** (Number).
- Zwraca:
- **true**: Jeśli długość jest za krótka.
- **false**: W przeciwnym przypadku.

max(arr, value, number)

Opis: Sprawdza, czy długość wartości nie przekracza określonej liczby.
Parametry:

- **arr** (Array), **value** (String), **number** (Number).
- Zwraca:
- **true**: Jeśli długość jest za długa.
- **false**: W przeciwnym przypadku.

compare(arr, value, value2)

Opis: Porównuje dwie wartości pod kątem ich równości.
Parametry:

- **arr** (Array), **value** i **value2** (String).
- Zwraca:
- **true**: Jeśli wartości się różnią.
- **false**: W przeciwnym przypadku.

nip(arr, nip)

Opis: Sprawdza, czy NIP istnieje i jest prawidłowy poprzez zapytanie do zewnętrznego API.
Parametry:

- **arr** (Array), **nip** (String).
- Zwraca:
- Dane odpowiedzi z API, jeśli NIP jest prawidłowy.
- **true**: Jeśli NIP jest nieprawidłowy.

Eksportowane Funkcje

Moduł eksportuje wszystkie funkcje walidacyjne:

module.exports = { check, text, email, password, date, number, equal, min, max, compare, nip };

Obsługa Błędów

Błędy są rejestrowane w tablicy **arr** jako komunikaty tekstowe. W przypadku wystąpienia błędu w funkcji **nip**, błąd jest logowany do konsoli.