COURSE PLATFORM

BAZY DANYCH - PROJEKT

Karolina Kucia 410701

Adam Misztal 412814

Szymon Wójcik 411681

Spis treści

Opis aplikacji

Aplikacja ‘COURSE PLATFORM’ jak sama nazwa wskazuje służy do obsługi platformy kursów. Jest możliwość zarezerwować miejsce na wybranym przez nas kursie. Każdy kurs ma swoją kategorie i może mieć wielu mentorów (prowadzących). Każdy użytkownik platformy ma prawo do zrobienia wielu rezerwacji, następnie może zapłacić za wszystkie (dotąd nieopłacone rezerwacje) jednocześnie, albo zapłacić za konkretną rezerwacje. Po zapłacie możliwy jest dostęp do historii opłat w zakładce faktury. Istnieje również możliwość zrezygnowania z danej rezerwacji. Wszystkie zmiany dotyczące statusu rezerwacji są automatycznie zapisywane w historii (tabela log). Przeglądanie danych oraz filtracje poszczególnych informacji wspomagają przygotowane widoki i funkcje.

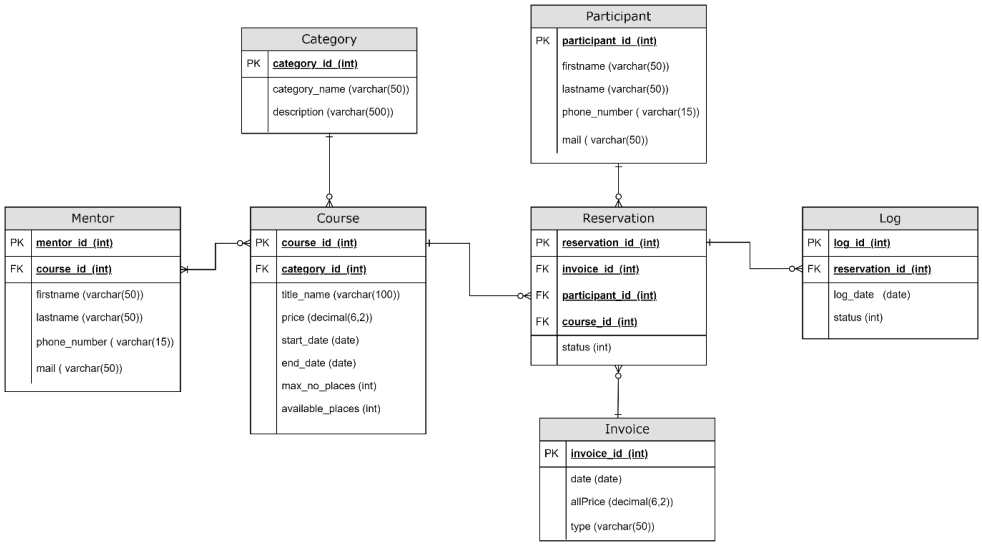
Zostały również dodane funkcjonalności przydatne dla administratora naszej platformy takie jak: możliwość tworzenia kategorii, kursów oraz dodawanie do nich poszczególnych mentorów. Przeglądanie danych, które nie są udostępnione dla zwykłego użytkownika są możliwe dzięki specjalnie stworzonym funkcjom i widokom.

Wszystkie niepoprawne operacje (np. dodanie do kursu nieistniejącego mentora albo zarezerwowanie miejsca na kursie, który już się odbył lub nie ma dostępnych miejsc) są zabezpieczone w bazie. Ponadto wiele procesów zachodzących w bazie jest zautomatyzowane i ‘dzieje się samo’, więc nie trzeba się martwić, że coś nie zostanie zapisane w historii bądź wykonamy niedozwoloną operacje.

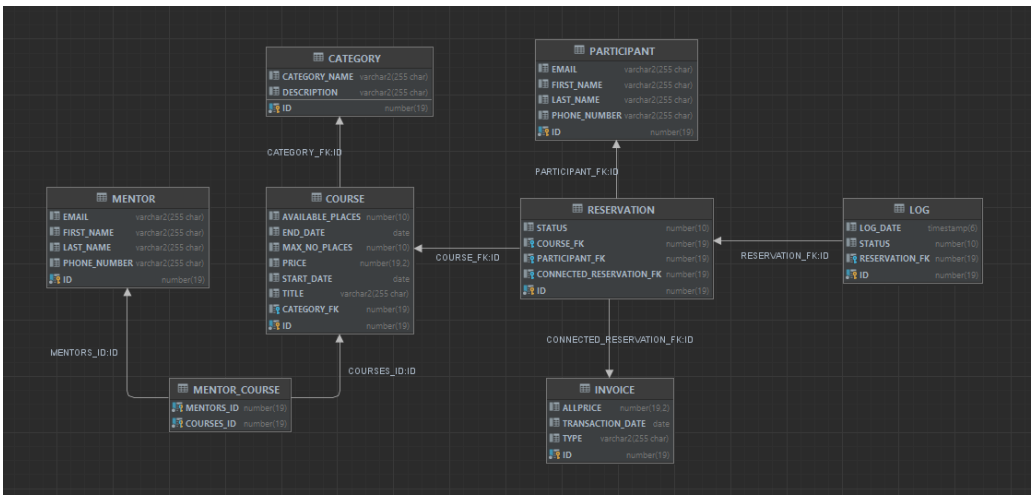
W skrócie: **Aplikacja do zarządzania bazą danych potrzebną do obsługi platformy kursów**

Baza danych (Oracle)

1. Schemat logiczny bazy danych



2. Schemat bazy danych



3. Tabele – tworzenie + ustawienie relacji

Do utworzenia tabel i relacji między nimi w projekcie wykorzystano Hibernate'a. Poniżej kod do poszczególnych klas/tabel.

Poniżej umieszczono tylko kod wskazujący na relacje między tabelami i ich zawartość. Dalsza część implementacji znajduje się w folderze **Backend/src**

Course

```java
@Entity
public class Course implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String title;
    @JsonFormat(shape = JsonFormat.Shape.STRING)
    private BigDecimal price;
    @Column(name = "start_date")
    private Date startDate;
    @Column(name = "end_date")
    private Date endDate;
    @Column(name = "max_no_places")
    private int maxNoPlaces;
    @Column(name = "available_places")
    private int availablePlaces;
    @ManyToOne
    @JoinColumn(name = "CATEGORY_FK")
    private Category category;
    @ManyToMany(mappedBy = "courses")
    private Set<Mentor> mentors;

```

	ID	AVAILABLE_PLACES	END_DATE	MAX_NO_PLACES	PRICE	START_DATE	TITLE	CATEGORY_FK
1	13	30	2024-09-15	30	10000.00	2023-06-15	Java	1
2	14	15	2024-12-01	15	15000.00	2023-09-01	Mobile App Development	1
3	15	10	2024-01-01	10	20000.00	2023-10-01	Artificial Intelligence	1
4	16	5	2024-02-01	5	25000.00	2023-11-01	Blockchain	1
5	17	55	2024-02-11	55	25000.00	2023-11-11	Starożytna grecja	2
6	18	33	2024-11-30	33	25000.00	2024-11-01	Szlakiem polskich królów	2
7	19	22	2024-01-30	22	25000.00	2024-01-01	Wyprawy Kolumba	2
8	20	42	2024-08-11	42	25000.00	2023-09-12	Azja	3
9	21	66	2024-06-12	66	25000.00	2023-10-13	Wyprawy Kolumba	3
10	22	11	2024-05-13	11	25000.00	2023-12-14	Afryka	3

Category

```java
@Entity
public class Category implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    @Column(name = "category_name")
    private String categoryName;
    private String description;

```

	ID	CATEGORY_NAME	DESCRIPTION
1	1	Programowanie	Kurs programowania składa się z teoretycznych wykładów …
2	2	Historia	Uczestnicy poznają kluczowe wydarzenia i postaci z prze…
3	3	Geografia	Uczestnicy poznają różne aspekty geografii, takie jak g…

Reservation

```java
@Entity
public class Reservation implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private long id;
    private ReservationStatus status;

    @ManyToOne
    @JoinColumn(name = "PARTICIPANT_FK")
    private Participant participant;
    @ManyToOne
    @JoinColumn(name = "COURSE_FK")
    private Course course;

```

```java
public enum ReservationStatus {
        PAID,  REJECTED, NEW  }
         //0   //1       //2
```

	ID	STATUS	COURSE_FK	PARTICIPANT_FK	CONNECTED_RESERVATION_FK
1	162	0	13	23	102
2	165	0	13	23	104
3	141	2	17	31	<null>
4	167	2	13	28	<null>
5	168	2	14	28	<null>
6	169	0	13	32	121
7	170	0	14	32	122
8	171	0	14	24	<null>
9	172	0	14	32	123

Invoice

```java
@Entity
public class Invoice implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private long id;
    @Column(name = "transaction_date")
    private Date transactionDate;
    private String type;
    private BigDecimal allPrice;

    @OneToMany
    @JoinColumn(name = "CONNECTED_RESERVATION_FK")
    private Set<Reservation> reservations;

```

	ID	ALLPRICE	TRANSACTION_DATE ▲ 1	TYPE
1	61	12.00	2023-06-09 16:51:35	CARD
2	62	12.00	2023-06-09 17:12:57	CARD
3	81	25012.00	2023-06-09 21:24:41	CARD
4	101	65000.00	2023-06-10 13:05:03	card
5	105	25000.00	2023-06-10 14:59:00	BLIK
6	106	25000.00	2023-06-10 15:08:16	PayPal
7	107	25000.00	2023-06-10 15:08:59	PayPal
8	114	15000.00	2023-06-10 15:20:14	PayPal

Log_table

```java
@Entity(name = "Log_table")
public class LogTable implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private long id;
    @ManyToOne
    @JoinColumn(name = "RESERVATION_FK")
    private Reservation reservation;
    @Column(name = "log_date")
    private Date logDate;
    private ReservationStatus status;

```

	ID	LOG_DATE	STATUS	RESERVATION_FK ▲ 1
1	41	2023-05-27 15:59:58.000000	2	101
2	82	2023-06-09 21:32:56.000000	1	121
3	61	2023-06-09 16:51:35.000000	2	121
4	62	2023-06-09 17:12:57.000000	2	122
5	81	2023-06-09 21:24:41.000000	2	141
6	101	2023-06-10 14:46:51.000000	2	161
7	102	2023-06-10 14:47:02.000000	2	162
8	107	2023-06-10 16:39:29.000000	0	162

User (cechy wspólne Participant oraz Mentor)

```java
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class User implements DbElement {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
    @Column(name = "phone_number")
    private String phoneNumber;
    private String email;

```

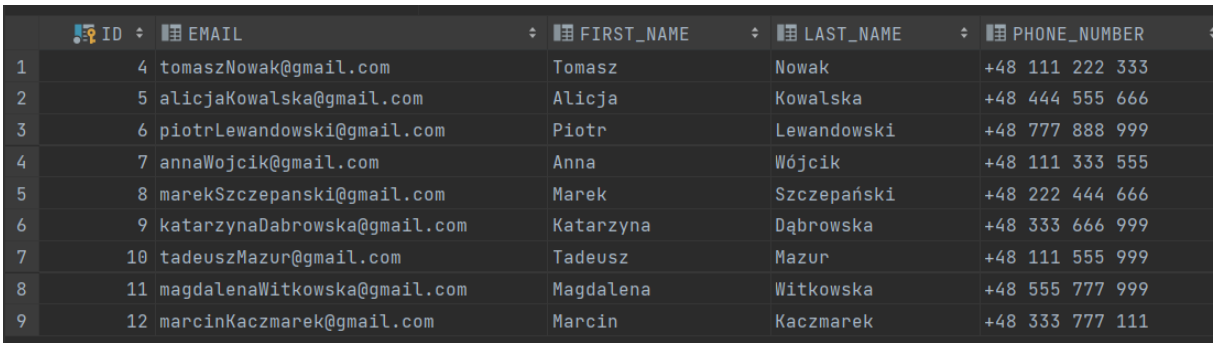
Participant

```java
@Entity
public class Participant extends User implements DbElement

```

	ID	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER
1	23	karolinaWajda@gmail.com	Karolina	Wajda	+48 555 333 111
2	24	jankowalski@gmail.com	Jan	Kowalski	+48 555 111 222
3	25	annanowak@hotmail.com	Anna	Nowak	+48 555 222 333
4	26	tomaszkaminski@yahoo.com	Tomasz	Kaminski	+48 555 444 555
5	27	ewaszymanska@outlook.com	Ewa	Szymanska	+48 555 666 777
6	28	piotrw@gmail.com	Piotr	Wojcik	+48 555 888 999
7	29	iwonakaczmarek@hotmail.com	Iwona	Kaczmarek	+48 555 000 111
8	30	adamlewandowski@gmail.com	Adam	Lewandowski	+48 555 222 333
9	31	magdalenawitkowska@yahoo.com	Magdalena	Witkowska	+48 555 444 555
10	32	krzysztofjaworski@outlook.com	Krzysztof	Jaworski	+48 555 666 777

Mentor

```java
@Entity
public class Mentor extends User implements DbElement {
    @ManyToMany
    private Set<Course> courses;

```



	ID	EMAIL	FIRST_NAME	LAST_NAME	PHONE_NUMBER
1	4	tomaszNowak@gmail.com	Tomasz	Nowak	+48 111 222 333
2	5	alicjaKowalska@gmail.com	Alicja	Kowalska	+48 444 555 666
3	6	piotrLewandowski@gmail.com	Piotr	Lewandowski	+48 777 888 999
4	7	annaWojcik@gmail.com	Anna	Wójcik	+48 111 333 555
5	8	marekSzczepanski@gmail.com	Marek	Szczepański	+48 222 444 666
6	9	katarzynaDabrowska@gmail.com	Katarzyna	Dąbrowska	+48 333 666 999
7	10	tadeuszMazur@gmail.com	Tadeusz	Mazur	+48 111 555 999
8	11	magdalenaWitkowska@gmail.com	Magdalena	Witkowska	+48 555 777 999
9	12	marcinKaczmarek@gmail.com	Marcin	Kaczmarek	+48 333 777 111

Obiekty

course_details

```sql
create or replace type course_details as OBJECT
(
    id               number(19),
    title            varchar2(255 char),
    category_name    varchar2(255 char),
    start_date       date,
    end_date         date,
    available_places number(10),
    max_no_places    number(10),
    price            number(19, 2)
);
```

```sql
create or replace type course_details_table is table of course_details;
```

reservation_details

```sql
create or replace type reservation_details as OBJECT
(
    reservation_id number(19),
    status         number(10),
    price          number(19, 2),
    course_id      number(19),
    course_title   varchar2(255 char),
    first_name     varchar2(255 char),
    last_name      varchar2(255 char)
);
```

```sql
create or replace type reservation_details_table is table of reservation_details;
```

user_details

```sql
create or replace type user_details as OBJECT
(
    id           number(19),
    first_name   varchar2(255 char),
    last_name    varchar2(255 char),
    email        varchar2(255 char),
    phone_number varchar2(255 char)
);
```

```sql
create or replace type user_details_table is table of user_details;
```

invoice_details

```
1. create or replace type invoice_details as OBJECT
2. (id    number(19),
3.     date  date,
4.     price number(19, 2));
```

```
1. create or replace type invoice_details_table is table of invoice_details;
```

Funkcje

f_available_courses_on_time

Funkcja zwracająca listę dostępnych kursów, które zaczynają się i kończą pomiędzy podanymi datami

```
 1. create function f_available_courses_on_time(start_date_course DATE, end_date_course DATE)
 2.     return course_details_table
 3. as
 4.     result course_details_table;
 5. begin
 6.     select course_details(
 7.                     c.id,
 8.                     c.title,
 9.                     ca.category_name,
10.                     c.start_date,
11.                     c.end_date,
12.                     c.available_places,
13.                     c.max_no_places,
14.                     c.price) bulk collect
15.     into result
16.     from course c
17.           inner join category ca on c.CATEGORY_FK = ca.ID
18.     where c.AVAILABLE_PLACES > 0
19.     and c.start_date >= start_date_course
20.     and c.end_date <= end_date_course;
21.
22.     return result;
23. end;
24.
```

f_available_courses_by_category_on_time

Funkcja zwracająca listę dostępnych kursów z danej kategorii, które zaczynają się i kończą pomiędzy podanymi datami

```
 1. create or replace function f_available_courses_by_category_on_time(
 2.     start_date_course date,
 3.     end_date_course date,
 4.     category_id CATEGORY.ID%type)
 5.     return course_details_table
 6. as
 7.     result course_details_table;
 8. begin
 9.     category_exist(category_id);
10.     select course_details(
11.                     c.id,
12.                     c.title,
13.                     ca.category_name,
14.                     c.start_date,
15.                     c.end_date,
16.                     c.available_places,
17.                     c.max_no_places,
18.                     c.price) bulk collect
19.     into result
20.     from course c
21.              inner join category ca on c.CATEGORY_FK = ca.ID
22.     where c.START_DATE >= start_date_course
23.       and c.END_DATE <= end_date_course
24.       and ca.ID = category_id;
25.     return result;
26. end;
```

f_reservations_from_course

Funkcja zwracająca rezerwacje dla danego kursu

```
 2. create or replace function f_reservations_from_course(course_id COURSE.ID%type)
 3.     return reservation_details_table
 4. as
 5.     result reservation_details_table;
 6. begin
 7.     course_exist(course_id);
 8.     select reservation_details(
 9.                     r.id,
10.                     r.status,
11.                     c.price,
12.                     course_id,
13.                     c.title,
14.                     p.first_name,
15.                     p.last_name) bulk collect
16.     into result
17.     from reservation r
18.              inner join course c on c.id = r.COURSE_FK
19.              inner join participant p on p.id = r.PARTICIPANT_FK
20.     where c.ID = course_id;
21.
22.     return result;
23. end;
24.
```

f_reservations_for_participant

Funkcja zwracająca wszystkie rezerwacje dla danego klienta

```
 1. create or replace function f_reservations_for_participant(participant_id PARTICIPANT.ID%type)
 2.     return reservation_details_table
 3. as
 4.     result reservation_details_table;
 5. begin
 6.     participant_exist(participant_id);
 7.     select reservation_details(
 8.                     r.id,
 9.                     r.status,
10.                     c.price,
11.                     c.id,
12.                     c.title,
13.                     p.first_name,
14.                     p.last_name) bulk collect
15.     into result
16.     from reservation r
17.              inner join course c on c.id = r.COURSE_FK
18.              inner join participant p on p.id = r.PARTICIPANT_FK
19.     where p.id = participant_id;
20.
21.     return result;
22. end;
```

f_unpaid_reservations_for_participant

Funkcja zwracająca nieopłacone rezerwacje dla danego klienta

```
create or replace function f_unpaid_reservations_for_participant(participant_id PARTICIPANT.ID%type)
    return reservation_details_table
as
    result reservation_details_table;
begin
    participant_exist(participant_id);
    select reservation_details(
                    r.id,
                    r.status,
                    c.price,
                    c.id,
                    c.title,
                    p.first_name,
                    p.last_name) bulk collect
    into result
    from reservation r
             inner join course c on c.id = r.COURSE_FK
             inner join participant p on p.id = r.PARTICIPANT_FK
    where p.id = participant_id
      and r.status = 2; --NEW

    return result;
end;

```

f_participants_from_course

Funkcja zwracająca klientów dla danego kursu

```
create or replace function f_participants_from_course(course_id COURSE.ID%type)
    return user_details_table
as
    result user_details_table;
begin
    course_exist(course_id);
    select user_details(
                    p.id,
                    p.first_name,
                    p.last_name,
                    p.email,
                    p.phone_number) bulk collect
    into result
    from participant p
             inner join reservation r on p.id = r.PARTICIPANT_FK
             inner join course c on c.id = r.COURSE_FK
    where c.ID = course_id;

    return result;
end;

```

f_mentors_from_course

Funkcja zwracająca mentorów i kontakt do nich dla danego kursu

```
create or replace function f_mentors_from_course(course_id COURSE.ID%type)
    return user_details_table
as
    result user_details_table;
begin
    course_exist(course_id);
    select user_details(
                    m.id,
                    m.first_name,
                    m.last_name,
                    m.email,
                    m.phone_number) bulk collect
    into result
    from mentor m
             inner join MENTOR_COURSE mc on mc.MENTORS_ID = m.ID
             inner join course c on mc.COURSES_ID = c.id
    where c.ID = course_id;

    return result;
end;

```

f_invoices_for_participant

Funkcja zwracająca faktury dla danego klienta

```
create or replace function f_invoices_for_participant(participant_id PARTICIPANT.ID%type)
    return invoice_details_table
as
    result invoice_details_table;
begin
    participant_exist(participant_id);
    select invoice_details(
                    i.id,
                    i.transaction_date,
                    i.allPrice) bulk collect
    into result
    from invoice i
             inner join reservation r on i.id = r.CONNECTED_RESERVATION_FK
             inner join participant p on p.id = r.PARTICIPANT_FK
    where p.id = participant_id;

    return result;
end;

```

f_amunt_to_pay_for_participant

Funkcja zwraca sume nieopłaconych rezerwacji dla danego klienta

```
create or replace function f_amount_to_pay_for_participant(participant_id PARTICIPANT.ID%type)
    return INVOICE.ALLPRICE%TYPE
as
    total_amount INVOICE.ALLPRICE%TYPE := 0;
begin
    participant_exist
(participant_id);

select sum(c.price)
into total_amount
from reservation r
         inner join course c on c.id = r.COURSE_FK
where r.PARTICIPANT_FK = participant_id
  and r.status = 2 --NEW
group by r.PARTICIPANT_FK;
return total_amount;
end;
```

f_update_available_places

Funkcja potrzebna triggerowi *tr_modify_no_places* do zmainy available_places

```
create or replace function f_update_available_places(new_max_places COURSE.MAX_NO_PLACES%type,
                                                     old_max_places COURSE.MAX_NO_PLACES%type,
                                                     p_old_available_places COURSE.AVAILABLE_PLACES%type)
    return COURSE.AVAILABLE_PLACES%type
    is
    new_available_places COURSE.AVAILABLE_PLACES%type;
begin
    if new_max_places < 0 then
        raise_application_error(-20001, 'ERROR: Number of places cannot be less than zero.');
    end if;

    new_available_places := p_old_available_places + (new_max_places - old_max_places);

    if new_available_places < 0 then
        raise_application_error(-20001, 'ERROR: Number of available places cannot be less than zero.');
    end if;

    return new_available_places;
end;

```

Procedury

add_log_reservation

Procedura zapisująca log przy zmianie statusu danej rezerwacji

```
create PROCEDURE add_log_reservation(new_status LOG_TABLE.STATUS%type,
    reservation_id LOG_TABLE.RESERVATION_FK%type)
AS
    log_id LOG_TABLE.ID%type;
BEGIN

    SELECT LOG_SEQ.NEXTVAL INTO log_id FROM dual;

    insert into LOG_TABLE (ID, log_date, status, reservation_fk)
    values (log_id, current_date, new_status, reservation_id);
    commit;
exception
    when others then
        rollback;
        raise;
end;

```

make_reservation

Procedura tworząca rezerwacje dla danej osoby na konkretny kurs.

```
create procedure make_reservation(
    course_id course.id%type,
    participant_id participant.id%type
)
as
    reservation_id reservation.id%type;
begin
    course_exist(course_id);
    participant_exist(participant_id);

    SELECT reservation_seq.NEXTVAL INTO reservation_id FROM dual;

    insert into reservation (ID, STATUS, COURSE_FK, PARTICIPANT_FK)
    values (reservation_id, 2, course_id, participant_id);

    ADD_LOG_RESERVATION(2, reservation_id);

    commit;
exception
    when others then
        rollback;
        raise;
end;

```

pay_for_reservation

Procedura płacąca za konkretną rezerwacje (tworzy fakture dla pojedyńczej rezerwacji)

```
create PROCEDURE pay_for_reservation(reservation_id IN RESERVATION.ID%TYPE, payment_type INVOICE.TYPE%type)
AS
    course_price       COURSE.PRICE%TYPE;
    invoice_id         INVOICE.ID%TYPE;
    reservation_status RESERVATION.STATUS%TYPE;
BEGIN
    reservation_exist(reservation_id);

    SELECT c.PRICE, r.STATUS
    INTO course_price, reservation_status
    FROM RESERVATION r
             INNER JOIN COURSE c on c.ID = r.COURSE_FK
    WHERE r.ID = reservation_id;

    if reservation_status != 2 then
        raise_application_error(-20001, 'ERROR: This reservation was already paid.');
    end if;

    SELECT invoice_seq.NEXTVAL INTO invoice_id FROM dual;

    INSERT INTO INVOICE (ID, ALLPRICE, TRANSACTION_DATE, "TYPE")
    VALUES (invoice_id, course_price, current_date, payment_type);

    update RESERVATION r
    set r.STATUS = 0, --PAID
     r.CONNECTED_RESERVATION_FK = invoice_id
    where reservation_id = r.ID;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        raise;
END;

```

make_main_invoice

Procedura tworząca główną fakturę – podsumowanie paragonu (Gdy płacimy za więcej niż jeden kurs, tworzy zapis w taeli invoice z podsumowaniem płatności za całość – działa jak suma na paragonie)

```
 1. create PROCEDURE make_main_invoice(
 2.     invoice_price INVOICE.ALLPRICE%TYPE,
 3.     payment_type INVOICE.TYPE%type)
 4. AS
 5.     invoice_id         INVOICE.ID%TYPE;
 6. BEGIN
 7.
 8.     SELECT invoice_seq.NEXTVAL INTO invoice_id FROM dual;
 9.
10.     INSERT INTO INVOICE (ID, ALLPRICE, TRANSACTION_DATE, "TYPE")
11.     VALUES (invoice_id, invoice_price, current_date, payment_type);
12.
13.     COMMIT;
14. EXCEPTION
15.     WHEN OTHERS THEN
16.         ROLLBACK;
17.         raise;
18. END;
19.
```

pay_for_all_unpaid_reservations

Procedura płacąca za wszystkie niezapłacone rezerwacje dla danego klienta

```
 1. create PROCEDURE pay_for_all_unpaid_reservations(participant_id IN PARTICIPANT.ID%TYPE, payment_type
INVOICE.TYPE%TYPE)
 2. AS
 3.   reservation_id RESERVATION.ID%TYPE;
 4.   invoice_price  INVOICE.ALLPRICE%TYPE;
 5. BEGIN
 6.     invoice_price := f_amount_to_pay_for_participant(participant_id);
 7.
 8.   FOR rec IN (SELECT r.ID
 9.               FROM RESERVATION r
10.               INNER JOIN COURSE c ON c.ID = r.COURSE_FK
11.               WHERE r.STATUS = 2
12.               AND r.PARTICIPANT_FK = participant_id)
13.   LOOP
14.     reservation_id := rec.ID;
15.     pay_for_reservation(reservation_id, payment_type);
16.   END LOOP;
17.
18.   COMMIT;
19.     make_main_invoice(invoice_price, payment_type );
20.     COMMIT;
21.
22. EXCEPTION
23.   WHEN OTHERS THEN
24.     ROLLBACK;
25.     RAISE;
26. END;
27.
```

cancel_reservation

Procedura anulująca daną rezerwacje

```
create or replace PROCEDURE cancel_reservation(reservation_id IN RESERVATION.ID%TYPE)
AS
    reservation_status RESERVATION.STATUS%TYPE;
BEGIN
    reservation_exist(reservation_id);

    SELECT r.STATUS
    INTO reservation_status
    FROM RESERVATION r
    WHERE r.ID = reservation_id;

    if reservation_status != 2 then
        raise_application_error(-20001, 'ERROR: This reservation was already paid or canceled.');
    end if;

    update RESERVATION r
    set r.STATUS = 1 --REJECTED
    where reservation_id = r.ID;

    ADD_LOG_RESERVATION(1, reservation_id);

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        raise;
END;

```

add_mentor_to_course

Procedura dodająca mentora do kursu, który prowadzi

```

create or replace procedure add_mentor_to_course(
    add_course_id course.id%type,
    add_mentor_id mentor.id%type
)
as
begin
    course_exist(add_course_id);
    mentor_exist(add_mentor_id);

    insert into MENTOR_COURSE(mentors_id, courses_id)
    values (add_mentor_id, add_course_id);
    commit;
exception
    when others then
        rollback;
        raise;
end;

```

reservation_exist

Procedura kontrolująca istnienie danej rezerwacji

```
create or replace procedure reservation_exist(r_ID RESERVATION.ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from RESERVATION r where r.ID = r_ID;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'ERROR: Reservation not found');
end;

```

participant_exist

Procedura kontrolująca istnienie danego użytkownika

```
create or replace procedure participant_exist(p_ID PARTICIPANT.ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from PARTICIPANT p where p.ID = p_ID;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'ERROR: Participant not found');
end;

```

mentor_exist

Procedura kontrolująca istnienie danego mentora

```
create or replace procedure mentor_exist(m_ID MENTOR.ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from MENTOR m where m.ID = m_ID;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'ERROR: Mentor not found');
end;

```

course_exist

Procedura kontrolująca istnienie danego kursu

```
create or replace procedure course_exist(c_ID COURSE.ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from COURSE c where c.ID = c_ID;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'ERROR: Course not found');
end;

```

category_exist

Procedura kontrolująca istnienie danej kategorii

```
create or replace procedure category_exist(c_ID CATEGORY.ID%type)
as
    tmp char(1);
begin
    select 1 into tmp from CATEGORY c where c.ID = c_ID;
exception
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'ERROR: Category not found');
end;

```

Triggery

tr_forbidden_remove_reservation

Trigger zabraniający usuwania rezerwacji. Zabezpiecza przed usunięciem danej rezerwacji. (Rezerwacjie mogą zmieniać status, ale nie mogą zostać usuwane)

```
1. create or replace trigger tr_forbidden_remove_reservation
2.     before delete
3.     on RESERVATION
4.     for each row
5. begin
6.     raise_application_error(-20001, 'ERROR: Removing reservations is forbidden.');
7. end;
8.
```

tr_forbidden_make_reservation

Trigger wyłapuje dodawanie nowej rezerwacji i sprawdza warunki, ponieważ zabronione jest robienie rezerwacji na nieistniejący kurs, przez nieistniejącego klienta i robienie rezerwacji w przeszłości

```
 1. create or replace trigger tr_forbidden_make_reservation
 2.     before insert
 3.     on RESERVATION
 4.     for each row
 5. declare
 6.     start_date_course COURSE.start_date%type;
 7. begin
 8.     course_exist(:new.COURSE_FK);
 9.     participant_exist(:new.PARTICIPANT_FK);
10.
11.     select c.start_date
12.     into start_date_course
13.     from COURSE c
14.     where c.ID = :new.COURSE_FK;
15.
16.     if start_date_course < current_date then
17.         raise_application_error(-20001, 'ERROR: Reservation for this course is not available.');
18.     end if;
19. end;
20.
```

tr_modify_no_places

Trigger wyłpujący zmiane maksymalnej liczby miejsc na danym kursie i aktualizujący dostępną liczbe miejsc.

```
 1. create or replace trigger tr_modify_no_places
 2.     before update of max_no_places
 3.     on COURSE
 4.     for each row
 5. declare
 6.     new_available_places COURSE.AVAILABLE_PLACES%type;
 7. begin
 8.     new_available_places := f_update_available_places(:new.max_no_places, :old.max_no_places,
:old.AVAILABLE_PLACES);
 9.
10.     :new.AVAILABLE_PLACES := new_available_places;
11. end;
12.
```

tr_change_places_reservation

Trigger wyłapujący zmiane statusu rezerwacji i zmienia liczbę dostępnych miejsc na kursie miejsc na kursie

```
create or replace trigger TR_CHANGE_PLACES_RESERVATION
    before insert or update of STATUS
    on RESERVATION
    for each row
declare
    available_places_course COURSE.AVAILABLE_PLACES%type;
    places                  COURSE.AVAILABLE_PLACES%type;

begin
    select c.AVAILABLE_PLACES
    into available_places_course
    from COURSE c
    where c.ID = :new.COURSE_FK;

    if available_places_course <= 0 and :new.status = 2 then
        raise_application_error(-20001, 'ERROR: Course is fully booked');
    end if;

    if :new.status = 1 --REJECTED
    then
        places := 1;
    elsif :new.status = 2 --NEW
    then
        places := -1;
    else --PAID
        places := 0;
    end if;

    update COURSE c
    set c.AVAILABLE_PLACES = available_places_course + places
    where :new.COURSE_FK = c.ID;
end;
```

2 Widoki

available_courses_view

Widok dostępnych kursów

```
create or replace view available_courses_view
as
select c.ID,
       c.TITLE,
       ca.CATEGORY_NAME,
       c.START_DATE,
       c.END_DATE,
       c.AVAILABLE_PLACES,
       c.MAX_NO_PLACES,
       c.PRICE
from course c
         inner join category ca on c.CATEGORY_FK = ca.ID
where c.AVAILABLE_PLACES > 0
  and c.START_DATE > current_date;
```

	ID	TITLE	CATEGORY_NAME	START_DATE	END_DATE	AVAILABLE_PLACES
1	63	Test course3	Programowanie	2023-10-25 02:00:00	2023-10-30 01:00:00	10
2	14	Mobile App Development	Programowanie	2023-09-01	2024-12-01	15
3	15	Artificial Intelligence	Programowanie	2023-10-01	2024-01-01	10
4	16	Blockchain	Programowanie	2023-11-01	2024-02-01	5
5	13	Java	Programowanie	2023-06-15	2024-09-15	30
6	85	Python	Programowanie	2023-06-22 02:00:00	2023-09-01 02:00:00	100
7	19	Wyprawy Kolumba	Historia	2024-01-01	2024-01-30	22
8	17	Starożytna grecja	Historia	2023-11-11	2024-02-11	55
9	18	Szlakiem polskich królów	Historia	2024-11-01	2024-11-30	33
10	22	Afryka	Geografia	2023-12-14	2024-05-13	11
11	20	Azja	Geografia	2023-09-12	2024-08-11	42
12	21	Wyprawy Kolumba	Geografia	2023-10-13	2024-06-12	66

courses_view

Widok wszystkich kursów

```
 1. create or replace view courses_view
 2. as
 3. select c.ID,
 4.        c.TITLE,
 5.        ca.CATEGORY_NAME,
 6.        c.START_DATE,
 7.        c.END_DATE,
 8.        c.AVAILABLE_PLACES,
 9.        c.MAX_NO_PLACES,
10.        c.PRICE
11. from course c
12.          inner join category ca on c.CATEGORY_FK = ca.ID;
13.
```

	ID	TITLE	CATEGORY_NAME	START_DATE 1	AVAILAB...	END_DATE	MAX_N...	PRICE
1	18	Szlakiem p…	Historia	2024-11-01	33	2024-11-30	33	25000.00
2	19	Wyprawy Ko…	Historia	2024-01-01	22	2024-01-30	22	25000.00
3	22	Afryka	Geografia	2023-12-14	11	2024-05-13	11	25000.00
4	41	Test course	Programowanie	2023-11-30 01:00:00	0	2023-12-30 0…	0	3.50
5	42	Test cours…	Historia	2023-11-30 01:00:00	0	2023-12-30 0…	0	3.50
6	17	Starożytna…	Historia	2023-11-11	55	2024-02-11	55	25000.00
7	16	Blockchain	Programowanie	2023-11-01	5	2024-02-01	5	25000.00
8	63	Test cours…	Programowanie	2023-10-25 02:00:00	10	2023-10-30 0…	10	5.53
9	21	Wyprawy Ko…	Geografia	2023-10-13	66	2024-06-12	66	25000.00
10	15	Artificial…	Programowanie	2023-10-01	10	2024-01-01	10	20000.00
11	20	Azja	Geografia	2023-09-12	42	2024-08-11	42	25000.00
12	14	Mobile App…	Programowanie	2023-09-01	15	2024-12-01	15	15000.00
13	85	Python	Programowanie	2023-06-22 02:00:00	100	2023-09-01 0…	100	1230.00

canceled_reservation_view

Widok anulowanych rezerwacji

```
 1. create or replace view canceled_reservation_view
 2. as
 3. select r.ID,
 4.        c.TITLE,
 5.        c.PRICE,
 6.        r.STATUS,
 7.        p.FIRST_NAME,
 8.        p.LAST_NAME
 9. from reservation r
10.          inner join participant p on r.PARTICIPANT_FK = p.ID
11.          inner join course c on r.COURSE_FK = c.ID
12. where STATUS = 1;
13.
```

	ID	TITLE	PRICE	STATUS	FIRST_NAME	LAST_NAME
1	166	Java	10000.00	1	Karolina	Wajda
2	164	Mobile App Development	15000.00	1	Karolina	Wajda
3	121	C++ Programming	12.00	1	Adam	Lewandowski

reservation_view

Widok wszystkich rezerwacji

```sql
create or replace view reservation_view
as
select r.ID,
       c.TITLE,
       c.PRICE,
       r.STATUS,
       p.FIRST_NAME,
       p.LAST_NAME
from reservation r
         inner join participant p on r.PARTICIPANT_FK = p.ID
         inner join course c on r.COURSE_FK = c.ID;

```

	ID	TITLE	PRICE	STATUS 1	FIRST_NAME	LAST_NAME
1	161	Java	10000.00	2	Karolina	Wajda
2	122	C++ Programming	12.00	2	Magdalena	Witkowska
3	141	Starożytna grecja	25000.00	2	Magdalena	Witkowska
4	168	Mobile App Development	15000.00	2	Piotr	Wojcik
5	167	Java	10000.00	2	Piotr	Wojcik
6	166	Java	10000.00	1	Karolina	Wajda
7	121	C++ Programming	12.00	1	Adam	Lewandowski
8	164	Mobile App Development	15000.00	1	Karolina	Wajda
9	163	Java	10000.00	0	Karolina	Wajda
10	165	Java	10000.00	0	Karolina	Wajda
11	162	Java	10000.00	0	Karolina	Wajda
12	101	Java	10000.00	0	Jan	Kowalski

invoices_view

Widok wszystkich faktur

```sql
create or replace view invoices_view
as
select i.ID,
       i.ALLPRICE,
       i.TRANSACTION_DATE,
       p.FIRST_NAME,
       p.LAST_NAME
from invoice i
         inner join RESERVATION r on i.ID = r.CONNECTED_RESERVATION_FK
         inner join PARTICIPANT P on P.ID = r.PARTICIPANT_FK;

```

WHERE ORDER BY

	ID	ALLPRICE	TRANSACTION_DATE	FIRST_NAME	LAST_NAME
1	62	12.00	2023-06-09 17:12:57	Magdalena	Witkowska
2	102	10000.00	2023-06-10 16:39:29	Karolina	Wajda
3	104	10000.00	2023-06-10 16:51:38	Karolina	Wajda
4	117	10000.00	2023-06-10 15:26:25	Karolina	Wajda
5	121	10000.00	2023-06-10 15:37:42	Krzysztof	Jaworski
6	122	15000.00	2023-06-10 15:37:42	Krzysztof	Jaworski
7	123	15000.00	2023-06-10 15:49:07	Krzysztof	Jaworski
8	124	15000.00	2023-06-10 15:49:07	Krzysztof	Jaworski

categories_view

Widok wszystkich kategorii

```
1. create or replace view categories_view
2. as
3. select c.ID,
4.        c.CATEGORY_NAME,
5.        c.DESCRIPTION
6. from category c;
7.
```

	ID	CATEGORY_NAME	DESCRIPTION
1	1	Programowanie	Kurs programowania składa się z teoretycznyc…
2	2	Historia	Uczestnicy poznają kluczowe wydarzenia i pos…
3	3	Geografia	Uczestnicy poznają różne aspekty geografii, …

participant_view

Widok wszystkich użytkowników (uczestników kursów)

```
1. create or replace view participant_view
2. as
3. select p.ID,
4.        p.FIRST_NAME,
5.        p.LAST_NAME,
6.        p.PHONE_NUMBER,
7.        p.EMAIL
8. from participant p;
```

	ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL
1	23	Karolina	Wajda	+48 555 333 111	karolinaWajda@gmail.com
2	24	Jan	Kowalski	+48 555 111 222	jankowalski@gmail.com
3	25	Anna	Nowak	+48 555 222 333	annanowak@hotmail.com
4	26	Tomasz	Kaminski	+48 555 444 555	tomaszkaminski@yahoo.com
5	27	Ewa	Szymanska	+48 555 666 777	ewaszymanska@outlook.com
6	28	Piotr	Wojcik	+48 555 888 999	piotrw@gmail.com
7	29	Iwona	Kaczmarek	+48 555 000 111	iwonakaczmarek@hotmail.com
8	30	Adam	Lewandowski	+48 555 222 333	adamlewandowski@gmail.com
9	31	Magdalena	Witkowska	+48 555 444 555	magdalenawitkowska@yahoo.com
10	32	Krzysztof	Jaworski	+48 555 666 777	krzysztofjaworski@outlook.com

Server (Java + Spring)

Aby połączyć Jave Spring z Reactem przy użyciu endpointów, można użyto komunikacji klient-serwer.

Aplikacja używa mappingów typu GET do pobierania danych z bazy i POST w przypadku dodawania nowych.

Spring boot jest odpowiedzialny, za odpowiednie działanie entpointów. W przypadku mappingu get java pobiera dane z bazy przy użyciu hibernate, a następnie udostępnia je pod wskazanym endpointem. Natomiast te które używają postmappingu przyjmują dane, które dodają do bazy i zwracają odpowiedź czy wszystko się powiodło.

1. Endpointy

W sprawozdaniu zostały wklejone tylko niektóre wyjścia endpointów, ale wszystkie działają i są używane.

MainController

```java
public abstract class MainController {
    protected final Session session;
    protected final Gson gson;

    public MainController() {
        this.session = DBConnection.getSession();
        this.gson = new Gson();
    }
}
```

```java
    @CrossOrigin
    @GetMapping("/")
    public String mainPageMessage() {
        return "Welcome";
    }
```

/courses

```java
    @CrossOrigin
    @GetMapping("/courses")
    public List<Course> getCourses() {
        Object[] queryResults = session.createQuery("from Course").stream().toArray();
        List<Course> courses = new ArrayList<>();
        Arrays.stream(queryResults).forEach(queryResult -> courses.add((Course) queryResult));
        return courses;}
```

```java
    @CrossOrigin
    @PostMapping("/courses")
    public ResponseEntity<HttpStatus> addCourse(@RequestBody Course course) {
        Transaction tx = session.beginTransaction();
        session.save(course);
        tx.commit();
        return ResponseEntity.ok(HttpStatus.OK);}
```


[{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]},{"id":14,"title":"Mobile App Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]},{"id":15,"title":"Artificial Intelligence","price":"20000","startDate":"2023-10-01","endDate":"2024-01-01","maxNoPlaces":10,"availablePlaces":10,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]},{"id":16,"title":"Blockchain","price":"25000","startDate":"2023-11-01","endDate":"2024-02-01","maxNoPlaces":5,"availablePlaces":5,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]},{"id":17,"title":"Starożytna grecja","price":"25000","startDate":"2023-11-11","endDate":"2024-02-11","maxNoPlaces":55,"availablePlaces":55,"category":{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":[]},{"id":18,"title":"Szlakiem polskich królów","price":"25000","startDate":"2024-11-01","endDate":"2024-11-30","maxNoPlaces":33,"availablePlaces":33,"category":{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":[]},{"id":19,"title":"Wyprawy Kolumba","price":"25000","startDate":"2024-01-01","endDate":"2024-01-30","maxNoPlaces":22,"availablePlaces":22,"category":{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":[]},{"id":20,"title":"Azja","price":"25000","startDate":"2023-09-12","endDate":"2024-08-11","maxNoPlaces":42,"availablePlaces":42,"category":{"id":3,"categoryName":"Geografia","description":"Uczestnicy poznają różne aspekty geografii, takie jak geografia fizyczna, polityczna czy ekonomiczna. Nauczą się również korzystać z narzędzi geograficznych i technologii, takich jak systemy informacji geograficznej."},"mentors":[]},{"id":21,"title":"Wyprawy Kolumba","price":"25000","startDate":"2023-10-13","endDate":"2024-06-12","maxNoPlaces":66,"availablePlaces":66,"category":{"id":3,"categoryName":"Geografia","description":"Uczestnicy poznają różne aspekty geografii, takie jak geografia fizyczna, polityczna czy ekonomiczna. Nauczą się również korzystać z narzędzi geograficznych i technologii, takich jak systemy informacji geograficznej."},"mentors":[]},{"id":22,"title":"Afryka","price":"25000","startDate":"2023-12-14","endDate":"2024-05-13","maxNoPlaces":11,"availablePlaces":11,"category":{"id":3,"categoryName":"Geografia","description":"Uczestnicy poznają różne aspekty geografii, takie jak geografia fizyczna, polityczna czy ekonomiczna. Nauczą się również korzystać z narzędzi geograficznych i technologii, takich jak systemy informacji geograficznej."},"mentors":[]},{"id":41,"title":"Test course","price":"3.5","startDate":"2023-11-30","endDate":"2023-12-30","maxNoPlaces":0,"availablePlaces":0,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]},{"id":42,"title":"Test course2","price":"3.5","startDate":"2023-11-30","endDate":"2023-12-30","maxNoPlaces":0,"availablePlaces":0,"category":{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":[]},{"id":43,"title":"Test in programming","price":"120","startDate":"2023-06-14","endDate":"2023-06-22","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":44,"title":"C++ Programming","price":"12","startDate":"2023-06-13","endDate":"2023-06-29","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":45,"title":"Unit Testing","price":"20","startDate":"2023-06-13","endDate":"2023-06-22","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":46,"title":"Unit Testing","price":"20","startDate":"2023-06-13","endDate":"2023-06-22","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":47,"title":"Drawing course","price":"20","startDate":"2023-06-07","endDate":"2023-06-22","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":48,"title":"Drawing course","price":"20","startDate":"2023-06-07","endDate":"2023-06-22","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":49,"title":"Drawing course","price":"20","startDate":"2023-06-07","endDate":"2023-06-01","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":61,"title":"Test course2","price":"3.5","startDate":"2023-11-30","endDate":"2023-12-30","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":[]},{"id":62,"title":"Test course2","price":"3.5","startDate":"2023-11-30","endDate":"2023-12-30","maxNoPlaces":30,"availablePlaces":30,"category":null,"mentors":[{"id":9,"firstName":"Katarzyna","lastName":"Dąbrowska","phoneNumber":"+48 333 666 999","email":"katarzynaDabrowska@gmail.com"}]},{"id":63,"title":"Test course3","price":"5.53","startDate":"2023-10-25","endDate":"2023-10-30","maxNoPlaces":10,"availablePlaces":10,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]},{"id":85,"title":"Python","price":"1230","startDate":"2023-06-22","endDate":"2023-09-01","maxNoPlaces":100,"availablePlaces":100,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]}]

/courses/available

```
@RestController
public class CoursesController extends MainController {

    @CrossOrigin
    @GetMapping("/courses/available")
    public String getAvailableCourses() {
        Query query = session.createSQLQuery("SELECT * FROM AVAILABLE_COURSES_VIEW");
        return returnPreparedAvailableCourses(query);
    }

```

/courses/available/between

```
    @CrossOrigin
    @GetMapping("/courses/available/between")
    public String getAvailableCoursesBetweenDates(@RequestParam Map<String, Date> allParams) {
        Query query = session.createSQLQuery("SELECT * FROM
F_AVAILABLE_COURSES_ON_TIME(:startDate,:endDate)")
                .setParameter("startDate", allParams.get("startDate"))
                .setParameter("endDate", allParams.get("endDate"));
        return returnPreparedAvailableCourses(query);
    }

```

/courses/categories/available/between

```
    @CrossOrigin
    @GetMapping("/courses/categories/available/between")
    public String getAvailableCoursesBetweenDatesByCategory(@RequestParam Map<String, String> allParams) {
        Query query = session.createSQLQuery("SELECT * FROM
f_available_courses_by_category_on_time(:startDate,:endDate, :category_id)")
                .setParameter("startDate", Date.valueOf(allParams.get("startDate")))
                .setParameter("endDate", Date.valueOf(allParams.get("endDate")))
                .setParameter("category_id", Long.parseLong(allParams.get("categoryId")));
        return returnPreparedAvailableCourses(query);
    }

    private String returnPreparedAvailableCourses(Query query) {
        List<AvailableCourse> availableCourses = new ArrayList<>();
        Object[] currObj;
        AvailableCourse availableCourse;
        try {
            for (Object result : query.getResultList()) {
                currObj = (Object[]) result;
                availableCourse = new AvailableCourse(currObj);
                availableCourses.add(availableCourse);
            }
        } catch (PersistenceException ignored) {
        }
        return gson.toJson(availableCourses);
    }

```

/courses/{courseName}

```
    @CrossOrigin
    @GetMapping("/courses/{courseName}")
    public Course getCourseByName(@PathVariable("courseName") String courseName) {
        Optional queryResult = session.createQuery("from Course c where c.title=:courseName")
                .setParameter("courseName", courseName).stream().findFirst();
        if (queryResult.isPresent()) {
            return (Course) queryResult.get();
        }
        return null;
    }

```

/courses/participants

```
1.     @CrossOrigin
2.     @GetMapping("/courses/participants")
3.     public String getParticipantsForCourse(@RequestParam("courseId") long courseId) {
4.         Query query = session.createSQLQuery("SELECT * FROM f_participants_from_course(:courseId)")
5.                 .setParameter("courseId", courseId);
6.         List<Participant> participants = new ArrayList<>();
7.         Object[] currObj;
8.         Participant participant;
9.         try {
10.            for (Object result : query.getResultList()) {
11.                currObj = (Object[]) result;
12.                participant = new Participant(currObj);
13.                participants.add(participant);
14.            }
15.        } catch (PersistenceException ignored) {
16.        }
17.        return gson.toJson(participants);
18.    }
```

/courses/mentors

```
1.     @CrossOrigin
2.     @GetMapping("/courses/mentors")
3.     public String getMentorsForCourse(@RequestParam long courseId) {
4.         Query query = session.createSQLQuery("SELECT * FROM f_mentors_from_course(:courseId)")
5.                 .setParameter("courseId", courseId);
6.         List<Mentor> mentors = new ArrayList<>();
7.         Object[] currObj;
8.         Mentor mentor;
9.         try {
10.          for (Object result : query.getResultList()) {
11.                currObj = (Object[]) result;
12.                mentor = new Mentor(currObj);
13.                mentors.add(mentor);
14.            }
15.        } catch (PersistenceException ignored) {}
16.        return gson.toJson(mentors);
17.    }
```

/courses/mentors/add

```
1.     @CrossOrigin
2.     @PostMapping("/courses/mentors/add")
3.     public ResponseEntity<HttpStatus> addMentorToCourse(@RequestBody Map<String, Long> json) {
4.         try {
5.             Query query = session.createSQLQuery(
6.                             "CALL add_mentor_to_course(:courseId, :mentorId)")
7.                     .setParameter("courseId", json.get("courseId"))
8.                     .setParameter("mentorId", json.get("mentorId"));
9.             System.out.println(query.getResultList());
10.        } catch (PersistenceException e) {
11.            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
12.        } catch (NegativeArraySizeException ignored) {
13.        }
14.        return ResponseEntity.ok(HttpStatus.OK);
15.    } }
```

/courses/id/{courseId}

```
1.     @CrossOrigin
2.     @GetMapping("/courses/id/{courseId}")
3.     public Course getCourseById(@PathVariable("courseId") long courseId) {
4.         Optional queryResult = session.createQuery("from Course c where c.id=:courseId")
5.                 .setParameter("courseId", courseId).stream().findFirst();
6.         if (queryResult.isPresent()) {
7.             return (Course) queryResult.get();
8.         }
9.         return null; }
```

/participants

```
1. @RestController
2. public class UsersController extends MainController {
3.     @CrossOrigin
4.     @GetMapping("/participants")
5.     public List<Participant> getParticipants() {
6.        Object[] queryResults = session.createQuery("from Participant").stream().toArray();
7.         List<Participant> participants = new ArrayList<>();
8.         Arrays.stream(queryResults).forEach(queryResult -> participants.add((Participant) queryResult));
9.         return participants;
10.     }
```


```
[{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333
111","email":"karolinaWajda@gmail.com"},{"id":24,"firstName":"Jan","lastName":"Kowalski","phoneNumber":"+48
555 111
222","email":"jankowalski@gmail.com"},{"id":25,"firstName":"Anna","lastName":"Nowak","phoneNumber":"+48 555
222
333","email":"annanowak@hotmail.com"},{"id":26,"firstName":"Tomasz","lastName":"Kaminski","phoneNumber":"+48
555 444
555","email":"tomaszkaminski@yahoo.com"},{"id":27,"firstName":"Ewa","lastName":"Szymanska","phoneNumber":"+48
555 666
777","email":"ewaszymanska@outlook.com"},{"id":28,"firstName":"Piotr","lastName":"Wojcik","phoneNumber":"+48
555 888
999","email":"piotrw@gmail.com"},{"id":29,"firstName":"Iwona","lastName":"Kaczmarek","phoneNumber":"+48 555
000
111","email":"iwonakaczmarek@hotmail.com"},{"id":30,"firstName":"Adam","lastName":"Lewandowski","phoneNumber"
:"+48 555 222
333","email":"adamlewandowski@gmail.com"},{"id":31,"firstName":"Magdalena","lastName":"Witkowska","phoneNumbe
r":"+48 555 444
555","email":"magdalenawitkowska@yahoo.com"},{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNum
ber":"+48 555 666 777","email":"krzysztofjaworski@outlook.com"}]
```


/participants/{participantId}

```
1.     @CrossOrigin
2.     @GetMapping("/participants/{participantId}")
3.     public Participant getParticipant(@PathVariable("participantId") long participantId) {
4.         Optional queryResults = session.createQuery("from Participant p where p.id = :participantId")
5.                 .setParameter("participantId", participantId)
6.                 .stream()
7.                 .findFirst();
8.         if (queryResults.isPresent()) {
9.             return (Participant) queryResults.get();
10.        }
11.        return null;
12.    }
13.
```

/mentors

```
1.     @CrossOrigin
2.     @GetMapping("/mentors")
3.     public List<Mentor> getMentors() {
4.         Object[] queryResults = session.createQuery("from Mentor").stream().toArray();
5.         List<Mentor> mentors = new ArrayList<>();
6.         Arrays.stream(queryResults).forEach(queryResult -> mentors.add((Mentor) queryResult));
7.         return mentors;
8.     }
9.
```


```
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222
333","email":"tomaszNowak@gmail.com"},{"id":5,"firstName":"Alicja","lastName":"Kowalska","phoneNumber":"+48
444 555
666","email":"alicjaKowalska@gmail.com"},{"id":6,"firstName":"Piotr","lastName":"Lewandowski","phoneNumber":"
+48 777 888
999","email":"piotrLewandowski@gmail.com"},{"id":7,"firstName":"Anna","lastName":"Wójcik","phoneNumber":"+48
111 333
555","email":"annaWojcik@gmail.com"},{"id":8,"firstName":"Marek","lastName":"Szczepański","phoneNumber":"+48
222 444
666","email":"marekSzczepanski@gmail.com"},{"id":9,"firstName":"Katarzyna","lastName":"Dąbrowska","phoneNumbe
r":"+48 333 666
```

999","email":"katarzynaDabrowska@gmail.com"},{"id":10,"firstName":"Tadeusz","lastName":"Mazur","phoneNumber":"+48 111 555
999","email":"tadeuszMazur@gmail.com"},{"id":11,"firstName":"Magdalena","lastName":"Witkowska","phoneNumber":"+48 555 777
999","email":"magdalenaWitkowska@gmail.com"},{"id":12,"firstName":"Marcin","lastName":"Kaczmarek","phoneNumber":"+48 333 777 111","email":"marcinKaczmarek@gmail.com"}]

/mentors/{mentorId}

```
    @CrossOrigin
    @GetMapping("/mentors/{mentorId}")
    public Mentor getMentor(@PathVariable("mentorId") long mentorId) {
        Optional queryResults = session.createQuery("from Mentor m where m.id = :mentorId")
                .setParameter("mentorId", mentorId)
                .stream()
                .findFirst();
        if (queryResults.isPresent()) {
            return (Mentor) queryResults.get();
        }
        return null;
    }
}

```

/categories

```
@RestController
public class CategoriesController extends MainController {
    @CrossOrigin
    @GetMapping("/categories")
    public List<Category> getCategories() {
        Object[] queryResults = session.createQuery("from Category").stream().toArray();
        List<Category> reservations = new ArrayList<>();
        Arrays.stream(queryResults).forEach(queryResult -> reservations.add((Category) queryResult));
        return reservations;
    }
```

[{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty.
"},{"id":3,"categoryName":"Geografia","description":"Uczestnicy poznają różne aspekty geografii, takie jak geografia fizyczna, polityczna czy ekonomiczna. Nauczą się rónież korzystać z narzędzi geograficznych i technologii, takich jak systemy informacji geograficznej."}]

/categories/courses

```
    @CrossOrigin
    @PostMapping("/categories/courses")
    public ResponseEntity<HttpStatus> addCourseToCategory(@RequestBody Map<String, Long> json) {
        try {
            Query query = session.createSQLQuery(
                            "CALL add_course_to_category(:courseId, :categoryId)")
                    .setParameter("courseId", json.get("courseId"))
                        .setParameter("categoryId", json.get("categoryId"));
            System.out.println(query.getResultList());
        } catch (PersistenceException e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
        } catch (NegativeArraySizeException ignored) {
        }
        return ResponseEntity.ok(HttpStatus.OK);
    }
}

```

/invoices

```
@RestController
public class InvoicesController extends MainController {
    @CrossOrigin
    @GetMapping("/invoices")
    public String getInvoices() {
        Object[] objects = session.createSQLQuery("SELECT * FROM INVOICES_VIEW").stream().toArray();
        List<InvoiceViewElement> invoiceViewElements = new ArrayList<>();
        Object[] currObj;
        InvoiceViewElement invoiceViewElement;
        for (Object result : objects) {
            currObj = (Object[]) result;
            invoiceViewElement = new InvoiceViewElement(currObj);
            invoiceViewElements.add(invoiceViewElement);
        }
        return gson.toJson(invoiceViewElements);
    }

```


```
[{"id":62,"allPrice":12,"transactionDate":"Jun 9, 2023, 5:12:57
PM","firstName":"Magdalena","lastName":"Witkowska"},{"id":102,"allPrice":10000,"transactionDate":"Jun 10,
2023, 4:39:29
PM","firstName":"Karolina","lastName":"Wajda"},{"id":104,"allPrice":10000,"transactionDate":"Jun 10, 2023,
4:51:38 PM","firstName":"Karolina","lastName":"Wajda"},{"id":117,"allPrice":10000,"transactionDate":"Jun 10,
2023, 3:26:25
PM","firstName":"Karolina","lastName":"Wajda"},{"id":121,"allPrice":10000,"transactionDate":"Jun 10, 2023,
3:37:42 PM","firstName":"Krzysztof","lastName":"Jaworski"},{"id":122,"allPrice":15000,"transactionDate":"Jun
10, 2023, 3:37:42
PM","firstName":"Krzysztof","lastName":"Jaworski"},{"id":123,"allPrice":15000,"transactionDate":"Jun 10,
2023, 3:49:07
PM","firstName":"Krzysztof","lastName":"Jaworski"},{"id":124,"allPrice":15000,"transactionDate":"Jun 10,
2023, 3:49:07 PM","firstName":"Krzysztof","lastName":"Jaworski"}]
```


/invoices/users

```
    @CrossOrigin
    @GetMapping("/invoices/users")
    public String getInvoicesForUser(@RequestParam("participantId") long participantId) {
        Query query = session.createSQLQuery("SELECT * FROM f_invoices_for_participant(:participantId)")
                .setParameter("participantId", participantId);
        List<Invoice> invoices = new ArrayList<>();
        Object[] currObj;
        Invoice invoice;
        try {
            for (Object result : query.getResultList()) {
                currObj = (Object[]) result;
                invoice = new Invoice(currObj);
                invoices.add(invoice);
            }
        } catch (PersistenceException ignored) {
        }
        return gson.toJson(invoices);
    }

```

/invoices/unpaid/sum

```
    @CrossOrigin
    @GetMapping("/invoices/unpaid/sum")
    public BigDecimal getSumOfUnpaidInvoices(@RequestParam("participantId") long participantId) {
        Query query = session.createSQLQuery("SELECT f_amount_to_pay_for_participant(:participantId) FROM
DUAL")
                .setParameter("participantId", participantId);
        BigDecimal finalSum = BigDecimal.ZERO;
        Object result = query.getSingleResult();
        if (result != null) {
            finalSum = BigDecimal.valueOf(Long.parseLong(result.toString()));
        }
        return finalSum;
    }
}
```

/logs

```java
@RestController
public class LogsController extends MainController {

    @CrossOrigin
    @GetMapping("/logs")
    public List<LogTable> getLogs() {
        Object[] queryResults = session.createQuery("from Log_table").stream().toArray();
        List<LogTable> logs = new ArrayList<>();
        Arrays.stream(queryResults).forEach(queryResult -> logs.add((LogTable) queryResult));
        return logs;
    }

```


[{"id":102,"reservation":{"id":162,"status":"PAID","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania
składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},"logDate":"2023-06-10T12:47:02.000+00:00","status":"NEW"},{"id":105,"reservation":
{"id":165,"status":"PAID","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania
składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},"logDate":"2023-06-10T12:47:12.000+00:00","status":"NEW"},{"id":121,"reservation":
{"id":164,"status":"REJECTED","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":{"id":14,"title":"Mobile App
Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się
z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]}},"logDate":"2023-06-
11T00:13:47.000+00:00","status":"REJECTED"},{"id":81,"reservation":{"id":141,"status":"NEW","participant":{"id":31,"firstName":"Magdalena","lastName":"Witkowska","phoneNumber":"+48 555 444
555","email":"magdalenawitkowska@yahoo.com"},"course":{"id":17,"title":"Starożytna grecja","price":"25000","startDate":"2023-11-11","endDate":"2024-02-11","maxNoPlaces":55,"availablePlaces":55,"category":
{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":
[]}},"logDate":"2023-06-09T19:24:41.000+00:00","status":"NEW"},{"id":82,"reservation":{"id":121,"status":"REJECTED","participant":{"id":30,"firstName":"Adam","lastName":"Lewandowski","phoneNumber":"+48 555 222
333","email":"adamlewandowski@gmail.com"},"course":{"id":44,"title":"C++ Programming","price":"12","startDate":"2023-06-13","endDate":"2023-06-29","maxNoPlaces":0,"availablePlaces":0,"category":null,"mentors":
[]}},"logDate":"2023-06-09T19:32:56.000+00:00","status":"REJECTED"},{"id":109,"reservation":{"id":167,"status":"NEW","participant":{"id":28,"firstName":"Piotr","lastName":"Wojcik","phoneNumber":"+48 555 888
999","email":"piotrw@gmail.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":
{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania
programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},"logDate":"2023-06-10T14:53:50.000+00:00","status":"NEW"},
{"id":110,"reservation":{"id":168,"status":"NEW","participant":{"id":28,"firstName":"Piotr","lastName":"Wojcik","phoneNumber":"+48 555 888 999","email":"piotrw@gmail.com"},"course":{"id":14,"title":"Mobile App
Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się
z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]}},"logDate":"2023-06-10T14:53:53.000+00:00","status":"NEW"},
{"id":111,"reservation":{"id":169,"status":"PAID","participant":{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNumber":"+48 555 666 777","email":"krzysztofjaworski@outlook.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania

(Ze względu na dużą ilość danych w sprawozdaniu powyższy zrzut ekranu zawiera tylko część danych z /logs)

/logs/{logId}

```java
    @CrossOrigin
    @GetMapping("/logs/{logId}")
    public LogTable getLog(@PathVariable("logId") long logID) {
        Optional queryResults = session.createQuery("from Log_table log where log.id = :logId")
                .setParameter("logId", logID)
                .stream()
                .findFirst();
        if (queryResults.isPresent()) {
            return (LogTable) queryResults.get();
        }
        return null;
    }
}

```

/payments/reservations

```java
@RestController
public class PaymentsController extends MainController {
    @CrossOrigin
    @PostMapping("/payments/reservations")
    public ResponseEntity<HttpStatus> payForReservation(@RequestBody Map<String, String> json) {
        try {
            Query query = session.createSQLQuery(
                            "CALL PAY_FOR_RESERVATION(:reservationId, :paymentType)")
                    .setParameter("reservationId", Long.parseLong(json.get("reservationId")))
                    .setParameter("paymentType", json.get("paymentType").toUpperCase());
            System.out.println(query.getResultList());
        } catch (PersistenceException e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
        } catch (NegativeArraySizeException ignored) {
        }
        return ResponseEntity.ok(HttpStatus.OK);
    }

```

/payments/participants/reservations

```
 1.     @CrossOrigin
 2.     @PostMapping("/payments/participants/reservations")
 3.     public ResponseEntity<HttpStatus> payForAllParticipantReservations(@RequestBody Map<String, String>
json) {
 4.         try {
 5.             Query query = session.createSQLQuery(
 6.                             "CALL PAY_FOR_ALL_UNPAID_RESERVATIONS(:participantId, :paymentType)")
 7.                     .setParameter("participantId", Long.parseLong(json.get("participantId")))
 8.                     .setParameter("paymentType", json.get("paymentType").toUpperCase());
 9.             System.out.println(query.getResultList());
10.         } catch (PersistenceException e) {
11.             return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
12.         } catch (NegativeArraySizeException ignored) {
13.         }
14.         return ResponseEntity.ok(HttpStatus.OK);
15.     }
16. }
```

/reservations

```
1. @RestController
2. public class ReservationsController extends MainController {
3.     @CrossOrigin
4.     @GetMapping("/reservations")
5.     public List<Reservation> getReservations() {
6.         Object[] queryResults = session.createQuery("from Reservation").stream().toArray();
7.         List<Reservation> reservations = new ArrayList<>();
8.         Arrays.stream(queryResults).forEach(queryResult -> reservations.add((Reservation) queryResult));
9.         return reservations;
10.     }
```

```
1.     @CrossOrigin
 2.     @PostMapping("/reservations")
 3.     public ResponseEntity<HttpStatus> makeReservation(@RequestBody Map<String, Long> json) {
 4.         try {
 5.             Query query = session.createSQLQuery(
 6.                             "CALL make_reservation(:courseId, :participantId)")
 7.                     .setParameter("courseId", json.get("courseId"))
 8.                     .setParameter("participantId", json.get("participantId"));
 9.             System.out.println(query.getResultList());
10.         } catch (PersistenceException e) {
11.             return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
12.         } catch (NegativeArraySizeException ignored) {
13.         }
14.         return ResponseEntity.ok(HttpStatus.OK);
15.     }
```


[{"id":162,"status":"PAID","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania
składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},{"id":165,"status":"PAID","participant":
{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-
09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których
uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},
{"id":141,"status":"NEW","participant":{"id":31,"firstName":"Magdalena","lastName":"Witkowska","phoneNumber":"+48 555 444 555","email":"magdalenawitkowska@yahoo.com"},"course":{"id":17,"title":"Starożytna
grecja","price":"25000","startDate":"2023-11-11","endDate":"2024-02-11","maxNoPlaces":55,"availablePlaces":55,"category":{"id":2,"categoryName":"Historia","description":"Uczestnicy poznają kluczowe wydarzenia i
postaci z przeszłości oraz nauczą się analizować historyczne źródła i interpretować fakty. "},"mentors":[]},{"id":167,"status":"NEW","participant":
{"id":28,"firstName":"Piotr","lastName":"Wojcik","phoneNumber":"+48 555 888 999","email":"piotrw@gmail.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-
15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których
uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},
{"id":168,"status":"NEW","participant":{"id":28,"firstName":"Piotr","lastName":"Wojcik","phoneNumber":"+48 555 888 999","email":"piotrw@gmail.com"},"course":{"id":14,"title":"Mobile App
Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się
z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[]}},{"id":169,"status":"PAID","participant":
{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNumber":"+48 555 666 777","email":"krzysztofjaworski@outlook.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-
15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych
ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222
333","email":"tomaszNowak@gmail.com"}]}},{"id":170,"status":"PAID","participant":{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNumber":"+48 555 666
777","email":"krzysztofjaworski@outlook.com"},"course":{"id":14,"title":"Mobile App Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":
{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania
programistyczne."},"mentors":[]}},{"id":171,"status":"PAID","participant":{"id":24,"firstName":"Jan","lastName":"Kowalski","phoneNumber":"+48 555 111 222","email":"jankowalski@gmail.com"},"course":
{"id":14,"title":"Mobile App Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":
{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania
programistyczne."},"mentors":[]}},{"id":172,"status":"PAID","participant":{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNumber":"+48 555 666
777","email":"krzysztofjaworski@outlook.com"},"course":{"id":14,"title":"Mobile App Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":
{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania
programistyczne."},"mentors":[]}},{"id":173,"status":"PAID","participant":{"id":32,"firstName":"Krzysztof","lastName":"Jaworski","phoneNumber":"+48 555 666
777","email":"krzysztofjaworski@outlook.com"},"course":{"id":14,"title":"Mobile App Development","price":"15000","startDate":"2023-09-01","endDate":"2024-12-01","maxNoPlaces":15,"availablePlaces":15,"category":
{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania
programistyczne."},"mentors":[]}},{"id":101,"status":"PAID","participant":{"id":24,"firstName":"Jan","lastName":"Kowalski","phoneNumber":"+48 555 111 222","email":"jankowalski@gmail.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania
składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},{"id":163,"status":"PAID","participant":
{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-
09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których
uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},
{"id":166,"status":"REJECTED","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":
{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania
składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":
[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},{"id":161,"status":"NEW","participant":
{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":{"id":13,"title":"Java","price":"10000","startDate":"2023-06-15","endDate":"2024-
09-15","maxNoPlaces":30,"availablePlaces":30,"category":{"id":1,"categoryName":"Programowanie","description":"Kurs programowania składa się z teoretycznych wykładów oraz praktycznych ćwiczeń, podczas których
uczestnicy piszą własny kod i rozwiązują zadania programistyczne."},"mentors":[{"id":4,"firstName":"Tomasz","lastName":"Nowak","phoneNumber":"+48 111 222 333","email":"tomaszNowak@gmail.com"}]}},
{"id":164,"status":"REJECTED","participant":{"id":23,"firstName":"Karolina","lastName":"Wajda","phoneNumber":"+48 555 333 111","email":"karolinaWajda@gmail.com"},"course":{"id":14,"title":"Mobile App

/reservations/canceled

```
    @CrossOrigin
    @GetMapping("/reservations/canceled")
    public String getCanceledReservations() {
        Object[] objects = session.createSQLQuery("SELECT * FROM
CANCELED_RESERVATION_VIEW").stream().toArray();
        List<CanceledReservation> canceledReservationList = new ArrayList<>();
        Object[] currObj;
        CanceledReservation canceledReservation;
        for (Object result : objects) {
            currObj = (Object[]) result;
            canceledReservation = new CanceledReservation(currObj);
            canceledReservationList.add(canceledReservation);
        }
        return gson.toJson(canceledReservationList);
    }

```

/reservations/courses

```
    @CrossOrigin
    @GetMapping("/reservations/courses")
    public String getReservationsFromCourse(@RequestParam long courseId) {
        Query query = session.createSQLQuery("SELECT * FROM f_reservations_from_course(:courseId)")
                .setParameter("courseId", courseId);
        return prepareQueryWithReservations(query);
    }

    private String prepareQueryWithReservations(Query query) {
        List<ReservationFromFunction> canceledReservationList = new ArrayList<>();
        Object[] currObj;
        ReservationFromFunction reservation;
        try {
            for (Object result : query.getResultList()) {
                currObj = (Object[]) result;
                reservation = new ReservationFromFunction(currObj);
                canceledReservationList.add(reservation);
            }
        } catch (PersistenceException ignored) {
        }
        return gson.toJson(canceledReservationList);
    }

```

/reservations/unpaid/users

```
    @CrossOrigin
    @GetMapping("/reservations/unpaid/users")
    public String getUnpaidReservationsForUser(@RequestParam long userId) {
        Query query = session.createSQLQuery("SELECT * FROM f_unpaid_reservations_for_participant(:userId)")
                .setParameter("userId", userId);
        return prepareQueryWithReservations(query);
    }

```

/reservations/users/{userID}

```
    @CrossOrigin
    @GetMapping("/reservations/users/{userID}")
    public List<Reservation> getReservationsForUser(@PathVariable("userID") long userID) {
        Object[] queryResults = session.createQuery("from Reservation r where r.participant.id=:userID")
                .setParameter("userID", userID)
                .stream()
                .toArray();
        List<Reservation> reservations = new ArrayList<>();
        Arrays.stream(queryResults).forEach(queryResult -> reservations.add((Reservation) queryResult));
        return reservations;
    }

```

/reservations/participants

```
    @CrossOrigin
    @GetMapping("/reservations/participants")
    public String getReservationsForParticipant(@RequestParam long participantId) {
        Query query = session.createSQLQuery("SELECT * FROM f_reservations_for_participant(:participantId)")
                .setParameter("participantId", participantId);
        return prepareQueryWithReservations(query);
    }

```

/reservations/cancel

```
    @CrossOrigin
    @PostMapping("/reservations/cancel")
    public ResponseEntity<HttpStatus> cancelReservation(@RequestBody Map<String, Long> json){
        try {
            Query query = session.createSQLQuery(
                            "CALL cancel_reservation(:reservationId)")
                    .setParameter("reservationId", json.get("reservationId"));
            System.out.println(query.getResultList());
        } catch (PersistenceException e) {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(HttpStatus.BAD_REQUEST);
        } catch (NegativeArraySizeException ignored) {
        }
        return ResponseEntity.ok(HttpStatus.OK);
    }
}

```

Hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:oracle:thin:@dbmanage.lab.ii.agh.edu.pl:1521:DBMANAGE
        </property>
        <property name="connection.driver_class">oracle.jdbc.OracleDriver</property>
        <property name="connection.username"> </property>
        <property name="connection.password"> </property>

        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <mapping class="org.example.models.courses.Course"/>
        <mapping class="org.example.models.courses.Reservation"/>
        <mapping class="org.example.models.courses.Category"/>
        <mapping class="org.example.models.courses.Invoice"/>
        <mapping class="org.example.models.others.LogTable"/>
        <mapping class="org.example.models.users.User"/>
        <mapping class="org.example.models.users.Mentor"/>
        <mapping class="org.example.models.users.Participant"/>
    </session-factory>
</hibernate-configuration>

```

Front-end (React.js) – Komunikacja z serwerem

W poniższym wklejonym kodzie zostały pominięte elementy, które służą samemu wyświetlaniu komponentów w aplikacji. Zostały umieszczone tylko fragmenty służące do komunikacji z serwerem.

Zapytania do Bazy danych wykonywane są za pomocą Rect.js + Axios

1. Przykładowe przesłanie danych do odpowiednich komponentów oraz routing
   (To tylko wybiórcza część, aby nie zagmatwać sprawozdania, a jedynie wskazać sposób podejścia do problemu).

```
// GeneralList
<Route path="/basic/courses" element={<GeneralList
srcLink={'http://localhost:8080/courses'} text={"List of all courses"}
extendedLink={'/basic/courses/'} fkColumn={'id'}
toShow={['title','price','availablePlaces']}/>} />
// OneElementExtended
<Route path="/basic/courses/:id" element={<OneElementExtended
srcLink={'http://localhost:8080/courses/id/'}></OneElementExtended>}/>
// GeneralList
<Route path="/basic/categories" element={<GeneralList
srcLink={'http://localhost:8080/categories'} text={"List of all categories"}
/>} />
// GeneralProcedurePage
<Route path="/procedures/cancelReservation" element={<GeneralProcedurePage
requiredData={["reservationId"]} inputTypes={["number"]} text={"Cancel a
reservation"}
srcLink={'http://localhost:8080/reservations/cancel'}></GeneralProcedurePage>}
/>
// GeneralFunctionPage
<Route path="/functions/availableCoursesBetweenDates"
element={<GeneralFunctionPage requiredData={["startDate", "endDate"]}
inputTypes={["date", "date"]} text={"Search for available courses between two
dates"}
srcLink={'http://localhost:8080/courses/available/between'}></GeneralFunctionPa
ge>} />
```

Opis zmiennych przekazywanych do większości komponentów:

srcLink - link używany do pobierania odpowiednich danych z backendu

text - wyświetlany tekst w nagłówku na odpowiedniej stronie

extendedLink - link używany przez Router do wyświetlania odpowiedniego elementu OneElementExtended.

fkColumn - dane używane w pobranych danych jako klucz obcy (używane do pobierania rozszerzonych danych)

toShow - dane używane do wyświetlania w prostych danych

requiredData - dane wymagane przez backend do procedury / funkcji. Pomaga w tworzeniu odpowiednich wejść.

inputTypes - definiuje typ danych wejściowych wymienionych w requiredData (każdy indeks opisuje odpowiedni typ danych wejściowych).

(Jedynym wyjątkiem jest komponent **addCourse**, gdzie wszystko jest obsługiwane wewnątrz komponentu)

```
<Route path="/procedures/addCourse" element={<AddCourses />} />
```

2. Post służący dodawaniu kursu

```
const AddCourses = () => {
  const [responseMessage, setResponseMessage] = useState("")
  const addCourse = (event) => {
    event.preventDefault()
    let course = {
      title: event.target.courseTitle.value,
      price: event.target.coursePrice.value,
      startDate: event.target.courseStartDate.value,
      endDate: event.target.courseEndDate.value,
      maxNoPlaces: event.target.courseMaxPlaces.value,
      availablePlaces: event.target.courseAvailablePlaces.value,
    }
    axios.post('http://localhost:8080/courses', course, { headers: { "Content-
Type": "application/json" } }).then((res) => {
      if (res.status === 200) {
        setResponseMessage("Added new " + course.title + " course")
      }
    })
      .catch((err) => {
        console.log(err.response.data)
        setResponseMessage("Error with adding " + course.title + " course.
Error:" + err.response.data.error)
      })

  }
```

3. Get służący pobieraniu z bazy potrzebnych danych (np. list kursów).

```
const GeneralList = ({ srcLink, text, extendedLink, fkColumn, toShow }) => {
  const [elements, setElements] = useState([])
  const navigate = useNavigate()

  useEffect(() => {
    setElements([])
    axios.get(srcLink)
      .then((res) => {
        setElements(res.data)
      })
      .catch((res) => {
        console.log("Error caught: " + res)
      })
  }, [text, srcLink])
```

4. Post służący do przekazywania do bazy potrzebnych danych (Przy pomocy procedur).

```
const GeneralProcedurePage = (input) => {
  const [responseMessage, setResponseMessage] = useState("")

  useEffect(() => {
    setResponseMessage("")
  }, [input.text])

  const handle = (event) => {
    event.preventDefault()
    let data = {}
    for (let i = 0; i < input.inputTypes.length; i++) {
      data = Object.assign({ [input.requiredData[i]]:
event.target[input.requiredData[i]].value }, data)
    }

    axios.post(input.srcLink, data, { headers: { "Content-Type":
"application/json" } }).then((res) => {
      if (res.status === 200) {
        setResponseMessage("Procedure run correctly with data " +
JSON.stringify(data))
      }
    }).catch((err) => {
        setResponseMessage("Error with procedure using data " +
JSON.stringify(data) + ". Error:" + err.response.data.error)})
  }
```

5. Get służący do pobrania z bazy potrzebnych danych (Przy pomocy funkcji).

```
const GeneralFunctionPage = (input) => {
  const [responseMessage, setResponseMessage] = useState(Array([]))

  useEffect(() => {
    setResponseMessage([])
  }, [input.text])

  const handle = (event) => {
    event.preventDefault()
    let data = {}
    for (let i = 0; i < input.inputTypes.length; i++) {
      data = Object.assign({ [input.requiredData[i]]:
event.target[input.requiredData[i]].value }, data)
    }
    axios.get(input.srcLink, { params: data }).then((res) => {
      if (res.status === 200) {
        let tmp = res.data
        if (!Array.isArray(tmp)) {
          tmp = [res.data]
        }
        setResponseMessage(tmp)
      }
    })
      .catch((err) => {
        console.log(err.response.data)
        setResponseMessage(["Error with function using data " +
JSON.stringify(data) + ". Error:" + err.response.data.error])
      })
  }
```

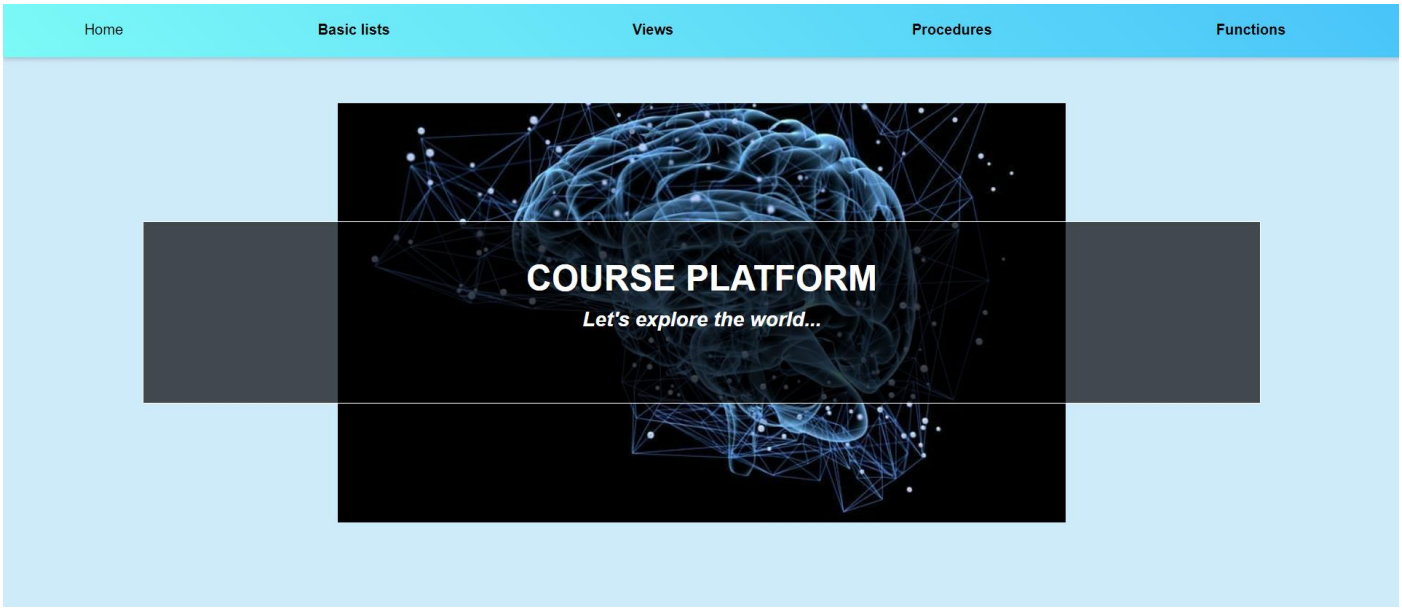
Front-end (React.js) - Wygląd

Front-end przedstawia wszystkie funkcjonalności zamieszczone w bazie. Jest to stricte aplikacja do zarządzania bazą danych potrzebną do obsługi platformy kursów.

Cała logika aplikacji znajduję się w folderze **Frontend/course_platform/src**

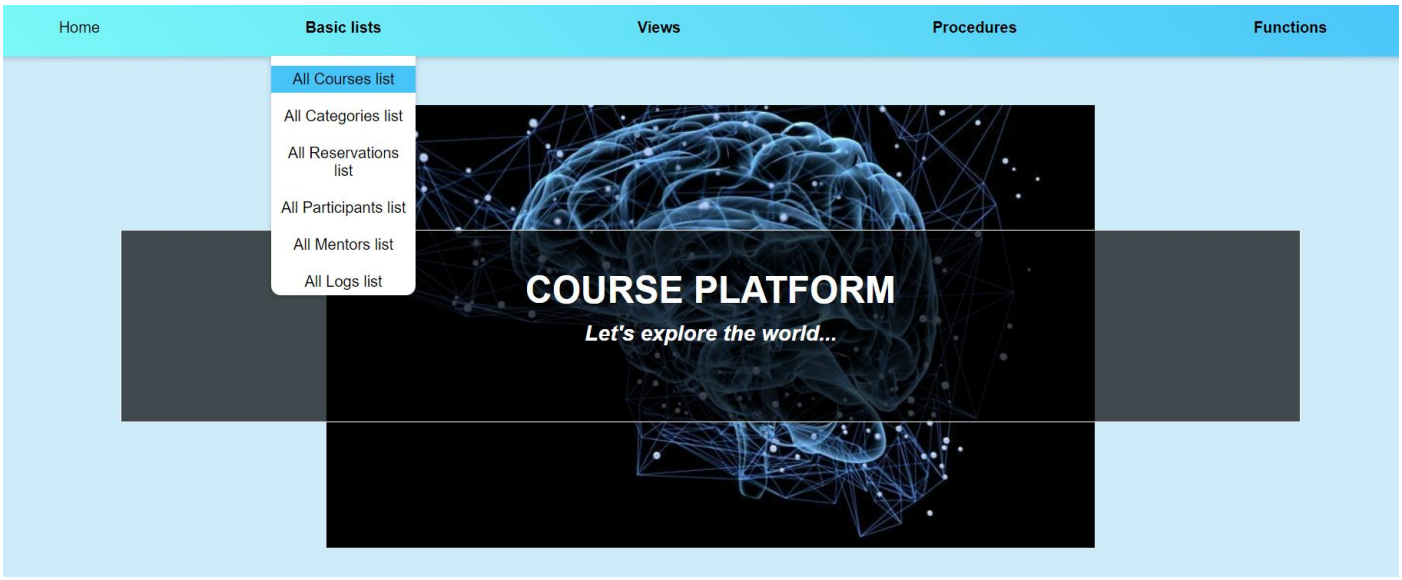
Strona główna - Home

W panelu nawigacyjnym są rozwijane opcje zobaczenia wszystkich elementów z bazy. (Możliwość wybrania widoków, procedur i funkcji, z których użytkownik chciałby skorzystać).

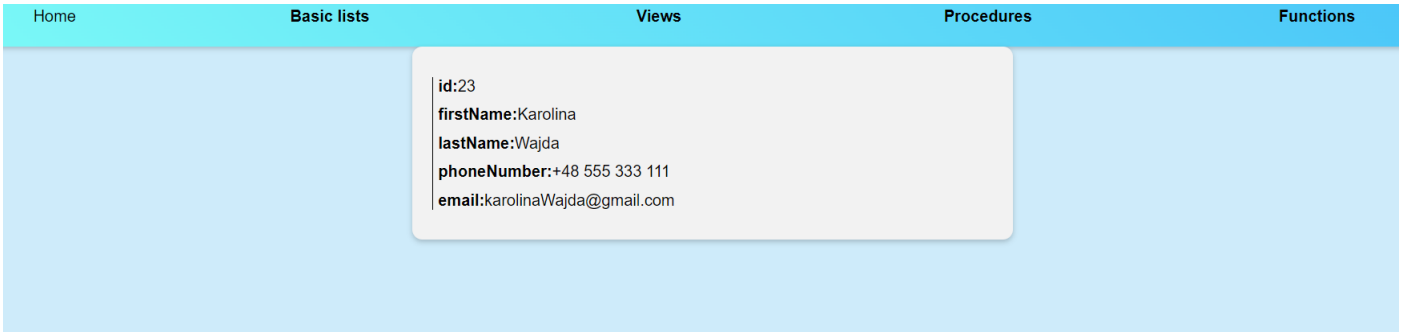


Menu – dostęp do funkcji/procedur

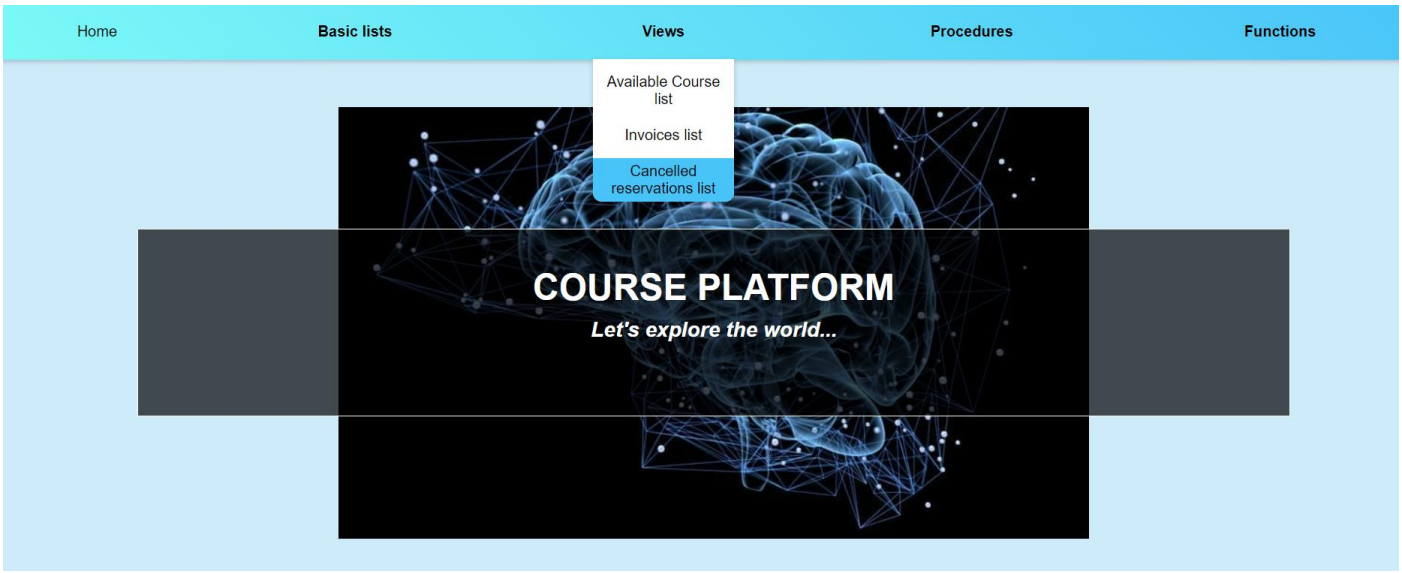
BASIC LIST – dostęp do podstawowych elementów budujących naszą bazę danych (Lista kursów, kategorii, rezerwacji, uczestników, logów)



Szczegółowy widok wybranego uczestnika (Po naciśnięciu na dowolny rekord zwrócony z bazy: Basic Lists -> All Participants List -> [kafelek]

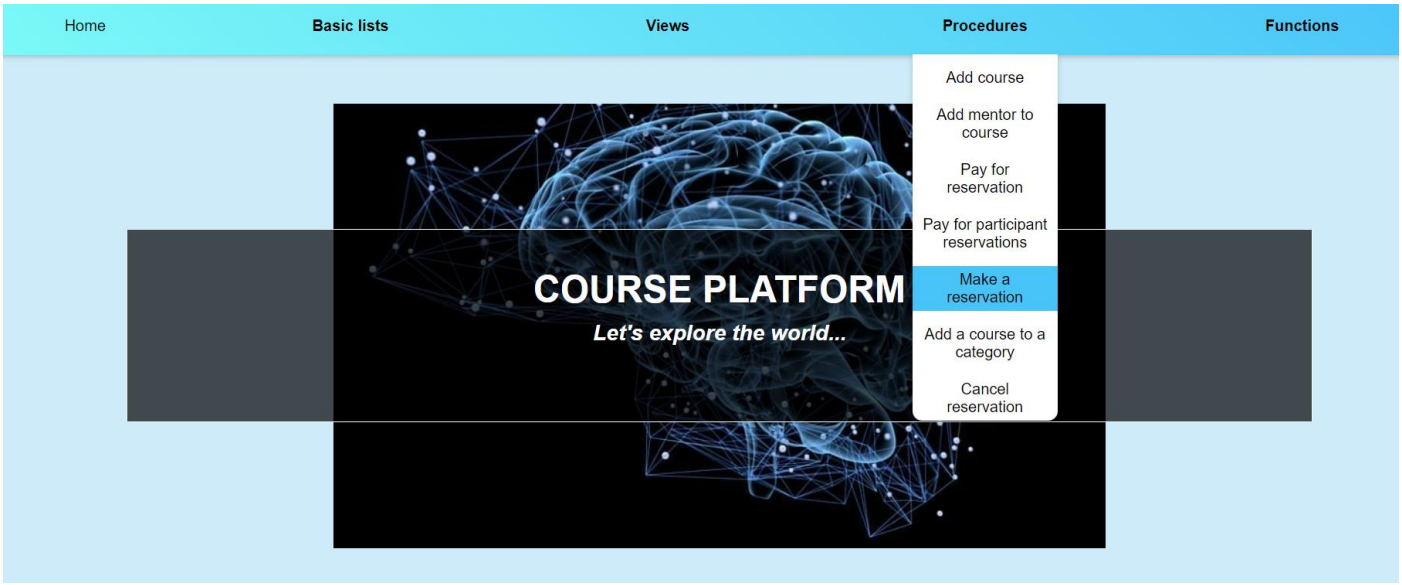


VIEWS – widoki dostępne w bazie

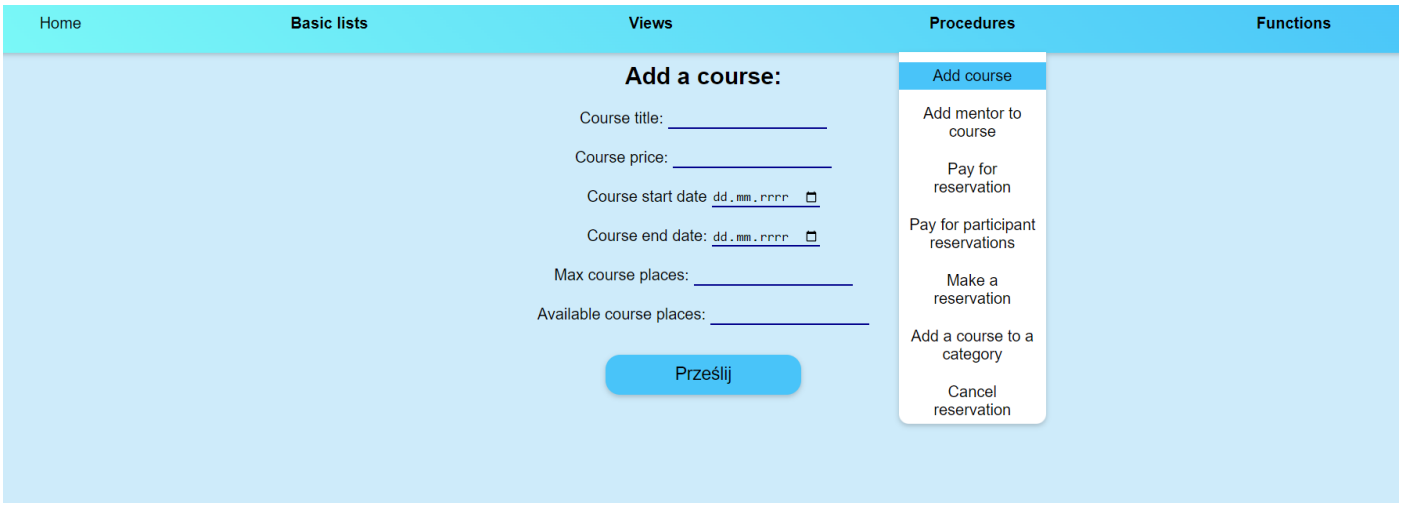


Szczegółowy widok wybranego widoku (Dostępne kursy)

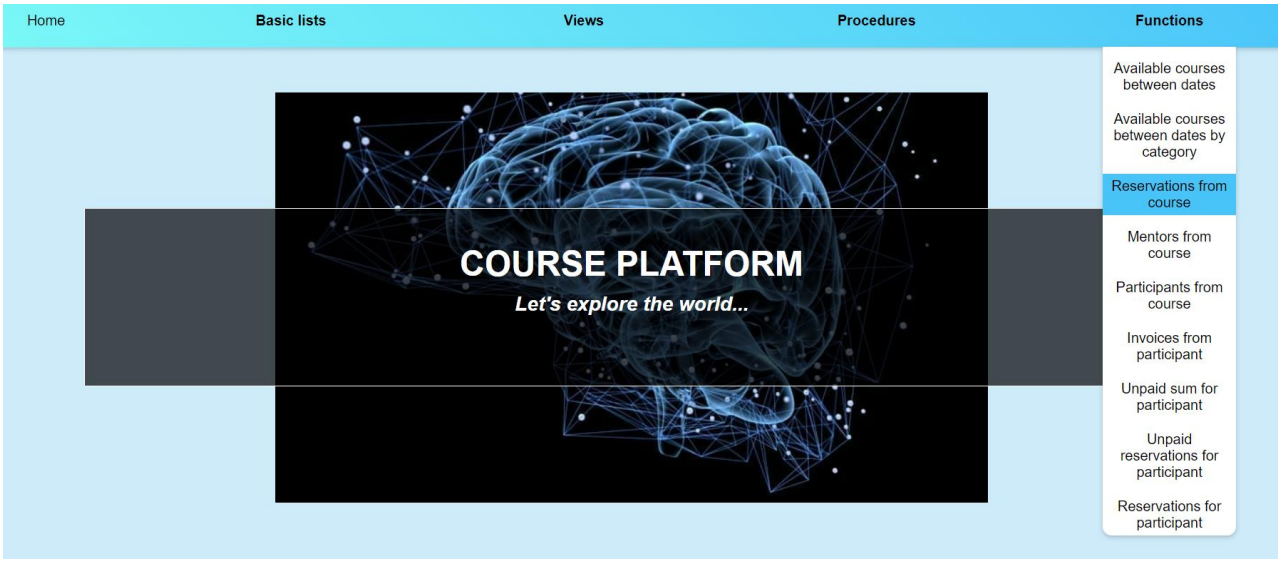
PROCEDURES – procedury dostępne w bazie



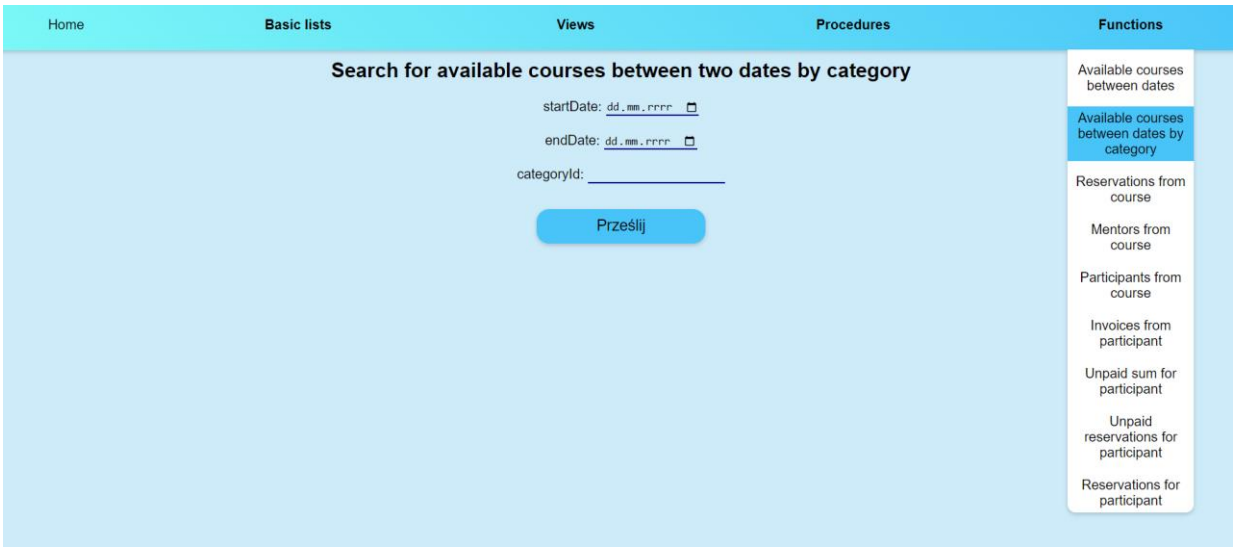
Szczegółowy widok wybranej procedury (Dodawanie kursu)



FUNCTIONS – funkcje dostępne w bazie

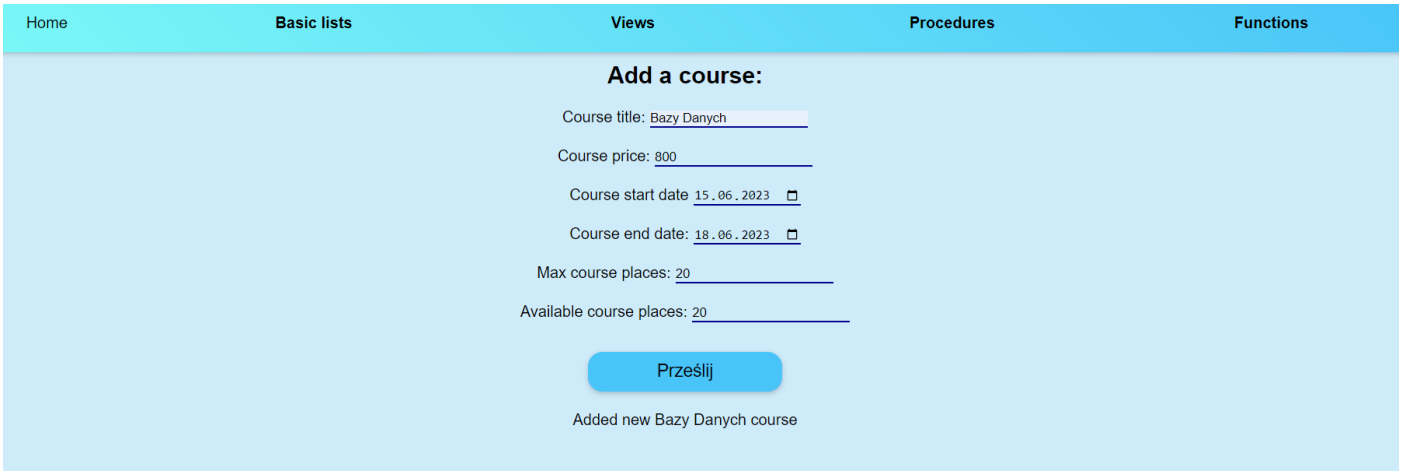


Szczegółowy widok wybranej funkcji (Dostępne kursy z danej kategorii pomiędzy konkretnymi datami)

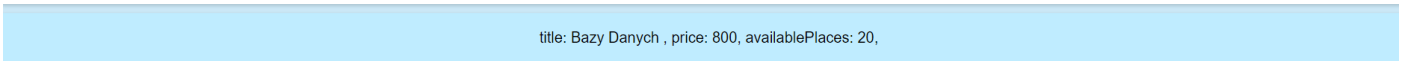


Przykładowe operacje

1. Dodanie kursu



Pojawia się na liście Basic lists -> all courses (i nie tylko)



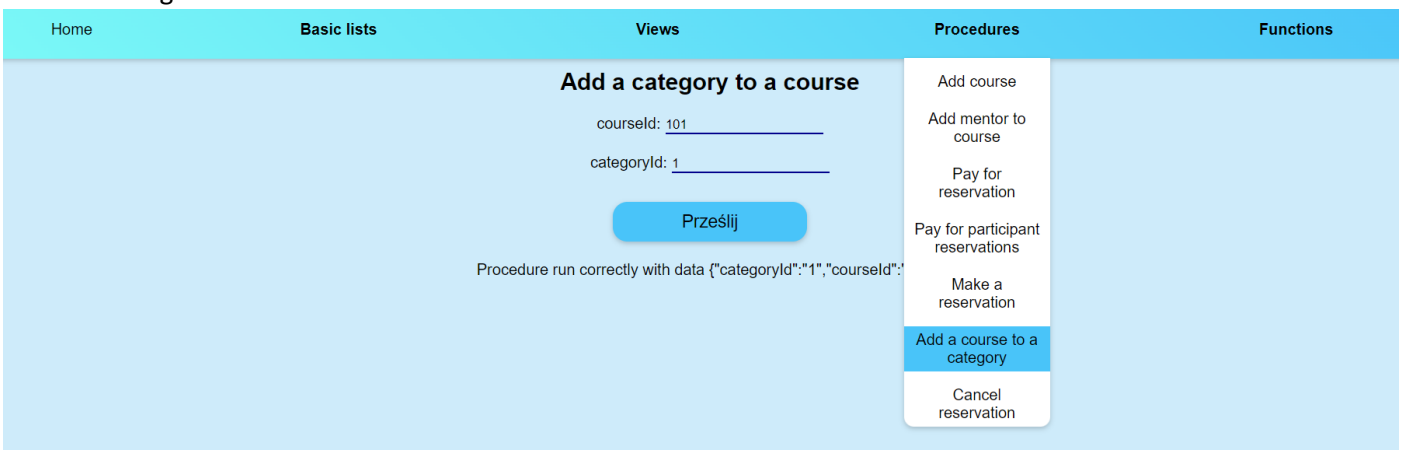
Jest w bazie

	ID 1	TITLE	AVAILABLE_PLACES	START_DATE	END_DATE	MAX_NO_PLACES	PRICE	CATEGORY_FK
1	101	Bazy Danych	20	2023-06-15 02:…	2023-06-18 02…	20	800.00	<null>

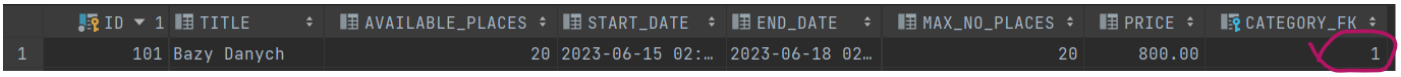
2. Nadanie kursowi kategorii

Lista kategorii:

Nadanie kategorii:



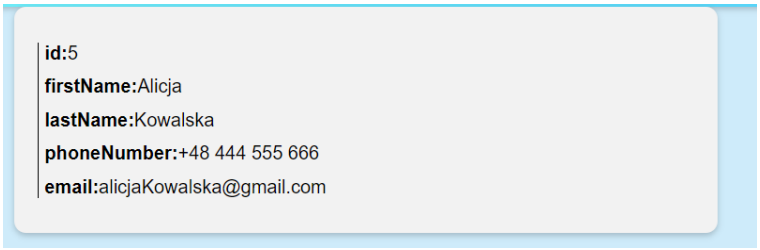
Kurs ma przypisaną kategorie w bazie danych



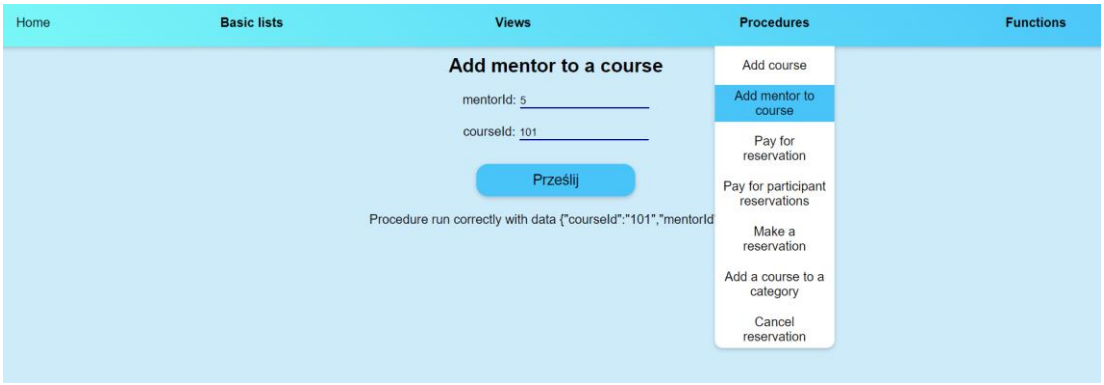
	ID	TITLE	AVAILABLE_PLACES	START_DATE	END_DATE	MAX_NO_PLACES	PRICE	CATEGORY_FK
1	101	Bazy Danych	20	2023-06-15 02:…	2023-06-18 02…	20	800.00	1

3. Dodanie mentora do kursu

Mentor:



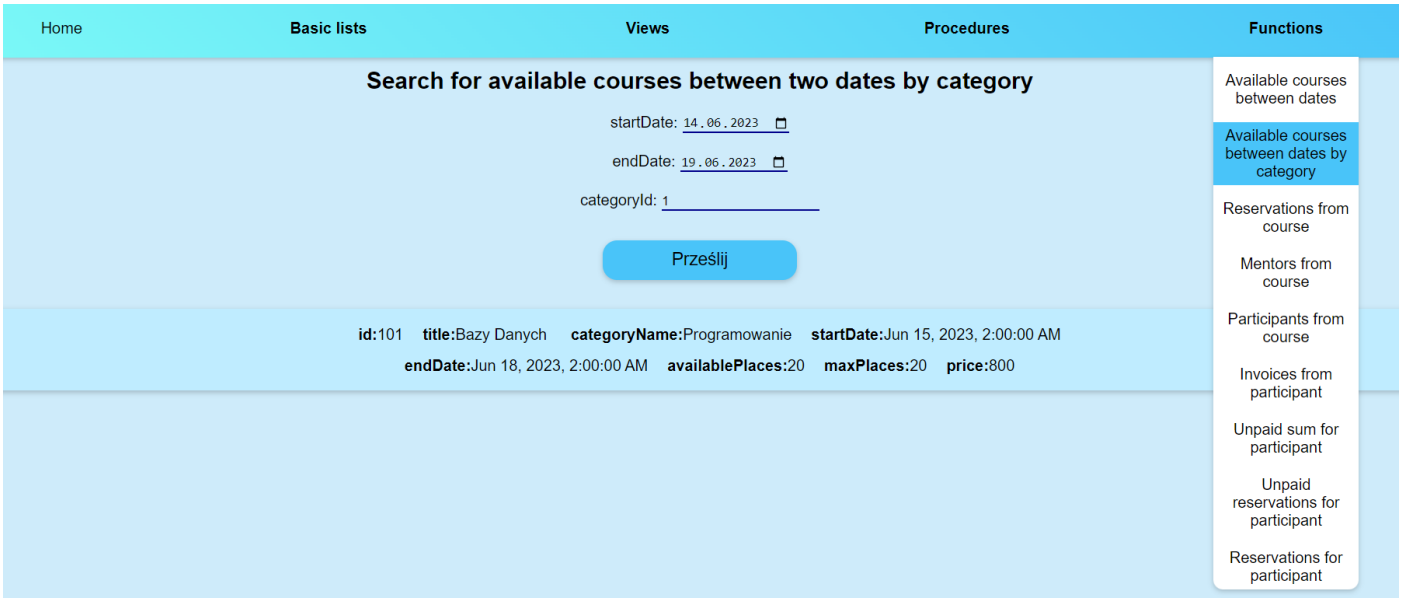
Dodanie mentora do kursu



Połączenie mentora z kursem w bazie

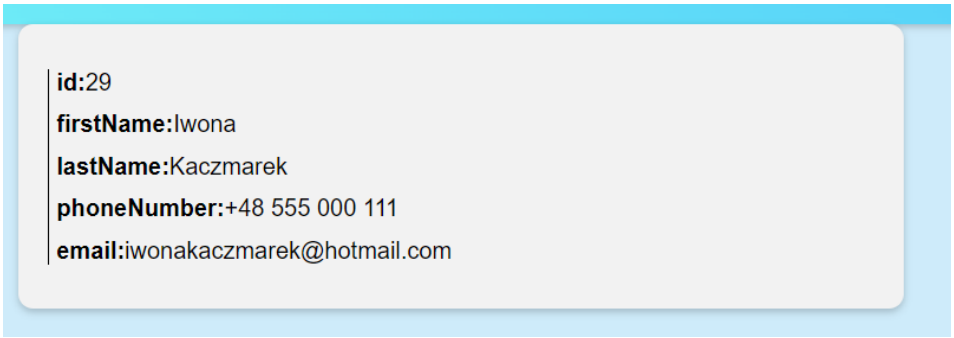
	MENTORS_ID	COURSES_ID 1
1	5	101

4. Sprawdzenie dostępnych kursów pomiędzy danymi datami w danej kategorii

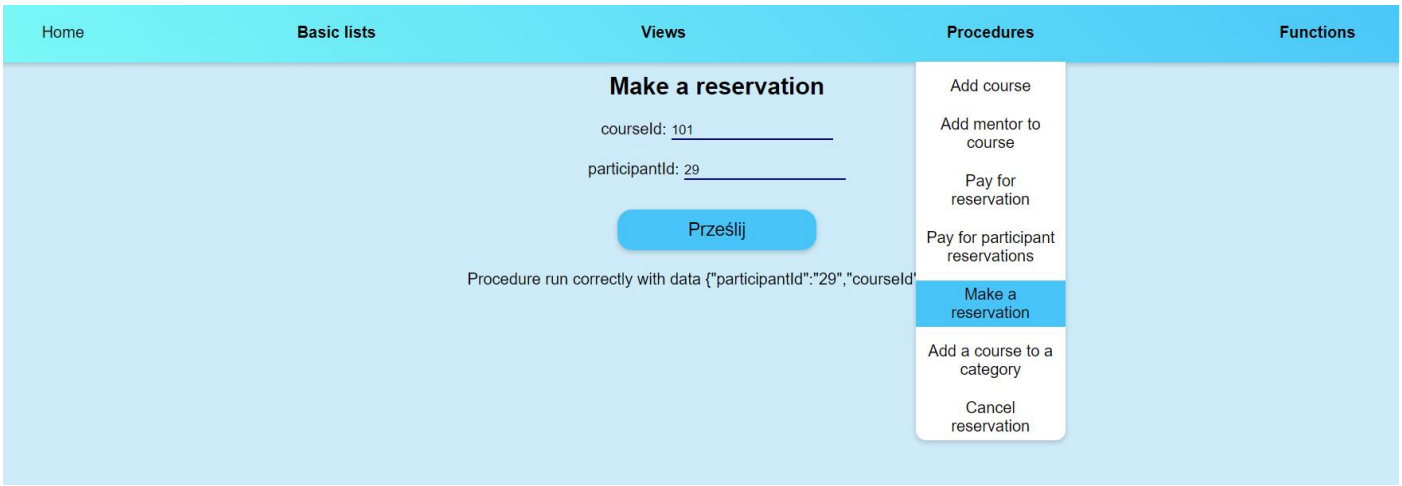


5. Zrobienie rezerwacji

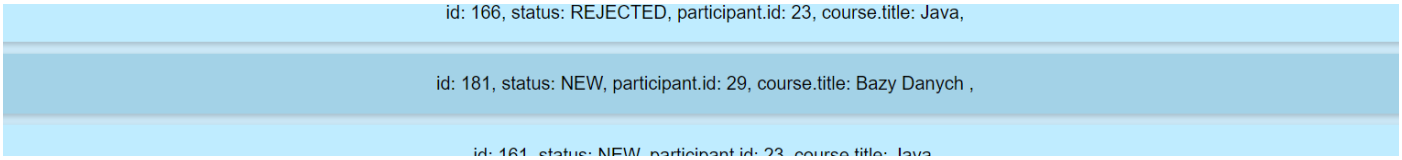
Uczestnik:



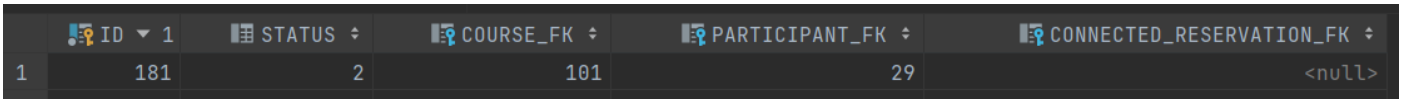
Kupno biletu/zrobienie rezerwacji



Rezerwacja wyświetlana na liście rezerwacji

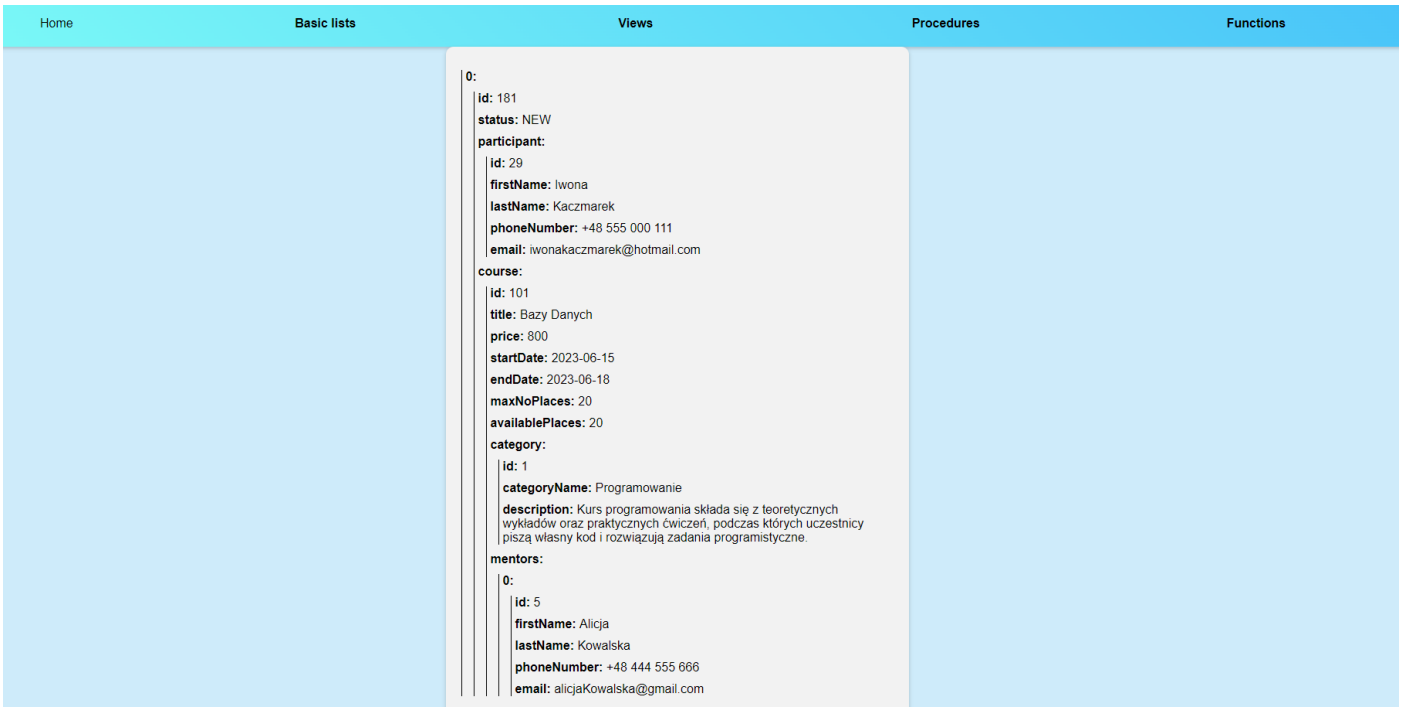


Rezerwacja znalazła się w bazie

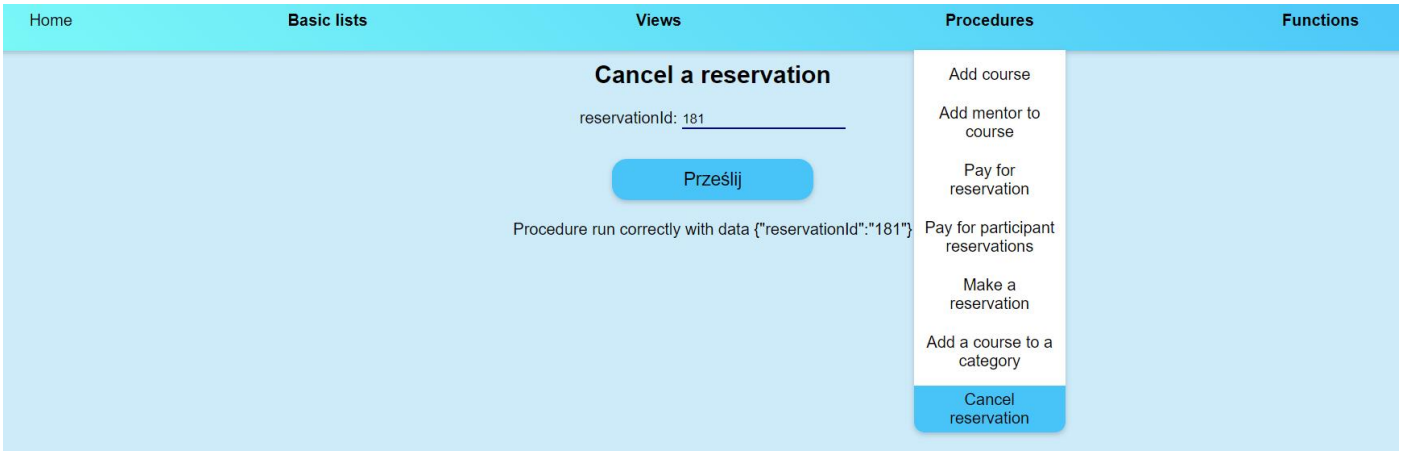


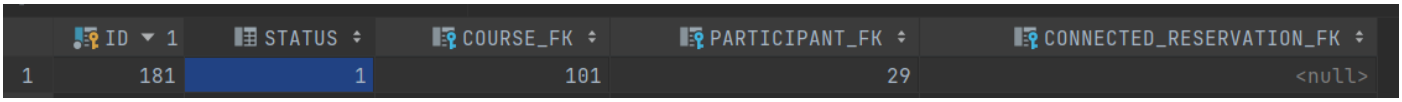
	ID 1	STATUS	COURSE_FK	PARTICIPANT_FK	CONNECTED_RESERVATION_FK
1	181	2	101	29	<null>

Szczegółowy widok rezerwacji



6. Rezygnacja z rezerwacji





	ID 1	STATUS	COURSE_FK	PARTICIPANT_FK	CONNECTED_RESERVATION_FK
1	181	1	101	29	<null>

7. Sprawdzenie Logów

	ID 1	LOG_DATE	STATUS	RESERVATION_FK
1	144	2023-06-13 19:38:00.000000	2	183
2	143	2023-06-13 19:37:33.000000	2	182
3	142	2023-06-13 19:27:44.000000	1	181
4	141	2023-06-13 19:24:01.000000	2	181

id: 141, status: NEW, reservation.id: 181,

id: 142, status: REJECTED, reservation.id: 181,

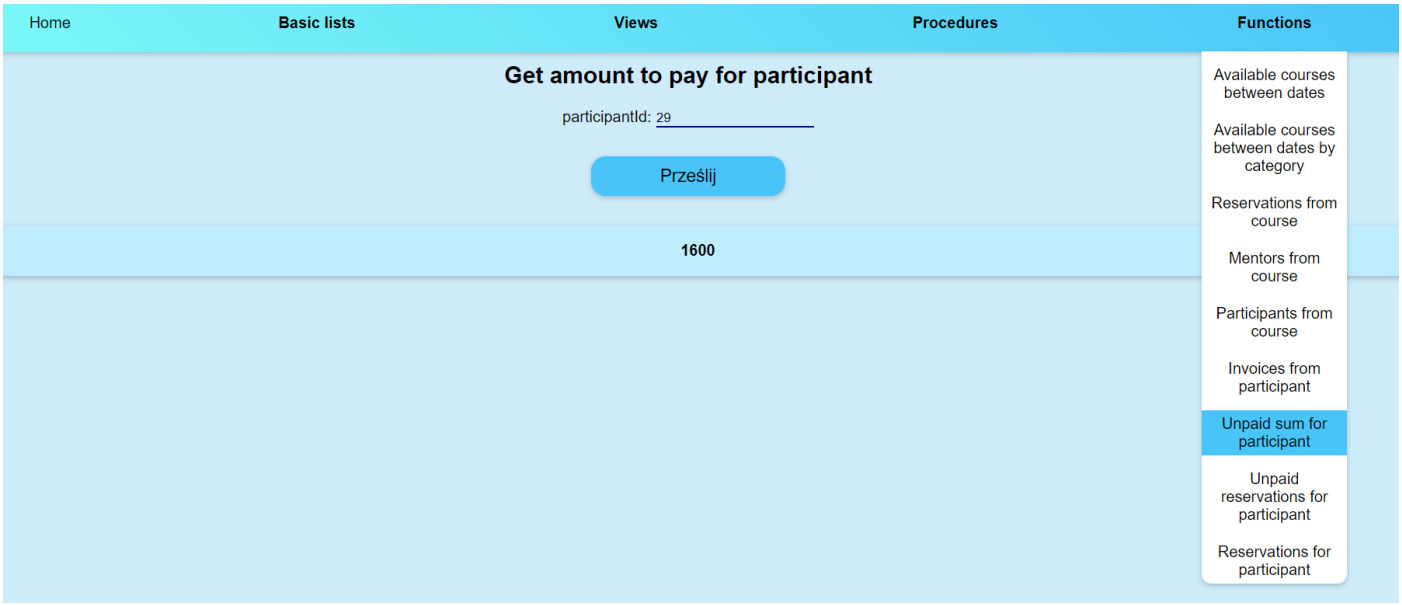
8. Zrobienie rezerwacji (2 razy)

Widzimy w bazie dokonane rezerwacje

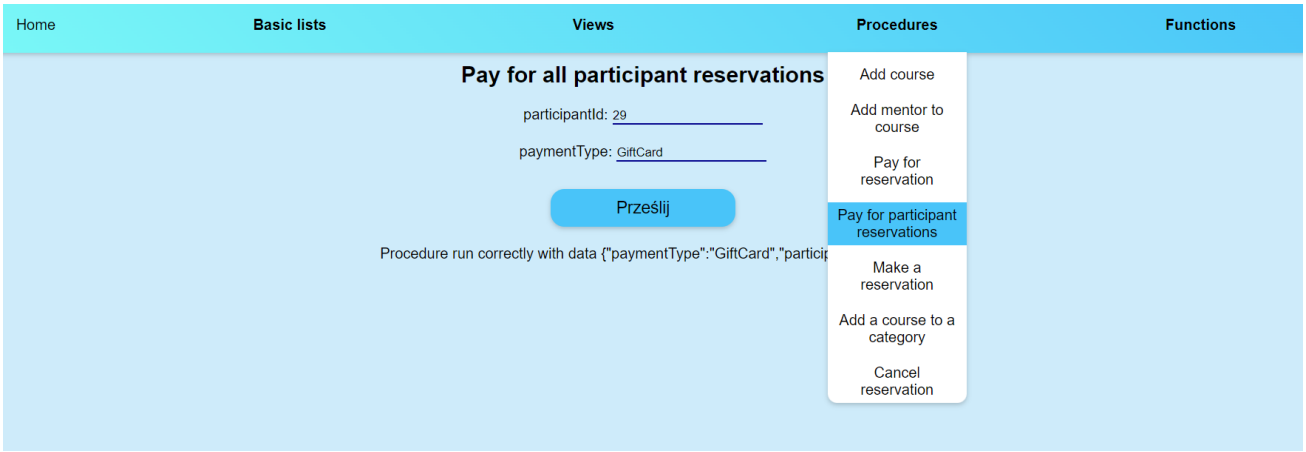
	ID 1	STATUS	COURSE_FK	PARTICIPANT_FK	CONNECTED_RESERVATION_FK
1	183	2	101	29	<null>
2	182	2	101	29	<null>
3	181	1	101	29	<null>

9. Sprawdzenie niezapłaconych rezerwacji dla użytkownika

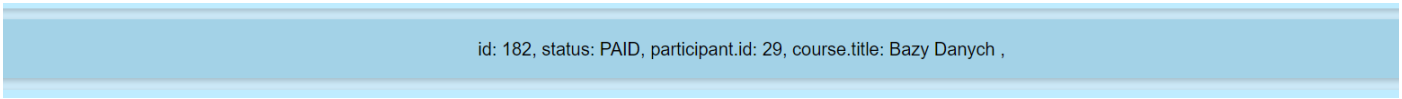
10. Policzenie sumy za niezapłacone rezerwacje

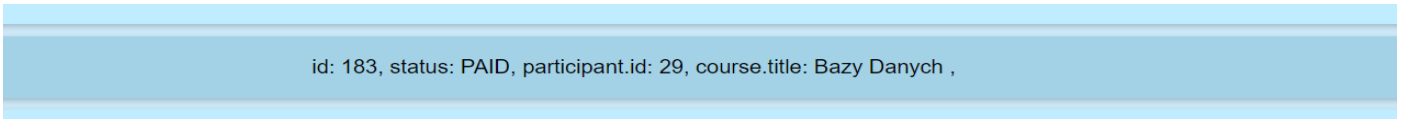


11. Zapłata za rezerwacje



Rezerwacje na stronie zmieniły status na PAID





Widzimy w bazie dokonane zmiany

	ID	STATUS	COURSE_FK	PARTICIPANT_FK	CONNECTED_RESERVATION_FK
1	183	0	101	29	142
2	182	0	101	29	141
3	181	1	101	29	<null>

12. Ilość aktualnych miejsc na kursie

	ID	TITLE	AVAILABLE_PLACES	START_DATE	END_DATE	MAX_NO_PLACES	PRICE	CATEGORY_FK
1	101	Bazy Danych	18	2023-06-15 02:00:00	2023-06-18 02:00:00	20	800.00	1

Wszystko działa :)