Βάσεις Δεδομένων

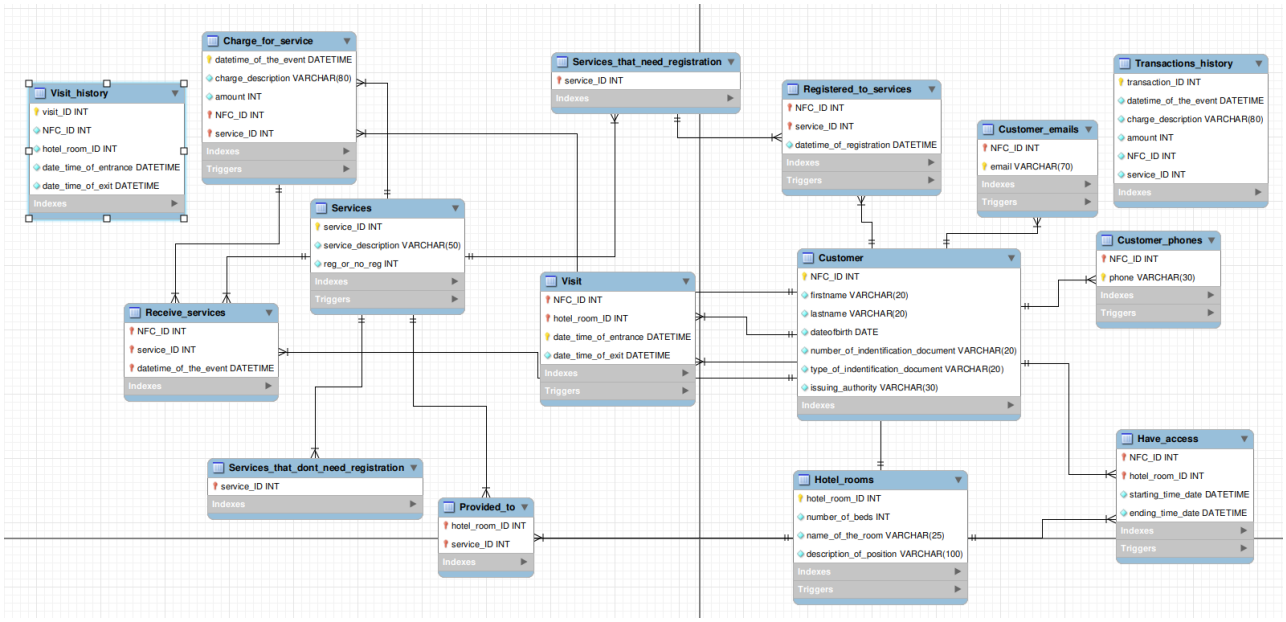
Αναφορά Εξαμηνιαίας Εργασίας

Ομάδα: AdditionalgroupA

Αναστασάκης Ζαχαρίας: 03116675
Αντωνίου Κωνσταντίνος: 03116169
Μετζάκης Ιωάννης: 03116202

1. ΣΧΕΣΙΑΚΟ ΔΙΑΓΡΑΜΜΑ

Το σχεσιακό διάγραμμα της εφαρμογής μας φαίνεται στην παρακάτω φωτογραφία:



Το παραπάνω διάγραμμα κατασκευάστηκε από το tables.sql με Reverse Engineer μέσω του εργαλείου MySQL-Workbench, το οποίο εγκαταστήσαμε σε περιβάλλον Linux. Συγκεκριμένα, ορίζουμε την βάση μας στο MySQL-Workbench και τρέχουμε το αρχείο tables.sql. Έπειτα, μέσω των επιλογών Database < Reverse Engineering παράγεται το παραπάνω διάγραμμα.

1.a. Περιορισμοί

Όσον αφορά τους περιορισμούς που έχουν οριστεί έχουμε τους εξής ανά κατηγορία:

- Σχεδιαστικοί περιορισμοί κατά τη σχεδίαση του ER:

Το ER που επιλέξαμε μας δόθηκε έτοιμο (απλή λύση) και γι' αυτό τον λόγο δεν έχουμε ορίσει δικούς μας σχεδιαστικούς περιορισμούς. Οι διαφορές μεταξύ του δωσμένου ER και αυτού που εν τέλει υλοποιήσαμε είναι πολύ μικρές, συνεπώς δεν προέκυψαν καινούργιοι.

- Περιορισμοί ακεραιότητας και επεξήγηση των πεδίων των πινάκων:

Αρχικά, κατά τη δημιουργία κάθε attribute, ανεξαρτήτως του τύπου του, το ορίσαμε ως NOT NULL με σκοπό η πληροφορία που αποθηκεύουμε στην βάση μας να είναι όσο πιο συνεπής γίνεται για κάθε πίνακα. Έπειτα, επειδή τα στοιχεία κάποιων πινάκων δεν μπορούν να προσδιοριστούν μοναδικά χρησιμοποιώντας μόνο ένα attribute, ορίσαμε για τον προσδιορισμό των στοιχείων αυτών complex primary keys, τα οποία θα αναλυθούν

παρακάτω σε αυτή την ενότητα. Κατά αυτόν τον τρόπο, κάθε primary key ή complex primary key είναι μοναδικό. Συγκεκριμένα, στην περίπτωση των complex primary keys υπάρχει μοναδικός συνδιασμός, n-αδα, όλων των attributes που είναι partial keys. Επίσης, όσον αφορά τα foreign keys, έχουν οριστεί με τον περιορισμό ON DELETE CASCADE έτσι ώστε να μπορούμε να διαγράψουμε στοιχεία από τους πίνακες γονείς. Χρησιμοποιώντας αυτόν τον περιορισμό όταν γίνεται μία διαγραφή από τον πίνακα γονέα, διαγράφονται αυτόματα όλες οι εγγραφές σε πίνακες παιδιά, οι οποίες συνδέονται μέσω foreign key με τα primary keys των δεδομένων που διαγράφηκαν. Αυτός ο περιορισμός είναι πιθανό να μας κάνει να χάσουμε πληροφορία την οποία χρειαζόμαστε, πράγμα το οποίο το αντιμετωπίζουμε δημιουργώντας πίνακες οι οποίοι κρατάνε ιστορικό, και ανανεώνονται μέσω triggers (θα γίνει εκτενέστερη ανάλυση στην αντίστοιχη ενότητα).

Παρακάτω θα παρουσιάσουμε για κάθε πίνακα τα primary και foreign keys του, καθώς και όλους τους τύπους των υπόλοιπων attributes του πίνακα:

- **Customer**

Ο πίνακας Customer έχει τις ακόλουθες στήλες:

- NFC_ID : int primary key. Έχει οριστεί με την μέθοδο AUTO_INCREMENT, ώστε να διασφαλίζεται σε κάθε περίπτωση η μοναδικότητα του.
- firstname : varchar(20)
- lastname : varchar(20)
- dateofbirth : date
- number_of_indentification_document : varchar(20)
- type_of_indentification_document : varchar(20)
- issuing_authority : varchar(30)

- **Customer_emails**

Επειδή κάθε πελάτης μπορεί να έχει παραπάνω από ένα email, ορίζουμε πίνακα Customer_emails ο οποίος έχει complex primary key που αποτελείται από το email και το NFC_ID. Ο πίνακας έχει τις ακόλουθες στήλες:

- NFC_ID : int foreign key από τον πίνακα Customer
- email : varchar(70)

- **Customer_phones**

Επειδή κάθε πελάτης μπορεί να έχει παραπάνω από ένα τηλέφωνο, ορίζουμε πίνακα Customer_phones, ο οποίος έχει complex primary key που αποτελείται από το phone και το NFC_ID. Ο πίνακας έχει τις ακόλουθες στήλες:

- NFC_ID : int foreign key από τον πίνακα Customer
- phone: varchar(30)

- **Services**

Σε αυτόν τον πίνακα προσθέσαμε ένα παραπάνω attribute, σε σχέση με το ER που μας δόθηκε, το οποίο μας δείχνει αν η υπηρεσία που περιγράφουμε απαιτεί εγγραφή ή όχι. Επίσης, επιλέξαμε το δωμάτιο να είναι μία “pseudo” υπηρεσία με μοναδικό service_ID = 0. Οι στήλες του πίνακα είναι οι εξής:

- service_ID : int primary key. Είναι μηδέν μόνο για την υπηρεσία του δωματίου. Για όλες τις υπόλοιπες παίρνει μία τιμή μεγαλύτερη του μηδενός.
- service_description : varchar(50)
- reg_or_no_reg : int. 0 αν απαιτείται εγγραφή, 1 αν όχι (ελέγχεται με triggers)

- **Services_that_need_registration**

Αυτός ο πίνακας παίρνει ως foreign key το service_ID των υπηρεσιών που χρειάζονται εγγραφή (οι τιμές μπαίνουν με triggers). Οι στήλες του είναι οι εξής:

- service_ID : int foreign key από τον πίνακα Service

- **Services_that_dont_need_registration**

Αυτός ο πίνακας παίρνει ως foreign key το service_ID των υπηρεσιών που δεν χρειάζονται εγγραφή (οι τιμές μπαίνουν με triggers). Οι στήλες του είναι οι εξής:

- service_ID : int foreign key από τον πίνακα Service

- **Hotel_rooms**

Ο πίνακας Hotel_rooms περιέχει όλους τους χώρους του ξενοδοχείου. Έχει γίνει η παραδοχή ότι τα δωμάτια έχουν hotel_room_ID μικρότερο του 100 (ελέγχεται με triggers), ενώ όλοι οι υπόλοιποι χώροι hotel_room_ID >= 100. Οι στήλες του πίνακα είναι:

- hotel_room_ID : int primary key
- number_of_beds : int. Αν ο αριθμός των κρεβατιών ενός χώρου είναι μηδέν τότε γνωρίζουμε ότι ο χώρος αυτός δεν είναι δωμάτιο.
- name_of_the_room : varchar(25)
- description_of_position : varchar(100)

- **Have_access**

Ο πίνακας Have_access έχει complex primary key που αποτελείται από το NFC_ID που προσδιορίζει τον Customer και από το hotel_room_ID που προσδιορίζει τον χώρο του ξενοδοχείου. Οι στήλες του πίνακα είναι οι εξής:

- NFC_ID : int foreign key από τον πίνακα Customer
- hotel_room_ID : int foreign key από τον πίνακα Hotel_rooms
- starting_time_date : datetime
- ending_time_date : datetime

- **Visit**

Ο πίνακας Visit έχει complex primary key που αποτελείται από το NFC_ID που προσδιορίζει τον Customer, από το hotel_room_ID που προσδιορίζει τον χώρο του ξενοδοχείου και από το date_time_of_entrance, το οποίο είναι το primary key του συγκεκριμένου πίνακα. Η προσθήκη του date_time_of_entrance ήταν αναγκαία, καθώς στην περίπτωση που κάποιος πελάτης επισκεφτεί τον ίδιο χώρο παραπάνω από μία φορές, χρειαζόμαστε και άλλο ένα στοιχείο για να εξασφαλίσουμε την μοναδικότητα την κάθε επίσκεψης. Το στοιχείο αυτό είναι το date_time_of_entrance. Οι στήλες του πίνακα είναι οι εξής:

- NFC_ID : int foreign key από τον πίνακα Customer
- hotel_room_ID : int foreign key από τον πίνακα Hotel_rooms
- date_time_of_entrance : datetime primary key
- date_time_of_exit : datetime

- **Registered_to_services**

Ο πίνακας Registered_to_services έχει complex primary key που αποτελείται από το NFC_ID που προσδιορίζει τον Customer και από το service_ID που προσδιορίζει την υπηρεσία. Οι στήλες του πίνακα είναι οι εξής:

- NFC_ID : int foreign key από τον πίνακα Customer
- service_ID : int foreign key από τον πίνακα Services
- datetime_of_registration : datetime

- **Provided_to**

Ο πίνακας Provided_to έχει complex primary key που αποτελείται από το hotel_room_ID που προσδιορίζει τον χώρο του ξενοδοχείου και από το service_ID που προσδιορίζει την υπηρεσία. Οι στήλες του πίνακα είναι οι εξής:

- hotel_room_ID : int foreign key από τον πίνακα Hotel_rooms
- service_ID : int foreign key από τον πίνακα Services

- **Charge_for_service**

Ο παραπάνω πίνακας έχει complex primary key το οποίο αποτελείται από το NFC_ID που προσδιορίζει τον Customer, από το service_ID το οποίο προσδιορίζει την υπηρεσία και από το primary key της οντότητας που είναι το datetime_of_the_event. Οι στήλες του πίνακα είναι οι εξής:

- datetime_of_the_event : datetime primary key
- charge_description : varchar(80)
- amount : int
- NFC_ID : int foreign key από τον πίνακα Customer
- service_ID : int foreign key από τον πίνακα Services

- **Receive_services**

Ο παραπάνω πίνακας έχει complex primary key το οποίο αποτελείται από το NFC_ID που προσδιορίζει τον Customer, από το service_ID, το οποίο προσδιορίζει την υπηρεσία και από το datetime_of_the_event που προσδιορίζει την χρέωση της υπηρεσίας. Οι στήλες του πίνακα είναι οι εξής:

- datetime_of_the_event : datetime foreign key από τον πίνακα Charge_for_service
- NFC_ID : int foreign key από τον πίνακα Customer
- service_ID : int foreign key από τον πίνακα Services

- **Visit_history**

Ο πίνακας αυτός έχει ακριβώς τα ίδια attributes με τον πίνακα Visit και γεμίζει με την χρήση triggers. Συγκεκριμένα, οι στήλες του είναι:

- visit_ID : int primary key AUTO_INCREMENT
- NFC_ID : int
- hotel_room_ID : int
- date_time_of_entrance : datetime primary key
- date_time_of_exit : datetime

- **Transactions_history**

Ο πίνακας αυτός έχει ακριβώς τα ίδια attributes με τον πίνακα Charge_for_service και γεμίζει με την χρήση triggers. Συγκεκριμένα οι στήλες του είναι:

- transaction_ID : int primary key AUTO_INCREMENT
- datetime_of_the_event : datetime primary key
- charge_description : varchar(80)
- amount : int
- NFC_ID : int foreign key από τον πίνακα Customer
- service_ID : int foreign key από τον πίνακα Services

- <u>Triggers:</u>

Στην ενότητα αυτή αναλύουμε όλους τους triggers που έχουμε ορίσει στη βάση δεδομένων μας οι οποίοι βρίσκονται στο αρχείο triggers.sql. Οι triggers αυτοί χωρίζονται στις εξής κατηγορίες:

1. Triggers ελέγχου τιμών
2. Triggers ελέγχου νομιμότητας ενεργειών
3. Triggers update τιμών
4. Triggers διατήρησης ιστορικού

- **Triggers ελέγχου τιμών**

  - **check_email_validity1** : Ο trigger αυτός τεστάρει το email που πάει να εισαχθεί σ' έναν Customer απέναντι σ' ένα regex expression που ελέγχει την ορθότητα του. Σε περίπτωση σφάλματος πετάει ένα exception.

  - **check_email_validity2** : Ο trigger αυτός τεστάρει το email που πάει να γίνει update σ' έναν Customer απέναντι σ' ένα regex expression που ελέγχει την ορθότητα του. Σε περίπτωση σφάλματος πετάει ένα exception.

  - **check_phone_validity1** : Ο trigger αυτός τεστάρει το τηλέφωνο που πάει να εισαχθεί σ' έναν Customer αν έχει 10 ψηφία. Σε περίπτωση σφάλματος πετάει ένα exception.

  - **check_phone_validity2** : Ο trigger αυτός τεστάρει το τηλέφωνο που πάει να γίνει update σ' έναν Customer αν έχει 10 ψηφία. Σε περίπτωση σφάλματος πετάει ένα exception.

  - **check_beds_validity1 :** Ο trigger αυτός τεστάρει αν κατά την εισαγωγή ενός δωματίου ο αριθμός των κρεβατίων είναι μικρότερος του 7. Σε αντίθετη περίπτωση πετάει ένα exception.

  - **check_beds_validity2 :** Ο trigger αυτός τεστάρει αν κατά το update ενός δωματίου ο αριθμός των κρεβατίων είναι μικρότερος του 7. Σε αντίθετη περίπτωση πετάει ένα exception.

  - **check_for_have_access_validity1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Have_access η ημερομηνία εισόδου είναι μικρότερη από την ημερομηνία εξόδου. Σε αντίθετη περίπτωση πετάει ένα exception.

  - **check_for_have_access_validity2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Have_access η ημερομηνία εισόδου είναι μικρότερη από την ημερομηνία εξόδου. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_for_visit_validity1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Visit η ημερομηνία εισόδου είναι μικρότερη από την ημερομηνία εξόδου. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_for_visit_validity2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Visit η ημερομηνία εισόδου είναι μικρότερη από την ημερομηνία εξόδου. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_for_amount_validity1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Charge_for_service το ποσό πληρωμής είναι μεγαλύτερο του μηδέν. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_for_amount_validity2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Charge_for_service το ποσό πληρωμής είναι μεγαλύτερο του μηδέν. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_time_have_access1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Have_access η ημερομηνία εισόδου ή η ημερομηνία εξόδου είναι μεγαλύτερες από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_time_have_access2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Have_access η ημερομηνία εισόδου ή η ημερομηνία εξόδου είναι μεγαλύτερες από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_time_registered1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Registered_to_services η ημερομηνία εγγραφής είναι μικρότερη από την σημερινή ημερομηνία. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_time_registered2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Registered_to_services η ημερομηνία εγγραφής είναι μικρότερη από την σημερινή ημερομηνία. Σε αντίθετη περίπτωση πετάει ένα exception.

- **check_time_visit1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Visit η ημερομηνία εισόδου ή η ημερομηνία εξόδου είναι μεγαλύτερες από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_time_visit2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Visit η ημερομηνία εισόδου ή η ημερομηνία εξόδου είναι μεγαλύτερες από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_time_charge1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Charge_for_services η ημερομηνία χρέωσης είναι μεγαλύτερη από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_time_charge2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Charge_for_services η ημερομηνία χρέωσης είναι μεγαλύτερη από την σημερινή ημερομηνία. Σε αυτή την περίπτωση πετάει ένα exception.

- **service_reg_no_reg1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή μίας υπηρεσίας στον πίνακα Services το attribute reg_or_no_reg έχει τιμή είτε 1 είτε 0. Σε αντίθετη περίπτωση πετάει ένα exception.

- **service_reg_no_reg2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση μίας υπηρεσίας στον πίνακα Services το attribute reg_or_no_reg έχει τιμή είτε 1 είτε 0. Σε αντίθετη περίπτωση πετάει ένα exception.

- **<u>Triggers ελέγχου νομιμότητας ενεργειών</u>**

- **check_reservation1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Have_access το δωμάτιο που εισάγεται είναι ήδη πιασμένο από άλλον πελάτη. Σε αυτή την περίπτωση πετάει ένα exception.

- **check_reservation2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Have_access το δωμάτιο που εισάγεται είναι ήδη πιασμένο από άλλον πελάτη. Σε αυτή την περίπτωση πετάει ένα exception.

- **have_access1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Have_access ο πελάτης έχει εγγραφεί στην υπηρεσία που προσπαθεί να μπει. Σε αντίθετη περίπτωση πετάει ένα exception.

- **have_access2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Have_access ο πελάτης έχει εγγραφεί στην υπηρεσία που προσπαθεί να μπει. Σε αντίθετη περίπτωση πετάει ένα exception.

- **can_visit1** : Ο trigger αυτός τεστάρει αν κατά την εισαγωγή στον πίνακα Visit ο πελάτης έχει πρόσβαση στο δωμάτιο που προσπαθεί να μπει. Σε αντίθετη περίπτωση πετάει ένα exception.

- **can_visit2** : Ο trigger αυτός τεστάρει αν κατά την ανανέωση στον πίνακα Visit ο πελάτης έχει πρόσβαση στο δωμάτιο που προσπαθεί να μπει. Σε αντίθετη περίπτωση πετάει ένα exception.

- **room_lower1** : Ο trigger αυτός τεστάρει αν πριν την εισαγωγή στον πίνακα Provided_to τα ID που αντιστοιχούν στα δωμάτια είναι μικρότερα του 100. Σε αντίθετη περίπτωση πετάει ένα exception.

- **room_lower2** : Ο trigger αυτός τεστάρει αν πριν την ανανέωση στον πίνακα Provided_to τα ID που αντιστοιχούν στα δωμάτια είναι μικρότερα του 100. Σε αντίθετη περίπτωση πετάει ένα exception.

➢ **<u>Triggers update τιμών</u>**

- **service_isa** : Ο trigger αυτός μετά την εισαγωγή μίας υπηρεσίας στον πίνακα Services ανάλογα την τιμή του attribute reg_or_no_reg βάζει το αντίστοιχο service_ID στους πίνακες Services_that_dont_need_registration ή Services_that_need_registration.

- **<u>Triggers διατήρησης ιστορικού</u>**

  - **update_history_customer_delete1** : Ο trigger αυτός μετά την εισαγωγή στον πίνακα Visit βάζει την γραμμή που εισάχθηκε στον πίνακα Visit_history. Με αυτόν τον τρόπο ακόμα και με την διαφραφή κάποιου πελάτη ή κάποιου χώρου θα μπορούμε να δούμε όλες τις επισκέψεις που έχουν γίνει. Αυτή η λεπτομέρεια είναι σημαντική όταν αναζητουμέ πιθανές επαφές με κάποιο κρούσμα.

  - **update_history_customer_delete2** : Ο trigger αυτός μετά την ανανέωση στον πίνακα Visit βάζει την γραμμή που εισάχθηκε στον πίνακα Visit_history.

  - **update_history_transactions1** : Ο trigger αυτός μετά την εισαγωγή στον πίνακα Charge_for_service βάζει την γραμμή που εισάχθηκε και στον πίνακα Transactions_history.

  - **update_history_transactions2** : Ο trigger αυτός μετά την ανανέωση στον πίνακα Charge_for_service βάζει την γραμμή που εισάχθηκε και στον πίνακα Transactions_history.

- <u>Περιορισμοί σε επίπεδο εφαρμογής:</u>

Οι περιορισμοί που θέσαμε σε επίπεδο εφαρμογής φαίνονται παρακάτω:

- Για να δει κάποιος όλα τα στοιχεία που αφορούν κάποιον πελάτη πρέπει να μπει στο Customer Information και να επιλέξει το ID ενός από τους πελάτες που βλέπει. Αυτό γίνεται για να μην υπερφορτώσουμε με πληροφορίες τον αρχικό πίνακα. Στην περίπτωση που ο χρήστης βάλει ένα ID που δεν υπάρχει τότε όλα τα δεδομένα που θα δει θα είναι undefined.
- Ψάχνοντας τις επισκέψεις που έχουν γίνει από πελάτες με πολλαπλά κριτήρια αν στο must registered δεν θέσει κάποια από τις τιμές yes ή no τότε απλά το φίλτρο θα είναι σαν να μην χρησιμοποιείται.
- Δεν μπορείς να ψάξεις τις επισκέψεις που έχουν γίνει βάσει ημερομηνίας αν δεν χρησιμοποιήσεις το σωστό format για την ημερομηνία, δηλαδή DD/MM/YYYY.
- Για να δείξει τους πιθανούς πελάτες οι οποίοι είναι πιθανόν να είναι θετικοί στον covid19 μετά από κάποιο επιβεβαιωμένο κρούσμα η εφαρμογή λαμβάνει υπόψιν της μόνο τους χώρους στους οποίους έχουν βρεθεί και οι 2 (όχι απαραίτητα ταυτόχρονα) και όχι το ότι μπορεί επειδή μένουν στον ίδιο όροφο και στον ίδιο διάδρομο να συναντήθηκαν έξω από τα δωμάτια τους.

- Views

Δημιουργήθηκαν 2 views, το customer_info το οποίο έχει όλα τα στοιχεία των πελατών μαζί με τα email και τα τηλέφωνα τους, και το sales_of_services, το οποίο περιέχει τις πωλήσεις ανά κατηγορία υπηρεσίας για τις υπηρεσίες που απαιτούν εγγραφή. Γνωρίζουμε ότι τα views είναι ουσιαστικά αποθηκευμένα queries που εκτελούνται κάθε φορά που θέλουμε να τα χρησιμοποιήσουμε, για αυτό και τα views είναι αυτόματα ανανεώσιμα.

1.b. Ευρετήρια

Γνωρίζουμε ότι τα ευρετήρια χρησιμοποιούνται για την ταχύτερη προσπέλαση σ' ένα table χωρίς να διαβαστούν όλες οι εγγραφές που περιέχει αυτό. Ορίζουμε ευρετήρια μόνο σε στήλες που χρησιμοποιούμε συχνά σε αρκετά queries. Η mysql ορίζει από μόνη της όλα τα primary και foreign keys ως ευρετήρια. Εμείς ορίσαμε τα παρακάτω:

```
CREATE INDEX visit_date_and_time on Visit_history(date_time_of_entrance, date_time_of_exit);
CREATE INDEX have_access_date_and_time on Have_access(starting_time_date, ending_time_date);
CREATE INDEX registered_time_and_date on Registered_to_services(datetime_of_registration);
```

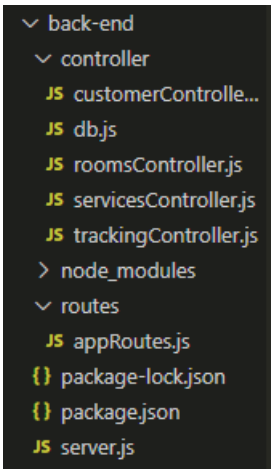
Η χρήση των ευρετηρίων έχει τα selections είναι πολύ περισσότερα από τα insertions και ιδιαίτερα όταν γίνονται με where clause που αφορά στήλες που δεν είναι keys. Αυτό που μας ενδιαφέρει κυριώς από την συγκεκριμένη είναι να βρούμε πράγματα βάσει του χρόνου που συνέβησαν τα γεγονότα. Για παράδειγμα μας ενδιαφέρει να ψάξουμε στον πίνακα Visit_history βάσει των ημερομηνιών για να βρούμε πιθανά κρούσματα. Παρόμοια και για τον πίνακα Have_access εκτός από το ID του δωματίου το οποίο είναι ορισμένο ως ευρετήριο από την mysql μας ενδιαφέρει και για σε ποια χρονική περίοδο έχει πρόσβαση σε κάποια υπηρεσία ο πελάτης.

1.c. Σύστημα και γλώσσες προγραμματισμού της εφαρμογής

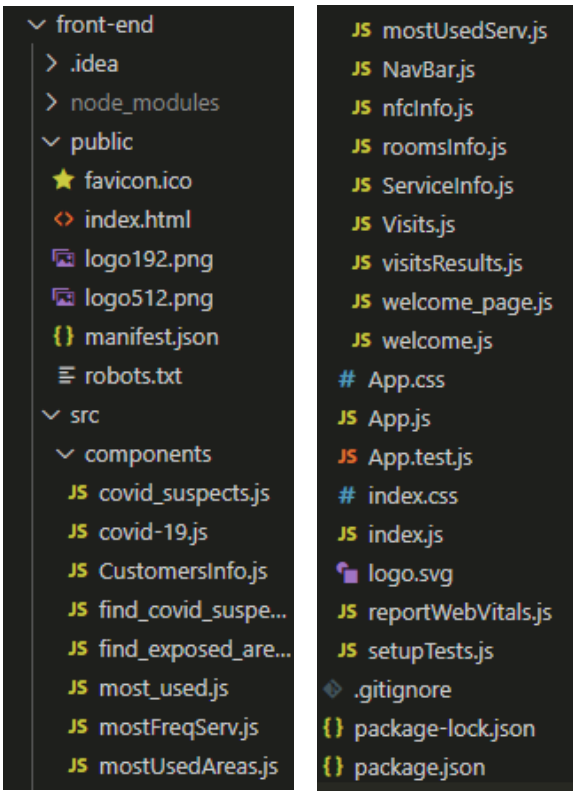
Για την ανάπτυξη της εφαρμογής αυτής εργαστήκαμε με REST API, χρησιμοποιώντας αρχιτεκτονική MVC, όπου το front-end (view) επικοινωνεί με το back-end (controller), και αυτό με τη σειρά του με τη βάση.

- Την εφαρμογή την υλοποιήσαμε σε Windows και Linux.
- Για τη βάση, προφανώς χρησιμοποιήσαμε MySQL Server.
- Για το Back-end και το Front-end χρησιμοποιήσαμε JavaScript, μέσω nodeJS/Express και reactJS αντίστοιχα, ενώ για τη διαμόρφωση του γραφικού περιβάλλοντος κάναμε και χρήση css. Η δομή των 2 αυτών μερών φαίνεται παρακάτω:

    - Back-end:



    - Front-end:



1.d. <u>Εγκατάσταση εφαρμογής</u>

Για την εγκατάσταση της εφαρμογής χρειάζεται να εκτελεστούν τα παρακάτω βήματα:

- Αλλαγή κωδικού στο πεδίο *"password"* στα αρχεία ***back-end/controller/db.js*** και ***back-end/server.js*** για σύνδεση στον MySQL Server.

- Δημιουργία και διαμόρφωση βάσης δεδομένων:
  - Έπειτα, σε ένα terminal πρέπει να εκτελεστούν τα παρακάτω:
    - ***"cd sql"*** για είσοδο στον φάκελο sql.
    - ***"mysql -u root"*** ή ***"mysql -u root -p"*** αν ο MySQL Server έχει κωδικό.
    - ***"source init.sql;"*** για δημιουργία και διαμόρφωση της βάσης.

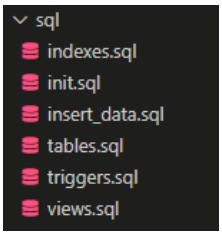
- Εγκατάσταση εξαρτήσεων και ενεργοποίηση του server (Back-end):
  - Σε νέο terminal, πρέπει να εκτελεστούν τα παρακάτω:
    - **"cd back-end"** για είσοδο στον φάκελο back-end.
    - **"npm install"** για εγκατάσταση εξαρτήσεων.
    - **"node server"** για ενεργοποίηση του server.

- Εγκατάσταση εξαρτήσεων για το γραφικό περιβάλλον (Front-end):
  - Σε νέο terminal, πρέπει να εκτελεστούν τα παρακάτω:
    - ***"cd front-end"*** για είσοδο στον φάκελο front-end.
    - ***"npm install"*** για εγκατάσταση εξαρτήσεων.
    - ***"npm start"*** για άνοιγμα του UI.

- Έπειτα, η εφαρμογή θα τρέχει στην διεύθυνση ***localhost:3000***.

2. ΚΩΔΙΚΑΣ SQL

Ο SQL κώδικας χωρίζεται στα παρακάτω 6 αρχεία:



- init.sql:

```
source ./tables.sql;
source ./triggers.sql;
source ./insert_data.sql;
source ./indexes.sql;
source ./views.sql;
```

Εκτελεί με συγκεκριμένη σειρά όλα τα υπόλοιπα sql αρχεία, για την δημιουργία και διαμόρφωση της βάσης.

- tables.sql:

```
DROP DATABASE IF EXISTS ntuadb;
CREATE DATABASE ntuadb;

USE ntuadb;

CREATE TABLE Customer(
    -- NFC_ID INT NOT NULL IDENTITY(1,1),
    NFC_ID INT NOT NULL AUTO_INCREMENT,
    firstname VARCHAR(20) NOT NULL,
    lastname VARCHAR(20) NOT NULL,
    dateofbirth DATE NOT NULL,
    number_of_indentification_document VARCHAR(20) NOT NULL,
    type_of_indentification_document VARCHAR(20) NOT NULL,
    issuing_authority VARCHAR(30) NOT NULL,
    PRIMARY KEY (NFC_ID)
);

CREATE TABLE Customer_emails(
    NFC_ID INT NOT NULL,
    email VARCHAR(70) NOT NULL,
    PRIMARY KEY (NFC_ID, email), FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE
);

CREATE TABLE Customer_phones(
    NFC_ID INT NOT NULL,
    phone VARCHAR(30) NOT NULL,
    PRIMARY KEY (NFC_ID, phone), FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE
);

CREATE TABLE Services(
    service_ID INT NOT NULL,
    service_description VARCHAR(50) NOT NULL,
    reg_or_no_reg INT NOT NULL, -- 0 if need registration and 1 if not
    PRIMARY KEY (service_ID)
);

CREATE TABLE Services_that_need_registration(
    service_ID INT NOT NULL,
    PRIMARY KEY (service_ID),
    FOREIGN KEY (service_ID) REFERENCES Services(service_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Services_that_dont_need_registration(
    service_ID INT NOT NULL,
    PRIMARY KEY (service_ID),
    FOREIGN KEY (service_ID) REFERENCES Services(service_ID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```sql

CREATE TABLE Hotel_rooms(
    hotel_room_ID INT NOT NULL PRIMARY KEY,
    number_of_beds INT NOT NULL,
    name_of_the_room VARCHAR(25) NOT NULL,
    description_of_position VARCHAR(100) NOT NULL
);

CREATE TABLE Have_access(
    NFC_ID INT NOT NULL,
    hotel_room_ID INT NOT NULL,
    starting_time_date DATETIME NOT NULL,
    ending_time_date DATETIME NOT NULL,
    PRIMARY KEY (NFC_ID, hotel_room_ID),
    FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE,
    FOREIGN KEY (hotel_room_ID) REFERENCES Hotel_rooms(hotel_room_ID) ON DELETE CASCADE
);

CREATE TABLE Visit(
    NFC_ID INT NOT NULL,
    hotel_room_ID INT NOT NULL,
    date_time_of_entrance DATETIME NOT NULL,
    date_time_of_exit DATETIME NOT NULL,
    PRIMARY KEY (NFC_ID, hotel_room_ID, date_time_of_entrance),
    FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE cascade,
    FOREIGN KEY (hotel_room_ID) REFERENCES Hotel_rooms(hotel_room_ID) ON DELETE cascade
);

CREATE TABLE Registered_to_services(
    NFC_ID INT NOT NULL,
    service_ID INT NOT NULL,
    datetime_of_registration DATETIME NOT NULL,
    PRIMARY KEY (NFC_ID, service_ID),
    FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE,
    FOREIGN KEY (service_ID) REFERENCES Services_that_need_registration(service_ID) ON DELETE CASCADE
);

CREATE TABLE Provided_to(
    hotel_room_ID INT NOT NULL,
    service_ID INT NOT NULL,
    PRIMARY KEY (hotel_room_ID, service_ID),
    FOREIGN KEY (service_ID) REFERENCES Services(service_ID) ON DELETE CASCADE,
    FOREIGN KEY (hotel_room_ID) REFERENCES Hotel_rooms(hotel_room_ID) ON DELETE CASCADE
);
```

```sql

CREATE TABLE Charge_for_service(
    datetime_of_the_event DATETIME NOT NULL,
    charge_description VARCHAR(80) NOT NULL,
    amount INT NOT NULL,
    NFC_ID INT NOT NULL,
    service_ID INT NOT NULL,
    PRIMARY KEY (datetime_of_the_event, service_ID, NFC_ID),
    FOREIGN KEY (service_ID) REFERENCES Services(service_ID) ON DELETE CASCADE,
    FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE
);

CREATE TABLE Receive_services(
    NFC_ID INT NOT NULL,
    service_ID INT NOT NULL,
    datetime_of_the_event DATETIME NOT NULL,
    PRIMARY KEY(NFC_ID, service_ID, datetime_of_the_event),
    FOREIGN KEY (service_ID) REFERENCES Services(service_ID) ON DELETE CASCADE,
    FOREIGN KEY (NFC_ID) REFERENCES Customer(NFC_ID) ON DELETE CASCADE,
    FOREIGN KEY (datetime_of_the_event) REFERENCES Charge_for_service(datetime_of_the_event) ON DELETE CASCADE
);

CREATE TABLE Visit_history(
    visit_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    NFC_ID INT NOT NULL,
    hotel_room_ID INT NOT NULL,
    date_time_of_entrance DATETIME NOT NULL,
    date_time_of_exit DATETIME NOT NULL
);

CREATE TABLE Transactions_history(
    transaction_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    datetime_of_the_event DATETIME NOT NULL,
    charge_description VARCHAR(80) NOT NULL,
    amount INT NOT NULL,
    NFC_ID INT NOT NULL,
    service_ID INT NOT NULL
);
```

- <u>triggers.sql:</u>

  Το αρχείο αυτό έχει 393 γραμμές κώδικα, συνεπώς δεν το βάλαμε ολόκληρο στην αναφορά (υπάρχει στα παραδοτέα). Για λόγους πληρότητας, βάλαμε ενδεικτικά κάποια triggers για κάθε κατηγορία που περιγράψαμε στο ζητούμενο 1.a.

  - <u>triggers ελέγχου τιμών:</u>

```
5   CREATE TRIGGER check_email_validity1
6   BEFORE UPDATE ON Customer_emails
7       FOR EACH ROW
8       BEGIN
9           if (new.email not like '%@%.%' or new.email like '@%' or new.email like '%@%@%' ) THEN
10              signal sqlstate '45000' set message_text = 'Wrong email format inserted';
11          end if;
12      end;
13  $$
```

```
26   CREATE TRIGGER check_phone_validity1
27  BEFORE UPDATE ON Customer_phones
28      FOR EACH ROW
29      BEGIN
30          if (length(new.phone) <> 10) THEN
31              signal sqlstate '45000' set message_text = 'Wrong phone inserted';
32          end if;
33      end;
34
35   $$
```

```
48   CREATE TRIGGER check_beds_validity1
49   BEFORE INSERT ON Hotel_rooms
50      FOR EACH ROW
51      BEGIN
52          if (new.number_of_beds >=7) THEN
53          signal sqlstate '45000' set message_text = 'Our hotel doesnt have rooms with more than 7 beds';
54          end if;
55      end;
56
57   $$
```

```
81   CREATE TRIGGER check_for_have_access_validity2
82   BEFORE UPDATE ON Have_access
83      FOR EACH ROW
84      BEGIN
85          if (new.starting_time_date > new.ending_time_date) THEN
86          signal sqlstate '45000' set message_text = 'Ending date cant be smaller than starting date (Have_access table)';
87          end if;
88      end;
89
90   $$
```

```
92   CREATE TRIGGER check_for_visit_validity1
93  BEFORE UPDATE ON Visit
94      FOR EACH ROW
95      BEGIN
96          if (new.date_time_of_entrance > new.date_time_of_exit) THEN
97          signal sqlstate '45000' set message_text = 'Ending date cant be smaller than starting date (Visit table)';
98          end if;
99      end;
100
101  $$
```

```
114   CREATE TRIGGER check_for_amount_validity1
115  BEFORE INSERT ON Charge_for_service
116      FOR EACH ROW
117      BEGIN
118          if (new.amount <= 0) THEN
119          signal sqlstate '45000' set message_text = 'Amount cant be lower or equal to zero';
120          end if;
121      end;
122
123   $$
```

```
202   CREATE TRIGGER check_time_visit1
203   BEFORE INSERT ON Visit
204       FOR EACH ROW
205       BEGIN
206           IF (new.date_time_of_entrance > NOW() OR new.date_time_of_exit > NOW()) THEN
207           signal sqlstate '45000' set message_text = 'Cant reference future time';
208               end if;
209           end;
```

- triggers ελέγχου νομιμότητας ενεργειών:

```
CREATE TRIGGER check_reservation1
BEFORE INSERT ON Have_access
    FOR EACH ROW
    BEGIN
        IF (EXISTS (SELECT * FROM Have_access WHERE new.hotel_room_ID = hotel_room_ID AND new.hotel_room_ID < 100 AND NOT (new.ending_time_date <= starting_time_date OR new.starting_time_date >= ending_time_date))) THEN
        signal sqlstate '45000' set message_text = 'Room is already occupied';
        end if;
    end;

$$
```

```
CREATE TRIGGER have_access1
BEFORE INSERT ON Have_access
    FOR EACH ROW
    BEGIN
        IF (NOT EXISTS (SELECT NFC_ID, hotel_room_ID FROM Registered_to_services JOIN Provided_to USING(service_ID)
                        WHERE (new.NFC_ID = NFC_ID AND new.hotel_room_ID = hotel_room_ID) OR new.hotel_room_ID < 399)) THEN
        signal sqlstate '45000' set message_text = 'Cant have access to that room';
        end if;
    end;

$$
```

```
CREATE TRIGGER can_visit1
BEFORE INSERT ON Visit
    FOR EACH ROW
    BEGIN
        IF (NOT EXISTS(SELECT NFC_ID, hotel_room_ID FROM Have_access
        WHERE (new.NFC_ID = NFC_ID AND new.hotel_room_ID = hotel_room_ID))) THEN
        signal sqlstate '45000' set message_text = 'Cant access that room';
        end if;
    end;

$$
```

```
CREATE TRIGGER room_lower1
BEFORE INSERT ON Provided_to
    FOR EACH ROW
    BEGIN
        IF ((new.hotel_room_ID < 100 AND new.service_ID <>0) OR
            (new.hotel_room_ID > 100 AND new.service_ID = 0)) THEN
        signal sqlstate '45000' set message_text = 'Rooms have an Id of less than 100';
        end if;
    end;

$$
```

- triggers update τιμών:

```
CREATE TRIGGER service_isa
AFTER INSERT ON Services
    FOR EACH ROW
    BEGIN
        IF (new.reg_or_no_reg = 0) THEN
            INSERT INTO Services_that_dont_need_registration(service_ID) VALUES (new.service_ID);
        end if;
        IF (new.reg_or_no_reg = 1) THEN
            INSERT INTO Services_that_need_registration(service_ID) VALUES (new.service_ID);
        end if;
    end;

$$
```

- triggers διατήρησης ιστορικού:

```
CREATE TRIGGER update_history_customer_delete1
AFTER INSERT ON Visit
    FOR EACH ROW
    BEGIN
        INSERT INTO Visit_history(NFC_ID, hotel_room_ID, date_time_of_entrance, date_time_of_exit) VALUES (new.NFC_ID, new.hotel_room_ID, new.date_time_of_entrance, new.date_time_of_exit);
    end;

$$
```

```
CREATE TRIGGER update_history_transactions1
AFTER INSERT ON Charge_for_service
    FOR EACH ROW
    BEGIN
        INSERT INTO Transactions_history(datetime_of_the_event, charge_description, amount, NFC_ID, service_ID) VALUES
                (new.datetime_of_the_event, new.charge_description, new.amount, new.NFC_ID, new.service_ID);
    end;

$$
```

- insert_data.sql:

Αντίστοιχα με πριν, το αρχείο είναι τεράστιο (757 γραμμές κώδικα), συνεπώς θα δείξουμε λίγα insert για κάθε πίνακα. Σε μερικούς πίνακες, όπως οι *"Visit_history"*, *"Services_that_need_registration"* και *"Services_that_dontneed_registration"*, η εισαγωγή γίνεται αυτόματα μέσω των κατάλληλων triggers.

```
USE ntuadb;

INSERT INTO Customer (firstname, lastname, dateofbirth, number_of_indentification_document, type_of_indentification_document, issuing_authority) VALUES
                ('Konstantinos', 'Antoniou', '1998-04-06', 'AL-344513', 'ID', 'T.A. Evoias'),
                ('Zacharias', 'Anastasakis', '1998-05-03', 'AI-339213', 'ID', 'T.A. HRAKLEIOU'),
                ('Ioannis', 'Metzakis', '1998-02-17', 'AK-635524', 'ID', 'T.A. HRAKLEIOU'),
                ('Giorgos', 'Anastasiou', '1998-01-23', 'AL-632593', 'ID', 'T.A. Evoias'),
                ('Apostolis', 'Molivdas', '1998-05-25', 'AE0052412', 'Passport', 'A.E.A./D.D.-N.P.C'),
```

```
INSERT INTO Customer_emails(NFC_ID, email) VALUES
        (1, 'kostas.an2016@gmail.com'),
        (2, 'zacharias_anastasakis@gmail.com'),
        (3, 'metzakis_ioannis@gmail.com'),
        (4, 'giorgos_anastasiou@gmail.com'),
        (5, 'apostolos_molyvdas@gmail.com'),
```

```
INSERT INTO Customer_phones(NFC_ID, phone) VALUES
        (1, '6976522415'),
        (1, '6971548542'),
        (2, '6978546990'),
        (3, '6974510090'),
        (4, '6941258840'),
        (5, '6936699512'),
```

```
INSERT INTO Services(service_ID, service_description, reg_or_no_reg) VALUES
        (0, 'room', 0), -- pseudo service
        (1, 'bar', 0),
        (2, 'restaurant', 0),
        (3, 'hair salon', 0),
        (4, 'gym', 1),
        (5, 'conference room', 1),
        (6, 'sauna', 1);
```

```
INSERT INTO Hotel_rooms(hotel_room_ID, number_of_beds, name_of_the_room, description_of_position) VALUES
        (1, 2, 'Room-1', 'First floor east corridor'),
        (2, 4, 'Room-2', 'First floor east corridor'),
        (3, 3, 'Room-3', 'First floor east corridor'),
        (4, 1, 'Room-4', 'First floor west corridor'),
        (5, 5, 'Room-5', 'First floor west corridor'),
        (6, 1, 'Room-6', 'First floor west corridor'),
```

```
INSERT INTO Registered_to_services(NFC_ID, service_ID, datetime_of_registration) VALUES
        (1, 4, '2021-04-23 15:10:00'),
        (1, 6, '2021-04-23 15:11:20'),
        (2, 5, '2021-06-12 09:13:55'),
        (3, 4, '2021-05-01 09:10:25'),
```

```
INSERT INTO Provided_to(hotel_room_ID, service_ID) VALUES
        (1, 0),
        (2, 0),
        (3, 0),
        (4, 0),
        (5, 0),
        (6, 0),
```

```
INSERT INTO Have_access(NFC_ID, hotel_room_ID, starting_time_date, ending_time_date) VALUES
        (1, 1, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 100, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 101, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 102, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 103, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 104, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
        (1, 105, '2021-04-23 12:10:00', '2021-04-30 12:00:00'),
```

```
INSERT INTO Visit(NFC_ID, hotel_room_ID, date_time_of_entrance, date_time_of_exit) VALUES
                (1, 1, '2021-04-23 12:15:00', '2021-04-23 15:25:05'),
        (1, 300, '2021-04-23 15:30:00', '2021-04-23 16:00:00'),
        (1, 1, '2021-04-23 20:15:00', '2021-04-25 02:25:05'),
        (1, 100, '2021-04-25 02:55:00', '2021-04-25 04:00:00'),
        (1, 101, '2021-04-25 04:15:23', '2021-04-25 06:30:11'),
        (1, 1, '2021-04-26 12:15:00', '2021-04-26 15:30:23'),
        (1, 200, '2021-04-26 21:00:15', '2021-04-26 22:15:00'),
        (1, 1, '2021-04-27 08:20:00', '2021-04-27 21:18:05'),
        (1, 1, '2021-04-29 09:55:00', '2021-04-29 15:45:14'),
        (1, 401, '2021-04-23 16:15:00', '2021-04-23 17:25:05'),
```

```
INSERT INTO Charge_for_service(NFC_ID, service_ID, datetime_of_the_event, charge_description, amount) VALUES
        (1, 0, '2021-04-23 12:10:00', 'Payment for the Room', 400),
        (1, 4, '2021-04-23 15:10:00', 'Payment for gym service', 30),
        (1, 6, '2021-04-23 15:11:20', 'Payment for sauna service', 25),
        (1, 3, '2021-04-23 16:00:00', 'Haircut', 10),
        (1, 1, '2021-04-25 03:00:00', 'Drink', 7),
        (1, 1, '2021-04-25 04:20:23', 'Coca Cola', 4),
        (1, 2, '2021-04-26 22:10:00', 'Dinner', 25),
        (2, 0, '2021-06-12 09:01:00', 'Payment for the Room', 600),
        (2, 5, '2021-06-12 09:13:55', 'Payment for conference room service', 40),
```

```
INSERT INTO Receive_services(NFC_ID, service_ID, datetime_of_the_event) VALUES
        (1, 0, '2021-04-23 12:10:00'),
        (1, 4, '2021-04-23 15:10:00'),
        (1, 6, '2021-04-23 15:11:20'),
        (1, 3, '2021-04-23 16:00:00'),
        (1, 1, '2021-04-25 03:00:00'),
        (1, 1, '2021-04-25 04:20:23'),
        (1, 2, '2021-04-26 22:10:00'),
        (2, 0, '2021-06-12 09:01:00'),
        (2, 5, '2021-06-12 09:13:55'),
```

- indexes.sql:

  Έχει αναλυθεί παιρετέρω και στο ζητούμενο 1.b., με το αρχείο να είναι το παρακάτω:

```sql
USE ntuadb;

CREATE INDEX visit_date_and_time on Visit_history(date_time_of_entrance, date_time_of_exit);
CREATE INDEX have_access_date_and_time on Have_access(starting_time_date, ending_time_date);
CREATE INDEX registered_time_and_date on Registered_to_services(datetime_of_registration);
```

- views.sql:

  Τελευταίο είναι το αρχείο που περιέχει τις 2 ζητούμενες όψεις του σχεσιακού μοντέλου για το ερώτημα 8:

```sql
CREATE VIEW sales_of_services AS
SELECT Services.service_description, count(Registered_to_services.service_ID)
FROM Registered_to_services, Services
WHERE Services.service_ID IN (Registered_to_services.service_ID)
GROUP BY Services.service_ID;

CREATE VIEW customer_info AS
SELECT * FROM Customer JOIN Customer_phones USING (NFC_ID) JOIN Customer_emails USING (NFC_ID);
```

3. CONFIGURATION FILES

Για την εγκατάσταση των απαραίτητων βιβλιοθηκών για τη λειτουργία του Server και του UI, υπάρχουν τα 2 αντίστοιχα *"package.json"* αρχεία.

back-end/package.json

```json
{
  "name": "ntuadb",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "experss": "0.0.1-security",
    "express": "^4.17.1",
    "mysql": "^2.18.1",
    "nodemon": "^2.0.7"
  }
}
```

front-end/package.json

```json
{
  "name": "my-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@date-io/date-fns": "^1.3.13",
    "@testing-library/jest-dom": "^5.14.1",
    "@testing-library/react": "^11.2.7",
    "@testing-library/user-event": "^12.8.3",
    "date-fns": "^2.22.1",
    "moment": "^2.29.1",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.1.2"
  },
  Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@material-ui/core": "^4.11.4",
    "@material-ui/icons": "^4.11.2",
    "@material-ui/pickers": "^3.3.10",
    "axios": "^0.21.1",
    "clsx": "^1.1.1",
    "react-router-dom": "^5.2.0"
  }
}
```