Εθνικό Μετσόβιο Πολυτεχνείο

Βάσεις Δεδομένων

Αναφορά Εξαμηνιαίο Project 2013-2014

Ηλίας Φωτόπουλος
03109106
Θανάσης Βράτιμος
03110769

1 Λεπτομέρειες Υλοποίησης

Για την κατασκευή του project έγινε χρήση των παρακάτω τεχνολογιών:

- Ruby on Rails (Framework)
- HTML Sass (Syntactically Awesome Style Sheets)
- Bootstrap (Sass Framework)
- CoffeeScript
- Git (Version Control)

Κατά την φάση του σχεδιασμού αποφασίστηκε η χρήση του RoR framework λόγω της αρχιτεκτονικής MVC που ακολουθεί αλλά και λόγω της φιλοσοφίας του Convention over Configuration (CoC) και Don't Repeat Yourself (DRY).

2 Views

Δημιουργήσαμε τα παρακάτω δύο views. Το view upcoming reservations περιέχει όλες τις κρατήσεις για τις οποίες η μέρα άφιξης είναι μεταγενέστερη της σημερινής μέρας. Επίσης εκτελεί inner join, με τα tables: clients,hotels,rooms έτσι ώστε να εμφανίσει όνομα πελάτη, ξενοδοχείου αλλά και τον τύπο του δωματίου. Η χρησιμότητα του view είναι η παρουσίαση όλων των κρατήσεων για τις οποίες ο πελάτης δεν έχει φτάσει ακόμη.

```
1  CREATE OR REPLACE VIEW upcoming_reservations AS
2  SELECT reservations.id, reservations.arrival_date, ↩
       reservations.departure_date, reservations.created_at, ↩
       reservations.updated_at, clients.first_name, clients.↩
       last_name, hotels.name, rooms.room_type
3  FROM reservations
4  INNER JOIN clients
5  ON reservations.client_id = clients.id
6  INNER JOIN hotels
7  ON reservations.hotel_id = hotels.id
8  INNER JOIN rooms
9  ON reservations.room_id = rooms.id
10 WHERE arrival_date > CURDATE()
11 ORDER BY arrival_date
```

Το view upcoming reservations updatable επιτελεί τις ίδιες λειτουργίες με το παραπάνω με δύο βασικές διαφορές:

- Δεν χρησιμοποιεί inner join για να πάρει τα names από τα άλλα tables.
- Περιέχει κρατήσεις όπου η μέρα αναχώρησης (όχι άφιξης) είναι μεταγενέστερη της σημερινής

Το εν λόγω view παίζει σημαντικό ρόλο στην εσωτερική αρχιτεκτονική και λογική της εφαρμογής μας. Χρησιμοποιείται από το ReservationService το οποίο ελέγχει αν ένα δωμάτιο είναι διαθέσιμο για κράτηση.

```
1 CREATE OR REPLACE VIEW upcoming_reservations_updatable AS
2 SELECT *
3 FROM reservations
4 WHERE departure_date > CURDATE()
5 ORDER BY arrival_date
```

3 Triggers

Δημιουργήσαμε τα παρακάτω δύο triggers.
Το trigger delete hotel rooms, το οποίο πριν διαγράψει ένα ξενοδοχείο διαγράφει και όλα τα δωμάτια του. Τα δωμάτια ανήκουν στο αδύναμο σύνολο οντοτήτων "Δωμάτιο" και εξαρτιούνται πλήρως από το ξενοδοχείο στο οποίο ανήκουν, οπότε και δεν έχει νόημα να υφίστανται αν διαγραφεί το ξενοδοχείο τους.

```
1 CREATE TRIGGER delete_hotel_rooms
2 BEFORE DELETE ON hotels
3 FOR EACH ROW
4 BEGIN
5   DELETE FROM rooms WHERE OLD.id = hotel_id;
6 END
```

Παρομοίως με παραπάνω δημιουργήσαμε ένα trigger που διαγράφει όλες τις κρατήσεις ενός πελάτη πριν διαγραφεί ο πελάτης.

```
1 CREATE TRIGGER delete_client_reservations
2 BEFORE DELETE ON clients
3 FOR EACH ROW
4 BEGIN
5   DELETE FROM reservations WHERE OLD.id = client_id;
6 END
```

Στο σημείο αυτό αξίζει να αναφερθεί ότι το παραπάνω θα μπορούσε να υλοποιηθεί εύκολα και μέσα από το περιβάλλον του RoR Framework (χρήση Rails ActiveRecord callbacks). Αρκεί να προστεθούν τα παρακάτω στα μοντέλα hotel και client αντίστοιχα.

```
1 has_many :rooms, dependent: :destroy
```

```
1 has_many :reservations, dependent: :destroy
```

3.1 Callback vs Trigger

Πλεονεκτήματα Callback:

- Όλη η Business Logic του μοντέλου βρίσκεται στην Rails κάτι το οποίο κάνει πιο εύκολη την συντήρηση και ανάπτυξη του.
- Εύκολο Debug
- Ο κώδικας Ruby γράφετε και διαβάζεται πιο εύκολα
- Η εφαρμογή μας δεν εξαρτάται από την υλοποίηση (και συντακτικό) της βάσης δεδομένων.

Πλεονεκτήματα Trigger:

- Είναι πιο γρήγορο από ένα callback. (Στο callback χρειάζεται να συνδεθείς στην βάση δεδομένων, ενώ στο trigger είσαι ήδη στο layer της βάσης δεδομένων)

Βάση των παραπάνω και δεδομένου το μέγεθος της εφαρμογής το ιδανικό θα ήταν να γίνει χρήση Rails ActiveRecord callbacks.