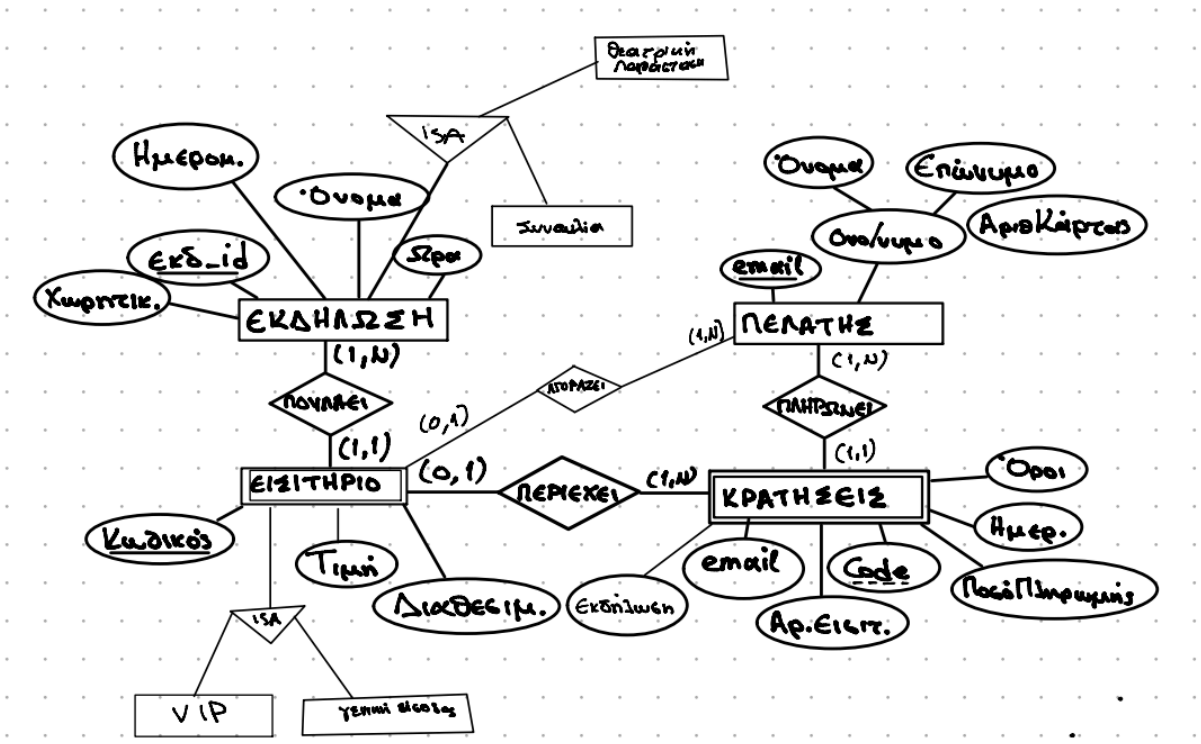
Project HY360

Ειρήνη Γρινιεζάκη csd5111

Αναστασία Συλβάνα Κακαρόντζα csd5196

Διάγραμμα οντοτήτων-σχέσεων για την εταιρία:



Μετάφραση του μοντέλου στο σχεσιακό μοντέλο:

Κράτηση ασθενής οντότητα υποθέτουμε ότι ενώ τα εισιτήρια υπάρχουν, η κράτηση αυτών πραγματοποιείται από τον πελάτη, άρα κράτηση χωρίς πελάτη δεν υπάρχει

Κράτηση

Code	email	εκδήλωση	Ημερ.	Αρ.Εισιτ	Ποσό	Όροι

Πελάτης

email	Όνομα	Επώνυμο	Αρ.Κάρτας

Εισιτήρια

VIP

κωδικός	τιμή	διαθεσιμότητα

γενική είσοδος

κωδικός	τιμή	διαθεσιμότητα

Εκδήλωση

Ζωνοπλία

εκδ_id	χωρ.	ημερ/νία	όνομα	ώρα

Θεατρική παράσταση

εκδ_id	χωρητικ.	ημερ/νία	όνομα	ώρα

VIP και γενική είσοδος υπολάβεις εισιτηρίων (ISA).

→ αναγνωριστικό πλειάδι ο κωδικός εισιτηρίου.

→ εισιτήρια ασθενής οντότητα, υποθέτουμε ότι αυτά δημιουργούνται κάθε φορά που υπάρχει μία εκδήλωση.

Πουλάει

εκδ_id	Κωδικός εισ.

→ Η εκδήλωση πουλάει 1 έως N εισιτήρια

Περιέχει

Code	Κωδικός εισ.

→ Η κάθε κράτηση περιέχει 1 έως N εισιτήρια

Πληρώνει

email	Code

→ Ο πελάτης μπορεί να πληρώσει 1 έως N κρατήσεις

Αγοράζει

email	κωδικός εισ.

→ Ο πελάτης αγοράζει 1 έως N εισιτ

Ζωνοπλία και θεατρική παράσταση υπολάβεις τις υλάσης εκδήλωση (ISA).

→ εκδ_id αναγνωριστικό πλειάδι για κάθε εκδήλωση (καθώς οι εκδηλώσεις μπορεί να έχουν το ίδιο όνομα μεταξύ τους με διαφορετική χρονολογία.

7)

```
CREATE TABLE Reservations (
    code        INT PRIMARY KEY,
    email       VARCHAR(100),
    event_id    VARCHAR(100),
    date        VARCHAR(20),
    ticket_num  INT,
    price       DOUBLE,
    terms       VARCHAR(200));
```

```
CREATE TABLE Tickets (
    ticket_num   INT PRIMARY KEY,
    ticket_type  ENUM (VIP, regular),
    price        INT,
    Availability BOOL);
```

```
CREATE TABLE Customers (
    email       VARCHAR(100) PRIMARY KEY,
    firstname   VARCHAR(100),
    lastname    VARCHAR(100),
    card_num    INT);
```

8) Κάποιοι περιορισμοί ακεραιότητας:

CHECK (price > 0)

NOT NULL event

UNIQUE ticket_num

(παραδείγματα πιο αναλυτικά στον κώδικα και στο σχολιό του)

```
CREATE TABLE Events (
    event_id    INT PRIMARY KEY,
    event_name  VARCHR(100),
    event_type  (concert, theater),
    capacity    INT,
    date        INT
    name        VARCHAR(100),
    time        VARCHAR(6));
```


Ερωτήματα SQL:

• Κατάσταση διαθέσιμων και κρατημένων θέσεων ανά εκδήλωση

```
String sql = "SELECT e.EventName, " +
             "SUM(t.Availability) AS Available, " +
             "SUM(t.InitialAvailability - t.Availability) AS Reserved " +
             "FROM Events e " +
             "JOIN Tickets t ON e.EventID = t.EventID " +
             "GROUP BY e.EventName";
```

• Έσοδα από πωλήσεις ανά εκδήλωση

```
String sql = "SELECT e.EventName, SUM(b.NumberOfTickets * t.Price) AS TotalRevenue " +
             "FROM Bookings b " +
             "JOIN Tickets t ON b.TicketID = t.TicketID " +
             "JOIN Events e ON b.EventID = e.EventID " +
             "GROUP BY e.EventName";
```

• Δημοφιλέστερη εκδήλωση βάσει κρατήσεων

```
String sql = "SELECT e.EventName, COUNT(b.BookingID) AS Bookings " +
             "FROM Bookings b " +
             "JOIN Events e ON b.EventID = e.EventID " +
             "GROUP BY e.EventName " +
             "ORDER BY Bookings DESC LIMIT 1";
```

• Εκδήλωση με τα περισσότερα έσοδα σε ένα χρονικό εύρος

```
String sql = "SELECT e.EventName, SUM(b.NumberOfTickets * t.Price) AS TotalRevenue " +
             "FROM Bookings b " +
             "JOIN Tickets t ON b.TicketID = t.TicketID " +
             "JOIN Events e ON b.EventID = e.EventID " +
             "WHERE b.BookingDate BETWEEN ? AND ? " +
             "GROUP BY e.EventName " +
             "ORDER BY TotalRevenue DESC LIMIT 1";
```

• Προβολή κρατήσεων ανά χρονική περίοδο

```
String sql = "SELECT b.BookingID, e.EventName, b.NumberOfTickets, b.BookingDate " +
             "FROM Bookings b " +
             "JOIN Events e ON b.EventID = e.EventID " +
             "WHERE b.BookingDate BETWEEN ? AND ?";
```

• Τα συνολικά έσοδα από την πώληση VIP ή γενικών εισιτηρίων ανά εκδήλωση ή συνολικά;

```
String sql = "SELECT e.EventName, SUM(b.NumberOfTickets * t.Price) AS Revenue " +
             "FROM Bookings b " +
             "JOIN Tickets t ON b.TicketID = t.TicketID " +
             "JOIN Events e ON b.EventID = e.EventID " +
             "WHERE t.TicketType = ? " +
             "GROUP BY e.EventName";
```

Περιγραφή σε ψευδοκώδικα των διαδικασιών:

• Εγγραφή νέου πελάτη: Καταχώριση των απαραίτητων στοιχείων πελάτη

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        String fullName = request.getParameter("fullName");
        String email = request.getParameter("email");
        String creditCardInfo = request.getParameter("creditCardInfo");
        String Password = request.getParameter("password");

        try (Connection conn = DB_Connection.getConnection()) {
            String sql = "INSERT INTO Customers (FullName, Email, CreditCardInfo, Password) VALUES (?, ?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, fullName);
            pstmt.setString(2, email);
            pstmt.setString(3, creditCardInfo);
            pstmt.setString(4, Password);
            pstmt.executeUpdate();

            response.getWriter().println("Customer added successfully!");
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("Error: " + e.getMessage());
        }
    }
```

• Αναζήτηση διαθέσιμων θέσεων: Προβολή διαθέσιμων εισιτηρίων ανά εκδήλωση και τύπο θέσης

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    int eventId = Integer.parseInt(request.getParameter("eventId"));

    try (Connection conn = DB_Connection.getConnection()) {
        String sqlCheckEvent = "SELECT * FROM Events WHERE EventID = ?";
            PreparedStatement pstmtCheckEvent = conn.prepareStatement(sqlCheckEvent);
            pstmtCheckEvent.setInt(1, eventId);
            ResultSet rsEvent = pstmtCheckEvent.executeQuery();

            if (!rsEvent.next()) {
                // Εάν δεν υπάρχει το EventID, επιστρέφουμε μήνυμα σφάλματος
                response.setContentType("text/html");
                response.getWriter().println("<h3>Error: Event ID " + eventId + " does not exist.</h3>");
                return;
            }

Ερωτήμα για τα διαθέσιμα εισιτήρια
        String sql = "SELECT TicketID, TicketType, Availability FROM Tickets WHERE EventID = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, eventId);
        ResultSet rs = pstmt.executeQuery();

        // Δημιουργούμε HTML απάντηση
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h3>Available Tickets for Event ID: " + eventId + "</h3>");
        out.println("<table border='1'><tr><th>Ticket ID</th><th>Type</th><th>Availability</th></tr>");

        while (rs.next()) {
            out.println("<tr>");
            out.println("<td>" + rs.getInt("TicketID") + "</td>");
            out.println("<td>" + rs.getString("TicketType") + "</td>");
            out.println("<td>" + rs.getInt("Availability") + "</td>");
            out.println("</tr>");
        }

        out.println("</table>");
    } catch (Exception e) {
        e.printStackTrace();
        response.setContentType("text/html");
        response.getWriter().println("<h3>Error: " + e.getMessage() + "</h3>");
    }
}
```

• Κράτηση εισιτηρίων: Επιλογή από τα διαθέσιμα εισιτήρια, καταχώριση κράτησης και ολοκλήρωση πληρωμής

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    int eventId = Integer.parseInt(request.getParameter("eventId"));

    try (Connection conn = DB_Connection.getConnection()) {
        // Check if the event exists
        String sqlCheckEvent = "SELECT EventName FROM Events WHERE EventID = ?";
        PreparedStatement pstmtCheckEvent = conn.prepareStatement(sqlCheckEvent);
        pstmtCheckEvent.setInt(1, eventId);
        ResultSet rsEvent = pstmtCheckEvent.executeQuery();

        if (!rsEvent.next()) {
            // Event does not exist
            response.setContentType("text/html");
            response.getWriter().println("<h3>Error: Event ID " + eventId + " does not exist.</h3>");
            return;
        }

        String eventName = rsEvent.getString("EventName");

        // Query available tickets
        String sql = "SELECT TicketID, TicketType, Price, Availability FROM Tickets WHERE EventID = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, eventId);
        ResultSet rs = pstmt.executeQuery();

        // Build HTML response
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Available Tickets</title>");
```

```java
out.println("<style>");
out.println("body { font-family: Arial, sans-serif; background-color: #f4f4f9; color: #333; }");
out.println("h3 { color: #4CAF50; text-align: center; }");
out.println("table { width: 80%; margin: 20px auto; border-collapse: collapse; box-shadow: 0 2px 5px rgba(0,0,0,0.1); }");
out.println("table th, table td { padding: 10px; text-align: left; border: 1px solid #ddd; }");
out.println("table th { background-color: #4CAF50; color: white; }");
out.println("table tr:nth-child(even) { background-color: #f9f9f9; }");
out.println("table tr:hover { background-color: #f1f1f1; }");
out.println("</style></head><body>");

out.println("<h3>Available Tickets for Event: " + eventName + "</h3>");
out.println("<table>");
out.println("<tr><th>Ticket ID</th><th>Type</th><th>Price</th><th>Availability</th></tr>");

boolean hasTickets = false; // Check if tickets exist for the event
while (rs.next()) {
    hasTickets = true;
    out.println("<tr>");
    out.println("<td>" + rs.getInt("TicketID") + "</td>");
    out.println("<td>" + rs.getString("TicketType") + "</td>");
    out.println("<td>$" + rs.getDouble("Price") + "</td>");
    out.println("<td>" + rs.getInt("Availability") + "</td>");
    out.println("</tr>");
}

if (!hasTickets) {
    out.println("<tr><td colspan='4' style='text-align:center;'>No tickets available for this event.</td></tr>");
}

out.println("</table>");
```

```java
out.println("</body></html>");
tch (Exception e) {
e.printStackTrace();
response.setContentType("text/html");
response.getWriter().println("<h3>Error: " + e.getMessage() + "</h3>");
```

• Ακύρωση κράτησης: Διαγραφή της κράτησης, με όρους επιστροφής χρημάτων ή χρέωση ακύρωσης

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    int bookingId = Integer.parseInt(request.getParameter("bookingId"));
    int numberOfTickets = Integer.parseInt(request.getParameter("numberOfTickets"));
    int ticketId = Integer.parseInt(request.getParameter("ticketId"));

    Connection conn = null;

    try {
        conn = DB_Connection.getConnection();
        conn.setAutoCommit(false); // Start transaction

        // Step 1: Delete the booking
        String deleteBookingSql = "DELETE FROM Bookings WHERE BookingID = ?";
        PreparedStatement pstmtDelete = conn.prepareStatement(deleteBookingSql);
        pstmtDelete.setInt(1, bookingId);
        pstmtDelete.executeUpdate();

        // Step 2: Update ticket availability
        String updateAvailabilitySql = "UPDATE Tickets SET Availability = Availability + ? WHERE TicketID = ?";
        PreparedStatement pstmtUpdate = conn.prepareStatement(updateAvailabilitySql);
        pstmtUpdate.setInt(1, numberOfTickets);
        pstmtUpdate.setInt(2, ticketId);
        pstmtUpdate.executeUpdate();

        conn.commit(); // Commit transaction

        // Success response
        response.setContentType("application/json");
        response.getWriter().println("{\"message\":\"Booking cancelled successfully.\"}");
    } catch (Exception e) {
        e.printStackTrace();
        if (conn != null) {
            try {
                conn.rollback(); // Rollback on error
            } catch (SQLException ex) {
                Logger.getLogger(CancelBooking.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        response.setContentType("application/json");
        response.getWriter().println("{\"error\":\"" + e.getMessage() + "\"}");
    } finally {
        if (conn != null) {
            try {
                conn.close(); // Close connection
            } catch (SQLException ex) {
                Logger.getLogger(CancelBooking.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

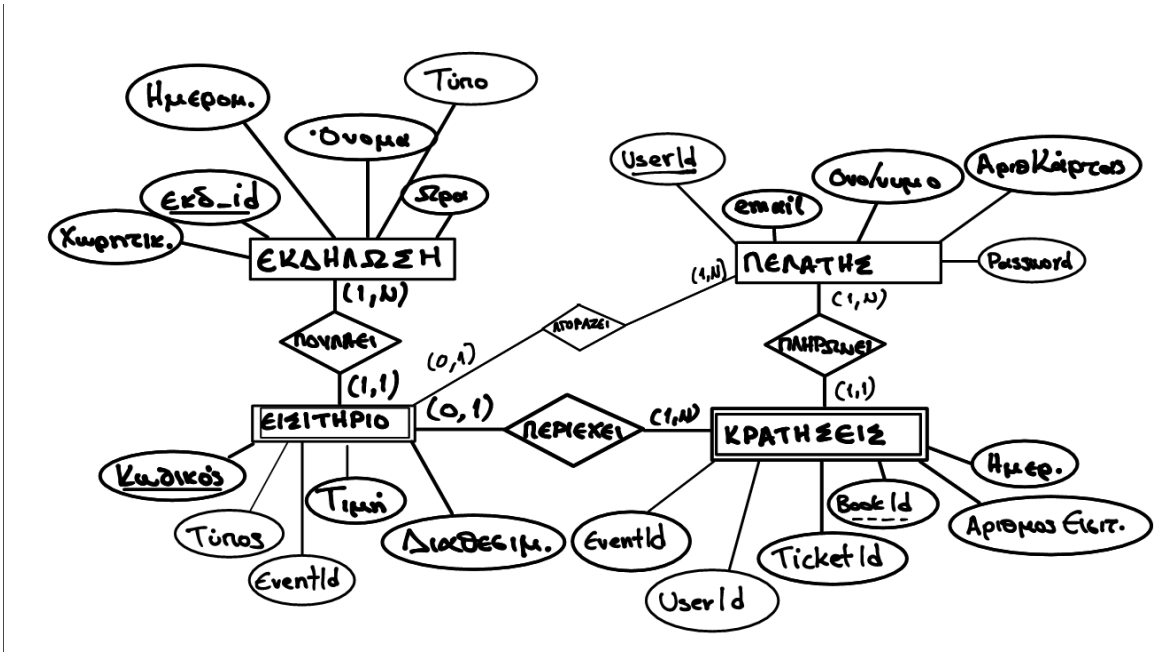
• Ακύρωση εκδήλωσης: Θα πρέπει να επιστρέφονται τα χρήματα στους πελάτες.

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String eventName = request.getParameter("eventName");

    if (eventName == null || eventName.trim().isEmpty()) {
        response.setContentType("application/json");
        response.getWriter().write("{\"error\": \"Event name is required.\"}");
        return;
    }

    Connection conn = null;

    try {
        conn = DB_Connection.getConnection();
        conn.setAutoCommit(false); // Begin transaction

        // Step 1: Get EventID from EventName
        String sqlGetEventID = "SELECT EventID FROM Events WHERE EventName = ?";
        PreparedStatement pstmtGetEventID = conn.prepareStatement(sqlGetEventID);
        pstmtGetEventID.setString(1, eventName);
        ResultSet rsEvent = pstmtGetEventID.executeQuery();

        if (!rsEvent.next()) {
            response.setContentType("application/json");
            response.getWriter().write("{\"error\": \"Event does not exist.\"}");
            return;
        }

        int eventId = rsEvent.getInt("EventID");
```

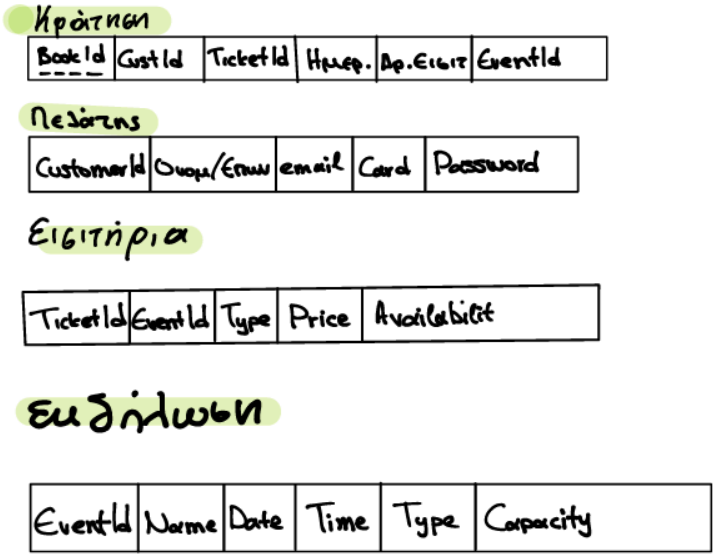
```java
// Step 2: Refund customers
String sqlRefunds = "SELECT CustomerID, SUM(b.NumberOfTickets * t.Price) AS RefundAmount " +
                    "FROM Bookings b " +
                    "JOIN Tickets t ON b.TicketID = t.TicketID " +
                    "WHERE b.EventID = ? GROUP BY CustomerID";
PreparedStatement pstmtRefunds = conn.prepareStatement(sqlRefunds);
pstmtRefunds.setInt(1, eventId);
ResultSet rsRefunds = pstmtRefunds.executeQuery();

while (rsRefunds.next()) {
    int customerId = rsRefunds.getInt("CustomerID");
    double refundAmount = rsRefunds.getDouble("RefundAmount");
    System.out.println("Refunding customer " + customerId + ": $" + refundAmount);
    // Add logic for processing refunds if necessary
}

// Step 3: Delete from Bookings table
String sqlDeleteBookings = "DELETE FROM Bookings WHERE EventID = ?";
PreparedStatement pstmtDeleteBookings = conn.prepareStatement(sqlDeleteBookings);
pstmtDeleteBookings.setInt(1, eventId);
pstmtDeleteBookings.executeUpdate();

// Step 4: Delete from Tickets table
String sqlDeleteTickets = "DELETE FROM Tickets WHERE EventID = ?";
PreparedStatement pstmtDeleteTickets = conn.prepareStatement(sqlDeleteTickets);
pstmtDeleteTickets.setInt(1, eventId);
pstmtDeleteTickets.executeUpdate();

// Step 5: Delete from Events table
String sqlDeleteEvent = "DELETE FROM Events WHERE EventID = ?";
```

```java
        PreparedStatement pstmtDeleteEvent = conn.prepareStatement(sqlDeleteEvent);
        pstmtDeleteEvent.setInt(1, eventId);
        pstmtDeleteEvent.executeUpdate();

        conn.commit(); // Commit transaction

        response.setContentType("application/json");
        response.getWriter().println("{\"message\":\"Event canceled and refunds issued successfully.\"}");
    } catch (Exception e) {
        if (conn != null) {
            try {
                conn.rollback(); // Rollback on error
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
        e.printStackTrace();
        response.setContentType("application/json");
        response.getWriter().println("{\"error\":\"" + e.getMessage() + "\"}");
    } finally {
        if (conn != null) {
            try {
                conn.close(); // Close connection
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Διάγραμμα οντοτήτων-σχέσεων για την εταιρία μετά τη Β φάση:



- Τα εισιτήρια πλέον δεν χωρίζονται σε δύο ξεχωριστά πινακάκια, υπάρχει αναγνωριστικό τύπος (VIP or General)
-
- Αντοίστοιχα οι εκδηλώσεις έχουν τύπο ως αναγνωριστικό



Κάποια ενδεικτικά αποτελέσματα από την εκτέλεση των διαδικασιών:

events:

	EventID	EventName	EventDate	EventTime	EventType	Capacity
Επεξεργασία Αντιγραφή Διαγραφή	1	new year's eve	2024-12-31	23:00:00	party	40
Επεξεργασία Αντιγραφή Διαγραφή	2	sakis rouvas	2024-12-15	20:00:00	concert	215
Επεξεργασία Αντιγραφή Διαγραφή	3	christmas party	2024-12-24	22:07:00	party	75
Επεξεργασία Αντιγραφή Διαγραφή	4	efi thodi	2024-12-18	21:08:00	concert	175

users:

	CustomerID	FullName	Email	CreditCardInfo	Password
Επεξεργασία Αντιγραφή Διαγραφή	1	Î□Î½Î±Î□Î±Î□Î¯Î± Î Î±Î□Î±Î¿Î□Î»Î·	anasia@gmail.com	1234567890087654	Arizona1@
Επεξεργασία Αντιγραφή Διαγραφή	2	Î□Î±Î□Î¯Î± Î Î±Î□Î±Î¿Î□Î»Î·	maria@gmail.com	1234567890087654	Arizona1@
Επεξεργασία Αντιγραφή Διαγραφή	3	Î□Î¹Î¬Î½Î½Î·Î□ Î Î□Î¿	john@gmail.com	1234567890087654	Arizona1@

Tickets:

←T→			TicketID	EventID	TicketType	Price	Availability
Επεξεργασία	Αντιγραφή	Διαγραφή	1	1	General	10.00	17
Επεξεργασία	Αντιγραφή	Διαγραφή	2	1	VIP	20.00	23
Επεξεργασία	Αντιγραφή	Διαγραφή	3	2	General	10.00	200
Επεξεργασία	Αντιγραφή	Διαγραφή	4	2	VIP	20.00	15
Επεξεργασία	Αντιγραφή	Διαγραφή	5	3	General	8.00	60
Επεξεργασία	Αντιγραφή	Διαγραφή	6	3	VIP	30.00	15
Επεξεργασία	Αντιγραφή	Διαγραφή	7	4	General	9.00	150
Επεξεργασία	Αντιγραφή	Διαγραφή	8	4	VIP	20.00	25