Booking Database

a hy360 Project

11/2024-12/2024

csd5087 - Κρητσωτάκης Μιχαήλ
csd4844 - Αρβανίτη Νικολέττα
csd4922 - Λαζαρίδης Νικόλαος

Περιεχόμενα

Εισαγωγή

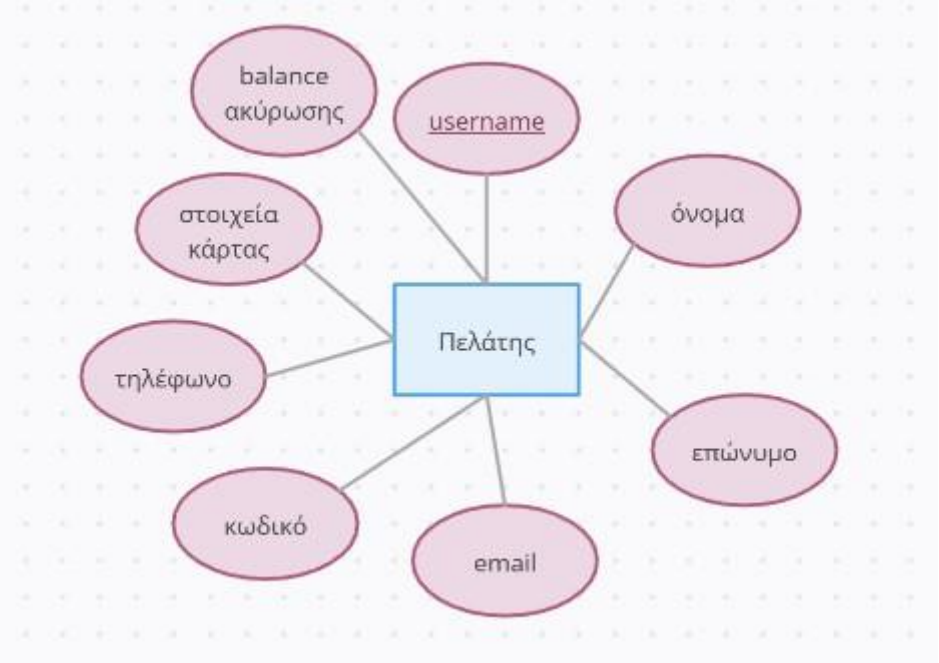
Το Project αυτό αποτελεί την υλοποίηση μιας βάσης δεδομένων για την διαχείριση κρατήσεων των θέσεων ενός Συναυλιακού Χώρου.

Η πλατφόρμα θα υποστηρίζει την Εγγραφή πελατών και τις Κρατήσεις/Ακυρώσεις εισιτηρίων για θέσεις σε διάφορες εκδηλώσεις, οι οποίες έχουν δικές τους Εγγραφές/Ακυρώσεις. Ο Πελάτης θα μπορεί να αναζητήσει διαθέσιμες Εκδηλώσεις σύμφωνα με την ημερομηνία, τον τύπο ή με το όνομα τους.

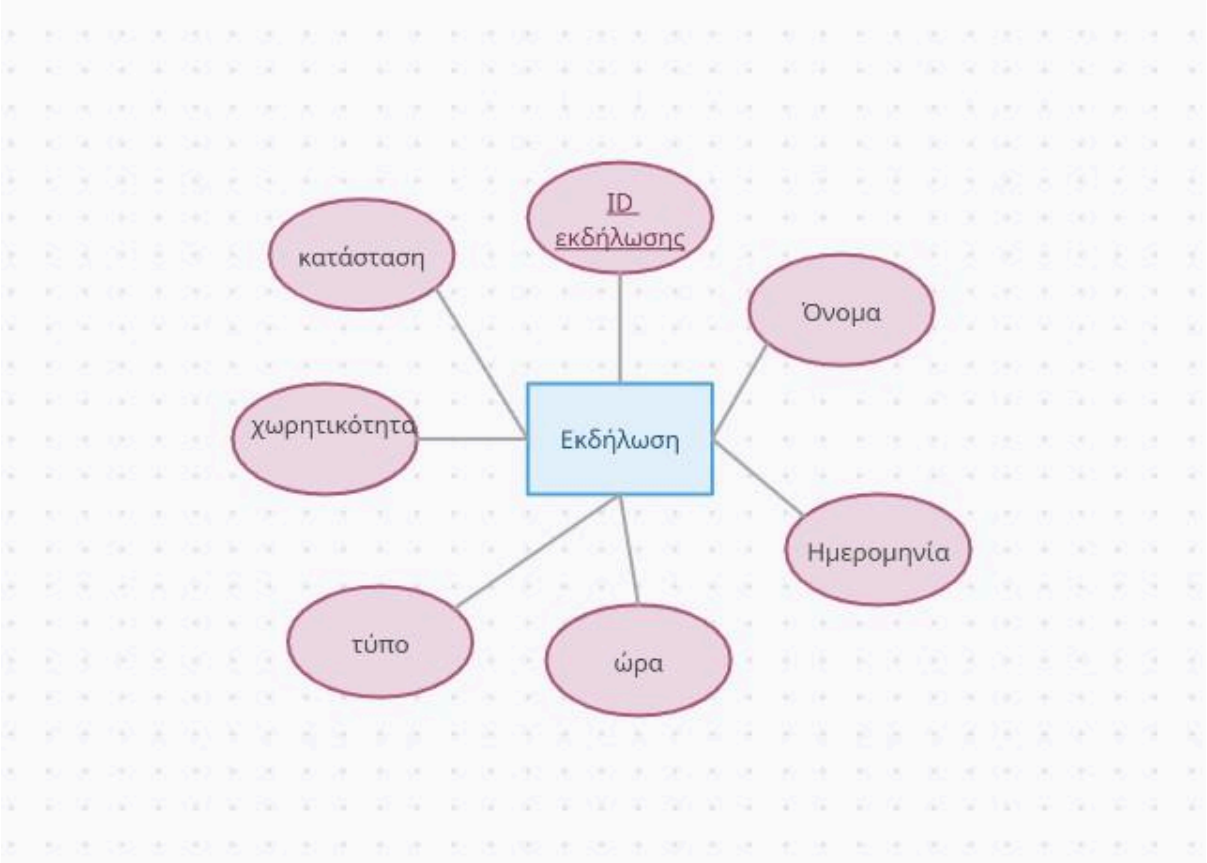
Η υλοποίηση θα γίνει σε γλώσσα Java, με χρήση τεχνολογιών Java Servlets μέσω Apache για επικοινωνία frontend με backend και MySQL για την βάση δεδομένων.

Διαγράμματα οντοτήτων-σχέσεων

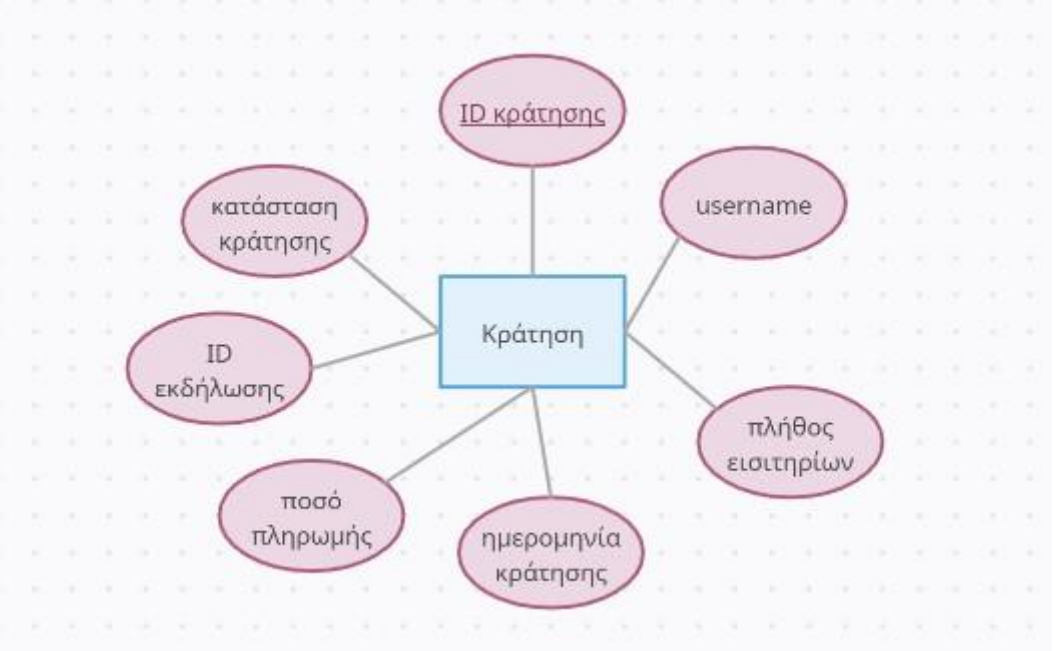
Πελάτης:



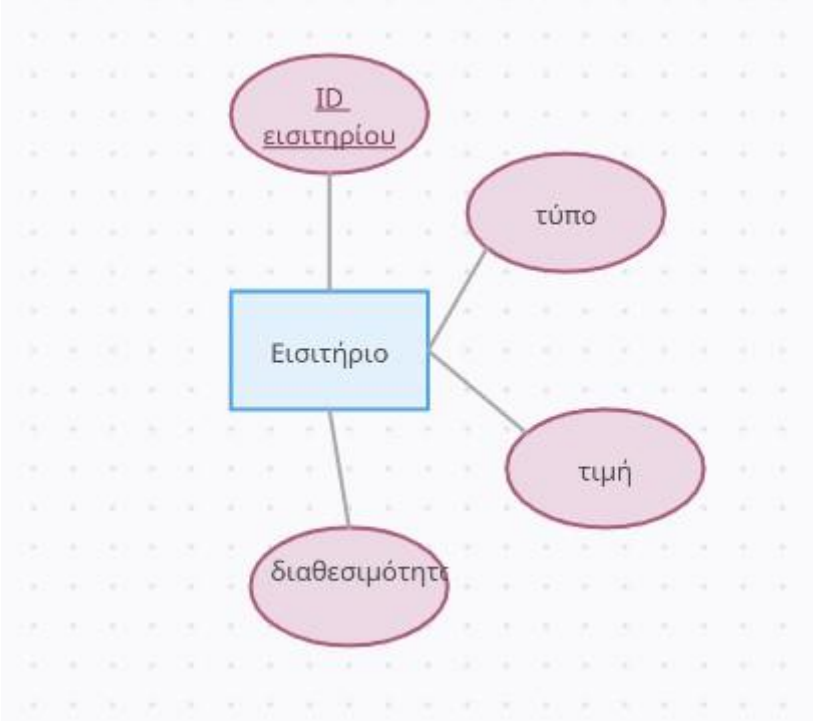
Εκδήλωση:



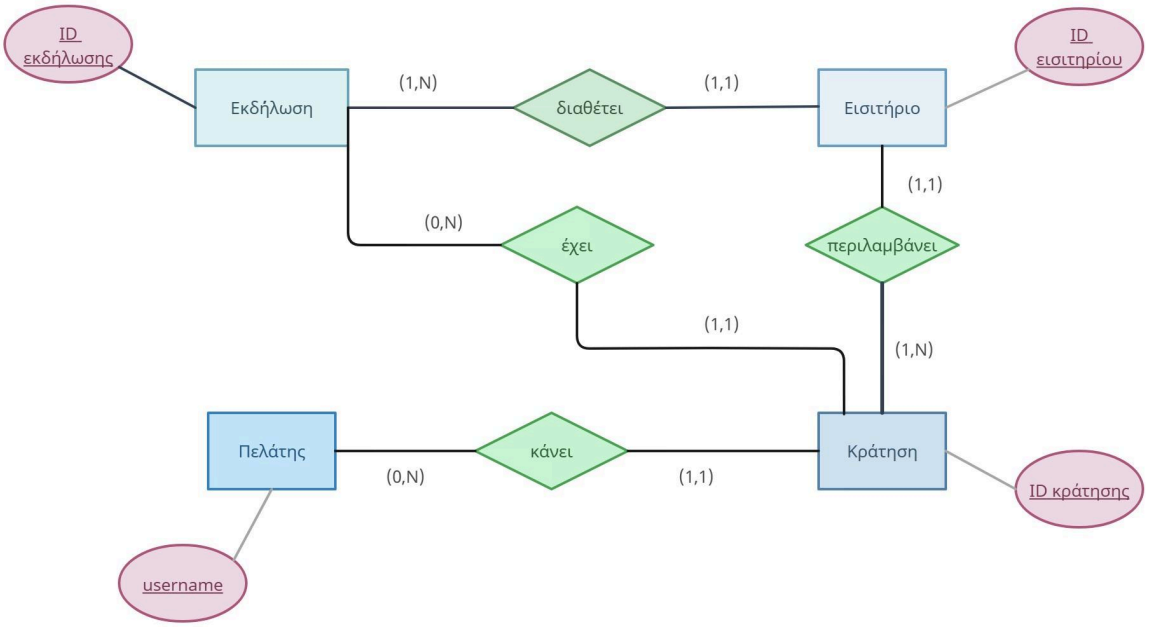
Κράτηση:



Εισιτήριο:



Ολοκληρωμένο ER διάγραμμα



Σχεσιακό Μοντέλο & 3η κανονική μορφή

Event:

EventID	Name	Date	Time	Type	Capacity	Status

Customer:

username	password	first name	last name	email	phone number	balance

Card number	Card ExpDate	Card CVV

Ticket:

Ticket ID	Type	Price	Availability	EventID	ReservationID

Booking:

Reservation ID	Status	Payment Amount	Date	Number of Tickets	Client Username	EventID

Tables

Παραδείγματα δημιουργίας πινάκων σε SQL

```sql
/* Client table */
CREATE TABLE clients (
    client_username VARCHAR(50) NOT NULL UNIQUE,
    client_password VARCHAR(50) NOT NULL,
    client_firstname VARCHAR(50) NOT NULL,
    client_lastname VARCHAR(50) NOT NULL,
    client_email VARCHAR(50) NOT NULL UNIQUE,
    client_phone INTEGER NOT NULL UNIQUE,
    client_balance INTEGER DEFAULT 0,
    card_number INTEGER,
    card_expdate DATE,
    card_cvv INTEGER,
    PRIMARY KEY (client_username)
)
```

```sql
/* Event table */
CREATE TABLE events (
    event_id INTEGER NOT NULL AUTO_INCREMENT,
    event_name VARCHAR(50) NOT NULL,
    event_date DATE NOT NULL,
    event_time TIME NOT NULL,
    event_type ENUM('CONCERT', 'PERFORMANCE', 'COMEDYNIGHT',
        'SPORTS', 'CONFERENCE', 'WORKSHOP') NOT NULL,
    event_capacity INTEGER NOT NULL,
    event_status ENUM('SCHEDULED', 'CANCELED', 'COMPLETED') NOT NULL,
    PRIMARY KEY (event_id)
)
```

```sql
/* Ticket table */
CREATE TABLE tickets (
    ticket_id INTEGER NOT NULL AUTO_INCREMENT,
    ticket_type ENUM('REGULAR', 'VIP', 'BALCONY') NOT NULL,
    ticket_price DECIMAL(10, 2) NOT NULL,
    ticket_availability INTEGER NOT NULL,
    event_id INTEGER NOT NULL,
    reservation_id INTEGER NULL,
    FOREIGN KEY (event_id) REFERENCES events(event_id),
    FOREIGN KEY (reservation_id) REFERENCES reservations(reservation_id),
    PRIMARY KEY (ticket_id)
)
```

```sql
/* Reservation table */
CREATE TABLE reservations (
    reservation_id INTEGER NOT NULL AUTO_INCREMENT,
    reservation_tickets INTEGER NOT NULL,
    reservation_date DATE NOT NULL,
    reservation_payment_amount INTEGER NOT NULL,
    reservation_status ENUM('ACTIVE', 'CANCELED', 'COMPLETE') NOT NULL,
    client_username VARCHAR(50) ,
    event_id INTEGER ,
    FOREIGN KEY (client_username) REFERENCES clients(client_username),
    FOREIGN KEY (event_id) REFERENCES events(event_id),
    PRIMARY KEY (reservation_id)
)
```

Tuples

Παραδείγματα δημιουργίας πλειάδων σε SQL

```
/* Create Client */
INSERT INTO clients (client_username, client_password, ...)
VALUES (name, password, ...)
```

```
/* Create Event */
INSERT INTO events (event_name, event_date, ...)
VALUES (name, date, ..)
/* STATUS initially SCHEDULED */
/* also create tickets equal to capacity */
INSERT INTO tickets(ticket_type, ticket_price, ...)
VALUES (type, price, ...)
/* Five VIP Tickets price 50 half Regular 10 the rest Balcony 20  */
/* event_id is equal to the id of the event created most recently */
/* availability is initially 1 */
/* reservation_id starts NULL  */
```

```
/* Create Reservation */
INSERT INTO reservations (reservation_tickets, reservation_date, ...)
VALUES (tickets, currentDate, ...)
/* event_id is selected by customer */
/* VIP, balcony, regular ticket count seperately */
/* IN A LOOP turn tickets unavailable and link to reservation */
SELECT ticket_id FROM Tickets
WHERE ticket_availability = 1
AND ticket_type = 'type' AND event_id = eventID
LIMIT 1
/* and for each */
UPDATE Tickets
SET ticket_availability = 0, reservation_id = resID
WHERE ticket_id = ticketID
```

Cancellations

Παραδείγματα ακύρωσης εκδηλώσεων & κρατήσεων σε SQL

```sql
/* Cancel Event given its eventID */
UPDATE events SET event_status='CANCELED' WHERE event_id=eventID
/* canculate client payments */
SELECT client_username,
SUM(reservation_payment_amount) AS total_payment_amount
FROM reservations
WHERE event_id = eventID AND reservation_status = 'ACTIVE'
GROUP BY client_username
/* in a loop return the money as credits */
SET client_balance = client_balance + total_payment_amount
WHERE client_username = username
/* and cancel his reservations */
UPDATE reservations SET reservation_status = 'CANCELED'
WHERE event_id = eventID AND client_username = username
/* finally delete the event's tickets */
DELETE FROM tickets WHERE event_id = eventID
```

```sql
/* Cancel Reservation given its reservationID and client_username */
UPDATE reservations SET reservation_status = 'CANCELED'
WHERE reservation_id = reservationID
/* return money with a small penalty */
UPDATE clients SET client_balance = client_balance + PaymentAmount*0.8
WHERE client_username = username
/* make tickets available */
UPDATE Tickets SET ticket_availability = 1, reservation_id=NULL
WHERE reservation_id = reservationID
```

SQL Questions

Παραδείγματα των ερωτήσεων σε SQL

```sql
/* Total Profits */
SELECT SUM(ticket_price) AS total_payment_amount
FROM tickets WHERE ticket_availability = 0

SELECT SUM(ticket_price) AS total_payment_amount
FROM tickets WHERE ticket_type = 'REGULAR' AND ticket_availability = 0

SELECT SUM(ticket_price) AS total_payment_amount
FROM tickets WHERE ticket_type = 'BALCONY' AND ticket_availability = 0

SELECT SUM(ticket_price) AS total_payment_amount
FROM tickets WHERE ticket_type = 'VIP' AND ticket_availability = 0

/* Profits per Event */
SELECT event_id, SUM(reservation_payment_amount) AS total_payment_amount
FROM reservations WHERE reservation_status = 'ACTIVE' GROUP BY event_id
```

```sql
/* Most Popular Event in time period */
SELECT event_id, COUNT(*) AS reservation_count
FROM reservations WHERE reservation_status = 'ACTIVE'
GROUP BY event_id ORDER BY reservation_count DESC LIMIT 1

/* Highest Event Profit in time period */
SELECT event_id, SUM(reservation_payment_amount) AS total_profit
FROM reservations
WHERE reservation_date BETWEEN 'startDate' AND 'endDate'
AND reservation_status = 'ACTIVE'
GROUP BY event_id ORDER BY total_profit DESC LIMIT 1
```

```sql
/* Reservations in time period */
SELECT reservation_id, client_username, event_id, reservation_date, reservation_payment_amount
FROM reservations WHERE reservation_date BETWEEN 'startDate' AND 'endDate'
```

User Manual

Εγγραφή Χρήστη

Αρχικά, εμφανίζεται μια φόρμα εγγραφής, μέσω της οποίας ο χρήστης μπορεί να δημιουργήσει λογαριασμό στο σύστημα. Είναι απαραίτητο να συμπληρωθούν όλα τα απαιτούμενα πεδία με τις σωστές πληροφορίες.

Σύνδεση Χρήστη

Μετά την ολοκλήρωση της εγγραφής, υπάρχει διαθέσιμη μια φόρμα σύνδεσης.

Ο χρήστης μπορεί:

- Να συνδεθεί στο λογαριασμό του.
- Να δει τις διαθέσιμες εκδηλώσεις με τις αντίστοιχες πληροφορίες.
- Να κλείσει εισιτήρια, ανάλογα με τον επιτρεπόμενο αριθμό.
- Να δει τις κρατήσεις που έχει πραγματοποιήσει.
- Να ακυρώσει κρατήσεις. Σημειώνεται ότι σε περίπτωση ακύρωσης,
  επιστρέφεται το 80% του ποσού που καταβλήθηκε.

Διαχείριση Εκδηλώσεων από τον Διαχειριστή (Admin)

Για τη δημιουργία και τη διαχείριση των εκδηλώσεων,
υπάρχει ένας λογαριασμός διαχειριστή
(username: admin - password: admin123)
Ο διαχειριστής μπορεί:

- Να δημιουργήσει τα απαραίτητα tables αν δεν υπάρχουν.
- Να βλέπει τα υπάρχοντα tables με όλες τις λεπτομέρειες.
- Να προσθέσει μια νέα εκδήλωση, εισάγοντας όλα τα απαιτούμενα στοιχεία. Η διαδικασία αυτή δημιουργεί επίσης τα κατάλληλα εισιτήρια.
- Να ακυρώσει μια εκδήλωση, επιστρέφοντας τα χρήματα των κρατήσεων στους πελάτες σε μορφή credits και διαγράφοντας τα αντίστοιχα εισιτήρια.

Στο τέλος της σελίδας του διαχειριστή υπάρχουν κουμπιά που εκτελούν προκαθορισμένα SQL ερωτήματα, όπως:

- Συνολικά έσοδα από την πώληση VIP, Balcony, Regular εισιτηρίων ή αθροιστικά.
- Έσοδα από πωλήσεις ανά εκδήλωση.
- Εύρεση της δημοφιλέστερης εκδήλωσης βάσει κρατήσεων σε ένα χρονικό διάστημα
- Εύρεση της εκδήλωσης με τα περισσότερα έσοδα σε ένα χρονικό διάστημα.
- Προβολή κρατήσεων ανά χρονική περίοδο.

Επιπλέον, υπάρχει ένα πεδίο εισαγωγής όπου μπορούν να εισαχθούν custom SQL ερωτήματα. Το σύστημα επιστρέφει την απάντηση που αντιστοιχεί στο κάθε ερώτημα.

Conclusions

Η εφαρμογή λειτουργεί πλήρως!

Υπάρχουν constraints για την ομαλή λειτουργία των διαδικασιών, την εξασφάλιση της ακεραιότητας των δεδομένων καθώς και για αποφυγή προβλημάτων που ίσως προκύψουν από λάθος χρήση από τον πελάτη.

Μοναδικός περιορισμός της υλοποίησης είναι η αδυναμία λειτουργίας της πλατφόρμας από πολλούς χρήστες ταυτόχρονα, καθώς η ταυτοποίηση γίνεται σε επίπεδο localhost.

Μαζί με το παρόν αρχείο επισυνάπτεται όλος ο κώδικας που υλοποιεί τις λειτουργίες που αναφέρθηκαν στην αναφορά.

Developers
csd5087 - Κρητσωτάκης Μιχαήλ
csd4844 - Αρβανίτη Νικολέττα
csd4922 - Λαζαρίδης Νικόλαος