

1. Dokładny opis implementacyjny systemu (na podstawie dokumentacji)

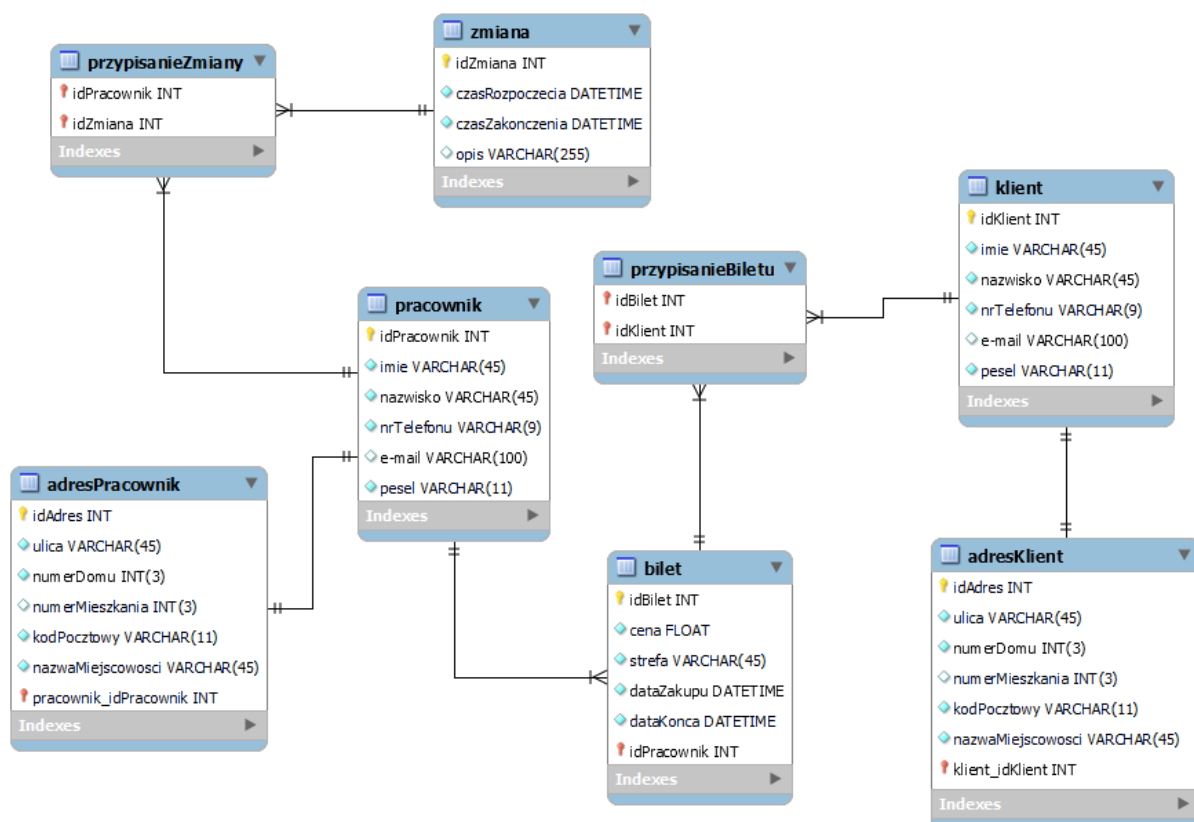
- W projekcie systemu zostały zaimplementowane funkcjonalności, na potrzebę przedstawionych w opisie firmy w dokumentacji projektowej takie jak rejestracja i logowanie użytkownika. Możliwość zakupu biletu przez wypełnienie formularza. Dla kierownika, księgowej i recepcjonistki są dostępne specjalne panele, których nie widzi podstawowy użytkownik, które dają im dostęp do specjalnych funkcjonalności. Kierownik posiada CRUDA z bazą pracowników i formularz do przypisywania im zmian, Recepcjonistka może zarządzać biletami, a księgowa sprawdzić i podsumować wydatki.
- W systemie każdy interfejs jest zgodny z diagramem przypadków użycia do podanych funkcjonalności.
- Nasz system informatyczny wdraża dokładne kroki podane w scenariuszach powiązanych z DPU. Nowy klient wchodzący na stronę z zaimplementowanym systemem ma dostęp każdej ze stref zdefiniowanych w opisie, które umożliwiają mu wykonanie powyższych funkcjonalności takich jak zakupienie biletu.
- System umożliwia przejście przez zawarte w scenariuszach przebiegi alternatywne takie jak na przykład wyświetlenie komunikatów o błędzie w momencie wprowadzenia błędnych danych do systemu.
- W systemie zostały zawarte wszystkie klasy wymienione w obiektowym modelu danych, między którymi funkcjonują określone wcześniej w dokumentacji związki i atrybuty. Funkcjonuje on w sposób, który umożliwia przeprowadzenie dokładnego przypadku zilustrowanego na diagramie obiektów.
- Do systemu zaimplementowany został projekt strony głównej, który zawiera w sobie podstawowe informacje podane w opisie firmy oraz możliwość przejścia do poszczególnych projektów interfejsów znajdujących się na kolejnych podstronach.
- Wszystkie interfejsy i ich funkcjonalności są zgodne z funkcjami wymienionymi w diagramie FHD dla poszczególnych użytkowników systemu. Powyższe funkcjonalności są zamieszczone w dedykowanych dla określonej grupy uprawnień użytkowników w postaci panelu użytkownika. Każdy z użytkowników z uprawnieniami wyższymi niż podstawowe ma dostęp do innego rodzaju funkcjonalności pokrywających się z diagramem FHD.
- Interfejs jest zgodny z wcześniej założonymi wymaganiami tj. przydatność, łatwość, estetyka, bezpieczeństwo i stabilność.
- Wszelkie wymagania zostały podstawowo przetestowane, a ich różnice od założonych będą opisane dokładniej w kolejnej części dokumentacji.

2. Zmiany w implementacji względem założeń projektowych

- Szata graficzna projektów została zmieniona, ponieważ projekty zawarte w dokumentacji były jedynie szkicami i zamysłami samych interfejsów. Z tymi zawartymi w dokumentacji zgadza się struktura i zawarte informacje, lecz są one rozszerzone o dodatkowe pola oraz odpowiednio ostylowane.
- Nie wykonaliśmy testów wymagań metrycznych zawartych w dokumentacji, ponieważ nie mamy na to wystarczających środków. Testy zostałyby wykonane w momencie zakupu systemu przez firmę i odpowiedniego dofinansowania. Wykonaliśmy natomiast podstawowe testy. Odnosząc się do bezpieczeństwa sprawdziliśmy czy strona jest odporna na podstawowe i proste ataki (SQL Injection, Blind SQL).
- Resztę testów metrycznych wykonaliśmy bez zatrudniania dodatkowych osób. Przetestowaliśmy je wśród znajomych, a stabilność systemu sprawdziliśmy za pomocą maszyny do DDOS.
- Nie umieściliśmy wybranego w dokumentacji rozwiązania dla osoby niepełnosprawnej. Nie byliśmy w stanie zaimplementować wystarczających udogodnień dla osoby niedowidzącej w określonym czasie do oddania systemu.
- Nie podłączyliśmy zewnętrznego systemu realizacji płatności do naszego systemu, ponieważ zostawiliśmy wybór współpracy dla firmy/klienta.
- W systemie nie zaimplementowaliśmy funkcji reklamacji, ponieważ nie zdążyliśmy jej opracować do momentu oddania dokumentacji. Możliwe, że reklamacja pojawi się jako przyszły „feature” w systemie przed momentem oddania projektu końcowego.
- Niektóre z interfejsów zmieniły swoją domyślną funkcjonalność i są teraz czytelniejsze i łatwiejsze do zrozumienia dla użytkowników systemu. Dodatkowo zmieniliśmy sposób w jaki funkcjonują niektóre z funkcjonalności na potrzebę optymalizacji i prostszego działania systemu takich jak inna forma wybrania pracownika. Skutek końcowy jest taki sam jak w pierwotnym projekcie.
- Postawiliśmy tymczasowo na brak implementacji możliwości zakupu biletów dla użytkowników, którzy nie posiadają konta w serwisie, ponieważ wdrożenie takiej funkcjonalności było zbyt czasochłonne i zdecydowaliśmy, że łatwiej będzie wdrożyć system początkowo jedynie dla osób posiadających aktywne konto.

3. Schemat zaimplementowanej bazy danych

W systemie został zastosowany schemat bazy danych na podstawie schematu uwzględnionego wcześniej w dokumentacji. Struktury i relacje podanych danych pokrywają się z wersją pierwotną z odpowiednim zastosowaniem kluczy obcych i ich referencyjności.



4. Specyfikacja środowiska realizacji i struktury aplikacji

Wybrane środowisko:

- PyCharm
- Visual Studio Code

Backend:

- Python

Frameworki:

- Django

Frontend:

- HTML
- CSS
- JavaScript

Baza Danych:

- SQLite

System kontroli wersji:

- GIT

Repozytorium:

- Github

Zdecydowaliśmy się na umieszczenie całego projektu na repozytorium github, ponieważ oferuje łatwą i szybką obsługę i wiele możliwości pracy zespołowej. Do zmian projektowych używamy systemu kontroli wersji GIT przy pomocy CLI jak i wbudowanych funkcji używanych w IDE. Jako, że w projekcie zdecydowaliśmy się na utworzenie backendu za pomocą Pythona i frameworka Django to wzorzec projektowy w jakim pracujemy to MVT (Model-View-Template)

W momencie gdy użytkownik żąda danego zasobu ze strony, Django działa jak kontroler i sprawdza dostępne zasoby dla podanego adresu URL. Wszystkie żądania aplikacji są przetwarzane na podstawie pliku konfiguracyjnego `urls.py`. Następnie framework zajmuje się parsowaniem adresu URL przychodzącego żądania i przekazuje dane do różnych modułów. W tym modułach znajdują się modele, widoki i szablony. Jeśli URL jest dostępny to wywoływany jest widok, który prowadzi interakcję z odpowiednim modelem i szablonem. Właśnie wtedy renderowany jest szablon, a następnie Django odpowiada użytkownikowi i wysyła dany szablon jako odpowiedź. W naszym przypadku szablony znajdują się w katalogu `templates` w postaci plików HTML i są one połączone z frontendem, który znajduje się w katalogu `static`. Do frontendu używamy jedynie CSS i JavaScript. Nie używamy żadnych dodatkowych frameworków. Nasz system komunikuje się z bazą danych SQLite za

pomocą modeli, które są reprezentacją bazy danych w kodzie. Ich komunikacja polega na migracji danych z modelu do bazy, w której przechowywane są rekordy.

5. Jak SI działa

Uruchamiane moduły i wykorzystywane biblioteki:

Do wyświetlenia funkcjonalności na stronie służą nam widoki. Na stronie głównej generujemy formularz kontaktowy za pomocą funkcji `homePage(request)`, która sprawdza czy formularz został odpowiednio wypełniony oraz renderuje go do szablonu html w postaci stringa. Następnie uruchamiana jest metoda `send_mail`, która wysyła maila na adres firmy i zwraca użytkownikowi wiadomość o sukcesie. Ostatecznie widok zwraca szablon `index.html` zawierający szablon strony.

```
def homePage(request):
    if request.method == "POST":
        form = ContactForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            email = form.cleaned_data['email']
            message = form.cleaned_data['message']

            html = render_to_string('App/contact_form.html', {
                'name': name,
                'email': email,
                'message': message
            })

            send_mail('Formularz kontaktowy', 'wiadomosc', 'pocalunekneptuna@gmail.com', ['pocalunekneptuna@gmail.com'],
                    html_message=html)
            messages.info(request, 'Successfully Sent The Message!')
            return redirect('home')
        else:
            form = ContactForm()

    context = {
        'form': form,
    }
    return render(request, 'App/index.html', context)
```

Logowanie odbywa się w widoku `login`, który sprawdza czy podane zostały prawidłowe dane i zwraca szablon `login.html`

```
def login(request):
    if request.method == 'POST':
        username = request.POST.get('email')
        password = request.POST.get('password')

        try:
            user = User.objects.get(username=username)
        except:
            messages.error(request, 'Nazwa użytkownika jest niepoprawna')
            return render(request, 'App/subpages/login.html')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            auth_login(request, user)
            return redirect('home')
        else:
            messages.error(request, 'Hasło nie jest zgodne.')
            return render(request, 'App/subpages/login.html')

    return render(request, 'App/subpages/login.html')
```

Operacje CRUD w panelach są tworzone schematycznie i wyglądają tak jak przykładowy widok dla panelu menadżera.

```
def manager_employees(request):
    workers = Worker.objects.all()
    context = {'workers': workers}
    return render(request, 'App/subpages/manager/manager_employees.html', context)
```

```
def manager_employees_show(request, worker_id):
    worker = get_object_or_404(Worker, pk=worker_id)
    workerAddress = get_object_or_404(WorkerAddress, worker=worker)
    context = {'worker': worker,
               'workerAddress': workerAddress}
    return render(request, 'App/subpages/manager/manager_employees_show.html', context)
```

Widok manager_employees odpowiada za wyświetlanie wszystkich pracowników w bazie danych i przekazuje je do szablonu, w którym są wypełniane odpowiednie rubryki

```
{% for worker in workers %}
<tr>
    <td>{{worker.name}}</td><td>{{worker.surname}}</td><td>{{worker.phoneNumber}}</td>
</tr>
{% endfor %}
```

Następnie po wybraniu opcji wyświetlenia jednego z pracowników odnosimy się do widoku manager_employees_show, który wyświetla dane z modelu pracownika oraz adresu pracownika w tabeli

```

<form action="">
  <h1>Dane Pracownika</h1>
  <table>
    <tr>
      <td>ID</td><td>{{ worker.id }}</td>
    </tr>
    <tr>
      <td>Imię</td><td>{{ worker.name }}</td>
    </tr>
    <tr>
      <td>Nazwisko</td><td>{{ worker.surname }}</td>
    </tr>
    <tr>
      <td>Numer Telefonu</td><td>{{ worker.phoneNumber }}</td>
    </tr>
    <tr>
      <td>E-mail</td><td>{{ worker.email }}</td>
    </tr>
    <tr>
      <td>Pesel</td><td>{{ worker.pesel }}</td>
    </tr>
    <tr>
      <td>Ulica</td><td>{{ workerAddress.street }}</td>
    </tr>
    <tr>
      <td>Numer Domu</td><td>{{ workerAddress.houseNumber }}</td>
    </tr>
    <tr>
      <td>Numer Mieszkania</td><td>{{ workerAddress.flatNumber }}</td>
    </tr>
    <tr>
      <td>Kod Pocztowy</td><td>{{ workerAddress.postcode }}</td>
    </tr>
    <tr>
      <td>Miejscowość</td><td>{{ workerAddress.placeName }}</td>
    </tr>
  </table>

  <a href="{% url 'manager_employees' %}">
    <input type="button" value="Wróć">
  </a>
</form>

```

Za podstawową wersję dodawania pracowników (CREATE) odpowiada widok `manager_employees_add`, który przesyła formularz, sprawdza czy jest poprawnie wypełniony i zwraca szablon strony i dodaje nowy rekord do bazy.

Modele, które odpowiadają za bazę danych wyglądają szablonoowo tak jak poniższy przykład:

```

class Worker(models.Model):
    name = models.CharField(max_length=45)
    surname = models.CharField(max_length=45)
    phoneNumber = models.CharField(max_length=9, unique=True)
    email = models.EmailField(null=True, blank=True, unique=True)
    pesel = models.CharField(max_length=11, unique=True)

    def __str__(self):
        return f'{self.name} {self.surname}'

class WorkerAddress(models.Model):
    worker = models.OneToOneField(Worker, on_delete=models.CASCADE)
    street = models.CharField(max_length=45)
    houseNumber = models.IntegerField()
    flatNumber = models.IntegerField(null=True, blank=True)
    postcode = models.CharField(max_length=11)
    placeName = models.CharField(max_length=45)

    def __str__(self):
        return f'{self.worker}'

```

Każde z pól ma określony typ oraz podstawową wbudowaną walidację na długość wpisanego tekstu, opcjonalności wypełnienia oraz metodę do wypisywania potrzebnych nam wartości.

Żeby użytkownik mógł zakupić bilet musi mieć konto w serwisie. Aby je utworzyć musi wypełnić formularz rejestracji, którego widok jest następujący

```

def registration(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('home')
        else:
            form = RegistrationForm()
    return render(request, 'App/subpages/registration.html', {'form': form})

```

I połączony z takim formularzem walidującym odpowiednio podstawowe warunki:


```
class CreateWorkerForm(ModelForm):
    class Meta:
        model = Worker
        fields = "__all__"

        labels = {
            'name': 'Imie',
            'surname': 'Nazwisko',
            'phoneNumber': 'Numer telefonu',
        }

    def clean_name(self):
        name = self.cleaned_data.get('name')
        if re.search(r'\d', name):
            raise ValidationError('Pole nie może zawierać liczb')
        return name

    def clean_surname(self):
        surname = self.cleaned_data.get('surname')
        if re.search(r'\d', surname):
            raise ValidationError('Pole nie może zawierać liczb')
        return surname

    def clean_phoneNumber(self):
        number = self.cleaned_data.get('phoneNumber')
        if re.search(r'[a-zA-Z]', number):
            raise forms.ValidationError("Pole nie może zawierać liter")
        if len(number) != 9:
            raise ValidationError('Numer telefonu musi zawierać 9 cyfr')
        if number[0] == '0':
            raise ValidationError('Numer nie może się zaczynać od 0')
        return number

    def clean_pesel(self):
        pesel = self.cleaned_data.get('pesel')
        if re.search(r'[a-zA-Z]', pesel):
            raise forms.ValidationError("Pole nie może zawierać liter")
        if len(pesel) != 11:
            raise ValidationError('Pesel musi zawierać 11 cyfr')
        return pesel
```

6. Umieszczenie aplikacji

Nasza aplikacja jest umiejscowiona lokalnie. Poniżej znajdzie się instrukcja instalacji i uruchomienia aplikacji:

1. Na początku trzeba udać się na repozytorium github, na którym znajduje się aplikacja (https://github.com/Vex0on/ICC_15_00/tree/main) i pobrać aplikację. Istnieją na to dwie możliwości:
 - a) Bezpośrednio z strony githuba pobrać zawartość gałęzi main jako ZIP. Aby to zrobić wystarczy wcisnąć zielony przycisk (<> CODE), a następnie po rozwinięciu menu wybrać opcję (Download ZIP)
 - b) Druga opcja, która jest przez nas rekomendowana to pobranie zawartości repozytorium za pomocą systemu kontroli wersji GIT. Aby to zrobić wystarczy wcisnąć zielony przycisk (<> CODE), a następnie skopiowanie linku do lokalizacji wyświetlającej się w zakładce HTTPS.

Na komputerze tworzymy katalog, który będzie naszym lokalnym repozytorium i za pomocą wiersza poleceń inicjalizujemy repozytorium komendą:

git init

Następnie łączymy się z repozytorium lokalnym za pomocą komendy:

git remote add origin < https://github.com/Vex0on/ICC_15_00.git >

Następnie ściągamy zawartość repozytorium zdalnego do lokalnego poleceniem:

git pull origin main

Informacje dostępu:

Recepcjonistka: recepcjonistka@gmail.com Hasło12345

Księgowa: ksiegowa@gmail.com Hasło12345

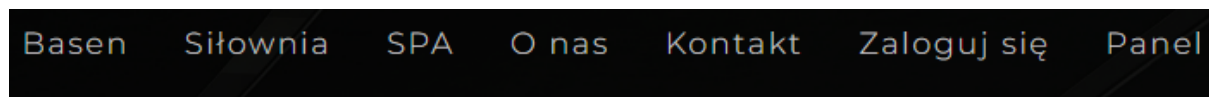
Kierownik: manager@gmail.com Hasło12345

Użytkownik: kowalczyk@gmail.com Hasło12345

Admin: superuser superuser

7. Instrukcja Obsługi

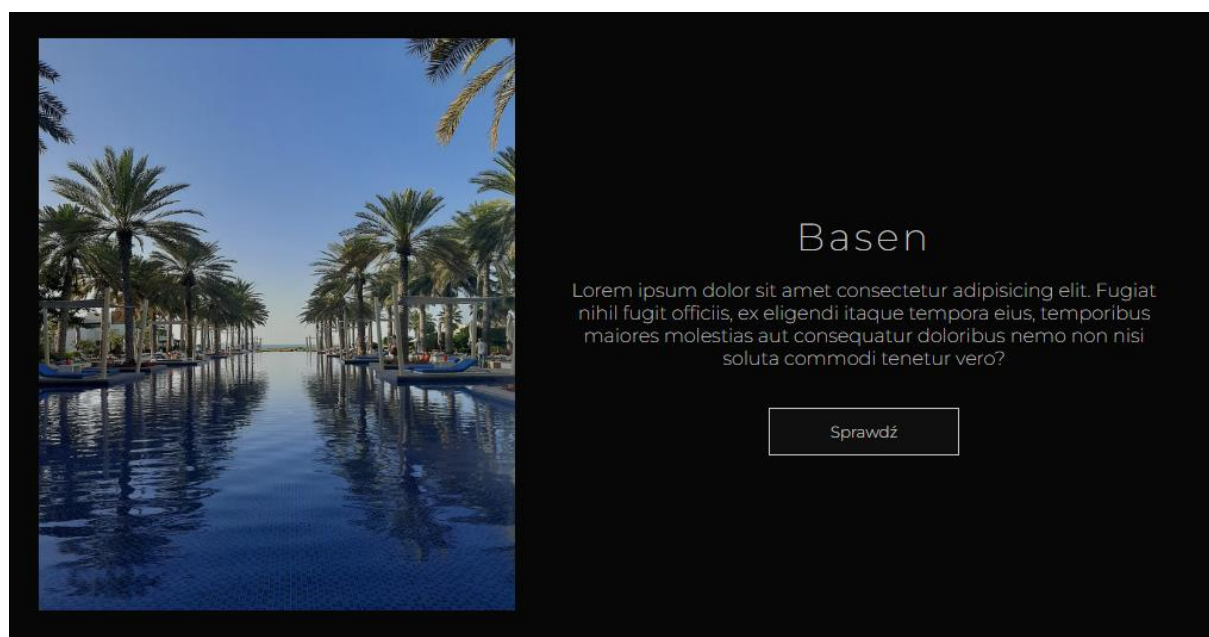
Po wejściu na stronę ukazuje nam się strona główna aplikacji. Na środku jest widoczna nazwa firmy, a na górze belka nawigacyjna.



Strona główna to rodzaj strony one-page. Oznacza to, że opcje dostępne w belce nie przeniosą użytkownika do oddzielnej podstrony, a jedynie zostanie on przeniesiony niżej do wybranej sekcji na tej samej stronie. Wyjątkiem jest „Zaloguj się”, które przenosi do osobnej podstrony z formularzem logowania oraz „Panel”, który jest dostępny jedynie dla zalogowanych użytkowników

Belka nawigacyjna:

1. **Basen/Siłownia/Spa** – Po kliknięciu użytkownik zostaje przekierowany do wybranej sekcji na stronie, która zawiera opis, zdjęcie obiektu i przycisk umożliwiający przejście do podstrony, na której znajduje się możliwość zakupienia biletu przypisanego do tej strefy. Aby zakupić bilet użytkownik musi wcisnąć przycisk „Sprawdź”.



- a) **Kupowanie Biletu** – Po przejściu do podstrony danej sekcji trafiamy na formularz, który użytkownik musi wypełnić swoimi danymi, aby móc zakupić bilet. Poniżej danych znajduje się możliwość dostosowania swojego zakupu czyli wybór rodzaju biletu. Dla każdej strefy dostępny jest bilet normalny jak i zniżka ulgowa lub grupowa. Do tego użytkownik może wybrać datę i godzinę o której bilet będzie aktywny. Jako, że bilet jest całodobowy to nie ma potrzeby wyboru daty zakończenia. Po wypełnieniu formularza użytkownikowi pozostaje wciśnięcie przycisku „Kup Bilet”

Kup Bilet

Dane Personalne

Imię

Nazwisko

E-mail

Pesel

Numer telefonu

Adres zamieszkania

Miejscowość

Kod pocztowy

Adres zamieszkania

Bilet

Rodzaj biletu

Wybierz bilet



Data i godzina wizyty:

dd.mm.rrrr --:--

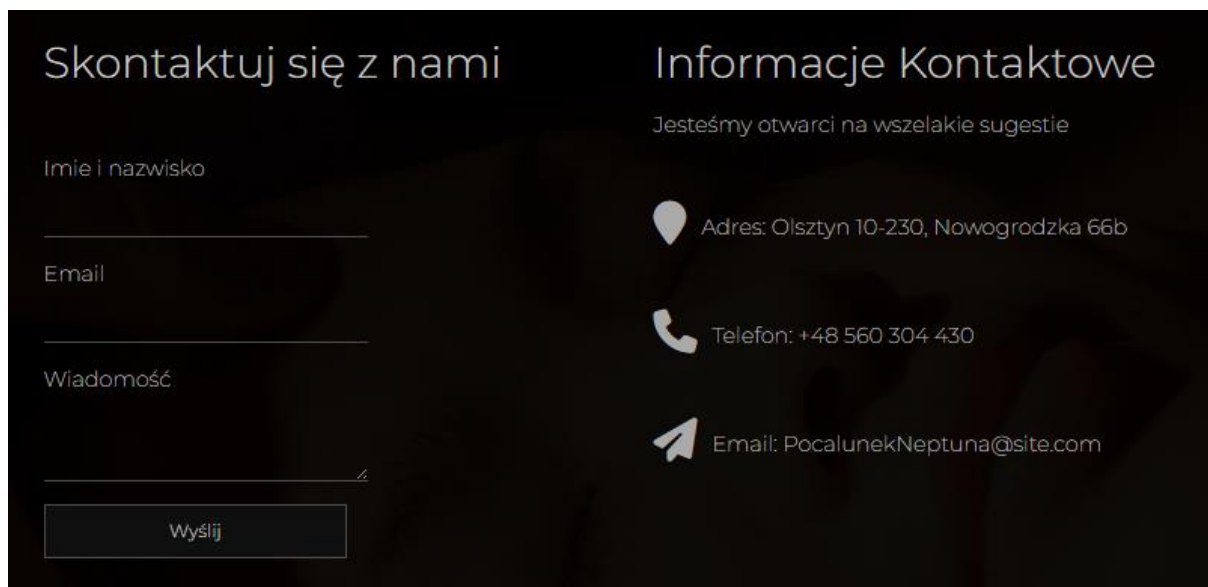


Kup Bilet

2. **O nas** - Po kliknięciu użytkownik zostaje przekierowany do sekcji zawierającej krótki opis oferty firmy.

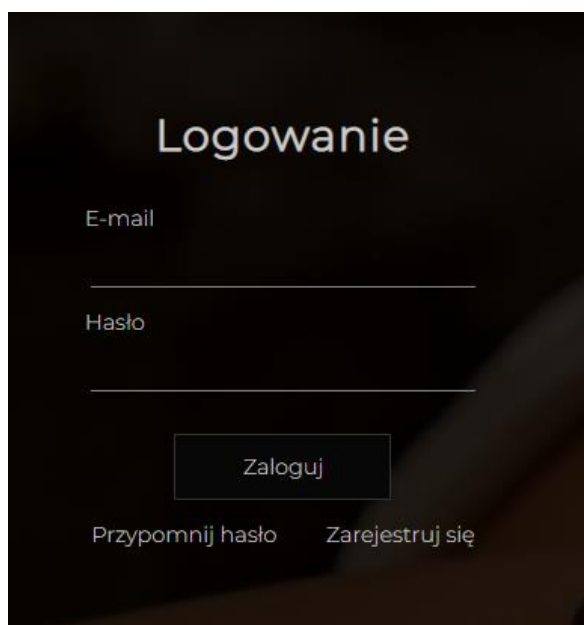
3. **Kontakt** – Po kliknięciu użytkownik zostaje przekierowany do sekcji kontaktowej podzielonej na dwie części. W lewej części znajduje się formularz kontaktowy z polami „Imię i nazwisko”, „Email” oraz „Wiadomość”. Po ich wypełnieniu i wciśnięciu przycisku „Wyślij” użytkownik przesyła wiadomość na maila firmowego.

W prawej części znajdują się informacje kontaktowe firmy takie jak adres, numer telefonu czy adres email.



The screenshot shows a contact page with two main sections. The left section, titled "Skontaktuj się z nami", contains a form with three input fields: "Imię i nazwisko", "Email", and "Wiadomość". Below these fields is a "Wyślij" button. The right section, titled "Informacje Kontaktowe", includes the text "Jesteśmy otwarci na wszelkie sugestie" and three contact details, each with an icon: a location pin for the address "Adres: Olsztyn 10-230, Nowogrodzka 66b", a telephone for "Telefon: +48 560 304 430", and an envelope for the email "Email: PocalunekNeptuna@site.com".

4. **Zaloguj się** – Po kliknięciu użytkownik zostaje przekierowany na osobną podstronę, na której znajduje się formularz logowania z polami „email” i „hasło”. Pod formularzem znajduje się opcja rejestracji dla nowych użytkowników jak i przypomnienia hasła dla osób już zarejestrowanych w serwisie.



The screenshot shows a login page titled "Logowanie". It features two input fields: "E-mail" and "Hasło". Below these fields is a "Zaloguj" button. At the bottom of the page, there are two links: "Przypomnij hasło" and "Zarejestruj się".

5. **Zarejestruj się** – Jeśli użytkownik nie ma jeszcze założonego konta w serwisie to może się w nim zarejestrować. Po wybraniu tej opcji zostanie przekierowany na osobną podstronę z dużym formularzem zawierającym dane osobowe oraz adres zamieszkania . Poniżej jest do zaznaczenia potwierdzenie zapoznania się z regulaminem serwisu oraz przycisk „Zarejestruj się” umożliwiający założenie konta w serwisie.

Dane rejestracyjne

Imię

Nazwisko

E-mail

Hasło

Powtórz hasło

Pesel

Data urodzenia

dd.mm.rrrr

Numer telefonu

Adres zamieszkania

Miejscowość

Kod pocztowy

Adres zamieszkania

☐ Zapoznałem się z [regulaminem](#).

Zarejestruj

Masz już konto? [Zaloguj się](#).

7. **Zapomniałeś hasła** – Opcja umożliwiająca użytkownikowi przypomnienie hasła przypisanego do jego adresu mailowego. Przenosi ona do podstrony, na której trzeba podać swój adres email i wcisnąć „Potwierdź”. W tym momencie na podany adres zostaje wysłane przypomnienie hasła.

Proszę podać swój adres e-mail używany podczas logowania się do serwisu i potwierdzić. Na podany adres zostanie wysłana wiadomość pozwalająca ustawić nowe hasło.

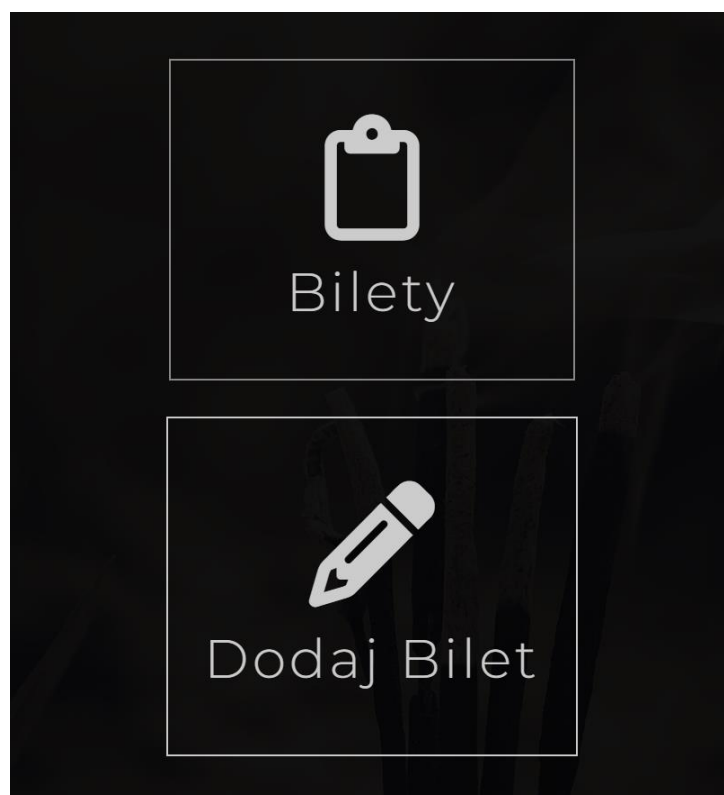
Email

Potwierdź

Wróć do logowania

8. Recepcjonistka

Panel Recepcjonistki wygląda następująco:



W panelu są przedstawione dwie funkcjonalności:

Bilety – przedstawia listę zakupionych biletów z informacjami takimi jak: ID, cena, strefa oraz data zakupu. Funkcjonalność ma na celu pomóc w weryfikacji czy dana osoba zakupiła bilet.

Lista Biletów

Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu
Id	Cena	Strefa	Data Zakupu

Wróć

Dodaj Bilet – jest to formularz z polami takimi jak: cena, strefa oraz data zakupu. Po wypełnieniu odpowiednich danych mamy możliwość dodania biletu do naszej bazy danych.

Dodaj Bilet

Cena

Strefa

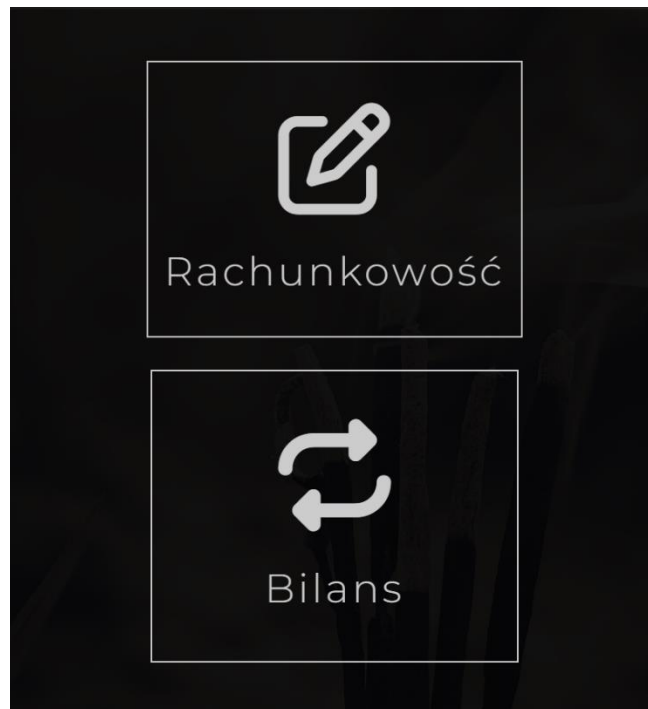
Data Zakupu

dd.mm.yyyy --:--

Wróć Dodaj

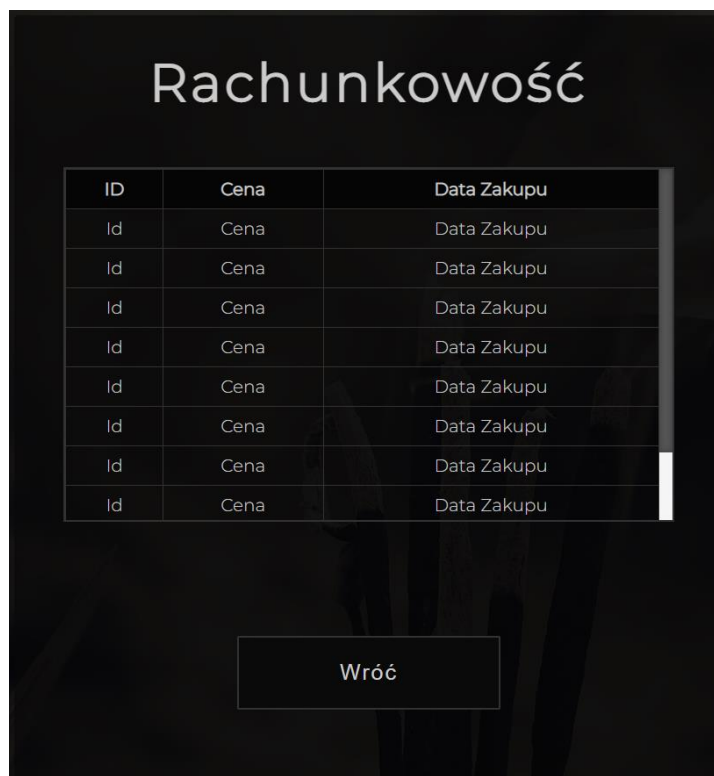
9. Księgowa

Panel Księgowej wygląda następująco:



W panelu są przedstawione dwie funkcjonalności:

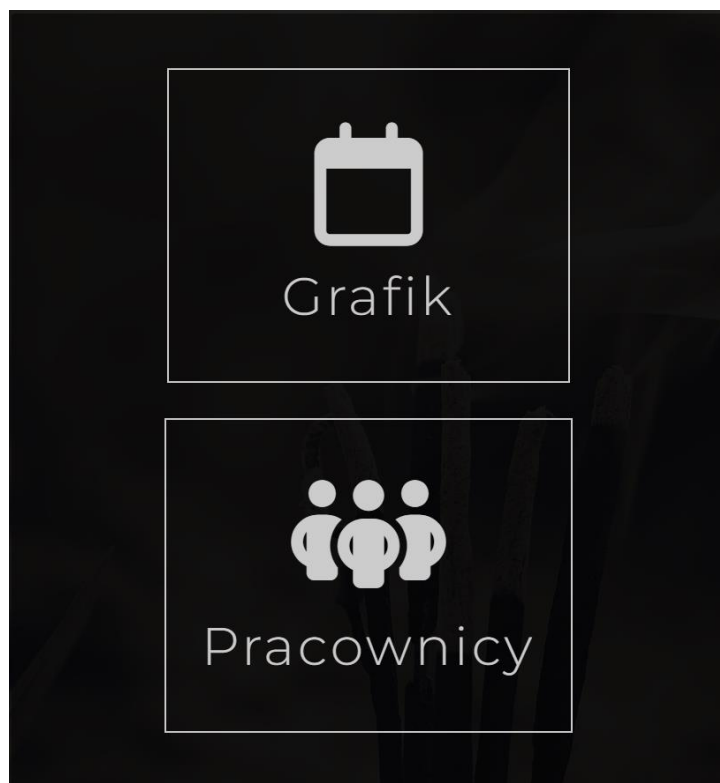
Rachunkowość – przedstawia listę wydatków oraz dochodów z takimi informacjami jak: ID, cena, Data Zakupu. Funkcjonalność ma na celu pomóc w analizie finansowej, przeglądzie i ocenie danych finansowych przedsiębiorstwa.



Bilans – funkcjonalność przedstawia formularz z polem „Data”. Po wybraniu określonego roku i miesiąca oraz kliknięciu przycisku „Oblicz”, system przedstawi sytuację finansową przedsiębiorstwa w danym momencie.

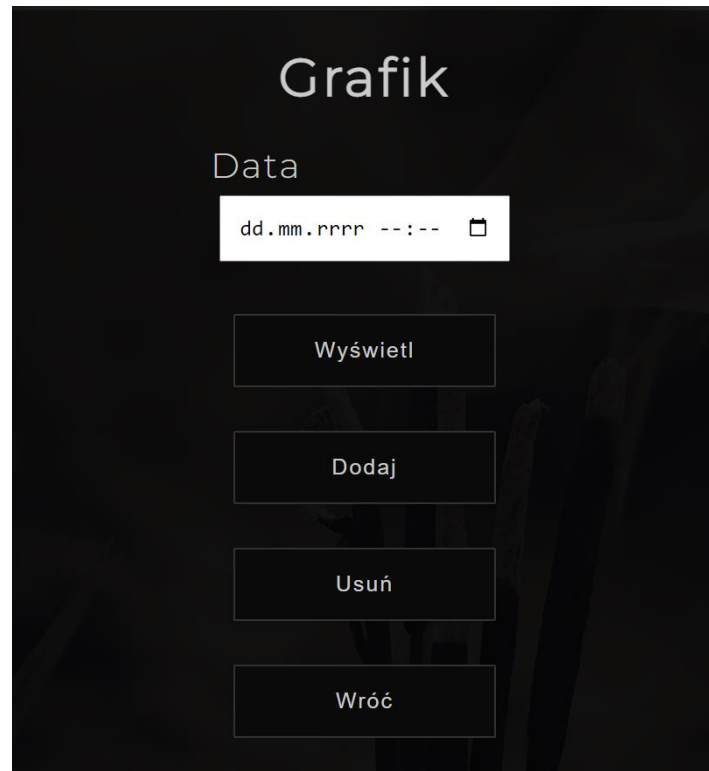
8. Kierownik

Panel Kierownika wygląda następująco:



W panelu są przedstawione dwie funkcjonalności:

Grafik – funkcjonalność przedstawia formularz z polem „Data” oraz 3 przyciskami odnoszącymi się do tego pola. Zastosowanie grafiku ma na celu pomóc w organizacji pracownikami.

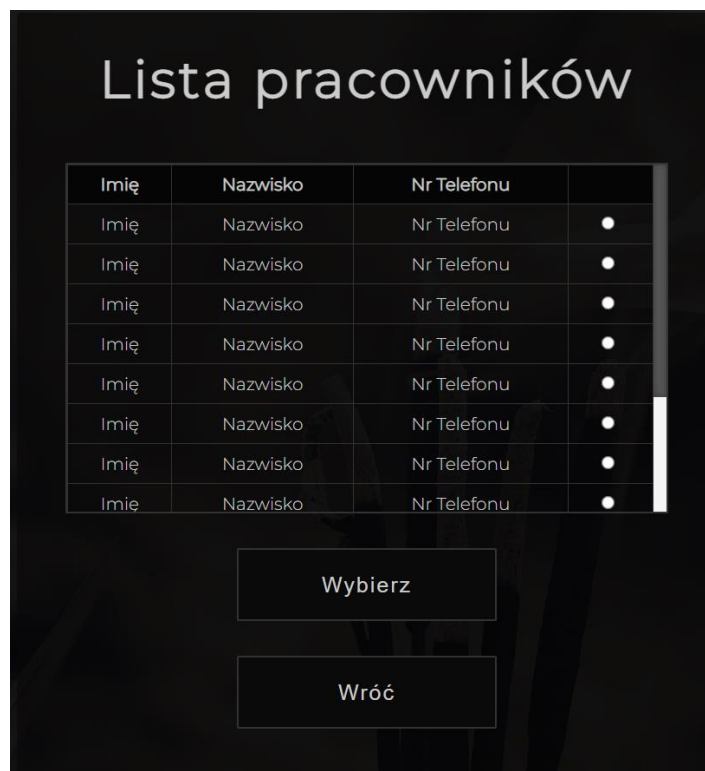


Przyciski:

Wyświetl – wyświetla listę pracowników, którzy są zapisani w grafiku pracy, dnia wpisanego w formularz.

Dodaj – wyświetla listę wszystkich pracowników, z możliwością dodania ich do grafiku pracy, dnia wpisanego w formularz.

Usuń – wyświetla listę pracowników, którzy są zapisani w grafiku pracy, dnia wpisanego w formularz z możliwością usunięcia dyżuru wybranego pracownika.



Pracownicy – funkcjonalność przedstawia listę pracowników oraz 2 przypisanego do niej przyciski. Celem tego jest edycja pracownikami w bazie danych. W wierszu każdego pracownika jest umieszczony przycisk, który przekierowuje nas do szczegółowych informacji na temat danego pracownika.



Dane Pracownika

ID	id
Imię	imie
Nazwisko	nazwisko
Numer Telefonu	telefon
E-mail	mail
Pesel	pesel
Ulica	ulica
Numer Domu	nrdom
Numer Mieszkania	nrmieszka
Kod Pocztowy	kod
Miejscowość	miestowosc

Wróć

Przyciski:

Dodaj – wyświetla formularz, w którym wpisujemy dane pracownika a następnie przyciskiem „prześlij” przesyłamy te dane do bazy danych co skutkuje dodaniem pracownika do bazy danych.

Dodaj Pracownika

Imię

Nazwisko

Numer telefonu

E-mail

Pesel

Adres zamieszkania

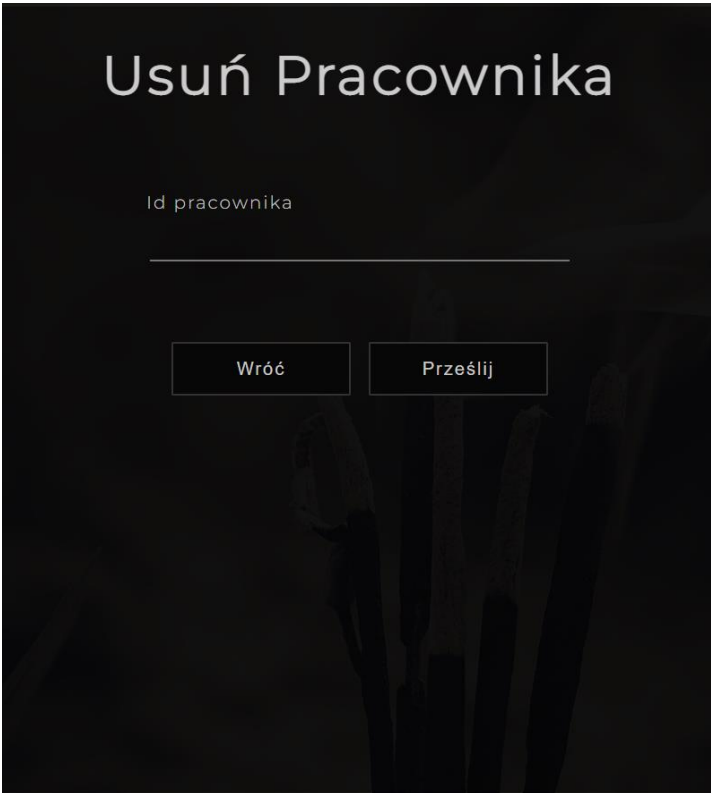
Kod pocztowy

Miejscowość

Wróć

Prześlij

Usuń – wyświetla formularz z polem „Id pracownika”, w którym wpisujemy ID danego pracownika, którego chcemy usunąć z bazy danych. Przyciskiem „Prześlij” potwierdzamy usunięcie.



Usuń Pracownika

Id pracownika

Wróć Prześlij

Dodatkowo system jest wyposażony w panel admina wbudowany w framework Django, który jest jeszcze w trakcie prac i zostanie uzupełniony do czasu oddania projektu.