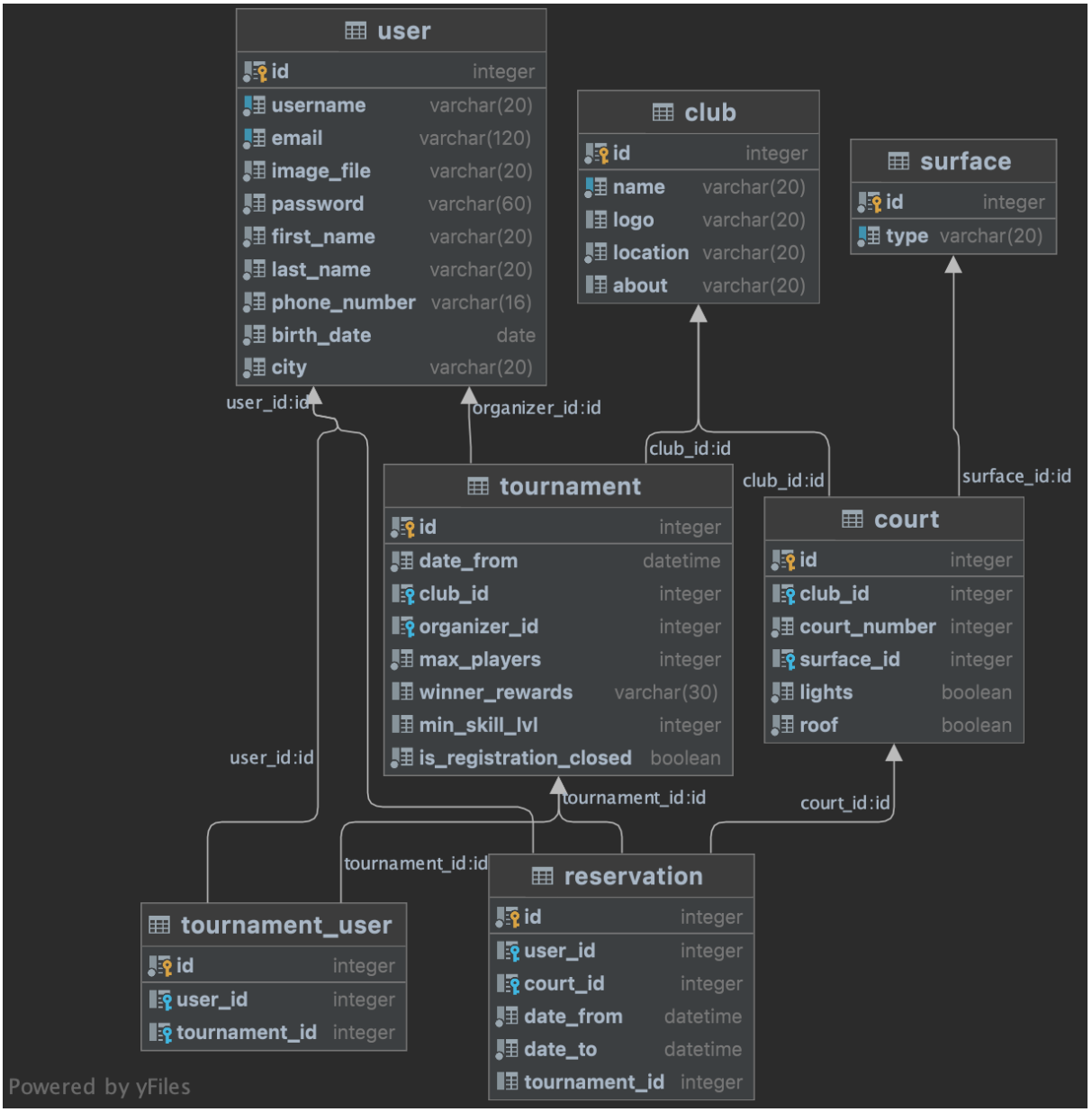
TENIS WORLD - DOKUMENTACJA

Krzysiek Miśkowicz, Bartosz Dudek, Dominik Jędraszek

Technologie:

- Flask
- SQLite
- Python

Diagram bazy:



Klasy:

- Użytkownik (User)
- Kort tenisowy (Court)
- Powierzchnia kortu (Surface)
- Rezerwacja (Reservation)
- Klub tenisowy (Club)

przykładowa klasa:

```python
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), unique=True, nullable=False)
    email = db.Column(db.String(20), unique=True, nullable=False)
    image_file = db.Column(db.String(20), nullable=False, default='default.jpg')
    password = db.Column(db.String(30), nullable=False)
    first_name = db.Column(db.String(20), nullable=False)
    last_name = db.Column(db.String(20), nullable=False)
    birth_date = db.Column(db.Date, nullable=False)
    phone_number = db.Column(db.String(16), nullable=False)
    city = db.Column(db.String(20), nullable=False)
```

Funkcjonalności:

- Otrzymanie kortów po: nazwie, numerze lub powierzchni,
- Pobieranie dostępnych klubów,
- Wypisywanie dostępnych powierzchni kortów,
- Zwracanie rezerwacji użytkownika,
- Zwracanie rezerwacji dla wybranej daty i kortu,
- Zwracanie lokalizacji dla wybranej rezerwacji (nazwa klubu, numer kortu)

Strony:

- Home - strona główna
- About - informacje o stronie
- Register - zakładanie konta
- Login - logowanie
- Logout - wylogowanie
- Account - informacje o użytkowniku
- My reservation - informacje o rezerwacjach użytkownika
- Cancel reservation - usuwanie rezerwacji użytkownika
- **Reservation** - informacje o wybranym korcie (dostępność dla wybranych godzin, możliwość rezerwacji terminu)

Formularze:

- Zakładanie konta
- Logowanie na nie
- Aktualizacja informacji o koncie
- Filtrowanie rezerwacji
- Rezerwowanie kortu

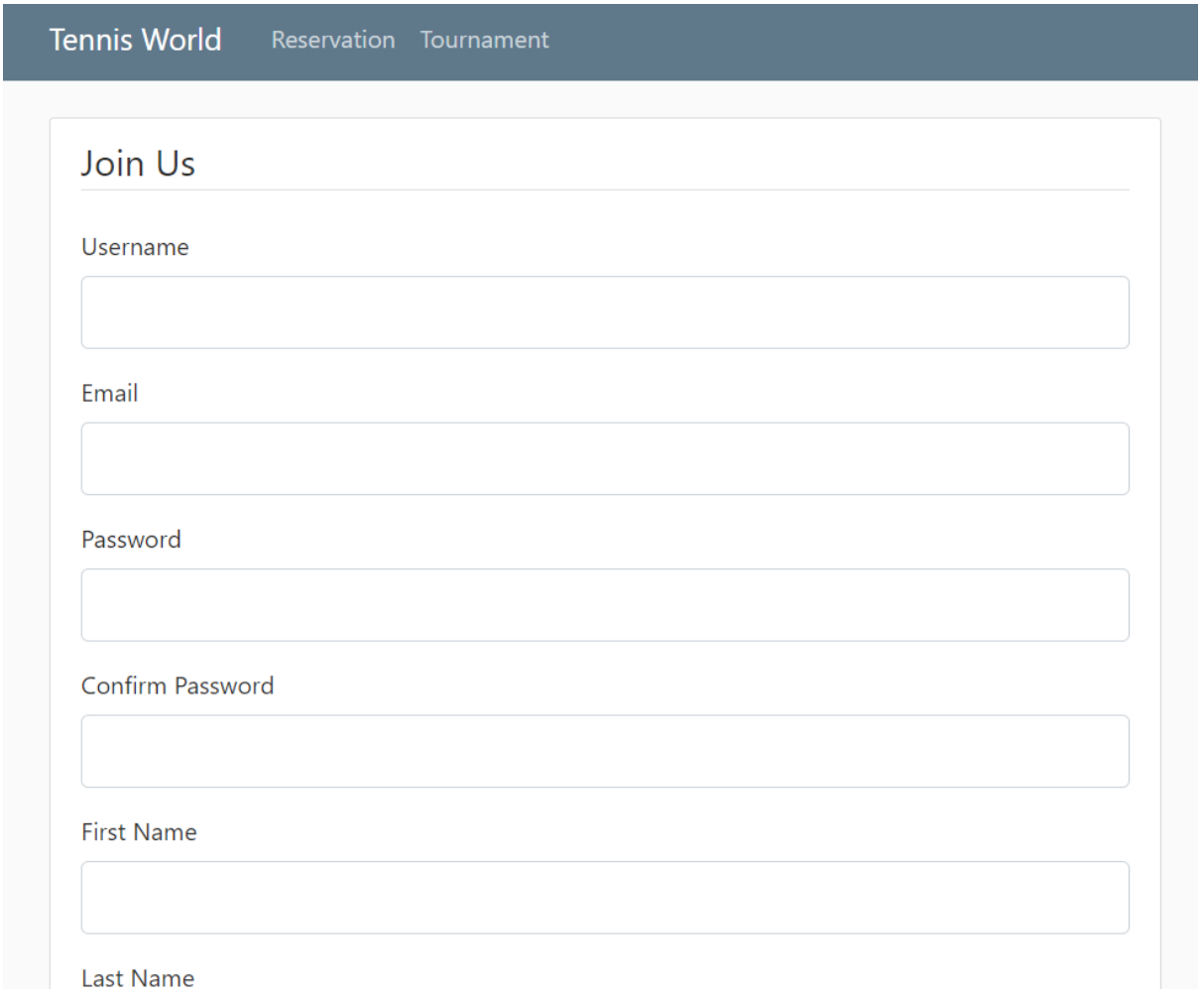
Przypadki użycia:

Zakładanie konta:

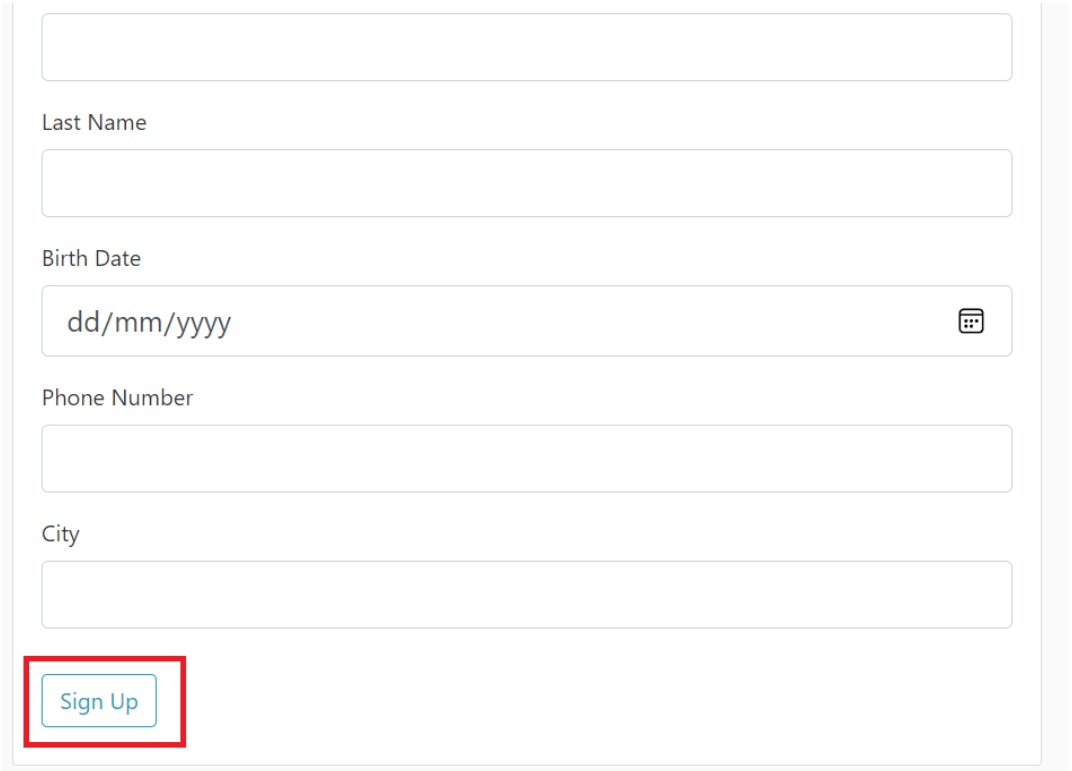
- wchodzimy na stronę główną ./home

- tam znajduje się przycisk register

- po kliknięciu uzupełniamy formularz rejestracji



- na koniec klikamy przycisk sign up na samym dole



Zakładanie konta - kod:

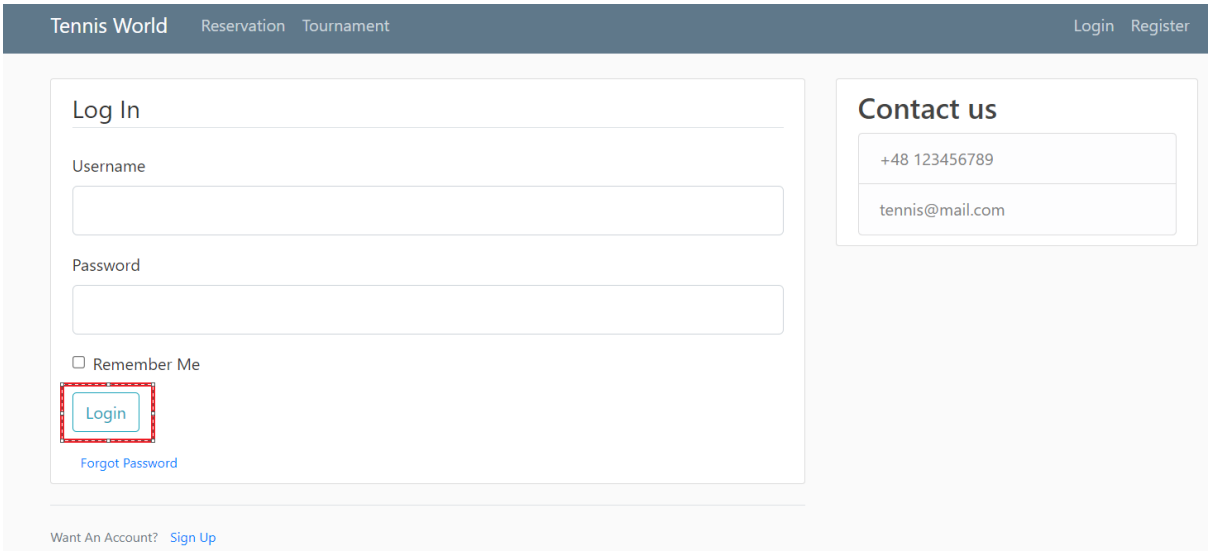
- podczas zakładania konta dane są pobierane z formularza i tworzony jest nowy obiekt user a następnie dodawany do tabeli

```python
@app.route("/register", methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data, password=hashed_password,
                    first_name=form.first_name.data, last_name=form.last_name.data, birth_date=form.birth_date.data,
                    phone_number=form.phone_number.data, city=form.city.data)
        db.session.add(user)
        db.session.commit()
        flash(f'{form.username.data} created account! You are able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', title='Register', form=form)
```

Logowanie - aktualizacja - wylogowywanie:

- klikamy przycisk login na stronie startowej

- następnie uzupełniamy hasło i login i klikamy przycisk login na dole



- po zalogowaniu pasek na górze strony zmienia wygląd - dostajemy nowe opcje do wyboru i w celu wylogowania klikamy przycisk logout

Logowanie - kod:

- w zależności czy dane logowania są poprawne wyświetlany jest komunikat bądź przekierowanie do odpowiedniego widoku osoby zalogowanej

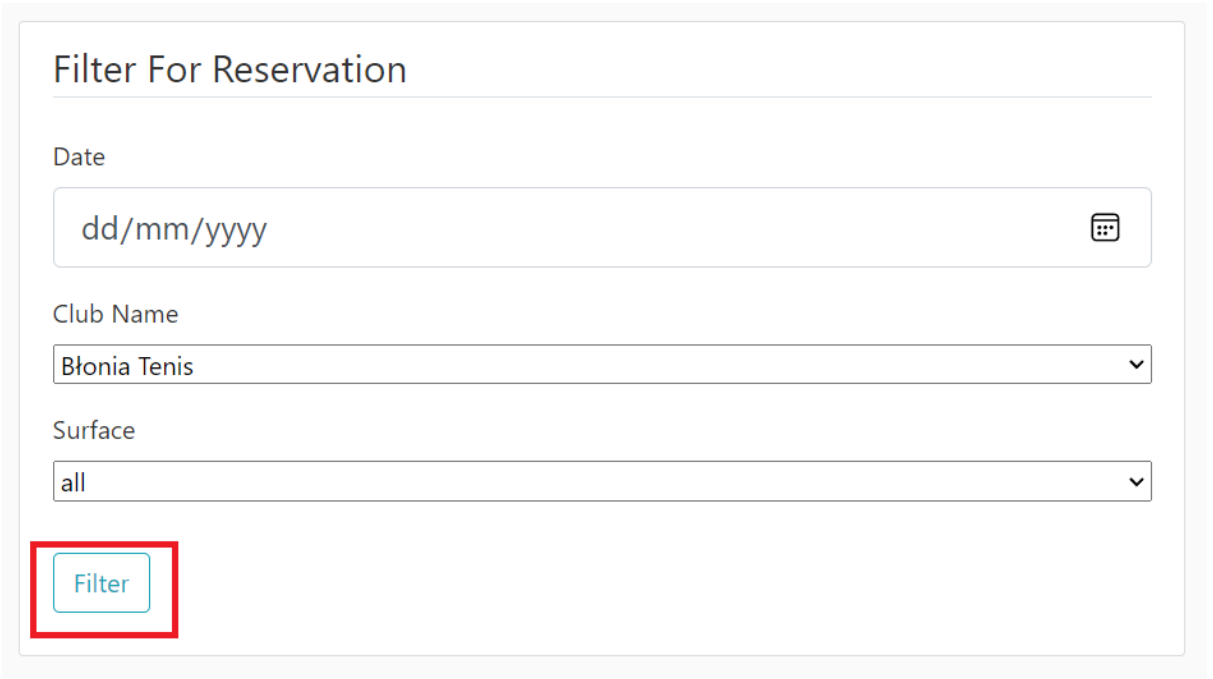
```python
@app.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash('Unsuccessful login. Check username and password', 'danger')

    return render_template('login.html', title='Login', form=form)
```

Filtrowanie rezerwacji - rezerwacja kortu:

- w celu rezerwacji najpierw musimy być zalogowani kiedy jesteśmy na górnym pasku klikamy przycisk reservation

- pokazuje się formularz w którym możemy przefiltrować dane naszej rezerwacji (datę, klub i nawierzchnię) po wybraniu szczegółów klikamy przyciski filter



- wyświetlają nam się szczegóły dostępnych kortów

Court Number	Surface	Roof	Lights	Hour	Status
1	hard	False	True	8	free
1	hard	False	True	9	free
1	hard	False	True	10	free
1	hard	False	True	11	free
1	hard	False	True	12	taken
1	hard	False	True	13	taken
1	hard	False	True	14	free
1	hard	False	True	15	free
1	hard	False	True	16	free
1	hard	False	True	17	free
1	hard	False	True	18	free
1	hard	False	True	19	free
1	hard	False	True	20	free

- w celu rezerwacji przewijamy na formularz który znajduje się na dole strony, uzupełniamy potrzebne informacje i klikamy przycisk reserve

Rezerwacja - kod:

- jeśli filtrujemy z formularza pobierane są dane i wyświetlane konkretne korty które podaliśmy w formularzu - elif('filter')
- jeśli chcemy potwierdzić rezerwację wykonuje się kolejny elif('submit') który sprawdza czy rezerwacja na konkretne dane z formularza jest możliwa i jeśli tak to rezerwuje kort na konkretne godziny

```python
@app.route("/reservation", methods=['GET', 'POST'])
def reservation():
    form = FilterReservations()
    reserve = MakeReservation()
    club_names = get_club_names()
    surface_names = get_surface_names()
    show = False
    if request.method == 'GET':
        reservation_date, club_name, surface_type, reservation.courts_reservations = None, None, None, None
    elif request.form.get('filter'):
        show = True
        reservation_date = form.date.data
        club_name = request.form.get('club_name')
        surface_type = request.form.get('surface_name')
        courts = get_courts_by_surface(club_name, surface_type)
        reservations = get_reservations(reservation_date, courts)
        courts_surface_names = get_courts_surface_names(courts)
        reservation.courts_reservations = tuple(zip(courts, courts_surface_names, reservations))
    elif request.form.get('submit'):
        valid_reservation = True
        show = True
        reservation_date = reserve.date.data
        court_number = reserve.court_number.data
        club_name = request.form.get('club_name')
        court = get_court_by_number(club_name, court_number)
        error_msg = 'Unable to commit reservation: '
        if court:
            reservations = get_reservation(reservation_date, court)
        else:
            valid_reservation = False
```

```python
            reservations = []
            flash(error_msg + 'selected court is invalid', 'danger')
        start_hour, end_hour = 8, 23
        available_hours = [hour not in reservations for hour in range(reserve.hour_from.data, reserve.hour_to.data)]

        if reserve.hour_from.data < start_hour or reserve.hour_to.data > end_hour or \
                reserve.hour_from.data >= reserve.hour_to.data:
            flash(error_msg + 'invalid hours selected', 'danger')
            valid_reservation = False
        elif False in available_hours:
            flash(error_msg + 'reservation for given datetime already made', 'danger')
            valid_reservation = False
        elif reserve.hour_from.data >= reserve.hour_to.data:
            flash(error_msg + 'reservation for given datetime already made', 'danger')
            valid_reservation = False

        if valid_reservation:
            date_from = datetime.combine(reserve.date.data, time(reserve.hour_from.data))
            date_to = datetime.combine(reserve.date.data, time(reserve.hour_to.data))
            if date_from <= datetime.now():
                valid_reservation = False

        if valid_reservation:
            new_reservation = Reservation(user_id=current_user.id, court_id=court.id, date_from=date_from,
                                          date_to=date_to)
            db.session.add(new_reservation)
            db.session.commit()
            flash('You committed reservation succesfully', 'success')
            return redirect(url_for('home'))
```

```python
    return render_template('reservation.html', title='Reservation', courts_reservations=reservation.courts_reservations,
                           form=form, show=show, club_names=club_names, surface_names=surface_names, reserve=reserve)
```

- wszystkie funkcje pomocnicze do pobierania konkretnych wierszy z tabel znajdują się w pliku queries.py - przykładowe funkcje:

- otrzymanie klubu o danej nazwie

```
def get_club_by_name(club_name):
    return Club.query.filter(Club.name == club_name).first()
```

- otrzymanie wszystkich kortów o danej nawierzchni z konkretnego klubu

```
def get_courts_by_surface(club_name, surface_type):
    club_id = Club.query.filter(Club.name == club_name).first().id
    if surface_type == 'all':
        courts = Court.query.filter(Court.club_id == club_id).all()
    else:
        surfaces = Surface.query.filter(Surface.type == surface_type).all()
        if surfaces:
            surface_id = Surface.query.filter(Surface.type == surface_type).first().id
            courts = Court.query.filter(and_(Court.club_id == club_id, Court.surface_id == surface_id)).all()
        else:
            courts = []
    return courts
```

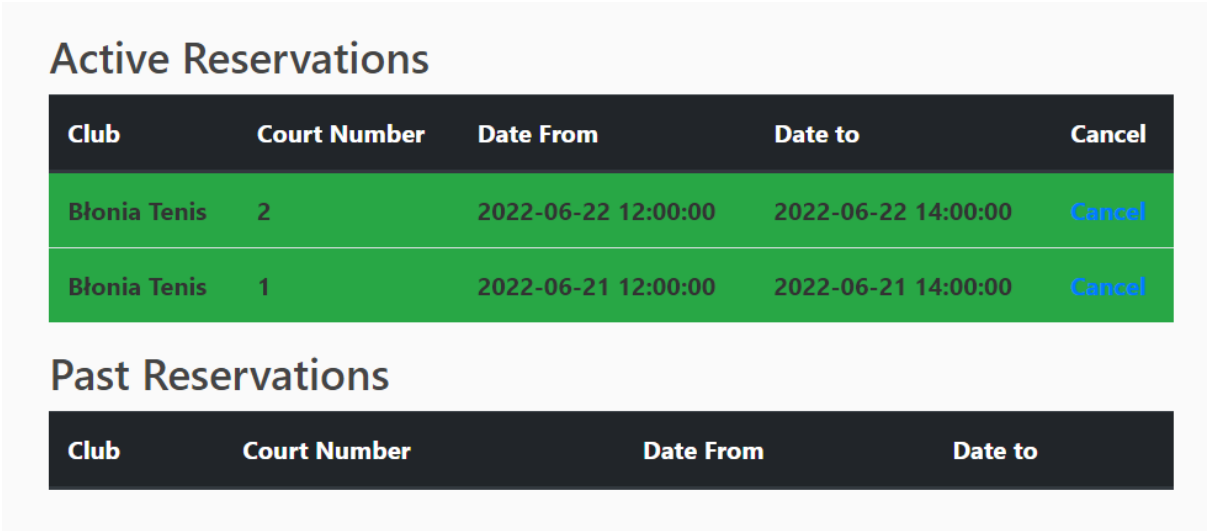
- otrzymanie wszystkich typów nawierzchni

```
def get_surface_names():
    surfaces = Surface.query.all()
    surface_names = [surface.type for surface in surfaces]
    return surface_names
```

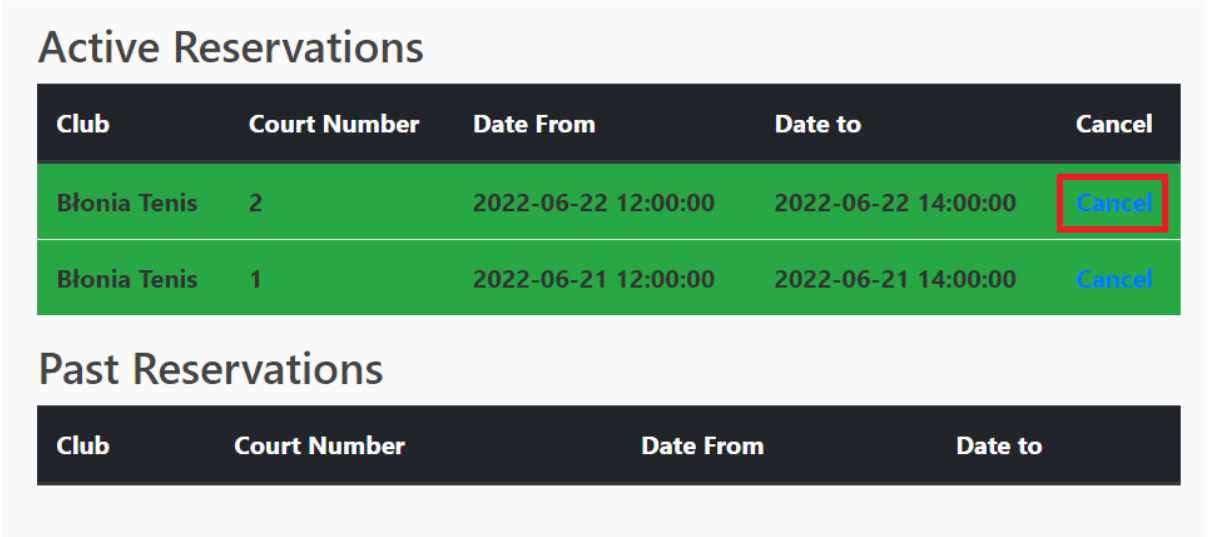
Sprawdzanie historii rezerwacji , aktywnych rezerwacji oraz odwoływanie rezerwacji :

- po zalogowaniu na pasku u góry strony klikamy przycisk “my reservations”

- wyświetlają nam się wszystkie rezerwacje



- w celu odwołania rezerwacji należy przeycisną przycisk cancle przy wybranej rezerwacji



Rezerwacje użytkownika - kod:

- do wyświetlenia rezerwacji używana jest pomocnicza funkcja (get_user_reservations) zwracająca aktywne i przeszłe rezerwacje jako odrębne listy

```python
@app.route("/my_reservations", methods=['GET', 'POST'])
@login_required
def user_reservations():
    active_reservations, past_reservations = get_user_reservations(current_user.id)
    return render_template('user_reservations.html', title='My Reservations',
                           active_reservations=active_reservations, past_reservations=past_reservations)
```

```python
def get_user_reservations(user_id):
    reservations = Reservation.query.filter(Reservation.user_id == user_id).all()
    active = tuple(filter(lambda reservation: reservation.date_to >= datetime.now(), reservations))
    active_reservations = [(reservation, *get_location_info_of_reservation(reservation)) for reservation in active]
    past = tuple(filter(lambda reservation: reservation.date_to < datetime.now(), reservations))
    past_reservations = [(reservation, *get_location_info_of_reservation(reservation)) for reservation in past]
    return active_reservations, past_reservations
```