ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΧΟΛΕΙΟ ΚΩΔΙΚΑ

ΤΙΤΛΟΣ ΕΡΓΑΣΙΑΣ

Εργαλείο κατάρτισης επιχειρηματικού σχεδίου

(Ομάδα Σχεδιασμού Βάσης Δεδομένων)

Φεβρουάριος 2018

Η παρούσα εργασία εκπονήθηκε από τους παρακάτω φοιτητές:

- Τζιλής Αντώνης (ΑΕΜ: 2809, 6ο εξάμηνο)
- Ναλπαντίδης Νικός (ΑΕΜ: 2780, 6ο εξάμηνο)
- Βενέτης Θεόδωρος (ΑΕΜ: 2270, 11ο εξάμηνο)
- Τσουργιάννης Κώνσταντινος (ΑΕΜ: 2609, 8ο εξάμηνο)
- Χριστόπουλος Γιώργος (ΑΕΜ: 2821, 6ο εξάμηνο)

1. Εισαγωγή

Η συγκεκριμένη εργασία, σε ευρύτερο πλαίσιο, πραγματεύεται την υλοποίηση λογισμικού ανοιχτού κώδικα, με σκοπό την παραγωγή ενός Επιχειρηματικού Σχεδίου. Με τον όρο «επιχειρηματικό σχέδιο», εννοούμε ένα εκτενές, επίσημο και κατάλληλα δομημένο έγγραφο, το οποίο περιγράφει αναλυτικά μια επιχειρηματική ιδέα. Εύκολα γίνεται αντιληπτό πως η κατασκευή ενός τέτοιου εγγράφου απαιτεί εξειδικευμένες γνώσεις. Επομένως, το παραπάνω γεγονός αποτελεί τροχοπέδη για όσους έχουν εμπνευστεί κάποια επιχειρηματική ιδέα, αλλά δε διαθέτουν τα κατάλληλα τεχνικά προσόντα για να συντάξουν το συγκεκριμένο έγγραφο και -εν τέλει- να πραγματοποιήσουν το όραμα τους. Τέλος, επισημαίνεται οτι δεν υπάρχει άλλο αντίστοιχο λογισμικό ανοιχτού κώδικα στο διαδίκτυο, που να ικανοποιεί τις παραπάνω απαιτήσεις. Συνεπώς, κρίνεται απαραίτητη η δημιουργία μιας ολοκληρωμένης πλατφόρμας διαχείρισης επιχειρηματικών σχεδίων, με σκοπό τη διευκόλυνση πρός το ευρύ κοινό.

Είναι προφανές οτι ένα τέτοιο λογισμικό πρέπει να εκμαιεύσει κάποιες πληροφορίες από τον τελικό χρήστη, προκειμένου να συντάξει αυτοματοποιημένα το επιχειρηματικό σχέδιο. Αυτές οι πληροφορίες αφορούν διάφορες πτυχές της επιχείρησης, οι οποίες καθορίζονται ρητά από τη δομή που πρέπει να ακολουθεί το συγκεκριμένο έγγραφο. Επίσης, στη συγκεκριμένη εφαρμογή θα διατηρείται αποθηκευμένη οποιαδήποτε πληροφορία έχει εισαχθεί, ώστε να είναι δυνατή η πιθανή μελλοντική προσπέλαση των δεδομένων από τον χρήστη. Καθίσταται, λοιπόν, αναγκαία η ύπαρξη κάποιας μορφής αποθηκευτικού χώρου, όπου θα φυλάγονται όλα τα συσχετιζόμενα δεδομένα. Τα τελευταία, πρέπει να είναι προσβάσιμα από οποιονδήποτε υπολογιστή με σύνδεση στο διαδίκτυο, μιας και η εφαρμογή θα είναι διαδικτυακή, συνεπώς η αποθήκη δεδομένων πρέπει να βρίσκεται κεντρικά και οχι τοπικά.

Στη συγκεκριμένη περίπτωση, αφού απαιτείται η αποθήκευση (πιθανόν) μεγάλου όγκου δεδομένων, τα οποία πρέπει να είναι οργανωμένα, κρίνεται απαραίτητη -όπως αναφέρεται και στο έγγραφο καθορισμού απαιτήσεων της Δέσποινας Βαχάρη- η χρήση βάσης δεδομένων, την υλοποίηση και σχεδιασμό της οποίας ανέλαβαν οι 5 φοιτητές που αναφέρθηκαν παραπάνω. Έτσι, τα οργανωμένα δεδομένα μπορούν να προσπελαστούν πολύ γρήγορα, όπως και να πραγματοποιηθούν με ευκολία οι διαδικασίες αναζήτησης, εισαγωγής, διαγραφής και ενημέρωσης δεδομένων (CRUD functions), διεργασίες που προσφέρουν όλα τα Ολοκληρωμενα Συστήματα Διαχείρισης (Σχεσιακών) Βάσεων Δεδομένων (Relational Database Management System – RDBMS), που αποτελούν ένα interface ανάμεσα στη Βάση Δεδομένων και στον προγραμματιστή. Στην περίπτωση μας, χρησιμοποιήθηκε το ευρέως διαδεδομένο σύστημα διαχείρισης MySQL, που ταιριάζει απόλυτα στις ανάγκες μιας επιχειρηματικής εφαρμογής και διαθέτει ισχυρό support, λόγω της φήμης του.

2. Ομάδα Βάσης Δεδομένων

Συγκεκριμένα, οι παραπάνω πέντε φοιτητές, οι οποίοι συνιστούν την ομάδα της βάσης δεδομένων, είχαν τα παρακάτω καθήκοντα:

- Σχεδιασμός και υλοποίηση Βάσης Δεδομένων
- Κατασκευή Application Programming Interface (API)

Επομένως, έγινε διαχωρισμός σε δύο υπο-ομάδες, η καθεμία από τις οποίες απασχολήθηκε με ένα από τα παραπάνω bullets.

2.1) Ομάδα σχεδιασμού και υλοποίησης Βάσης Δεδομένων

Σε αυτή την ομάδα ανήκουν οι παρακάτω δυο φοιτητές:

- Τσουργιάννης Κωνσταντίνος
- Χριστόπουλος Γιώργος

2.1.1) Σχεδιασμός Βάσης Δεδομένων

Σε πρώτη φάση, κύριο αντικείμενο της συγκεκριμένης ομάδας ήταν ο σχεδιασμός της βάσης δεδομένων (μέσω της κατασκευής του σχεσιακού μοντέλου), ώστε αυτή να συμμορφώνεται με τις καθορισμένες απαιτήσεις.

Σημειώνεται οτι οι παραπάνω φοιτητές εργάστηκαν συνεργατικά στο ίδιο θέμα, χωρίς να διαχωριστεί ο φόρτος στον καθένα ξεχωριστά.

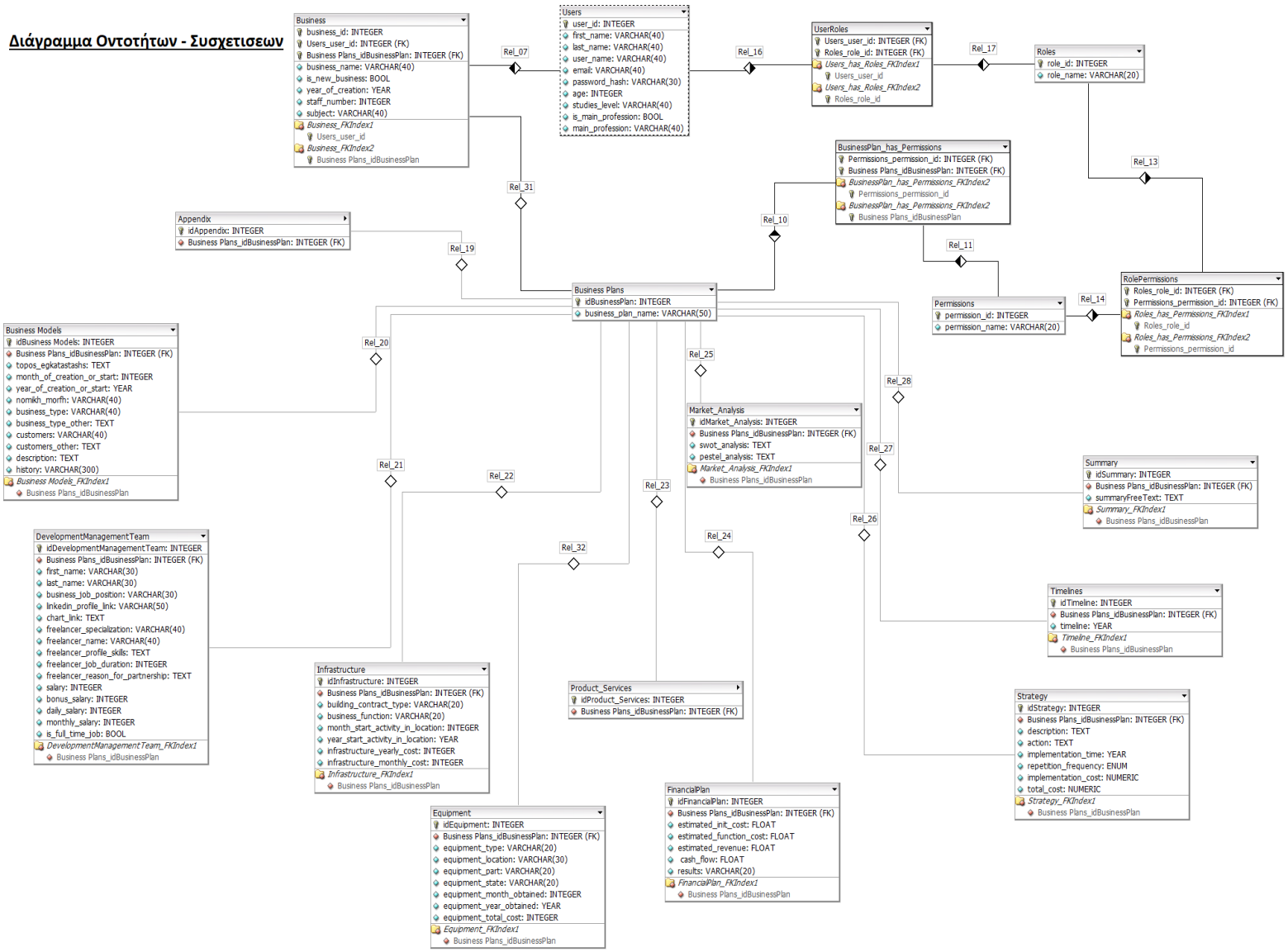
Αρχικά, τα βήματα που ακολουθήθηκαν για το σχεδιασμό της βάσης ήταν τα ακόλουθα:

1. Ανάγνωση του εγγράφου καθορισμού απαιτήσεων, με σκοπό να κατανοηθεί το γενικό πλαίσιο και ο σκοπός ύπαρξης της βάσης δεδομένων.
2. Εύρεση των πληροφοριών που κρίνεται αναγκαίο να αποθηκευτούν στη βάση δεδομένων.
3. Οργάνωση των πληροφοριών που έχουν συλλεχθεί σε πίνακες (tables).
4. Εντοπισμός των χαρακτηριστικών κάθε πίνακα και των κυρίων κλειδιών (primary keys), που χαρακτηρίζουν μοναδικά κάθε εγγραφή – πλειάδα του συγκεκριμένου πίνακα.
5. Καταγραφή των σχέσεων μεταξύ πινάκων, οι οποίες υποδηλώνουν πως υπάρχει κάποια σύνδεση μεταξύ των εγγραφών τους.

Επισημαίνεται, πως λόγω του πρώιμου στάδιου στον οποίο βρισκόταν η πρόοδος της εργασίας στο σύνολο όλων των ομάδων, έγιναν κάποιες (αυθαίρετες) υποθέσεις, σχετικά με τα δεδομένα που θα πρέπει να αποθηκεύονται στη βάση δεδομένων, αλλά και τη μορφή τους.

Συνεπώς, προέκυψε το παρακάτω σχεσιακό μοντέλο, με χρήση του σχεδιαστικού εργαλείου DB Designer Fork:

Διάγραμμα Οντοτήτων - Συσχετισεων



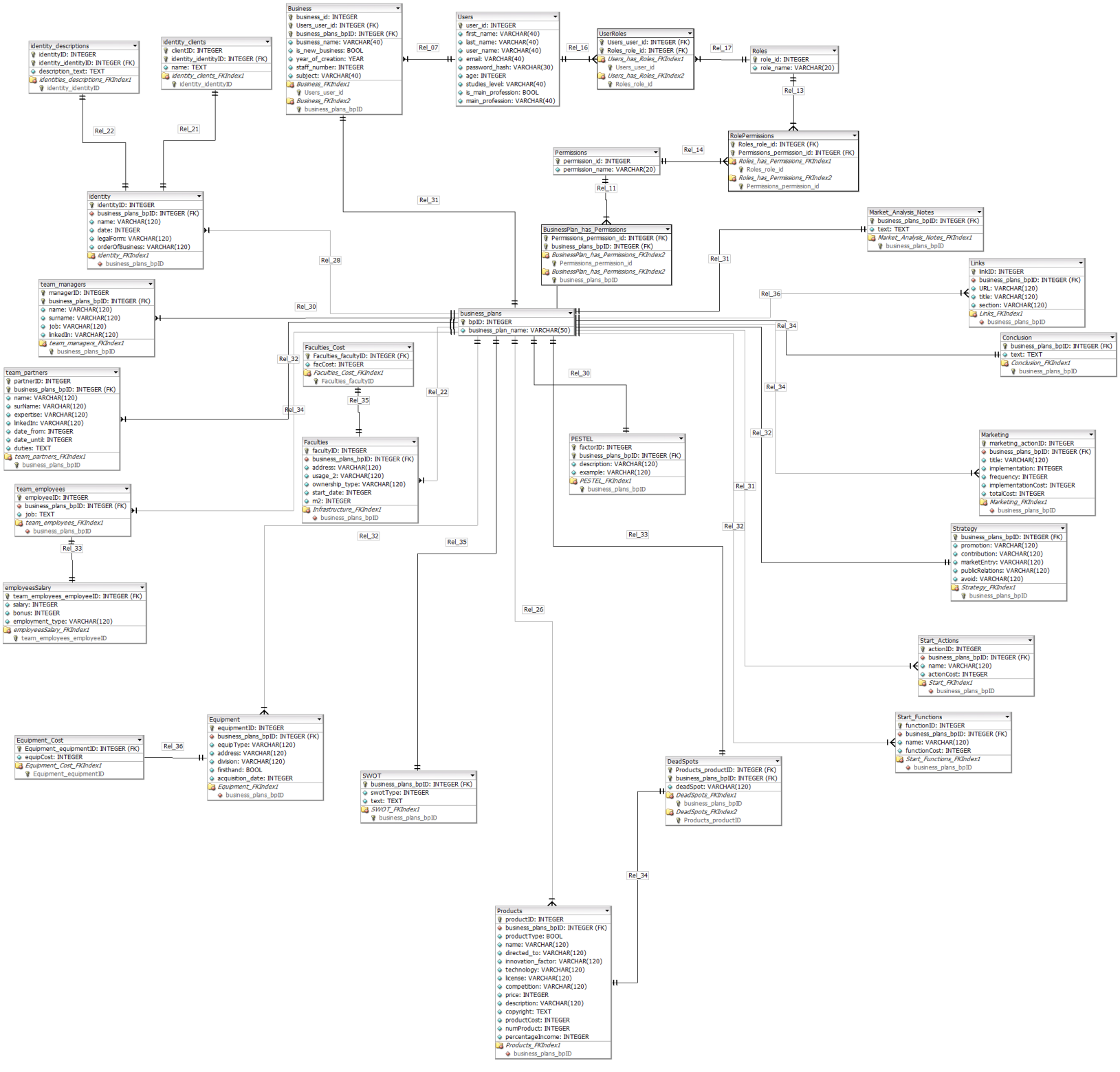
Σε αυτό το σχήμα, το κάθε ορθογώνιο αναπαριστά έναν πίνακα, ενώ τα περιεχόμενα του αναπαριστούν τα χαρακτηριστικά του συγκεκριμένου πίνακα. Επίσης, οι γραμμές δείχνουν τις σχέσεις μεταξύ των πινάκων. Σημειώνεται οτι το notation που χρησιμοποιείται δεν είναι το «επίσημο» (crows foot). Οι λευκοί ρόμβοι αναπαριστούν σχέση πληθικότητας 1:1 (ένα – πρός – ένα). Αντίστοιχα, οι μάυροι – άσπροι ρόμβοι αναπαριστούν σχέση 1:n (ένα – πρός – πολλά, τα πολλά αφορούν την «μαύρη πλευρά»). Για κάθε μια από τις δέκα (10) ενότητες του συστήματος χρησιμοποιήθηκε επι το πλείστον ένας πίνακας.

Παρ'όλα αυτά, στη συνέχεια η ομάδα του front-end δημιούργησε και δημοσίευσε ένα αρχείο μορφής JSON, το οποίο αποτελεί αρχείο κειμένου με συγκεκριμένη

δομή. Μέσω αυτού, παρουσιάζεται η μορφή με την οποία είναι επιθυμητό (από την ομάδα του front-end design) να αποθηκεύονται τα δεδομένα στη βάση δεδομένων.

Προφανώς (λόγω των προαναφερθέντων αυθαίρετων υποθέσεων), υπήρχαν σημαντικές διαφορές με την αρχική σχεδίαση της βάσης δεδομένων, με αποτέλεσμα την ολοκληρωτική ανακατασκευή – επανασχεδιασμό της.

Έτσι, προέκυψε το παρακάτω νεο σχεσιακό μοντέλο:



Παρατηρούμε, οτι και στις δυο παραπάνω περιπτώσεις – σχήματα, έχουν παραμείνει ανέπαφοι οι πίνακες Users, Roles, Permissions καθώς και οι πίνακες που προκύπτουν από τις συσχετίσεις τους. Παρακάτω επισημαίνεται ο λόγος ύπαρξης τους.

Αξίζει να σημειωθεί, λοιπόν, οτι χρησιμοποιήθηκε το μοντέλο Role Based Access Control (RBAC). Αυτό, είναι υπεύθυνο για τον έλεγχο των δικαιωμάτων των χρηστών, με σκοπό την απόρριψη ή όχι ενός αιτήματος τους. Για να επιτευχθεί αυτό, διατηρείται αντιστοιχία μεταξύ χρηστών, ρόλων και δικαιωμάτων τους, ώστε το σύστημα να είναι σε θέση να αναγνωρίζει τον κάθε χρήστη και την ιδιότητα του. Στη συγκεκριμένη υλοποίηση, ο κάθε χρήστης έχει έναν συγκεκριμένο ρόλο (π.χ διαχειριστής, τελικός χρήστης) και ο κάθε ρόλος χαρακτηρίζεται από κάποια δικαιώματα. Συνεπώς, παρέχεται μεγάλη ευελιξία για αλλαγές, αφού προκειμένου να γίνει κάποια αλλαγή στα δικαιώματα ενός συγκεκριμένου ρόλου, αρκεί να μεταβληθεί μόνο ο πίνακας Permissions, χωρίς να υπάρχει ανάγκη επέμβασης στις ιδιότητες του κάθε χρήστη ξεχωριστά. Ομοίως, μπορεί να προστεθεί νέος ρόλος με χαρακτηριστικη ευκολία. Η αντιστοίχηση μεταξυ χρηστών – ρόλων διατηρείται στον πίνακα UserRoles, ενώ μεταξύ ρόλων – δικαιωμάτων στον πίνακα RolePermissions. Έτσι, μεταβατικά προκύπτουν οι ιδιότητες των χρηστών.

Η γενική ιδέα στην οποία στηρίζονται τα παραπάνω σχήματα, είναι οτι στο κέντρο βρίσκεται το επιχειρηματικό σχέδιο – business plan. Αυτό, αποτελείται από τις 10 ενότητες, οι οποίες σχετίζονται με όλους τους πίνακες εκτός αυτών που αφορούν το μοντέλο RBAC.

2.1.2) Υλοποίηση Βάσης Δεδομένων

Το επόμενο βήμα, μετά τη σχεδίαση της βάσης δεδομένων, ήταν η κατασκευή μιας βάσης δεδομένων που θα περιέχει τους προαναφερθέντες πίνακες και τα αντίστοιχα χαρακτηριστικά, δηλαδή η μετατροπή του σχεσιακού μοντέλου σε πραγματική βάση δεδομένων. Αρχικά (όπως αναφέρθηκε και στην εισαγωγή), υπήρξε η ιδέα της κεντρικής διατήρησης της βάσης δεδομένων (σε κάποια κεντρική υποδομή), ώστε να υπάρχει καθολική πρόσβαση μέσω διαδικτύου, γεγονός που θα προσέφερε ευελιξία σε όλες τις συνεργαζόμενες ομάδες. Παρ’όλα αυτά, χάριν ευκολίας, πάρθηκε η απόφαση να διατηρηθεί τοπικά (στους

υπολογιστές των μελών των ομάδων), ώστε να υπάρχει η δυνατότητα αυτόνομων δοκιμών, χωρίς να επηρεάζεται μεγάλη ομάδα χρηστών.

Συνεπώς, κατασκευάστηκε ένα DDL Script δημιουργίας βάσης, δηλαδή ένα τμήμα κώδικα που περιέχει εντολές SQL σε μορφή MySQL. Αυτό, με την εκτέλεση του, κατασκευάζει αυτόματα όλα τα στοιχεία της βάσης (πίνακες, χαρακτηριστικά πινάκων, κύρια και δευτερεύοντα κλειδιά και περιορισμούς), το οποίο διαμοιράστηκε με όλα τα μέλη της εργασίας(*). Έτσι, υπάρχει η δυνατότητα της κατασκευής της βάσης τοπικά, απλα με την εκτέλεση του παραπάνω script.

(*) Πρίν τη δημοσίευση του script, αυτό δοκιμάστηκε στο περιβάλλον MySQL Workbench, όπου δημιουργήθηκε ορθώς η αναμενόμενη βάση δεδομένων, χωρίς λάθη.

2.2) Ομάδα υλοποίησης Application Programming Interface (API) και σύνδεσης με τη Βάση Δεδομένων

Σε αυτή την ομάδα ανήκουν οι παρακάτω φοιτητές:

- Βενέτης Θοδωρής
- Τζιλής Αντώνης
- Ναλπαντίδης Νίκος

Κύριο αντικείμενο της συγκεκριμένης ομάδας ήταν η δημιουργία του Application Programming Interface της εφαρμογής, το οποίο είναι υπεύθυνο για την επικοινωνία μεταξύ του back-end (της βάσης δεδομένων) και του front-end του λογισμικού. Αποτελείται από ένα σύνολο λειτουργιών, οι οποίες δίνουν τη δυνατότητα πρόσβασης στα δεδομένα της βάσης.

Για την οργάνωση του κώδικα του API, χρησιμοποιήθηκε η αρχιτεκτονική Model – View – Controller (MVC 4.0).

Σύμφωνα με αυτό το μοντέλο, υπάρχουν 3 μεγάλες κατηγορίες:

1. Models

Τα μοντέλα είναι υπεύθυνα για τη διατήρηση των πληροφοριών της εφαρμογής (στη συγκεκριμένη περίπτωση, αντλούν πληροφορίες από τη βάση δεδομένων).

Ανταποκρίνονται σε:

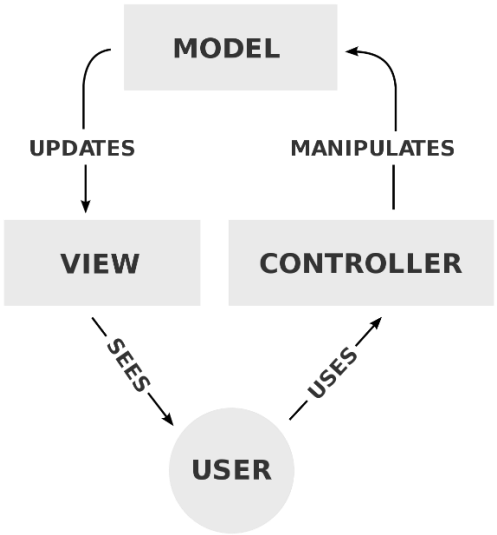
- Αιτήματα από τα Views.
- Εντολές του Controller, ώστε να πραγματοποιήσουν κάποια αλλαγή στα ίδια τους τα δεδομένα.

2. Views:

Οι όψεις είναι υπεύθυνες για την παρουσίαση των πληροφοριών των μοντέλων, σε συγκεκριμένη μορφή.

3. Controllers:

Οι ελεκτές δέχονται κάποια μορφή εισόδου και πραγματοποιούν κάποια αλλαγή στα αντικείμενα των μοντέλων.



Και αυτή η ομάδα χωρίστηκε σε δύο υπο-ομάδες, ώστε να γίνει καλύτερος καταμερισμός του φόρτου εργασίας, στις εξής ομάδες:

- Ομάδα υλοποίησης σύνδεσης με τη Βάση Δεδομένων
- Application Programming Interface (API)

2.2.1) Ομάδα υλοποίησης σύνδεσης με τη Βάση Δεδομένων

Σε αυτή την ομάδα συμμετείχαν οι παρακάτω φοιτητές:

- Τζιλής Αντώνης
- Ναλπαντίδης Νίκος

Παρακάτω περιγράφεται αναλυτικά το αντικείμενο της συγκεκριμένης ομάδας.

Services

Η υλοποίηση τών services έγινε με χρήση του εργαλείου Visual Studio (C#). Services είναι οι λειτουργίες που παρέχονται στήν βάση, όπως:

- Εισαγωγή δεδομένων στον πίνακα
- Διαγραφή δεδομένων από τον πίνακα
- Επιστροφή όλων των στοιχείων του πίνακα
- Επιστροφή συγκεκριμένου στοιχείου από τον πίνακα

Η κάθε μία από τις παραπάνω λειτουργίες περιγράφεται από μία συνάρτηση, η οποία δημιουργεί ένα SQL Query (αποθηκεύεται σε μεταβλητή τύπου συμβολοσειράς). Η συγκεκριμένη μεταβλητή, παρέχεται ως είσοδος σε μία άλλη συνάρτηση, η οποία το εκτελεί το Ερώτημα SQL και επεξεργάζεται τα δεδομένα που αυτό επιστρέφει.

Επίσης, υλοποιήθηκαν οι κλάσεις:

- ActionCost_Service
- RunningCost_Service
- SalaryCost_Service.

Αυτές, με ένα query επιλέγουν όλες τις εγγραφές του αντιστοίχου πίνακα και αθροίζουν το κόστος τους, το οποίο τελικά επιστρέφουν.

Σημείωση: Πριν ολοκληρωθεί η βάση, οι δοκιμές έγιναν πάνω σε dummy database (βάση δεδομένων, η οποία δε περιέχει χρήσιμες πληροφορίες και χρησιμοποιείται για δοκιμαστικούς σκοπούς).

Αρχικά, η ομάδα ακολούθησε την τεχνική pair programming (ο ένας προγραμματιστής -ο οδηγός (driver)- κωδικοποιεί και ο άλλος -ο πλοηγός (navigator)- ελέγχει τον κώδικα συνεχώς, για την αποφυγή λαθών και προτείνει λύσεις για την ανάπτυξη), ενώ όταν ολοκληρώθηκε η βάση οι φοιτητές εργάστηκαν ανεξάρτητα.

Στον παρακάτω πίνακα παρουσιάζεται ο καταμερισμός της συγγραφής των κλάσεων από τους φοιτητές της ομάδας:

Νίκος Ναλπαντίδης	**Αντώνης Τζιλής**
ActionCost_Service	Function_Service
Client_Service	IdentityClient_Service
Conclusion_Service	Identity_Service
Deadspot_Service	Income_Service
Description_Service	Link_Service
EmployeeSalary_Service	Manager_Service
Employee_Service	MarketingAction_Service
EquipmentCost_Service	Note_Service
Equipment_Service	Partner_Service
Factor_Service	Product_Service
Faculty_Service	StartAction_Service
FunctionCost_Service	Strategy_Service
RunningCost_Service	Swot_Service
SalaryCost_Service	

2.2.2) Application Programming Interface (API)

Σε αυτή την ομάδα συμμετείχε ο παρακάτω φοιτητής:

- Βενέτης Θεόδωρος

Models

Το παραπάνω project δημιουργήθηκε για χρήση, ως εξωτερική βιβλιοθήκη η οποία μπορεί να «ενώνει» τα επιμέρους κομμάτια του backend και περιλαμβάνει όλες τις custom κλάσεις που χρησιμοποιούνται για την εισαγωγή και εξαγωγή των στοιχείων. Με αυτόν τον τρόπο, τα άτομα που ασχολήθηκαν με την σύνδεση της βάσης γνώριζαν ποιά δεδομένα και με ποιά μορφή να ανακτήσουν και τα άτομα του API ποιά δεδομένα είναι διαθέσιμα και πώς μπορούν να τα χρησιμοποιήσουν.

Οι κλάσεις αυτές δημιουργήθηκαν σύμφωνα με τις ανάγκες του front-end και την μορφή των πινάκων στην βάση δεδομένων (οι οποίοι με την σειρά τους πηγάζουν απο τις ανάγκες των διάφορων σημείων της εφαρμογής). Υπάρχουν κλάσεις που απαντούν απευθείας σε συγκεκριμένα σημεία της εφαρμογής όπως οι Employee, Conclusion ή Faculty και άλλες, που χρησιμοποιούνται ώς ενδιάμεσες σε περιπτώσεις που χρειάζονται παραπάνω πληροφορίες όπως FacultyCost, EquipmentCost, ή RunningCost.

Στο σύνολο τους αποτελούνται απο τα επιμέρους πεδία που αντιστοιχούν σε στήλες των αντίστοιχων πινάκων στην βάση καθώς και μία βοηθητική συνάρτηση για αντιγραφή όλων των πεδίων ίδιου αντικειμένου.

Κατα την διάρκεια του project, όπως και άλλα κομμάτια έτσι και στα μοντέλα προέκυψαν αλλαγές προκειμένου να ανταπεξέλθουν στα ζητήματα που προέκυψαν.

Τέλος ,τα μοντέλα που δημιουργήθηκαν, αποτελούν και τα τελικά Serialized JSON Objects, τα οποία προσφέρονται κατα το request απο το REST API .Σχετικά με αυτήν την μορφή, εφόσον τα διακριτά κομμάτια front-end και back-end εργάζονταν ταυτόχρονα, η μέχρι στιγμής μορφή αυτών υπήρχε πάντα αναρτημένη σε αντίστοιχο αρχείο στον φάκελο της ομάδας στο GitHub.

REST API

Το κομμάτι του API κατασκευάστηκε με χρήση Model-View-Controller framework (έκδοση 4.0 του ASP.NET) καθώς ταίριαζε περισσότερο στην περίπτωση .Με χρήση των μοντέλων απο πρίν και δεδομένου οτι στην συγκεκριμένη περίπτωση δεν υπήρχε “View” κομμάτι ,το μόνο που έμενε να υλοποιηθεί ήταν οι controllers.

Κάθε controller απαντάει στα διάφορα πιθανά http requests που μπορούν να σταλούν στο API ,κάνοντας χρήση των διαφόρων services που δημιουργήθηκαν στο back-end προκειμένου να ετοιμάσει και να «επιστρέψει» το αντίστοιχο response .

Συγκεκριμένα υποστηρίζονται τα ακόλουθα calls :

- GET api_site/resource_name : όπου επιστρέφεται ένας πίνακας με όλες τις εγγραφές στην που υπάρχουν στην βάση για το αντίστοιχο resource .
- GET api_site/resource_name/id : όπου επιστρέφεται ένας πίνακας με όλα τα resource που αναφέρονται στο business plan με το δοσμένο id.(Για όποια resource δεν αναφέρονται σε business plan το id απαντάει στο id του ίδιου του resource)
- POST api_site/resource_name : όπου δίνεται ένα αντικείμενο σε μορφή Json του αντίστοιχου resource και σε περίπτωση που αυτό εισαχθεί με επιτυχία στην βάση επιστρέφεται ώς response το αντίστοιχο νέο αντικείμενο με όλα τα πεδία του .
- PUT api_site/resource_name/id : όπου δίνεται πάλι ένα αντικείμενο σε μορφή Json του resource και γίνεται αλλαγή των πεδίων του αντικειμένου που έχει ως id το δοσμένο με τα πεδία που υπάρχουν στο αντικείμενο της εισόδου .Σε περίπτωση επιτυχίας αλλαγής των πεδίων ,επιστρέφεται το νέο,αλλαγμένο αντικείμενο .
- DELETE api_site_resource_name/id : όπου διαγράφεται το αντικείμενο του οποίου το id συμφωνεί με το δοσμένο και σε περίπτωση επιτυχίας επιστρέφεται το διεγραμένο στοιχείο με όλα τα πεδία του .

Σε περίπτωση αποτυχίας των παραπάνω ,επιστρέφεται κενό response .

Τα παραπάνω ελέγχθηκαν με ξεχωριστές εφαρμογές παραγωγής requests και κατα την δημιουργία τους έγιναν αρκετές διαφορές προκειμένου να μπορέσουν να ικανοποιήσουν τις ανάγκες της εφαρμογής .

Το τελικό API έχει «ανέβει» ,τουλάχιστον προσωρινά ,στον ιστότοπο play-trinity.com/theo/bplantool όπου προκειμένου π.χ. να γίνει κλήση για το resource “manager” αυτό θα πρέπει να ανακατευθυνθεί στο : play-trinity.com/theo/bplantool/api/manager.

Επίσης ,υπάρχει σχετικός οδηγός στον φάκελο του backend του project στο github προκειμένου να μπορέσουν όλοι οι ενδιαφερόμενοι να «στήσουν» τοπικά το API και να δούν πώς αυτό δουλεύει καθώς και τα διακριτά σημεία του που ενδεχομένως στο τελικό αποτέλεσμα εμφανίζονται ως ενα .

3.Επικοινωνία με την κοινότητα του έργου

Για την επικοινωνία με την κοινότητα του έργου, χρησιμοποιήθηκε το διαδικτυακό μέσο επικοινωνίας “Slack”, το οποίο χρησιμοποιήθηκε κατά κόρον για την αλληλεπίδραση όλων των μελών της εργασίας.

Σε αυτό, η κάθε ομάδα διατηρούσε τον δικό της «χώρο» - chat-room, όπου μπορούσε να συντονίσει τις εργασίες της. Επίσης, υπήρξε και επικοινωνία και με μέλη άλλων ομάδων -κυρίως του front-end- τα οποία μας παρείχαν χρήσιμες πληροφορίες και ανατροφοδότηση σχετικά με το σχεδιασμό της βάσης δεδομένων.

Επίσης, υπήρχε ένα chat-room το οποίο λειτουργούσε σαν πίνακας ανακοινώσεων. Σε αυτό, κάθε μήνυμα απευθυνόταν σε όλα τα μέλη του επιχειρηματικού σχεδίου και χρησιμοποιήθηκε για τον συντονισμό μεταξύ ομάδων. Με άλλα λόγια, βοήθησε να παρθούν αποφάσεις που αφορούσαν το έργο σαν σύνολο, όπως τον τρόπο υλοποίησης (για παράδειγμα, προτιμήθηκαν οι γλώσσες HTML, CSS, JavaScript για το front-end και MySQL, C# για το back-end αντί για Desktop Application με Java για το front-end, σύνδεση με την βάση δεδομένων μέσω της βιβλιοθήκης JDBC (Java DataBase Connector,την οποία δυστιχώς υλοποιήσαμε σε ένα σημείο μέχρι που τελικά άλλαξε ο τρόπος υλοποίησης και υλοποίηση της βάσης δεδομένων σε MySQL).

Για τον διαμοιρασμό του κώδικα και γενικότερα χρήσιμων πληροφοριών, τόσο εντός της Ομάδας Βάσης Δεδομένων, όσο και με τις υπόλοιπες ομάδας, χρησιμοποιήθηκε το εργαλείο GitHub, το οποίο αποτελεί λογισμικό Version Control, ώστε να υπάρχει πλήρης έλεγχος των διαμοιραζόμενων δεδομένων.

Επιπροσθέτως, για την επικοινωνία μεταξύ των μελών της ομάδας της Βάσης Δεδομένων, εκτός του slack χρησιμοποιήθηκε η εφαρμογή διαδικτυακών κλήσεων Skype, αλλά και πραγματοποιήθηκαν δια ζώσης συναντήσεις, με σκοπό την καλύτερη και πιο άμεση επικοινωνία.

Τέλος για τον συντονισμό του back-end με το front-end, υπήρξε επικοινωνία με τον συντονιστή της εκάστοτε ομάδας, για να κατανοήσουμε καλύτερα τις απαιτήσεις, καθώς και τις αλλαγές αυτών με την πάροδο της υλοποίησης του έργου.

4. Τελικά συμπεράσματα

Μέσω της φετινής εργασίας, όλοι οι φοιτητές είχαν την ευκαιρία να εργαστούν σε ομάδες, λύνοντας επιμέρους προβλήματα και συνθέτοντας τις λύσεις τους, ώστε να καταλήξουν σε ένα τελικό συνολικό αποτέλεσμα. Ήταν μια πρωτόγνωρη εμπειρία, αλλά και πολύ σημαντική, αφού εφαρμόζεται σε όλους τους επαγγελματικους κλάδους της επιστήμης της πληροφορικής. Συνεπώς, όλοι ήρθαν, κάτα κάποιον τρόπο, σε επαφή με τον «επαγγελματικό» τρόπο εργασίας. Παρ'όλα τα προβλήματα που παρουσιάστηκαν, κυρίως λόγω απειρίας, αξίζει να σημειωθεί πως υπήρξε επιτυχής επικοινωνία μεταξύ 11 ομάδων. Η επικοινωνία μεταξύ των μελών της ομάδας βάσης δεδομένων ήταν ικανοποιητική. Φυσικά υπήρξαν και δυσκολίες, λόγω της εκμάθησης και χρήσης καινούριων γλωσσών προγραμματισμού ή τεχνικών (όπως και στις περισσότερες ομάδες), άλλα σίγουρα θα αποτελέσουν σημαντική εμπειρία για το μέλλον.

ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ