Τεχνολογία Λογισμικού
Τομέας Ηλεκτρονικής και Υπολογιστών
Τμήμα ΗΜΜΥ
Α.Π.Θ.

8ο Εξάμηνο
Άνοιξη 2017

Σύστημα *Γραμματεία*

Student life made simpler

Σχεδίαση Συστήματος

Del.3.1

Έκδοση 0.1
(draft)

Διαμαντή Μαρία mfdiamanti@ece.auth.gr

Μπέκος Χριστόφορος mpekchri@ece.auth.gr

Ντζιώνη Δήμητρα dntzioni@ece.auth.gr

Πάχος Νικόλαος npaschos@ece.auth.gr

24/5/2017

Ιστορικό Αλλαγών

Όνομα	Ημ/νία	Περιγραφή Αλλαγής	Εκδ.
Α. Συμεωνίδης	29/05/2009	Δημιουργία Εγγράφου Προσαρμογή του ESA software engineering standards guidelines (1991) και του εγγράφου SDD document, από τους Bruegge και Dutoit (2004).	0.1
Ομάδα DIAS	18/05/2017	Συγγραφή κεφαλαίου 1.	0.2
Ομάδα DIAS	19/05/2017	Προσθήκη τμημάτων στην παράγραφο 2.1.	0.3
Ομάδα DIAS	20/05/2017	Ολοκλήρωση παραγράφου 2.1 και προσθήκη του πίνακα πρόσβασης της παραγράφου 2.3.	0.4
Ομάδα DIAS	21/05/2017	Συγγραφή παραγράφου 2.2 και προσθήκη διαγράμματος ανάπτυξης.	0.5
Ομάδα DIAS	22/05/2017	Συγγραφή κεφαλαίου 3.	0.6
Ομάδα DIAS	23/05/2017	Τελική μορφοποίηση εγγράφου και προσθήκη Πίνακα Ιχνηλασιμότητας και παραρτήματος Ανοιχτών Θεμάτων.	0.7

Μέλη Ομάδας Ανάπτυξης

Όνομα	ΟΑ	Email
Διαμαντή Μαρία	Ομάδα DIAS	mfdiamanti@ece.auth.gr
Μπέκος Χριστόφορος	Ομάδα DIAS	mpekchri@ece.auth.ge
Ντζιώνη Δήμητρα	Ομάδα DIAS	dntzioni@ece.auth.gr
Πάσχος Νικόλαος	Ομάδα DIAS	npaschos@ece.auth.gr

Πίνακας Περιεχομένων

Λίστα Σχημάτων

1. Τρέχουσα Αρχιτεκτονική Λογισμικού

Το παρόν κεφάλαιο έχει ως στόχο να παρουσιάσει μια σειρά από υποψήφιες αρχιτεκτονικές λογισμικού, οι οποίες θα μπορούσαν να υιοθετηθούν κατά τη σχεδίαση του παρόντος συστήματος γραμματείας. Έπειτα από μελέτη των προτερημάτων και των μειονεκτημάτων, τα οποία φέρει η καθεμία από αυτές, γίνεται η επιλογή της/των αρχιτεκτονικών λογισμικού με τρόπο ώστε να ικανοποιούνται οι απαιτήσεις του συστήματος που σχεδιάζουμε όπως τις έχουμε ορίσει σε προηγούμενα παραδοτέα.

1.1 Αρχιτεκτονική Πελάτη – Διακομιστή (Client – Server Architecture)

Η αρχιτεκτονική λογισμικού Πελάτη – Διακομιστή είναι μια συνήθης αρχιτεκτονική, κατά την οποία οι πελάτες (clients) καλούν τους διακομιστές (servers), ζητώντας κάποιον πόρο, και οι τελευταίοι επιστρέφουν τον πόρο αυτό, αφού πρώτα εκτελέσουν την απαιτούμενη υπηρεσία. Πρόκειται για ένα μοντέλο το οποίο εφαρμόζεται σε συστήματα βάσεων δεδομένων και αποτελεί μία από τις μεθόδους ανάπτυξης και λειτουργίας κατανεμημένων συστημάτων. Υπό αυτήν την έννοια, θεωρούμε τον πελάτη και το διακομιστή ξεχωριστά συστήματα, τα οποία επικοινωνούν μέσω ενός δικτύου σύνδεσης. Αυτός είναι και ο λόγος που αναφέρεται και ως αρχιτεκτονική δύο επιπέδων (2 tier).

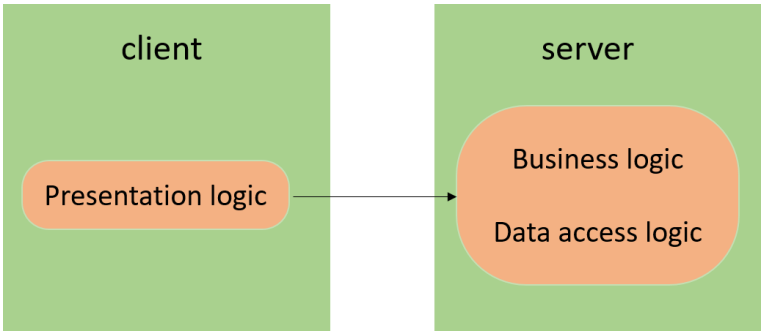
Συγκεκριμένα, ένας πελάτης έχει τη δυνατότητα, μέσω του γραφικού περιβάλλοντος της εφαρμογής, να δημιουργήσει αιτήσεις για πληροφορίες και να τις αποστείλει στο διακομιστή, να επεξεργαστεί δεδομένα στο front – end και να έχει πρόσβαση στη βάση δεδομένων μέσω του δικτύου. Από την άλλη πλευρά, ένας διακομιστής είναι σε συνεχή λειτουργία προκειμένου να λαμβάνει και να διαχειρίζεται αιτήματα για πληροφορίες από περισσότερους του ενός πελάτες. Επιπλέον, είναι υπεύθυνος για την ακεραιότητα των δεδομένων, αλλά και την ασφάλεια και τη συνεκτικότητα της βάσης δεδομένων.

Τα κύρια χαρακτηριστικά του μοντέλου αρχιτεκτονικής λογισμικού πελάτη – διακομιστή είναι τα εξής:

- **Υψηλή ασφάλεια (security)**: Όλα τα δεδομένα είναι αποθηκευμένα στο διακομιστή, ο οποίος προσφέρει μεγαλύτερο έλεγχο από ότι ένα σύστημα πελάτη.
- **Εύκολη πρόσβαση και ενημέρωση δεδομένων**: Το γεγονός πως όλα τα δεδομένα είναι αποθηκευμένα στο ίδιο μέρος επιτρέπει ευκολότερη πρόσβαση σε αυτά, αλλά και ευκολότερη ενημέρωση αυτών, συγκριτικά με άλλα μοντέλα αρχιτεκτονικής.
- **Εύκολη συντήρηση**: Το γεγονός πως ο πελάτης και ο διακομιστής είναι ξεχωριστά συστήματα συνεπάγεται ότι οποιαδήποτε συντήρηση από πλευράς του διακομιστή δεν πρόκειται να επηρεάσει τον πελάτη και αντίστροφα.
- **Ευελιξία:** Το μοντέλο αυτό είναι κατάλληλο για εφαρμογές οι οποίες πρέπει να υποστηρίζουν διαφορετικά είδη πελατών και τελικών συσκευών.

Ωστόσο, χρήζει ιδιαίτερης προσοχής κατά την εφαρμογή του, καθώς έχει τις εξής ιδιαιτερότητες:

- **Πρόβλημα υπολογιστικής ισχύος**: Όσον αφορά την υπολογιστική του ισχύ, θα πρέπει να διαθέτει περίσσευμα πόρων, έτσι ώστε ανά πάσα στιγμή να είναι σε θέση να εξυπηρετήσει αιτήματα μεγαλύτερου πλήθους πελατών. Συνεπώς, η ορθή επιλογή του κατάλληλου διακομιστή είναι υψίστης σημασίας.
- **Πρόβλημα επεκτασιμότητας**: Επιπρόσθετα, το γεγονός ότι τα δεδομένα των εφαρμογών είναι συχνά στενά συνδεδεμένα με το business logic τμήμα, επηρεάζει αρνητικά την επεκτασιμότητα του συστήματος, με αποτέλεσμα την ανάγκη εξέλιξης της αρχιτεκτονικής αυτής σε αρχιτεκτονική τριών επιπέδων (3 tier).



Εικόνα 1.1 Μοντέλο αρχιτεκτονικής Πελάτη – Διακομιστή

1.2 Αρχιτεκτονική τριών επιπέδων (3 tier)

Η αρχιτεκτονική λογισμικού τριών επιπέδων είναι ένα μοντέλο το οποίο περιγράφει το διαχωρισμό των λειτουργικοτήτων Presentation logic, Business logic και Data access logic σε τρία διαφορετικά επίπεδα (tiers), με κύριο χαρακτηριστικό το γεγονός πως καθένα επίπεδο μπορεί να βρίσκεται σε ένα φυσικά ξεχωριστό υπολογιστικό σύστημα. Ίδια ακριβώς είναι και η λογική των αρχιτεκτονικών n επιπέδων γενικότερα.

Κάθε επίπεδο είναι εντελώς ανεξάρτητο από όλα τα υπόλοιπα, πέραν του προηγούμενου και του επόμενου επιπέδου από αυτό. Πιο συγκεκριμένα, αν θεωρήσουμε ένα επίπεδο n, αυτό πρέπει να γνωρίζει πώς να διαχειρίζεται ένα αίτημα από το n+1 επίπεδο και στη συνέχεια να μεταβιβάζει το αίτημα αυτό στο n-1 επίπεδο. Επιπλέον, πρέπει να είναι σε θέση να γνωρίζει πώς να διαχειρίζεται τα αποτελέσματα ενός αιτήματος. Η επικοινωνία μεταξύ των επιπέδων είναι συνήθως ασύγχρονη για την καλύτερη λειτουργία του συστήματος. Όσον αφορά την αρχιτεκτονική τριών επιπέδων, αποτελείται από τα εξής επίπεδα:

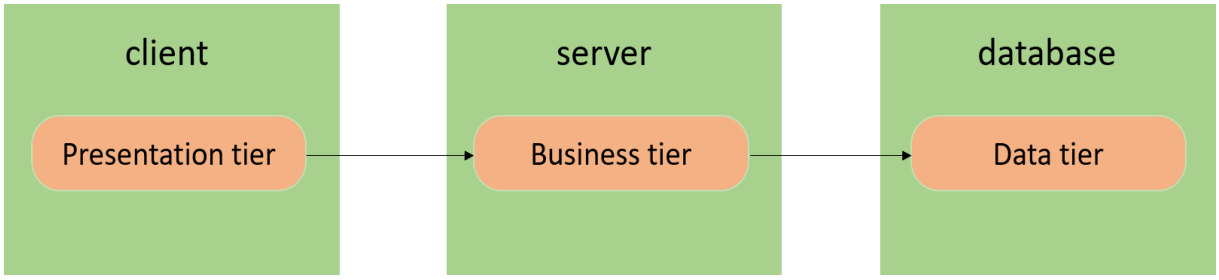
Presentation tier: Αφορά την απεικόνιση των αντικειμένων και διαφόρων άλλων πληροφοριών στην οθόνη του χρήστη, υπό τη μορφή ενός γραφικού περιβάλλοντος με το οποίο μπορεί να αλληλοεπιδρά ο χρήστης.

Business tier: Αφορά το επίπεδο που λαμβάνει αιτήματα μέσω του Presentation tier από το χρήστη, επεξεργάζεται τα αιτήματα αυτά, διαχειρίζεται τα απαιτούμενα δεδομένα κάνοντας χρήση του Data tier και τελικά, εμφανίζει το αποτέλεσμα της επεξεργασίας στον τελικό χρήστη.

Data tier: Αφορά όλα εκείνα τα δεδομένα με τα οποία αλληλοεπιδρά ο χρήστης, δηλαδή τα δεδομένα τα οποία μεταφέρονται μεταξύ του presentation tier και του business tier και προβάλλονται στο χρήστη, είτε άλλα δεδομένα απαιτούμενα, προκειμένου να φέρει εις πέρας το business tier τις υπηρεσίες που του ζητούνται.

Το μοντέλο αρχιτεκτονικής τριών επιπέδων επιλύει τα προβλήματα της προηγούμενης αρχιτεκτονικής, όπως προαναφέρθηκε. Τα κύρια προτερήματά του είναι τα ακόλουθα:

- **Εύκολη συντήρηση**: Το γεγονός ότι κάθε επίπεδο είναι ανεξάρτητο από τα υπόλοιπα, επιτρέπει την πραγματοποίηση ενημερώσεων χωρίς να επηρεαστεί ολόκληρο το σύστημα.
- **Επεκτασιμότητα**: Ο διαχωρισμός των λειτουργικοτήτων σε διαφορετικά επίπεδα επιτρέπει την ευκολότερη επέκταση καθενός από αυτά χωριστά.
- **Ευελιξία**: Όλα τα παραπάνω οδηγούν γενικότερα στη σχεδίαση ενός ευέλικτου συστήματος.
- **Διαθεσιμότητα**: Αυτός ο αρθρωτός τρόπος σχεδίασης λογισμικού επιτρέπει την επέκταση των τμημάτων λογισμικού, με αποτέλεσμα να αυξάνεται η διαθεσιμότητα της εφαρμογής γενικότερα.



Εικόνα 1.2 Μοντέλο αρχιτεκτονικής τριών επιπέδων (σε διαφορετικά φυσικά συστήματα)

1.3 Αρχιτεκτονική Μοντέλο – Όψη – Ελεγκτής (MVC)

Το παρόν μοντέλο είναι ένα από τα πιο συχνά χρησιμοποιούμενα μοντέλα αρχιτεκτονικής λογισμικού στη βιομηχανία, προκειμένου να δημιουργηθούν εύκολα επεκτάσιμα έργα λογισμικού. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη, έτσι ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στο χρήση από την μορφή που έχει αποθηκευτεί αυτή στο σύστημα. Συγκεκριμένα, τα μέρη αυτά είναι, όπως υποδηλώνει και το όνομά του, το μοντέλο (model), η όψη (view) και ο ελεγκτής (controller).

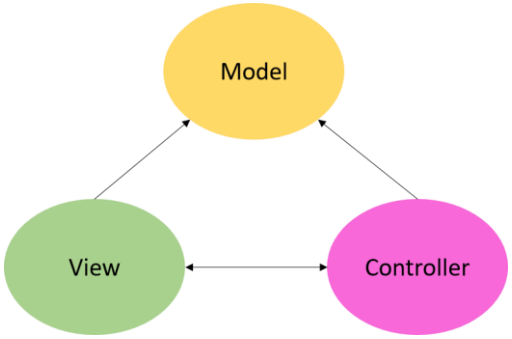
***Μοντέλο (model)*:** Σχετίζεται με όλα εκείνα τα δεδομένα με τα οποία αλληλοεπιδρά ο χρήστης, δηλαδή με δεδομένα τα οποία μεταφέρονται μεταξύ της όψης και του ελεγκτή και προβάλλονται στο χρήστη, είτε με άλλα δεδομένα απαιτούμενα, προκειμένου να φέρει εις πέρας ο ελεγκτής τις υπηρεσίες που του ζητούνται.

***Όψη (view)*:** Είναι υπεύθυνη για την απεικόνιση των αντικειμένων και διαφόρων άλλων πληροφοριών στην οθόνη του χρήστη, υπό τη μορφή ενός γραφικού περιβάλλοντος με το οποίο μπορεί να αλληλοεπιδρά ο χρήστης.

***Ελεγκτής (controller)*:** Λειτουργεί ως μία διασύνδεση μεταξύ της όψης και του μοντέλου. Είναι υπεύθυνος για την λήψη αιτημάτων που λαμβάνει μέσω της όψης από το χρήστη, την επεξεργασία των αιτημάτων αυτών, τη διαχείριση των απαιτούμενων δεδομένων κάνοντας χρήση του τμήματος Μοντέλο και τελικά την εμφάνιση του αποτελέσματος της επεξεργασίας στον τελικό χρήστη.

Το μοντέλο αυτό, πέραν του διαχωρισμού των διαφορετικών λειτουργικοτήτων σε τρία χωριστά τμήματα, ορίζει ακριβώς και τον τρόπο με τον οποίο αυτά επικοινωνούν μεταξύ τους. Αυτό είναι και το σημείο στο οποίο διαφοροποιείται αρκετά από τις αρχιτεκτονικές που αναλύθηκαν προηγουμένως. Πιο συγκεκριμένα, ο ελεγκτής μπορεί να στέλνει εντολές τόσο στο μοντέλο, έτσι ώστε να ενημερώσει την κατάστασή του, όσο και στην όψη, προκειμένου να ενημερώσει και αυτή την πληροφορία που προβάλλεται στο χρήστη. Επιπλέον, το μοντέλο είναι και αυτό σε θέση να στέλνει εντολές προς την όψη έπειτα από κάθε ανανέωση του περιεχομένου του, ώστε να προβληθεί η ανανεωμένη αυτή πληροφορία και στην οθόνη. Τέλος, η όψη μπορεί να στέλνει αιτήματα του χρήστη για επεξεργασία από τον ελεγκτή και να δέχεται εντολές από τα δύο άλλα τμήματα, όπως αναφέρθηκε.

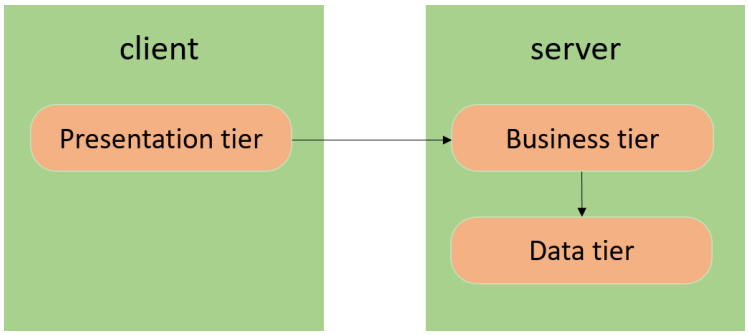
Όπως και το μοντέλο αρχιτεκτονικής τριών επιπέδων χαρακτηρίζεται από **ευελιξία**, **επεκτασιμότητα**, **εύκολη συντήρηση** και **διαθεσιμότητα** λόγω του διαχωρισμού των λειτουργικοτήτων σε χωριστά επίπεδα, έτσι και το παρόν μοντέλο διακρίνεται από τα χαρακτηριστικά αυτά, δεδομένου ότι οι λειτουργικότητες είναι τώρα χωρισμένες σε διαφορετικά τμήματα.



Εικόνα 1.3 Μοντέλο αρχιτεκτονικής Μοντέλο – Όψη – Ελεγκτής

1.4 Αρχιτεκτονική Συστήματος Γραμματείας – Instasis

Τα τρία μοντέλα αρχιτεκτονικής που αναπτύχθηκαν νωρίτερα ήταν τα κύρια υποψήφια μοντέλα, τα οποία θα μπορούσαν να υιοθετηθούν για το σχεδιασμό του Συστήματος Γραμματείας της εφαρμογής Instasis. Έπειτα από μελέτη αυτών, η αρχιτεκτονική που θα χρησιμοποιηθεί είναι ο **συνδυασμός της αρχιτεκτονικής Πελάτη – Διακομιστή και της αρχιτεκτονικής τριών επιπέδων**. Ο συνδυασμός αυτός μας επιτρέπει να διατηρήσουμε το μοντέλο Πελάτη – Διακομιστή, αλλά να διαχωρίσουμε εντός αυτού τις λειτουργικότητες Presentation logic, Business logic και Data access logic σε τρία χωριστά επίπεδα (tiers), προκειμένου να επιλύσουμε το πρόβλημα της επεκτασιμότητας. Με αυτόν τον τρόπο, το σύστημά μας θα διαθέτει όλα τα πλεονεκτήματα των δύο αυτών αρχιτεκτονικών, δηλαδή υψηλή ασφάλεια, ευελιξία, ανεξαρτησία μεταξύ των επιπέδων, επεκτασιμότητα, διαθεσιμότητα, ευκολία συντήρησης και ευκολία πρόσβασης και ενημέρωσης των δεδομένων. Η παρακάτω εικόνα παρουσιάζει το συνδυασμό των αρχιτεκτονικών αυτών γραφικά, ώστε να γίνει απόλυτα κατανοητός.



Εικόνα 1.4 Αρχιτεκτονική Συστήματος Γραμματείας – Instasis

Το κάθε ένα από τα τρία αυτά επίπεδα έχει τις εξής αρμοδιότητες:

Presentation tier: Αποτελείται από το σύνολο των γραφικών διεπαφών που προβάλλονται κάθε φορά στο χρήστη. Βρίσκονται από φυσικής πλευράς στη φορητή συσκευή του χρήστη, δέχονται εισόδους μέσω της οθόνης αφής, αποστέλλουν τα αντίστοιχα αιτήματα στον server και έπειτα λαμβάνουν εντολές για την απεικόνιση του αποτελέσματος της επεξεργασίας του αιτήματος.

Business tier: Περιλαμβάνει τους ελεγκτές του συστήματος, οι οποίοι είναι υπεύθυνοι για την επεξεργασία των δεδομένων, την εκτέλεση των αιτούμενων λειτουργιών και την αποστολή των αποτελεσμάτων στο χρήστη. Βρίσκεται από φυσικής πλευράς στον server της εφαρμογής, έτσι ώστε να μην επιβαρύνεται η συσκευή του χρήστη.

Data tier: Αποτελείται από τους εντολοδόχους (proxies) των βάσεων δεδομένων, οι οποίοι επικοινωνούν με τις βάσεις δεδομένων και τους θέτουν τα κατάλληλα ερωτήματα εγγραφής και ανάγνωσης. Βρίσκεται από φυσικής πλευράς στον server της εφαρμογής.

2. Προτεινόμενη Αρχιτεκτονική Λογισμικού

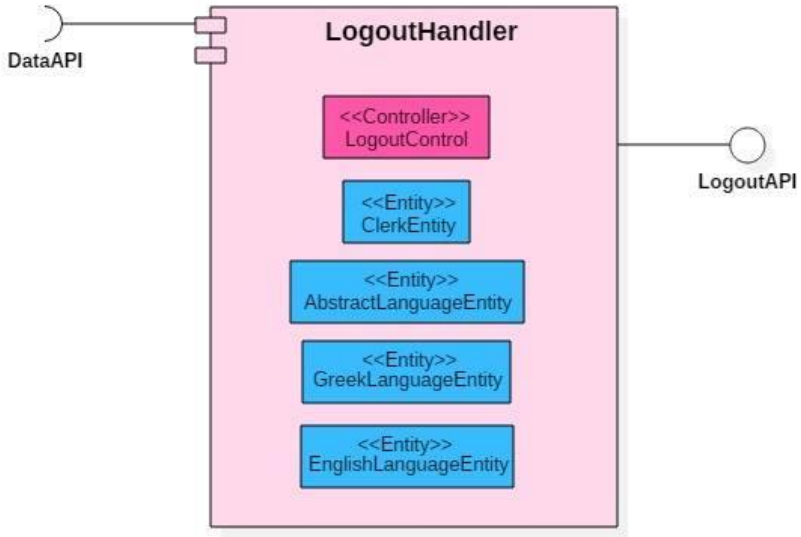
Το παρόν κεφάλαιο περιγράφει το μοντέλο σχεδίασης του συστήματος γραμματείας της εφαρμογής instasis.

2.1 Αποδόμηση Συστήματος

Στην παράγραφο αυτή περιγράφεται η αποδόμηση του συστήματος, δηλαδή ο διαχωρισμός του συστήματος σε επιμέρους υποσυστήματα ανάλογα με τις λειτουργίες και τις αρμοδιότητες καθενός από αυτά. Κάθε υποσύστημα αποτελείται ένα τμήμα (component), μέσα στο οποίο έχουν ομαδοποιηθεί κλάσεις του συστήματος που είχαν άμεση συσχέτιση, όπως αυτές αναπτύχθηκαν στο δεύτερο παραδοτέο. Η αποδόμηση έγινε με κριτήριο την ύπαρξη **υψηλής συνεκτικότητας** και **χαμηλής σύζευξης** μεταξύ των κλάσεων κάθε τμήματος, έτσι ώστε αλλαγές σε κάποια κλάση κάποιου τμήματος να μην επηρεάζουν το υπόλοιπο σύστημα, ή στην έσχατη να το επηρεάζουν κατά το ελάχιστο δυνατό.

2.1.1 Υποσύστημα LogoutHandler

Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται για την αποσύνδεση της γραμματείας από το σύστημα και την προβολή της σελίδας σύνδεσης, εφόσον αυτή αποσυνδεθεί. Γενικότερα, διαχειρίζεται κάποια απαραίτητα προσωπικά στοιχεία που χαρακτηρίζουν τη γραμματεία, αλλά και τη γλώσσα του συστήματος. Αποτελείται από τον ελεγκτή LogoutControl και τις οντότητες ClerkEntity, Language, GreekLanguageEntity και EnglishLanguageEntity. Επικοινωνεί με το υποσύστημα DataHandler μέσω της διεπαφής DataAPI, με απώτερο σκοπό την επικοινωνία με τη βάση δεδομένων του instasis-backend, όπου είναι αποθηκευμένα τα στοιχεία της γραμματείας, αλλά και στοιχεία που αφορούν τη γλώσσα του συστήματος. Επιπλέον, επικοινωνεί με το υποσύστημα LogoutInterface μέσω της διεπαφής LogoutAPI, παρέχοντάς του την πληροφορία που πρέπει να εμφανιστεί στην οθόνη.

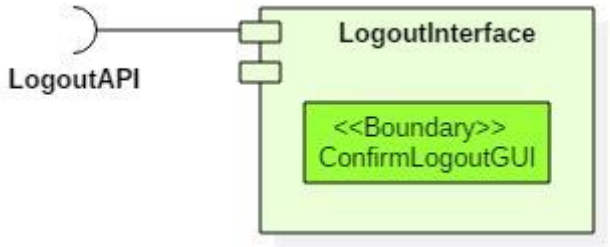


Εικόνα 2.1 Υποσύστημα LogoutHandler

Σημειώνεται πως οι κλάσεις οντοτήτων AbstractLanguageEntity, GreekLanguageEntity και EnglishLanguageEntity προστέθηκαν έπειτα από αλλαγή του δεύτερου παραδοτέου. Παρακαλούμε να ανατρέξετε στον Πίνακα Ιχνηλασιμότητας.

2.1.2 Υποσύστημα LogoutInterface

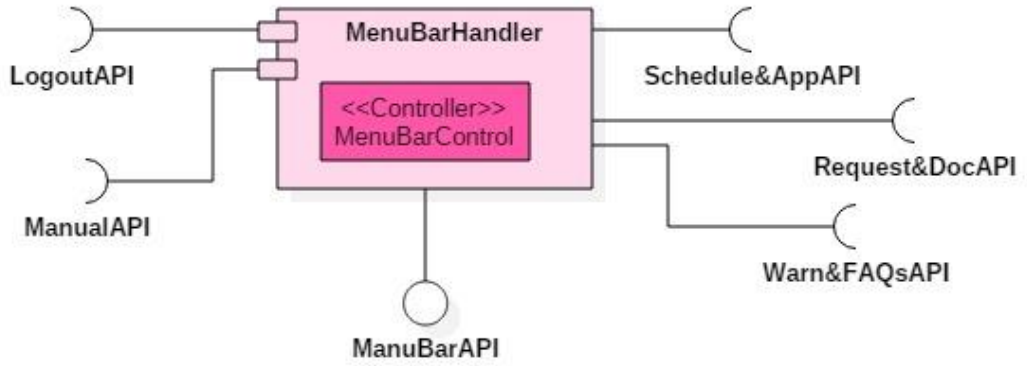
Το υποσύστημα αυτό περιλαμβάνει την γραφική διεπαφή μέσω της οποίας η γραμματεία μπορεί να επιβεβαιώσει ή όχι την αποσύνδεσή της από το σύστημα. Αποτελείται από την οριακή κλάση ConfirmLogoutGUI και επικοινωνεί με το υποσύστημα LogoutHandler μέσω του LogoutAPI, απαιτώντας από αυτό την πληροφορία που θα εμφανίσει στην οθόνη του.



Εικόνα 2.2 Υποσύστημα LogoutInterface

2.1.3 Υποσύστημα MenuBarHandler

Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται για την προβολή του μενού στο επάνω μέρος της οθόνης καθεμίας γραφικής διεπαφής. Μέσω του μενού αυτού επιτυγχάνεται η πλοήγηση στο σύνολο των σελίδων της εφαρμογής, η αλλαγή γλώσσας από ελληνικά σε αγγλικά και το αντίστροφο και τέλος, η αποσύνδεση της γραμματείας από το σύστημα. Αποτελείται αποκλειστικά από τον ελεγκτή MenuBarControl. Επιπλέον, επικοινωνεί με το υποσύστημα MenuBarInterface μέσω της διεπαφής MenuBarAPI προκειμένου να ανταποκριθεί σε κάποια ενέργεια της γραμματείας. Αξίζει να σημειωθεί πως το υποσύστημα αυτό επικοινωνεί και με τα υποσυστήματα HelpHandler (μέσω του ManualAPI), LogoutHandler (μέσω του LogoutAPI), ScheduleHandler (μέσω του Schedule&AppAPI), Request&DocHandler (μέσω του Request&DocAPI) και Warnings&FAQsHandler (μέσω του Warn&FAQsAPI). Ουσιαστικά, επικοινωνεί με τα υποσυστήματα που βρίσκονται στο ίδιο επίπεδο (tier) με αυτό, προκειμένου να αναθέσει στο αρμόδιο υποσύστημα μια υπηρεσία, έπειτα από το πάτημα κάποιου συνδέσμου στο μενού.

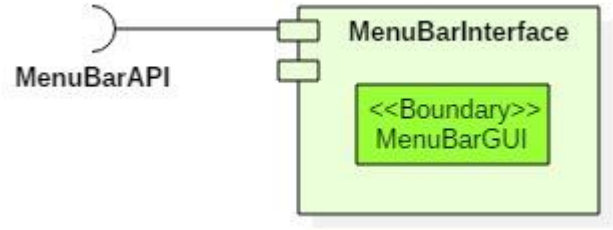


Εικόνα 2.3 Υποσύστημα MenuBarHandler

2.1.4 Υποσύστημα MenuBarInterface

Το υποσύστημα αυτό περιλαμβάνει την γραφική διεπαφή που εμφανίζει το μενού συνδέσμων στο επάνω μέρος της οθόνης, σε κάθε γραφικό περιβάλλον το οποίο το απαιτεί. Περιλαμβάνει την

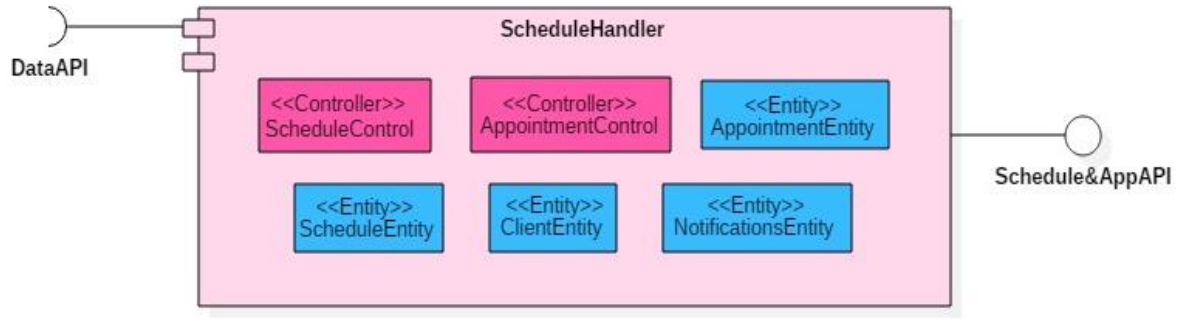
οριακή κλάση MenuBarGUI και επικοινωνεί με το υποσύστημα MenuBarHandler μέσω του MenuBarAPI, απαιτώντας από αυτό την πληροφορία που θα εμφανίσει στην οθόνη.



Εικόνα 2.4 Υποσύστημα MenuBarInterface

2.1.5 Υποσύστημα ScheduleHandler

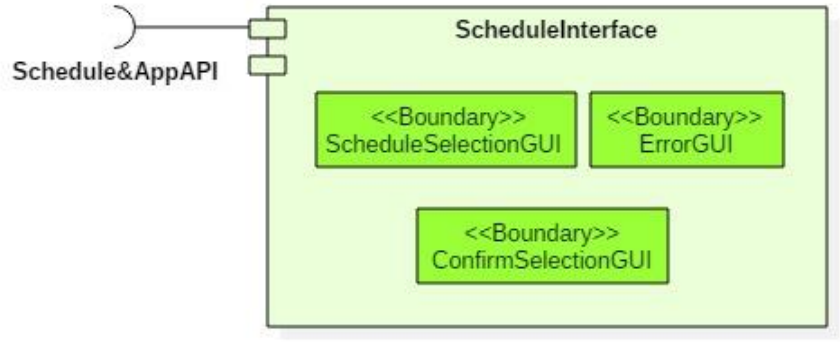
Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται προκειμένου η γραμματεία να μπορεί να βλέπει τα ημερήσια προγράμματα, να επιλέγει ένα εξ' αυτών για εξυπηρέτηση, να μεταθέτει ή να ακυρώνει ένα ή πολλά ραντεβού και να αποστέλλει ειδοποίηση προς το instasis-backend, με απώτερους παραλήπτες τους αιτούμενους των ραντεβού που μετατέθηκαν ή ακυρώθηκαν. Επιπλέον, εμπεριέχει την κατάλληλη λειτουργικότητα έτσι ώστε να μπορεί να βλέπει τις πληροφορίες καθενός αιτούμενου και επίσης, να ενημερώνει το σύστημα για την εξέλιξη του τρέχοντος ραντεβού. Αποτελείται από τους ελεγκτές ScheduleControl και AppointmentControl και από τις οντότητες ScheduleEntity, AppointmentEntity, ClientEntity και NotificationsEntity. Επικοινωνεί με το υποσύστημα DataHandler μέσω της διεπαφής DataAPI, προκειμένου είτε να φορτώσει από τη βάση δεδομένων του instasis-backend ένα ή περισσότερα ημερήσια προγράμματα, είτε να αποθηκεύσει σε αυτήν κάποιο τυχόν τροποποιημένο πρόγραμμα. Επιπλέον, επικοινωνεί με τα υποσυστήματα ScheduleInterface και AppointmentInterface μέσω της διεπαφής Schedule&AppAPI, παρέχοντάς του την πληροφορία που πρέπει να εμφανιστεί στην οθόνη.



Εικόνα 2.5 Υποσύστημα ScheduleHandler

2.1.6 Υποσύστημα ScheduleInterface

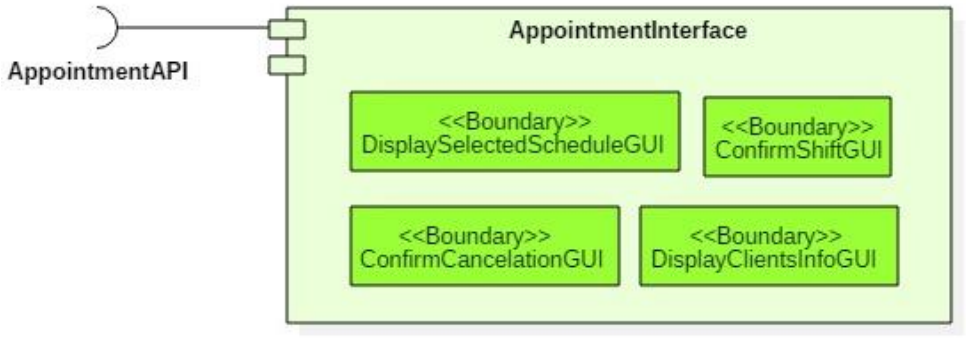
Το υποσύστημα αυτό περιλαμβάνει τις γραφικές διεπαφές οι οποίες είναι απαραίτητες προκειμένου η γραμματεία να μπορεί να δει όλα τα ημερήσια προγράμματα και να επιλέξει ένα εξ' αυτών προς εξυπηρέτηση, να επιβεβαιώσει την ενέργεια επιλογής, καθώς και να λάβει τυχόντα μηνύματα σφάλματος κατά την επιλογή αυτή. Περιλαμβάνει τις οριακές κλάσεις ScheduleSelectionGUI, ErrorGUI καθώς και την ConfirmSelectionGUI. Επικοινωνεί με το υποσύστημα ScheduleHandler μέσω του Schedule&AppAPI, απαιτώντας από αυτό την πληροφορία που θα εμφανιστεί στην αντίστοιχη γραφική διεπαφή.



Εικόνα 2.6 Υποσύστημα ScheduleInterface

2.1.7 Υποσύστημα AppointmentInterface

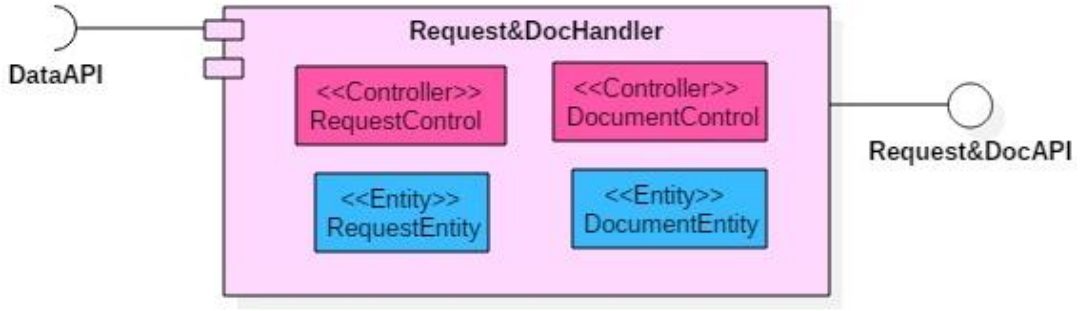
Το υποσύστημα αυτό περιλαμβάνει τις γραφικές διεπαφές οι οποίες είναι απαραίτητες προκειμένου η γραμματεία να μπορεί να δει το επιλεγμένο πρόγραμμα προς εξυπηρέτηση, να μεταθέσει ή να ακυρώσει ένα ή περισσότερα ραντεβού και να επιβεβαιώσει την ενέργεια αυτή, καθώς και να προβάλλει τις πληροφορίες ενός αιτούμενου. Περιλαμβάνει τις οριακές κλάσεις DisplaySelectedScheduleGUI, ConfirmShiftGUI, ConfirmCancelationGUI καθώς και την DisplayClientsInfoGUI. Επικοινωνεί με το υποσύστημα ScheduleHandler μέσω του Schedule&AppAPI, απαιτώντας από αυτό την πληροφορία που θα εμφανιστεί στην αντίστοιχη γραφική διεπαφή.



Εικόνα 2.7 Υποσύστημα AppointmentInterface

2.1.8 Υποσύστημα Request&DocHandler

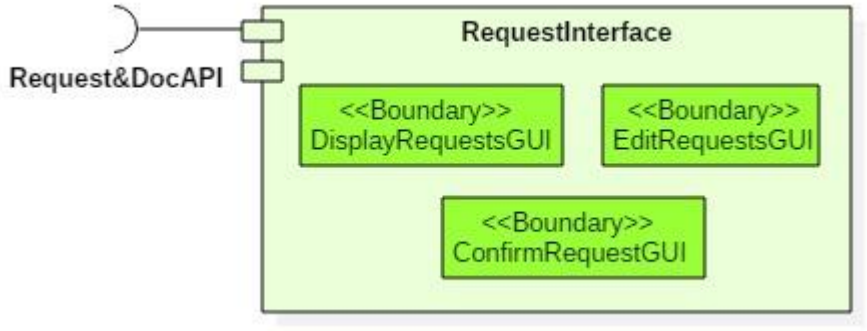
Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται προκειμένου η γραμματεία να βλέπει τα αιτήματα που εξυπηρετεί και να τροποποιεί τυχόν χαρακτηριστικά τους, όπως τη θεματολογία, τον τρόπο εξυπηρέτησης, τον εκτιμώμενο χρόνο και τα απαιτούμενα έγγραφα. Επιπλέον, εμπεριέχει και τη λειτουργικότητα του να βλέπει τα απαιτούμενα έγγραφα και να τροποποιεί τυχόν χαρακτηριστικά αυτών, όπως το όνομά τους ή ολόκληρο το pdf αυτών. Αποτελείται από τους ελεγκτές RequestControl και DocumentControl και από τις οντότητες RequestEntity και DocumentEntity. Επικοινωνεί με το υποσύστημα DataHandler μέσω της διεπαφής DataAPI, προκειμένου είτε να φορτώσει από τη βάση δεδομένων του instasis-backend αιτήματα ή έγγραφα, είτε να αποθηκεύσει τα τυχόν τροποποιημένα αιτήματα ή έγγραφα. Επιπλέον, επικοινωνεί με τα υποσυστήματα RequestInterface και DocumentInterface μέσω του Request&DocAPI, παρέχοντάς τους την πληροφορία που πρέπει να εμφανιστεί στην οθόνη.



Εικόνα 2.8 Υποσύστημα Request&DocHandler

2.1.9 Υποσύστημα RequestInterface

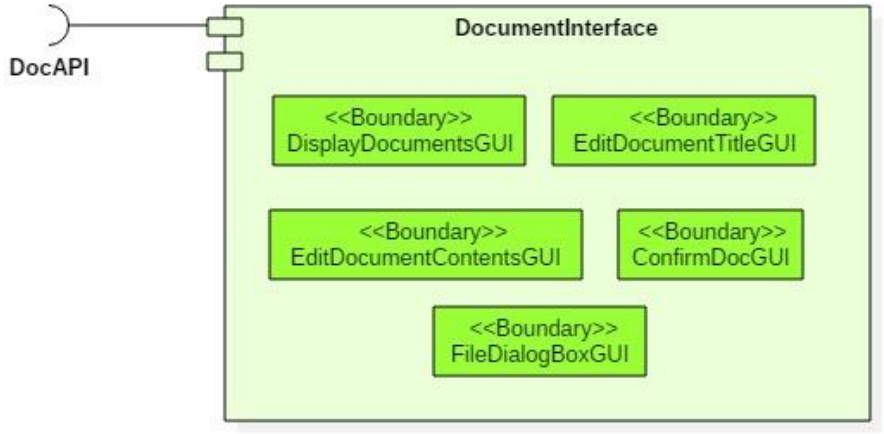
Το υποσύστημα αυτό αφορά τις γραφικές διεπαφές οι οποίες είναι απαραίτητες προκειμένου η γραμματεία να μπορεί να δει τα υπάρχοντα αιτήματα, να τροποποιήσει κάποιο εξ'αυτών, καθώς επίσης και να επιβεβαιώσει τυχόν αλλαγές στα περιεχόμενα ενός αιτήματος, που ο ίδιος επεξεργαζόταν. Περιλαμβάνει τις οριακές κλάσεις DisplayRequestsGUI, EditRequestsGUI και ConfirmRequestGUI και επικοινωνεί με το υποσύστημα Request&DocHandler μέσω του Request&DocAPI, απαιτώντας από αυτό την πληροφορία που πρέπει να εμφανίσει στην αντίστοιχη γραφική διεπαφή.



Εικόνα 2.9 Υποσύστημα RequestInterface

2.1.10 Υποσύστημα DocumentInterface

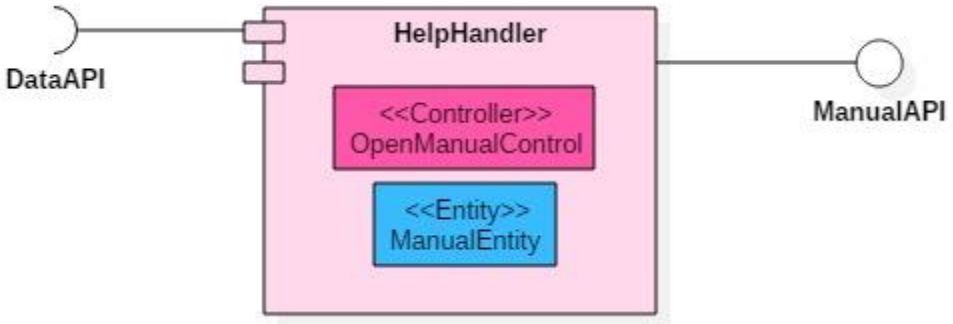
Το υποσύστημα αυτό αφορά τις γραφικές διεπαφές οι οποίες είναι απαραίτητες προκειμένου η γραμματεία να μπορεί να δει τα υπάρχοντα έγγραφα, να τροποποιήσει κάποιο εξ' αυτών, καθώς επίσης και να επιβεβαιώσει τις τυχούσες αυτές αλλαγές που προέκυψαν έπειτα από την τροποποίηση. Περιλαμβάνει τις οριακές κλάσεις DisplayDocumentsGUI, EditDocumentTitleGUI, EditDocumentContentsGUI, ConfirmDocGUI και FileDialogBoxGUI. Επιπλέον, επικοινωνεί με το υποσύστημα DocumentHandler μέσω της διεπαφής DocAPI, απαιτώντας από αυτό την πληροφορία που πρέπει να εμφανιστεί στην αντίστοιχη γραφική διεπαφή.



Εικόνα 2.10 Υποσύστημα DocumentInterface

2.1.11 Υποσύστημα HelpHandler

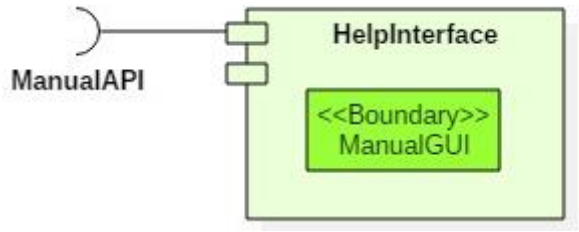
Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται προκειμένου η γραμματεία να μπορεί να βλέπει τον οδηγό χρήσης. Αποτελείται από τον ελεγκτή OpenManualControl και την οντότητα ManualEntity. Επικοινωνεί με το υποσύστημα DataHandler μέσω της διεπαφής DataAPI, προκειμένου να φορτώσει από τη βάση δεδομένων του instasis-backend τον οδηγό χρήσης και έπειτα να τον προβάλει στην οθόνη. Επιπλέον, επικοινωνεί με το υποσύστημα HelpInterface, παρέχοντάς του την πληροφορία που πρέπει να εμφανιστεί στην οθόνη.



Εικόνα 2.11 Υποσύστημα HelpHandler

2.1.12 Υποσύστημα HelpInterface

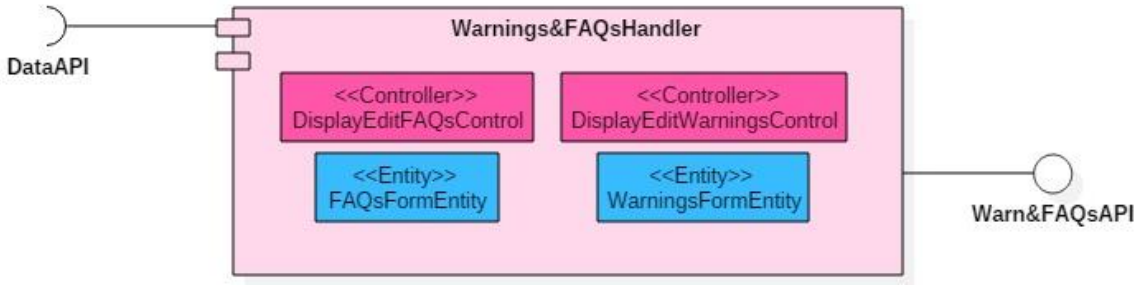
Το υποσύστημα αυτό περιλαμβάνει τη γραφική διεπαφή για την εμφάνιση της σελίδα «Οδηγός χρήσης». Αποτελείται αποκλειστικά από την οριακή κλάση ManualGUI και επικοινωνεί με το υποσύστημα HelpHandler μέσω του ManualAPI. Μέσω της επικοινωνίας αυτής απαιτεί την πληροφορία που πρέπει να εμφανίσει στην οθόνη.



Εικόνα 2.12 Υποσύστημα HelpInterface

2.1.13 Υποσύστημα Warnings&FAQsHandler

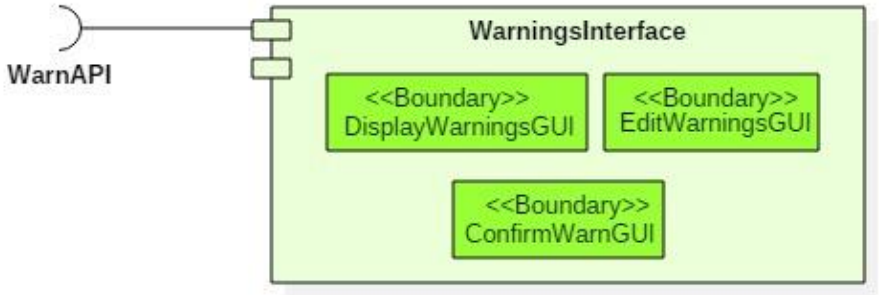
Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται προκειμένου η γραμματεία να βλέπει τις προειδοποιήσεις και τις συχνές ερωτήσεις, να τροποποιεί καθεμία από αυτές και τέλος, να αποθηκεύει τις αλλαγές. Αποτελείται από τους ελεγκτές DisplayEditWarningsControl και DisplayEditFAQsControl και από τις οντότητες WarningsFormEntity και FAQsFormEntity. Επικοινωνεί με το υποσύστημα DataHandler μέσω της διεπαφής DataAPI, έτσι ώστε να είτε φορτώσει από τη βάση δεδομένων του instasis-backend τη φόρμα των προειδοποιήσεων ή των συχνών ερωτήσεων, είτε να αποθηκεύσει στη βάση δεδομένων την ανανεωμένη φόρμα προειδοποιήσεων ή συχνών ερωτήσεων, σε περίπτωση που η γραμματεία προχωρήσει στην τροποποίηση αυτής. Επιπλέον, επικοινωνεί με τα υποσυστήματα WarningsInterface και FAQsInterface μέσω του Warn&FAQsAPI, παρέχοντάς τους την πληροφορία που πρέπει να εμφανιστεί στην οθόνη.



Εικόνα 2.13 Υποσύστημα Warnings&FAQsHandler

2.1.14 Υποσύστημα WarningsInterface

Το υποσύστημα αυτό περιλαμβάνει τις οριακές κλάσεις DisplayWarningsGUI, EditWarningsGUI και ConfirmWarnGUI, οι οποίες αποτελούν τις διεπαφές μέσω των οποίων η γραμματεία μπορεί να βλέπει τις προειδοποιήσεις, να τις τροποποιεί και να αποθηκεύει τις αλλαγές, αφού προηγουμένως επιβεβαιώσει την ενέργειά της αυτή. Προκειμένου να επιτευχθούν οι παραπάνω ενέργειες, επικοινωνεί με το υποσύστημα Warnings&FAQsHandler μέσω της διεπαφής Warn&FAQsAPI, απαιτώντας από αυτό την απαραίτητη πληροφορία που καλείται να εμφανίσει στην οθόνη.

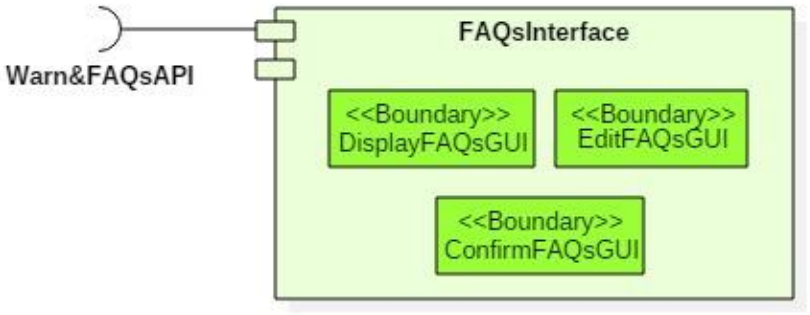


Εικόνα 2.14 *Υποσύστημα WarningsInterface*

2.1.15 Υποσύστημα FAQsInterface

Το υποσύστημα αυτό περιλαμβάνει τις γραφικές διεπαφές οι οποίες είναι απαραίτητες προκειμένου ο χρήστης να μπορεί να δει τις συχνές ερωτήσεις, να τις τροποποιήσεις, καθώς και να επιβεβαιώσει την διαδικασία τροποποίησης. Περιλαμβάνει τις οριακές κλάσεις DisplayFAQsGUI, EditFAQsGUI καθώς και την ConfirmFAQsGUI. Επικοινωνεί με το Warnings&FAQsInterface μέσω του

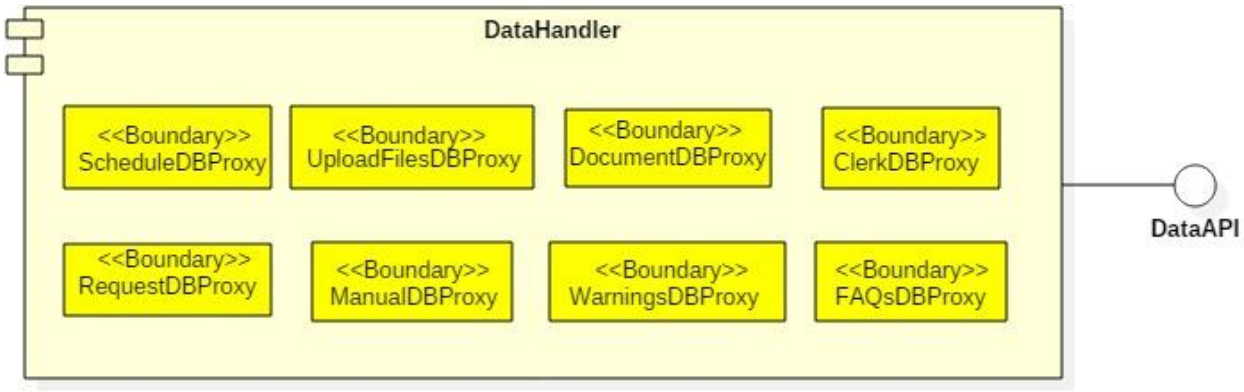
Warn&FAQsAPI, απαιτώντας από αυτό την πληροφορία που θα εμφανιστεί στην αντίστοιχη γραφική διεπαφή.



Εικόνα 2.15 Υποσύστημα FAQsInterface

2.1.16 Υποσύστημα DataHandler

Όλα τα παραπάνω υποσυστήματα, τα οποία προσφέρουν τις διάφορες υπηρεσίες που απαιτεί η γραμματεία, απαιτούν επικοινωνία με τη βάση δεδομένων του instasis-backend. Με αυτόν τον τρόπο, αντλούν τα δεδομένα που χρειάζονται προκειμένου να παράσχουν μια υπηρεσία. Πιο συγκεκριμένα, η βάση δεδομένων του instasis-backend εμπεριέχει δεδομένα τα οποία σχετίζονται με την ίδια τη γραμματεία (ClerkDBProxy), τα προγράμματα με τα ραντεβού τα οποία πρέπει να εξυπηρετήσει η γραμματεία (ScheduleDBProxy), τα αιτήματα που εξυπηρετούνται από αυτήν (RequestDBProxy), καθώς και τα απαραίτητα έγγραφα ενός αιτήματος (DocumentDBProxy). Επιπλέον, εμπεριέχει δεδομένα που αφορούν τις προειδοποιήσεις (WarningsDBProxy) και τις συχνές ερωτήσεις (FAQsDBProxy) που συντάσσει η γραμματεία, αλλά και τον οδηγό χρήσης (ManualDBProxy) στον οποίο έχει πρόσβαση αυτή. Η πρόσβαση στα δεδομένα αυτά επιτυγχάνεται μέσω των οριακών αντικειμένων τα οποία αναφέρθηκαν.



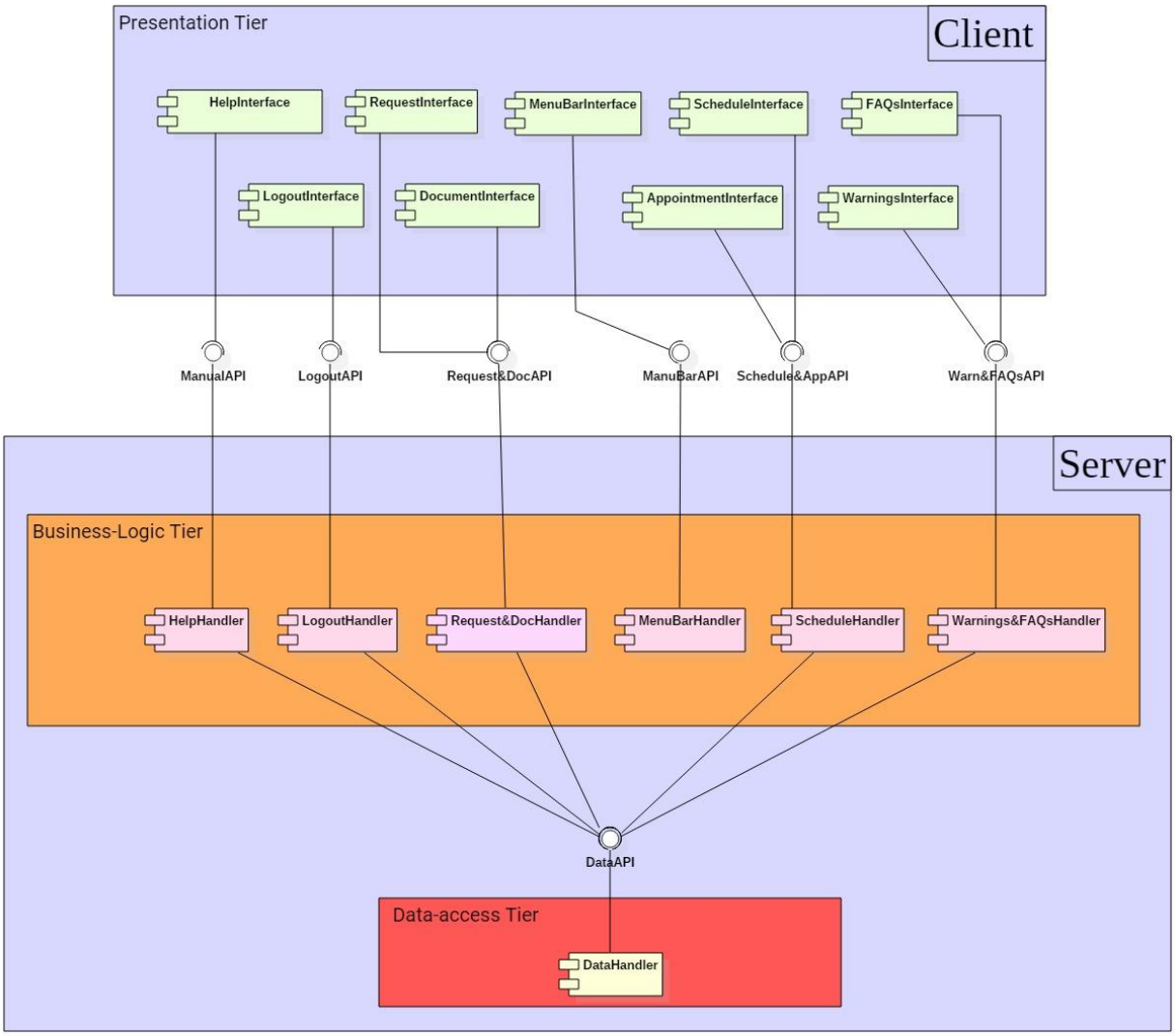
Εικόνα 2.16 Υποσύστημα Data Handler

2.1.17 Διάγραμμα Τμημάτων (Component Diagram)

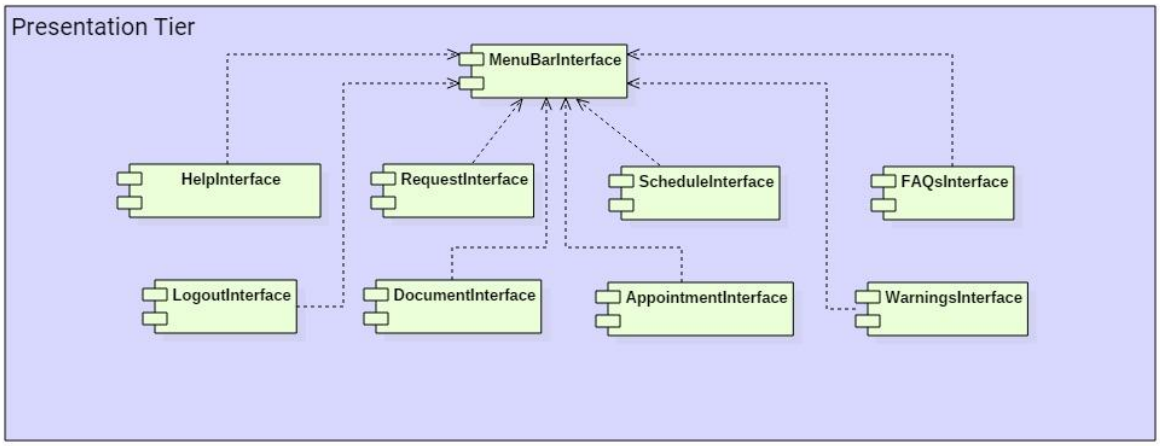
Πριν προχωρήσουμε στην παρουσίαση του διαγράμματος τμημάτων, είναι σημαντικό να αναφερθούν ορισμένες παρατηρήσεις σχετικά με το αυτό. Πιο συγκεκριμένα, όπως προαναφέρθηκε η αρχιτεκτονική που υιοθετήθηκε συνιστά το συνδυασμό της αρχιτεκτονικής Πελάτη - Διακομιστή (client-server) και της αρχιτεκτονικής των τριών επιπέδων (3-tier).

Όσον αφορά την πρώτη, η λογική η οποία ακολουθήθηκε είναι το γεγονός ότι η γραμματεία, η οποία είναι και ο χρήστης αυτού του συστήματος, επιθυμεί να αλληλοεπιδράσει με τις γραφικές διεπαφές επιλέγοντας κάθε φορά τους εκάστοτε συνδέσμους ή πατώντας τα κατάλληλα κουμπιά που υπάρχουν σε καθεμία από αυτές. Πρόκειται, δηλαδή, για μία ενέργεια που αντικατοπτρίζει την επιθυμία για αλλαγή της κατάστασης του συστήματος. Στο σημείο αυτό, το αξιοσημείωτο είναι πως οποιαδήποτε ενέργεια έπεται αποτελεί την απόκριση του server system προς την επιθυμία της γραμματείας. Για παράδειγμα, το σύστημα που σχεδιάστηκε από την ομάδα DIAS δεν εκτελεί αυτόματες διεργασίες, όπως η εμφάνιση ενός μηνύματος στην οθόνη, οι οποίες δεν προέρχονται από κάποια ενέργεια του χρήστη (π.χ. αυτόματες ειδοποιήσεις). Συνεπώς, γίνεται εύκολα κατανοητή η σύνδεση των δύο επιπέδων presentation tier και business-logic tier, κατά την οποία όλα τα υλοποιημένα Interfaces μόνο απαιτούν από τους υπάρχοντες Handlers κάποια πληροφορία. Με την ίδια λογική προχωρούμε και στη σύνδεση των επόμενων επιπέδων, δηλαδή αυτό του business-logic tier με εκείνο του data-access tier. Κατά τη σύνδεση αυτή, οι υπάρχοντες Handlers μόνο απαιτούν την αναγκαία πληροφορία και το Data Handler τους την προσφέρει.

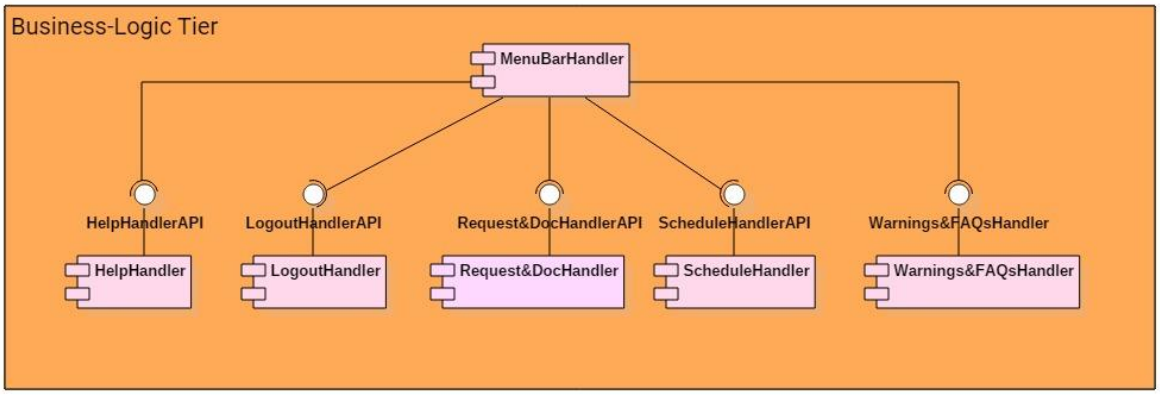
Ακόμη, προκειμένου να γίνει πλήρως κατανοητή η λογική που υιοθετήθηκε, παρουσιάζονται επιπρόσθετα δύο διαγράμματα όπου φαίνονται οι συσχετίσεις των υποσυστημάτων και ανά επίπεδο.



Εικόνα 2.17 Συνολικό διάγραμμα τμημάτων



Εικόνα 2.18 Συσχετίσεις εντός του επιπέδου presentation



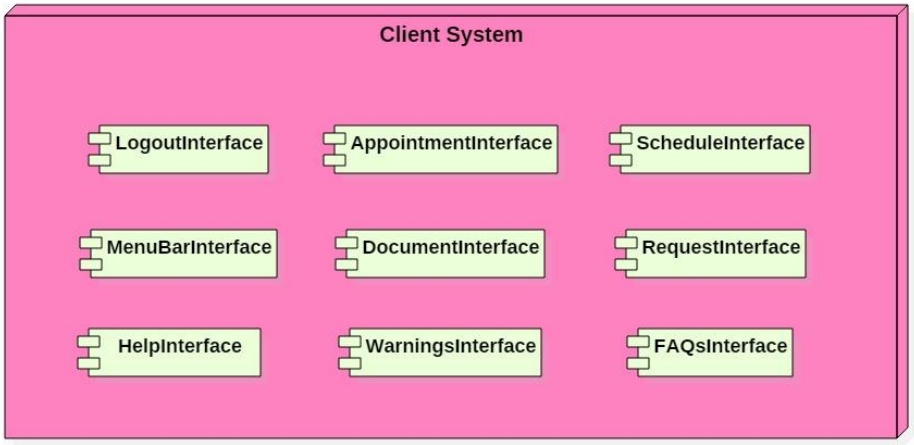
Εικόνα 2.19 Συσχετίσεις εντός του επιπέδου business-logic

2.2 Απεικόνιση Υλικού/Λογισμικού

Παρακάτω παρουσιάζεται με διαγράμματα UML η απεικόνιση υλικού/λογισμικού. Η απεικόνιση αυτή γίνεται με σκοπό οι αναγνώστες να κατανοήσουν τον τρόπο με τον οποίο συνδέονται τα τμήματα του συστήματος, αλλά και τον τρόπο με τον οποίο σχετίζονται αυτά με το υλικό του συστήματος.

2.2.1 Client System

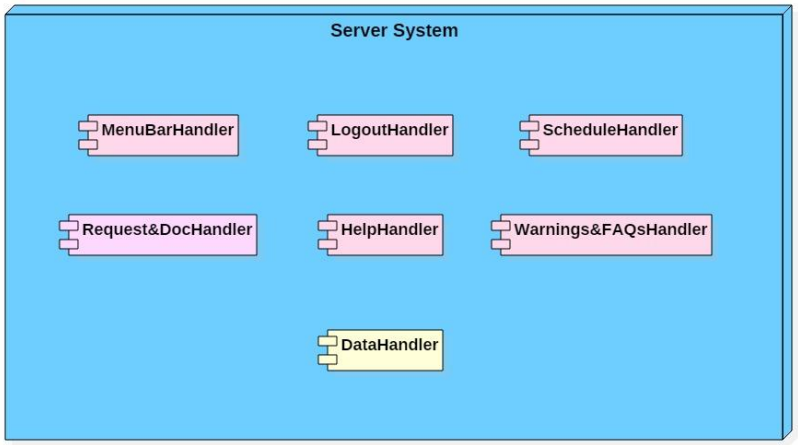
Το Client System είναι ένας κόμβος του συστήματος και αφορά αποκλειστικά τις γραφικές διεπαφές που απαιτούνται προκειμένου η γραμματεία να είναι σε θέση να χρησιμοποιήσει το σύστημα. Το υλικό αυτό θα μπορούσε να είναι οποιαδήποτε συσκευή διαθέτει σύνδεση στο διαδίκτυο, όπως για παράδειγμα ένα smartphone, ένα tablet, ένα laptop κλπ. Δίχως σύνδεση στο διαδίκτυο η επικοινωνία με τον κόμβο του server είναι αδύνατη και συνεπώς, η γραμματεία δε θα είναι σε θέση να αποστείλει αιτήματα και να παραλάβει τα αποτελέσματα αυτών.



Εικόνα 2.20 Κόμβος Client System

2.2.2 Server System

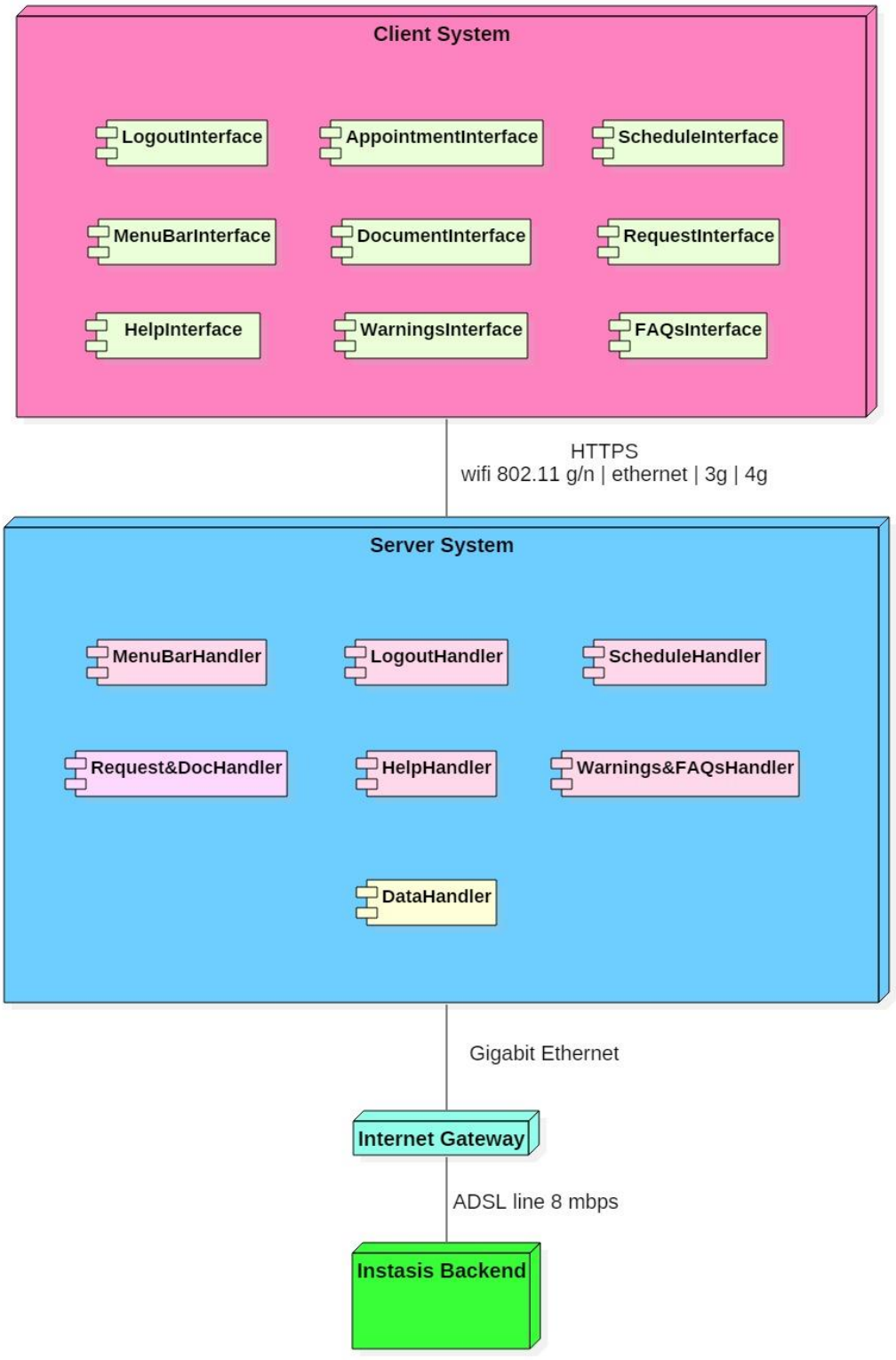
Το Server System είναι ένας κόμβος του συστήματος και μάλιστα ο κυριότερος. Σε αυτό το κόμβο ανήκουν όλα εκείνα τα υποσυστήματα που αναλαμβάνουν να επικοινωνήσουν με το χρήστη και να του προσφέρουν τις υπηρεσίες που επιθυμεί. Αυτό επιτυγχάνεται μέσω της επικοινωνίας του κόμβου αυτού με τη βάση δεδομένων του instasis-backend και έπειτα, με την επεξεργασία των πληροφοριών που λαμβάνει από αυτή. Γενικότερα, ο κόμβος αυτός οργανώνει την ορθή λειτουργία όλου του συστήματος.



Εικόνα 2.21 Κόμβος Server System

2.2.3 Διάγραμμα Ανάπτυξης (Deployment Diagram)

Στην παράγραφο αυτή παρατίθεται το διάγραμμα ανάπτυξης του συστήματος, όπου παρουσιάζονται οι φυσικές συνδέσεις μεταξύ των προαναφερθέντων κόμβων, καθώς και οι τεχνολογίες που επιλέχθηκαν για τη διασύνδεση των καναλιών μεταξύ πελάτη και διακομιστή.



Εικόνα 2.22 Διάγραμμα ανάπτυξης του συστήματος Γραμματεία του Instasis

Παρακάτω ακολουθεί μια πλήρης επεξήγηση και αιτιολόγηση των παραπάνω επιλογών, με βάση τους περιορισμούς που θέτει το σύστημα και αυτούς που έχουμε θέσει στα προηγούμενα παραδοτέα.

Γενικότερα, η επικοινωνία μεταξύ των διαφόρων κόμβων του συστήματος γραμματείας περιορίζεται από την ανάγκη για απόκριση του συστήματός προς τα αιτήματα της γραμματείας εντός χρονικού διαστήματος ενός δευτερολέπτου. Επίσης, επιτακτική είναι και η ανάγκη για ασφάλεια των δεδομένων που μεταφέρονται μεταξύ της γραμματείας και του συστήματος. Ένας ακόμη παράγοντας που έπρεπε να ληφθεί υπόψιν είναι η ανάγκη υποστήριξης διαφορετικών λειτουργικών συστημάτων. Για τους λόγους αυτούς, επιλέχθηκε η σύνδεση μεταξύ client και server να γίνεται με χρήση του πρωτοκόλλου HTTPS και μέσω μιας τυπικής σύνδεσης διαδικτύου (wifi, ethernet, 3G, 4G).

Ειδικότερα, το πρωτόκολλο HTTPS υλοποιεί ένα συνδυασμό του HTTP και του πρωτοκόλλου SSL. Το HTTP αποτελεί το κύριο πρωτόκολλο που χρησιμοποιείται στους φυλλομετρητές (browsers) του Παγκοσμίου Ιστού για να μεταφέρει δεδομένα ανάμεσα σε ένα διακομιστή (server) και έναν πελάτη (client). Το SSL (Secure Sockets Layer) χρησιμοποιεί μεθόδους κρυπτογράφησης των δεδομένων που ανταλλάσσονται μεταξύ δύο συσκευών εγκαθιδρύοντας μία ασφαλή σύνδεση μεταξύ τους μέσω του διαδικτύου. Το πρωτόκολλο αυτό χρησιμοποιεί το TCP/IP για τη μεταφορά των δεδομένων και είναι ανεξάρτητο από την εφαρμογή που χρησιμοποιεί ο τελικός χρήστης. Συνεπώς, με το HTTPS τα δεδομένα που ανταλλάσσονται μεταξύ του client και του server είναι αφενός κρυπτογραφημένα και αφετέρου ανεξάρτητα του λογισμικού που χρησιμοποιούν ο client και ο server. Να σημειωθεί ότι το TCP/IP εξασφαλίζει αξιοπιστία στην μεταφορά πακέτων, με την έννοια ότι τα πακέτα παραδίδονται στον προορισμό τους αναλλοίωτα και με την σειρά που στάλθηκαν.

Επιπροσθέτως, ο server συνδέεται με τον κόμβο Internet Gateway μέσω ενσύρματης σύνδεσης, σύμφωνα με το πρότυπο Gigabit Ethernet, το οποίο πετυχαίνει ταχύτητες της τάξης του 1Gbps. Τα πακέτα που αποστέλλονται από και προς τη βάση δεδομένων του instasis-backend ακολουθούν το πρότυπο που ορίζεται από τη MySQL, η οποία αποτελεί ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Η MySQL αποτελεί την πιο συνήθη επιλογή σε διαδικτυακές εφαρμογές που επικοινωνούν με βάσεις δεδομένων.

Τέλος, η σύνδεση του Internet Gateway με το διαδίκτυο γίνεται με σύνδεση ADSL2+, η οποία εξασφαλίζει μέγιστο ρυθμό λήψης δεδομένων 24 Mbit/s και μέγιστο ρυθμό αποστολής 3.5 Mbit/s.

2.3 Έλεγχος Πρόσβασης και Ασφάλεια

Στο σημείο αυτό αξίζει να υπενθυμιστεί πως το σύστημα γραμματεία της εφαρμογής instasis, το οποίο σχεδιάζουμε, έχει ως μοναδικό χρήστη την ίδια τη γραμματεία. Η διαχείριση του συνόλου των χρηστών ολόκληρης της εφαρμογής instasis και ο ορισμός των δικαιωμάτων που έχει καθένας από αυτούς, είναι αρμοδιότητα του συστήματος διαχειριστή της εφαρμογής instasis. Κατά τη σχεδίασή μας θεωρούμε πως η γραμματεία είναι ήδη συνδεδεμένη στο σύστημα instasis και δεν εστιάζουμε στον τρόπο με τον οποίο έχει συμβεί αυτό. Συνεπώς, οι ενέργειες οι οποίες μπορούν να εκτελεστούν από αυτήν (από πλευράς σχεδίασης του συστήματος γραμματείας) παρουσιάζονται στον ακόλουθο πίνακα πρόσβασης. Η αριστερή στήλη αυτού αναφέρεται στη γραμματεία, ενώ στις επόμενες αναλύονται οι συναρτήσεις καθεμίας γραφικής διεπαφής, οι οποίες μπορούν να καλεστούν από αυτήν.

Actor	**Class**	**Class**	**Class**
Clerk	**ConfirmLogoutGUI**	**ScheduleSelectionGUI**	**ConfirmSelectionGUI**
	okClick()	checkBoxClick()	okClick()
	cancelClick()	dateClick()	cancelClick()
		scrollClick()	
Clerk	**MenuBarGUI**	**DisplaySelectedScheduleGUI**	**ConfirmCancelationGUI**
	menuClick()	appointmentClick()	okClick()
		appointmentCheck()	cancelClick()
		shiftClick()	
		cancelClick()	
		scrollClick()	
Clerk	**ConfirmShiftGUI**	**DisplayClientsInfoGUI**	**DisplayRequestsGUI**
	okClick()	finishedClick()	editClick()
	cancelClick()	didntComeClick()	scrollClick()
		scrollClick()	
		docsClick()	

Actor	Class	Class	Class
Clerk	**EditRequestGUI**	**ConfirmRequestGUI**	**DisplayDocumentsGUI**
	checkBoxClick()	okClick()	editClick()
	scrollClick()	cancelClick()	scrollClick()
	saveClick()		
	cancelClick()		
	modifyTextField()		
Clerk	**EditDocumentTitleGUI**	**EditDocumentContentsGUI**	**ConfirmDocGUI**
	modifyTextField()	chooseFileClick()	okClick()
	nextClick()	saveClick()	cancelClick()
		cancelClick()	
Clerk	**FileDialogBox**	**ManualGUI**	**DisplayWarningsGUI**
	fileButtonClick()	scrollClick()	editClick()
	doneClick()	display()	scrollClick()
	backClick()		
	display()		
Clerk	**EditWarningsGUI**	**ConfirmWarnGUI**	**DisplayFAQsGUI()**
	modifyText()	okClick()	editClick()
	scrollClick()	cancelClick()	scrollClick()
	saveClick()		
	cancelClick()		

Actor	Class	Class	Class
Clerk	**EditFAQsGUI**	**ConfirmFAQsGUI**	
	modifyText()	okClick()	
	scrollClick()	cancelClick()	
	saveClick()		
	cancelClick()		

Όσον αφορά σε θέματα ασφαλείας, σημειώνεται πως το Σύστημα Γραμματεία σχεδιάζεται με σκοπό να παρέχει ασφάλεια και προστασία στους χρήστες του, δηλαδή στις γραμματείες. Το γεγονός αυτό εντοπίζεται κυρίως στα εξής τρία σημεία:

- ***Εχεμύθεια (confidentiality)***: Το Σύστημα Γραμματεία διαχειρίζεται ευαίσθητες πληροφορίες χρηστών καθώς και προσωπικά δεδομένα. Για το λόγο αυτό, οι πληροφορίες αυτές βρίσκονται κρυπογραφημένες μέσω AES (Advanced Encryption Standard) και μέσω χρήσης 256bit κλειδιών στη βάση δεδομένων.

- ***Αξιοπιστία (integrity):*** Εξασφαλίζεται από το σύστημα ότι οι αλλαγές που γίνονται στην εφαρμογή δεν παραβιάζουν την ιδιωτικότητα, καθώς και την ασφάλεια της γραμματείας και των λοιπών χρηστών του ευρύτερου συστήματος instasis.

- ***Διαθεσιμότητα (availability):*** Εξασφαλίζεται το γεγονός ότι η γραμματεία θα έχει πρόσβαση στο σύστημα και στα δεδομένα του, όποτε το επιθυμεί.

3. Προδιαγραφές Τμηματικού Σχεδιασμού (Component Design Specifications)

Το παρόν κεφάλαιο περιγράφει κάποιες συγκεκριμένες προδιαγραφές, τις οποίες θα πρέπει να πληρούν τα τμήματα (components) που περιγράφηκαν παραπάνω και οι κλάσεις που εμπεριέχονται σε αυτά. Οι προδιαγραφές αυτές καθορίζονται με βάση τις ΜΛΑ που έχουν διατυπωθεί σε προηγούμενα παραδοτέα, αλλά και με βάση τη γενικότερη απαίτηση ενός έργου λογισμικού, αυτή της τροποποιήσιμης και επεκτάσιμης σχεδίασης. Για το σκοπό αυτό, εφαρμόζονται τα απαραίτητα πρότυπα σχεδίασης, προσφέροντας τη δυνατότητα προσθήκης νέων λειτουργικοτήτων μελλοντικά δίχως να επηρεαστεί η υπάρχουσα σχεδίαση.

3.1 Πρότυπα Σχεδιασμού που υιοθετήθηκαν

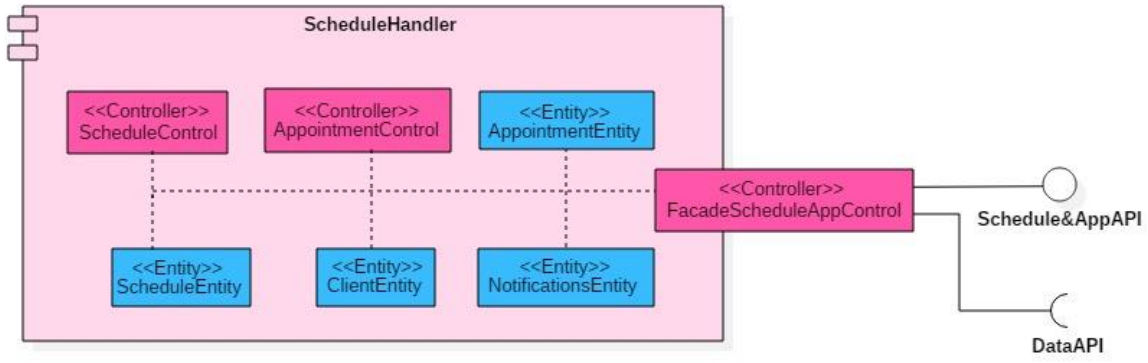
3.1.1 Δομικά Πρότυπα

Τα δομικά πρότυπα ασχολούνται με τον τρόπο με τον οποίο μπορούν να δημιουργηθούν νέες κλάσεις μελλοντικά, ώστε να σχηματίσουν μεγαλύτερες δομές και απλοποιούν τις σύνθετες αυτές δομές, προσδιορίζοντας επακριβώς τις σχέσεις μεταξύ των κλάσεων. Αυτό επιτυγχάνεται με την προσθήκη μιας abstract κλάσης.

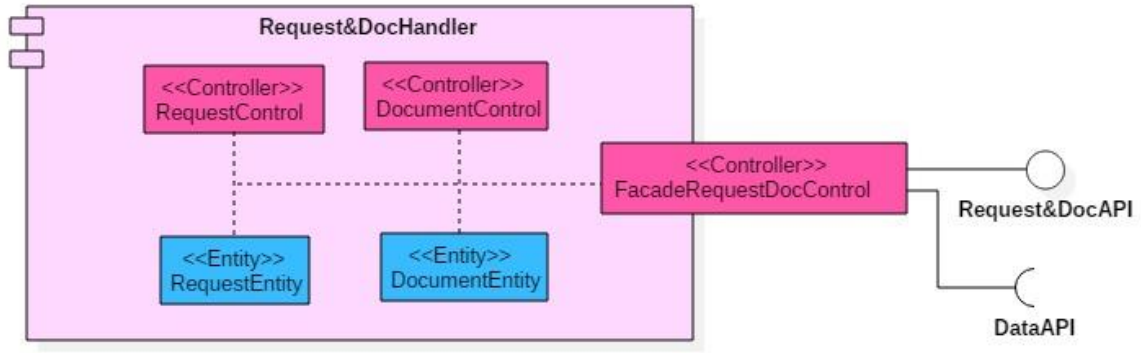
Facade

Καθένα από τα τμήματα Request&DocHandler, ScheduleHandler και Warnings&FAQsHandler ενσωματώνουν περισσότερες από μια λειτουργικότητες και συνεπώς, διαθέτουν περισσότερες από μια κλάσεις ελεγκτή. Επιπλέον, το τμήμα DataHandler εμπεριέχει όλες τις οριακές κλάσεις που απαιτούνται για την επικοινωνία με τη βάση δεδομένων του instasis-backend. Η ομαδοποίηση των διαφόρων κλάσεων στα τμήματα αυτά έγινε με απώτερο σκοπό την επίτευξη χαμηλής σύζευξης, εφόσον οι κλάσεις αυτές εμπεριείχαν συσχετίσεις. Ωστόσο, η ομαδοποίηση αυτή απαιτεί την παρουσία μιας κλάσης, η οποία αναλαμβάνει εσωτερικά τον επιμερισμό των αρμοδιοτήτων μεταξύ των κλάσεων και μεταφέρει την κατάλληλη πληροφορία από το τμήμα (component) αυτό σε οποιοδήποτε άλλο τμήμα. Με τον τρόπο αυτό, επιτυγχάνεται απλούστερη και ευκρινέστερη σχεδίαση, καθώς και η προσαρμοστικότητά του σε μελλοντικές αλλαγές και αναβαθμίσεις.

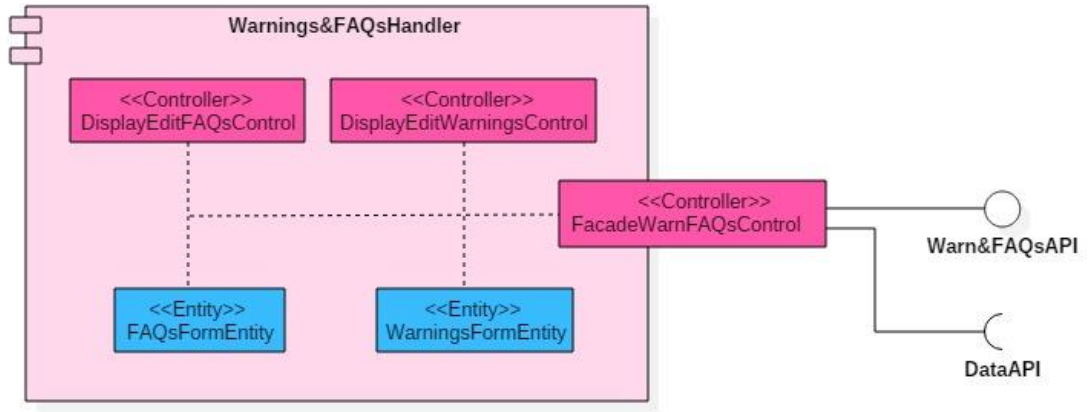
Τα υποσυστήματα που αναδιοργανώθηκαν είναι τα εξής: **ScheduleHandler, Request&DocHandler**, **Warnings&FAQsHandler** και **DataHandler**.



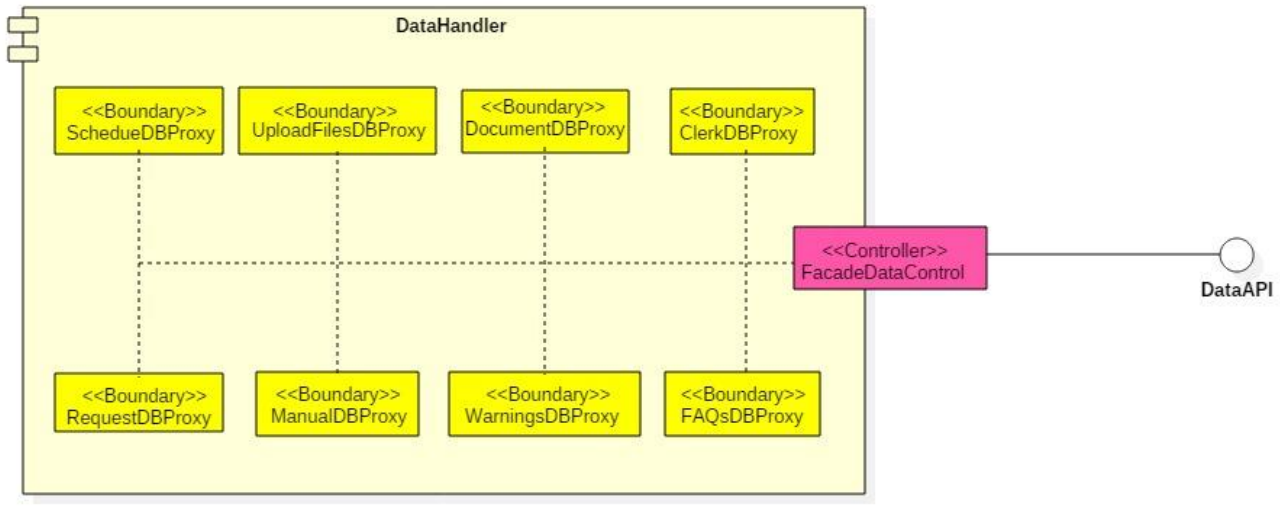
Εικόνα 3.1 Πρότυπο Façade στο υποσύστημα ScheduleHandler



Εικόνα 3.2 Πρότυπο Façade στο υποσύστημα Request&DocHandler



Εικόνα 3.3 Πρότυπο Façade στο υποσύστημα Warnings&FAQsHandler

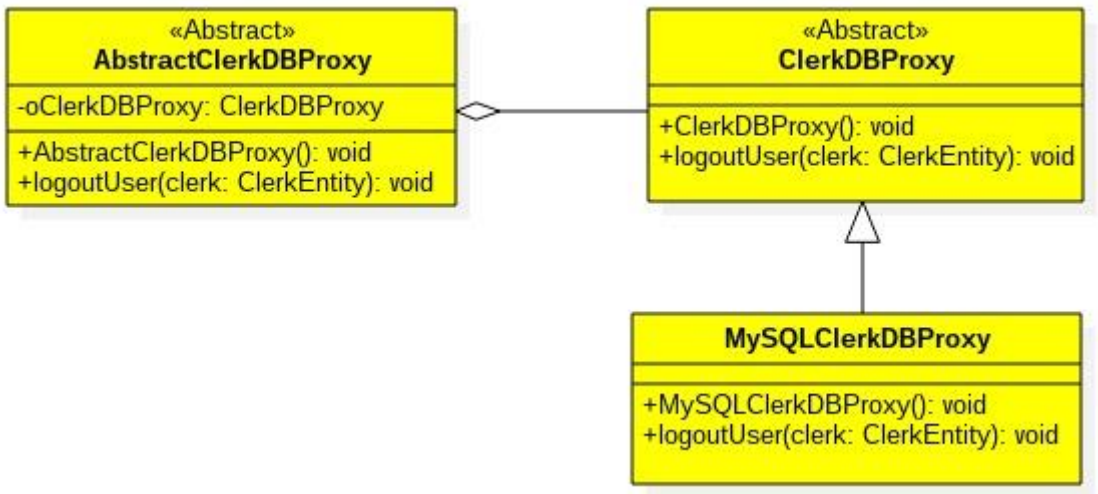


Εικόνα 3.4 Πρότυπο Façade στο υποσύστημα DataHandler

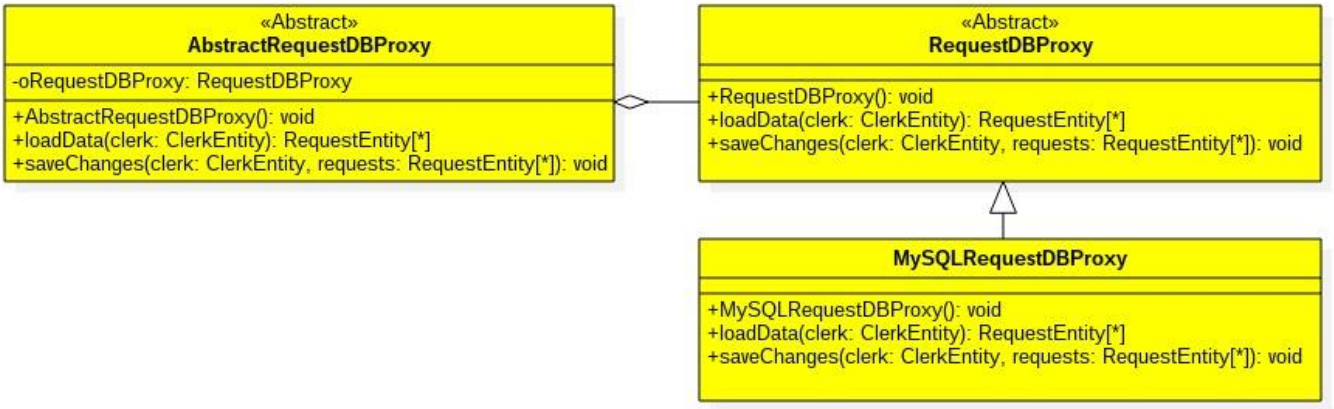
Bridge

Τα οριακά αντικείμενα (ScheduleDBProxy, UploadFilesDBProxy, DocumentDBProxy, ClerkDBProxy, RequestDBProxy, ManualDBProxy, WarningsDBProxy, FAQsDBProxy) που χρησιμοποιούνται για την επικοινωνία με τη βάση δεδομένων του instasis-backend έχουν αναπτυχθεί στη γλώσσα MySQL, όπως προαναφέρθηκε. Παρόλα αυτά η ύπαρξη οριακών αντικειμένων που υποστηρίζει πρόσβαση μόνο σε βάσεις δεδομένων γραμμένες σε γλώσσα MySQL αποτελεί τροχοπέδη στην ανάπτυξη του συστήματος, ώστε να υποστηρίζονται μελλοντικά και νέοι τύποι βάσεων δεδομένων. Έτσι, με στόχο την εξασφάλιση ευελιξίας στην ανάπτυξη του λογισμικού και την εξελιξιμότητα του συστήματος, υιοθετήθηκε το πρότυπο αυτό που παρέχει υποστήριξη για τη λειτουργία των ήδη εγκατεστημένων βάσεων δεδομένων, αλλά και τυχόν βάσεων δεδομένων που θα εγκατασταθούν μελλοντικά, επιφέροντας τις λιγότερες δυνατόν αλλαγές στο σύστημα.

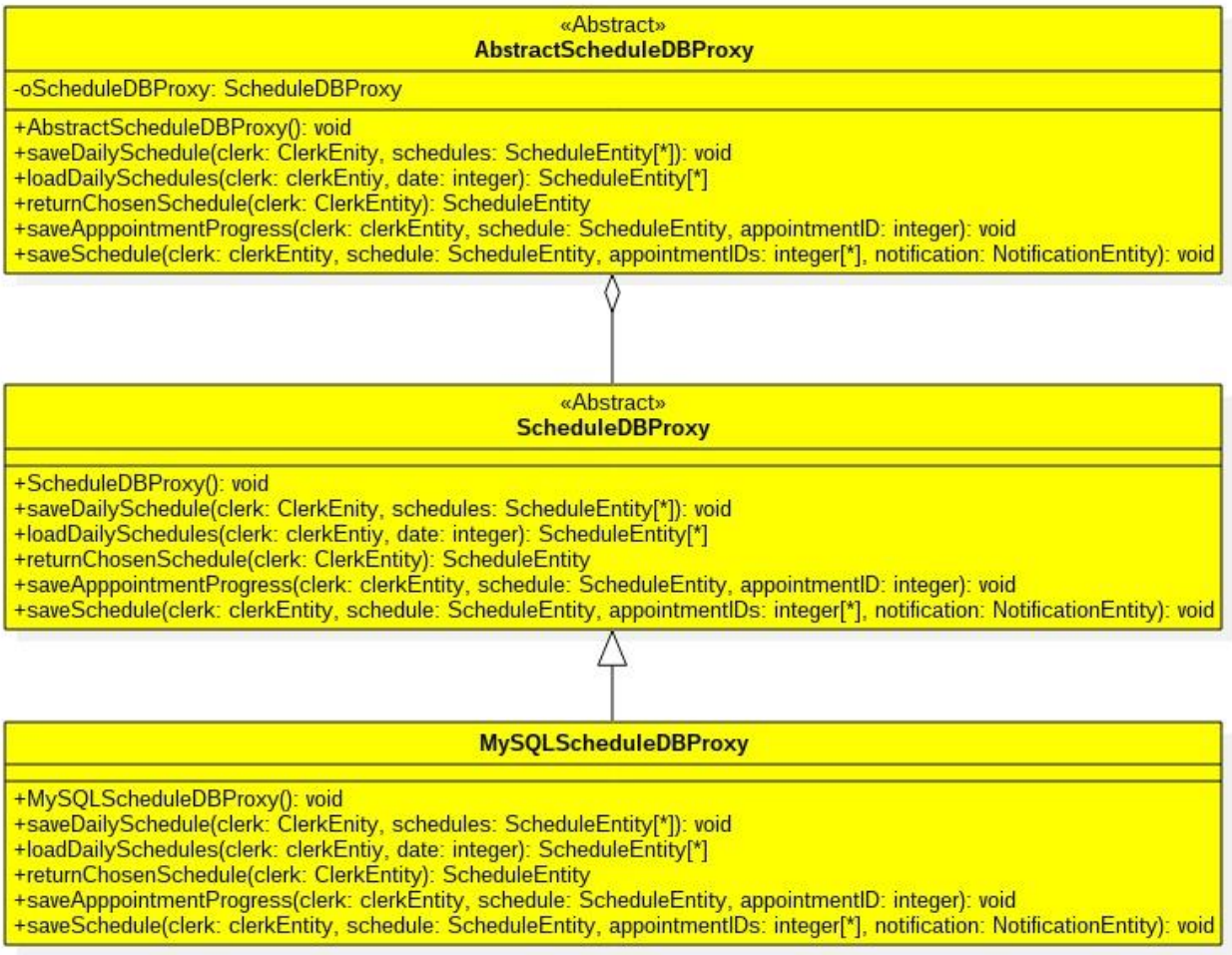
Το υποσύστημα που αναδιοργανώθηκε είναι το εξής: **DataHandler**.



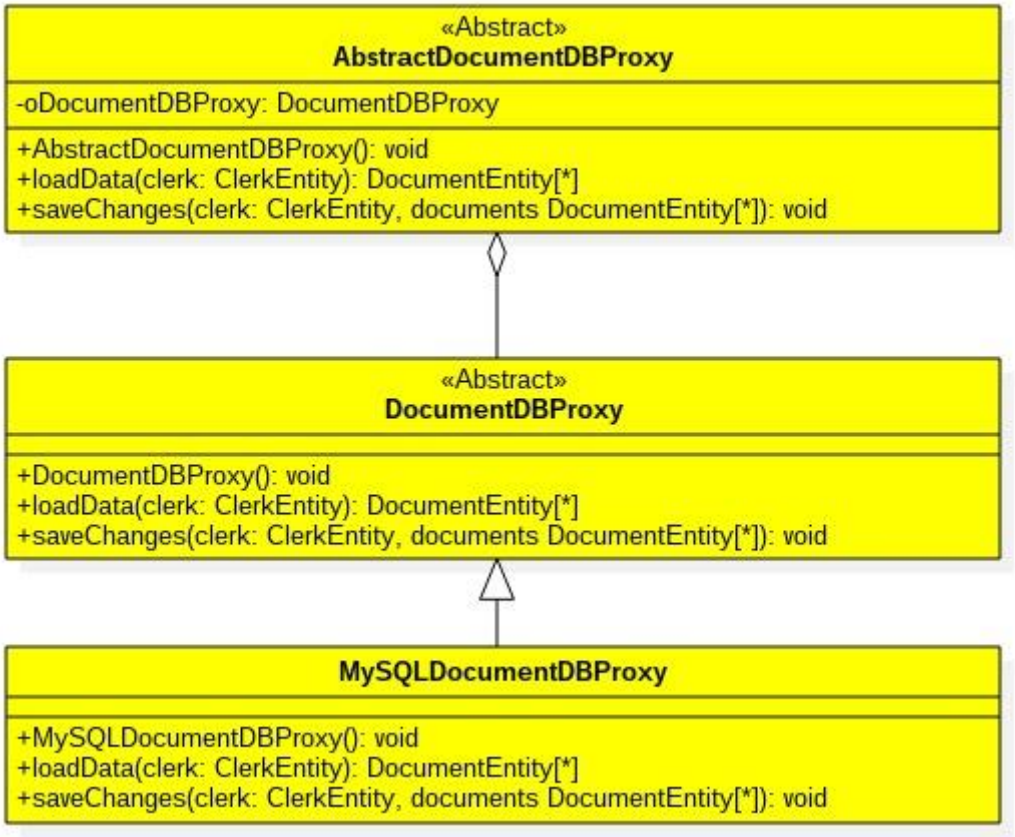
Εικόνα 3.5 Πρότυπο Bridge στην οριακή κλάση ClerkDBProxy



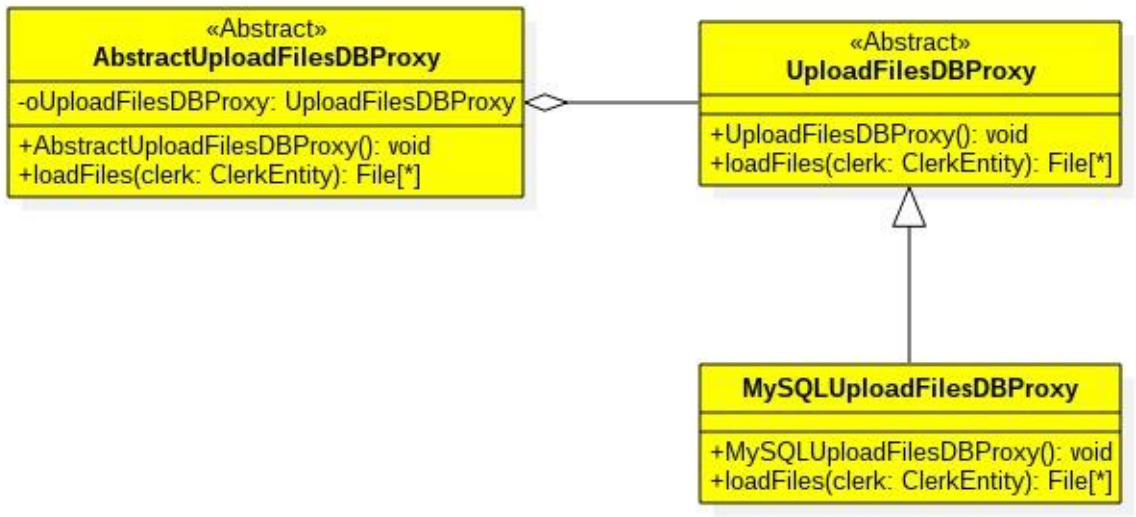
Εικόνα 3.6 Πρότυπο Bridge στην οριακή κλάση RequestDBProxy



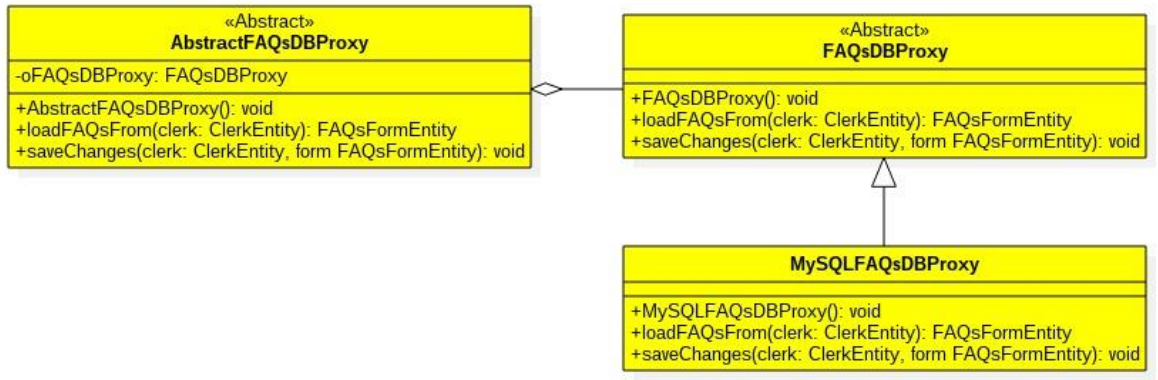
Εικόνα 3.7 Πρότυπο Bridge στην οριακή κλάση ScheduleDBProxy



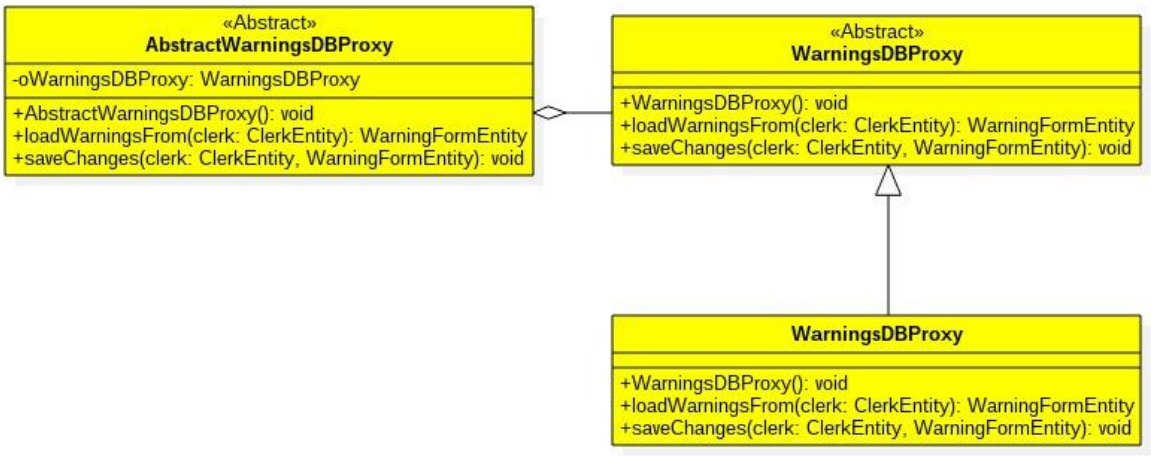
Εικόνα 3.8 Πρότυπο Bridge στην οριακή κλάση DocumentDBProxy



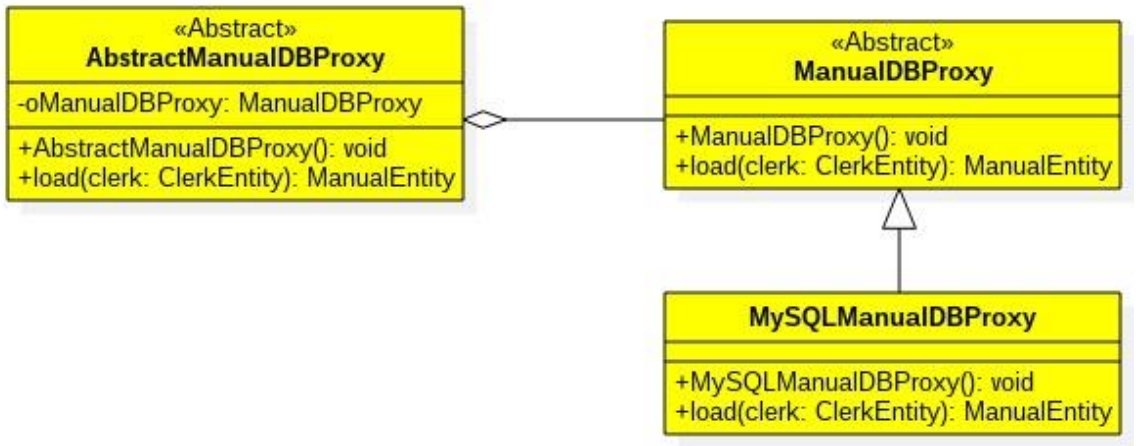
Εικόνα 3.9 Πρότυπο Bridge στην οριακή κλάση UploadFilesDBProxy



Εικόνα 3.10 Πρότυπο Bridge στην οριακή κλάση FAQsDBProxy



Εικόνα 3.11 Πρότυπο Bridge στην οριακή κλάση WarningsDBProxy



Εικόνα 3.12 Πρότυπο Bridge στην οριακή κλάση ManualDBProxy

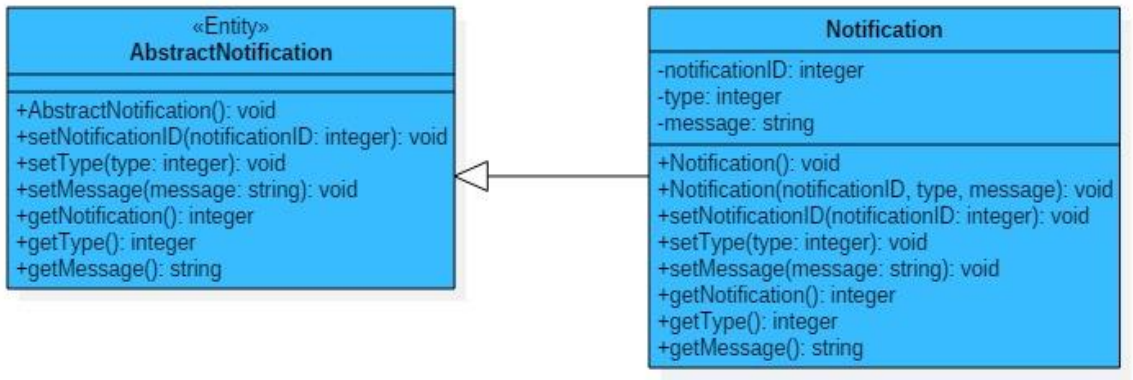
Composite

Η οντότητα NotificationEntity είναι αυτή μέσω της οποίας δημιουργούνται τα κατάλληλα αντικείμενα ειδοποιήσεων, τα οποία αποστέλλονται στη βάση δεδομένων του instasis-backend, με απώτερους παραλήπτες τους αιτούμενους των ραντεβού, προκειμένου να τους ενημερώσουν για τυχόν ακύρωση ή μετάθεση του ραντεβού τους. Η ειδοποίηση αυτή έχει τη μορφή email ή sms. Ωστόσο, το σύστημα θα πρέπει να είναι σε θέση να υποστηρίξει και άλλες μορφές ειδοποιήσεων, ανάλογα με τα τεχνολογικά δεδομένα και φυσικά τις επιθυμίες των αιτούμενων, προκειμένου να είναι εύχρηστο. Συνεπώς, επιτακτική είναι η ανάγκη εφαρμογής του συγκεκριμένου προτύπου, το οποίο παρέχει τη δυνατότητα προσθήκης μελλοντικά κλάσεων που θα αντιπροσωπεύουν κάποιο διαφορετικό είδος ειδοποίησης. Για το λόγο αυτό, προστέθηκε μια abstract κλάση την οποία και θα κληρονομούν οι τυχούσες νέες κλάσεις ειδοποιήσεων στο μέλλον.

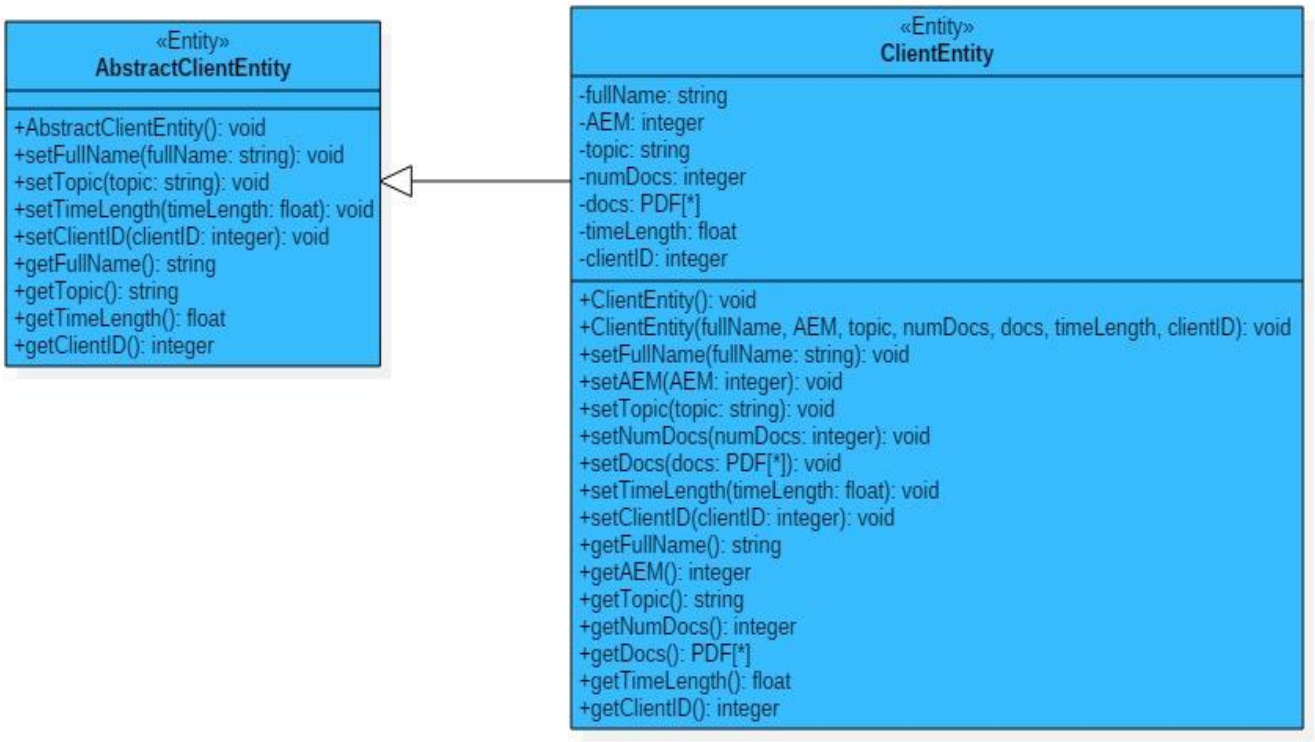
Επιπρόσθετα, η οντότητα Client Entity είναι αυτή μέσω της οποία δηλώνονται οι προς εξυπηρέτηση πελάτες. Σε πρώτο στάδιο υλοποίησης του συστήματος instasis παρουσιάζεται η δυνατότητα να κλείσει ραντεβού κάποιος με τα χαρακτηριστικά που φαίνονται στην κλάση ClientEntity. Ωστόσο, επειδή η γραμματεία των σχολών προσεγγίζεται και από άτομα τα οποία μπορεί να έχουν διαφορετικές ιδιότητες, το σύστημα πρέπει να μπορεί να εξυπηρετεί όλους τους πελάτες. Εφαρμόζοντας, λοιπόν, το πρότυπο composite και στην οντότητα Client Entity, επιλύεται το πρόβλημα αυτό. Η εφαρμογή του προτύπου στην οντότητα αυτή έγινε με παρόμοιο τρόπο όπως αναφέρθηκε, προσθέτοντας μία abstract κλάση την οποία και θα κληρονομούν οι τυχούσες νέες κλάσεις πελατών στο μέλλον.

Το υποσύστημα που αναδιοργανώθηκε είναι το εξής: **ScheduleHandler**.

Στο συγκεκριμένο σημείο πρέπει να επισημάνουμε το γεγονός ότι το πρότυπο Composite έχει κύρια εφαρμογή σε περιπτώσεις όπου υπάρχει η επιθυμία διαχείρισης ενός συνόλου διαφορετικών αντικειμένων με όμοιο τρόπο. Τα αντικείμενα αυτά μπορεί να υπάρχουν στο σύστημα είτε να γίνουν περαιτέρω προσθήκες αντικείμενων στο μέλλον. Η επεκτασιμότητα, επομένως, είναι μια παρενέργεια αυτού και όχι ο σκοπός εφαρμογής του. Αυτό που μας ώθησε στην εφαρμογή του για λόγους επεκτασιμότητας είναι το ότι επιτρέπει την ύπαρξη όλων αυτών των διαφορετικών αντικειμένων την ίδια χρονική στιγμή στο σύστημα και δεν επιλέγει ένα εξ' αυτών αποκλειστικά, σε αντίθεση με άλλα πρότυπα όπως το Template ή το Strategy. Παραδείγματος χάριν, εάν προστεθεί ένα νέο είδος ειδοποίησης στο μέλλον, τότε μπορεί να υπάρχει η επιθυμία να αποστέλλονται και το παλιό είδος ειδοποίησης, αλλά και το νέο είδος ειδοποίησης στον αιτούμενο. Εάν εφαρμόζαμε κάποιο άλλο πρότυπο, όπως το Template, δε θα παρεχόταν η δυνατότητα αυτή, αλλά το σύστημα θα ήταν αναγκασμένο να αποστείλει ένα μονάχα είδος ειδοποίησης αναγκαστικά.



Εικόνα 3.13 Πρότυπο Composite στην οντότητα ClientEntity



Εικόνα 3.14 Πρότυπο Composite στην οντότητα NotificationEntity

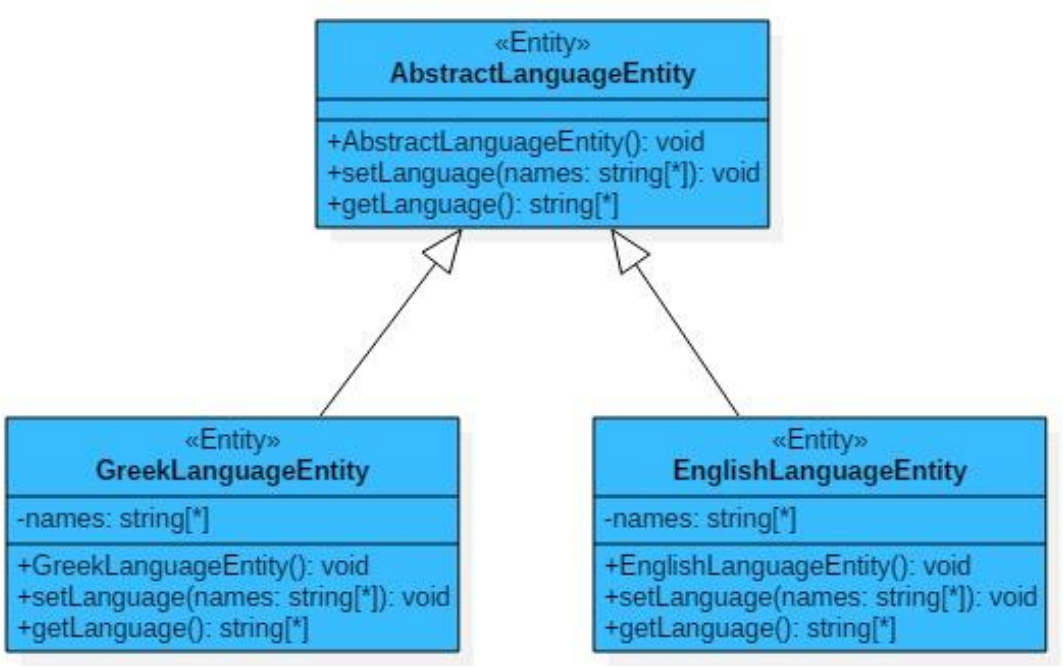
3.1.2 Πρότυπα Συμπεριφοράς

Τα πρότυπα συμπεριφοράς αφορούν την αλληλεπίδραση και τις ευθύνες των αντικειμένων και επιτρέπουν την επιλογή του κατάλληλου αλγορίθμου κάθε φορά. Επιπροσθέτως, αποφεύγουν τη στενή σύζευξη σε συγκεκριμένες λύσεις και προσφέρουν επεκτασιμότητα.

Template

Οι κλάσεις οντοτήτων GreekLanguageEntity και EnglishLanguageEntity που αφορούν τη γλώσσα του συστήματος, μοιράζονται τον ίδιο αλγόριθμο και διαφέρουν μόνο στο περιεχόμενο των attributes. Κατ' αυτόν τον τρόπο, ομαδοποιήθηκαν εξ' αρχής μέσω του προτύπου αυτού και προστέθηκε για το σκοπό αυτό η abstract κλάση AbstractLanguageEntity. Έτσι, επιτυγχάνεται η δυνατότητα ευκολότερης επέκτασης του συστήματος με την προσθήκη νέων κλάσεων γλωσσών μελλοντικά που κληρονομούν την κλάση AbstractLanguageEntity.

Το υποσύστημα που αναδιοργανώθηκε είναι το εξής: **LogoutHandler**



Εικόνα 3.15 Πρότυπο Template στην οντότητα LanguageEntity

4. Πίνακας ιχνηλασιμότητας εγγράφων Σχεδίασης και Απαιτήσεων Λογισμικού

Τροποποίηση στο δεύτερο παραδοτέο για τη μετάβαση στο τρίτο:

Κατά τη συγγραφή του παρόντος εγγράφου εντοπίστηκαν κάποια σημεία του εγγράφου SRD τα οποία θα μπορούσαν να τροποποιηθούν στο παρόν έγγραφο SDD. Η διαφοροποίηση έγκειται κατά την υλοποίηση της ΜΛΑ–2 «Το σύστημα πρέπει να υποστηρίζει την ελληνική και την αγγλική γλώσσα». Πιο αναλυτικά, προστέθηκε μία νέα κλάση η οποία θα ονομάζεται AbstractLanguageEntity και την οποία θα κληρονομούν οι επίσης νέες κλάσεις GreekLanguageEntity και EnglishLanguageEntity. Προκειμένου να αποφύγουμε οποιαδήποτε σύγχυση στην ήδη υπάρχουσα σχεδίαση, επιλέχθηκε από τη σχεδιαστική ομάδα, οι προσθήκες αυτές να πραγματοποιηθούν στο πακέτο LogoutHandle (1.1.1 όπως αναφέρεται στο έγγραφο SRD). Με αυτόν τον τρόπο, η σύνδεση των πακέτων μεταξύ τους παραμένει η ίδια όπως εμφανίζεται στα διαγράμματα κλάσεων με τη διαφορά ότι όλες οι κλάσεις ελεγκτών θα συνδέονται με το εκτός από το ClerkEntity και με αυτό της AbstractLanguageEntity. Για να υπάρχει αυτή η σύνδεση που μόλις αναφέρθηκε, είναι απαραίτητο να προστεθεί σε όλες τις κλάσεις ελεγκτών πλην του MenuBarControl και σε όλες τις οριακές κλάσεις Proxy μία επιπλέον μέθοδος. Σε κάθε περίπτωση, η συνάρτηση αυτή θα καλείται έχοντας ως όρισμα ένα entity τύπου ClerkEntity. Όσον αφορά τους ελεγκτές, σε αυτούς, η μέθοδος θα είναι τύπου void, ενώ στα proxy θα είναι τύπου AbstractLanguageEntity. Η αλλαγή αυτή πληροί πλέον τις ιδιότητες του προτύπου Template, όπως αυτό παρουσιάζεται στην αντίστοιχη παράγραφο.

Παρακάτω ακολουθεί επακριβής αναφορά των σημείων του δεύτερου παραδοτέου, στα οποία καλούνται να ανατρέξουν οι αναγνώστες, έτσι ώστε να γίνει κατανοητός ο τρόπος εφαρμογής των διαφόρων προτύπων του παρόντος παραδοτέου:

Αξιοσημείωτη είναι, λοιπόν, η τροποποίηση όλων των οριακών κλάσεων Proxy, στις οποίες εφαρμόστηκε το πρότυπο Bridge, όπως αναφέρεται αναλυτικότερα ήδη σε προηγούμενη παράγραφο.

Μία επιπλέον τροποποίηση, αφορά την εφαρμογή του προτύπου Façade στα υποσυστήματα του παρόντος εγγράφου, όπως αυτό αναλύεται προηγουμένως. Η διαφοροποίηση από το έγγραφο SRD είναι η προσθήκη τριών επιπλέον κλάσεων ελεγκτών, στα ακόλουθα πακέτα:

- Πακέτα: ScheduleHandle (1.1.2 όπως αναφέρεται στο έγγραφο SRD) και AppointmentHandle (1.1.3 όπως αναφέρεται στο έγγραφο SRD), προσθήκη ενός ελεγκτή FacadeScheduleAppControl που συσχετίζεται με τις κλάσεις των πακέτων, όπως εμφανίζεται προηγουμένως.
- Πακέτα: RequestsHandle (1.2.1 όπως αναφέρεται στο έγγραφο SRD) και DocumentsHandle (1.2.2 όπως αναφέρεται στο έγγραφο SRD), προσθήκη ενός ελεγκτή FacadeRequestDocControl που συσχετίζεται με τις κλάσεις των πακέτων, όπως εμφανίζεται προηγουμένως.
- Πακέτα: WarningsHandle (1.3.2 όπως αναφέρεται στο έγγραφο SRD) και FAQsHandle (1.3.3 όπως αναφέρεται στο έγγραφο SRD), προσθήκη ενός ελεγκτή WarnFAQsControl που συσχετίζεται με τις κλάσεις των πακέτων, όπως εμφανίζεται προηγουμένως.

- Όλα τα πακέτα: έχει προστεθεί ένας επιπλέον ελεγκτής ο FacadeDataControl, ο οποίος επικοινωνεί με όλες τις οριακές κλάσεις Proxy.

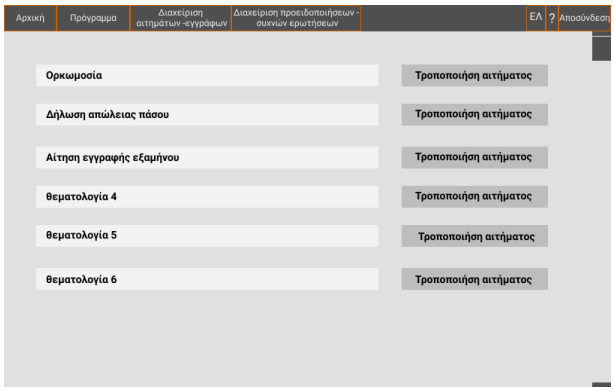
Τέλος, προκειμένου να είναι εφικτή η επεκτασιμότητα του συστήματος σε μελλοντική χρήση, εφαρμόστηκε το πρότυπο Composite στις κλάσεις NotificationEntity και ClientEntity, όπως αναφέρεται αναλυτικότερα προηγούμενη παράγραφο. Η εφαρμογή αυτή επιφέρει όπως επεξηγείται αλλαγές, καθώς προστίθενται δύο επιπλέον abstract κλάσεις. Συνεπώς, τροποποιείται το πακέτο AppointmentHandle (1.1.3 όπως αναφέρεται στο έγγραφο SRD).

5. Παράρτημα Ι – Ανοιχτά Θέματα

Ένα θέμα το οποίο συζητήθηκε εκτενέστερα με τα μέλη της ομάδας, αφορούσε την προσθήκη του προτύπου Proxy, έτσι ώστε να αποφεύγεται η φόρτωση μεγάλου όγκου δεδομένων χωρίς να είναι απαραίτητο. Το πρότυπο αυτό θα μπορούσε να βρει εφαρμογή στις γραφικές διεπαφές στις οποίες εμφανίζονται:

- Το επιλεγμένο πρόγραμμα της γραμματείας, όταν δε χρειάζεται να φαίνεται περισσότερη πληροφορία, όπως αυτά παρουσιάζονται στην οθόνη «Προβολή Επιλεγμένου Προγράμματος»

- στην προβολή όλων των αιτημάτων, όπως αυτά παρουσιάζονται στην οθόνη «Αιτήματα»



- στην προβολή των εγγράφων, όπως αυτά φαίνονται στην οθόνη «Έγγραφα».