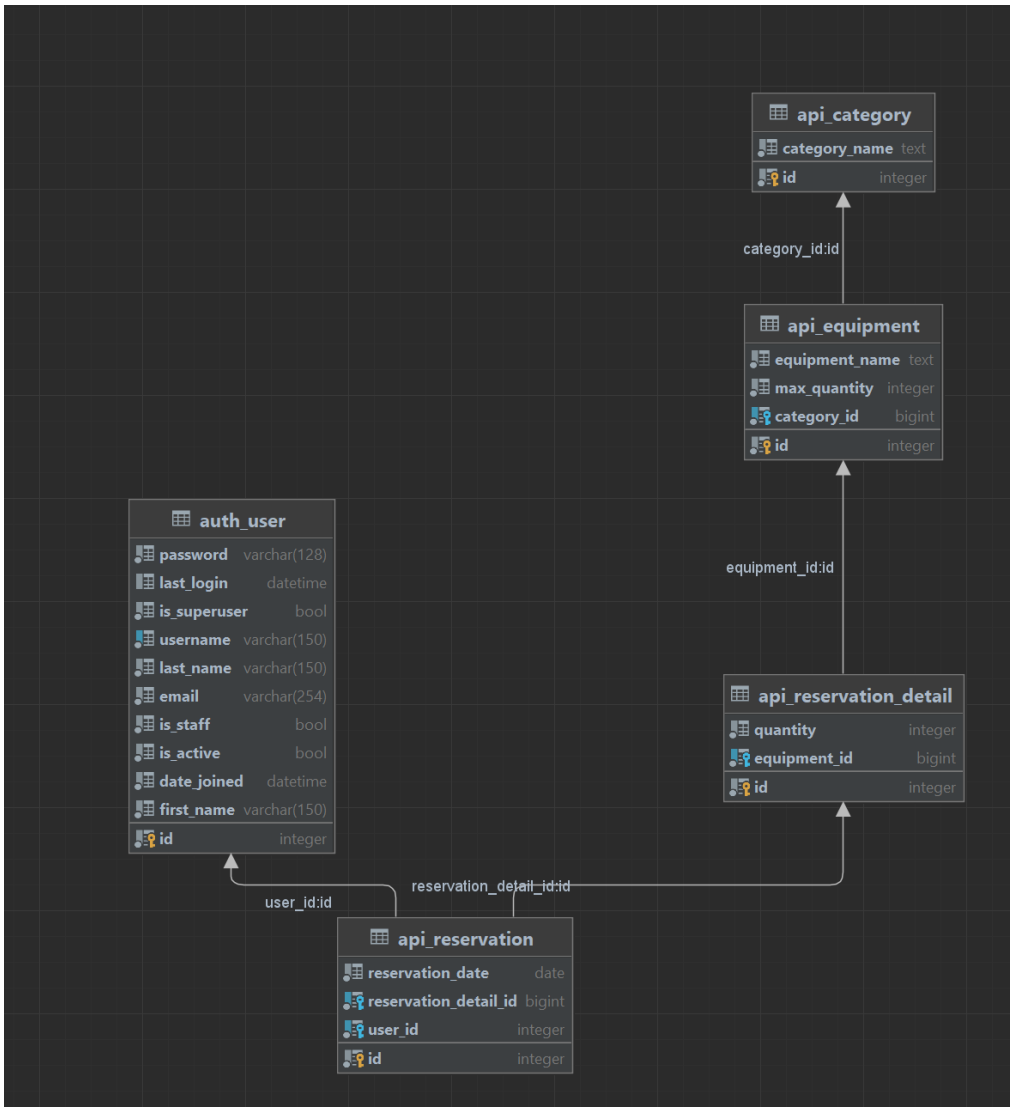
Aplikacja do rezerwacji sprzętu na siłowni

Autorzy: Bartłomiej Kozera, Mateusz Śmigała, Kacper Ćwiertnia

Spis treści

1. Schemat Bazy danych



2. Tabele

Słowem wstępu, prefiksy api oznaczają tabele stworzone przez nas, natomiast prefiks auth przy tabeli auth_user oznacza tabele stworzoną automatycznie przez django.

2.1. Tabela auth_user

Tabela przechowująca informacje o użytkownikach, domyślnie tworzona przez django.

```
create table auth_user
(
    id           integer      not null
        primary key autoincrement,
    password     varchar(128) not null,
    last_login   datetime,
    is_superuser bool         not null,
    username     varchar(150) not null
        unique,
    last_name    varchar(150) not null,
    email        varchar(254) not null,
    is_staff     bool         not null,
    is_active    bool         not null,
    date_joined  datetime     not null,
    first_name   varchar(150) not null
);
```

id	password	last_login	is_superuser
1	pbkdf2_sha256$600000$K3JWuu2PLIkrM31otFPzBu$bO2MMZbtP5BdS2h+fSZM0Ll13iB1yX023EHDtVidhjQ=	2023-06-18 16:11:18.030274	1
2	pbkdf2_sha256$600000$R0cnGKVQxvhvv3NOVwRXbZ$SHw2EdA9lKHAApjC7V4/biwaRCRuIHoTVxvFwXXP+EI=	<null>	0
3	pbkdf2_sha256$600000$7udFyMtHFfpCnZ25MEAdWo$+cj9ENKzpDk+svp3FXxv7jb0uQzLnnQIU6Wn7JQyj4A=	<null>	0
4	pbkdf2_sha256$600000$hGswoS4y0iie5J2mfV35EM$ncCVDuoDKX4V9V9wijqRPR3xx3uimHzZ2uPxHq3wIYM=	<null>	0

username	last_name	email	is_staff	is_active	date_joined	first_name
dbproject			1	1	2023-06-18 16:10:59.894794	
adam_zwycz	Zwyczajny		0	1	2023-06-18 16:11:38	Adam
piotroo	Nadzwyczajny		0	1	2023-06-18 16:12:06	Piotrek
tomas94	Niedziela		0	1	2023-06-18 16:12:33	Tomek

2.2. Tabela api_category

Tabela przechowująca informacje o dostępnych kategoriach sprzętu

```
create table api_category
(
    id            integer not null
        primary key autoincrement,
    category_name text    not null
);
```

id	category_name
1	Wolne ciężary
2	Gumy
3	Ławki
4	Maszyny

2.3. Tabela api_equipment

Tabela przechowująca dane o sprzęcie który można wypożyczyć w tej siłowni takie jak nazwa sprzętu, maksymalna dostępna ilość, oraz id kategorii do której dany sprzęt należy.

```
create table api_equipment
(
    id             integer not null
        primary key autoincrement,
    equipment_name text    not null,
    max_quantity   integer not null,
    category_id    bigint  not null
        references api_category
            deferrable initially deferred
);
```

id	equipment_name	max_quantity	category_id
1	Sztanga (5kg)	1	1
2	Sztanga (10kg)	2	1
3	Sztanga (15kg)	2	1
4	Hantel (2kg)	2	1
5	Hantel (5kg)	4	1
6	Hantel (8kg)	6	1
7	Guma Zielona (słaba)	3	2
8	Guma Czerwona (średnia)	2	2
9	Guma Fioletowa (mocna)	1	2
10	Ławka krótka	4	3
11	Ławka łamana	1	3
12	Ławka prosta	2	3
13	Ławka dodatnia	1	3
14	Bieżnia	10	4
15	Sztanga z suwnicą	1	4
16	Suwnica	1	4
17	Wyciąg	5	4

2.4. Tabela api_reservation

Tabela przechowująca dane o wszystkich rezerwacjach takie jak data rezerwacji, id użytkownika który zarezerwował dany sprzęt.

```
create table api_reservation
(
    id                    integer not null
        primary key autoincrement,
    reservation_date      date    not null,
    reservation_detail_id bigint  not null
        references api_reservation_detail
            deferrable initially deferred,
    user_id               integer not null
        references auth_user
            deferrable initially deferred
);
```

id	reservation_date	reservation_detail_id	user_id

2.5. Tabela api_reservation_details

Tabela przechowująca dane o rezerwacjach takie jak ilość zarezerwowanego sprzętu oraz identyfikator sprzętu.

```
create table api_reservation_detail
(
    id           integer not null
        primary key autoincrement,
    quantity     integer not null,
    equipment_id bigint  not null
        references api_equipment
            deferrable initially deferred
);
```

id	quantity	equipment_id

3. Relacje w bazie danych

3.1. Relacja api_reservation-> auth_users

Jest to relacja 1:N ze względu na połączenie klucza obcego user_id z tabeli api_reservation z kluczem głównym id tabeli auth_users.

```
create index api_reservation_user_id_46aa34ed
    on api_reservation (user_id);
```

3.2. Relacja api_reservation-> api_reservation_detail

Jest to relacja N:1 ze względu na połączenie klucza obcego reservation_detail_id z tabeli api_reservation z kluczem głównym id tabeli api_reservation_detail.

```
create index api_reservation_reservation_detail_id_02fdd2d3
    on api_reservation (reservation_detail_id);
```

3.3. Relacja api_reservation_detail-> api_equipment

Jest to relacja N:1 ze względu na połączenie klucza obcego equipment_id z tabeli api_reservation_detail z kluczem głównym id tabeli api_equipment.

```
create index api_reservation_detail_equipment_id_e50878d4
    on api_reservation_detail (equipment_id);
```

3.4. Relacja api_equipment-> api_category

Jest to relacja N:1 ze względu na połączenie klucza obcego category_id z tabeli api_equipment z kluczem głównym id tabeli api_category.

```
create index api_equipment_category_id_12b9f04b
    on api_equipment (category_id);
```

4. Modele

Modele w django są obiektami, w których definiujemy w jaki sposób elementy naszej bazy danych mają ze sobą wchodzić w interakcję. Definiują one w prost jak wyglądają tabele w naszej bazie danych oraz jak są połączone.

```
class Reservation(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    reservation_date = models.DateField()
    reservation_detail = models.ForeignKey('Reservation_detail', on_delete=models.CASCADE)

class Equipment(models.Model):
    equipment_name = models.TextField(max_length=50)
    max_quantity = models.IntegerField(validators=[validate_max_quantity])
    category = models.ForeignKey('Category', on_delete=models.CASCADE)

class Category(models.Model):
    category_name = models.TextField(max_length=50)

class Reservation_detail(models.Model):
    equipment = models.ForeignKey(Equipment, on_delete=models.CASCADE)
    quantity = models.IntegerField()
```

5. Serializery

Serializery to widoki na których pracujemy w naszej aplikacji, dane które możemy wysłać na frontend lub w dowolny sposób operować nimi na backendzie.

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'first_name', 'last_name']

class CategorySerializer(ModelSerializer):
    class Meta:
        model = Category
        fields = '__all__'

class ReservationSerializer(ModelSerializer):
    class Meta:
        model = Reservation
        fields = '__all__'

class ReservationDetailSerializer(ModelSerializer):
    class Meta:
        model = Reservation_detail
        fields = '__all__'

class EquipmentSerializer(ModelSerializer):
    class Meta:
        model = Equipment
        fields = '__all__'
```

6. Komunikacja z bazą danych

6.1. Funkcja znajdująca wszystkich użytkowników

```python
@api_view(['GET'])
def getUsers(request):
    users = User.objects.filter(is_staff=False)
    serializer = UserSerializer(users, many=True)
    return Response(serializer.data)
```

6.2. Funkcja znajdująca wszystkie kategorie

```python
@api_view(['GET'])
def getCategories(request):
    categories = Category.objects.all()
    serializer = CategorySerializer(categories, many=True)
    return Response(serializer.data)
```

6.3. Funkcja znajdująca wszystkie sprzęty z danej kategorii

```python
@api_view(['GET'])
@permission_classes([AllowAny])
def EquipmentByCategoryView(request, pk):
    category = Category.objects.get(id=pk)
    equipments = Equipment.objects.filter(category=category)
    serializer = EquipmentSerializer(equipments, many=True)
    return Response(serializer.data)
```

6.4. Funkcja zliczająca ile rezerwacji zostało zrobionych na dany sprzęt o danej dacie

```python
@api_view(['GET'])
@permission_classes([AllowAny])
def getCurrentReservedQuantity(request, pk, date):
    category = Category.objects.get(id=pk)
    equipments = category.equipment_set.all()
    equipment_serializer = EquipmentSerializer(equipments, many=True)

    data = []

    for el in equipment_serializer.data:
        choosen_equipment = el['id']
        reservation_details = Reservation_detail.objects.filter(equipment=choosen_equipment)
        reservation_detail_ids = reservation_details.values_list('id', flat=True)
        reservation = Reservation.objects.filter(reservation_detail__id__in=reservation_detail_ids)
        reservation_detail_serializer = ReservationDetailSerializer(reservation_details, many=True)
        reservation_serializer = ReservationSerializer(reservation, many=True)
        reservedYet = 0

        for i in range(len(reservation_detail_serializer.data)):
            if reservation_serializer.data[i]['reservation_date'] == date:
                reservedYet += reservation_detail_serializer.data[i]['quantity']

        data.append({'equipmentId': el['id'], 'equipmentName': el['equipment_name'],
                     'maxQuantity': el['max_quantity'],
                     'quantityLeft': el['max_quantity'] - reservedYet})

    return Response(data)
```

6.5. Funkcja tworząca widok rezerwacji zrobionych przez danego użytkownika.

Funkcja ta znajduje rezerwacje zrobione przez użytkownika i wypisuje dane na ich temat takie jak, id rezerwacji, data rezerwacji, id z tabeli api_reservation_detail, ilość zrobionych przez użytkonika rezerwacji, nazwę zarezerwowanego sprzętu.

```
@api_view(['GET'])
@permission_classes([AllowAny])
def getReservationsView(request, pk):
    active_user = User.objects.get(id=pk)
    reservation = Reservation.objects.filter(user=active_user)
    reservation_serializer = ReservationSerializer(reservation, many=True)
    reservation_detail_ids = [item['reservation_detail'] for item in reservation_serializer.data]
    reservation_detail = Reservation_detail.objects.filter(id__in=reservation_detail_ids)
    reservation_detail_serializer = ReservationDetailSerializer(reservation_detail, many=True)

    data = []

    for i in range(len(reservation_serializer.data)):
        equipment = Equipment.objects.filter(id = reservation_detail_serializer.data[i]['equipment'])
        equipment_serializer = EquipmentSerializer(equipment, many=True)
        data.append({'reservationId': reservation_serializer.data[i]['id'],
                     'date': reservation_serializer.data[i]['reservation_date'],
                     'reservationDetailsId': reservation_detail_serializer.data[i]['id'],
                     'quantity': reservation_detail_serializer.data[i]['quantity'],
                     'equipment': equipment_serializer.data[0]['equipment_name']
                     })

    return Response(data)
```

6.6. Funkcja dodająca rezerwację do tabel api_reservation_detail, api_reservation.

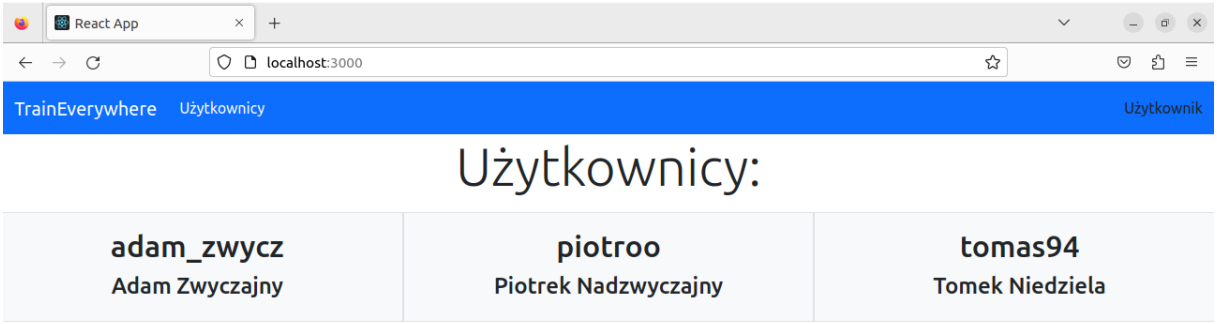
```
@api_view(['POST'])
def makeReservation(request):
    data = request.data
    reservation_owner = User.objects.get(id = data['user_id'])
    category = Category.objects.get(id = data['selected_category'])
    reservation_equipment = Equipment.objects.get(id = data['selected_equipment'])
    reservation_details = Reservation_detail.objects.create(equipment = reservation_equipment, quantity = data['reservation_qantity'])
    reservation = Reservation.objects.create(user = reservation_owner, reservation_date = data['reservation_date'], reservation_detail = reservation_details)

    serializer = ReservationDetailSerializer(reservation_details, many=False)
    serializer2 = ReservationSerializer(reservation, many=False)

    return Response(serializer2.data)
```

6.7. Funkcja usuwająca rezerwację o zadanym indeksie

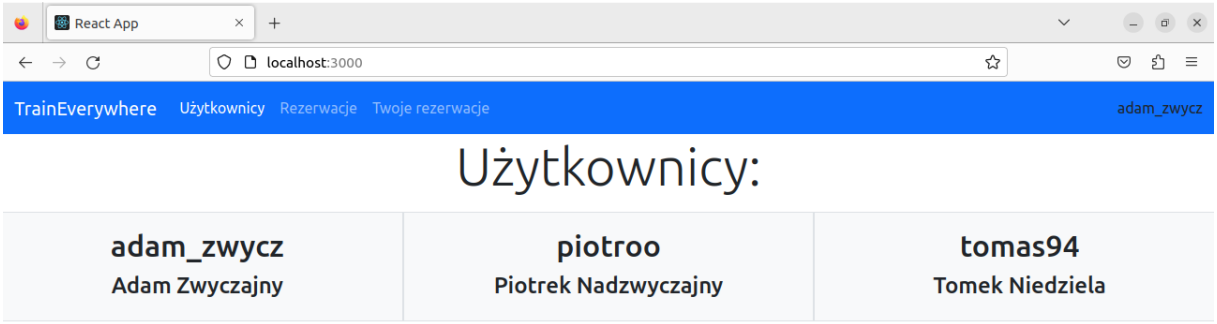
```
@api_view(['DELETE'])
def deleteReservation(request, pk):
    reservation = Reservation.objects.get(id=pk)
    reservation_serializer = ReservationSerializer(reservation, many=False)
    reservation_detail = Reservation_detail.objects.filter(id=reservation_serializer.data['reservation_detail'])

    reservation.delete()
    reservation_detail.delete()

    return Response("DB was updated!")
```

7. Widok z frontendu

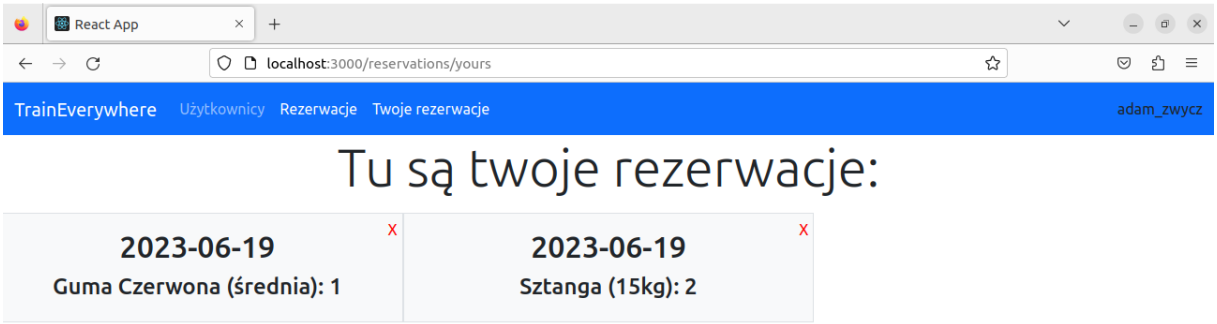
7.1. Ekran wyboru użytkowników, przed wybraniem użytkownika



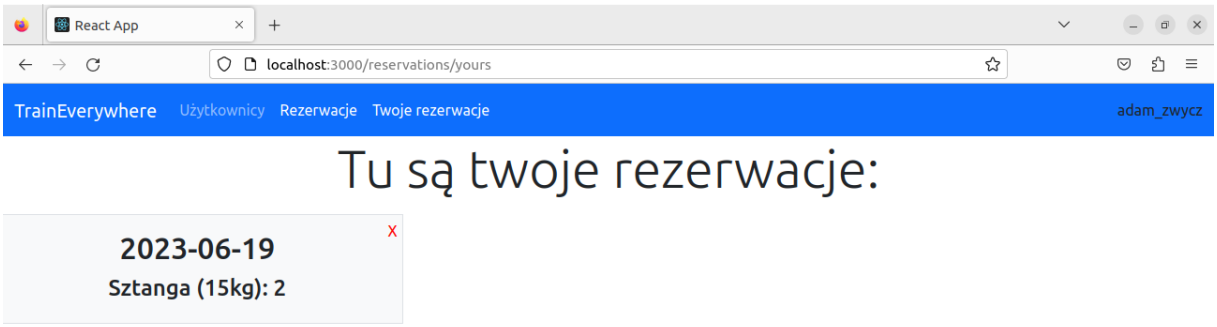
7.2. Ekran panelu wyboru użytkowników, po wybraniem użytkownika



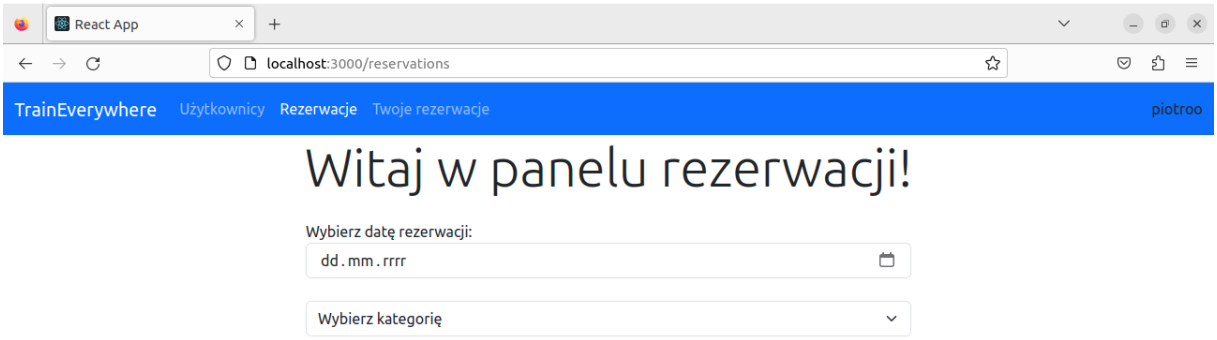
7.3. Ekran rezerwacji użytkownika



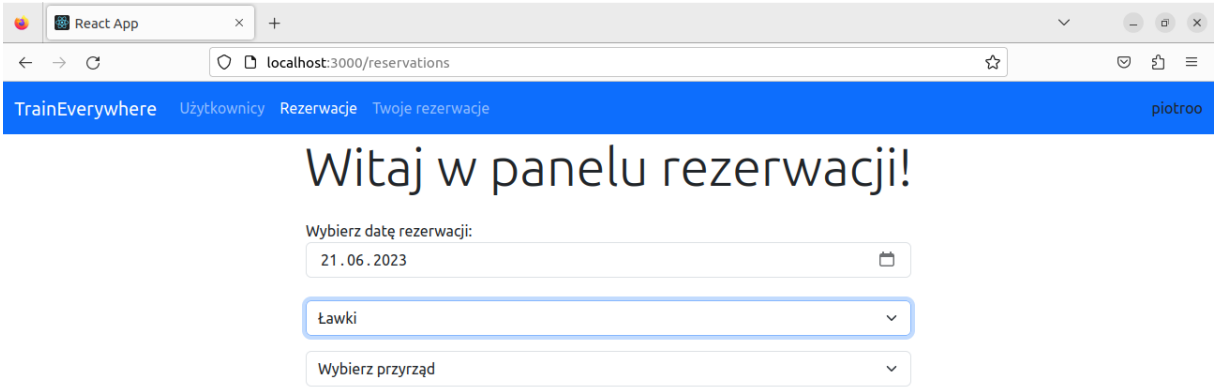
7.4. Ekran rezerwacji użytkownika po usunięciu rezerwacji na czerwoną gumę.



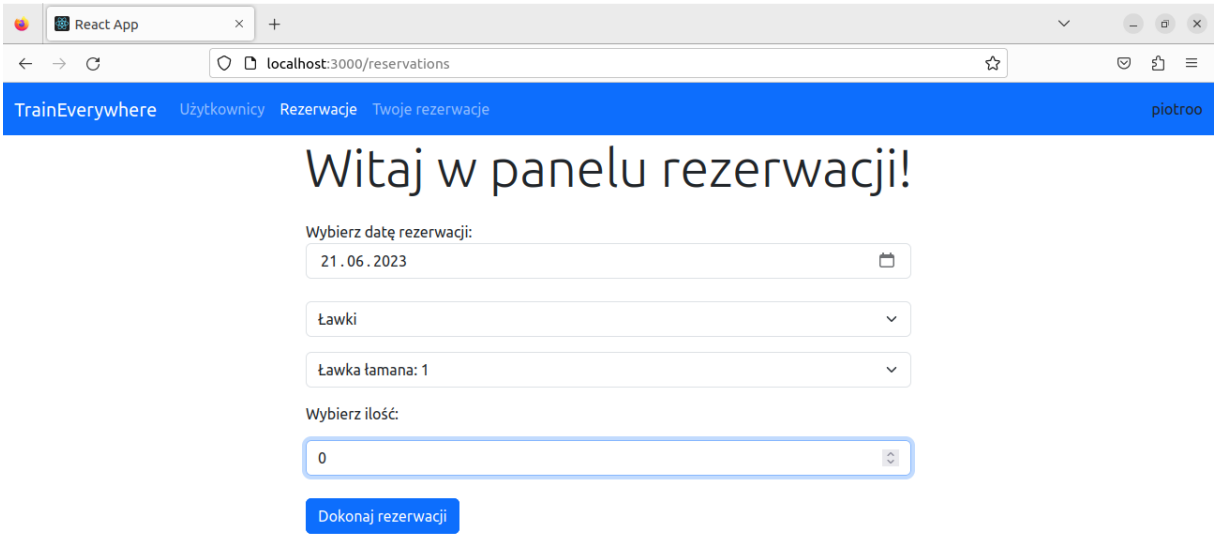
7.5. Ekran rezerwacji sprzętu, przed wyborem daty oraz kategorii



7.6. Ekran rezerwacji po wybraniu daty oraz kategorii, przed wybraniem sprzętu do rezerwacji



7.7. Ekran rezerwacji po zaznaczeniu wszystkich pól, przed dokonanie rezerwacji



7.8. Reset ekranu rezerwacji po jej dokonaniu