

Dostęp do drzwi

Jakub Iwaszkiewicz 264506
264506@student.pwr.edu.pl

Krzysztof Marszałek 263835
263835@student.pwr.edu.pl

Streszczenie

System ma na celu udostępnienie panelu pozwalającego na nadawanie dostępu do pomieszczeń w zakładzie pracy (np. przez kontrolę kart zbliżeniowych). Z systemu mogą korzystać wyłącznie zalogowani użytkownicy. Pracownik uzyskuje informację o pomieszczeniach, do których ma uprawnienia, oraz może zgłosić zgubienie karty. Administrator systemu nadaje uprawnienia dostępowe dla użytkowników. Opcjonalnie, system powinien gromadzić informację o wykorzystaniu karty.

Spis treści

1	Opis projektu	2
1.1	Skład grupy	2
1.2	Cel projektu	2
1.3	Typy użytkowników	2
2	Wymagania funkcjonalne	3
2.1	Logowanie	3
2.2	Wylogowanie	3
2.3	Tworzenie drzwi	3
2.4	Usuwanie drzwi	4
2.5	Tworzenie konta pracowników	4
2.6	Usuwanie konta pracownika	5
2.7	Przyznawanie uprawnień pracownikom do drzwi	5
2.8	Usuwanie uprawnień pracownikom do drzwi	5
2.9	Wyświetlanie dostępów pracownika	6
2.10	Zmiana właściciela systemu	6
3	Wykorzystane technologie	6
3.1	Wykorzystane technologie programistyczne	6
4	Architektura informacji	7
4.1	Schemat bazy danych	7
4.2	Stworzenie bazy danych	8
4.3	Opis encji w bazie danych	8
4.3.1	Tabela Labourers	8
4.3.2	Tabela Door	8
4.3.3	Tabela Access	8
5	Wprowadzenie do API	8
5.1	Konfiguracja i użycie API	8
5.2	Endpointy API	8
5.2.1	Usuwanie zasobów (Delete Endpoints)	10
5.2.2	Odczyt zasobów (Read Endpoints)	10
5.2.3	Aktualizacja zasobów (Update Endpoints)	11
5.2.4	Zestawienie graficzne endpointów	13
6	Konfiguracja serwera	14
7	Graficzny projekt interfejsu użytkownika	15
7.1	Ekran logowania	15
7.2	Profil pracownika	16
7.3	Profil Administratora	17
7.3.1	Edyowanie dostępów	18
7.3.2	Edytowanie drzwi	18
7.3.3	Edytuj pracowników	19
7.4	Profil właściciela	20
7.4.1	Edycja przywilejów	20
7.4.2	Usuwanie przywilejów	21
8	Podsumowanie	22
8.1	Wnioski	22
8.2	Możliwości rozwoju	22

1 Opis projektu

1.1 Skład grupy

1. Jakub Iwaszkiewicz - React.js, Node.js, MySQL, Express.js
2. Krzysztof Marszałek - React.js, styles, MySQL

1.2 Cel projektu

Celem projektu jest stworzenie nowoczesnej aplikacji internetowej, która poprawi efektywność i wydajność zarządzania dostępem do drzwi. Aplikacja "dostęp do drzwi" jest dla zakładu pracy i umożliwia dostęp do drzwi dla pracowników, którzy obecnie są w nim zatrudnieni. Aplikacja jest przeznaczona dla intranetu danego zakładu pracy. Użytkownicy niezalogowani mogą zalogować się jako pracownicy lub jak administrator, w zależności od tego czy dany użytkownik odpowiada za dostęp do drzwi. Administrator ma dostęp do wszystkich informacji jak i usług w firmie, czyli: może przyznawać dostęp do drzwi dla pracowników, może także usuwać te dostęp. Dodatkowo w każdej chwili może zmienić uprawnienia dostępu do drzwi w zależności od różnych sytuacji. Administrator dostaje prośbę o dostęp do drzwi w formie powiadomienia w momencie kiedy pracownik ją wyśle. Może zaakceptować albo anulować request. Co również odróżnia administratora to fakt, że ma on wgląd w całą historię przyznawania dostępu do drzwi, przez siebie jak i przez innych administratorów. Pracownik może odczytać jakie ma przyznane dostęp do drzwi oraz wysłać prośbę o dostęp do nowych i tę prośbę również usunąć. Projekt ma na celu nie tylko stworzenie funkcjonalnej aplikacji, ale także zdobycie doświadczenia w pracy zespołowej i rozwijanie umiejętności w zakresie nowoczesnych technologii webowych.

1.3 Typy użytkowników

1. **Właściciel**
Właściciel jest użytkownikiem, który ma dostęp do wszystkich funkcjonalności systemu. Może on zarządzać użytkownikami, drzwiami, dostępami do drzwi oraz administratorami.
2. **Administrator**
Administrator jest użytkownikiem, który ma dostęp do wszystkich funkcjonalności systemu. Może on zarządzać użytkownikami, drzwiami oraz dostępami do drzwi.
3. **Pracownik**
Pracownik jest użytkownikiem, który ma dostęp do uprawnień przyznanych przez administratora.
4. **Niezalogowany użytkownik**
Niezalogowany użytkownik nie ma dostępu do żadnych funkcjonalności systemu. Może on jedynie zalogować się do systemu.

2 Wymagania funkcjonalne

2.1 Logowanie

1. **Opis:** Użytkownik może zalogować się do systemu.

2. **Aktorzy:** Niezalogowany użytkownik.

3. **Scenariusz główny:**

- (a) Użytkownik wypełnia formularz logowania.
- (b) Użytkownik klika przycisk "Zaloguj".
- (c) System sprawdza poprawność danych.
- (d) System loguje użytkownika.

4. **Scenariusz alternatywny:**

- (a) Użytkownik wypełnia formularz logowania.
- (b) Użytkownik klika przycisk "Zaloguj".
- (c) System sprawdza poprawność danych.
- (d) System wyświetla komunikat o błędzie.

2.2 Wylogowanie

1. **Opis:** Użytkownik może wylogować się z systemu.

2. **Aktorzy:** Zalogowany użytkownik.

3. **Scenariusz główny:**

- (a) Użytkownik klika przycisk "Wyloguj" w panelu sterowania.
- (b) System wylogowuje użytkownika i przekierowuje na stronę logowania.

2.3 Tworzenie drzwi

1. **Opis:** Administrator może dodać drzwi do systemu.

2. **Aktorzy:** Administrator.

3. **Scenariusz główny:**

- (a) Administrator klika przycisk "Edytuj drzwi" w panelu sterowania.
- (b) System wyświetla formularz dodawania drzwi.
- (c) Administrator wypełnia formularz dodawania drzwi.
- (d) Administrator klika przycisk "Dodaj".
- (e) System sprawdza poprawność danych.
- (f) System dodaje drzwi do systemu.

4. **Scenariusz alternatywny:**

- (a) Administrator klika przycisk "Edytuj drzwi" w panelu sterowania.
- (b) System wyświetla formularz dodawania drzwi.
- (c) Administrator wypełnia formularz dodawania drzwi.
- (d) Administrator klika przycisk "Dodaj".
- (e) System sprawdza poprawność danych.
- (f) System wyświetla komunikat o błędzie.

2.4 Usuwanie drzwi

1. **Opis:** Administrator może usunąć drzwi z systemu.
2. **Aktorzy:** Administrator.
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Edytuj drzwi" w panelu sterowania.
 - (b) System wyświetla formularz usuwania drzwi.
 - (c) Administrator wypełnia formularz usuwania drzwi.
 - (d) Administrator klika przycisk "Usuń".
 - (e) System sprawdza poprawność danych.
 - (f) System usuwa drzwi z systemu.
4. **Scenariusz alternatywny:**
 - (a) Administrator klika przycisk "Edytuj drzwi" w panelu sterowania.
 - (b) System wyświetla listę istniejących drzwi
 - (c) Administrator klika przycisk "Usuń".
 - (d) System sprawdza poprawność danych.
 - (e) System wyświetla komunikat o błędzie.

2.5 Tworzenie konta pracowników

1. **Opis:** Administrator może dodać konto pracownika do systemu.
2. **Aktorzy:** Administrator.
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Edytuj pracowników" w panelu sterowania.
 - (b) System wyświetla formularz dodawania pracownika.
 - (c) Administrator wypełnia formularz dodawania pracownika.
 - (d) Administrator klika przycisk "Dodaj".
 - (e) System sprawdza poprawność danych.
 - (f) System dodaje konto pracownika do systemu.
4. **Scenariusz alternatywny:**
 - (a) Administrator klika przycisk "Edytuj pracowników" w panelu sterowania.
 - (b) System wyświetla formularz dodawania pracownika.
 - (c) Administrator wypełnia formularz dodawania pracownika.
 - (d) Administrator klika przycisk "Dodaj".
 - (e) System sprawdza poprawność danych.
 - (f) System wyświetla komunikat o błędzie.

2.6 Usuwanie konta pracownika

1. **Opis:** Administrator może usunąć konto pracownika z systemu.
2. **Aktorzy:** Administrator.
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Edytuj pracowników" w panelu sterowania.
 - (b) System wyświetla listę istniejących pracowników.
 - (c) Administrator klika przycisk "Usuń".
 - (d) System sprawdza poprawność danych.
 - (e) System usuwa konto pracownika z systemu.

2.7 Przyznawanie uprawnień pracownikom do drzwi

1. **Opis:** Administrator może przyznać uprawnienia pracownikowi do drzwi.
2. **Aktorzy:** Administrator.
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Edytuj dostęp" w panelu sterowania.
 - (b) System wyświetla dwie listy rozwijane: lista pracowników i lista drzwi.
 - (c) Administrator wybiera pracownika i drzwi, które chce mu przydzielić.
 - (d) Administrator klika przycisk "Przydziel dostęp".
 - (e) System sprawdza poprawność danych.
 - (f) System zapisuje zmiany.
4. **Scenariusz alternatywny:**
 5. (a) Administrator klika przycisk "Edytuj dostęp" w panelu sterowania.
 - (b) System wyświetla dwie listy rozwijane: lista pracowników i lista drzwi.
 - (c) Administrator wybiera pracownika i drzwi, które chce mu przydzielić.
 - (d) Administrator klika przycisk "Przydziel dostęp".
 - (e) System sprawdza poprawność danych.
 - (f) System wyświetla komunikat o błędzie.

2.8 Usuwanie uprawnień pracownikom do drzwi

1. **Opis:** Administrator może usunąć uprawnienia pracownikowi do drzwi.
2. **Aktorzy:** Administrator.
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Edytuj dostęp" w panelu sterowania.
 - (b) System wyświetla listę istniejących uprawnień.
 - (c) Administrator wybiera pracownika i drzwi, których uprawnienia chce mu odebrać.
 - (d) Administrator klika przycisk "Usuń".
 - (e) System sprawdza poprawność danych.
 - (f) System zapisuje zmiany.

2.9 Wyświetlanie dostępów pracownika

1. **Opis:** Pracownik może wyświetlić listę drzwi, do których ma dostęp pracownik.
2. **Aktorzy:** Pracownik
3. **Scenariusz główny:**
 - (a) Administrator klika przycisk "Dostęp" w panelu sterowania.
 - (b) System wyświetla listę drzwi, do których ma dostęp pracownik.

todo: dodac funkcjonalnosc gubienia karty

2.10 Zmiana właściciela systemu

1. **Opis:** Właściciel zmienia właściciela systemu.
2. **Aktorzy:** Właściciel
3. **Scenariusz główny:**
 - (a) Właściciel klika przycisk "Dodaj przywileje" w panelu sterowania.
 - (b) System wyświetla listę administratorów
 - (c) Administrator klika przycisk "Ustaw jako ownera".
 - (d) System odbiera uprawnienia właściciela od aktualnego właściciela.
 - (e) System nadaje uprawnienia właściciela wybranemu administratorowi.
 - (f) System przeładuje stronę i były właściciel jest teraz zwykłym administratorem i nie możliwości zmiany właściciela.

3 Wykorzystane technologie

Wykorzystana technologia: System kontroli wersji - Git

System kontroli wersji, który został wybrany do projektu to git. W znaczym stopniu ułatwił on pracę nad projektem. Pozwolił na łatwe śledzenie zmian w kodzie, a także na łatwe przywracanie poprzednich wersji. Dzięki niemu możliwa była również praca nad projektem przez dwie osoby jednocześnie.

Stworzyliśmy repozytorium na platformie github.com.

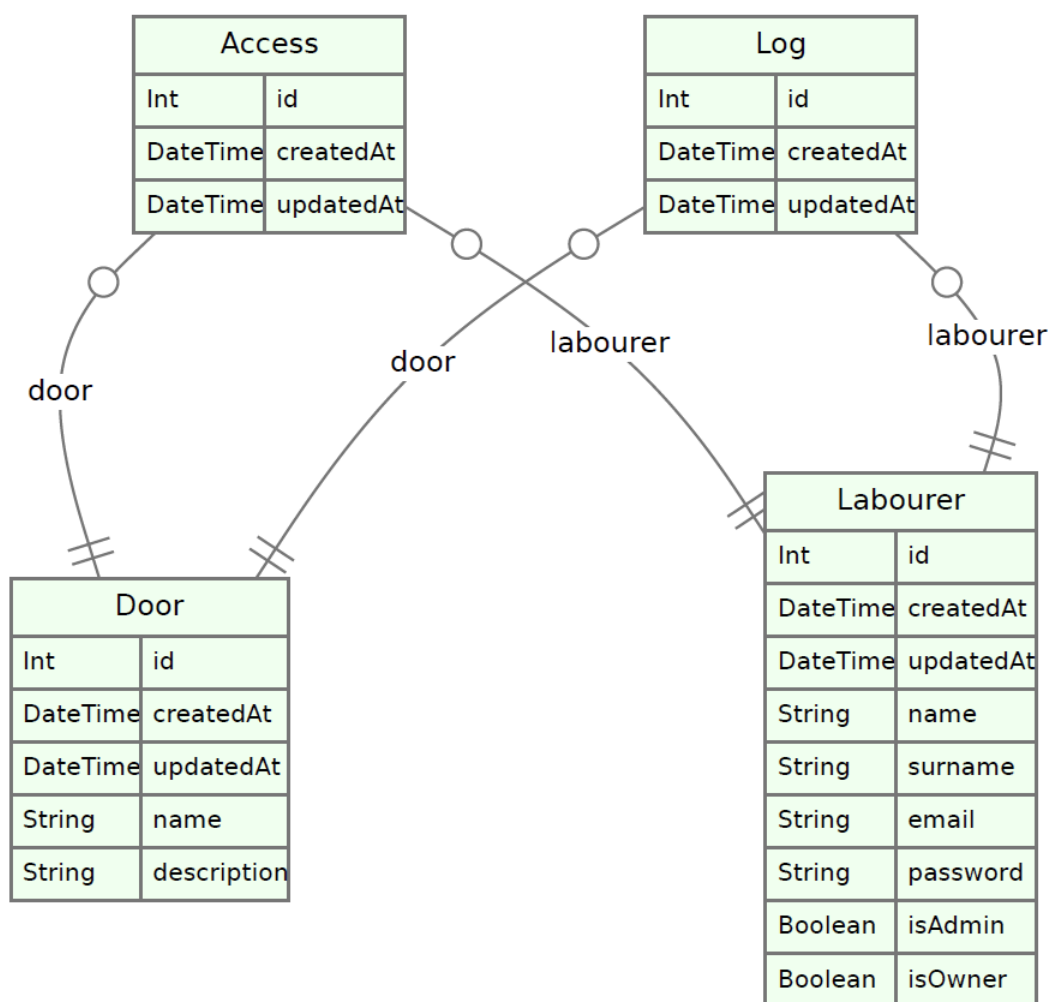
3.1 Wykorzystane technologie programistyczne

- **React.js:** Biblioteka React.js, która została wykorzystana do stworzenia i ustrukturyzowania interfejsu użytkownika przy pomocy JSX'a.
- **Express.js:** Biblioteka express.js, która została wykorzystana do stworzenia serwera, API i endpointów.
- **SASS:** Biblioteka SASS, która została wykorzystana do stylizacji interfejsu użytkownika.
- **Prisma:** ORM, który został wykorzystany do połączenia się, wykonywania zmian jak i komunikacji z bazą danych.
- **AWS:** Chmura AWS, która została wykorzystana do hostowania bazy danych.
- **REST:** Architektura REST, która została wykorzystana do komunikacji między serwerem a klientem.

- **Figma:** Program Figma, który został wykorzystany do stworzenia projektu interfejsu użytkownika.
- **Visual Studio Code:** Środowisko programistyczne Visual Studio Code, które zostało wykorzystane do pisania kodu serwera, kodu aplikacji jak i kodu dokumentacji.
- **LaTeX:** Środowisko programistyczne LaTeX, które zostało wykorzystane do pisania dokumentacji.

4 Architektura informacji

4.1 Schemat bazy danych



Rysunek 1: Schemat bazy danych

4.2 Stworzenie bazy danych

4.3 Opis encji w bazie danych

4.3.1 Tabela Labourers

Tabela zawierająca wszystkie informacje dotyczących użytkowników aplikacji - są w niej zawarte informacje o użytkownikach, którzy są pracownikami, administratorami i właścicielami.

id	createdAt	updatedAt	name	surname	email	password	isAdmin	isOwner
int AI PK	datetime(3)	datetime(3)	varchar(191)	varchar(191)	varchar(191)	varchar(191)	tinyint(1)	tinyint(1)

Tabela 1: Tabela Labourers

4.3.2 Tabela Door

Tabela zawierająca wszystkie informacje dotyczące drzwi - są w niej zawarte informacje: kiedy zostały stworzone, kiedy zostały ostatnio do kogoś przypisane, nazwa oraz opis drzwi.

id	createdAt	updatedAt	name	description
int AI PK	datetime(3)	datetime(3)	varchar(191)	varchar(191)

Tabela 2: Tabela Door

4.3.3 Tabela Access

Tabela zawierająca wszystkie informacje dotyczące uprawnień do drzwi - są zawarte w niej informacje: kiedy zostały stworzone, kiedy zostały ostatnio zmodyfikowane, jakie drzwi są do jakiego użytkownika przypisane.

id	createdAt	updatedAt	doorID	labourerID
int AI PK	datetime(3)	datetime(3)	int	int

Tabela 3: Tabela Access

5 Wprowadzenie do API

API naszej aplikacji webowej są kluczowymi elementami które umożliwiają interakcje pomiędzy front-endem a bazą danych. Dzięki nim użytkownicy mogą wykonywać operacje na danych takie jak: tworzenie, odczytywanie, aktualizowanie i usuwanie danych w responsywny sposób.

5.1 Konfiguracja i użycie API

Nasze API zostało zaprojektowane z myślą o łatwości integracji i elastyczności. W tej sekcji przedstawimy kluczowe aspekty konfiguracji i podstawowe kroki potrzebne do rozpoczęcia pracy z naszym API, w tym uwierzytelnianie i podstawowe zapytania.

5.2 Endpointy API

- Tworzenie zasobów (Create Endpoints)
- Usuwanie zasobów (Delete Endpoints)
- Odczyt zasobów (Read Endpoints)
- Aktualizacja zasobów (Update Endpoints)

Dodawanie Pracownika (Labourer)

- **Metoda:** POST
- **Ścieżka:** '/labourers'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na dodanie tylko administratorowi nowego pracownika do systemu. Wymaga podania danych takich jak:
 - **name** - imię pracownika
 - **surname** - nazwisko pracownika
 - **email** - adres email pracownika
 - **password** - hasło pracownika
- **Walidacja:** Sprawdza, czy email i hasło spełniają określone kryteria (np. długość hasła, format emaila)
- **Odpowiedź:** Status 200 - OK przy udanym dodaniu, 403 Forbidden przy błędach walidacji lub braku uprawnień

Dodawanie Drzwi (Door)

- **Metoda:** POST
- **Ścieżka:** '/doors'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na dodanie tylko administratorowi nowych drzwi do systemu. Wymaga podania danych takich jak:
 - **name** - nazwa drzwi
 - **description** - opis drzwi
- **Walidacja:** Sprawdza, czy nazwa drzwi jest unikalna
- **Odpowiedź:** Status 200 - OK przy udanym dodaniu, 403 Forbidden przy błędach walidacji lub braku uprawnień

Dodawanie Uprawnień (Access)

- **Metoda:** POST
- **Ścieżka:** '/accesses'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na dodanie tylko administratorowi nowych uprawnień do systemu. Sprawdza, czy dany pracownik i drzwi istnieją w systemie.
- **Odpowiedź:** Status 200 - OK przy udanym dodaniu, 403 Forbidden przy błędach walidacji lub braku uprawnień

5.2.1 Usuwanie zasobów (Delete Endpoints)

Usuwanie Pracownika (Labourer)

- **Metoda:** DELETE
- **Ścieżka:** '/labourers'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na usunięcie tylko administratorowi pracownika z systemu. Wymaga podania email pracownika.
- **Odpowiedź:** Status 200 - OK przy udanym usunięciu, 403 Forbidden przy błędach walidacji lub braku uprawnień.

Usuwanie Drzwi (Door)

- **Metoda:** DELETE
- **Ścieżka:** '/doors'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na usunięcie tylko administratorowi drzwi z systemu. Wymaga podania nazwy drzwi.
- **Odpowiedź:** Status 200 - OK przy udanym usunięciu, 403 Forbidden przy błędach walidacji lub braku uprawnień.

Usuwanie Uprawnień (Access)

- **Metoda:** DELETE
- **Ścieżka:** '/access'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na usunięcie tylko administratorowi uprawnień z systemu. Wymaga podania id pracownika i id drzwi.
- **Odpowiedź:** Status 200 - OK przy udanym usunięciu, 403 Forbidden przy błędach walidacji lub braku uprawnień.

5.2.2 Odczyt zasobów (Read Endpoints)

Odczyt wszystkich Pracowników (Labourers)

- **Metoda:** GET
- **Ścieżka:** '/labourers'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na odczyt tylko administratorowi pracowników z systemu.
- **Odpowiedź:** Zwraca listę pracowników w formacie JSON.

Odczyt danych o Pracowniku (Labourer)

- **Metoda:** GET
- **Ścieżka:** '/labourer'

- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Umożliwia pracownikowi odczytanie własnych danych z systemu.
- **Odpowiedź:** Zwraca dane zalogowanego pracownika w formacie JSON.

Odczyt wszystkich Drzwi (Door)

- **Metoda:** GET
- **Ścieżka:** '/doors'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na odczyt tylko administratorowi wszystkich drzwi z systemu.
- **Odpowiedź:** Zwraca listę drzwi w formacie JSON.

Odczyt dostępów (Access)

- **Metoda:** GET
- **Ścieżka:** '/access'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na odczytanie pracownikowi odczytanie informacji o swoich uprawnieniach do drzwi. Z
- **Odpowiedź:** Zwraca listę uprawnień w formacie JSON, każdy z dostępów zawiera nazwę drzwi.

Odczyt administratorów (Administrators)

- **Metoda:** GET
- **Ścieżka:** '/administrators'
- **Autentykacja:** Wymagany token uwierzytelniający
- **Opis:** Pozwala na odczyt tylko właścicielowi wszystkich administratorów z systemu.
- **Odpowiedź:** Zwraca listę administratorów w formacie JSON.

5.2.3 Aktualizacja zasobów (Update Endpoints)

Dodanie uprawnień Administratora

- **Metoda:** PUT
- **Ścieżka:** '/privileges/add'
- **Autentykacja:** Wymagany token uwierzytelniający z uprawnieniami właścicielami.
- **Opis:** Pozwala właścicielowi aplikacji na przyznanie uprawnień administratora istniejącemu pracownikowi. Wymaga podania adresu email pracownika, który ma otrzymać uprawnienia.
- **Walidacja:** Sprawdzane jest, czy zalogowany użytkownik jest właścicielem, czy podany email istnieje w systemie, i czy pracownik nie jest już administratorem.
- **Odpowiedź:** Status 200 OK po udanym przyznaniu uprawnień, 403 Forbidden, jeśli użytkownik nie jest właścicielem, 404 Not Found, jeśli pracownik nie istnieje, 400 Bad Request, jeśli pracownik już jest administratorem.

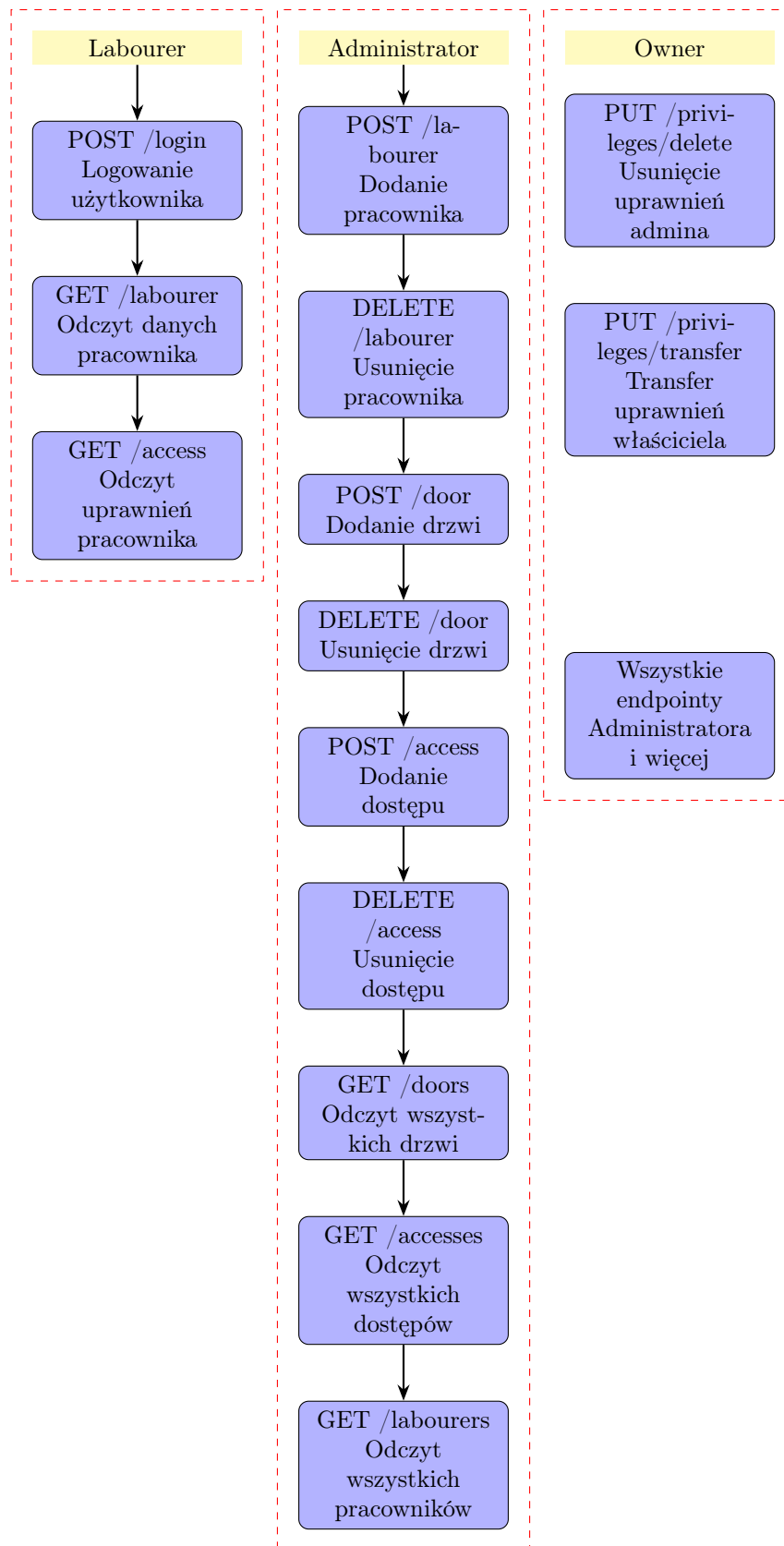
Usunięcie uprawnień Administratora

- **Metoda:** PUT
- **Ścieżka:** '/privileges/delete'
- **Autentykacja:** Wymagany token uwierzytelniający z uprawnieniami właścicielami.
- **Opis:** Pozwala właścicielowi aplikacji na odebranie uprawnień administratora istniejącemu pracownikowi. Wymaga podania adresu email pracownika, który ma stracić uprawnienia.
- **Walidacja:** Sprawdzane jest, czy zalogowany użytkownik jest właścicielem, czy podany email istnieje w systemie, i czy pracownik jest administratorem.
- **Odpowiedź:** Status 200 OK po udanym odebraniu uprawnień, 403 Forbidden, jeśli użytkownik nie jest właścicielem, 404 Not Found, jeśli pracownik nie istnieje, 400 Bad Request, jeśli pracownik nie jest administratorem.

Transfer uprawnień Właściciela

- **Metoda:** PUT
- **Ścieżka:** '/privileges/transfer'
- **Autentykacja:** Wymagany token uwierzytelniający z uprawnieniami właścicielami.
- **Opis:** Umożliwia właścicielowi na przekazanie swoich uprawnień innemu administratorowi. Wymaga podania emaila pracownika, który ma otrzymać uprawnienia właściciela.
- **Walidacja:** Sprawdzane jest, czy zalogowany użytkownik jest właścicielem, czy pracownik istnieje w systemie, i czy jest administratorem.
- **Odpowiedź:** Status 200 OK po udanym transferze uprawnień, 403 Forbidden, jeśli użytkownik nie jest właścicielem, 404 Not Found, jeśli pracownik nie istnieje, 400 Bad Request, jeśli pracownik nie jest administratorem.

5.2.4 Zestawienie graficzne endpointów



6 Konfiguracja serwera

1. Środowisko i konfiguracja:

- **dotenv** - Wykorzystanie pakietu `dotenv` do zarządzania zmiennymi środowiskowymi, co ułatwia konfigurację aplikacji bez bezpośredniego umieszczania wrażliwych danych w kodzie.
- **Port** - Serwer jest skonfigurowany do nasłuchiwania na porcie określonym w zmiennej środowiskowej `PORT`, z domyślnym fallbackiem na port `3002`.

2. Middleware

- **CORS**: Konfiguracja Cross-Origin Resource Sharing (CORS) ogranicza żądania do zaufanych domen (tutaj `http://localhost:5173`), zezwalając na metody HTTP takie jak `GET`, `POST`, `DELETE` itp. Jest to istotne z punktu widzenia
- **JSON Parser**: Użycie `express.json()` jako middleware do parsowania przychodzących żądań z treścią w formacie JSON.

3. Routing

- **Read Routes**: Import i użycie route'ów do odczytu danych (`readRoutes`) pod ścieżką `/api/read`.
- **Create Routes**: Import i użycie route'ów do tworzenia danych (`createRoutes`) pod ścieżką `/api/create`.
- **Delete Routes**: Import i użycie route'ów do usuwania danych (`deleteRoutes`) pod ścieżką `/api/delete`.
- **Update Routes**: Import i użycie route'ów do aktualizacji danych (`updateRoutes`) pod ścieżką `/api/update`.

4. Generowanie Tokenów JWT

- Tokeny te są wykorzystywane do uwierzytelniania i zarządzania sesjami użytkowników.

5. Konfiguracja ORM Prisma

- Prisma jest ORM (Object Relational Mapping) do Node.js i TypeScript, który umożliwia bezpośredni dostęp do bazy danych z poziomu kodu źródłowego aplikacji.
- Prisma Client jest generowany na podstawie schematu bazy danych, który jest zdefiniowany w pliku `schema.prisma`.
- Użyto MySQL jako bazy danych.
- Prisma jest skonfigurowana do połączenia z bazą danych za pomocą zmiennej środowiskowej `DATABASE_URL`.

7 Graficzny projekt interfejsu użytkownika

7.1 Ekran logowania

Pierwszym widokiem, który widzi użytkownik po wejściu na stronę jest ekran logowania. Na środku ekranu znajduje się nazwa aplikacji, a pod nią formularz logowania. W celu potwierdzenia logowania należy kliknąć przycisk "Zaloguj się".



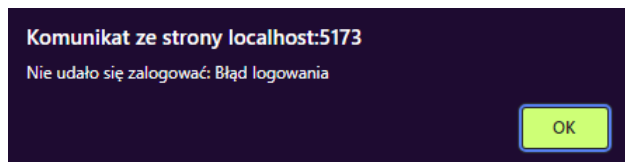
Rysunek 2: Ekran logowania

W przypadku niepoprawnego wprowadzenia danych, użytkownik zostanie poinformowany o błędnym formacie emaila:



Rysunek 3: Ekran logowania - błędny email

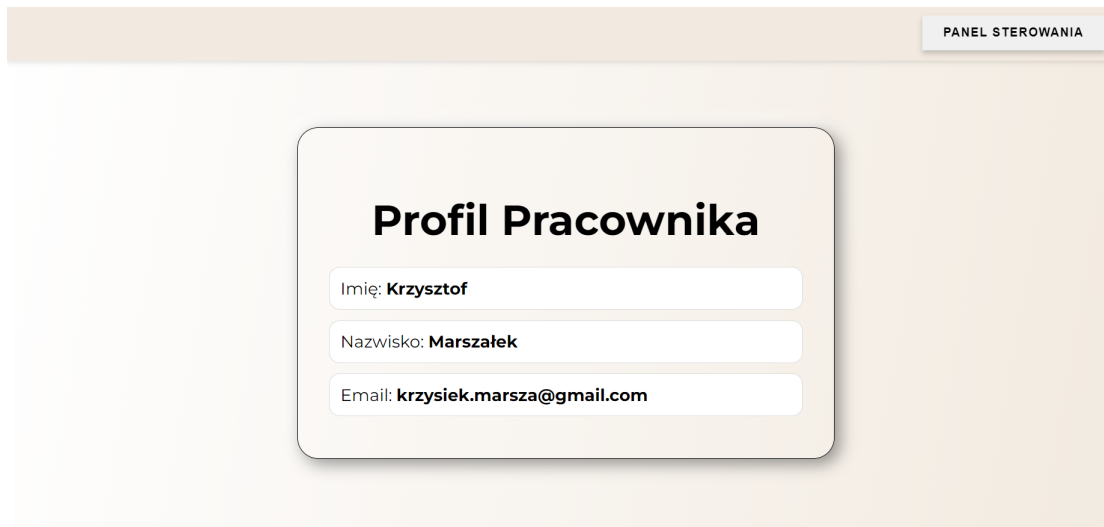
Analogicznie po wprowadzeniu błędnego hasła wyświetli się następujący komunikat:



Rysunek 4: Ekran logowania - błędne hasło

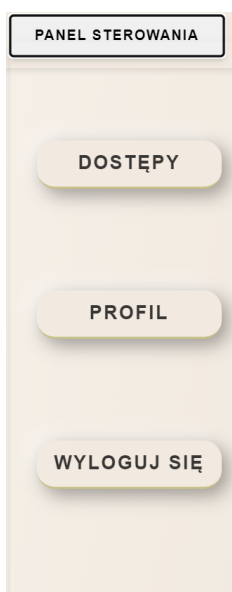
7.2 Profil pracownika

Pracownik po zalogowaniu się do systemu zostanie przekierowany na stronę swojego profilu, gdzie zostają mu wyświetlone informacje o nim:



Rysunek 5: Profil pracownika

W prawym górnym rogu znajduje się tzw. hamburger menu pod nazwą "Panel sterowania", które po kliknięciu rozwinie się i pokaże opcje, które są dostępne dla pracownika:



Rysunek 6: Panel sterowania pracownika

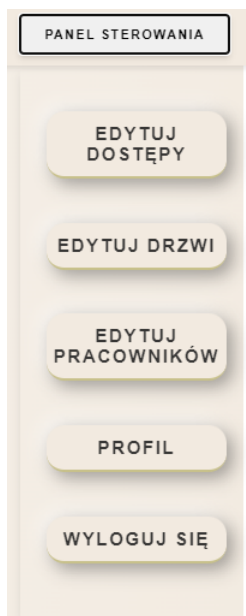
Po kliknięciu w opcję "Dostępy" zostanie wyświetlona lista dostępów do drzwi, które zostały przydzielone pracownikowi:



Rysunek 7: Lista dostępów do drzwi pracownika

7.3 Profil Administratora

Administrator analogicznie jak pracownik po zalogowaniu zostanie przekierowany na stronę swojego profilu, natomiast panel sterowania będzie wyglądał następująco:



Rysunek 8: Panel sterowania administratora

7.3.1 Edytowanie dostępów

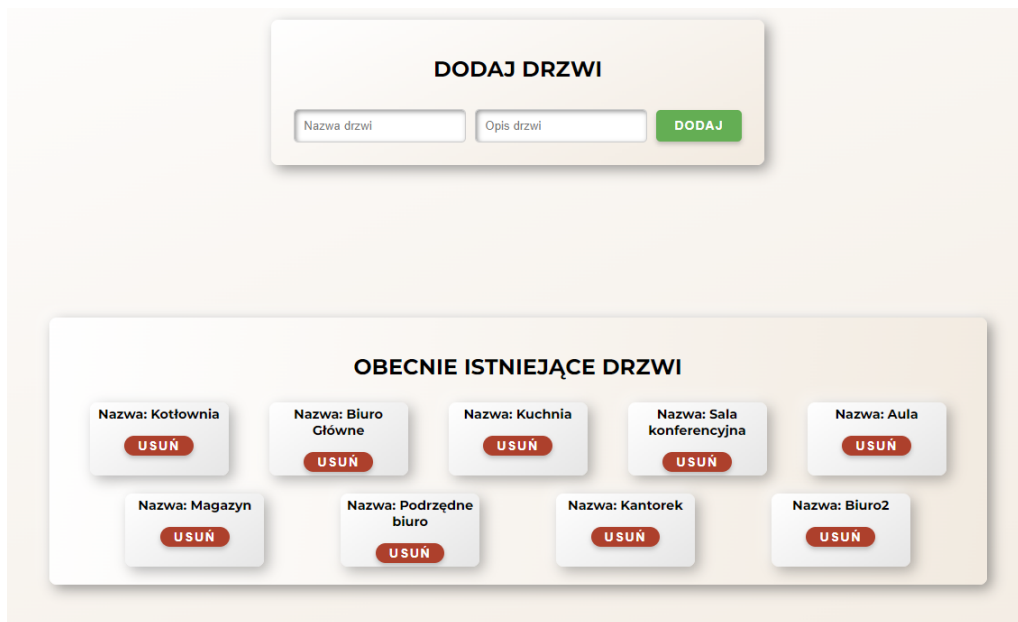
Administrator po kliknięciu w opcję "Edytuj dostęp" zostanie przekierowany na stronę, gdzie będzie mógł edytować dostęp do drzwi. Z listy wybieranej może wybrać pracownika, któremu chce zmienić dostęp do drzwi, a następnie z listy rozwijanej wybrać drzwi, które chce przydzielić pracownikowi. Po kliknięciu w przycisk "Przydziel dostęp" zostanie zapisana zmiana w bazie danych. Poniżej znajduje się lista wszystkich dostępów do drzwi, które zostały przydzielone pracownikom. Administrator może usunąć dostęp klikając w przycisk "Usuń".



Rysunek 9: Edytowanie dostępów

7.3.2 Edytowanie drzwi

Administrator po kliknięciu w opcję "Edytuj drzwi" zostanie przekierowany na stronę, gdzie będzie mógł edytować drzwi. Funkcjonalność dodawania drzwi wygląda w taki sposób, że administratorom musi podać nazwę drzwi oraz jego opis. Po kliknięciu w przycisk "Dodaj" zostanie zapisana zmiana w bazie danych. Poniżej znajduje się lista wszystkich drzwi z nazwami, które zostały dodane do systemu. Administrator może usunąć drzwi klikając w przycisk "Usuń".



Rysunek 10: Edytowanie drzwi

7.3.3 Edytuj pracowników

Administrator po kliknięciu w opcję "Edytuj pracowników" zostanie przekierowany na stronę, gdzie będzie mógł edytować pracowników. Funkcjonalność dodawania pracownika wygląda w taki sposób, że administrator musi podać imię, nazwisko, email oraz hasło pracownika. W każdej chwili może odświeżyć hasło klikając przycisk "Pokaż hasło". Po kliknięciu w przycisk "Dodaj pracownika" zostanie zapisana zmiana w bazie danych. Poniżej znajduje się lista wszystkich pracowników, którzy zostali dodani do systemu. Administrator może usunąć pracownika klikając w przycisk "Usuń". W przypadku kliknięcia usuń, wyskoczy potwierdzenie usunięcia pracownika.

EDYTUJ PRACOWNIKÓW

Na tej podstronie z uprawnieniami Administratora lub Właściciela, możesz usuwać lub dodawać istniejące konta Pracowników.

DODAJ PRACOWNIKA

E-mail: Imię:

Nazwisko: Hasło:

POKAŻ HASŁO **DODAJ PRACOWNIKA**

USUŃ PRACOWNIKA

id: 1 E-mail: kyle.smith@gmail.com Imię: Kyle Nazwisko: Smith USUŃ	id: 2 E-mail: jim.kowalski@gmail.com Imię: Jim Nazwisko: Kowalski USUŃ	id: 39 E-mail: krzysiek.marsza@gmail.com Imię: Krzysztof Nazwisko: Marszałek USUŃ	id: 40 E-mail: jakub.iwaszkiewicz@gmail.com Imię: Jakub Nazwisko: Iwaszkiewicz USUŃ
id: 41 E-mail: andrzej.matczak@gmail.com Imię: Andrzej Nazwisko: Matczak USUŃ			

Rysunek 11: Edytowanie pracowników

7.4 Profil właściciela

Właściciel analogicznie jak pracownik i administrator po zalogowaniu zostanie przekierowany na stronę swojego profilu. Właściciel ma dostęp do wszystkich opcji, które są dostępne dla administratora, oraz do dodatkowych opcji, takich jak "Dodaj przywileje" oraz "Usuń przywileje".



Rysunek 12: Panel sterowania właściciela

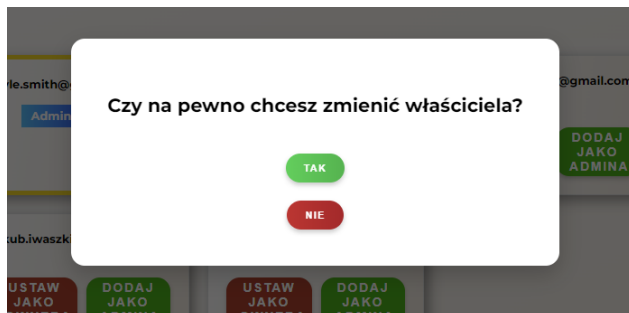
7.4.1 Edycja przywilejów

Właściciel po kliknięciu w opcję "Edytuj przywileje" zostanie przekierowany na stronę, na której może przyznać przywileje administratora wybranemu pracownikowi. Na tej podstronie zostaną wyświetleni wszyscy pracownicy. Właściciel może wybrać pracownika, któremu chce przyznać przywileje administratora. Po kliknięciu w przycisk "Dodaj jako Admina" zostanie zapisana zmiana w bazie danych. W momencie kiedy pracownik zostanie dodany jako administrator, mogą mu zostać przyznane przywileje właściciela.



Rysunek 13: Edytowanie przywilejów

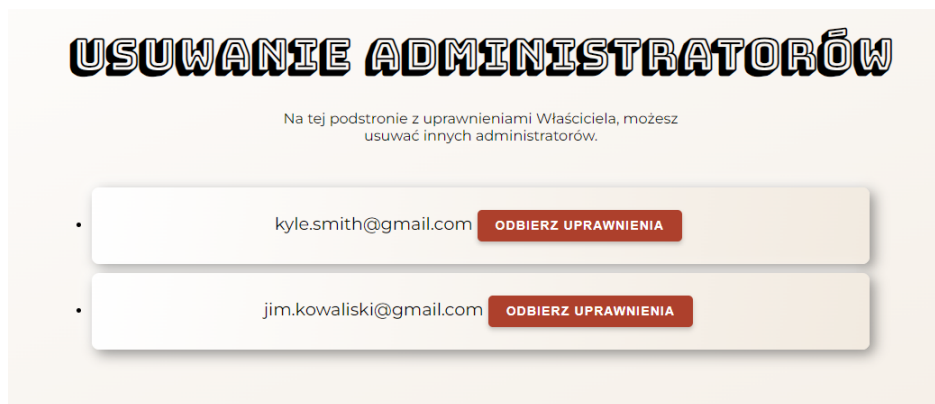
W momencie kiedy właściciel przyzna przywileje administratora wybranemu pracownikowi, zostanie wyświetlony popout, który pyta właściciela czy chce przyznać przywileje właściciela wybranemu pracownikowi. Po kliknięciu w przycisk "Tak" zostanie zapisana zmiana w bazie danych.



Rysunek 14: Potwierdzenie zmiany właściciela

7.4.2 Usuwanie przywilejów

Właściciel po kliknięciu w opcję "Usuń przywileje" zostanie przekierowany na stronę, na której może usunąć przywileje administratora wybranemu pracownikowi. Na tej podstronie zostaną wyświetleni wszyscy pracownicy z uprawnieniami administratora. Właściciel może wybrać pracownika, któremu chce usunąć przywileje klikając przycisk "Odbierz uprawnienia". Po kliknięciu w przycisk "Usuń jako Admina" zostanie zapisana zmiana w bazie danych:



Rysunek 15: Usuwanie przywilejów

8 Podsumowanie

8.1 Wnioski

Tworzenie aplikacji internetowej czy aplikacji mobilnej to bardzo złożona procedura wymagająca od programisty bardzo złożonej wiedzy na temat narzędzi jak i samego wykorzystania narzędzia. Naszym ułatwieniem było to że jedna osoba z grupy już wcześniej przez jakiś czas miała do czynienia z narzędziem React.js co pozwoliło nam na w miarę swobodne nauczanie się Express.js'a który pozwolił nam na ułożenie sobie wszystkiego w głowie a potem przelanie wszystkiego na kod. Pierwszą rzeczą jaką musieliśmy zrobić to wygląd graficzny naszej aplikacji która została wykonana w Figmie - samo znalezienie odpowiedniego narzędzia do tego i nauczanie się go było trudnym zadaniem, a co dopiero znalezienie palety kolorów, czcionek czy stylu naszej aplikacji - która i tak w trakcie programowania została nieco zaniechana przez brak czasu. Kolejnym wyzwaniem było rozpisanie funkcjonalności jak i zastosowanie się do nich i wykorzystanie ich w Express.js'ie. Następnym wyzwaniem było połączenie się z bazą danych za pomocą Prisma, co znacznie ułatwiło nam pracę z backendem. W naszym projekcie React.js odegrał fundamentalną rolę, stanowiąc kręgosłup interfejsu użytkownika. Wykorzystanie tej biblioteki umożliwiło nam stworzenie dynamicznych, interaktywnych i responsywnych widoków, które dopasowują się do różnorodnych potrzeb użytkowników. Dzięki modularnej naturze React, nasz zespół był w stanie efektywnie zarządzać kodem, tworząc reużywalne komponenty, które znacząco przyspieszyły rozwój aplikacji.

Kluczowym elementem naszej pracy z React.js było zastosowanie zaawansowanych funkcji zarządzania stanem i cyklu życia komponentów. Hooki takie jak `useState` i `useEffect` odegrały zasadniczą rolę w dynamicznym zarządzaniu danymi aplikacji, umożliwiając natychmiastową reakcję interfejsu na zmiany.

Następnym krokiem było stworzenie podstron dla każdego rodzaju użytkownika oraz zaimplementowanie React Router Dom, który pozwolił nam na przekierowanie użytkownika na odpowiednią podstronę w zależności od jego uprawnień. Kolejnym krokiem było stworzenie interfejsu graficznego dla każdej podstrony. Wykorzystaliśmy do tego bibliotekę SASS, która pozwoliła nam na zastosowanie stylu w naszej aplikacji.

8.2 Możliwości rozwoju

Dzięki zastosowaniu `express.js`'a jako tak na prawdę osobnej aplikacji, skalowalność aplikacji jest bardzo prosta, od strony serwerowej na pewno zabrakło możliwości napisania o odzyskanie zgubionej karty, jak i możliwości otworzenia drzwi w sposób zdalny (na stronie a nie przy pomocy karty) przez pracownika. Kolejną rzeczą jaką bym dołożył, to zwiększona ilość przechowywanych informacji przez naszą bazę danych pod względem wykonywanych działań przez pracowników (zapisywanie w bazie danych informacji o tym że ktoś przeszedł o tej godzinie przez te drzwi) jak i informacji na temat wykonywanych działań przez administratorów (zapisywanie informacji o tym że ktoś dodał pracownika lub właściciel został zmieniony również widoczny dla wszystkich administratorów).

Jeżeli chodzi o interfejs graficzny to przez brak czasu niektóre elementy nie zostały w 100% wystylizowane i niektóre mogą wyglądać nieco inaczej niż wszystkie - właśnie przez nie wykorzystanie pełnego potencjału komponentów `reactowych` i oddelegowywanie ich i wykorzystywanie we wszystkich podstronach, gdzie czasem mamy komponent w tym samym pliku co element wykorzystujący komponent, zamiast ułożenia w innym pliku, gdzie jest to standardowa procedura dla `Reacta`.

Sama aplikacja może zostać rozwinięta przez jeszcze bardziej zaawansowane elementy, takie jak odzyskiwanie hasła, dwuetapowa weryfikacja, wysyłanie wiadomości do współpracowników, lecz zabrakło nam niestety czasu. Jeżeli pozwoli nam motywacja to prawdopodobnie będziemy chcieli ją dostosować do standardów rekrutacyjnych i będzie naszą pomocą w znalezieniu praktyk na wakacje.

Spis rysunków

1	Schemat bazy danych	7
2	Ekran logowania	15
3	Ekran logowania - błędny email	15
4	Ekran logowania - błędne hasło	15
5	Profil pracownika	16
6	Panel sterowania pracownika	16
7	Lista dostępów do drzwi pracownika	17
8	Panel sterowania administratora	17
9	Edytowanie dostępów	18
10	Edytowanie drzwi	18
11	Edytowanie pracowników	19
12	Panel sterowania właściciela	20
13	Edytowanie przywilejów	20
14	Potwierdzenie zmiany właściciela	21
15	Usuwanie przywilejów	21

Spis tabel

1	Tabela Labourers	8
2	Tabela Door	8
3	Tabela Access	8