ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ηλεκτρονική Διακυβέρνηση: Ορισμός, Ανάλυση & Τεχνολογίες Ανάπτυξης Μοντέλου Αιτήσεων-Εγκρίσεων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΡΑΛΑΜΠΟΥ ΠΛΟΥΜΗ

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ηλεκτρονική Διακυβέρνηση:
Ορισμός, Ανάλυση & Τεχνολογίες Ανάπτυξης Μοντέλου Αιτήσεων-Εγκρίσεων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΧΑΡΑΛΑΜΠΟΥ ΠΛΟΥΜΗ

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 23η Ιουλίου 2009.

.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Ελευθέριος Καγιάφας Καθηγητής Ε.Μ.Π.	Βασίλειος Λούμος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

.....

ΧΑΡΑΛΑΜΠΟΣ ΠΛΟΥΜΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ένας πολύ μεγάλος αριθμός κυβερνητικών οργανισμών έχει «αγκαλιάσει» την Ηλεκτρονική Διακυβέρνηση σαν μια καινοτόμο και αναπόφευκτη δομή δημόσιας υπηρεσίας και διοίκησης. Έχουν σαφώς αναγνωριστεί πολλά οφέλη χρήση της, τόσο για τον δημόσιο τομέα όσο και για τον πολίτη. Στοχεύοντας στην αναγνώριση κάποιων μοντέλων που επαναλαμβάνονται σε διαφορετικούς τύπους υπηρεσιών μπορεί να αναπτυχθεί μια διαδικασία περιγραφής και ανάπτυξης συστημάτων που θα δώσει στην Ηλεκτρονική Διακυβέρνηση το προνόμιο να παρέχει υπηρεσίες και πληροφόρηση πολύ πιο αποτελεσματικά και αποδοτικά. Ένα από τα βασικότερα μοντέλα υπηρεσιών που μπορούν να εντοπισθούν είναι το Μοντέλο Κατάθεσης Αιτήσεων - Έγκρισης Αιτήσεων/Εκδοσης Πιστοποιητικών. Με βάση το μοντέλο αυτό έγινε η ανάλυση, σχεδίαση και υλοποίηση ενός πρότυπου ηλεκτρονικού επιχειρησιακού συστήματος που υλοποιεί το σύνολο των λειτουργιών μιας υπηρεσίας έκδοσης πιστοποιητικών – αδειών κατόπιν αιτήματος στον αρμόδιο φορέα.

Η ανάλυση και ανάπτυξη της επιχειρησιακής λογικής του συστήματος έγινε με παραμετρικό τρόπο έτσι ώστε να είναι ανεξάρτητη από την κατηγορία ή την ιδιαιτερότητα των αιτήσεων – εγκρίσεων. Για την δημιουργία μιας εφαρμογής ευέλικτης, αξιόπιστης και προσαρμόσιμης σε οποιαδήποτε ενδεχόμενη αλλαγή σε λογικό ή τεχνικό – τεχνολογικό επίπεδο επιλέχθηκε το αρχιτεκτονικό μόρφημα MVC και υλοποιήθηκε με τη χρήση τεχνολογικών εργαλείων που βασίζονται στη γλώσσα προγραμματισμού JAVA. Συγκεκριμένα για την υλοποίηση των λειτουργικών και μη λειτουργικών απαιτήσεων χρησιμοποιήθηκε το Spring Framework, που αποτελεί ένα σύγχρονο πλαίσιο ανάπτυξης εφαρμογών J2EE, και υποστηρίζει πλήρως το αρχιτεκτονικό μόρφημα που επιλέχθηκε.

Το αποτέλεσμα είναι η υλοποίηση μιας εφαρμογής που πρώτον μπορεί να προσαρμοστεί εύκολα στις ιδιαίτερες λειτουργικές απαιτήσεις οποιουδήποτε διαδικτυακού συστήματος αδειών – εγκρίσεων, δεύτερον μπορεί να αποτελέσει τη βάση για την ανάπτυξη οποιασδήποτε διαδικτυακής υπηρεσίας ενός δημόσιου φορέα και τρίτον μπορεί να συντηρηθεί ή να επεκταθεί χωρίς σημαντικό κόστος.

Λέξεις Κλειδιά: <<Ηλεκτρονική Διακυβέρνηση,JAVA,J2EE,Spring Framework,MVC>>

Abstract

A large number of governmental organizations have embraced e-government as an innovative and inevitable structure of public service and administration. Many advantages for the public sector as much as for the citizen have been acknowledged. Targeting for the identification of certain models that are repeating in different types of services, a process of system description and development can be developed, which will give e-government the privilege of providing services and information much more effectively and efficiently. One of the most basic service models is the Model Application Submission – Application Approval/Certificate Issue. Based on this model, the analysis, designing and implementation of an original electronic business model that implements the total of operations of an issue certificate – permission service to the appropriate body has been done.

The analysis and development of the business logic of the system has been done in a parametric way so that it can be independent of the category or the particularity of applications – approvals. For the creation of an application flexible, reliable and adaptive to any potential change at logical or technical – technological level, the architectural pattern MVC was selected and implemented with the use of technological tools that are based to the program language JAVA. Specifically, for the implementation of functional and non functional requirements the Spring Framework was used, that constitutes a modern framework of development of J2EE applications, and totally supports the architectural module chosen.

As a result, the application implemented, firstly can be adjusted easily to the particular functional requirements of any web application permission – approval system, secondly can form the base for the development of any web service of a public body and thirdly can be supported or extended without significant cost.

Keywords:<<e-Government,JAVA,J2EE,Spring Framework,MVC>>

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την κα Βαρβαρίγου Θεοδώρα, Καθηγήτρια της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών του ΕΜΠ για την ευκαιρία που μου έδωσε να συνεργαστώ μαζί της, την προθυμία της να παρέχει λύσεις σε ζητήματα που παρουσιάστηκαν και την εμπιστοσύνη που μου έδειξε για τη λήψη αποφάσεων και πρωτοβουλιών.
Επίσης, ευχαριστώ τον διδάκτορα κ. Χάλκο Δημήτριο και τον επιστημονικό συνεργάτη κ. Πολυχνιάτη Θεόδωρο για την άψογη συνεργασία μαζί τους. Η πολύτιμη καθοδήγηση και συμβολή τους, υπήρξαν καθοριστικής σημασίας.
Οφείλω να εκφράσω ιδιαιτέρως τις ευχαριστίες μου καθώς και την ευγνωμοσύνη μου προς την οικογένεια μου, για την αμέριστη συμπαράσταση και υποστήριξη που μου παρείχε κατά τη διάρκεια των σπουδών μου.
Ευχαριστώ πολύ τη Κατερίνα Κουτσοβούλου για την πολύτιμη βοήθεια της στη συγγραφή αυτής της διπλωματικής και για την υποστήριξή της όλα αυτά τα χρόνια.
Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου, για ό λα αυτά που μοιραστήκαμε, όλα αυτά τα χρόνια στην κοινή μας διαδρομή.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 ΕΙΣΑΓΩΓΗ 1
1.1 Ηλεκτρονικές Υπηρεσίες 1
1.2 Αντικείμενο διπλωματικής 2
1.3 Οργάνωση κειμένου 2
2 ΗΛΕΚΤΡΟΝΙΚΗ ΔΙΑΚΥΒΕΡΝΗΣΗ 5
2.1 Ορισμός 5
2.2 Πλεονεκτήματα χρήσης 6
2.3 Προϋποθέσεις χρήσης - σωστού σχεδιασμού 7
2.4 Επίπεδα Ηλεκτρονικής Διακυβέρνησης 8
2.5 Τεχνολογικά ζητήματα 10
2.5.1 Διαλειτουργικότητα 10
2.5.2 Προστασία προσωπικών δεδομένων – Ασφάλεια συστημάτων 12
2.5.3 Πρόσβαση-Αυθεντικοποίηση 13
2.5.4 Διαθεσιμότητα και απόδοση εξυπηρετητών 14
2.5.5 Προσβασιμότητα 14
2.6 Πρωτοβουλία eEurope-i2010 15
2.7 Ελληνική Ψηφιακή Στρατηγική και Ηλεκτρονική Διακυβέρνηση 16
2.8 Νομικό πλαίσιο 18
2.9 Διαδικτυακές Πύλες – Διαδικτυακοί Τόποι 18
2.10 Στατιστικά στοιχεία για την χρήση του Internet στην Ελλάδα 20
2.10.1 Στατιστικά Χρήσης Ίντερνετ για το 2007 20
2.10.2 Στατιστικά Χρήσης Ίντερνετ για το 2008 21
3 ΤΥΠΟΙ & ΜΟΝΤΕΛΑ ΥΠΗΡΕΣΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΚΥΒΕΡΝΗΣΗΣ 23
3.1 Τύποι Υπηρεσιών Ηλεκτρονικής Διακυβέρνησης 23
3.2 Πλαίσιο & Μοντέλα Υπηρεσιών Ηλεκτρονικής Διακυβέρνησης 25
3.3 Μοντέλο Κατάθεσης Αιτήσεων - Έγκρισης/Έκδοσης Πιστοποιητικών 27
3.3.1 Υπηρεσίες Ηλεκτρονικής Διακυβέρνησης που υιοθετούν το μοντέλο 27
3.3.2 Γενική Ροή Διαδικασιών Έκδοσης Αδειών 28
4 ΑΝΑΛΥΣΗ ΜΟΝΤΕΛΟΥ ΑΙΤΗΣΕΩΝ-ΕΓΚΡΙΣΕΩΝ 31
4.1 Κατηγορίες Χρηστών 31
4.2 Λειτουργικές Απαιτήσεις 32
4.2.1 Λειτουργίες εκτελούμενες από εγκεκριμένο χρήστη 32
4.2.2 Λειτουργίες εκτελούμενες από την Αρμόδια Αρχή 33
4.2.3 Κατάσταση Αίτησης, Άδειας - Έγκρισης 35
4.3 Ανάλυση των λειτουργιών του συστήματος ανά περίπτωση χρήσης 37
4.3.1 Αίτηση 38
4.3.2 Άδεια-Έγκριση 40
4.3.3 Έλεγχοι του συστήματος κατά την υποβολή – αποδοχή μιας αίτησης 45
4.4 Κύκλος Ζωής Της Αίτησης – Άδειας 46
5 ΤΕΧΝΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ JAVA 2 ENTERPRISE EDITION (J2EE) 49
5.1 Η ανάγκη για τεχνολογίες ανάπτυξης επιχειρησιακών εφαρμογών ανεξάρτητες από δεσμεύσεις 49
5.2 Γιατί οι εφαρμογές J2EE είναι επιθυμητές; 50
5.2.1 Η αρχιτεκτονική πελάτη - εξυπηρετητή (client - server) 50
5.2.2 Το πολυστρωματικό πρότυπο αρχιτεκτονικής των εφαρμογών 51
5.3 Συστατικά λογισμικού (components) 52
5.4 Πλεονεκτήματα του μοντέλου πολλαπλής στρωμάτωσης 54
5.5 Αρχιτεκτονική της πλατφόρμας J2EE 56
5.5.1 Κατανεμημένες πολυστρωματικές εφαρμογές 57
5.6 Τεχνολογικά Χαρακτηριστικά 58
5.7 Εργαλεία J2EE 59
5.7.1 Servlets (Μικροϋπηρεσίες) 60
5.7.2 Java Server Pages (JSPs) 61
5.7.3 Enterprise Java Bean - EJB (Επιχειρησιακά συστατικά λογισμικού) 62
5.7.4 APIs και άλλα εργαλεία J2EE 63
5.8 Εξυπηρετητές εφαρμογών J2EE 66
5.9 Συσκευασία εφαρμογών 67

6 ΠΛΑΤΦΟΡΜΑ ΑΝΑΠΤΥΞΗΣ SPRING 69
6.1 Εισαγωγή στη Spring 69
6.1.1 Τα modules της Spring 71
6.1.2 Παράδειγμα 74
6.1.3 Κατανοώντας το dependency injection 77
6.1.4 Aspect-Oriented Programming 77
6.2 Αρχιτεκτονικό Μόρφημα MVC 78
6.3 Η αρχιτεκτονική μιας εφαρμογής Spring MVC 80
6.3.1 Επίπεδα Αφαιρέσεων 81
6.3.2 Απομόνωση στρωμάτων 82
6.4 Στρώματα αφαίρεσης σε μια εφαρμογή Spring MVC 82
6.4.1 Στρώμα διεπαφής χρήστη (User Interface) 82
6.4.2 Στρώμα Ιστού (Web Layer) 85
6.4.3 Στρώμα υπηρεσιών (Service Layer) 87
6.4.4 Στρώμα Μοντέλου Τομέα (Domain Model Layer) 89
6.4.5 Σύνοψη 93
7 ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΑΔΕΙΩΝ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΑΙΤΗΣΕΩΝ - ΕΓΚΡΙΣΕΩΝ 95
7.1 Υλοποίηση του στρώματος Μοντέλου (Model Layer) 95
7.1.1 Αντικείμενα Μεταφοράς Δεδομένων (DTO) 96
7.1.2 Αντικείμενα Πρόσβασης Δεδομένων (DAO) 97
7.1.3 Διαχειριστής (Manager) 101
7.2 Υλοποίηση του στρώματος της Προβολής (View Layer) 105
7.3 Υλοποίηση του στρώματος του Ελεγκτή (Controller Layer) 107
7.4 Έλεγχος Γνησιότητας Χρήστη (Authentication) 111
7.4.1 Εφαρμογή διαχείρισης εσωτερικών χρηστών 111
7.4.2 Εφαρμογή διαχείρισης εξωτερικών χρηστών 112
7.5 Βάση Δεδομένων 112
8 ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ 115
9 ΒΙΒΛΙΟΓΡΑΦΙΑ 117

1

Εισαγωγή

1.1 Ηλεκτρονικές Υπηρεσίες

Τη τελευταία δεκαετία παρατηρείται μια αλματώδης ανάπτυξη στις τηλεπικοινωνίες και τα δίκτυα υπολογιστών καθώς και μια ανεξάντλητη ροή πληροφοριών που προσφέρεται μέσω διαδικτύου. Επακόλουθη είναι η εκμετάλλευση αυτών των σύγχρονων τεχνολογιών πληροφορικής και επικοινωνιών για τη βελτίωση των συναλλαγών του πολίτη με τους Δημόσιους Φορείς (διαδικασία που αναφέρεται με τον όρο Ηλεκτρονική Διακυβέρνηση). Η προσπάθεια ψηφιοποίησης όλων των στοιχείων των Δημόσιων Υπηρεσιών και η παροχή υπηρεσιών μέσω διαδικτύου σκοπεύουν στην αύξηση της παραγωγικότητας στην Δημόσια Διοίκηση και την παροχή καλύτερων υπηρεσιών για πολίτες και επιχειρήσεις. Με αυτόν τον τρόπο οι πολίτες κερδίζουν πολύτιμο χρόνο και αποφεύγουν την γραφειοκρατία, ενώ οι υπηρεσίες του Δημόσιου Φορέα γίνονται εύκολα προσβάσιμες, πιο αποτελεσματικές και πιο υπεύθυνες απέναντι στους πολίτες.

Ανάμεσα στις υπηρεσίες που παρέχονται ηλεκτρονικά παρατηρούνται κάποια μοντέλα διαδικασιών τα οποία επαναλαμβάνονται ανεξάρτητα από το είδος της υπηρεσίας. Αναγνωρίζοντας τα μοντέλα αυτά είναι δυνατόν να υλοποιηθούν ηλεκτρονικά ακολουθώντας ένα κοινό αρχιτεκτονικό - σχεδιαστικό πρότυπο. Η κατασκευή τέτοιων αρχιτεκτονικών προτύπων έχει πολλά πλεονεκτήματα. Συντελεί στην εύκολη, γρήγορη και χαμηλού κόστους ανάπτυξη ηλεκτρονικών κυβερνητικών υπηρεσιών, προωθώντας την ηλεκτρονική διακυβέρνηση, αφού για κάθε κατηγορία υπηρεσιών ο βασικός κορμός της υλοποίησης

παραμένει ο ίδιος. Επιπλέον, στις περιπτώσεις που στη ροή εκτέλεσης των διαδικασιών μιας υπηρεσίας είναι απαραίτητη η συμμετοχή πολλών παρεμφερών φορέων, η χρήση ενός κοινού προτύπου καθιστά πιο αποτελεσματική τη συνεργασία των φορέων. Η πολιτική αυτή μπορεί να εφαρμοστεί σε ηλεκτρονικές υπηρεσίες όπως η χορήγηση πιστοποιητικών, η έκδοση αδειών, η κατάθεση φορολογικών δηλώσεων κ.ά.

1.2 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική εργασία με βάση μια συνοπτική μελέτη του γενικότερου πλαισίου της Ηλεκτρονικής Διακυβέρνησης σκοπό έχει την ανάλυση, σχεδίαση και υλοποίηση ενός πρότυπου ηλεκτρονικού επιχειρησιακού συστήματος που υλοποιεί το σύνολο των λειτουργιών μιας υπηρεσίας έκδοσης πιστοποιητικών – αδειών κατόπιν αιτήματος στον αρμόδιο φορέα. Μέσω του διαδικτύου καθίσταται δυνατή η απομακρυσμένη σύνδεση με τον φορέα, κατόπιν της πιστοποίησης του χρήστη μέσω κάποιας ενδιάμεσης υπηρεσίας, για την κατάθεση του αιτήματος του χρήστη προ ς την ίδια την υπηρεσία ή την προώθηση του αιτήματος σε άλλη αρμόδια αρχή.

Η ανάλυση και ανάπτυξη της επιχειρησιακής λογικής του συστήματος που πραγματοποιείται γίνεται με παραμετρικό τρόπο έτσι ώστε να είναι ανεξάρτητη από την κατηγορία ή την ιδιαιτερότητα των πιστοποιητικών που εκδίδει το τελικό σύστημα. Η αρχιτεκτονική του συστήματος Αιτήσεων – Εγκρίσεων υποστηρίζει τις βασικές απαιτήσεις μιας σύγχρονης διαδικτυακής εφαρμογής, όπως ασφάλεια, αξιοπιστία, εγκυρότητα, λειτουργικότητα, ταχύτητα και επεκτασιμότητα. Για την υλοποίηση των απαιτήσεων αυτών χρησιμοποιήθηκαν σύγχρονα τεχνολογικά εργαλεία που βασίζονται στη γλώσσα προγραμματισμού JAVA. Συγκεκριμένα έγινε επιλογή συστατικών λογισμικού από δυο διαφορετικά πλαίσια ανάπτυξης, το Spring και το J2EE και συνδυάστηκαν με σκοπό τη σωστή κατανομή των εργασιών – λειτουργιών του συστήματος σε χωριστά υποσυστήματα. Αποτέλεσμα της συγκεκριμένης εργασίας είναι η υλοποίηση μιας εφαρμογής που μπορεί να προσαρμοστεί εύκολα στις ιδιαίτερες λειτουργικές απαιτήσεις οποιουδήποτε συστήματος αδειών – εγκρίσεων, καθώς και να συντηρηθεί ή να επεκταθεί χωρίς σημαντικό κόστος.

1.3 Οργάνωση κειμένου

Το πρώτο κεφάλαιο αποτελεί την εισαγωγή της διπλωματικής. Αρχικά τονίζεται η αναγκαιότητα ύπαρξης των ηλεκτρονικών υπηρεσιών στις μέρες μας, ενώ επισημαίνεται η ύπαρξη πολλών κοινών γνωρισμάτων μεταξύ τους, γεγονός που μας προτρέπει στην μοντελοποίηση των υπηρεσιών.

Στο δεύτερο κεφάλαιο γίνεται μια θεωρητική προσέγγιση της έννοιας Ηλεκτρονική Διακυβέρνηση, καθώς και μια αναφορά στα είδη της και στα πλεονεκτήματα που εισάγει η χρήση της στη καθημερινότητα των πολιτών. Επιπλέον επισημαίνονται οι προϋποθέσεις που πρέπει να τηρούνται για να είναι εύκολη η χρήση της, καθώς και τα τεχνολογικά ζητήματα που προκύπτουν για την σωστή και ασφαλή εφαρμογή της.

Στο τρίτο κεφάλαιο γίνεται αναφορά στις κατηγορίες υπηρεσιών της Ηλεκτρονικής Διακυβέρνησης, καθώς και στην αναγκαιότητα οργάνωσης των χαρακτηριστικών λειτουργιών που παρέχει το Διαδίκτυο ανά κατηγορία χρήσης. Στη συνέχεια εντοπίζονται κάποια μοντέλα υπηρεσιών που επαναλαμβάνονται σε διαφορετικούς τύπους υπηρεσιών, επισημαίνοντας ότι το μοντέλο αιτήσεων – εγκρίσεων/έκδοσης πιστοποιητικών έχει τη μεγαλύτερη συχνότητα εμφάνισης

Στο τέταρτο κεφάλαιο αναλύονται οι λειτουργικές απαιτήσεις του μοντέλου αιτήσεων-εγκρίσεων, ανεξάρτητα από τον φορέα που ενδέχεται να αντιπροσωπεύει που παρέχει την υπηρεσία και το ή το είδος των εγκρίσεων που εκδίδει.

Στο πέμπτο κεφάλαιο γίνεται μια εισαγωγή στην πλατφόρμας ανάπτυξης Java 2 Enterprise Edition (J2EE), όπου αρχικά τονίζεται η αναγκαιότητα ανάπτυξης της συγκεκριμένης πλατφόρμας. Στη συνέχεια παρουσιάζεται η αρχιτεκτονική της και ο τρόπος λειτουργίας της, καθώς επίσης το πλήθος των τεχνολογιών που χρησιμοποιεί και τα είδη των εφαρμογών που μπορούν να αναπτυχθούν με αυτή. Τέλος, αναφέρονται όλοι οι λόγοι που την καθιστούν ελκυστική για την ανάπτυξη δικτυακών εφαρμογών.

Στο έκτο κεφάλαιο γίνεται μια παρουσίαση της πλατφόρμας ανάπτυξης Spring ως ένα υψηλής ποιότητας και ευχρηστίας project το οποίο ικανοποιεί τις ανάγκες και τις απαιτήσεις των πραγματικών Java εφαρμογών, ενσωματώνοντας όλα τα βασικά χαρακτηριστικά της πλατφόρμας J2EE και προσαρμόζοντάς τα για την αποδοτικότερη ανάπτυξη των εφαρμογών. Στη συνέχεια του κεφαλαίου αναλύεται το αρχιτεκτονικό μόρφημα MVC (Model View Controller) και ο τρόπος που υποστηρίζει η πλατφόρμα Spring την ανάπτυξη εφαρμογών σύμφωνα με αυτό το αρχιτεκτονικό πρότυπο.

Στο έβδομο κεφάλαιο αναλύεται ο τρόπος υλοποίησης του μοντέλου αιτήσεων – εγκρίσεων σε ένα σύστημα ηλεκτρονικής διαχείρισης αιτήσεων-εγκρίσεων σύμφωνα με το αρχιτεκτονικό μόρφημα MVC και τη χρήση των τεχνολογιών που μας προσφέρουν τα πλαίσια J2EE και Spring. Σε όλο το εύρος της ανάλυσης επισημαίνονται τα πλεονεκτήματα χρήσης του πλαισίου εφαρμογών που επιλέχθηκε ώστε να οδηγηθούμε σε μια εφαρμογή ευέλικτη, αξιόπιστη και προσαρμόσιμη σε οποιαδήποτε ενδεχόμενη αλλαγή σε λογικό ή τεχνικό – τεχνολογικό επίπεδο.

Στο όγδοο κεφάλαιο γίνεται μια συνοπτική αναφορά σε κάποια γενικά συμπεράσματα και σκέψεις για τη μελλοντική επέκταση της εργασίας.

Στο ένατο κεφάλαιο παρατίθεται η βιβλιογραφία.

2

ΗΛΕΚΤΡΟΝΙΚΗ ΔΙΑΚΥΒΕΡΝΗΣΗ

2.1 Ορισμός

Ο όρος «Ηλεκτρονική Διακυβέρνηση» αναφέρεται στη χρησιμοποίηση Τεχνολογιών Πληροφορικής και Επικοινωνιών (ΤΠΕ) στη Δημόσια Διοίκηση και Τοπική Αυτοδιοίκηση με στόχο την ψηφιακή παροχή υπηρεσιών προς πολίτες και επιχειρήσεις. Ουσιαστικά η Ηλεκτρονική Διακυβέρνηση αποτελεί μία προσπάθεια στο γενικότερο πλαίσιο εκμετάλλευσης των σύγχρονων τεχνολογιών προκειμένου να μπορεί ο απλός ο πολίτης να διεκπεραιώσει τις υποχρεώσεις του προς τους Δημόσιους Φορείς με τη χρήση των υπολογιστών και του διαδικτύου κερδίζοντας έτσι πολύτιμο χρόνο και αποφεύγοντας την γραφειοκρατία. Επίσης με τον τρόπο αυτό υπάρχει μεγαλύτερη ασφάλεια στην πραγματοποίηση των συναλλαγών, είναι όλα πιο διαφανή και μπορεί να καταπολεμηθεί η διαφθορά αφού όλες οι ενέργειες μπορούν να ελεγχθούν.

2.2 Πλεονεκτήματα χρήσης

Τα πλεονεκτήματα χωρίζονται σε δύο βασικές κατηγορίες ως εξής:

- Αύξηση παραγωγικότητας στην Δημόσια Διοίκηση
- Καλύτερες υπηρεσίες για πολίτες και επιχειρήσεις

Η πρώτη επιτυγχάνεται με την μείωση του κόστους παροχής υπηρεσιών και επικοινωνίας με το κοινό, τον καλύτερο συντονισμό ανάμεσα στους φορείς λόγω χρήσης κοινών προτύπων, την καλύτερη αξιοποίηση των τεχνολογιών πληροφορικής και επικοινωνιών που οδηγεί σε αναδιοργάνωση των διαδικασιών και τη δυνατότητα παροχής νέων υπηρεσιών και μεθόδων λειτουργίας, όπως τηλε-εργασία, τηλε-εκπαίδευση και forums. Η δεύτερη εξασφαλίζεται με την μείωση του χρόνου εξυπηρέτησης και του κόστους για πολίτες και επιχειρήσεις, την αύξηση ασφάλειας και ακεραιότητας δεδομένων, την παροχή υπηρεσιών σε βάση <<24*7>> και την δυνατότητα παροχής υπηρεσιών που δεν κάνουν διακρίσεις σε φύλο, χρώμα και ηλικία. Επιπλέον ο χρήστης μιας ηλεκτρονικής υπηρεσίας δεν χρειάζεται να γνωρίζει τον τρόπο λειτουργίας, τη δομή και τις αρμοδιότητες των οργανωτικών μονάδων της Δημόσιας Διοίκησης που εμπλέκονται στην εξυπηρέτησή του. Η μόνη του ευθύνη είναι να παραλαμβάνει το αποτέλεσμα της υπηρεσίας από ένα σημείο εξόδου χωρίς να εμπλέκεται σε ενδιάμεσα στάδια εξυπηρέτησης (One Stop Shop). Εκτός από τα διαδικαστικά όμως πλεονεκτήματα υπάρχει και μία γενικότερη φιλοσοφία που εμπεριέχεται στην πολιτική της Ηλεκτρονικής Διακυβέρνησης. Ένας από τους βασικούς στόχους της ψηφιακής επανάστασης είναι να ενισχύσει την δημοκρατία και να κάνει τις κυβερνήσεις να ανταποκρίνονται περισσότερο στις ανάγκες των πολιτών. Ουσιαστικά η Ηλεκτρονική Διακυβέρνηση ή αλλιώς το *e*-government, όπως είναι ο όρος που χρησιμοποιείται ευρέως, είναι η χρήση των τεχνολογιών πληροφορικής και επικοινωνιών ώστε να γίνουν η κυβέρνηση και οι υπηρεσίες της πιο προσβάσιμες, πιο αποτελεσματικές και πιο υπεύθυνες απέναντι στους πολίτες. Αυτό μπορεί να επιτευχθεί παρέχοντας μεγαλύτερη πρόσβαση στις κυβερνητικές πληροφορίες, προωθώντας την δημόσια ενασχόληση με την αλληλεπίδραση του κοινού με κυβερνητικά όργανα, μειώνοντας την πιθανότητα διαφθοράς και κερδοσκοπίας κάνοντας όλες τις διαδικασίες πιο προσιτές και κατανοητές στον απλό πολίτη και παρέχοντας δυνατότητες ανάπτυξης σε γεωργικές και παραδοσιακά υποβαθμισμένες περιοχές. Το βασικό όμως μέλημα της Ηλεκτρονικής Διακυβέρνησης είναι να παρέχονται αυτές οι δυνατότητες σε όλους τους πολίτες, ανεξαρτήτως εισοδήματος, και σε όλες τις χώρες, αναπτυγμένες, αναπτυσσόμενες και υποανάπτυκτες. Αυτό με μία λέξη στα αγγλικά αποτελεί το *e*-inclusion. Όλα αυτά βέβαια δεν θα συμβούν απλά με την αγορά περισσότερων υπολογιστών και την δημιουργία δικτυακών τόπων για την παροχή υπηρεσιών. Απαιτείται σοβαρός σχεδιασμός,

σημαντική διάθεση πόρων και πολιτική βούληση για την βελτίωση της καθημερινότητας και την άνοδο του βιοτικού επιπέδου των πολιτών.

2.3 Προϋποθέσεις χρήσης - σωστού σχεδιασμού

Όσον αφορά στην πλήρη εκμετάλλευση των υπηρεσιών Ηλεκτρονικής Διακυβέρνησης είναι απαραίτητη η τήρηση ορισμένων ελάχιστων απαιτήσεων. Το πιο βασικό είναι οι υπηρεσίες της Δημόσιας Διοίκησης να διαθέτουν την απαραίτητη τεχνολογική υποδομή για να παρέχουν τη δυνατότητα ηλεκτρονικής εξυπηρέτησης. Συνακόλουθα οι διαδικασίες της πρέπει να έχουν την κατάλληλη δομή και διασύνδεση ώστε να μπορούν να αξιολογούν τις ΤΠΕ, ενώ και τα στελέχη πρέπει να είναι τεχνολογικά καταρτισμένα για να ανταποκριθούν στις νέες απαιτήσεις και αρμοδιότητες του ρόλου τους. Από την άλλη μεριά βέβαια οι πολίτες και οι επιχειρήσεις πρέπει να έχουν πρόσβαση σε υπολογιστικά συστήματα και επικοινωνιακά μέσα για να έχουν τη δυνατότητα σύνδεσης στο Διαδίκτυο. Τέλος πρέπει να διαθέτουν τις βασικές γνώσεις πληροφορικής ώστε να μπορούν να κάνουν χρήση των ηλεκτρονικών υπηρεσιών που θα τους παρέχονται. Για να γίνουν όμως όλα αυτά εφικτά πρέπει το κράτος και οι δημόσιοι φορείς να κινητοποιηθούν και να παρέχουν κίνητρα σε πολίτες και επιχειρήσεις για την χρησιμοποίηση των νέων τεχνολογιών καθώς και τη δυνατότητα παρακολούθησης σεμιναρίων για την πρόσβαση και τα οφέλη από την χρήση του διαδικτύου στη καθημερινότητα μας. Επίσης τα ΜΜΕ πρέπει να παρέχουν διαρκή ενημέρωση για τις συνεχώς αναπτυσσόμενες τεχνολογίες και τα πλεονεκτήματα που προκύπτουν από αυτές. Τέλος ενδείκνυται το κράτος να προσφέρει επιδοτούμενα προγράμματα προς τις επιχειρήσεις κινητοποιώντας αυτές να εισάγουν τις νέες τεχνολογίες και κυρίως το Διαδίκτυο στην παραγωγική διαδικασία. Κατά την ηλεκτρονικοποίηση, πέντε είναι τα βασικά στοιχεία για να επιτευχθεί μια άκρως λειτουργική χρήση της Ηλεκτρονικής Διακυβέρνησης:

1. Μεταρρύθμιση – Ανασχεδιασμός των διαδικασιών πριν πραγματοποιηθεί η ηλεκτρονικοποίησή τους, γεγονός που θα επιτευχθεί αν αυτές αντιμετωπιστούν από την οπτική γωνία του χρήστη και αποσκοπούν στην εξυπηρέτηση των αναγκών του.
2. Συγκρότηση μιας ομάδας ατόμων που διαθέτουν τις απαραίτητες τεχνικές γνώσεις και εξουσιοδοτηθούν με την επίβλεψη του εγχειρήματος της Ηλεκτρονικής Διακυβέρνησης.
3. Εξασφάλιση των απαραίτητων πόρων, που επιτυγχάνεται κάνοντας ένα σωστό οικονομικό σχεδιασμό τόσο βραχυπρόθεσμο όσο και μακροπρόθεσμο θέτοντας ξεκάθαρους στόχους.
4. Συνεργασία μεταξύ των φορέων της Δημόσιας Διοίκησης και του ιδιωτικού τομέα.
5. Δημόσια ενασχόληση για την βελτίωση της καθημερινότητας των πολιτών.

2.4 Επίπεδα Ηλεκτρονικής Διακυβέρνησης

Το Διαδίκτυο αποτελεί ένα από τα βασικά κανάλια εξυπηρέτησης πολιτών και επιχειρήσεων από τους φορείς του δημοσίου τομέα. Η αποδοχή που γνωρίζει το Διαδίκτυο ως μέσο πρόσβασης σε ηλεκτρονικές υπηρεσίες των φορέων της Δημόσιας Διοίκησης είναι πολύ μεγάλη σε διεθνές επίπεδο. Στην Ελλάδα, η παρουσία των φορέων της Δημόσιας Διοίκησης στο Διαδίκτυο περιορίζεται, εκτός μερικών εξαιρέσεων, στην παροχή πληροφοριακού υλικού για τις υπηρεσίες που παρέχει κάθε φορέας. Επίσης, αρκετοί φορείς διαθέτουν, μέσω του διαδικτυακού τους τόπου, σε ηλεκτρονική μορφή τα έντυπα των αιτήσεων και άλλα έγγραφα που απαιτούνται για τη διεκπεραίωση των συναλλαγών των πολιτών και επιχειρήσεων με το φορέα. Οι παραπάνω πρακτικές ικανοποιούν τα επίπεδα 1 και 2 της Ηλεκτρονικής Διακυβέρνησης. Πιο συγκεκριμένα:

Υπηρεσίες Επιπέδου 1: Αφορούν μόνο στην Πληροφόρηση (Information) των χρηστών για τις διάφορες υπηρεσίες, τον τρόπο παροχής τους, τους εμπλεκόμενους φορείς, τα απαραίτητα δικαιολογητικά και τον συνολικό χρόνο διεκπεραίωσης. Βέβαια για να θεωρηθούν αξιόπιστες και χρήσιμες για τους πολίτες πρέπει να υπάρχει συνεχής ενημέρωση και ανανέωση όποτε υπάρχει αλλαγή επί της διαδικασίας. Χαρακτηριστικό παράδειγμα υπηρεσίας επιπέδου 1 είναι η ενημέρωση από τους Οργανισμούς Τοπικής Αυτοδοίκησης, για παράδειγμα τον δήμο κάποιας περιοχής, για τα δικαιολογητικά που χρειάζονται για τη χορήγηση πιστοποιητικού γέννησης.

Υπηρεσίες Επιπέδου 2: Αφορούν στην Αλληλεπίδραση (Interaction) των χρηστών με τους φορείς της Δημόσιας Διοίκησης. Παρέχουν πληροφοριακό υλικό για τον τρόπο διεκπεραίωσης της υπηρεσίας καθώς και επίσημο υλικό (πρότυπα αιτήσεων, βεβαιώσεων κ.τ.λ.) το οποίο οι χρήστες μπορούν να "κατεβάσουν" στον υπολογιστή τους, να εκτυπώσουν, να επεξεργαστούν και να χρησιμοποιήσουν κατά τη συναλλαγή τους με το φορέα σε φυσικό επίπεδο. Χαρακτηριστικό παράδειγμα αποτελούν ένα μεγάλο μέρος των υπηρεσιών που παρέχονται από τα ΚΕΠ και περιλαμβάνουν τα έντυπα των αιτήσεων με δυνατότητα τοπικής αποθήκευσης στον υπολογιστή, όπως για παράδειγμα η χορήγηση αντιγράφου πτυχίου πανεπιστημίου για στρατολογία ή άλλη χρήση.

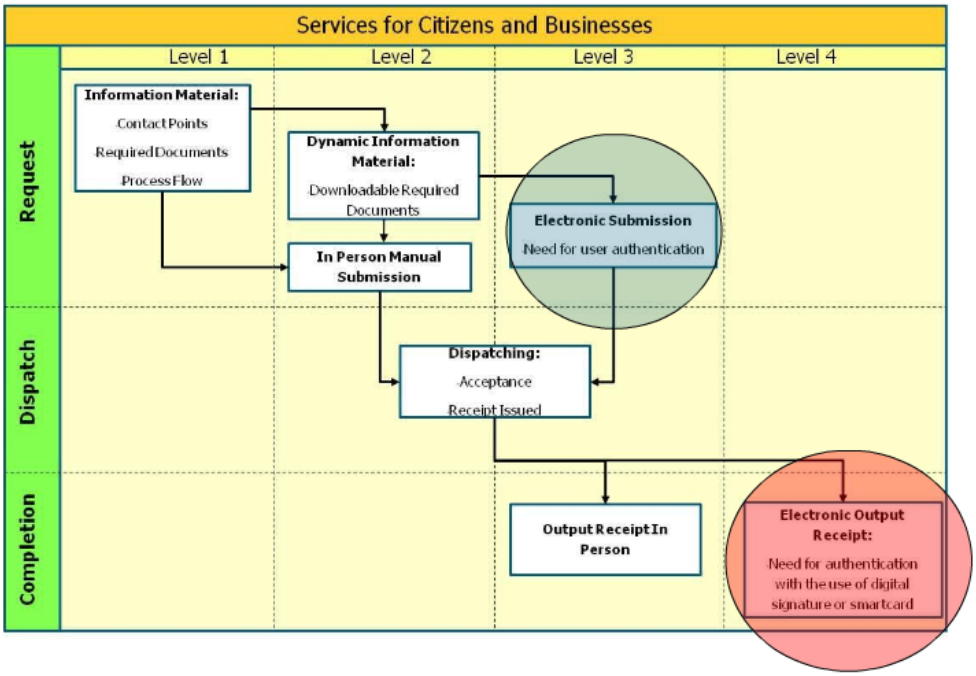
Ωστόσο, η χρησιμότητα των διαδικτυακών τόπων μεγιστοποιείται από την παροχή υπηρεσιών στα επίπεδα 3 και 4 της Ηλεκτρονικής Διακυβέρνησης:

Υπηρεσίες Επίπεδου 3: Αφορούν την Αμφίδρομη διάδραση (Two-way interaction) όπου ο χρήστης αποκτά πρόσβαση σε υπηρεσίες του φορέα με ηλεκτρονικό τρόπο αλλά η διαδικασία ολοκληρώνεται με μη ηλεκτρονικό τρόπο. Χαρακτηριστικό παράδειγμα υπηρεσίας αυτού του επιπέδου είναι η συμπλήρωση και κατάθεση αίτησης για μια βεβαίωση μέσω του διαδικτυακού τόπου του αρμόδιου φορέα και η λήψη της βεβαίωσης με επίσκεψη στο φορέα,

οπού γίνεται και εξακρίβωση των στοιχείων του χρήστη. Εκτός από πληροφορίες, προσφέρουν δηλαδή online φόρμες για συμπλήρωση και ηλεκτρονική αποστολή. Δεδομένου ότι περιλαμβάνουν online υποβολή στοιχείων από μέρους του χρήστη, προϋποθέτουν μηχανισμό αναγνώρισης, ταυτοποίησης και προστασίας των δεδομένων που αποστέλλει ο χρήστης της υπηρεσίας. Παράδειγμα υπηρεσίας επιπέδου 3 είναι η ηλεκτρονική αναζήτηση εργασίας από το δικτυακό τόπο του ΟΑΕΔ.

Υπηρεσίες Επιπέδου 4: Εδώ πρόκειται για Συναλλαγή (Transaction) οπού οι υπηρεσίες αυτού του επιπέδου εκτελούνται πλήρως ηλεκτρονικά, με το αποτέλεσμά τους (π.χ. βεβαίωση) να λαμβάνεται απευθείας από το διαδικτυακό τόπο του φορέα. Σε πολλές από αυτές πραγματοποιούνται και οικονομικές συναλλαγές ηλεκτρονικά. Αυτό σημαίνει δηλαδή ότι έχουμε πλήρη υποκατάσταση της αντίστοιχης μη-ηλεκτρονικής υπηρεσίας. Ομοίως και εδώ απαιτούνται μηχανισμοί αναγνώρισης και ταυτοποίησης οι οποίοι μάλιστα οφείλουν να είναι πιο αυστηροί από τους αντίστοιχους του επιπέδου 3 λόγω της υλοποίησης της υπηρεσίας πλήρως ηλεκτρονικά. Παράδειγμα υπηρεσίας επιπέδου 4 αποτελεί η συμπλήρωση και κατάθεση της φορολογική δήλωσης.

Οι φορείς της Δημόσιας Διοίκησης συνίσταται να προσφέρουν υπηρεσίες επιπέδων 3 και 4 για όσο το δυνατόν μεγαλύτερο αριθμό συναλλαγών τους με πολίτες, γεγονός αυτονόητο που αποσκοπεί στην αποσυμφόρηση της γραφειοκρατίας και στην ταχύτερη εξυπηρέτηση πολιτών και επιχειρήσεων. Σε κάθε περίπτωση, η παροχή ηλεκτρονικών υπηρεσιών στα επίπεδα 3 και 4 προϋποθέτει από το χρήστη τη δήλωση της ταυτότητάς του με την υποβολή κάποιων διακριτικών στοιχείων που του χορηγο ύνται κατά την εγγραφή του στις ηλεκτρονικές υπηρεσίες του φορέα. Στην Εικόνα 2-1 αποδίδονται με τη μορφή σχεδιαγράμματος τα τέσσερα επίπεδα ηλεκτρονικών υπηρεσιών.



Εικόνα **2-1. Επίπεδα Ηλεκτρονικών Υπηρεσιών.**

2.5 Τεχνολογικά ζητήματα

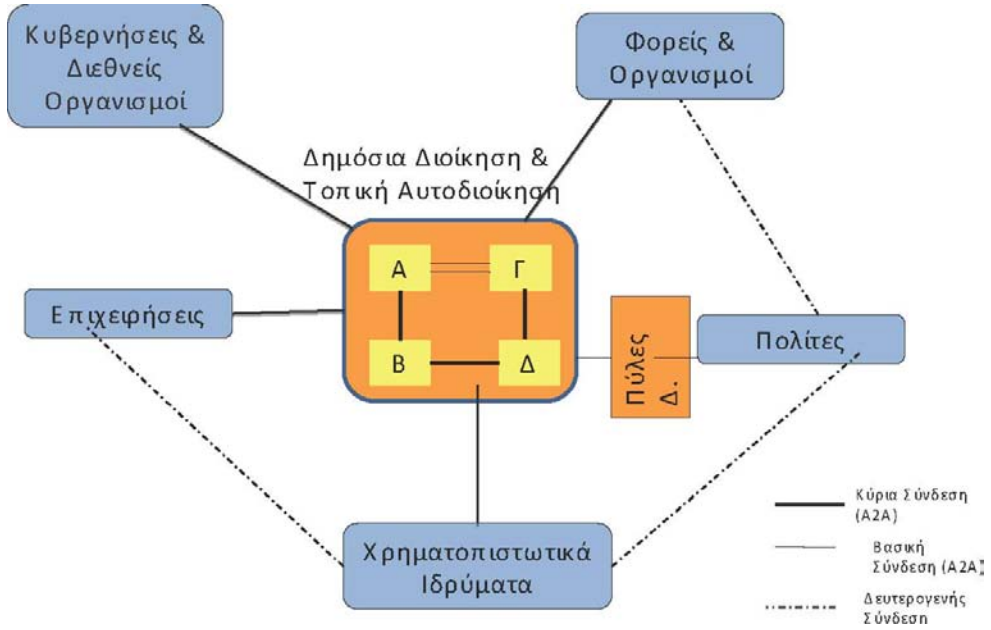
Οι φορείς της Δημόσιας Διοίκησης έχουν να ασχοληθούν με δύο πολύ σημαντικά τεχνολογικά ζητήματα κατά το σχεδιασμό και τη δημιουργία των Δημόσιων Δικτυακών τους Τόπων. Αυτό έχει να κάνει με την Διαλειτουργικότητα του ΔΔΤ και την Αυθεντικοποίηση-Προστασία δεδομένων των χρηστών. Αυτά τα δυο σχετίζονται με τη γρήγορη εξυπηρέτηση και το αίσθημα ασφάλειας και εμπιστοσύνης που πρέπει να νιώσει ο χρήστης του δικτυακού τόπου αντίστοιχα. Επιπλέον για να μπορεί να θεωρηθεί ένας ΔΔΤ αξιόπιστος πρέπει να είναι πάντα διαθέσιμος, γεγονός που εξασφαλίζεται σε επίπεδο τεχνολογικής υποδομής από μία σειρά παραμέτρων που σχετίζονται με τις δικτυακές υποδομές που υποστηρίζουν τη λειτουργία του, τα συστήματα (υλικό και λογισμικό), καθώς και την ανοχή-αντοχή σε παράγοντες-κινδύνους. Μια τελευταία παράμετρος που επηρεάζει τη διαθεσιμότητα ενός δικτυακού τόπου στους δυνητικούς επισκέπτες του είναι η διαθεσιμότητα των υποδομών πρόσβασης των χρηστών στο δικτυακό τόπο.

2.5.1 Διαλειτουργικότητα

Οι διαδικτυακοί τόποι χρησιμοποιούνται από τους φορείς της Δημόσιας Διοίκησης ως μέσο πρόσβασης των πολιτών και επιχειρήσεων στις υπηρεσίες Ηλεκτρονικής Διακυβέρνησης που προσφέρουν και κατά κανόνα δεν παρέχουν από μόνοι τους τις ηλεκτρονικές υπηρεσίες. Αντίθετα, οι ΔΔΤ δρομολογούν τα αιτήματα των χρηστών των υπηρεσιών στα κατάλληλα υποστηρικτικά συστήματα (back-office) των φορέων και γενικά δρουν ως ενδιάμεσοι στην αλληλεπίδραση μεταξύ των φορέων και των χρηστών των ηλεκτρονικών υπηρεσιών.

Συνεπώς, η παροχή υπηρεσιών Ηλεκτρονικής Διακυβέρνησης από τους φορείς της Δημόσιας Διοίκησης μέσω των διαδικτυακών τους τόπων, ιδιαίτερα στα επίπεδα 3 και 4 προϋποθέτει τη διαλειτουργικότητα των συστημάτων των διαδικτυακών τόπων με τα πληροφοριακά συστήματα των φορέων που αναλαμβάνουν τη διεκπεραίωση των υποθέσεων των πολιτών και επιχειρήσεων. Η διαλειτουργικότητα των ΔΔΤ με τα back-office συστήματα των φορέων πρέπει να ακολουθεί κάποια πρότυπα και να ικανοποιεί ορισμένες απαιτήσεις. Όλα αυτά αναφέρονται λεπτομερώς στο Πλαίσιο Διαλειτουργικότητας & Υπηρεσιών Ηλεκτρονικών Συναλλαγών (ΠΔΗΔ) ή όπως είναι ο όρος στα αγγλικά *e*-Government Interoperability Framework (*e*-GIF), το οποίο βέβαια ανανεώνεται και αναπροσαρμόζεται ανάλογα με τις τεχνολογικές εξελίξεις ανά τακτά χρονικά διαστήματα. Αναμφίβολα, η ανταλλαγή δεδομένων μεταξύ ΔΔΤ και back-office συστημάτων καθώς και η διαδικασία εξυπηρέτησης μιας υπόθεσης του πολίτη από τα back-office συστήματα πρέπει να γίνονται με αδιαφανή για το χρήστη τρόπο.

Εκτός από το θέμα της διαλειτουργικότητας μέσα στον ίδιο τον τόπο υπάρχει και θέμα διαλειτουργικότητας διαδικτυακού τόπου με third parties. Υπάρχουν πολλές περιπτώσεις όπου ένας πολίτης δεν γνωρίζει σε ποιον φορέα πρέπει να απευθυνθεί για μία συγκεκριμένη υπηρεσία και αναγκάζεται να αναζητά σχετικές πληροφορίες σε διάφορους διαδικτυακούς τόπους φορέων της Δημόσιας Διοίκησης. Για παράδειγμα, ένας πολίτης μπορεί να μην γνωρίζει εάν για την έκδοση άδειας λειτουργίας ενός καταστήματος πρέπει να απευθυνθεί στην Περιφέρεια, τη Νομαρχία ή τον Δήμο της έδρας του καταστήματος. Ένας τρόπος αντιμετώπισης του προβλήματος αυτού είναι μέσω διαδικτυακών τόπων που συγκεντρώνουν και παρουσιάζουν περιεχόμενο από άλλους τόπους (content aggregation). Βέβαια οι πολιτικές, τα πρότυπα και οι τεχνολογίες που θα χρησιμοποιηθούν είτε για τη συγκέντρωση είτε για τη χρήση περιεχομένου από άλλους διαδικτυακούς τόπους πρέπει να είναι σύμφωνα με τα οριζόμενα στο Πλαίσιο Διαλειτουργικότητας και Υπηρεσιών Ηλεκτρονικών Συναλλαγών. Ακολουθεί ένα διάγραμμα (Εικόνα 2-2) που απεικονίζει τη σημασία της Διαλειτουργικότητας.



Εικόνα 2-2. Σχηματική αναπαράσταση της Διαλειτουργικότητας.

2.5.2 Προστασία προσωπικών δεδομένων – Ασφάλεια συστημάτων

Η ασφάλεια των Δημόσιων Διαδικτυακών Τόπων είναι άμεσα συνυφασμένη με την αξιοπιστία τους και την αποδοχή τους από τους χρήστες-επισκέπτες τους. Οι Δημόσιοι Δικτυακοί Τόποι πρέπει να παρέχουν επαρκές επίπεδο ασφάλειας και αξιοπιστίας διασφαλίζοντας τις εξής παραμέτρους:

1. Ακεραιότητα (integrity): Η πληροφορία που δημοσιεύεται, διακινείται, επεξεργάζεται και αποθηκεύεται παραμένει αναλλοίωτη.
2. Εμπιστευτικότητα (confidentiality): Πρόσβαση στην πληροφορία έχουν μόνο όσοι διαθέτουν κατάλληλη εξουσιοδότηση.
3. Αναγνώριση (identification): Ο προσδιορισμός της ταυτότητας του χρήστη.
4. Πιστοποίηση ταυτότητας (authentication): Η ενέργεια που διασφαλίζει ότι η ταυτότητα που δηλώνει ο χρήστης είναι η πραγματική.
5. Εξουσιοδότηση (authorization): Η εξασφάλιση ότι κάθε οντότητα έχει πρόσβαση στους επιτρεπόμενους σε αυτή πόρους του συστήματος, συμπεριλαμβανομένης της ίδιας της πληροφορίας.
6. Διαθεσιμότητα (availability): Η πληροφορία είναι διαθέσιμη κάθε στιγμή που ένας εξουσιοδοτημένος χρήστης επιχειρεί να αποκτήσει πρόσβαση σε αυτή.
7. Μη άρνηση συμμετοχής (non-repudiation): Ένας χρήστης δεν μπορεί να αρνηθεί ότι εκτέλεσε μία ενέργεια σχετική με πρόσβαση, καταχώρηση και επεξεργασία πληροφορίας.

Η ασφάλεια των ΔΔΤ αποτελείται από ένα σύνθετο πλαίσιο κανόνων και οδηγιών που σχετίζονται με την οργάνωση του φορέα-ιδιοκτήτη του δικτυακού τόπου και του παρόχου που τον φιλοξενεί (στις περιπτώσεις hosting του τόπου σε υποδομές ISP), τις διαδικασίες που εφαρμόζει, τις υπηρεσίες που παρέχει, τις τεχνικές του υποδομές και το νομικό πλαίσιο για ασφάλεια επικοινωνιών και προστασία προσωπικών δεδομένων.

2.5.3 Πρόσβαση-Αυθεντικοποίηση

Οι δημόσιοι διαδικτυακοί τόποι περιλαμβάνουν μεγάλο όγκο περιεχομένου, το οποίο αποτελεί ως επί το πλείστον δημόσια πληροφορία. Από την άλλη πλευρά, οι ηλεκτρονικές υπηρεσίες που παρέχονται από τους ΔΔΤ μπορεί να περιλαμβάνουν την καταχώρηση προσωπικών στοιχείων των χρηστών, την πρόσβασή τους σε δεδομένα που τους αφορούν, την υποβολή αιτήσεων για βεβαιώσεις και άλλες ενέργειες που γενικά σχετίζονται με την πρόσβαση, καταχώρηση και τροποποίηση δεδομένων που δεν αποτελούν δημόσια πληροφορία αλλά συνδέονται άμεσα με τον κάθε χρήστη. Η πρόσβαση σε υπηρεσίες και δεδομένα που δεν έχουν δημόσιο χαρακτήρα είναι απαραίτητο να ελέγχεται. Το περιεχόμενο και οι υπηρεσίες που διατίθενται μέσω ενός ΔΔΤ πρέπει να κατηγοριοποιούνται ανάλογα με το επίπεδο διαβάθμισης/ευαισθησίας του και τις κατηγορίες χρηστών στις οποίες απευθύνονται. Γενικά, σύμφωνα με το ΠΠΔΔΤ προτείνεται να ακολουθούνται οι εξής αρχές:

- Κατά την πρόσβαση σε δημόσια πληροφορία, λειτουργίες του ΔΔΤ όπως αναζήτηση πληροφορίας και γενικά υπηρεσίες επιπέδων 1 και 2, η ταυτοποίηση των χρηστών δεν είναι απαραίτητη.
- Για την πρόσβαση σε πληροφορίες που αφορούν το χρήστη (πολίτη, επιχείρηση, φορέα) και υπηρεσίες επιπέδων 3 και 4, πρέπει να προηγείται εξακρίβωση της ταυτότητας των χρηστών. Το επίπεδο ασφάλειας καθορίζεται ανάλογα με την κρισιμότητα ή ευαισθησία των δεδομένων και υπηρεσιών, ειδικότερα:
    - Για υπηρεσίες που η διαδικασία εξυπηρέτησης ξεκινά με την ηλεκτρονική υποβολή στοιχείων και εγγράφων μέσω του ΔΔΤ αλλά ολοκληρώνεται με μη ηλεκτρονικό τρόπο (π.χ. παραλαβή βεβαίωσης/πιστοποιητικού αυτοπροσώπως, μέσω ΚΕΠ ή ταχυδρομείου, επίπεδο 3), ως διακριτικά ασφάλειας του χρήστη μπορεί να χρησιμοποιούνται το όνομα (username) και το συνθηματικό (password) χρήστη.
    - Για υπηρεσίες που η διαδικασία εξυπηρέτησης είναι πλήρως ηλεκτρονική (επίπεδο 4), πρέπει να χρησιμοποιούνται ισχυρότερα μέτρα δήλωσης και εξακρίβωσης της ταυτότητας όπως πιστοποιητικά που εκδίδονται από Υποδομές Δημοσίου Κλειδιού.

Σύμφωνα με τα παραπάνω, η εξυπηρέτηση των χρηστών ηλεκτρονικών υπηρεσιών που παρέχονται μέσω ΔΔΤ ξεκινά με τη δήλωση της ταυτότητας του χρήστη και την εξακρίβωσή της από τα συστήματα του φορέα. Στη συντριπτική πλειοψηφία των περιπτώσεων, στη διαδικασία εξυπηρέτησης εμπλέκονται επιπλέον αρκετά υποστηρικτικά (back-office) συστήματα του φορέα, τα οποία επεξεργάζονται τα δεδομένα που εισάγει ο χρήστης, διεκπεραιώνουν τη διαδικασία εξυπηρέτησης και παρουσιάζουν πληροφορίες ή το τελικό αποτέλεσμα της υπηρεσίας στο χρήστη μέσω του ΔΔΤ του φορέα. Καθένα από τα συστήματα αυτά μπορεί να απαιτεί επίσης την πιστοποίηση της ταυτότητας του χρήστη, προκειμένου να διεκπεραιωθεί η υπόθεσή του. Οι φορείς της Δημόσιας Διοίκησης πρέπει να διασφαλίζουν ότι τα στοιχεία που υποβάλλει ο πολίτης για την πρόσβασή του στις ηλεκτρονικές υπηρεσίες που παρέχονται μέσω των ΔΔΤ επαρκούν για την εξακρίβωση της ταυτότητάς του και τη διεκπεραίωση των υπηρεσιών. Τα στοιχεία που ανταλλάσσονται κατά την επικοινωνία ενός χρήστη με ένα ΔΔΤ πρέπει να προστατεύονται επαρκώς μέσω χρήσης του πρωτοκόλλου HTTPS (HyperText Transfer Protocol Secure).

2.5.4 Διαθεσιμότητα και απόδοση εξυπηρετητών

Οι εξυπηρετητές που κατά κανόνα σχετίζονται με τη λειτουργία ενός ΔΔΤ είναι:

- ο εξυπηρετητής Διαδικτύου (web server), ο οποίος υποστηρίζει την παρουσίαση του ΔΔΤ στο Διαδίκτυο και τη διεπαφή των χρηστών-επισκεπτών με το ΔΔΤ
- ο εξυπηρετητής εφαρμογών (application server), στον οποίο φιλοξενούνται οι εφαρμογές που υποστηρίζουν τη λειτουργία του ΔΔΤ και τις υπηρεσίες που παρέχει
- ο εξυπηρετητής βάσεων δεδομένων (database server), στον οποίο τηρούνται τα δεδομένα των διαφόρων εφαρμογών.

Οι εξυπηρετητές που υποστηρίζουν τη λειτουργία ενός ΔΔΤ πρέπει να καλύπτουν επαρκώς τις ανάγκες του φορέα και των επισκεπτών του τόπου, σε επίπεδο αριθμού επισκεπτών-χρηστών και όγκου διακινούμενων δεδομένων.

2.5.5 Προσβασιμότητα

Κάθε διαδικτυακός τόπος οφείλει να απευθύνεται στο μεγαλύτερο δυνατό ακροατήριο. Η πρόσβαση στο διαδικτυακό τόπο πρέπει να είναι -κατά το δυνατό- ανεξάρτητη της υποδομής που διαθέτουν οι χρήστες και να λαμβάνει υπόψη τυχόν ιδιαίτερες ανάγκες του κοινού στο οποίο απευθύνεται.

Το περιεχόμενο κάθε Δημόσιου Διαδικτυακού Τόπου πρέπει να αναπτύσσεται με τέτοιο τρόπο ώστε να μην απαιτείται η χρήση συγκεκριμένου φυλλομετρητή ιστού για την πρόσβαση σε αυτό. Κάθε ΔΔΤ πρέπει να είναι προσβάσιμος τουλάχιστον με Internet Explorer και Mozilla Firefox.

2.6 Πρωτοβουλία eEurope-i2010

Στα πλαίσια της πρωτοβουλίας *e*Europe (i2010) γίνεται μία έντονη προσπάθεια στροφής προς το Διαδίκτυο και υλοποίησης όλων των υπηρεσιών μέσω αυτού. Πιο συγκεκριμένα γίνεται προώθηση εννοιών όπως *e*Government (Ηλεκτρονική Διακυβέρνηση), *e*Health (Ηλεκτρονικές Υπηρεσίες Υγείας), *e*Learning (Ηλεκτρονικές Υπηρεσίες Μάθησης) και *e*Business (Ηλεκτρονικό Επιχειρείν). Όλα αυτά βέβαια προσπαθούν να συνδυαστούν με μαζική διάθεση ευρυζωνικής πρόσβασης σε πολίτες και επιχειρήσεις και ταυτόχρονη διασφάλιση των προσωπικών δεδομένων των χρηστών. Το σχέδιο δράσης *e*Europe αποτελεί μέρος της στρατηγικής της Λισσαβόνας, η οποία αποσκοπεί στο να καταστεί η Ευρώπη η πλέον ανταγωνιστική και δυναμική οικονομία της γνώσης έως το 2010.Ο σχεδιασμός του *e*Europe αναπτύσσεται πάνω στους ακόλουθους τέσσερις άξονες δράσης:

Α. Μέτρα πολιτικής για την ανασκόπηση και προσαρμογή της νομοθεσίας σε εθνικό και περιφερειακό επίπεδο, για την ενίσχυση του ανταγωνισμού και της διαλειτουργικότητας, για την ευαισθητοποίηση, καθώς και για την υπογράμμιση της πολιτικής βούλησης.

Β. Η εφαρμογή μέτρων πολιτικής υποστηρίζεται από την ανάπτυξη, ανάλυση και διάδοση ορθής πρακτικής. Θα δρομολογηθούν έργα για την επιτάχυνση της εξάπλωσης εφαρμογών και υποδομής αιχμής.

Γ. Τα μέτρα πολιτικής θα παρακολουθούνται και θα εστιάζονται καλύτερα μέσω συγκριτικής αξιολόγησης της επιτευχθείσας προόδου στην εκπλήρωση των στόχων και των πολιτικών που υποστηρίζουν τους στόχους αυτούς.

Δ. Ο συνολικός συντονισμός των υφιστάμενων πολιτικών θα επιφέρει συνέργια μεταξύ προτεινόμενων δράσεων. Μια διευθύνουσα επιτροπή θα παρέχει καλύτερη εποπτεία των εξελίξεων όσον αφορά τις πολιτικές και θα εξασφαλίζει ικανοποιητική ανταλλαγή πληροφοριών μεταξύ πολιτικών ιθυνόντων σε εθνικό και ευρωπαϊκό επίπεδο και του ιδιωτικού τομέα.

Το πρόθεμα “i” στη στρατηγική i2010 αφορά στα:

- internal market for information services (εσωτερική αγορά για παροχή πληροφοριακών υπηρεσιών)
- investment in ICT innovation for competitiveness (επένδυση στην πρωτοβουλία για χρήση των Τεχνολογιών Πληροφορικής και Επικοινωνιών για την ανταγωνιστικότητα)
- *e*-inclusion and better quality of life (η πολιτική i2010 απευθύνεται σε όλους ανεξαρτήτως διακρίσεων και αποσκοπεί σε μία καλύτερη ποιότητα ζωής).

2.7 Ελληνική Ψηφιακή Στρατηγική και Ηλεκτρονική Διακυβέρνηση

Στην Ελλάδα η Ηλεκτρονική Διακυβέρνηση αποτελεί μέρος της ψηφιακής στρατηγικής 2006-2013,η οποία βασίστηκε στη διάγνωση και τον εντοπισμό της ρίζας των προβλημάτων, στη διεθνή εμπειρία των πιο ανεπτυγμένων τεχνολογικά χωρών στην Ευρώπη, στη μελέτη των διεθνών και ευρωπαϊκών εξελίξεων στον τομέα της κοινωνίας της πληροφορίας (π.χ. i2010) και στη διαμόρφωση πολιτικών σε συνεργασία με τους φορείς και λαμβάνοντας υπ' όψιν τις ιδιαιτερότητες της ελληνικής οικονομίας. Οι 6 βασικές κατευθύνσεις της ψηφιακής στρατηγικής και το πώς αυτές θα προσπαθήσουν να υλοποιηθούν εντοπίζονται στα εξής:

1. στην προώθηση χρήσης ΤΠΕ στις επιχειρήσεις: αυτό θα γίνει με αύξηση της διαθεσιμότητας ευρυζωνικής πρόσβασης, την υποστήριξη ηλεκτρονικών συναλλαγών και τη διάχυση βέλτιστων επιχειρηματικών πρακτικών
2. στην παροχή ψηφιακών υπηρεσιών προς τις επιχειρήσεις και στην αναδιοργάνωση του δημόσιου φορέα: αυτό θα γίνει με βελτίωση διαδικασιών του δημόσιου φορέα μέσω ανασχεδιασμού (BRP) και την παροχή ηλεκτρονικών προμηθειών, ηλεκτρονικών πιστοποιητικών και ηλεκτρονικού one-stop-shop για τις επιχειρήσεις
3. στην υποστήριξη του κλάδου των ΤΠΕ: αυτό θα γίνει με απλούστευση του θεσμικού πλαισίου σχετικού με έργα ΤΠΕ και συντονισμό προώθησης Ελληνικών εταιριών ΤΠΕ στο εξωτερικό
4. στην προώθηση επιχειρηματικότητας σε τομείς που αξιοποιούν ΤΠΕ: αυτό θα γίνει με την απλούστευση της διαδικασίας έναρξης επιχειρήσεων ιδίως αυτών που βασίζονται σε καινοτομικά επιχειρηματικά μοντέλα και με τη βελτίωση επιχειρηματικών δεξιοτήτων στην πανεπιστημιακή εκπαίδευση
5. στη βελτίωση της καθημερινότητας μέσω ΤΠΕ: αυτό θα υλοποιηθεί με ενίσχυση διείσδυσης της ευρυζωνικότητας, με μεγάλης κλίμακας καμπάνιας ενημέρωσης-εξοικείωσης πολιτών με ΤΠΕ, με ηλεκτρονικές υπηρεσίες χρήσιμες για την καθημερινότητα των πολιτών και με την ενίσχυση του ρόλου των ΤΠΕ στην εκπαίδευση ως υποστηρικτικό μέσο σε όλα τα μαθήματα-προγράμματα
6. στην ανάπτυξη ψηφιακών υπηρεσιών για τον πολίτη: αυτό θα γίνει με την μεταφορά των 20 πιο συχνά χρησιμοποιούμενων υπηρεσιών σε πλήρη ηλεκτρονική μορφή και τη διαμόρφωση στοχευμένων δράσεων για την εξυπηρέτηση πολιτών της Περιφέρειας βάσει των τοπικών αναγκών (π.χ. κινητά κέντρα εκπαίδευσης).

Το πρόγραμμα μετασχηματισμού της Ελληνικής Κοινωνίας με εργαλείο τις Τεχνολογίες Πληροφορικής και Επικοινωνιών εμπλέκει στο σχεδιασμό και την υλοποίησή του όλους τους

Δημόσιους Φορείς σε κεντρικό και περιφερειακό επίπεδο και πολλές χιλιάδες μικρομεσαίες επιχειρήσεις και υποστηρίζει πολλαπλούς στόχους κοινωνικής και οικονομικής ανάπτυξης. Ακολουθούν οι 20 βασικές δημόσιες υπηρεσίες οι οποίες έχουν τεθεί σαν στόχος από το i2010 για ηλεκτρονικοποίηση και οι οποίες υιοθετούνται από την ελληνική ψηφιακή στρατηγική. Αυτές αποτελούν σημείο αναφοράς από τα κράτη μέλη της Ευρωπαϊκής Ένωσης για να γίνονται συγκριτικές μελέτες μεταξύ τους χρησιμοποιώντας κοινούς δείκτες αξιολόγησης. Αποτελούνται από 12 υπηρεσίες προς πολίτες και 8 προς επιχειρήσεις και είναι οι ακόλουθες:

Υπηρεσίες προς Πολίτες

- Φόρος εισοδήματος: δήλωση, ειδοποίηση εισφορών, πληρωμή
- Υπηρεσίες εύρεσης εργασίας (συμπεριλαμβανομένης αίτησης για εργασία σε φορείς του δημοσίου)
- Κοινωνική ασφάλιση: επιδόματα ανεργίας, οικογενειακές παροχές, ιατρικά έξοδα, επιδόματα σπουδών
- Προσωπικά έγγραφα (ταυτότητα, διαβατήριο, δίπλωμα οδήγησης, εκλογικό βιβλιάριο, κ.τ.λ.)
- Άδεια κυκλοφορίας (νέα, μεταχειρισμένα και εισαγόμενα οχήματα – αρχική έκδοση και ανανέωση)
- Οικοδομικές άδειες
- Δήλωση στην αστυνομία (π.χ. σε περίπτωση κλοπής)
- Πρόσβαση σε δημόσιες βιβλιοθήκες (κατάλογοι περιεχομένου, εργαλεία αναζήτησης)
- Πιστοποιητικά (γάμου, γέννησης): αίτηση και παράδοση
- Εισαγωγή στην ανώτερη και ανώτατη εκπαίδευση
- Αναγγελία μετακίνησης (αλλαγή διεύθυνσης)
- Υπηρεσίες σχετικές με θέματα υγείας (π.χ. λίστες αναμονής στα νοσοκομεία)

Υπηρεσίες προς Επιχειρήσεις

- Εργοδοτικές εισφορές
- Φόρος εισοδήματος: δήλωση, ειδοποίηση
- Φόρος Προστιθέμενης Αξίας: δήλωση, ειδοποίηση
- Σύσταση εταιρίας
- Δήλωση στατιστικών στοιχείων
- Άδειες εξαγωγών – τελωνεία
- Άδειες σχετικές με περιβαλλοντικά θέματα

- Δημόσιες Προμήθειες

2.8 Νομικό πλαίσιο

Δυστυχώς παρατηρείται έλλειψη ενός ενιαίου νομοθετικού πλαισίου σε επίπεδο ΕΕ, το οποίο δυσχεραίνει την πλήρη υιοθέτηση και λειτουργία συστημάτων Ηλεκτρονικής Διακυβέρνησης. Κάθε χώρα έχει υιοθετήσει διαφορετικό νομοθετικό πλαίσιο. Συνεπώς Θα πρέπει να οριστεί ένα κοινό σύνολο, συμβατό με όλες τις χώρες το οποίο θα διέπει όλες τις ηλεκτρονικές συναλλαγές. Θα πρέπει να υπάρξει ενιαία αντιμετώπιση και θέσπιση κοινών–διακρατικών νόμων σχετικά με θέματα:

- Ασφάλειας Δεδομένων
- Μεταφοράς Δεδομένων
- Έκδοσης και Χορήγησης Ηλεκτρονικών Πιστοποιητικών
- Ηλεκτρονικών Υπογραφών
- Διασφάλισης Ιδιωτικού Απορρήτου

Στην Ελλάδα δεν υπάρχει συγκεκριμένη νομοθεσία αυτή τη στιγμή γύρω από το θέμα της Ηλεκτρονικής Διακυβέρνησης. Σαφώς έχουν θεσπιστεί νόμοι περί Προστασίας Δεδομένων και Ιδιωτικότητας και περί Ηλεκτρονικών Επικοινωνιών ενώ και η Ελευθερία της Πληροφορίας κατοχυρώνεται με άρθρο του Ελληνικού Συντάγματος. Οι Ηλεκτρονικές Υπογραφές και Ηλεκτρονικές Ταυτότητες έχουν κατοχυρωθεί με Προεδρικό Διάταγμα που υλοποιεί συγκεκριμένη ευρωπαϊκή οδηγία.

2.9 Διαδικτυακές Πύλες – Διαδικτυακοί Τόποι

Στόχος της Ηλεκτρονικής Διακυβέρνησης είναι η αξιοποίηση των τεχνολογιών πληροφορικής και επικοινωνιών (ΤΠΕ) με τέτοιο τρόπο ώστε να αναβαθμιστούν ουσιαστικά οι υπηρεσίες εξυπηρέτησης και πληροφόρησης προς όλους τους συναλλασσόμενους (πολίτες, επιχειρήσεις, κ.τ.λ.) με φορείς της Δημόσιας Διοίκησης. Το βασικό μέσο για την πρόσβαση των πολιτών στις ηλεκτρονικά παρεχόμενες υπηρεσίες ενός συστήματος Ηλεκτρονικής Διακυβέρνησης είναι οι Κυβερνητικές Δικτυακές Πύλες (Government Portals), οι οποίες θα πρέπει:

- Να επιτυγχάνουν εύκολη και ασφαλή πρόσβαση σε υπηρεσίες εξυπηρέτησης και πληροφόρησης για πολίτες, επιχειρήσεις, κοινότητες και ομάδες πολιτών.
- Να παρέχουν εύστοχο και προσαρμοσμένο για τις ανάγκες του κάθε χρήστη περιεχόμενο.
- Να προβάλουν ένα φιλικό πρόσωπο προς τον πολίτη.

- Να αποτελούν προοδευτικά, τον προτιμούμενο για τους πολίτες τρόπο συναλλαγής με τις Δημόσιες Υπηρεσίες.

Μια Δικτυακή Πύλη ουσιαστικά είναι ένας one-stop ιστοχώρος προσαρμοσμένος στις απαιτήσεις των χρηστών, ο οποίος έχει τη δυνατότητα να διαμορφώνει τα εργαλεία και τις πληροφορίες που προσφέρει ανάλογα με τις ανάγκες και τα χαρακτηριστικά του ατόμου εκείνου που επισκέπτεται τον ιστοχώρο, με την χρήση πληροφοριών που έχουν αποθηκευτεί σε βάσεις δεδομένων.

Βασικός στόχος της Δικτυακής Πύλης ενός κυβερνητικού φορέα είναι να καταστεί ένα κεντρικό σημείο επαφής του πολίτη για οτιδήποτε αφορά το συγκεκριμένο φορέα, προσφέροντας:

- Ένα μοναδικό σημείο παροχής κάθε πληροφορίας που είναι διαθέσιμη από το φορέα προς το κοινό και κάθε υπηρεσίας-συναλλαγής που διαχειρίζεται ο φορέας.
- Ένα σημείο κεντρικής ενημέρωσης για τις πρωτοβουλίες που αναλαμβάνονται ή εκτελούνται καθώς και για τις δραστηριότητες της πολιτικής ηγεσίας.
- Ένα χώρο πρόσβασης σε οδηγίες για πρακτικά ζητήματα αρμοδιότητας του φορέα.
- Ένα αρχικό σημείο πρόσβασης προς όλους τους επιμέρους φορείς που τελούν υπό την εποπτεία του φορέα.

Για να εκπληρωθεί ο προσδιορισμός αυτός, απαραίτητη προϋπόθεση είναι η δημιουργία κλίματος εμπιστοσύνης στους πολίτες -χρήστες. Η εμπιστοσύνη εμπνέεται εξασφαλίζοντας μια καλή «εμπειρία του χρήστη» από την επίσκεψή του στην κυβερνητική πύλη. Ως εκ τούτου η χρηστικότητα και η λειτουργικότητα πρέπει να λαμβάνονται υπόψη κατά το σχεδιασμό με ιδιαίτερη βαρύτητα. Δεν είναι σπάνιο το φαινόμενο συστημάτων με προβλήματα σχεδίασης, που οι χρήστες θεώρησαν δύσχρηστα και περίπλοκα. Μη μπορώντας να βελτιώσουν την εξυπηρέτηση των χρηστών τους, τα συστήματα αυτά τέθηκαν αναπόφευκτα σε αχρηστία.

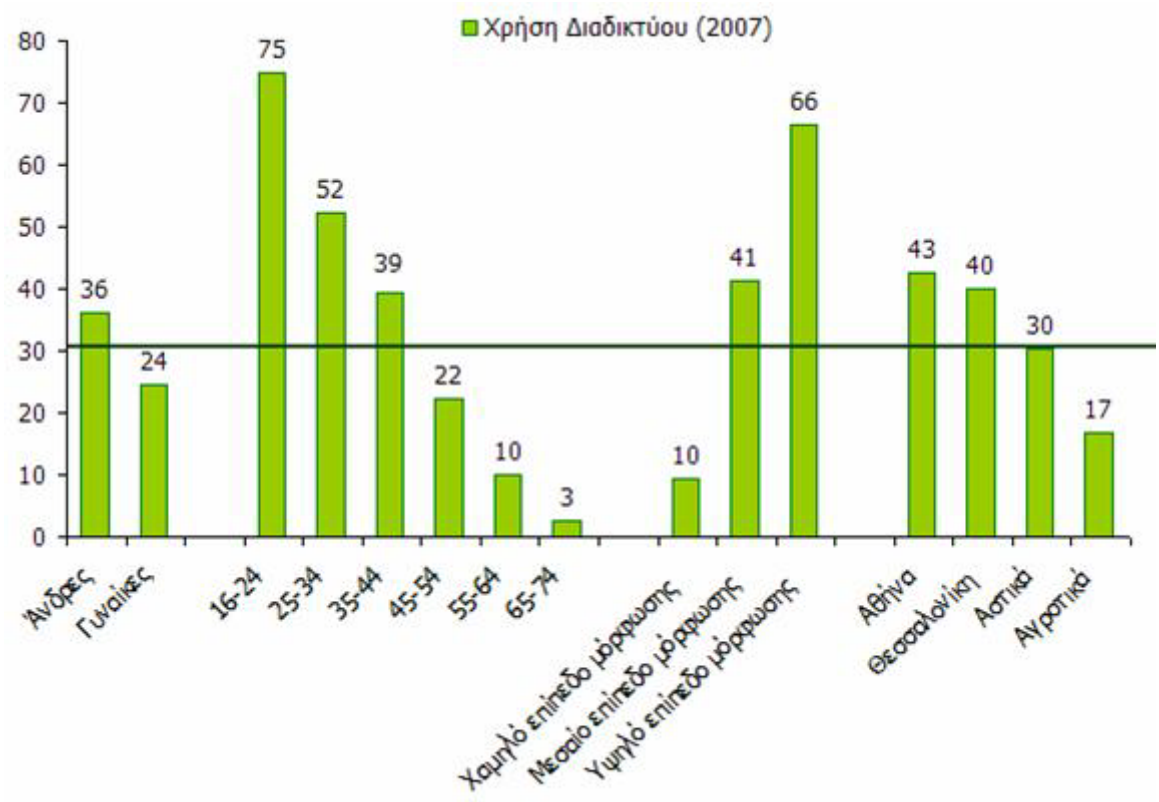
Οι Δικτυακοί Τόποι από την άλλη είναι τα γνωστά sites όπως αυτά είναι ευρέως διαδεδομένα στο Internet και αποτελούν σίγουρα ένα υποσύνολο των Διαδικτυακών Πυλών. Η βασική διαφορά είναι ότι στην πύλη καλύπτονται όλες εκείνες οι πληροφορίες που χρειάζεται να αντλήσει ένας χρήστης γύρω από τις υπηρεσίες και τους εποπτευόμενους φορείς ενός Δημόσιου Φορέα, ενώ κάτι τέτοιο σίγουρα δεν συμβαίνει σε έναν Δικτυακό Τόπο, όπου πολλές φορές υπάρχουν σύνδεσμοι (links) σε άλλους τόπους για την καλύτερη και πληρέστερη ενημέρωση του πολίτη-χρήστη.

2.10Στατιστικά στοιχεία για την χρήση του Internet στην Ελλάδα

2.10.1 Στατιστικά Χρήσης Ίντερνετ για το 2007

Σύμφωνα με μελέτη που δημοσίευσε το «ΠΑΡΑΤΗΡΗΤΗΡΙΟ για την Κοινωνία της Πληροφορίας» σχετικά με το προφίλ των Ελλήνων χρηστών για το 2007 (Εικόνα 2-3), προέκυψαν τα εξής συμπεράσματα:

1) Ο Έλληνας χρήστης εξακολουθεί να είναι άνδρας, νέος, υψηλού μορφωτικού επιπέδου και εισοδήματος που κατοικεί στα μεγάλα αστικά κέντρα.
2) Η χρήση υπολογιστή αλλά και διαδικτύου εξακολουθεί να είναι καθολική σε κατόχους μεταπτυχιακών τίτλων σπουδών.
3) Παρατηρείται αυξητική τάση στην χρήση των νέων τεχνολογιών στις αγροτικές και αστικές περιοχές.



Εικόνα 2-3. Στατιστικά Χρήσης Ίντερνετ για το 2007.

Πιο αναλυτικά, οι άνδρες εξακολουθούν να έχουν την πρώτη θέση στην χρήση του υπολογιστή με ποσοστό 45,1%, έναντι εκείνου των γυναικών, το οποίο αγγίζει μόλις το 32,8%. Επίσης, οι άνδρες χρήστες του Διαδικτύου είναι περισσότεροι από τις γυναίκες, με

ποσοστά 36% και 24% αντίστοιχα. Σημειωτέον, δεν υπάρχει μεγάλη ψαλίδα στην χρήση νέων τεχνολογιών μεταξύ των δύο φύλων στις νεαρές ηλικίες.

Όσον αφορά το μορφωτικό επίπεδο, παρατηρείται σχεδόν καθολική χρήση διαδικτύου και ηλεκτρονικών υπολογιστών σε κατόχους μεταπτυχιακών τίτλων σπουδών (91%). Υψηλά ποσοστά χρήσης εμφανίζουν και οι απόφοιτοι τριτοβάθμιας εκπαίδευσης (με ποσοστό που φτάνει το 78%), στους οποίους μάλιστα παρατηρείται αύξηση σε σχέση με τα προηγούμενα χρόνια, ενώ οι απόφοιτοι δημοτικού και γυμνασίου εμφανίζουν χαμηλά ποσοστά εξοικείωσης και χρήσης του Ίντερνετ.

Θετικό είναι το γεγονός της ανοδικής τάσης χρήσης των νέων τεχνολογιών στο σύνολο σχεδόν της χώρας, με τα υψηλότερα ποσοστά να καταγράφονται στις Περιφέρειες Αττικής (41,2%), Νοτίου Αιγαίου (31,4%) και Κεντρικής Μακεδονίας (29,7%). Επίσης, στις αγροτικές περιοχές, παρατηρείται αυξητική τάση ύψους 11 ποσοστιαίων μονάδων, παρά το γεγονός ότι υστερούν στην χρήση των νέων τεχνολογιών σε σχέση με την υπόλοιπη χώρα.

Τέλος, ως προς τους λόγους χρήσης του Διαδικτύου για τους νέους, πρώτη θέση κατέχουν η ψυχαγωγία και η επικοινωνία, στις ηλικίες 16-24. Οι λίγο μεγαλύτεροι (25-54) το χρησιμοποιούν σε επίπεδο ψηφιακών υπηρεσιών που σχετίζονται με ταξίδια και διαμονή, για παραγγελία αγαθών και υπηρεσιών αλλά και για τραπεζικές συναλλαγές.

Η εν λόγω μελέτη, η οποία ανέλυσε και αξιολόγησε την χρήση των νέων τεχνολογιών κατά φύλο, ηλικία, εκπαίδευση και περιφέρεια δίνει πολύ ενθαρρυντικά αποτελέσματα, καθώς πλέον οι Έλληνες πλησιάζουν αρκετά τα ευρωπαϊκά ποσοστά χρήσης και εξοικείωσης με τις νέες τεχνολογίες, τουλάχιστον σε νεαρές ηλικίες.

2.10.2 Στατιστικά Χρήσης Ίντερνετ για το 2008

Ανοδική είναι η πορεία της χώρας μας όσον αφορά τον ρυθμό αύξησης της χρήσης ηλεκτρονικών υπολογιστών και του Διαδικτύου για το 2008, σύμφωνα τουλάχιστον με τα τελευταία στοιχεία της Εθνικής Στατιστικής Υπηρεσίας. Όπως προκύπτει από την τελευταία δειγματοληπτική Έρευνα Χρήσης Τεχνολογιών Πληροφόρησης και Επικοινωνίας από τα νοικοκυριά, το ποσοστό των ατόμων που χρησιμοποιούσαν υπολογιστή κατά το α' τρίμηνο του 2008 ήταν στο 44,4% (+10% σε σχέση με την αντίστοιχη έρευνα του 2007 που ήταν στο 40,2%), ενώ εκείνο για τους Έλληνες με πρόσβαση στο Διαδίκτυο ήταν στο 38,2% (+14%). Η έρευνα διενεργήθηκε σε τελικό δείγμα 5.045 νοικοκυριών και ισάριθμα μέλη αυτών σε ολόκληρη τη χώρα με κριτήριο την ύπαρξη ενός, τουλάχιστον, μέλους ηλικίας 16-74 ετών σε κάθε νοικοκυριό.

Σε σχέση με το 2007, ο πληθυσμός ηλικίας 16-74 ετών που δεν έχει χρησιμοποιήσει ποτέ υπολογιστή μειώθηκε κατά 4,7 ποσοστιαίες μονάδες. Η μείωση αυτή, όπως επισημαίνεται στη σχετική ανακοίνωση, αντικατοπτρίζει τη συνεχή καθοδική τάση. Στο πρώτο έτος

διενέργειας της έρευνας (2002) το ποσοστό των ατόμων που δεν είχαν χρησιμοποιήσει ποτέ προσωπικό υπολογιστή ήταν περίπου 75%, ενώ σήμερα μόνο 48%. Το ποσοστό χρήσης του Διαδικτύου σε επίπεδο νοικοκυριών διαμορφώθηκε στο 31%, ποσοστό αυξημένο σε σχέση με εκείνο προηγούμενων ετών. Από τα νοικοκυριά που έχουν πρόσβαση στο Διαδίκτυο το 72,5% διαθέτει ευρυζωνική σύνδεση. Τον τελευταίο χρόνο παρατηρείται ραγδαία αύξηση στις ευρυζωνικές συνδέσεις κατά 146% περίπου, και αντιστρόφως μείωση των λοιπών συνδέσεων με modem, μέσω τηλεφωνικής γραμμής (αναλογικής ή ISDN) κατά 60% περίπου.

3

ΤΥΠΟΙ & ΜΟΝΤΕΛΑ ΥΠΗΡΕΣΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΚΥΒΕΡΝΗΣΗΣ

3.1 Τύποι Υπηρεσιών Ηλεκτρονικής Διακυβέρνησης

Οι υπηρεσίες Ηλεκτρονικής Διακυβέρνησης χωρίζονται σε τρεις κατηγορίες ανάλογα με το ποιος τις παρέχει και σε ποιους απευθύνονται. Οι υπηρεσίες αυτές παρέχονται από τη Δημόσια Διοίκηση προς πολίτες, επιχειρήσεις ή εργαζομένους ή ανάμεσα σε φορείς της Διοίκησης και αναλύονται λεπτομερώς στη συνέχεια.

Από τη Δημόσια Διοίκηση προς άλλα μέρη:

- G2C (Government to Citizen) όσον αφορά πολίτες. Στην κατηγορία αυτή περιλαμβάνονται όλες εκείνες οι υπηρεσίες οι οποίες παρέχονται προς πολίτες-χρήστες από τους φορείς της Δημόσιας Διοίκησης, όπως Υπουργεία, Γενικές Γραμματείες και Οργανισμούς Τοπικής Αυτοδιοίκησης (ΟΤΑ) και άλλες δημόσιες υπηρεσίες, όπως ΙΚΑ και ΟΑΕΔ. Ενδεικτικά αναφέρουμε κάποια παραδείγματα, όπως είναι η αίτηση χορήγησης άδειας παραμονής στη χώρα και η αίτηση έγκρισης οικογενειακού επιδόματος από τον ΟΑΕΔ.
- G2B (Government to Business) όσον αφορά επιχειρήσεις. Στην κατηγορία αυτή ανήκουν όλες οι υπηρεσίες που απευθύνονται σε επιχειρήσεις και επίσης παρέχονται από τους κρατικούς φορείς και αφορούν κυρίως στη χορήγηση διαφόρων τύπων

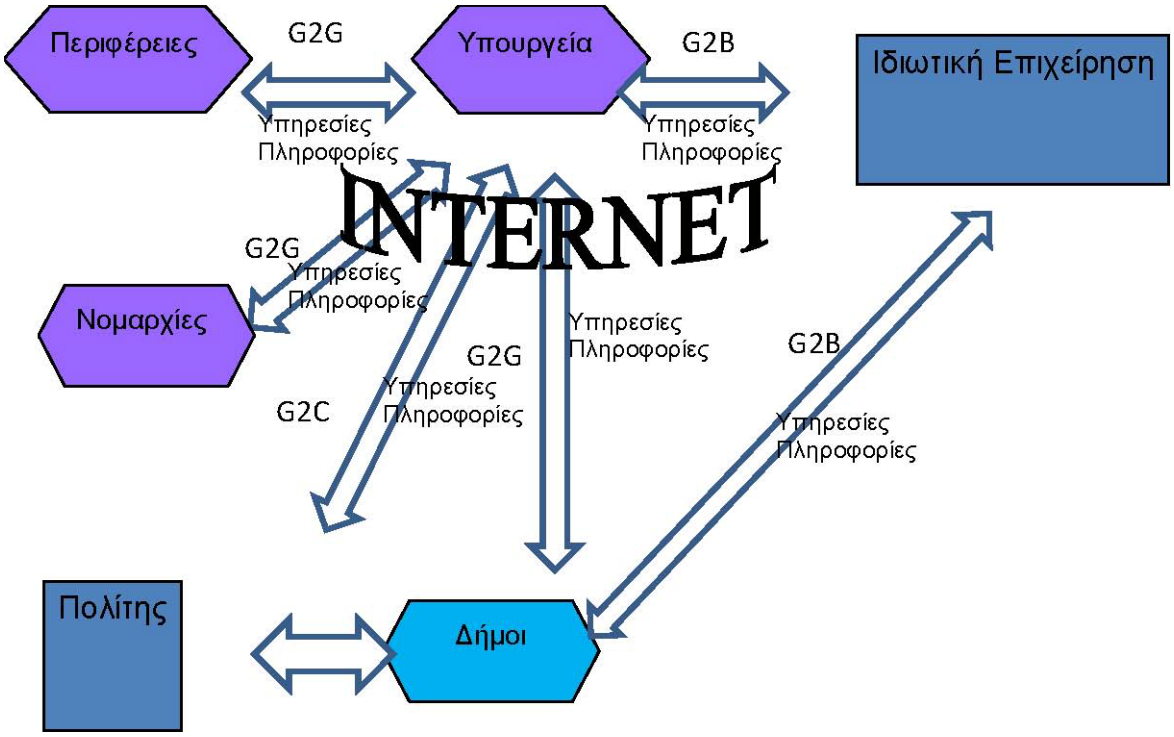
δικαιολογητικών, βεβαιώσεων και αιτήσεων που είναι απαραίτητα για τη σωστή και νόμιμη λειτουργίας τους. Ενδεικτικά ο έλεγχος του δικαιώματος χρήσης της επωνυμίας και του διακριτικού τίτλου της επιχείρησης από το Εμπορικό Βιομηχανικό Επιμελητήριο Αθηνών.

- G2E (Government to Employee) όσον αφορά εργαζομένους. Αυτές αποτελούν ουσιαστικά μία υποκατηγορία των υπηρεσιών προς πολίτες και απευθύνονται σε όλους εκείνους που εργάζονται. Χαρακτηριστικό παράδειγμα η αναζήτηση θέσεων εργασίας από τον ΟΑΕΔ.

Ανάμεσα σε φορείς της Διοίκησης και συγκεκριμένα:

- G2G (Government to Government-national):Η κατηγορία αυτή αναφέρεται στα διάφορα είδη υπηρεσιών που πραγματοποιούνται μεταξύ των φορέων της Δημόσιας Διοίκησης σε εθνικό επίπεδο.
- G2G (Government to Government-international): Στην κατηγορία αυτή ανήκουν όλες εκείνες οι υπηρεσίες που "ξεπερνούν" τα εθνικά σύνορα και υλοποιούνται από την συνεργασία ανάμεσα σε δύο ή περισσότερες χώρες.

Ακολουθεί ένα ενδεικτικό σχεδιάγραμμα (Εικόνα 3-1) όπου επιχειρείται να αποτυπωθούν όσο το δυνατόν καλύτερα τα παραπάνω.

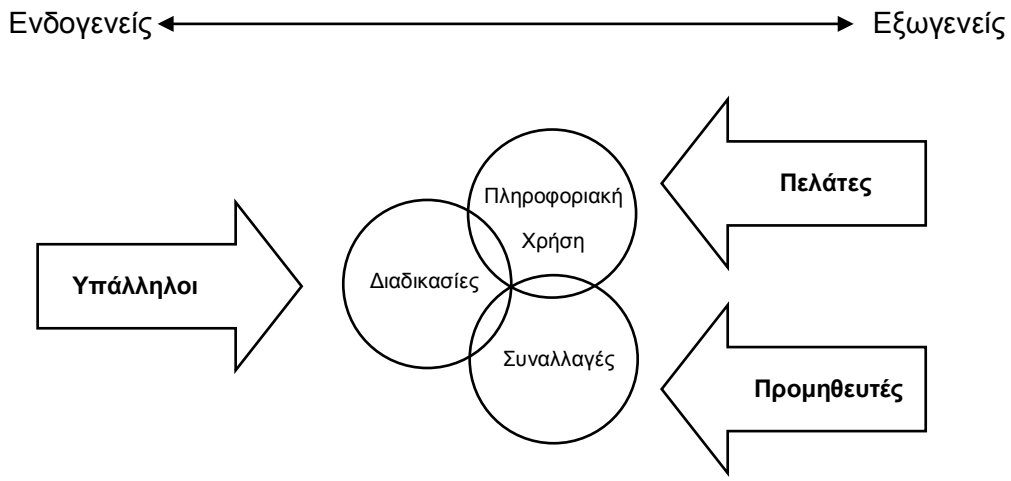


Εικόνα 3-1. Σχηματική απεικόνιση για τους τύπους υπηρεσιών Ηλεκτρονικής Διακυβέρνησης.

3.2 Πλαίσιο & Μοντέλα Υπηρεσιών Ηλεκτρονικής Διακυβέρνησης

Ένας πολύ μεγάλος αριθμός κυβερνητικών οργανισμών έχει «αγκαλιάσει» την Ηλεκτρονική Διακυβέρνηση σαν μια καινοτόμο και αναπόφευκτη δομή δημόσιας υπηρεσίας και διοίκησης. Πολλές λειτουργίες καθώς και αρκετά πιθανά οφέλη έχουν σαφώς αναγνωριστεί. Ωστόσο, δεν υπάρχει κάποιο σχετικό μοντέλο ή πρακτικό πλαίσιο μέσω του οποίου οι διακριτές λειτουργίες της Ηλεκτρονικής Διακυβέρνησης να μπορούν να μελετηθούν και να διαχειριστούν συστηματικά και αποτελεσματικά. Σε αυτή την ενότητα παραθέτουμε το μοντέλο των Τριών Δακτυλίων που προτάθηκε από τους Koh και Balthazard. Αποτελεί ουσιαστικά ένα απλό, διαισθητικό αλλά και αρκετά κατανοητό - επεξηγηματικό πλαίσιο οργάνωσης των χαρακτηριστικών λειτουργιών που παρέχει το Διαδίκτυο. Το επονομαζόμενο λοιπόν μοντέλο των Τριών Δακτυλίων, αιχμαλωτίζει όλες τις εφαρμογές του Διαδικτύου και τις διαχωρίζει σε τρεις κύριες κατηγορίες χρήσης (Εικόνα 3-2):

- Πληροφοριακή χρήση
- Συναλλαγές
- Διαδικασίες



Εικόνα 3-2. Τρεις Πρωταρχικές Χρήσεις του Διαδικτύου.

(1) Πληροφοριακή Χρήση. Οι οργανισμοί χρησιμοποιούν το Διαδίκτυο προκειμένου να διαχέουν την πληροφόρηση, με σκοπό την εκπαίδευση, την ψυχαγωγία, την επιρροή ή απλά την επαφή - επικοινωνία με τον καταναλωτή. Για παράδειγμα, η δημοτική αρχή μιας πόλης μπορεί να χρησιμοποιήσει το Διαδίκτυο για να εκδώσει πληροφορίες σχετικά με τις

υπηρεσίες που προσφέρει στους πολίτες. Αυτή η πληροφοριακή χρήση του Διαδικτύου είναι η πιο πρώιμη μορφή τεχνολογικής εφαρμογής και για πολλούς οργανισμούς αποτελεί ακόμα και σήμερα την επικρατέστερη από τις παρεχόμενες εφαρμογή.

(2) Συναλλαγές. Τη σημερινή εποχή πολλοί οργανισμοί χρησιμοποιούν το Διαδίκτυο προκειμένου να υποστηρίξουν μια καθοδηγούμενη συνέχεια διαδικασιών μεταξύ χρηστών και συστήματος, που εν τέλει έχει ως αποτέλεσμα στη δημιουργία και μεταφορά προστιθέμενης αξίας. Χρησιμοποιώντας το Διαδίκτυο, ένας πολίτης μιας χώρας όχι μόνο είναι σε θέση να παρακολουθήσει και να ενημερωθεί για τους λογαριασμούς του, απέναντι στο δημόσιο, αλλά μπορεί ακόμα να δώσει εντολή για πληρωμή. Αυτή η συναλλαγματική χρήση των εφαρμογών του Διαδικτύου φέρει στο προσκήνιο θέματα τα οποία μέχρι πρότινος δεν είχαν ληφθεί υπόψη ή θεωρούνταν ασήμαντα στις εφαρμογές της πληροφορικής, όπως κατά κύριο λόγο η ασφάλεια.

(3) Διαδικασίες. Το Διαδίκτυο παρέχει εντελώς νέους μηχανισμούς με τους οποίους συνάπτονται επιχειρηματικές διαδικασίες, ολοκληρώνοντας και διασυνδέοντας τη δύναμη που παρέχει η τεχνολογία, με αυτή της ανθρώπινης διανόησης, καθώς και με άλλους πόρους, σε δίκτυα συνεργιών. Η ευρέως διαδεδομένη χρήση καθώς και η «πανταχού παρουσία» του Διαδικτύου, η ικανότητα παρουσίασης και παράθεσης της πληροφορίας με πολυμεσικό τρόπο, η οικειότητα που έχει αναπτύξει το κοινό όσον αφορά τη χρήση των τυποποιημένων browsers καθώς και η διαθεσιμότητα πολλών επιλογών εργαλείων για τη σχεδίαση και κατασκευή ιστοσελίδων, μετατρέπουν το Διαδίκτυο σε μια ολοένα και πιο ελκυστική εναλλακτική λύση για την ολοκλήρωση όχι μόνο των εφαρμογών της εποχής μας, αλλά και συστημάτων της προ Διαδικτυακής εποχής, σε μια και μοναδική πλατφόρμα.

Η χρήση εφαρμογών όπως η διαχείριση των ροών εργασίας, η διαχείριση έργου και το CRM, θα βελτιώσουν την παραγωγικότητα και θα αυξήσουν την επικοινωνία δια-τμηματικά αλλά και δια-επιχειρησιακά, την καθοδήγηση και την συνεργασία μεταξύ επιχειρήσεων.

Η σημαντική πτυχή της ταξινόμησης των εφαρμογών ηλεκτρονικής διακυβέρνησης χρησιμοποιώντας ένα ευρύτερο πλαίσιο όπως αυτό του μοντέλου των Τριών Δακτυλίων, είναι ουσιαστικά διττή.

Κατά πρώτον, επιτρέπει σε αυτούς που σχεδιάζουν και διαχειρίζονται έργα Ηλεκτρονικής Διακυβέρνησης, μια γενικότερη και πιο πολύπλευρη άποψη της συνεχώς αυξανόμενης και συνεχώς μεταβαλλόμενης σειράς Διαδικτυακών εφαρμογών, ούτως ώστε να μην «χαθεί το δάσος, για το δέντρο». Κατά δεύτερον, επιτρέπει στους διαχειριστές (managers) αυτών των πρωτοβουλιών να αναγνωρίσουν και να εστιάσουν μια σειρά από κρίσιμα ζητήματα της κάθε κατηγορίας των λειτουργιών της Ηλεκτρονικής Διακυβέρνησης. Γνωρίζοντας λοιπόν ποια θέματα είναι κρίσιμα και εστιάζοντας την προσοχή τους σε αυτά, θα μειωθεί το κόστος και θα

επιτραπεί στην Ηλεκτρονική Διακυβέρνηση το προνόμιο να παρέχει υπηρεσίες και πληροφόρηση πολύ πιο αποτελεσματικά και αποδοτικά.

Στο πλαίσιο αυτό μπορούν να εντοπισθούν κάποια μοντέλα υπηρεσιών που επαναλαμβάνονται σε διαφορετικούς τύπους υπηρεσιών. Η εκάστοτε λειτουργικότητα μπορεί να διαφέρει ανάλογα με την υπηρεσία που παρέχεται, ωστόσο τα βήματα που ακολουθούνται για την παροχή των υπηρεσιών που ακολουθούν το ίδιο μοντέλο είναι ίδια. Αναγνωρίζοντας τα μοντέλα αυτά μπορεί να αναπτυχθεί μια διαδικασία περιγραφής και ανάπτυξης συστημάτων που ακολουθούν το ίδιο μοντέλο υπηρεσιών Ηλεκτρονικής Διακυβέρνησης.

Τα βασικότερα μοντέλα υπηρεσιών που μπορούν να εντοπισθούν στο πλαίσιο τριών δακτύλων είναι τα έξης:

Αναζήτησης-Παρουσίασης Πληροφοριών (όπως για παράδειγμα η ηλεκτρονική αναζήτηση εργασίας από το δικτυακό τόπο του ΟΑΕΔ και η ενημέρωση από τον δήμο για τα δικαιολογητικά που χρειάζονται για τη χορήγηση πιστοποιητικού γέννησης)

Καταχώρησης Πληροφοριών (όπως για παράδειγμα η συμπλήρωση και κατάθεση της φορολογική δήλωσης)

Κατάθεσης Αιτήσεων- Έγκρισης Αιτήσεων/Εκδοσης Πιστοποιητικών (όπως για παράδειγμα η αίτηση χορήγησης αντιγράφου φορολογικής ενημερότητας από το Υπουργείο Οικονομίας, η αίτηση χορήγησης άδειας οικοδόμησης, η αίτηση για χορήγηση πιστοποιητικού γάμου κ.ά.)

Ηλεκτρονικές ψηφοφορίες (όπως για παράδειγμα η ψηφοφορία των πολιτών ενός δήμου σχετικά με την ανέγερση ή όχι εμπορικού κέντρου στην περιοχή).

3.3 Μοντέλο Κατάθεσης Αιτήσεων - Έγκρισης/Έκδοσης Πιστοποιητικών

Από τα πιο σύνθετα μοντέλα Ηλεκτρονικής Διακυβέρνησης είναι το μοντέλο αυτό, το οποίο χρησιμοποιείται σε πολλά ήδη υπηρεσιών Ηλεκτρονικής Διακυβέρνησης.

3.3.1 Υπηρεσίες Ηλεκτρονικής Διακυβέρνησης που υιοθετούν το μοντέλο

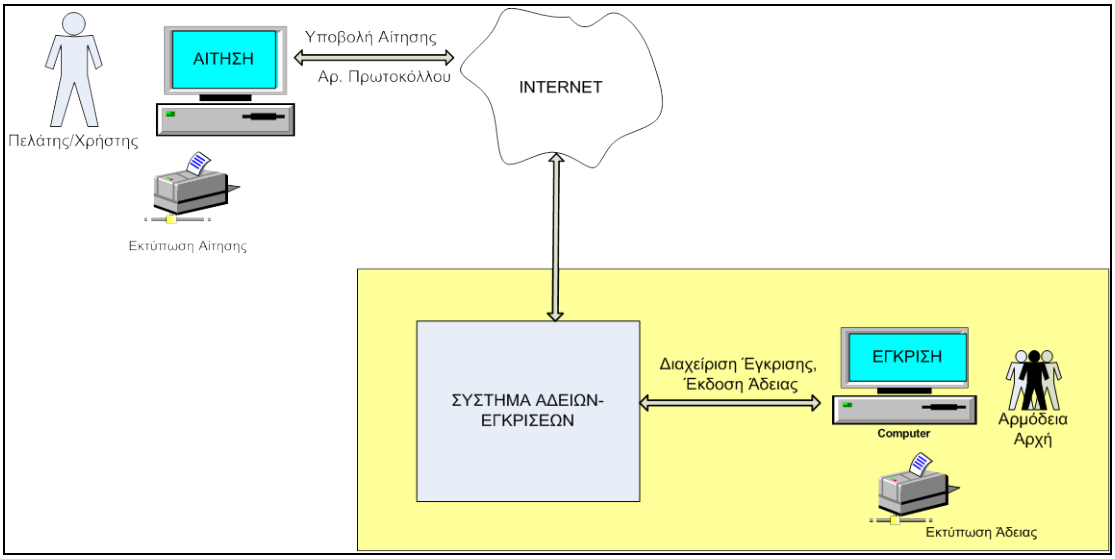
Το σύστημα ηλεκτρονικής διαχείρισης αιτήσεων - εγκρίσεων συγκεντρώνει όλες τις λειτουργίες κεντρικής διαχείρισης των αδειών που απαιτούνται για την διεκπεραίωση διαφόρων συναλλαγών. Οι κύριες λειτουργίες που περιλαμβάνει το σύστημα είναι:

- Έκδοση - Κεντρική Διαχείριση Αδειών.
    - Ηλεκτρονική Υποβολή αίτησης έκδοσης άδειας από πιστοποιημένους χρήστες μέσω διαδικτύου

- Δρομολόγηση αίτησης έκδοσης άδειας στην αρμόδια υπηρεσία
- Έκδοση άδειας
- Έλεγχος υποβολής πρόσθετων πληροφοριών και σχετική γνωστοποίηση του ενδιαφερόμενου
- Ενημέρωση του ενδιαφερόμενου με την ολοκλήρωση της διαδικασίας έγκρισης.

- Παρακολούθηση Αδειών σχετικά με την:
  - Χρονική διάρκεια ισχύος των αδειών (και δυνατότητα ηλεκτρονικής ενημέρωσης των δικαιούχων για ανανέωσή τους)
  - Τήρηση όρων αδειών (και δυνατότητα ηλεκτρονικής ενημέρωσης για τη διενέργεια ελέγχων)
- Ενημέρωση για τις παρεχόμενες άδειες μέσω αναζητήσεων αδειών - εγκρίσεων με πολλαπλά κριτήρια.
- Εκτύπωση των αδειών.

Στην Εικόνα 3-3 παρουσιάζεται διαγραμματικά το σύστημα αιτήσεων - εγκρίσεων:



Εικόνα 3-3. Σύστημα Αιτήσεων – Εγκρίσεων.

3.3.2 Γενική Ροή Διαδικασιών Έκδοσης Αδειών

Κατά την διαδικασία υποβολής και έγκρισης της αίτησης καθώς και έκδοσης της άδειας υπάρχουν δύο βασικές λειτουργίες:

1. Η λειτουργία υποβολής Αίτησης Άδειας
2. Η λειτουργία Έγκρισης Άδειας

Με χρήση της πρώτης λειτουργίας υποβάλλεται μέσω portal η αίτηση από τον αιτούντα την άδεια ενώ με χρήση της δεύτερης λειτουργίας η αρμόδια αρχή ελέγχει και τελικά εκδίδει την άδεια.

3.3.2.1 Υποβολή Αίτησης Άδειας

Οι πιστοποιημένοι χρήστες υποβάλλουν στην αρμόδια αρχή αίτηση για άδεια/πιστοποιητικό μέσω Internet. Η αίτηση υποβάλλεται από το Portal όπου ανάλογα με την αίτηση επιλέγεται και η κατάλληλη ιστοσελίδα. Έπειτα καταχωρούνται όλα τα υπόλοιπα στοιχεία της αίτησης και στη συνέχεια γίνονται οι κατάλληλοι έλεγχοι από το σύστημα Αιτήσεων-Εγκρίσεων. Μετά την επιτυχή υποβολή της αίτησης παράγεται αριθμός πρωτοκόλλου. Ο αριθμός αυτός θα ακολουθεί και την φόρμα της έγκρισης σε όλα τα στάδια μέχρι και την έκδοση της άδειας. Επιπρόσθετα θα υπάρχει δυνατότητα τροποποίησης ή ακύρωσης της αίτησης από τον αιτούντα.

3.3.2.2 Διαδικασία της Έγκρισης-Έκδοση Άδειας

Με τον αριθμό πρωτοκόλλου ο ενδιαφερόμενος προσκομίζει τα απαραίτητα δικαιολογητικά μέσω του διαδικτύου στην αρμόδια αρχή όπου έγινε η αίτηση. Η αίτηση ανακτάται από την αρμόδια αρχή και εάν η υποβολή της αίτησης έγινε σύμφωνα με τις προβλεπόμενες προϋποθέσεις γίνεται αποδοχή της αίτησης αλλιώς η αίτηση δεν γίνεται αποδεκτή. Με την αποδοχή της αίτησης ξεκινάει η διαδικασία της έγκρισης μέχρι την έκδοση της άδειας/πιστοποιητικού. Κατά την έκδοση της άδειας παράγεται ένας μοναδικός κωδικός. Μετά την έκδοση της άδειας η αρμόδια αρχή μπορεί να τροποποιήσει, να αναστείλει ή και να ανακαλέσει την άδεια μετά από δική της απόφαση ή μετά από αίτηση του χρήστη.

4

ΑΝΑΛΥΣΗ ΜΟΝΤΕΛΟΥ ΑΙΤΗΣΕΩΝ-ΕΓΚΡΙΣΕΩΝ

4.1 Κατηγορίες Χρηστών

Υπάρχουν δύο βασικές κατηγορίες χρηστών:

1. **Οι πιστοποιημένοι χρήστες**. Θα έχουν τη δυνατότητα να υποβάλλουν αίτηση από το Portal μέσω Internet ή με φυσική παρουσία στην αρμόδια υπηρεσία. Η αίτηση θα γίνεται προς την αρμόδια αρχή ανάλογα με τον τύπο της άδειας.
2. **Οι εξουσιοδοτημένοι χρήστες-υπάλληλοι**. Θα κάνουν διαχείριση της έγκρισης ή ακόμη θα μπορούν να υποβάλουν αίτηση άδειας για λογαριασμό του χρήστη.

Κατά την διαδικασία εισόδου του χρήστη στην εφαρμογή παρέχεται ένα πρωταρχικό επίπεδο ασφάλειας για την πιστοποίηση και εξουσιοδότησή του από τον εξυπηρετητή. Υπάρχουν δύο κατηγορίες εξουσιοδοτημένου χρήστη: α) Διαχειριστή-Ελεγκτή (supervisor) και β) Συναλλασσόμενου-Πελάτη (client). Στον καθένα παρέχεται ένα διαφορετικό πλήθος δυνατοτήτων χρήσης του συστήματος.

Με την είσοδο, ο συναλλασσόμενος μπορεί και να αναζητήσει παλαιότερες αιτήσεις που έχει υποβάλλει ή έχει αποθηκεύσει προσωρινά καθώς και τις άδειες που έχουν ήδη εκδοθεί. Αντίστοιχα ο ελεγκτής μπορεί να επιλέξει έναν τύπο άδειας από ένα μενού επιλογών και να αναζητήσει όλες τις αιτήσεις και άδειες των πελατών. Συγκεκριμένα έχει τη δυνατότητα

αναζήτησης αιτήσεων και αδειών σύμφωνα με τα ακόλουθα κριτήρια: α) τύπος άδειας, β) όνομα/ΑΦΜ συναλλασσόμενου, γ) ημερομηνία υποβολής αιτήσεων και αδειών.

4.2 Λειτουργικές Απαιτήσεις

Στην ενότητα αυτή παρουσιάζονται οι λειτουργικές απαιτήσεις του συστήματος ανά χρήστη ή ανά ομάδα/ενότητα κάποιας αρμόδιας αρχής.

4.2.1 Λειτουργίες εκτελούμενες από εγκεκριμένο χρήστη

Οι ενέργειες που μπορούν να εκτελεστούν από τον εγκεκριμένο χρήστη συνοψίζονται παρακάτω:

4.2.1.1 Υποβολή Αρχικής Αίτησης

Η διαδικασία ξεκινάει με την υποβολή αρχικής αίτησης του συναλλασσόμενου από το Portal και έχει ως αποτέλεσμα την δημιουργία αρχικής αίτησης με την δημιουργία αριθμού πρωτοκόλλου.

ΠΑΡΑΤΗΡΗΣΗ:

Υπάρχει η δυνατότητα πρόχειρης αποθήκευσης των αιτήσεων σύμφωνα με την οποία:

1. Δεν παράγεται αριθμός πρωτοκόλλου και ο αριθμός αίτησης είναι μία αυτόματη αρίθμηση από το σύστημα.
2. Η πρόχειρη αίτηση αποθηκεύεται στην Βάση Δεδομένων όπως και στην κανονική υποβολή που περιγράφτηκε πιο πάνω.
3. Η πρόχειρη αίτηση μπορεί να ανακτάται και να τροποποιείται από το συναλλασσόμενο. Ανάκτηση της πρόχειρης αίτησης δεν είναι δυνατή από τον ελεγκτή.
4. Η πρόχειρη αίτηση μπορεί μετά από ανάκτηση και τροποποίηση να υποβληθεί κανονικά οπότε πλέον παράγεται και αριθμός πρωτοκόλλου. Η αίτηση αυτή διεκπεραιώνεται στην συνέχεια από τον ελεγκτή.

Ο σκοπός της πρόχειρης αίτησης είναι να μπορεί ο συναλλασσόμενος να αποθηκεύει πρόχειρα την αίτησή του μέχρι να συμπληρώσει όλα τα απαραίτητα στοιχεία. Στην συνέχεια όταν η πρόχειρη αίτηση συμπληρωθεί κανονικά τότε γίνεται και η οριστική υποβολή της αίτησης και παράγεται και αριθμός πρωτοκόλλου.

4.2.1.2 Υποβολή Τροποποιητικής Αίτησης

Η διαδικασία ξεκινάει με την υποβολή τροποποιητικής αίτησης του συναλλασσόμενου από το Portal και έχει ως αποτέλεσμα την δημιουργία τροποποιητικής αίτησης με την δημιουργία αριθμού πρωτοκόλλου.

4.2.1.3 Υποβολή Τροποποιητικής Άδειας

Η διαδικασία ξεκινάει με την υποβολή τροποποιητικής άδειας του συναλλασσόμενου από το Portal και έχει ως αποτέλεσμα την δημιουργία τροποποιητικής άδειας με την δημιουργία αριθμού πρωτοκόλλου. Η διαδικασία είναι παρόμοια με την προηγούμενη με την διαφορά ότι εδώ έχει προηγηθεί έκδοση άδειας και επομένως η διαδικασία της έκδοσης άδειας θα πρέπει να επαναληφθεί.

4.2.1.4 Υποβολή Ακύρωσης Αίτησης

Η διαδικασία ξεκινάει με την υποβολή αίτησης ακύρωσης του συναλλασσόμενου από το Portal και έχει ως αποτέλεσμα την δημιουργία αίτησης ακύρωσης και την ταυτόχρονη δημιουργία αριθμού πρωτοκόλλου. Ο σκοπός είναι να τερματισθεί η διαδικασία της έκδοσης της άδειας με αίτηση από τον συναλλασσόμενο. Η ακύρωση της σχετικής έγκρισης εκτελείται μετά από απόφαση του ελεγκτή.

4.2.1.5 Υποβολή Αίτησης Ανάκλησης Άδειας

Η διαδικασία ξεκινάει με την υποβολή αίτησης ανάκλησης άδειας του συναλλασσόμενου από το Portal και έχει ως αποτέλεσμα την δημιουργία αίτησης ανάκλησης άδειας με την δημιουργία αριθμού πρωτοκόλλου. Ο σκοπός είναι η ανάκληση της άδειας με αίτηση από τον συναλλασσόμενο. Η ανάκληση της άδειας εκτελείται μετά από σχετική απόφαση του ελεγκτή.

4.2.2 Λειτουργίες εκτελούμενες από την Αρμόδια Αρχή

Οι ενέργειες που μπορούν να εκτελεστούν από τον επιβλέποντα, ο οποίος ανήκει σε μια αρμόδια αρχή, συνοψίζονται παρακάτω:

4.2.2.1 Αποδοχή/Μη Αποδοχή Αίτησης-Δημιουργία Έγκρισης

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας αίτησης από τον ελεγκτή. Σε περίπτωση που η αίτηση είναι κατάλληλα συμπληρωμένη και πληροί κάποιες προϋποθέσεις αυτή γίνεται αποδεκτή και έχει ως αποτέλεσμα την δημιουργία της έγκρισης. Σε περίπτωση που υπάρξει ανάγκη τροποποίησης κάποιων στοιχείων της αίτησης ο ελεγκτής μπορεί να κάνει μεταβολή των στοιχείων αυτών οπότε και η παραγόμενη έγκριση θα περιέχει τα

διορθωμένα στοιχεία. Σε περίπτωση που δεν πληρούνται οι προϋποθέσεις η αίτηση δεν θα γίνεται αποδεκτή.

4.2.2.2 Τροποποίηση Έγκρισης

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας έγκρισης από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η μεταβολή των στοιχείων της έγκρισης.

4.2.2.3 Έκδοση Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας έγκρισης από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η έκδοση της άδειας με την δημιουργία αριθμού άδειας και ο καθορισμός της ημερομηνίας έναρξης ισχύος της άδειας.

4.2.2.4 Τροποποίηση Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας άδειας από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η τροποποίηση των στοιχείων της άδειας. Υπάρχουν τρεις περιπτώσεις:

- Να δημιουργηθεί νέος αριθμός άδειας
- Να δημιουργηθεί νέα έκδοση άδειας με τον ίδιο αριθμό άδειας.
- Να παραμείνει η άδεια στην κατάσταση που ήταν πριν την μεταβολή (περίπτωση που η τροποποίηση οφείλεται σε προηγούμενο λάθος του ελεγκτή).

4.2.2.5 Απόρριψη Έγκρισης

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας έγκρισης από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η απόρριψη της έγκρισης και επομένως και ο τερματισμός της διαδικασίας της έγκρισης.

4.2.2.6 Ακύρωση Απόρριψης Έγκρισης

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας ακυρωμένης έγκρισης από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η μεταβολή της κατάστασης της έγκρισης από ακυρωθείσα σε ενεργή και επομένως η διαδικασία της έγκρισης μέχρι και την έκδοση της άδειας μπορεί να συνεχισθεί.

4.2.2.7 Ανάκληση/Ακύρωση Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας άδειας από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η ανάκληση ή ακύρωση της άδειας και επομένως η ισχύς της άδειας λήγει.

4.2.2.8 Ακύρωση Ανάκλησης/Ακύρωσης Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας ανακληθείσας/ακυρωθείσας άδειας από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η μεταβολή της κατάστασης της άδειας σε εκδοθείσα (ενεργή) και επομένως η άδεια επανέρχεται σε ισχύ.

4.2.2.9 Αναστολή Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας άδειας από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η μεταβολή της κατάστασης της άδειας σε άδεια σε αναστολή και επομένως η ισχύς της άδειας παύει προσωρινά.

4.2.2.10 Ακύρωση Αναστολής Άδειας

Η διαδικασία ξεκινάει με την ανάκτηση των στοιχείων μιας άδειας σε αναστολή από τον ελεγκτή. Το αποτέλεσμα της διαδικασίας αυτής είναι η μεταβολή της κατάστασης της άδειας σε άδεια εκδοθείσα και επομένως η άδεια επανέρχεται σε ισχύ.

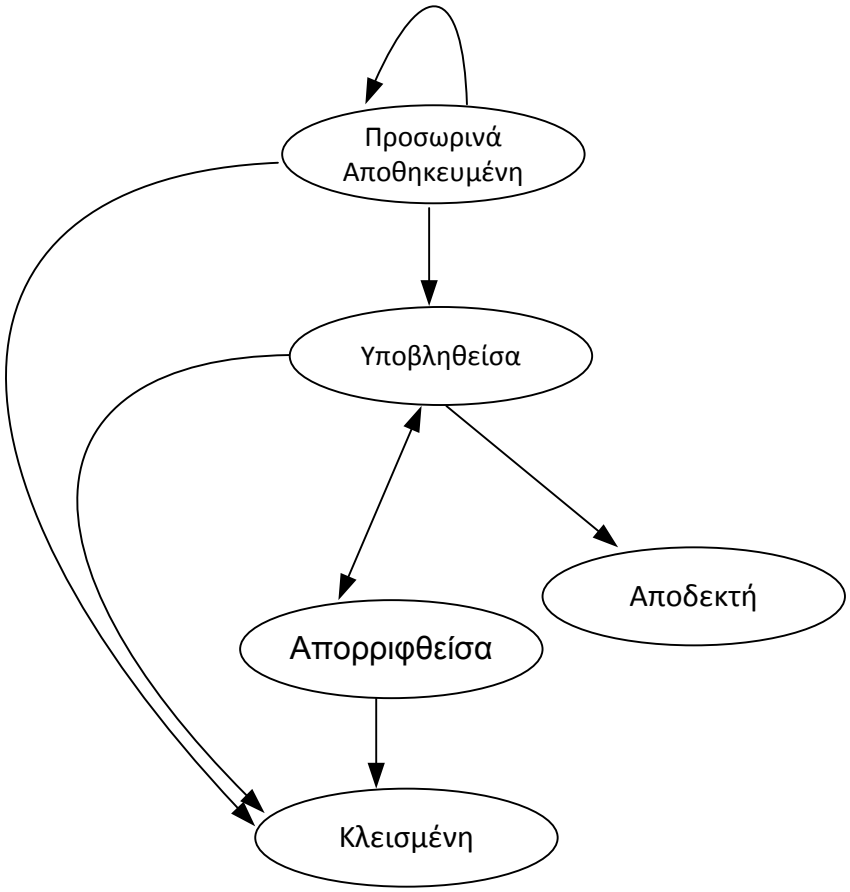
4.2.3 Κατάσταση Αίτησης, Άδειας - Έγκρισης

Κάθε αίτηση – έγκριση από τη πρώτη στιγμή δημιουργίας της χαρακτηρίζεται από έναν δείκτη που αφορά τη κατάσταση «status» στην οποία βρίσκεται και ενημερώνεται κάθε φορά που καταχωρείται μια αλλαγή είτε από τον εγκεκριμένο χρήστη είτε από τον υπεύθυνο της αρμόδιας αρχής.

4.2.3.1 Σύνολο καταστάσεων μιας Αίτησης

Η κατάσταση (status) μιας αίτησης (Εικόνα 4-1) μπορεί να καθοριστεί από το σύνολο των παρακάτω κατηγοριών:

- Προσωρινά Αποθηκευμένη
- Υποβληθείσα
- Αποδεκτή
- Απορριφθείσα
- Κλεισμένη

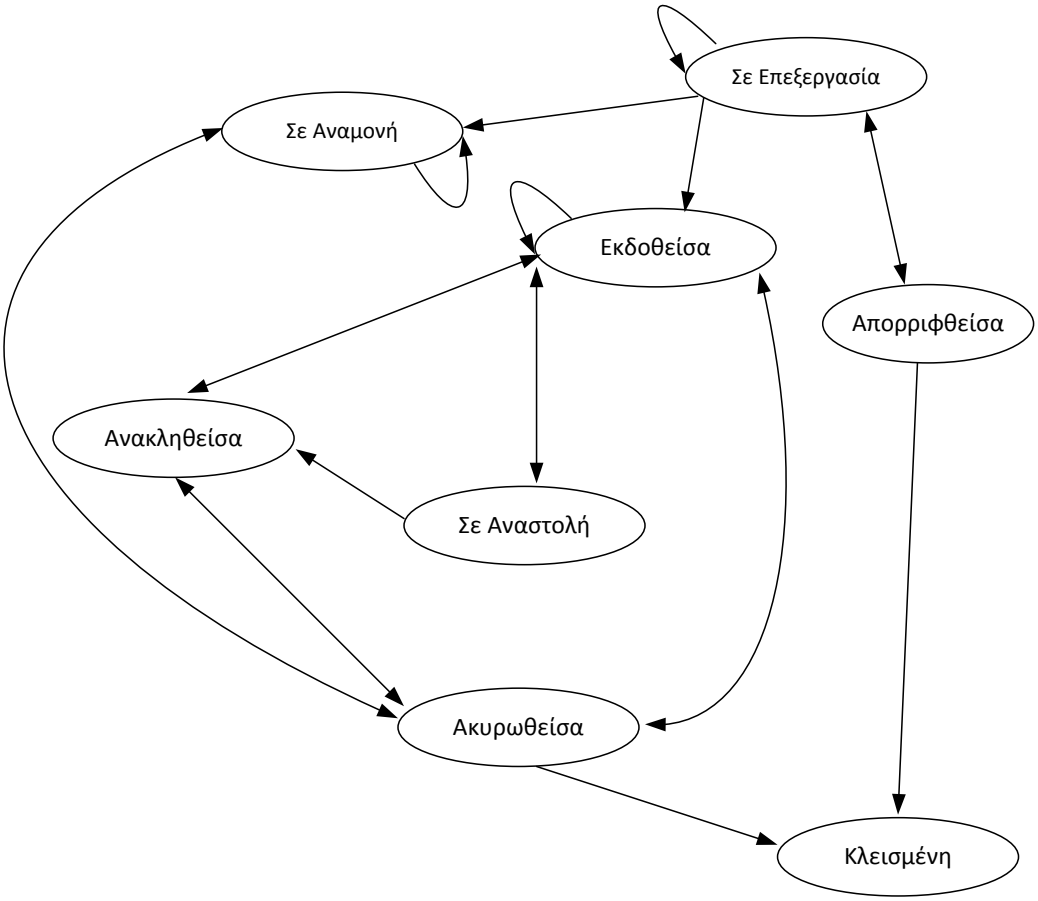


Εικόνα 4-1. Κατάσταση (status) μιας Αίτησης.

4.2.3.2 Σύνολο καταστάσεων μιας Άδειας - Έγκρισης

Η κατάσταση (status) μιας άδειας - έγκρισης (Εικόνα 4-2) μπορεί να καθοριστεί από το σύνολο των παρακάτω κατηγοριών:

- Σε Επεξεργασία
- Απορριφθείσα
- Σε Αναμονή
- Εκδοθείσα
- Σε Αναστολή
- Ανακληθείσα
- Ακυρωθείσα
- Κλεισμένη



Εικόνα 4-2. Κατάσταση (status) μιας Άδειας - Έγκρισης.

4.3 Ανάλυση των λειτουργιών του συστήματος ανά περίπτωση χρήσης

Στη συνέχεια αναλύονται οι ενέργειες στις οποίες μπορεί να προβεί ο συναλλασσόμενος ή ο επιβλέπων ανά κατάσταση αίτησης ή άδειας και οι αντίστοιχες διαδικασίες που ακολουθούνται. Αρχικά αναφέρεται η κατάσταση στην οποία βρίσκεται μια αίτηση-άδεια και στη συνέχεια αναλύονται οι επιμέρους διαδικασίες που ακολουθούνται για κάθε ενέργεια.

Αναλόγως την ενέργεια την οποία θα εκτελέσει ο συναλλασσόμενος ή ο επιβλέπων, η κατάσταση μιας αίτησης ή μιας άδειας μπορεί να αλλάξει. Όταν η αίτηση γίνεται αποδεκτή και δεν είναι αίτηση ακύρωσης, δημιουργείται αντίγραφό της, το οποίο αντιπροσωπεύει την άδεια-έγκριση και είναι σε κατάσταση «Σε Επεξεργασία». Οποιαδήποτε ενέργεια πάνω στην άδεια ή την αίτηση, εκτός από την ενέργεια της αποθήκευσης, έχει σαν αποτέλεσμα την δημιουργία ενός νέου αντιγράφου που περιέχει τις αλλαγές.

4.3.1 Αίτηση

4.3.1.1 «Προσωρινά Αποθηκευμένη»

⇒ Αποθήκευση

Αποθηκεύεται η αίτηση σε κατάσταση «Προσωρινά Αποθηκευμένη». Αν στην αίτηση έχει καθοριστεί ο τύπος άδειας τότε πραγματοποιείται έλεγχος (αναλύεται παρακάτω) αν ο συγκεκριμένος συναλλασσόμενος μπορεί να υποβάλει την αίτηση. Αν δεν πληρούνται οι προϋποθέσεις του ελέγχου η αίτηση δεν αποθηκεύεται.

⇒ Διαγραφή

Διαγράφεται η αίτηση από την Βάση Δεδομένων.

⇒ Υποβολή

Αν υπάρχει εγγραφή που αντιστοιχεί σε αίτηση που είναι στην κατάσταση «Προσωρινά αποθηκευμένη», η εγγραφή αυτή περνάει στο ιστορικό και δημιουργείται μια νέα εγγραφή σε κατάσταση «Υποβληθείσα». Διαφορετικά εισάγεται κατευθείαν μια νέα εγγραφή σε κατάσταση «Υποβληθείσα».

Πριν την υποβολή πραγματοποιείται έλεγχος (αναλύεται παρακάτω) αν ο συγκεκριμένος συναλλασσόμενος μπορεί να υποβάλει την αίτηση. Αν δεν πληρούνται οι προϋποθέσεις του ελέγχου η αίτηση δεν υποβάλλεται.

Αν η αίτηση που υποβάλλεται αφορά την τροποποίηση αίτησης πραγματοποιείται (αναλύεται παρακάτω) για το αν η αίτηση που τροποποιείται είναι σε κατάσταση «Υποβληθείσα». Αν αυτό ισχύει τότε η αρχική αίτηση προωθείται σε κατάσταση «Κλεισμένη» και η αίτηση τροποποίησης περιέρχεται σε κατάσταση «Υποβληθείσα».

Αν η αίτηση που υποβάλλεται αφορά την ακύρωσης αίτησης πραγματοποιείται έλεγχος (αναλύεται παρακάτω) για το αν η αίτηση που πρόκειται να ακυρωθεί είναι σε κατάσταση «Υποβληθείσα». Αν αυτό ισχύει τότε η αρχική αίτηση και η αίτηση ακύρωσης προωθούνται σε κατάσταση «Κλεισμένη».

Συνεπώς όταν ο ελεγκτής δεν έχει επεξεργαστεί μια αίτηση (κατάσταση «Υποβληθείσα») μπορεί ο συναλλασσόμενος να την τροποποιεί ή να την ακυρώσει χωρίς να απαιτείται κάποια άλλη ενέργεια από μέρους του ελεγκτή. Με κάθε υποβολή αίτησης δημιουργείται νέος αριθμός πρωτοκόλλου.

Σε περίπτωση που έχει επιλεγεί αίτηση για τροποποίηση άδειας, αποθηκεύεται ο κωδικός της άδειας που πρόκειται να τροποποιηθεί για διατήρηση ιστορικότητας των αδειών του συγκεκριμένου συναλλασσόμενου.

4.3.1.2 «Υποβληθείσα»

⇒ Αποδοχή

Η αίτηση προωθείται σε κατάσταση «Αποδεκτή» και στην συνέχεια ανάλογα με τον τύπο της αίτησης που γίνεται αποδεκτή πραγματοποιούνται οι αντίστοιχες ενέργειες αποδοχής μιας Αίτησης (αναλύονται παρακάτω).

⇒ Απόρριψη

Η αίτηση προωθείται σε κατάσταση «Απορριφθείσα».

4.3.1.3 «Αποδεκτή»

Δεν υπάρχουν ενέργειες για τις αιτήσεις που βρίσκονται στην κατάσταση αυτή.

4.3.1.4 «Απορριφθείσα»

⇒ Ακύρωση Απόρριψης

Ακολουθείται ξανά όλη η διαδικασία της Υποβολής.

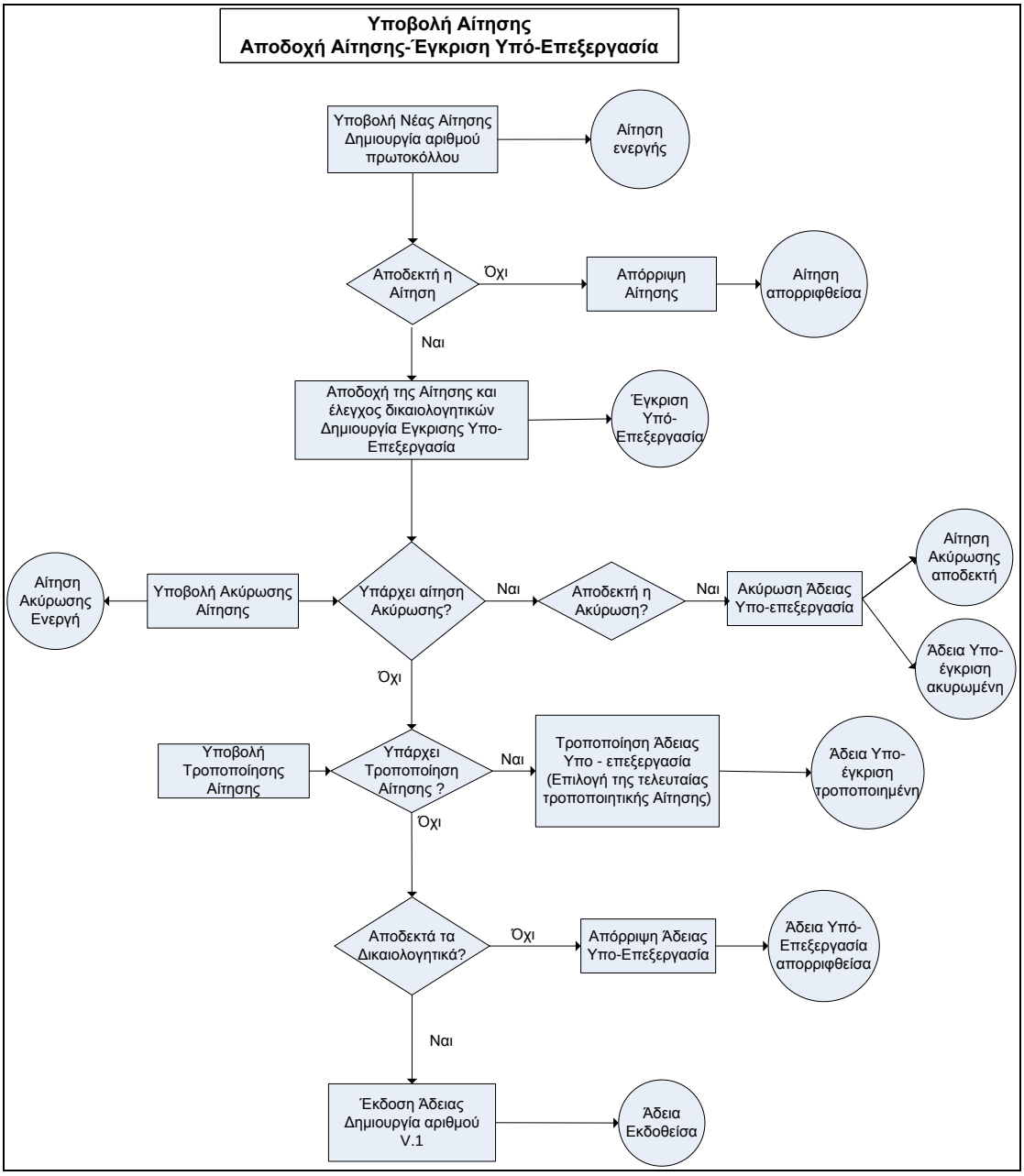
⇒ Επιβεβαίωση Απόρριψης

Η αίτηση προωθείται σε κατάσταση «Κλεισμένη».

4.3.1.5 «Αίτηση Κλεισμένη»

Δεν υπάρχουν ενέργειες για τις αιτήσεις που βρίσκονται στην κατάσταση αυτή και λογικά οι αιτήσεις αυτές δεν πρέπει να φαίνονται στην εφαρμογή πέρα από το ιστορικό των αιτήσεων.

Ακολουθεί διαγραμματικά η ροή των διαδικασιών κατά την υποβολή μιας Αίτησης (Εικόνα 4-3).



Εικόνα 4-3. Ροή Διαδικασιών κατά την υποβολή μιας Αίτησης.

4.3.2 Άδεια-Έγκριση

4.3.2.1 «Σε Επεξεργασία»

⇒ Αποθήκευση

Ενημερώνεται το περιεχόμενο της εγγραφής χωρίς να αλλάζει η κατάσταση της άδειας. Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

⇒ Απόρριψη

Η άδεια προωθείται σε κατάσταση « Απορριφθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

⇒ Έκδοση Άδειας

Κατά την έκδοση της άδειας πραγματοποιούνται με την σειρά τα παρακάτω:

- Ελέγχεται η ημερομηνία ενεργοποίησης τροποποίησης και αν αυτή είναι ίση με την σημερινή τότε η άδεια προωθείται σε κατάσταση «Εκδοθείσα» διαφορετικά προωθείται σε κατάσταση «Σε Αναμονή».
- Αν η άδεια δεν έχει αριθμό πρωτοκόλλου (δηλαδή προέρχεται από νέα αίτηση) τότε δημιουργείται ένας νέος αριθμός πρωτοκόλλου.
- Αν η άδεια έχει αριθμό πρωτοκόλλου (προέρχεται δηλαδή από αίτηση τροποποίησης άδειας) γίνεται ανάκτηση της άδειας που τροποποιείται. Αν η άδεια που τροποποιείται είναι σε κατάσταση:
    i. «Σε Αναμονή» τότε αντιγράφεται ο αριθμός έκδοσης της άδειας. Στην περίπτωση που η νέα άδεια που θα καταχωρηθεί είναι σε κατάσταση «Εκδοθείσα» ελέγχεται αν αυτή πρέπει να αντικαταστήσει κάποια άδεια που βρίσκεται στην ίδια κατάσταση.
    ii. «Εκδοθείσα» ο αριθμός έκδοσης αυξάνεται κατά ένα και η νέα άδεια αντικαθιστά την παλιά μόνο στην περίπτωση που η νέα άδεια θα καταχωρηθεί σε κατάσταση «Εκδοθείσα».

Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

4.3.2.2 «Απορριφθείσα»

⇒ Ακύρωση Απόρριψης

Ακολουθείται ξανά όλη η διαδικασία της Έγκρισης.

⇒ Επιβεβαίωση Απόρριψης

Η άδεια προωθείται σε κατάσταση «Κλεισμένη».

4.3.2.3 «Εκδοθείσα»

⇒ Τροποποίηση

Ενημερώνεται το περιεχόμενο της εγγραφής χωρίς να αλλάζει η κατάσταση της άδειας. Δεν μπορεί να τροποποιηθεί η ημερομηνία ενεργοποίησης της άδειας ούτε να καθοριστεί

ημερομηνία ενεργοποίησης τροποποίησης. Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

⇒ Αναστολή

Η άδεια προωθείται σε κατάσταση «Σε Αναστολή». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

⇒ Ανάκληση

Η άδεια προωθείται σε κατάσταση «Ανακληθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία» ή αν υπάρχει άδεια του ίδιου τύπου και συναλλασσόμενου σε κατάσταση «Σε Αναμονή».

⇒ Ακύρωση

Η άδεια προωθείται σε κατάσταση «Ακυρωθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία» ή αν υπάρχει άδεια του ίδιου τύπου και συναλλασσόμενου σε κατάσταση «Σε Αναμονή».

4.3.2.4 «Σε Αναμονή»

⇒ Τροποποίηση

Ενημερώνεται το περιεχόμενο της εγγραφής χωρίς να αλλάζει η κατάσταση της άδειας. Δεν μπορεί να τροποποιηθεί η ημερομηνία ενεργοποίησης της άδειας ούτε να καθοριστεί ημερομηνία ενεργοποίησης τροποποίησης. Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

⇒ Αναστολή

Η άδεια προωθείται σε κατάσταση «Κλεισμένη». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

4.3.2.5 «Σε Αναστολή»

⇒ Ακύρωση

Η άδεια προωθείται σε κατάσταση «Εκδοθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

4.3.2.6 «Ανακληθείσα»

⇒ Ακύρωση

Η άδεια προωθείται σε κατάσταση «Εκδοθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

4.3.2.7 «Ακυρωθείσα»

⇒ Ακύρωση

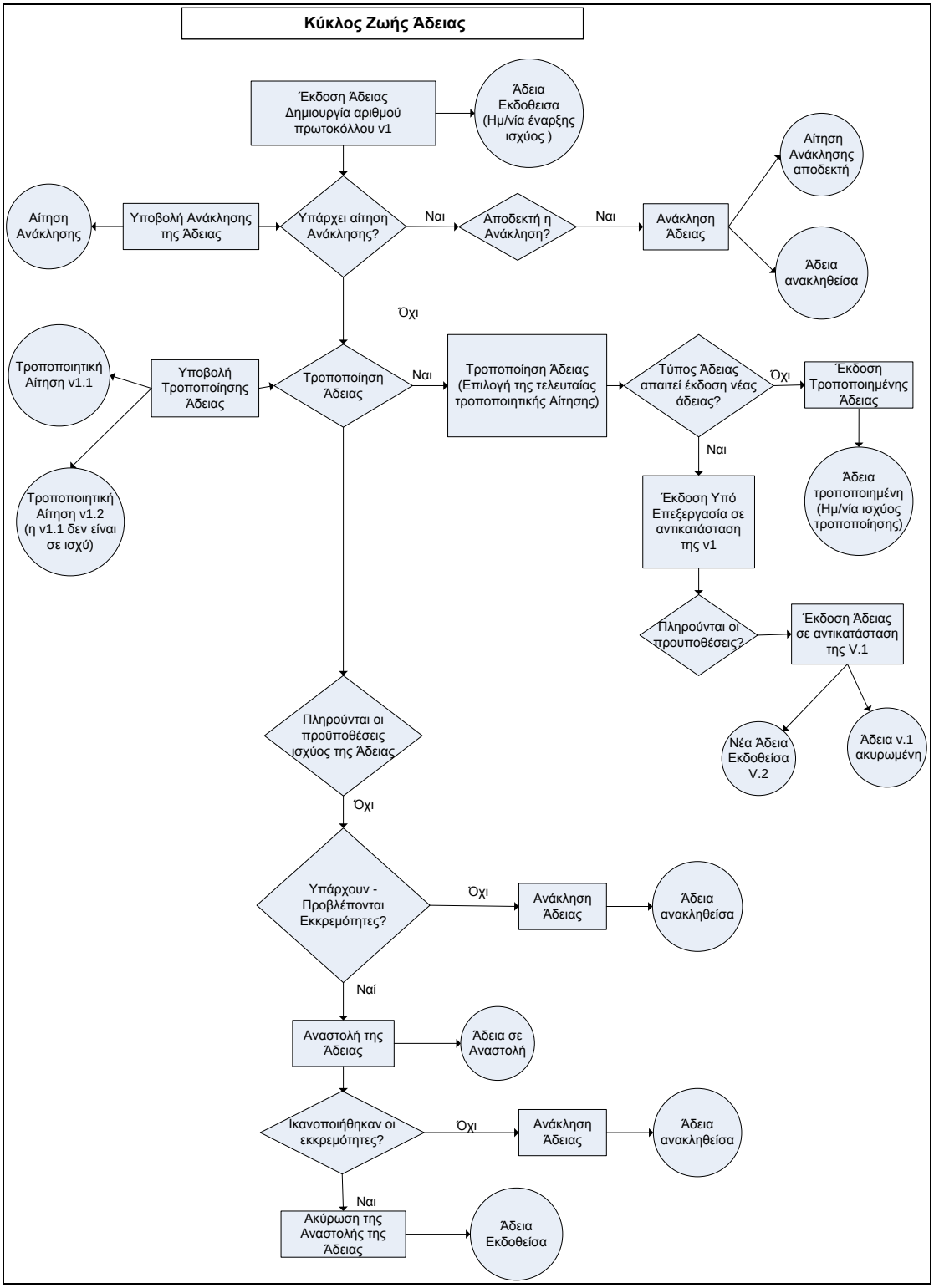
Η άδεια προωθείται σε κατάσταση «Εκδοθείσα». Η ενέργεια αυτή δεν μπορεί να πραγματοποιηθεί αν υπάρχει αίτηση τροποποίησης ή ακύρωσης του ίδιου τύπου και συναλλασσόμενου που βρίσκεται σε κατάσταση «Αποδεκτή» ή «Σε Επεξεργασία».

4.3.2.8 «Κλεισμένη»

Δεν υπάρχουν ενέργειες για τις άδειες που βρίσκονται στην κατάσταση αυτή. Στην κατάσταση αυτή έρχεται μια άδεια

i. Όταν γίνεται αποδεκτή μια νέα αίτηση για έκδοση νέας άδειας του ίδιου τύπου και συναλλασσόμενου
ii. Όταν γίνεται ανάκληση μιας άδειας που είναι σε κατάσταση «Αναμονής»

Ακολουθεί διαγραμματικά η ροή των διαδικασιών κατά την υποβολή μιας Άδειας (Εικόνα 4-4).



Εικόνα 4-4. Κύκλος Ζωής της Άδειας.

4.3.3 Έλεγχοι του συστήματος κατά την υποβολή – αποδοχή μιας αίτησης

Στη συνέχεια περιγράφονται κάποιοι επιπλέον έλεγχοι που εκτελούνται κατά την:

⇒ υποβολή – αποθήκευση της αίτησης

⇒ αποδοχή αίτησης

ανάλογα με το είδος της αίτησης και τη προηγούμενη κατάσταση (status) της αίτησης.

4.3.3.1 Έλεγχος υποβολής – αποθήκευσης αίτησης

Οι έλεγχοι συνοψίζονται παρακάτω ανάλογα με το είδος της αίτησης που έχει επιλεχθεί από τον χρήστη:

⇒ **Νέα Αίτηση**: Για να γίνει δεκτή η υποβολή θα πρέπει ο συγκεκριμένος συναλλασσόμενος να μην έχει καμία αίτηση του ίδιου τύπου στις παρακάτω καταστάσεις:

i. «Προσωρινά Αποθηκευμένη»
i. «Υποβληθείσα»
ii. «Σε Επεξεργασία»

Επίσης θα πρέπει ο συναλλασσόμενος να μην έχει και καμία άδεια του ίδιου τύπου στις παρακάτω καταστάσεις:

i. «Εκδοθείσα»
ii. «Σε Αναμονή»
iii. «Σε Αναστολή»

⇒ **Αίτηση Τροποποίησης/Ακύρωσης Αίτησης**: Για να γίνει δεκτή η υποβολή θα πρέπει η αίτηση που τροποποιείται ή ακυρώνεται να είναι το ίδιου τύπου με την αίτηση που καταθέτεται να ανήκει στον ίδιο συναλλασσόμενο και επιπλέον να βρίσκεται σε μια από τις δυο παρακάτω καταστάσεις

i. «Υποβληθείσα»
ii. «Σε Επεξεργασία»

Επίσης δεν επιτρέπεται να γίνει αίτηση για τη τροποποίηση ή ακύρωση μιας Αίτησης Ακύρωσης

⇒ **Αίτηση Τροποποίησης/Ακύρωσης Άδειας**: Για να γίνει δεκτή η αίτηση θα πρέπει η άδεια που τροποποιείται ή ακυρώνεται να είναι του ίδιου τύπου με την αίτηση που κατατίθεται, να ανήκει στον ίδιο συναλλασσόμενο, να μην υπάρχει άλλη αίτηση που αφορά την συγκεκριμένη άδεια και να βρίσκεται σε μια από τις παρακάτω καταστάσεις:

i. «Υποβληθείσα»
ii. «Σε Επεξεργασία»

Επίσης δεν πρέπει να υπάρχει άλλη άδεια που είναι σε κατάσταση «Σε Αναμονή» και πρόκειται να αντικαταστήσει την άδεια που επιχειρείται να τροποποιηθεί ή ακυρωθεί. Τέλος η άδεια που επιχειρείται να τροποποιηθεί ή ακυρωθεί θα πρέπει να είναι σε μια από τις παρακάτω καταστάσεις:

i. «Εκδοθείσα»
ii. «Σε Αναμονή»
iii. «Σε Αναστολή»

4.3.3.2 Έλεγχος αποδοχής αίτησης

Οι έλεγχοι αυτοί συνοψίζονται παρακάτω ανάλογα με το είδος της αίτησης που έχει επιλεχθεί από τον χρήστη:

⇒ **Νέα Αίτηση:** Όλες οι άδειες του ίδιου τύπου και συναλλασσόμενου, που βρίσκονται σε κατάσταση ακυρωμένη ή ανακληθείσα προωθούνται στην κατάσταση «Κλεισμένη». Στην συνέχεια δημιουργείται μια νέα εγγραφή σε κατάσταση «Σε Επεξεργασία».

⇒ **Τροποποίηση Αίτησης:** Η έγκριση σε κατάσταση «Σε Επεξεργασία» που είχε δημιουργηθεί για την αίτηση που τροποποιείται προωθείται στην κατάσταση «Αίτηση Κλεισμένη» και δημιουργείται μια νέα εγγραφή σε κατάσταση «Σε Επεξεργασία» για την τροποποιημένη αίτηση που γίνεται αποδεκτή.

⇒ **Τροποποίηση Άδειας:** Δημιουργείται μια νέα εγγραφή σε κατάσταση «Σε Επεξεργασία».

⇒ **Ακύρωση Αίτησης:** Η εγγραφή σε κατάσταση «Σε Επεξεργασία» που είχε δημιουργηθεί για την αίτηση που τροποποιείται προωθείται στην κατάσταση «Αίτηση Κλεισμένη».

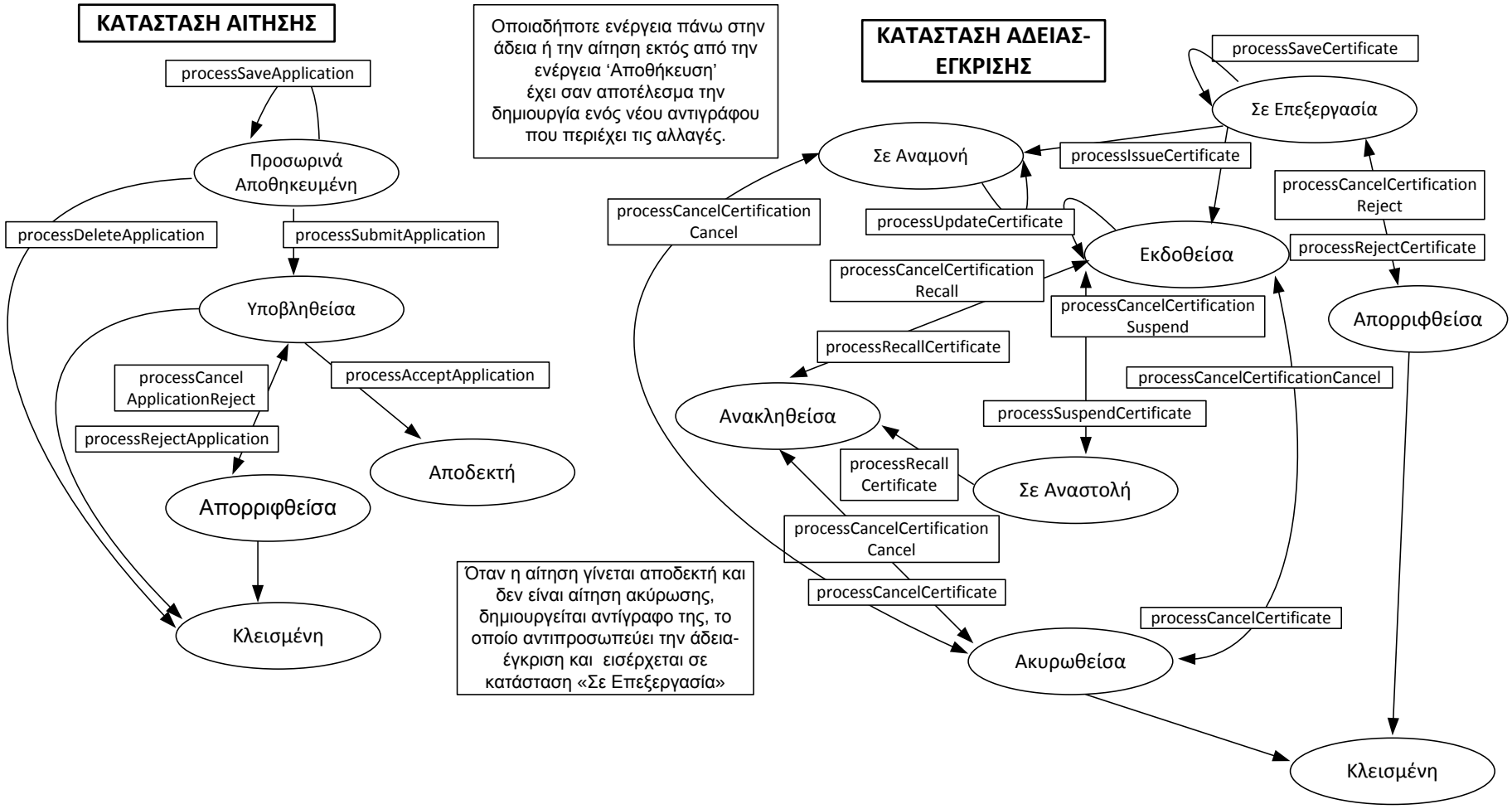
⇒ **Ακύρωση Άδειας:** Η εγγραφή που αντιστοιχεί στην τρέχουσα κατάσταση της άδειας προωθείται

i. Σε κατάσταση «Ανακληθείσα» αν η αρχική της κατάσταση είναι «Εκδοθείσα» ή «Σε Αναστολή»

ii. Σε κατάσταση «Άδεια Κλεισμένη» αν η αρχική της κατάσταση είναι «Σε Αναμονή».

4.4 Κύκλος Ζωής Της Αίτησης – Άδειας

Μια πιο λεπτομερής ροή των διαδικασιών της αίτησης και έγκρισης της άδειας ανάλογα με τη κατάσταση που βρίσκεται και τις ενέργειες που πραγματοποιούνται παρουσιάζεται στην εικόνα που ακολουθεί (Εικόνα 4-5).



Εικόνα 4-5. Σχηματική απεικόνιση των καταστάσεων αιτήσεων και αδειών.

5

ΤΕΧΝΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ JAVA 2 ENTERPRISE EDITION (J2EE)

5.1 Η ανάγκη για τεχνολογίες ανάπτυξης επιχειρησιακών εφαρμογών ανεξάρτητες από δεσμεύσεις

Η J2EE (Java 2 Enterprise Edition) είναι μια πλατφόρμα για την εκτέλεση εφαρμογών εξυπηρετητή (server side) που είναι γραμμένες σε γλώσσα Java. Πριν την πλατφόρμα J2EE, η υλοποίηση των εφαρμογών αυτών γινόταν με χρήση των ειδικών προγραμματιστικών διεπαφών API (Application Programming Interface) των κατασκευαστών λογισμικού. Ό κάθε κατασκευαστής είχε τα δικά του μοναδικά API και εφάρμοζε τις δικές του αρχιτεκτονικές κατά την ανάπτυξη εφαρμογών εξυπηρετητή με γλώσσα Java. Αποτέλεσμα της παραπάνω κατάστασης ήταν η απαίτηση, για γνώση μεγάλου όγκου πληροφορίας και ικανότητας προγραμματισμού για κάθε API από τους προγραμματιστές και αρχιτέκτονες, και τελικά υψηλότερο κόστος ανάπτυξης εφαρμογών. Επακόλουθο, ήταν η διαίρεση της κοινότητας και η απομόνωση όσων έκαναν ανάπτυξη εφαρμογών σε γλώσσα Java. Η περίοδος εκείνη χαρακτηρίστηκε ως αρνητική, αφού δεν προώθησε το κλίμα για την ανάπτυξη εφαρμογών και έκανε πολύ δύσκολο την υλοποίηση σοβαρών επιχειρησιακών (enterprise) εφαρμογών με γλώσσα Java.

Τη λύση στην παραπάνω κατάσταση έφερε η εισαγωγή της αρχιτεκτονικής J2EE, που δεν είναι τίποτα άλλο από μια πλατφόρμα ανάπτυξης Java κατανεμημένων επιχειρησιακών εφαρμογών, η οποία αναπτύχθηκε από τη Sun Microsystems. Η πλατφόρμα αυτή έχει υιοθετηθεί σήμερα, από τους περισσότερους κατασκευαστές λογισμικού αφού άλλωστε ήταν προϊόν κοινής τους συμφωνίας και προτυποποίησης των ίδιων τους των API.

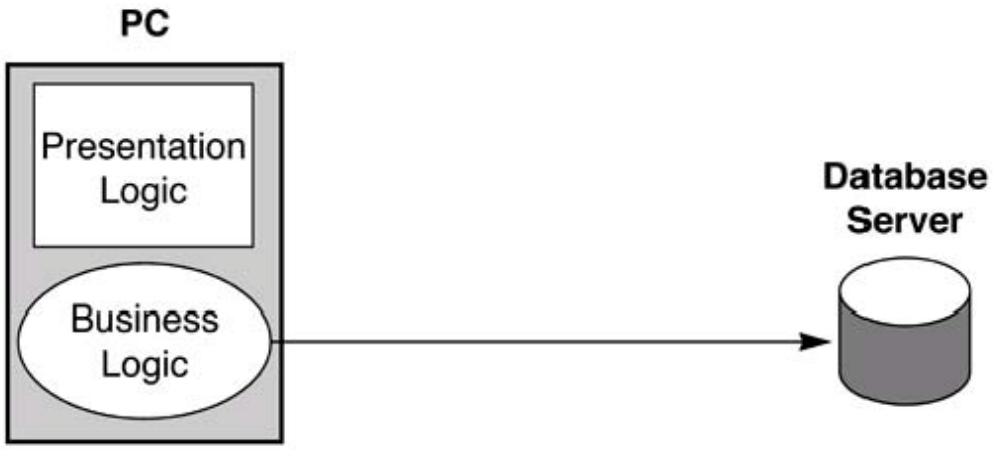
5.2 Γιατί οι εφαρμογές J2EE είναι επιθυμητές;

5.2.1 Η αρχιτεκτονική πελάτη - εξυπηρετητή (client - server)

Στις αρχές του 1990, Πληροφοριακά Συστήματα (Information Systems) συχνά χρησιμοποιούσαν το μοντέλο αρχιτεκτονικής πελάτη - εξυπηρετητή (client - server). Η διεπαφή (interface) του χρήστη της εφαρμογής συνήθως έτρεχε σε έναν επιτραπέζιο υπολογιστή που βρίσκονταν στην πλευρά του πελάτη. Τα επιχειρησιακά δεδομένα που προσπελάζονταν από τον πελάτη - εξυπηρετητή βρίσκονταν στη βάση δεδομένων και προσφέρονταν μέσω του εξυπηρετητή.

Η παραπάνω προσέγγιση αρχικά υποσχέθηκε μεγάλη ανάπτυξη και ευελιξία. Η μετέπειτα εμπειρία έδειξε μολαταύτα, ότι η υλοποίηση και η συντήρηση ενός ευέλικτου κατανεμημένου συστήματος είναι δύσκολο να γίνει χρησιμοποιώντας το μοντέλο πελάτη - εξυπηρετητή. Για παράδειγμα η επιχειρησιακή λογική (business logic), δηλαδή το τμήμα που έχει να κάνει με τον κύριο μηχανισμό επεξεργασίας της εφαρμογής, ήταν τότε στην πλευρά του πελάτη. Κάθε φορά που το κομμάτι της επιχειρησιακής λογικής (business logic) της εφαρμογής χρειαζόταν τροποποίηση, η αναθεωρημένη έκδοση της εφαρμογής έπρεπε να εγκατασταθεί σε κάθε μηχανή πελάτη της επιχείρησης. Η συντήρηση εκείνα τα χρόνια ήταν ένας εφιάλτης. Επίσης, αυτές οι εφαρμογές έπρεπε να διαχειρίζονται τις διάφορες δοσοληψίες (transactions) με την μεγαλύτερη δυνατή ασφάλεια, και να επεξεργάζονται τα δεδομένα με τρόπο αποδοτικό ενώ ταυτόχρονα να παρέχουν και ένα ελκυστικό, εύκολα κατανοητό, περιβάλλον αλληλεπίδρασης στους χρήστες.

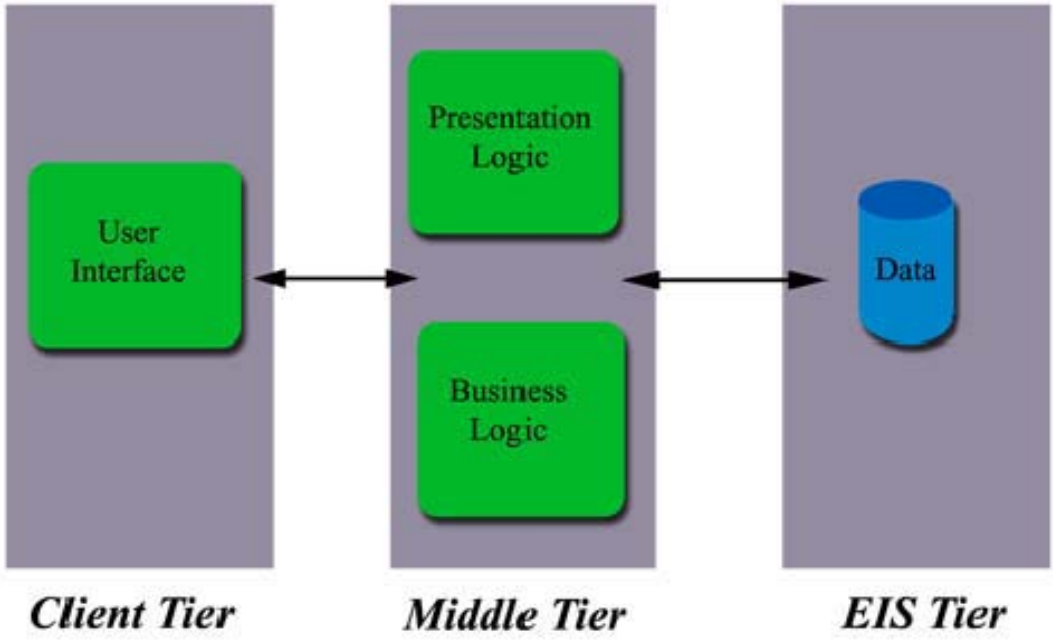
Λίγοι είναι οι προγραμματιστές οι οποίοι έχουν ταλέντο σε όλα αυτά τα αντικείμενα. Ενώ το πρότυπο πελάτη – εξυπηρετητή (Εικόνα 5-1) είναι επαρκές για κάποια περιβάλλοντα γίνεται φανερό ότι δεν είναι δυνατόν να ικανοποιήσει τις σύγχρονες απαιτήσεις στο πεδίο των επιχειρησιακών εφαρμογών.



Εικόνα 5-1. Αρχιτεκτονική εφαρμογής δύο στρωμάτων (client – server).

5.2.2 Το πολυστρωματικό πρότυπο αρχιτεκτονικής των εφαρμογών

Με δεδομένους τους παραπάνω περιορισμούς, αναζητήθηκε μια νέα προσέγγιση. Το αποτέλεσμα αυτής της αναζήτησης είναι το πολυστρωματικό (multi-tier) πρότυπο αρχιτεκτονικής, το οποίο φαίνεται στο παρακάτω δομικό διάγραμμα (Εικόνα 5-2).

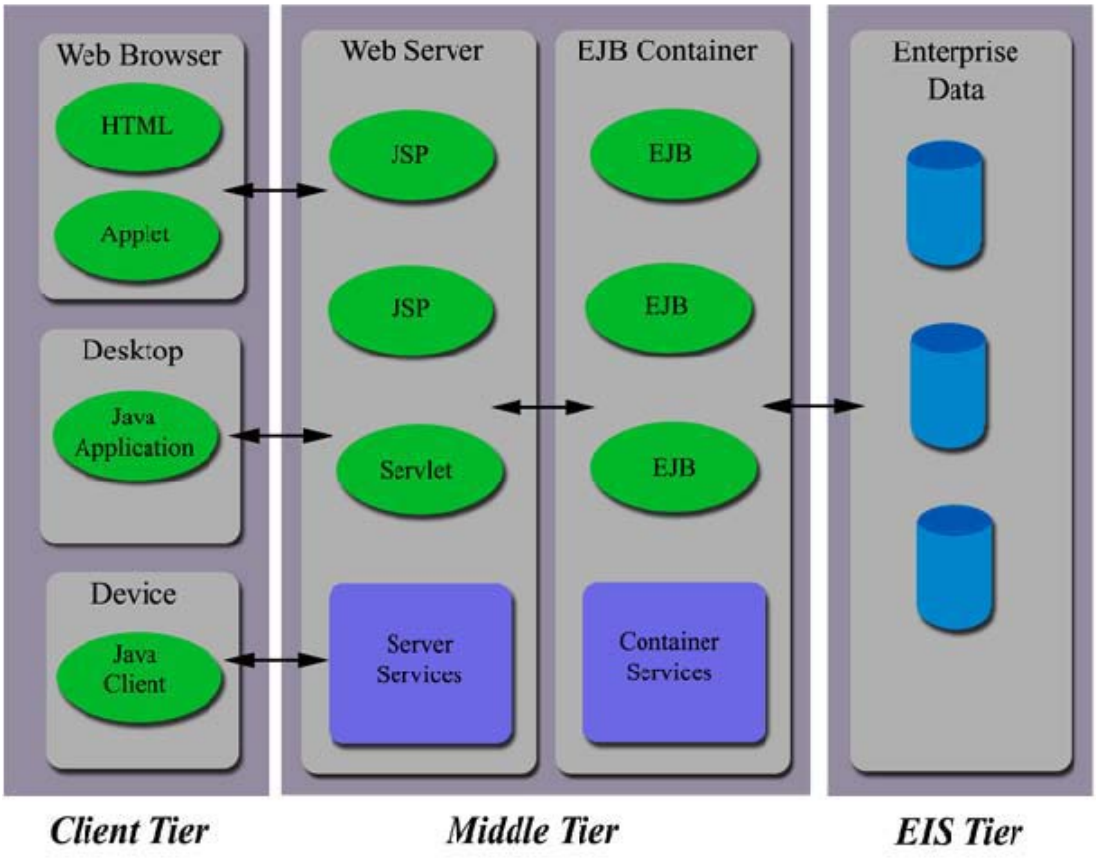


Εικόνα 5-2. Αρχιτεκτονική εφαρμογής πολλών στρωμάτων - πολυστρωματική προδιαγραφή.

Στην πολυστρωματική (multi-tier) προδιαγραφή η παρουσία του χρήστη στο μεσαίο στρώμα βρίσκεται υπό την μορφή διεπαφής χρήστη. Το κομμάτι της επιχειρησιακής λογικής (business logic) της εφαρμογής βρίσκεται επίσης στο μεσαίο στρώμα και επικοινωνεί με το χρήστη, μέσω της διεπαφής χρήστη. Όταν χρειαστεί μια αλλαγή για την εφαρμογή, θα ενημερωθεί ένα

μόνο σημείο, αυτό της διεπαφής χρήστη στο μεσαίο στρώμα, αντί να ενημερωθεί κάθε μηχανή πελάτη όπως γινόταν παλαιότερα.

Το παρακάτω αναλυτικό δομικό διάγραμμα (Εικόνα 5-3) δείχνει τα διάφορα συστατικά λογισμικού τα οποία εκτελούνται στα επιμέρους στρώματα της εφαρμογής στην πλατφόρμα J2EE.



Εικόνα 5-3. Η πολυστρωματική προδιαγραφή με παρουσίαση των συστατικών λογισμικού (components) που εκτελούνται σε κάθε στρώμα.

5.3 Συστατικά λογισμικού (components)

Ο όρος συστατικό λογισμικού (component) έχει πάρει πολλά νοήματα στο χώρο της ανάπτυξης λογισμικού, και έτσι είναι αναγκαίο να δοθεί καταρχάς ένας ορισμός. Ένα συστατικό λογισμικού είναι μια αυτοτελής (self-contained) επαναχρησιμοποιήσιμη μονάδα λογισμικού η οποία ολοκληρώνεται μέσα σε μια εφαρμογή. Οι πελάτες αλληλεπιδρούν με αυτές τις λογισμικές μονάδες μέσω καλά ορισμένων διεπαφών. Στην γλώσσα Java το πιο απλό συστατικό λογισμικού είναι το Java Bean.

Στον επιχειρησιακό χώρο, τα συστατικά λογισμικού εστιάζονται περισσότερο στην υλοποίηση επιχειρησιακών (business) υπηρεσιών και λιγότερο για υπηρεσίες ιστού -

συστατικά λογισμικού ιστού. Η βασική προδιαγραφή ενός συστατικού λογισμικού για την πλατφόρμα J2EE είναι η προδιαγραφή των Enterprise Java Beans(EJBs), η οποία ορίζει τρόπους για τη συσκευασία (packaging) και εγκατάσταση μιας επιχειρησιακής εφαρμογής καθώς και διασύνδεσης με αυτόνομες υπηρεσίες. Η προδιαγραφή EJB ορίζει τις συμβάσεις που απαιτούνται για την επικοινωνία των επιμέρους συστατικών λογισμικού μέσω προτυποποιημένων διεπαφών στη γλώσσα Java.

Αρκετές εφαρμογές παρακάμπτουν τα EJBs. Σε αυτές τις εφαρμογές, τα συστατικά λογισμικού του μεσαίου στρώματος (JSP/servlets) προσπελάζουν απευθείας τη βάση δεδομένων. Η χρήση των συστατικών λογισμικού απαιτεί την οργάνωση της εφαρμογής σε στρώματα, με τις επιχειρησιακές υπηρεσίες (services) να βρίσκονται μέσα στο επιχειρησιακό (business) στρώμα ακολουθώντας το πρότυπο συστατικών EJB.

Ιστορικά, μια από τις αντιρρήσεις για την υιοθέτηση των συστατικών λογισμικού EJB από την Java EE ήταν η πολυπλοκότητα στην υλοποίηση τους. Το πρόβλημα αυτό έχει λυθεί κατά ένα μεγάλο μέρος, και έτσι αυτό που έχει μείνει δεν είναι τίποτα άλλο από τα πλεονεκτήματα που προσφέρουν και το πλήθος των καλά ορισμένων υπηρεσιών σε μια εφαρμογή:

- Χαλαρή σύζευξη: Η χρήση συστατικών λογισμικού για την υλοποίηση υπηρεσιών βοηθά στην χαλαρή σύζευξή μεταξύ των στρωμάτων μιας εφαρμογής. Η υλοποίηση ενός συστατικού λογισμικού μπορεί να αλλάζει χωρίς να επηρεάζει τους πελάτες ή άλλα συστατικά λογισμικού από τα οποία εξαρτάται.
- Διαχείριση εξαρτήσεων: Οι εξαρτήσεις ενός συστατικού λογισμικού δηλώνονται με τη μορφή μεταδεδομένων (metadata) και μετά αυτομάτως διευθετούνται από τον υποδοχέα (container), δηλαδή από το περιβάλλον του εξυπηρετητή εφαρμογών (application server) όπου εγκαθίσταται και εκτελείται το συστατικό λογισμικού.
- Διαχείριση κύκλου ζωής: Ο κύκλος ζωής των συστατικών λογισμικού είναι καλά προσδιορισμένος και είναι διαχειριζόμενος από τον εξυπηρετητή εφαρμογών. Οι υλοποιήσεις των συστατικών λογισμικού μπορούν να συμμετέχουν στις λειτουργίες κύκλου ζωής για την απόκτηση και απελευθέρωση πόρων ή για την εκτέλεση άλλων αρχικοποιήσεων και τερματισμών.
- Ορισμός υπηρεσιών υποδοχέα (container): Οι επιχειρησιακές (business) μέθοδοι των συστατικών λογισμικού μπορούν να διακόπτονται από τον εξυπηρετητή εφαρμογών για την παραχώρηση προτεραιότητας σε αυτοματοποιημένες υπηρεσίες της εφαρμογής, όπως: συγχρονισμού, διαχείρισης δοσοληψιών, ασφάλειας και διαχείριση απομακρυσμένων κλήσεων (remoting).
- Φορητότητα: Τα συστατικά λογισμικού που ακλουθούν τις προδιαγραφές Java EE και εγκαθίστανται σε κατάλληλους εξυπηρετητές , οι οποίοι επίσης ακολουθούν

αυτές τις προδιαγραφές, μπορούν πολύ εύκολα να μεταφερθούν από έναν εξυπηρετητή σε έναν άλλο, με την προϋπόθεση ότι ο τελευταίος ακολουθεί επίσης τις παραπάνω προδιαγραφές.

- Επεκτασιμότητα και αξιοπιστία: Οι εξυπηρετητές εφαρμογών σχεδιάζονται ώστε να διασφαλίζεται ότι διαχειρίζονται τα συστατικά λογισμικού αποδοτικά και με προοπτική πάντα την επεκτασιμότητα τους. Οι λειτουργίες που υλοποιούνται χρησιμοποιώντας συστατικα λογισμικού, είναι πάντα εξαρτημένες από τον εκάστοτε τύπο συστατικών καθώς και από και τις ρυθμίσεις (configurations) του κάθε εξυπηρετητή. Έτσι μπορεί να επαναλαμβάνουν εσφαλμένες μεθόδους ή να αποτυγχάνουν να λειτουργήσουν όταν εκτελεστούν σε κάποιον άλλο εξυπηρετητή.

5.4 Πλεονεκτήματα του μοντέλου πολλαπλής στρωμάτωσης

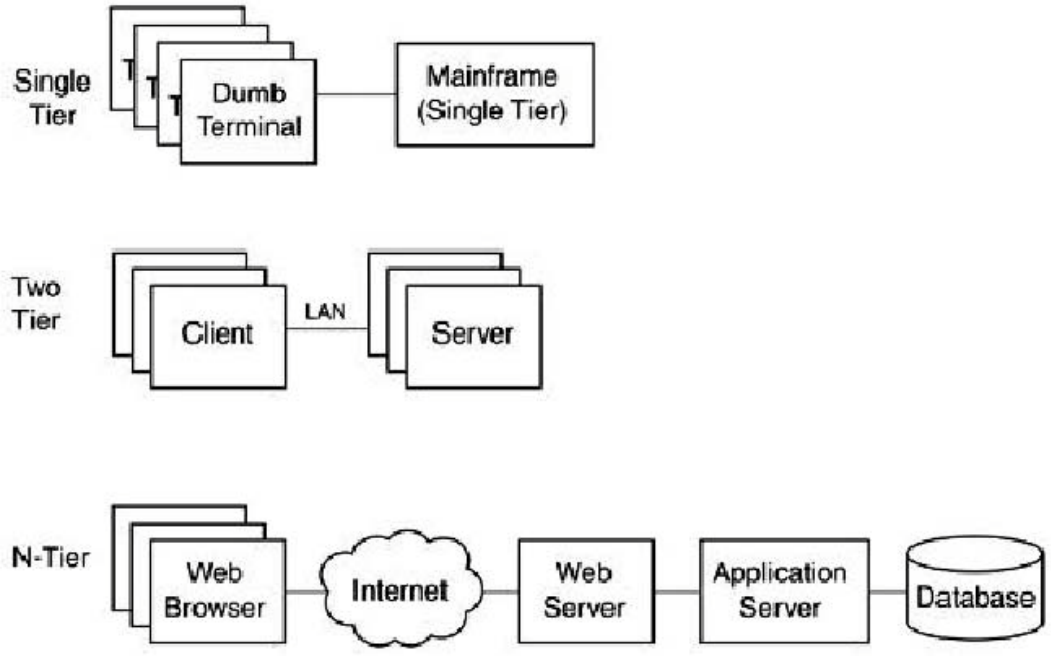
Η πολυστρωματική προσέγγιση που έχει υιοθετηθεί από την J2EE πλατφόρμα έχει αρκετά πλεονεκτήματα:

- Μειώνει την πολυπλοκότητα της κατανεμημένης ανάπτυξης κάνοντας χρήση μιας απλοποιημένης αρχιτεκτονικής και μοιράζοντας τον φόρτο εργασίας μεταξύ διαφόρων τμημάτων.
- Το κομμάτι της επιχειρησιακής λογικής της εφαρμογής εκτελείται στο μεσαίο στρώμα, μέσα στον Enterprise Java Bean (EJB) υποδοχέα και/ή πάνω στον εξυπηρετητή ιστού. Οι συγκεκριμένοι υποδοχείς και εξυπηρετητές μπορούν να διαχειριστούν δύσκολες εργασίες αντί για τους προγραμματιστές. Για παράδειγμα, ένας EJB υποδοχέας μπορεί να διαχειριστεί, τις δοσοληψίες (transactions) μεταξύ των διαφόρων συστατικών λογισμικού, τη διαχείριση – άντληση στιγμιοτύπων (instance pooling), και τη διαχείριση των δεδομένων της βάσης (data persistence), χωρίς την απαίτηση από τον προγραμματιστή να ασχοληθεί με την συγγραφή κώδικα που να εκτελεί τις σύνθετες αυτές λειτουργίες. Ένας εξυπηρετητής ιστού μπορεί να δημιουργεί και να διαχειρίζεται τα στιγμιότυπα των κλάσεων των servlet και να χειρίζεται πολλαπλά νήματα εκτέλεσης και πολλά διαφορετικά κανάλια επικοινωνίας. Ο προγραμματιστής εφαρμογών, δε χρειάζεται να γράψει τον κώδικα για την πραγμάτωση αυτών των λειτουργιών, προσδιορίζει απλώς τη συγκεκριμένη συμπεριφορά τους κατά το χρόνο της εγκατάστασης των εφαρμογών στο εξυπηρετητή.
- Τα μέλη που απαρτίζουν την ομάδα ανάπτυξης έχουν συνήθως διαφορετικούς ρόλους. Ο καθένας είναι ειδικός σε ένα η περισσότερα πεδία. Για παράδειγμα, το περιεχόμενο μιας HTML σελίδας ή ενός εκμαγείου στιλ σελίδων (stylesheet) θα μπορούσε να είχε δημιουργηθεί από έναν σχεδιαστή γραφικών. Ένας έμπειρος

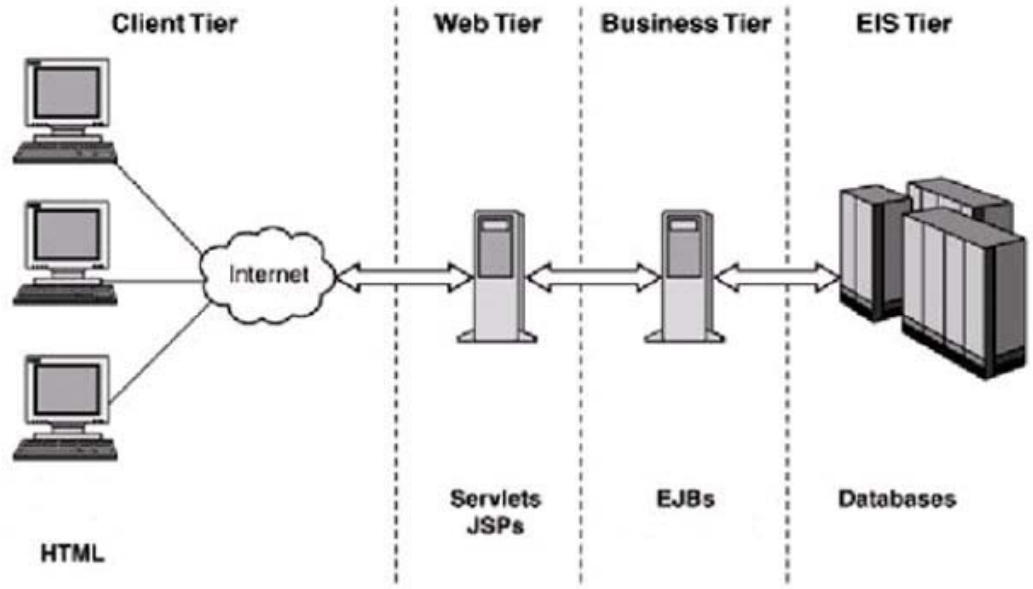
προγραμματιστής είναι υπεύθυνος για την υλοποίηση του επιχειρησιακού τμήματος της εφαρμογής το οποίο είναι ενσωματωμένο στα συστατικά λογισμικού Enterprise Java Bean. Κάποιος που αναπτύσσει το κομμάτι ιστού της εφαρμογής οφείλει να αναπτύξει τη διεπαφή χρήστη και γενικότερα τον τρόπο παρουσίασης συστατικών λογισμικού (presentation logic) χρησιμοποιώντας JSPs (Java Server Pages) και servlets. Εκείνος που έχει το ρόλο του κτισίματος της εφαρμογής παίρνει τα διάφορα συστατικά λογισμικού της εφαρμογής και τα τοποθετεί όλα μαζί, δημιουργώντας ένα EAR αρχείο και ένα αρχείο περιγραφέα εγκατάστασης το οποίο περιγράφει πως ή εφαρμογή είναι διατεταγμένη (deployed).

- Είναι επεκτάσιμη, επιτρέποντας την ανάπτυξη συστημάτων που συνδυάζουν διαφορετικές λειτουργίες οι οποίες μπορούν να μεταβάλλονται συχνά.
- Όταν οι απαιτήσεις του συστήματος αυξάνονται, το κομμάτι του λογισμικού που εμπεριέχει την επιχειρησιακή λογική της εφαρμογής μπορεί να ενημερώνεται εύκολα σε ένα σημείο του μεσαίου επιπέδου χωρίς να υπάρχει η ανάγκη της ενημέρωσης κάθε μηχανής πελάτη.

Τα παρακάτω δομικά διαγράμματα, (Εικόνα 5-4, 5-5) δείχνουν την εξέλιξη της αρχιτεκτονικής και τον τρόπο κατανομής των στρωμάτων των επιχειρησιακών εφαρμογών αντιστοίχως.



Εικόνα 5-4. Η εξέλιξη της αρχιτεκτονικής των επιχειρησιακών εφαρμογών.

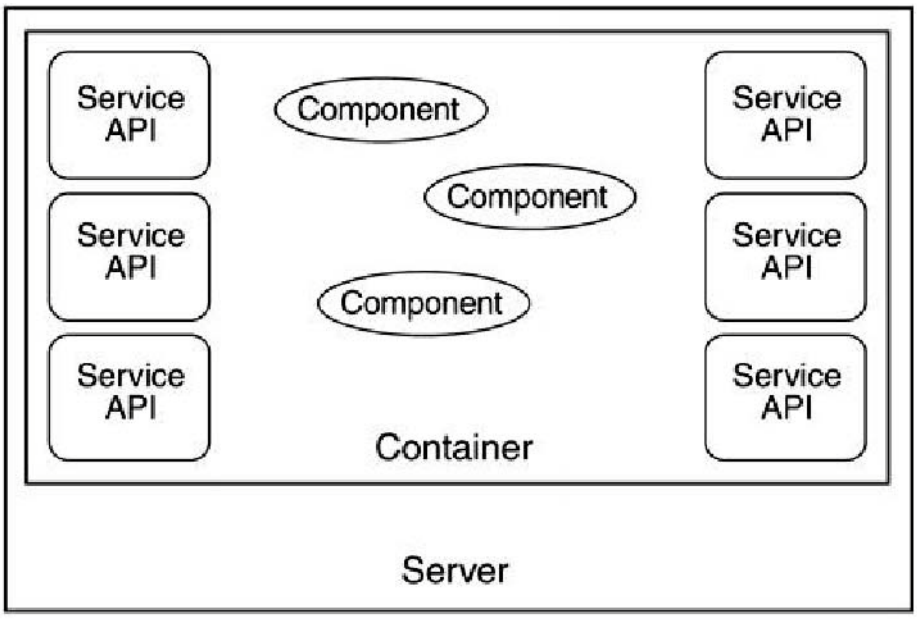


Εικόνα 5-5. Η κατανεμημένη αρχιτεκτονική, των επιχειρησιακών εφαρμογών.

5.5 Αρχιτεκτονική της πλατφόρμας J2EE

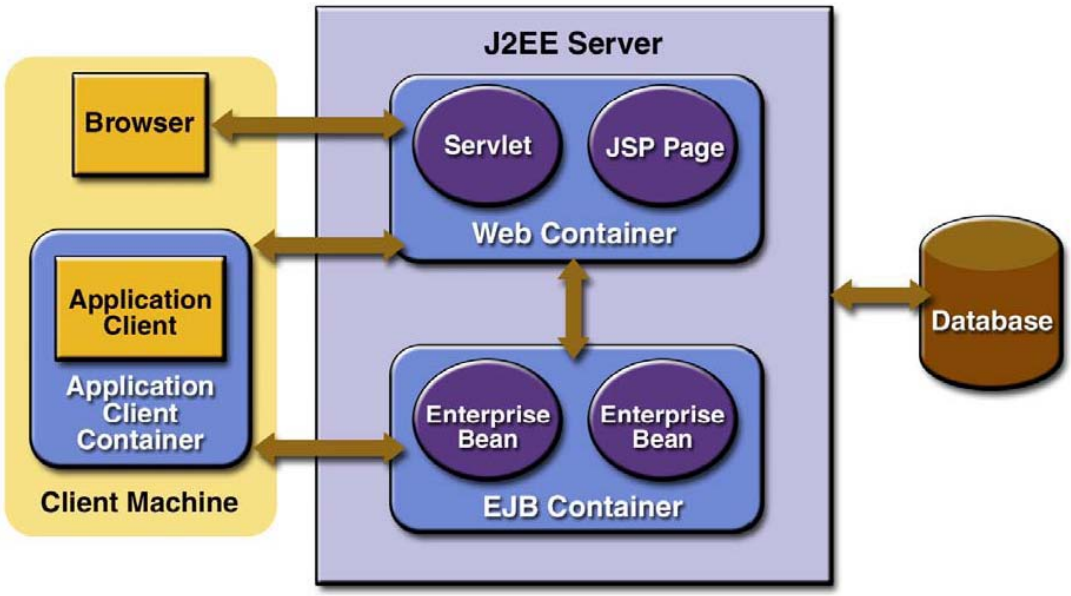
Μελετώντας την αρχιτεκτονική της πλατφόρμας J2EE σαν μια στοίβα στρωμάτων, όπως θα την έβλεπε ένας προγραμματιστής, στο κατώτερο στρώμα της στοίβας υπάρχει η υποδομή της Java 2 Standard Edition (J2SE).

Οι εξυπηρετητές εφαρμογών J2EE εκτελούν τις εφαρμογές στο επίπεδο της εικονικής μηχανής Java (JVM:Java Virtual Machine) και παρέχουν κατάλληλες διεπαφές στους προγραμματιστές των εφαρμογών (API). Δύο (κυρίως) τύποι εφαρμογών μπορούν να αναπτυχθούν στους εξυπηρετητές εφαρμογών J2EE:οι εφαρμογές ιστού και οι εφαρμογές EJB(Enterprise Java Bean). Αυτές οι εφαρμογές εγκαθίστανται και εκτελούνται μέσα σε κατάλληλους υποδοχείς (containers) (Εικόνα 5-6). Η προδιαγραφή J2EE ορίζει υποδοχείς για τη διαχείριση του κύκλου ζωής των συστατικών λογισμικού μιας εφαρμογής από την πλευρά του εξυπηρετητή.



Εικόνα 5-6. Γενική άποψη ενός υποδοχέα (container).

Υπάρχουν δύο τύποι υποδοχέων: οι υποδοχείς servlet και οι υποδοχείς EJB. Οι υποδοχείς servlet διαχειρίζονται τον κύκλο ζωής των εφαρμογών ιστού ενώ οι υποδοχείς EJB διαχειρίζονται τον κύκλο ζωής των EJB (Εικόνα 5-7).



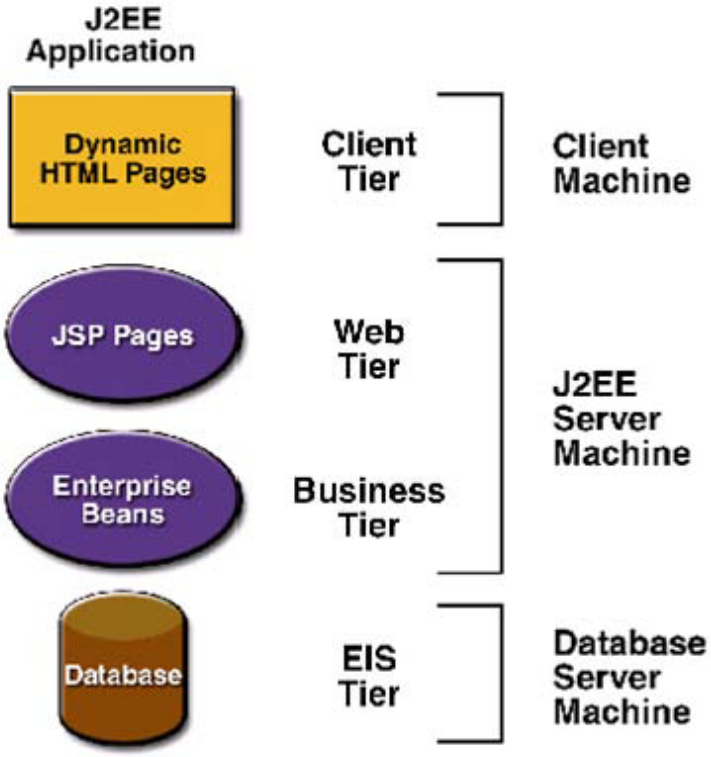
Εικόνα 5-7. Γενική άποψη των υποδοχέων ιστού (web) και EJB ενός εξυπηρετητή εφαρμογών J2EE.

5.5.1 Κατανεμημένες πολυστρωματικές εφαρμογές

Τα διάφορα στοιχεία λογισμικού που συνιστούν την εφαρμογή J2EE εγκαθίστανται σε διαφορετικές μηχανές, ανάλογα με το στρώμα που ανήκουν στην πολυστρωματική δομή

J2EE. Η Εικόνα 5-8 δείχνει μια πολυστρωματική εφαρμογή J2EE χωρισμένη σε στρώματα τα οποία περιγράφονται στην παρακάτω λίστα συστατικών λογισμικού J2EE:

- Στρώμα πελάτη (client tier) το οποίο εκτελείται στη μηχανή του πελάτη.
- Στρώμα ιστού (web tier) που εκτελείται στον εξυπηρετητή εφαρμογών J2EE.
- Λογισμικό επιχειρησιακού στρώματος (business tier) που εκτελείται στον εξυπηρετητή εφαρμογών J2EE.
- Στρώμα Συστήματος Επιχειρησιακών Πληροφοριών (EIS:Enterprise Information System tier) που εκτελείται στον εξυπηρετητή EIS.



Εικόνα 5-8. Πολυστρωματική Εφαρμογή.

5.6 Τεχνολογικά Χαρακτηριστικά

Η πλατφόρμα J2 EE, για τη μείωση του κόστους και την ταχύτερη σχεδίαση και ανάπτυξη, παρέχει μια προσέγγιση βασισμένη σε συστατικά λογισμικού, για τη σχεδίαση ,ανάπτυξη, δόμηση, και εγκατάσταση των επιχειρησιακών εφαρμογών. Η πλατφόρμα J2EE προσφέρει μια κατανεμημένη, πολυστρωματική προδιαγραφή εφαρμογών, επαναχρησιμοποιήσιμα συστατικά λογισμικού, ένα ενιαίο πρότυπο ασφάλειας, ευέλικτο έλεγχο δοσοληψιών, και υποστήριξη υπηρεσιών ιστού δια της ολοκληρωμένης ανταλλαγής δεδομένων πάνω σε γλώσσα XML(XML:EXtensible Markup Language) βασιζόμενη σε ανοικτές προτυποποιήσεις και πρωτόκολλα.

Η ανάπτυξή ενός επιχειρησιακού λογισμικού αποτελεί μια σύνθετη εργασία ή οποία χρειάζεται βαθιά γνώση σε αρκετά και διαφορετικά πεδία. Για παράδειγμα, μια τυπική επιχειρησιακή εφαρμογή απαιτεί ότι ο προγραμματιστής θα είναι εξοικειωμένος με αρκετές λειτουργίες όπως, ο τρόπος επικοινωνίας μεταξύ των διαδικασιών, λειτουργίες που αφορούν την ασφάλεια, διαχείρισης της βάσης δεδομένων μέσω ειδικών ερωτημάτων SQL και άλλα. Η πλατφόρμα J2EE παρέχει την υποστήριξη, με αρκετά διαφανή τρόπο ανάλογων υπηρεσιών έτσι ώστε οι προγραμματιστές να εστιάζουν κυρίως στην υλοποίηση του κώδικα του επιχειρησιακού μέρους της εφαρμογής.

Ίσως ένα από τα μεγαλύτερα πλεονεκτήματα της πλατφόρμας J2EE είναι η υποστήριξη των συστατικών λογισμικού. Οι εφαρμογές που βασίζονται σε συστατικά λογισμικού έχουν αρκετά πλεονεκτήματα σε σχέση με την συνηθισμένη ανάπτυξη:

- **Υψηλή παραγωγικότητα**: Λιγότεροι μπορούν να πετύχουν περισσότερα μέσω της επαναχρησιμοποίησης μιας εφαρμογής η οποία είχε πραγματοποιηθεί για παραπλήσιο σκοπό, μαζί με ήδη ξαναδουλεμένα στοιχεία λογισμικού αντί για μια υλοποίηση από την αρχή με το συνηθισμένο τρόπο.
- **Γρήγορή ανάπτυξη:** Συστατικά λογισμικού που ήδη υπάρχουν, μπορούν να συνδυαστούν με γρήγορο τρόπο και να δημιουργηθούν νέες εφαρμογές.
- **Υψηλή ποιότητα**: Αποφυγή ελέγχου συνολικά μιας εφαρμογής. Ο έλεγχος των εφαρμογών που βασίζεται σε συστατικά λογισμικού μπορεί να γίνει πιο ευέλικτος με τον έλεγχο όλης της εφαρμογής μέσω του ελέγχου των ξεχωριστών συστατικών λογισμικού.
- **Ευκολότερη συντήρηση**: Τα συστατικά λογισμικού αποτελούν αυτόνομες οντότητες και έτσι η συντήρηση τους έτσι είναι πιο εύκολη και λιγότερο δαπανηρή.

5.7 Εργαλεία J2EE

Για την κατανόηση των τεχνολογιών J2EE είναι απαραίτητη, καταρχάς η κατανόηση του ρόλου του υποδοχέα (container) στην αρχιτεκτονική J2EE. Όλες οι τρέχουσες τεχνολογίες που χρησιμοποιούνται στην προδιαγραφή J2EE στηρίζονται σε αυτή την ισχυρή έννοια.

Ένας υποδοχέας δεν είναι τίποτα άλλο από μια οντότητα λογισμικού η οποία εκτελείται εντός του εξυπηρετητή και είναι υπεύθυνη για τη διαχείριση ειδικών τύπων συστατικών λογισμικού. Ο υποδοχέας αποτελεί το περιβάλλον εκτέλεσης για τα συστατικά λογισμικού που αναπτύσσει ένας προγραμματιστής. Είναι προφανές ότι υποδοχείς σαν και αυτούς που υποστηρίζει η αρχιτεκτονική J2EE, είναι ικανοί να παρέχουν ανεξαρτησία μεταξύ των διαδικασιών της ανάπτυξης μιας εφαρμογής από την εγκατάσταση της, και να προσφέρουν τη δυνατότητα μεταφοράς δεδομένων, μεταξύ διαφόρων τύπων εξυπηρετητών μεσαίου στρώματος.

Ο υποδοχέας είναι επίσης υπεύθυνος για τη διαχείριση του κύκλου ζωής των συστατικών λογισμικού που είναι εγκαταστημένα μέσα σε αυτόν καθώς και για λειτουργίες όπως αυτές της διαχείρισης πόρων (p o ώnl g resou ce) καθώς και της επιβολής της ασφάλειας. Για παράδειγμα, ο διαχειριστής της εφαρμογής έχει τη δυνατότητα να περιορίσει την πρόσβαση σε μια συγκεκριμένη μέθοδο για μια μικρή ομάδα χρηστών. Ο υποδοχέας επιβάλλει τον παραπάνω περιορισμό μέσω της ανάσχεσης των κλήσεων μιας μεθόδου σε περίπτωση που η οντότητα που εκτελεί την κλήση δεν έχει τα κατάλληλα δικαιώματα.

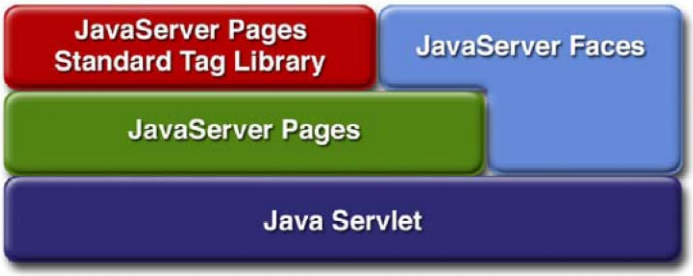
Εξαρτάται από τον τύπο του υποδοχέα εάν θα παρέχεται πρόσβαση σε κάποια ή σε όλα από τα APIs της πλατφόρμας J2EE. Όλα τα συστατικά λογισμικού J2EE εγκαθίστανται και εκτελούνται εντός ενός υποδοχέα κάποιου τύπου. Για παράδειγμα, τα EJBs εκτελούνται στον υποδοχέα EJB, και τα servlets εκτελούνται στον υποδοχέα ιστού. Συνολικά η πλατφόρμα έχει τέσσερις τύπους υποδοχέων:

- **Υποδοχέας εφαρμογών** (application container): Φιλοξενεί αυτόνομες εφαρμογές Java.
- **Υποδοχέας μικροεφαρμογών** (applet container): Παρέχει ένα περιβάλλον εκτέλεσης για μικροεφαρμογές (applets).
- **Υποδοχέας** ιστού (web container): Φιλοξενεί εφαρμογές ιστού, όπως servlets και Java Server Pages (JSP).
- **Επιχειρησιακός υποδοχέας** (enterprise container): Φιλοξενεί συστατικά λογισμικού EJB.

5.7.1 Servlets (Μικροϋπηρεσίες)

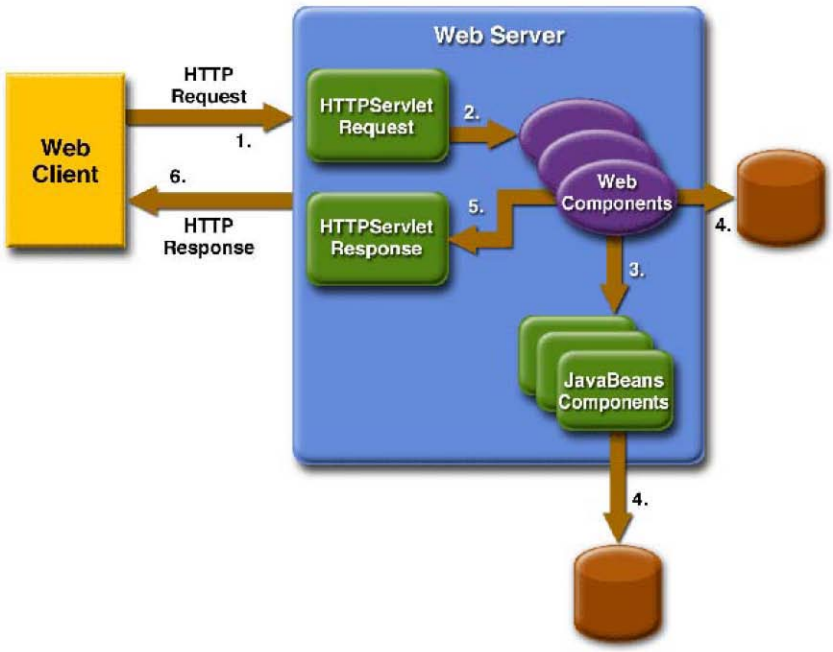
Τα servlets είναι συστατικά λογισμικού ιστού ικανά για τη δημιουργία δυναμικού περιεχομένου. Είναι από τα πιο συχνά χρησιμοποιούμενα συστατικά λογισμικού J2EE στον παγκόσμιο ιστό (World Wide Web) (Εικόνα 5-9). Παρέχουν ένα αποδοτικό μηχανισμό για τη συναλλαγή μεταξύ του πελάτη που βρίσκεται σε περιβάλλον ιστού και του επιχειρησιακού (business) μέρους της εφαρμογής που βρίσκεται στον εξυπηρετητή. Επίσης είναι σαφώς πιο «ελαφριά» και περισσότερο εύκολα στη διαχείριση από την παλαιοτέρων ετών δημοφιλή προσέγγιση CGI.

Επειδή τα servlets είναι απλούστερα και χρειάζονται γενικά λιγότερους πόρους, αρκετοί προγραμματιστές προ τιμούν να τα χρησιμοπο ιούν, μαζί με JSPs, σχεδόν αποκλειστικά στις υλοποιήσεις τους από το να κάνουν χρήση των αρκετά πολύπλοκων συστατικών λογισμικού EJB. Η χρήση απλών servlets είναι αποδοτική για πολύ απλές εφαρμογές, αλλά υπολείπεται όταν απαιτείται υποστήριξη δοσοληψιών (transaction) από την εφαρμογή.



Εικόνα 5-9. Τεχνολογίες ιστού Java που έχουν ως υποδομή την τεχνολογία των servlet.

Τα servlets είναι πιο αποδοτικά όταν διαχειρίζονται απλές εφαρμογές, όπως η συλλογή παραμέτρων και ο έλεγχος πληροφορίας εισόδου από τα πεδία εισόδου μιας σελίδας ιστού. Όταν οι αρχικοί έλεγχοι γίνουν, η πληροφορία πρέπει να διοχετεύεται σε κατάλληλα συστατικά λογισμικού για την εκτέλεση των λειτουργιών. Τα servlets εκτελούνται μέσα στον υποδοχέα servlet (πολλές φορές μπορεί να αναφέρεται και ως μηχανή servlet «servlet-engine») που φιλοξενείται μέσα σε έναν εξυπηρετητή ιστού. Ο υποδοχέας servlet διαχειρίζεται τις κλήσεις ιστού του πελάτη, οι οποίες γίνονται μέσω του πρωτοκόλλου HTTP (Hypertext Transfer Protocol), και τις μετασχηματίζει σε κλήσεις αντικειμένων (objects). Παρομοίως, ο υποδοχέας servlet μετασχηματίζει τις αποκρίσεις των servlets και τις αντιστοιχεί σε αποκρίσεις για αντικείμενα κατάλληλου πρωτοκόλλου ιστού (Εικόνα 5-10).



Εικόνα 5-10. Διαχείριση αίτησης εφαρμογής ιστού.

5.7.2 Java Server Pages (JSPs)

Τα JSP είναι ένας άλλος τύπος συστατικού λογισμικού ιστού J2EE και έχει σχέση με την τεχνολογία servlet. Στην πραγματικότητα, τμήματα των JSP μεταφράζονται σε servlets τα

οποία στη συνέχεια εκτελούνται εντός του περιβάλλοντος του servlet container. Τα JSP δημιουργήθηκαν για να γίνεται πιο εύκολη η συντήρηση τμημάτων ενός συστήματος, τα οποία υποστηρίζουν το κομμάτι της παρουσίασης ιστού της εφαρμογής, από τα μέλη μιας ομάδας ανάπτυξης των διεπαφών. Οι σχεδιαστές σελίδων ιστού (web designers) συντηρούν τον κώδικά παρουσίασης, σε γλώσσα HTML (HyperText Markup Language), κατι που είναι σαφώς πιο δύσκολο να γίνει όταν ή HTML δημιουργείται από εντολές Java που περιέχονται εντός των servlet.

5.7.3 Enterprise Java Bean - EJB (Επιχειρησιακά συστατικά λογισμικού)

Η EJB προδιαγραφή αποτελεί τον πυρήνα της πλατφόρμας J2EE. Αυτή ορίζει ένα αναλυτικό μοντέλο συστατικών λογισμικού για αποδοτικό κτίσιμο εφαρμογών και κατανεμημένων, από την πλευρά του εξυπηρετητή, επιχειρησιακών συστατικών λογισμικού Java.

Υπάρχουν τρείς τύποι EJB που συνοψίζονται ως εξής:

- Τα session bean, προορίζονται για την υλοποίηση υπηρεσιών. Δεν συγκρατούν τη κατάστασή τους (δεν είναι persistent) και συνήθως εμπεριέχουν το μεγαλύτερο επιχειρησιακό κομμάτι σε μία εφαρμογή Java. Τα session bean μπορεί να είναι stateful, πράγμα που σημαίνει ότι διατηρούν τη σχέση τους με έναν πελάτη, μεταξύ αλλεπάλληλων επαφών με αυτόν. Ο άλλος τύπος των session beans είναι τα stateless. Στην περίπτωση των stateless session bean κάθε αλλεπάλληλη κλήση του ίδιου session bean από τον ίδιο πελάτη εκλαμβάνεται ως νέα διαδικασία, μη συσχετιζόμενη με τυχόν προηγούμενες.
- Τα entity bean υπάρχουν για την καταχώριση πληροφορίας της βάσης δεδομένων σε ένα μόνιμο μέσω αποθήκευσης, δηλαδή αποθήκευση τυπικά μια πλήρους ή ενός τμήματος γραμμής πληροφορίας από αυτές που βρίσκονται σε ένα πίνακα μιας βάσης δεδομένων. Τα entity bean παρέχουν αυτοματοποιημένες υπηρεσίες για την επιβεβαίωση ότι η αντικειμενοστραφής απεικόνιση αυτών των μόνιμων δεδομένων παραμένει συγχρονισμένη συνεχώς με τα ενεργά δεδομένα της υποδομής τη βάσης δεδομένων. Επίσης τα entity bean χρησιμοποιούνται για τη μορφοποίηση των δεδομένων των βάσεων δεδομένων, είτε για να βοηθήσουν σε μια εργασία ή απ' την άλλη να προετοιμάσουν τα δεδομένα για παρουσίαση σε μια σελίδα ιστού. Για παράδειγμα, σε ένα πίνακα υπαλλήλων μιας βάσης δεδομένων, κάθε έγγραφή (γραμμή του πίνακα) αντιστοιχίζεται σε ένα στιγμιότυπο π.χ. του Employee entity bean.
- Τα **message-driven beans** έχουν σχεδιαστεί για να παρέχουν ασύγχρονή ανταλλαγή μηνυμάτων JMS (Java Messaging Service), δηλαδή ό καλών δεν περιμένει απάντηση κατά την κλήση μιας λειτουργίας. Αντίθετα από τα session και τα entity beans, τα

message-driven beans δεν έχουν δημοσιευμένες διεπαφές χρήστη. Τα message-driven beans δρουν ανώνυμα στο παρασκήνιο, δεν διατηρούν την κατάσταση τους (είναι stateless) και είναι σχετικά ένας καινούριος τύπος συστατικού λογισμικού EJB ο οποίος εισήχθηκε με την έκδοση 1.3 της προδιαγραφής J2EE.

5.7.4 APIs και άλλα εργαλεία J2EE

Η πλατφόρμα J2EE υποστηρίζει αρκετές διαφορετικές διεπαφές (API) (Εικόνα 5-11). Παρόλο που η χρήση τους δεν περιορίζεται σε αποκλειστικά ένα αρχιτεκτονικό επίπεδο, αυτές οι τεχνολογίες είναι που κάνουν τα διάφορα στρώματα να επικοινωνούν. Μερικά από τα πιο δημοφιλή παρουσιάζονται στις επόμενες παραγράφους.

5.7.4.1 Java Database Connectivity (JDBC)

Η διασύνδεση του στρώματος EIS (EIS: Enterprise Information Systems) με τις βάσεις δεδομένων των επιχειρησιακών πληροφοριακών συστημάτων αποτελεί ένα ζωτικό σημείο για μια Java εφαρμογή. Το JDBC API δημιουργήθηκε για να κάνει την παραπάνω διαδικασία ευκολότερη. Μέσω του JDBC ο προγραμματιστής αλληλεπιδρά με μια βάση δεδομένων μέσω ερωτημάτων SQL. Το JDBC API, το οποίο έχει παρόμοιες λειτουργίες με το ODBC (ODBC: Open Database Connectivity) API της Microsoft, απλοποιεί την πρόσβαση στις σχεσιακές βάσεις δεδομένων και αποτελεί μια γενική, ανεξάρτητη από κατασκευαστή, διεπαφή για βάσεις δεδομένων. Η χρήση του JDBC προσδίδει στις εφαρμογές φορητότητα υποστηρίζοντας τις δυνατότητες της βάσης δεδομένων ανεξάρτητα της πλατφόρμας υλοποίησης της εφαρμογής. Παρόλα αυτά η έκδοση J2EE προσθέτει, κατά κύριο λόγο την υποστήριξη κάποιων προχωρημένων λειτουργιών για τους υποδοχείς που χρησιμοποιεί, όπως η εναλλαγή συνδέσεων (connection pooling) καθώς επίσης και κάποια επιπρόσθετη υποστήριξη για Java Bean.

Το JDBC API παρέχει μια κοινόχρηστη διεπαφή για να απομονώσει τον χρήστη από τις διαφορετικές προδιαγραφές του κάθε κατασκευαστή. Έτσι διαφορετικές υλοποιήσεις βάσεων δεδομένων, διαφόρων κατασκευαστών, μπορούν να υπάρχουν ξεχωριστά κάτω από τη διαπροσωπεία του JDBC.

Στις επιχειρησιακές εφαρμογές, δεν είναι απαραίτητη η απευθείας χρήση του JDBC. Για παράδειγμα, μπορεί κανείς να χρησιμοποιήσει entity beans να κάνουν τις απαραίτητες κλήσεις στη βάση δεδομένων αντί για τον χρήστη. Η πρακτική της απευθείας χρήσης JDBC αναμένεται να περιοριστεί όταν οι εξυπηρετητές εφαρμογών παρέχουν περισσότερη και πιο συντονισμένη υποστήριξη για entity beans.

5.7.4.2 Java Naming and Directory Interface (JNDI)

Όλοι οι J2EE servers χρησιμοποιούν JNDI, η οποία είναι μια υπηρεσία αναζήτησης αντικειμένων με βάση το όνομα που χρησιμοποιείται για την εύρεση κατανεμημένων αντικειμένων (object). Οι υπηρεσίες σαν και αυτή επιτρέπουν σε κάποιον να καλεί τις υπηρεσίες ενός αντικειμένου, για το οποίο δεν υπάρχει κάποια αναφορά κατά το χρόνο μεταγλώττισης του προγράμματος. Ένα σύστημα αρχείων είναι ένα παράδειγμα υπηρεσίας αναζήτησης αντικειμένων με βάση το όνομα. Μια υπηρεσία καταλόγου (directory) είναι παρόμοια με την υπηρεσία αναζήτησης αντικειμένων με βάση το όνομα και παρέχει βελτιωμένες δυνατότητες αναζήτησης. Στην πραγματικότητα, μια υπηρεσία καταλόγου περιέχει μια υπηρεσία αναζήτησης αντικειμένων με βάση το όνομα (και όχι το αντίθετο).

Υπάρχει ποικιλία διαθέσιμων υπηρεσιών αναζήτησης αντικειμένων με βάση το όνομα, και οι κλήσεις που δέχεται αυτή η διασύνδεση είναι παρόμοιες με εκείνες των βάσεων δεδομένων. Το JNDI έχει σχεδιαστεί για την διευθέτηση αυτών των κλήσεων μέσω της παροχής ενός γενικού και ενιαίου τρόπου για την προσπέλαση των υπηρεσιών. Το ολοκληρωμένο JNDI παρόλο που έχει καταχωρηθεί ως τεχνολογία επιχειρησιακού τύπου API υπάρχει ήδη ως μέρος της J2SE. Οι πιο πολλές κατανεμημένες επιχειρησιακές εφαρμογές κάνουν χρήση αυτής της υπηρεσίας σε κάποιο σημείο. Για παράδειγμα, κάθε χρήση των EJBs σε μια επιχειρησιακή εφαρμογή καθιστά αναγκαίο ότι το JNDI χρησιμοποιείται για την εύρεση της συσχετιζόμενης EJB διεπαφής.

5.7.4.3 Java Message Service (JMS)

Είναι μια υπηρεσία μηνυμάτων που επιτρέπει την επικοινωνία μεταξύ κατανεμημένων εφαρμογών. Κατευθύνει τα μηνύματα που ανταλλάσσονται μεταξύ διαδικασιών και λογισμικών στοιχείων σε μια κατανεμημένη εφαρμογή, χρησιμοποιώντας αυτόνομα entities, τα οποία είναι υπεύθυνα για την κλήση των μηνυμάτων. Αυτή η επικοινωνία είναι τυπικά ασύγχρονη. Οι διάφοροι κατασκευαστές παρέχουν υπηρεσίες μηνυμάτων οι οποίες είναι προσανατολισμένες στο μεσαίο στρώμα των εφαρμογών. Η JMS υπηρεσία παρέχει μια ενιαία και γενική διεπαφή στο παραπάνω στρώμα. Η JMS υπηρεσία μπορεί να χρησιμοποιηθεί απευθείας σε μια επιχειρησιακή εφαρμογή ή μέσω ενός τύπου EJB γνωστού ως message-driven bean.

5.7.4.4 Java Authentication and Authorization Service (JAAS)

Το JAAS (Java Authentication and Authorization Service) είναι μια υπηρεσία που δίνει τη δυνατότητα σε μια J2EE εφαρμογή, για τη πιστοποίηση (authentication) και εξουσιοδότηση (authorization) χρηστών ή ομάδων χρηστών που έχουν τη δυνατότητα να την χρησιμοποιήσουν. Το JAAS είναι μια έκδοση της γλώσσας προγραμματισμού Java που προτυποποιεί την Pluggable Authentication Module (PAM) πλατφόρμα ανάπτυξης

(framework), η οποία επεκτείνει την αρχιτεκτονική ασφάλειας της πλατφόρμας Java 2 για την παροχή εξουσιοδότησης βασισμένη στον χρήστη.

5.7.4.5 Remote Method Invocation (RMI)

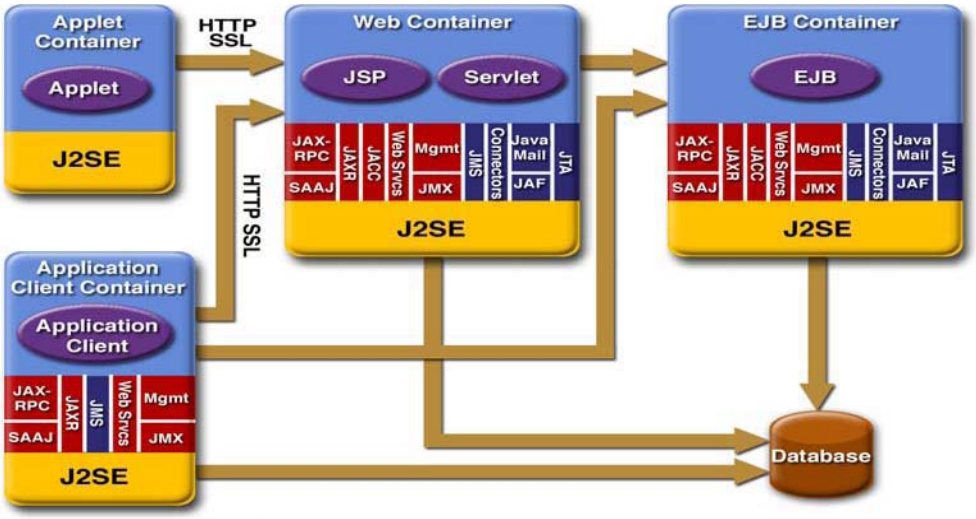
Αποτελεί προτυποποίηση για κλήσεις σε απομακρυσμένα αντικείμενα στη γλώσσα Java. Τα EJBs χρησιμοποιούν RMI. Το RMI ενεργοποιεί την πρόσβαση σε συστατικά λογισμικού, σε ένα κατανεμημένο περιβάλλον, επιτρέποντας σε αντικείμενα Java να καλούν μεθόδους άλλων απομακρυσμένων αντικειμένων Java. Η μέθοδος στην πραγματικότητα καλείται από ένα εξουσιοδοτημένο αντικείμενο, το οποίο είναι υπεύθυνο για τη μεταγωγή των παραμέτρων της μεθόδου μεταξύ του αντικειμένου το οποίο παρέχει την απόκριση και του αντικειμένου που ξεκίνησε την κλήση της απομακρυσμένης μεθόδου. Το RMI δεν υπάρχει αποκλειστικά για την J2EE πλατφόρμα. Παρόλα αυτά, το RMI είναι η καρδιά κάποιων τεχνολογιών J2EE, όπως της EJB.

5.7.4.6 Extensible Markup Language (XML)

Αν και δεν είναι και τόσο μια τεχνολογία J2EE, η XML χρησιμοποιείται από πολλές άλλες τεχνολογίες J2EE. Για παράδειγμα, στα συστατικά λογισμικού ιστού και στα επιχειρησιακά συστατικά λογισμικού (EJBs) οι περιγραφείς εγκατάστασης τους (deployment descriptors) είναι γραμμένοι σε γλώσσα XML. Αυτοί οι περιγραφείς εγκατάστασης περιγράφουν πως συμπεριφέρονται τα στοιχεία λογισμικού αφού εγκατασταθούν.

5.7.4.7 Java Transaction API

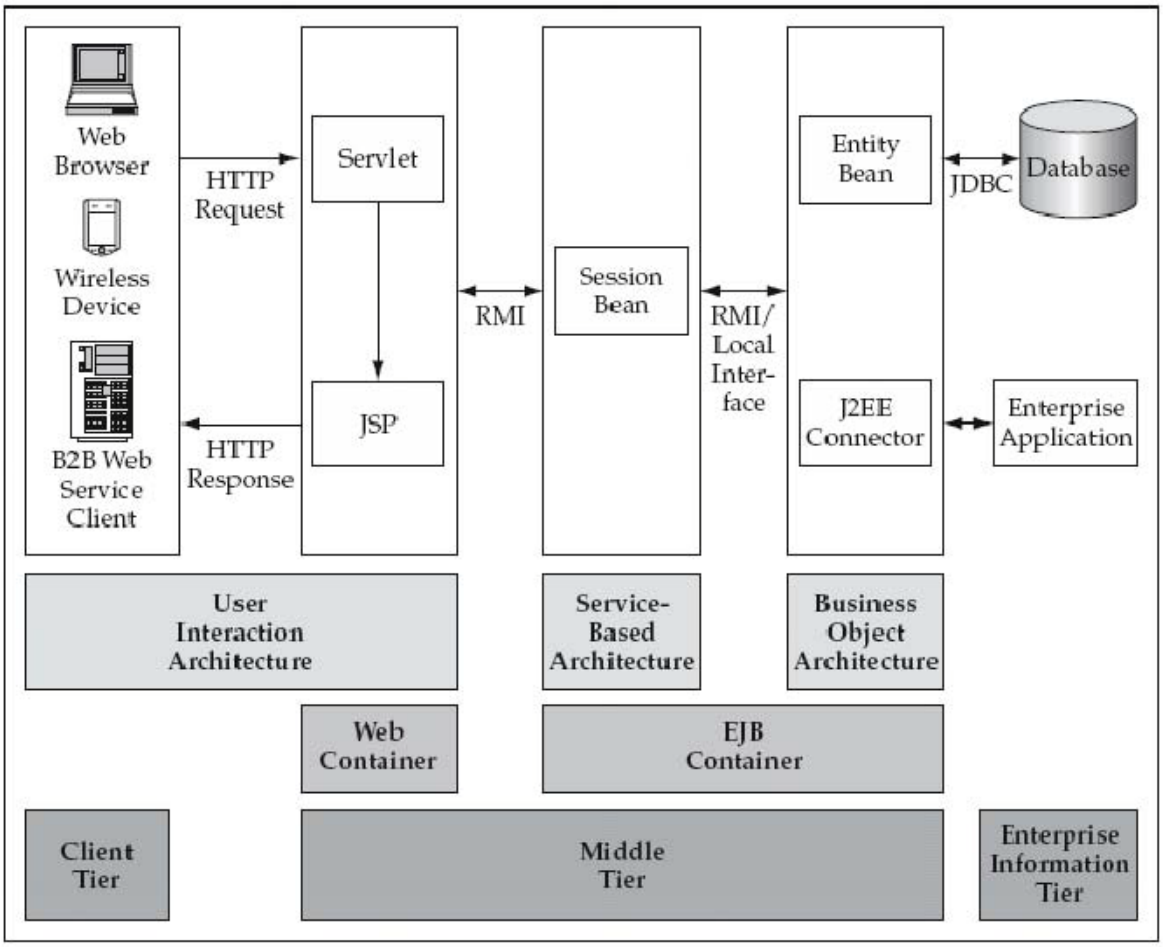
Το Java Transaction API (JTA) αποτελεί προδιαγραφή για την οριοθέτηση των δοσοληψιών (transactions). Η αρχιτεκτονική J2EE παρέχει ένα προκαθορισμένο τρόπο αυτόματης διενέργειας δοσοληψιών. Η έννοια αυτόματη διενέργεια δοσοληψιών σημαίνει, ότι οποιαδήποτε άλλη εφαρμογή έχει αναφορά στα δεδομένα που επηρεάζει μια δοσοληψία θα μπορεί να παρέμβει, στα ενημερωμένα δεδομένα, μόνο μετά από κάθε λειτουργία διαβάσματος ή γραψίματος της βάσης δεδομένων. Παρόλα αυτά, εάν μια εφαρμογή εκτελεί δύο διαφορετικές λειτουργίες πρόσβασης οι οποίες έχουν αλληλεξάρτηση μεταξύ τους, θα πρέπει να γίνει χρήση του JTA API για να οριοθετήσει, την εκκίνηση, το κλείσιμο και την διενέργεια μιας εσωτερικής δοσοληψίας που εμπεριέχει μια διπλή λειτουργία.



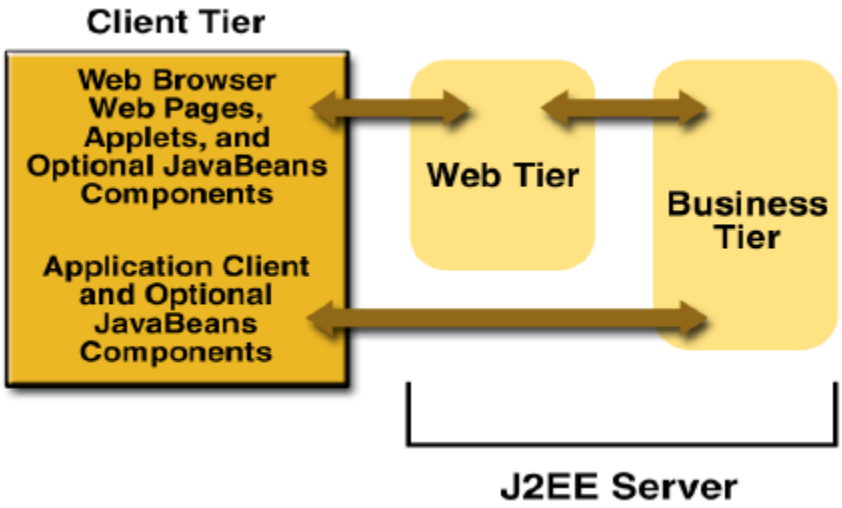
Εικόνα 5-11. Τα APIs της πλατφόρμας J2EE.

5.8 Εξυπηρετητές εφαρμογών J2EE

Η προδιαγραφή J2 Œ ορίζει ένα σύνολο από διεπαφές και κλάσεις (Εικόνα 5-12). Η κατασκευαστές λογισμικού παρέχουν εφαρμογές για τις παραπάνω διεπαφές, οι οποίες ακολουθούν πιστά τις προδιαγραφές του J2EE. Οι εφαρμογές αυτές ονομάζονται εξυπηρετητές εφαρμογών (application servers) J2EE (Εικόνα 5-13).Οι εξυπηρετητές εφαρμογών J2EE προσφέρουν την υποδομή για την παροχή υπηρεσιών όπως νημάτωσης (threading), πολιτικής εναλλαγής πόρων (pooling) και διαχείρισης δοσοληψιών (transaction management), ώστε ο προγραμματιστής, όσο είναι δυνατόν, να μην ασχολείται με την ανάπτυξη διαχειριστικού κώδικα και να ασχολείται κυρίως με την υλοπο ήση του επιχειρησιακού τμήματος της εφαρμογής.



Εικόνα 5-12. Βασική Αρχιτεκτονική J2EE.

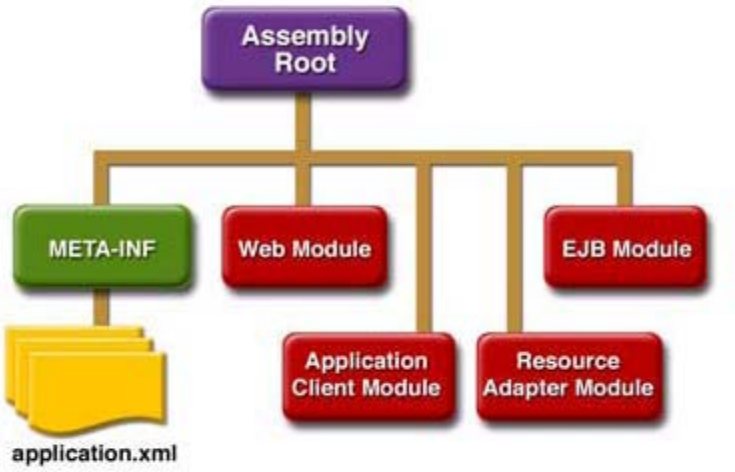


Εικόνα 5-13. Επικοινωνίες εξυπηρετητή J2EE.

5.9 Συσκευασία εφαρμογών

Μια εφαρμογή J2EE πρέπει τοποθετείται σε ένα επιχειρησιακό αρχείο (EAR Enterprise Archive) το οποίο έχει την επέκταση .EAR. Όταν δημιουργεί κανείς μια εφαρμογή J2EE,

αυτή αποτελείται συνήθως, από μια εφαρμογή ιστο ύ (web) και από μια enterp ise bean εφαρμογή (ejb). Κατά τη μεταγλώττιση των παραπάνω εφαρμογών δημιουργούνται αυτόματα, ένα αρχείο με επέκταση .WAR (Web Archive) και ένα αρχείο EJB-JAR (EJB Java Archive) τα οποία ενθυλακώνονται σε ένα αρχείο EAR (Enterprise Archive). Αυτό είναι το αρχείο που τοποθετείται εντός του εξυπηρετητή εφαρμογών. Δεν χρειάζεται επιπλέον κώδικας, είναι μονάχα ζήτημα δόμησης (ή συσκευασίας) των διαφόρων προδιαγραφών J2EE μέσα σε EAR αρχεία J2EE (Εικόνα 5-14).



Εικόνα 5-14. Δομή αρχείων μιας επιχειρησιακής εφαρμογής.

6

ΠΛΑΤΦΟΡΜΑ ΑΝΑΠΤΥΞΗΣ SPRING

Τα τελευταία χρόνια έχει παρατηρηθεί δραματική αλλαγή στον τρόπο ανάπτυξης μεγάλων Java εφαρμογών. Η χρήση των λεγόμενων 'βαρέων' αρχιτεκτονικών (Heavyweight Architectures) έχει περιοριστεί και τη θέση τους έχουν πάρει πιο 'ελαφριά' περιβάλλοντα εργασίας (frameworks). Πολύπλοκες και εξαρτώμενες από την κάθε τεχνολογία υπηρεσίες, όπως το object/relational mapping (ORM) και τα συστήματα διαχείρισης συναλλαγών (transaction management systems), έχουν αντικατασταθεί από πιο απλές εναλλακτικές. Ένα τέτοιο, πολλά υποσχόμενο και δημοφιλές, περιβάλλον εργασίας είναι η Spring. Η Spring αποτελεί ένα υψηλής ποιότητας και ευχρηστίας project το οποίο ικανοποιεί τις ανάγκες και τις απαιτήσεις των πραγματικών Java εφαρμογών, ενσωματώνοντας όλα τα βασικά χαρακτηριστικά της πλατφόρμας J2EE και προσαρμόζοντάς τα για την αποδοτικότερη ανάπτυξη των εφαρμογών όσον αφορά το κόστος υλοποίησης και λειτουργίας τους.

6.1 Εισαγωγή στη Spring

Η Spring είναι ένα ελεύθερο (open source) περιβάλλον εργασίας για εφαρμογές Java που δημιουργήθηκε από τον Rob Johnson και περιγράφηκε στο βιβλίο του ''Expert One-on-One: J2EE Design and Development''. Αναπτύχθηκε κατά κύριο λόγο, για να αντιμετωπίσει την πολυπλοκότητα ανάπτυξης επιχειρησιακών εφαρμογών. Η Spring κάνει εφικτή την χρήση απλών JavaBeans για την επίτευξη λειτουργιών που προηγουμένως μπορούσαν να γίνουν μόνο μέσω EJBs. Παρόλα αυτά, η Spring δεν είναι μόνο χρήσιμη για την ανάπτυξη server-side εφαρμογών. Η χρήση της μπορεί να βοηθήσει στην απλοποίηση του κώδικα, τον

ευκολότερο και αποτελεσματικό έλεγχο και τη χαλαρή διασύνδεση (loose coupling) κάθε Java εφαρμογής.

Η Spring εκτελεί πολλές διαφορετικές λειτουργίες, αλλά, αν αναλυθεί στα βασικά της τμήματα, θα φανεί ότι είναι ένας ελαφρύς (lightweight) aspect-oriented container ο οποίος εφαρμόζει dependency injection, ενώ παράλληλα αποτελεί και περιβάλλον εργασίας (framework). Πιο αναλυτικά:

- Lightweight (Ελαφρύς) — Ο όρος αυτός αναφέρεται τόσο στο μέγεθος της πλατφόρμας όσο και στον φόρτο εργασίας (overhead) που απαιτείται. Η διανομή του Spring Framework μπορεί να συμπεριληφθεί σε ένα απλό jar αρχείο μεγέθους 2,5 MB, ενώ ο επιπρόσθετος φόρτος εργασίας που απαιτείται είναι αμελητέος. Ο προγραμματιστής χρειάζεται να κάνει ελάχιστες, αν όχι καθόλου, αλλαγές στον κώδικα της εφαρμογής που αναπτύσσει έτσι ώστε να επωφεληθεί από τον πυρήνα της, ενώ μπορεί οποιαδήποτε στιγμή να σταματήσει να τη χρησιμοποιεί. Βέβαια αυτή η δυνατότητα παρέχεται μόνο στον πυρήνα της Spring. Πολλά επιπλέον μέρη της, όπως είναι η πρόσβαση δεδομένων, απαιτούν πολύ στενότερη 'διασύνδεση' (coupling) από ότι το Spring framework. Εντούτοις, το κέρδος στις περισσότερες από αυτές τις περιπτώσεις είναι πολύ σημαντικό.
- Dependency Injection (Εξαρτώμενη Έγχυση) — Η Spring προάγει τη χαλαρή διασύνδεση χρησιμοποιώντας μία τεχνική που είναι γνωστή ως dependency injection (DI). Όταν εφαρμόζεται η DI, τα αντικείμενα λαμβάνουν παθητικά τις εξαρτήσεις τους (dependencies) αντί να τις δημιουργούν ή να τις αναζητούν μόνα τους. Είναι κάτι σαν ένα αντίστροφο Java Naming and Directory Interface (JN-DI) όπου αντί ένα αντικείμενο να ψάχνει μόνο του για τις εξαρτήσεις του σε έναν υποδοχέα (container), ο υποδοχέας δίνει τις εξαρτήσεις στο αντικείμενο χωρίς να περιμένει πρώτα να ερωτηθεί.
- Aspect - Oriented — Η Spring παρέχει πλούσια υποστήριξη σε Aspect - Oriented Programming (AOP). Αυτό έχει σαν αποτέλεσμα τη δημιουργία εφαρμογών με περισσότερη συνοχή ακόμη και αν είναι απαραίτητη η συνύπαρξη διαφορετικών λογικών λειτουργίας. Έτσι, κάθε αντικείμενο της εφαρμογής περιορίζεται στις δικές του λειτουργίες και δεν είναι υπεύθυνο (ενδεχομένως δε γνωρίζει καν) για λειτουργίες που έχουν να κάνουν με άλλα συστήματα. Τέτοιου είδους λειτουργίες μπορεί να είναι η υποστήριξη συνδιαλλαγών ή η καταγραφή των διαφόρων κινήσεων (logging).
- Container (Υποδοχέας) — Η Spring αποτελεί έναν container με την έννοια ότι περιέχει τα αντικείμενα της εφαρμογής και διαχειρίζεται τον κύκλο ζωής τους. Ο προγραμματιστής μπορεί να ορίσει τον τρόπο με τον οποίο θέλει να διαμορφώνονται

και να δημιουργούνται τα αντικείμενα, καθώς και ποιες θέλει να είναι οι σχέσεις μεταξύ τους.

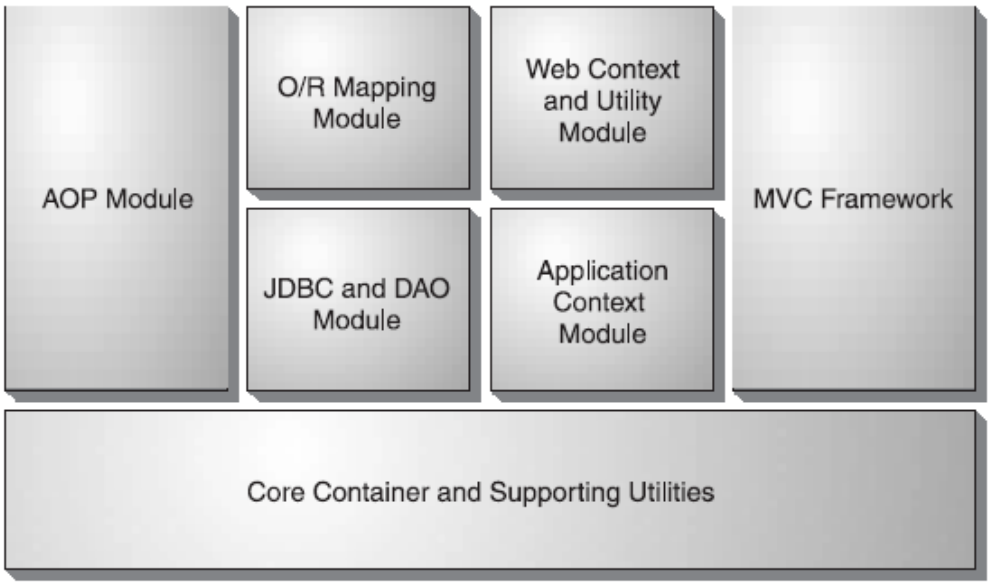
- Framework (Περιβάλλον Εργασίας) — Η Spring δίνει τη δυνατότητα να δημιουργηθούν πολύπλοκες εφαρμογές με το συνδυασμό άλλων, λιγότερο πολύπλοκων, κομματιών. Στη Spring τα αντικείμενα δημιουργούνται μέσω δηλώσεων σε απλά XML αρχεία. Επίσης παρέχει πολλές δομικές λειτουργίες, όπως η διαχείριση συναλλαγών κ.ά., επιτρέποντας στον προγραμματιστή να ασχοληθεί περισσότερο με τη λογική της εφαρμογής του.

Έτσι, μπορούμε να πούμε ότι η Spring είναι ένα περιβάλλον εργασίας το οποίο βοηθάει τον προγραμματιστή να δημιουργήσει εφαρμογές με χαλαρά διασυνδεδεμένο κώδικα. Ακόμη και αν αυτό ήταν το μόνο πλεονέκτημα αυτής της πλατφόρμας, τα οφέλη θα ήταν πάρα πολλά από άποψη συντηρησιμότητας και ελεγξιμότητας. Ωστόσο τα πλεονεκτήματα της Spring δεν περιορίζονται στα παραπάνω. Περιέχει κομμάτια που χρησιμοποιούν DI και AOP και έτσι συνθέτουν μία αρκετά αξιόλογη πλατφόρμα στην οποία μπορούν να αναπτυχθούν εφαρμογές.

6.1.1 Τα modules της Spring

Το περιβάλλον εργασίας της Spring αποτελείται από αρκετά, καλά ορισμένα, κομμάτια (modules). Τα κομμάτια αυτά σαν σύνολο δίνουν στον προγραμματιστή ότι χρειάζεται, για να αναπτύξει enterprise εφαρμογές. Δεν απαιτείται από τον προγραμματιστή να βασίσει όλη την εφαρμογή του πάνω στη Spring. Αντίθετα, του παρέχεται η δυνατότητα να επιλέξει όποια από τα modules πιστεύει ότι καλύπτουν τις ανάγκες του. Επιπλέον παρέχονται τρόποι σύνδεσης με άλλα περιβάλλοντα εργασίας και βιβλιοθήκες έτσι ώστε ο προγραμματιστής να μη χρειαστεί να τα αναπτύξει από την αρχή.

Όπως φαίνεται στην Εικόνα 6-1 όλα τα modules της Spring είναι χτισμένα πάνω από τον πυρήνα. Στον πυρήνα της Spring εκτελούνται οι κυριότερες λειτουργίες όπως ο καθορισμός του τρόπου δημιουργίας των beans καθώς και του τρόπου αρχικοποίησης και χειρισμού τους. Όμως ο προγραμματιστής ενδιαφέρεται περισσότερο για τα modules που χρησιμοποιούν τον πυρήνα και τις υπηρεσίες που αυτός παρέχει.



Εικόνα 6-1. Τα modules της Spring.

6.1.1.1 Ο πυρήνας — core container

Ο πυρήνας της Spring παρέχει όλη τη βασική λειτουργικότητα του Spring Framework. Το module αυτό περιέχει το BeanFactory, το οποίο είναι ο θεμελιώδης container και η βάση με την οποία η Spring υλοποιεί το DI.

6.1.1.2 Το DAO module

Η δουλειά με JDBC συχνά περιλαμβάνει τη δημιουργία αρκετού κώδικα, ο οποίος κάνει βασικές και τετριμμένες ενέργειες όπως σύνδεση με τη βάση, δημιουργία κάποιας εντολής (statement), επεξεργασία του αποτελέσματος και κλείσιμο της σύνδεσης. Μέσω του Data Access Object (DAO) module η Spring δίνει τη δυνατότητα απλοποίησης όλης αυτή της διαδικασίας περιορίζοντας τα προβλήματα που προκύπτουν. Επιπλέον, χτίζει ένα στρώμα (layer) από εξαιρέσεις (exceptions) πάνω από τα μηνύματα σφάλματος που παράγουν αρκετοί εξυπηρετητές βάσεων δεδομένων. Επιπρόσθετα, αυτό το module (χρησιμοποιώντας το AOP module) παρέχει υπηρεσίες διαχείρισης συναλλαγών για τα αντικείμενα των εφαρμογών που χρησιμοποιούν τη Spring.

6.1.1.3 Το ORM module

Για τους προγραμματιστές που προτιμούν να χρησιμοποιούν το εργαλείο object - relational mapping (ORM) (αντιστοίχηση σχέσης - αντικειμένου) αντί για την απευθείας χρήση του JDBC, η Spring παρέχει το ORM module. Παρόλο που η Spring δεν παρέχει το δικό της ORM, υποστηρίζει αρκετά δημοφιλή περιβάλλοντα εργασίας, συμπεριλαμβανομένων των Hibernate, Java Persistence API, Java Data Objects και iBatis SQL Maps. Το σύστημα διαχείρισης συναλλαγών της Spring υποστηρίζει εκτός από αυτά τα περιβάλλοντα εργασίας και το JDBC.

6.1.1.4 To AOP module

Η Spring προσφέρει υποστήριξη για aspect - oriented programming στο AOP module της. Όπως και το DI, το AOP υποστηρίζει τη χαλαρή διασύνδεση μεταξύ των αντικειμένων κάθε εφαρμογής, όμως με το AOP σημαντικά θέματα που αφορούν το σύνολο των εφαρμογών (όπως οι συναλλαγές και η ασφάλεια) χωρίζονται από τα αντικείμενα στα οποία αυτά εφαρμόζονται.

6.1.1.5 To Web module

Η Spring παρέχει μία πλούσια συλλογή κλάσεων για τη δημιουργία εφαρμογών που στηρίζονται στο διαδίκτυο (web - based applications) μέσω του Web module της. Υποστηρίζεται η χρήση του προτύπου σχεδίασης εφαρμογών Model/View/Controller (MVC), ο χειρισμός ηλεκτρονικής αλληλογραφίας (mail support) καθώς και η χρήση απομακρυσμένων υπηρεσιών (remoting support).

Όσον αφορά τη χρήση του προτύπου σχεδίασης εφαρμογών Model/View/Controller (MVC), η Spring υποστηρίζει πλήρως το πιο γνωστό MVC περιβάλλον εργασίας, το Apache Struts. Με αυτόν τον τρόπο δίνεται η δυνατότητα στον προγραμματιστή να χρησιμοποιήσει στις ήδη σχεδιασμένες κλάσεις του, τις αρχές του dependency injection που προσφέρει η Spring. Βέβαια, πέραν της υποστήριξης του Apache Struts, η Spring προσφέρει και τη δική της υλοποίηση του MVC. Μέσω του αρχιτεκτονικού προτύπου MVC μπορεί να γίνει χρήση μίας ευρείας γκάμας τεχνολογιών παρουσίασης, από σελίδες JSP και Apache Jakarta Velocity μέχρι Microsoft Excel και Adobe PDF.

Σε σχέση με την αποστολή μηνυμάτων ηλεκτρονικής αλληλογραφίας, η Spring παρέχει ένα αρκετά απλοποιημένο API, το οποίο ταιριάζει με την όλη φιλοσοφία του DI που αυτή προσφέρει. Για το σκοπό αυτό παρέχονται δύο υλοποιήσεις, μία του JavaMail και μία της MailMessage κλάσης από το πακέτο com.oreilly.servlet του Jason Hunter. Μέσω αυτών δίνεται η δυνατότητα δημιουργίας ενός πρότυπου μηνύματος το οποίο στη συνέχεια μπορεί να χρησιμοποιηθεί σαν βάση για την αποστολή αλληλογραφίας μέσω της εφαρμογής.

Τέλος, σε σχέση με χρήση απομακρυσμένων υπηρεσιών, παρέχεται εκτεταμένη υποστήριξη μίας μεγάλης συλλογής τεχνικών απομακρυσμένης πρόσβασης για τη γρήγορη δημιουργία και πρόσβαση σε απομακρυσμένες υπηρεσίες. Τέτοιες τεχνικές είναι το Java RMI, το JAXRPC, το Caucho Hessian και το Caucho Burlap. Εκτός αυτών, η Spring παρέχει και το δικό της πρωτόκολλο, που χρησιμοποιεί το HTTP πρωτόκολλο επικοινωνίας και το οποίο βασίζεται στο απλό Java serialization.

6.1.1.6 Java EE Connector API (JCA)

Αν και τα τελευταία χρόνια όλο και περισσότεροι προγραμματιστές τείνουν να χρησιμοποιούν τις λεγόμενες ελαφριές πλατφόρμες με τεχνολογίες όπως η Spring και ο

Tomcat, υπάρχουν πολλές περιπτώσεις όπου η χρήση αρκετών παραδοσιακών J2EE APIs κρίνεται απαραίτητη. Η σύνδεση και επικοινωνία μεταξύ εφαρμογών τέτοιου τύπου και εφαρμογών που χρησιμοποιούν άλλες τεχνολογίες μπορεί να είναι δύσκολη. Το JCA της Spring παρέχει ένα σταθερό (stable) και αξιόπιστο τρόπο για την επικοινωνία με μία σειρά από τέτοιου είδους enterprise συστήματα. Τα κυριότερα APIs που υποστηρίζονται είναι το Java Naming and Directory Interface (JNDI), τα EJB καθώς και η Java Message Service (JMS).

6.1.2 Παράδειγμα

Το dependency injection αποτελεί το βασικότερο χαρακτηριστικό της Spring. Στη συνέχεια θα παρουσιαστεί μία απλή εφαρμογή, μία διαφοροποίηση του συνηθισμένου ‘‘Hello World’’, η οποία θα παρουσιάσει τα βασικά σημεία της Spring.

Η πρώτη κλάση που χρειάζεται η Hello World εφαρμογή είναι μία κλάση εξυπηρετητή (service class), ο σκοπός της οποίας είναι να τυπώσει το γνωστό χαιρετισμό. Παρακάτω φαίνεται το interface GreetingService το οποίο ορίζει τις συναρτήσεις της κλάσης εξυπηρετητή (Λίστα 6-1).

Λίστα 6-1. GreetingService.java

```
public interface GreetingService {
   void sayGreeting();
}
```

Ακολούθως παρουσιάζεται η συνάρτηση GreetingServiceImpl η οποία υλοποιεί (implements) το παραπάνω interface (Λίστα 6-2). Η χρήση interface για τον ορισμό των ενεργειών που μπορούν να γίνουν με τα διάφορα αντικείμενα δεν είναι απαραίτητη, εντούτοις συνιστάται.

Λίστα 6-2. GreetingServiceImpl.java

```
public class GreetingServiceImpl implements GreetingService {
   private String greeting;
   public GreetingServiceImpl () {}
   public GreetingServiceImpl (String greeting) {
      this.greeting = greeting;
   }
   void sayGreeting() {
      System.out.println(greeting);
   }
   public void setGreeting(String greeting) {
      this.greeting = greeting;
```

```
    }
}
```

Η κλάση GreetingServiceImpl έχει μία μόνο μεταβλητή μέλος, την greeting. Η μεταβλητή αυτή είναι ένα απλό String το οποίο θα κρατάει τον χαιρετισμό που θα τυπωθεί, όταν θα κληθεί η μέθοδος sayGreeting(). Επιπλέον έχει και δύο μεθόδους δημιουργούς (constructors), ο πρώτος είναι κενός και ο δεύτερος παίρνει σαν όρισμα τον χαιρετισμό. Οποιοσδήποτε από τους δύο αυτούς δημιουργούς μπορεί να κληθεί, ανάλογα με τις ρυθμίσεις που θα οριστούν. Στη συνέχεια (Λίστα 6-3) παρατίθεται το αρχείο ρυθμίσεων (configuration file) (hello.xml) το οποίο ορίζει στον πυρήνα της Spring πώς ακριβώς πρέπει να εκτελέσει την υπηρεσία που δημιουργήθηκε.

Λίστα 6-3. hello.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns=
       "http://www.springframework.org/schema/beans"
      xmlns:xsi="http://w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation=
       "http://www.springframework.org/
         shema/beans/spring-beans-2.0.xsd">
  <bean id="greetingService" class="GreetingServiceImpl">
    <property name="greeting" value="Hello World!">
  </bean>
</beans>
```

Στο παραπάνω αρχείο δηλώνεται ένα στιγμιότυπο της GreetingServiceImpl στον container της String στο οποίο ορίζεται ότι η μεταβλητή μέλος (property) θα έχει την τιμή ‘‘Hello World!’’.

Αναλύοντας λίγο τον παραπάνω XML κώδικα βλέπουμε ότι το root στοιχείο είναι το <beans>, κάτι που ισχύει για οποιοδήποτε αρχείο ρυθμίσεων της Spring. Το στοιχείο <bean> λέει στον container για μία κλάση και πώς αυτή πρέπει να αρχικοποιηθεί. Το χαρακτηριστικό (attribute) id του στοιχείου <bean> χρησιμοποιείται, για να ορίσει το όνομα του bean, ενώ το class χρησιμοποιείται για να οριστεί το ακριβές όνομα της κλάσης (class path).

Μέσα στο στοιχείο <bean> ορίζεται ένα στοιχείο <property> το οποίο χρησιμοποιείται, για να οριστεί μία μεταβλητή μέλος, σε αυτή την περίπτωση η greeting. Το στοιχείο property λέει στον πυρήνα να καλέσει την μέθοδο sayGreeting δίνοντάς της την τιμή ‘‘Hello World’’ κατά την αρχικοποίηση του bean.

Η λειτουργία που επιτελείται κατά την αρχικοποίηση της υπηρεσίας σύμφωνα με το XML αρχείο ρυθμίσεων αντιστοιχεί στον παρακάτω κώδικα:

```
GreetingServiceImpl greetingService =
   new GreetingServiceImpl();
greetingService.setGreeting("Hello World!");
```

Τέλος απομένει η δημιουργία μίας κλάσης η οποία να φορτώνει τον πυρήνα της Spring και να τον χρησιμοποιεί για την εκτέλεση της υπηρεσίας (Λίστα 6-4).

Λίστα 6-4. HelloApp.java.

```
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;
public class HelloApp {
  public static void main(String[] args)
      throws Exception
  {
    BeanFactory factory = new XmlBeanFactory
      (new FileSystemResource("hello.xml"));
    GreetingService service = (GreetingService)
      factory.getBean("greetingService");
    service.sayGreeting();
  }
}
```

Η κλάση BeanFactory που χρησιμοποιείται παραπάνω αποτελεί τον πυρήνα της Spring. Μετά τη φόρτωση του hello.xml αρχείου από τον πυρήνα η main() καλεί τη μέθοδο getBean(), ώστε να ανακτήσει τις εξαρτήσεις της υπηρεσίας. Έχοντας αυτές τις εξαρτήσεις καλείται τελικά η μέθοδος sayGreeting(). Όταν εκτελεστεί η εφαρμογή, όπως αναμενόταν, τυπώνεται το: Hello World!

Η παραπάνω εφαρμογή αποτελεί το πιο απλό πιθανό παράδειγμα χρήσης της Spring. Παρά την απλότητά του παρουσιάζει το βασικό τρόπο ρύθμισης και χρήσης κλάσεων μέσω της Spring. Όμως, λόγω της απλότητάς του, δε δείχνει πώς μπορεί να ρυθμιστεί κάποιο bean, ώστε να περαστεί η τιμή ενός String στο πεδίο (injection). Η πραγματική δύναμη της Spring έγκειται στο πώς μπορούν τα διάφορα beans να συνδεθούν με άλλα beans κάνοντας χρήση του DI.

6.1.3 Κατανοώντας το dependency injection

Αν και η Spring έχει πολλές δυνατότητες, το DI αποτελεί την καρδιά του περιβάλλοντος εργασίας. Παρά το γεγονός ότι ο όρος αυτός ακούγεται σαν μια περίπλοκη προγραμματιστική τεχνική ή τρόπο σχεδιασμού (design pattern) στην πραγματικότητα δεν είναι τόσο σύνθετο όσο ακούγεται. Αντίθετα, η χρήση του κάνει τον κώδικα πολύ πιο απλό, ευκολονόητο και εύκολα συντηρήσιμο.

Κάθε μη τετριμμένη εφαρμογή (δηλαδή οτιδήποτε πιο πολύπλοκο από το κλασικό παράδειγμα (HelloWorld.java) αποτελείται από δύο ή περισσότερες κλάσεις οι οποίες συνεργάζονται, για να εκτελέσουν τις απαραίτητες ενέργειες. Παραδοσιακά κάθε αντικείμενο είναι υπεύθυνο να διατηρεί τις αναφορές του (reference) στα αντικείμενα με τα οποία 'συνεργάζεται' (δηλαδή τις εξαρτήσεις του — dependencies). Κάτι τέτοιο οδηγεί σε έναν κώδικα με πολλές εξαρτήσεις, ο οποίος είναι δύσκολο να ελεγχθεί.

Όταν γίνεται χρήση του DI, τα αντικείμενα λαμβάνουν τις εξαρτήσεις τους κατά την ώρα της δημιουργίας τους από μία εξωτερική οντότητα η οποία συντονίζει κάθε αντικείμενο στο όλο σύστημα. Με άλλα λόγια, οι εξαρτήσεις (dependencies) δίνονται (injected) στα αντικείμενα. Έτσι, το DI αποτελεί ενός είδους αντιστροφής στην ευθύνη που αφορά τη διατήρηση των αναφορών των αντικειμένων. Η λειτουργικότητα αυτή παρέχεται από τον πυρήνα της Spring, ο οποίος είναι υπεύθυνος να δίνει στα αντικείμενα τις εξαρτήσεις τους. Ο πυρήνας γνωρίζει πώς πρέπει να δώσει αυτές τις εξαρτήσεις στα κατάλληλα αντικείμενα διαβάζοντας το XML αρχείο ρυθμίσεων της Spring.

Το κλειδί της παραπάνω τεχνικής είναι η χαλαρή διασύνδεση. Αν ένα αντικείμενο γνωρίζει για τις εξαρτήσεις του μόνο μέσω μίας διεπαφής (όχι μέσω της υλοποίησής τους ή μέσω του τρόπου που δημιουργούνται), η εξάρτηση μπορεί να αλλαχτεί πολύ εύκολα από μια διαφορετική υλοποίηση, χωρίς το αντικείμενο να γνωρίζει για τη διαφορά.

6.1.4 Aspect-Oriented Programming

Όπως περιγράφηκε στην προηγούμενη παράγραφο, το DI καθιστά εφικτή τη χαλαρή διασύνδεση τμημάτων λογισμικού σε εφαρμογές. Αντίθετα, το aspect-oriented programming επιτρέπει τη συγκέντρωση λειτουργιών οι οποίες χρησιμοποιούνται σε όλη την εφαρμογή, σε επαναχρησιμοποιήσιμα κομμάτια.

Το aspect-oriented programming συχνά ορίζεται ως μια τεχνική η οποία προάγει το διαχωρισμό ευθυνών μέσα στο σύστημα λογισμικού. Τα συστήματα αποτελούνται από πολλά ξεχωριστά κομμάτια, το καθένα από τα οποία είναι υπεύθυνο για συγκεκριμένο κομμάτι λειτουργικότητας. Συχνά όμως αυτά τα κομμάτια έχουν επιπλέον ευθύνες πέρα από τις βασικές λειτουργίες για τις οποίες προορίζονται. Υπηρεσίες όπως η διατήρηση αρχείου, η

διαχείριση συναλλαγών και η ασφάλεια, συχνά υλοποιούνται μέσα σε κομμάτια λογισμικού των οποίων η βασική ευθύνη είναι τελείως διαφορετική.

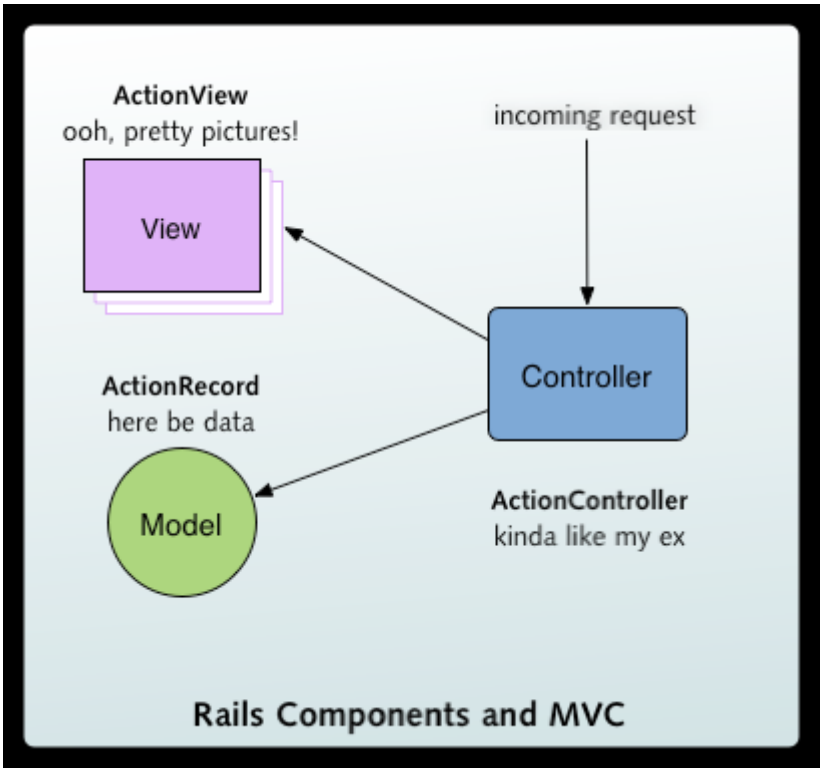
Μοιράζοντας τέτοιου είδους ευθύνες κατά μήκος πολλαπλών κομματιών κώδικα δημιουργούνται δύο επίπεδα πολυπλοκότητας στον κώδικα που αναπτύσσεται:

- Ο κώδικας ο οποίος υλοποιεί λειτουργίες που αφορούν όλη την εφαρμογή είναι διασκορπισμένος σε πολλά διαφορετικά τμήματα αντί να είναι συγκεντρωμένος. Αυτό σημαίνει ότι, αν ο προγραμματιστής αποφασίσει να αλλάξει τον τρόπο που εκτελούνται τέτοιου είδους λειτουργίες, πρέπει να ανατρέξει σε πολλά διαφορετικά σημεία του κώδικα. Ακόμα και αν η διαφοροποίηση έγκειται μόνο στην αλλαγή του τρόπου κλήσης μίας και μόνο μεθόδου, η διόρθωση θα πρέπει να γίνει σε πολλά διαφορετικά σημεία του κώδικα.
- Τα διάφορα κομμάτια κώδικα είναι επιφορτισμένα με λειτουργίες οι οποίες δεν τα αφορούν, προσθέτοντάς τους επιπλέον κώδικα. Για παράδειγμα, αν η λειτουργία μιας μεθόδου είναι να προσθέσει μία εγγραφή σε έναν τηλεφωνικό κατάλογο, δεν πρέπει αυτή να ελέγχει αν αυτή η διαδικασία θα γίνει με ασφάλεια.

Το AOP κάνει εφικτό το διαχωρισμό των διαφόρων υπηρεσιών επιτρέποντας να γίνεται χρήση τους, όποτε κάτι τέτοιο κρίνεται επιθυμητό, από τα κομμάτια κώδικα (components) που τις χρειάζονται. Αυτό οδηγεί στην ανάπτυξη πιο συνεκτικών κομματιών, τα οποία είναι περισσότερο συγκεντρωμένα στο βασικό τους στόχο, αγνοώντας τελείως τις διάφορες άλλες υπηρεσίες του συστήματος. Με λίγα λόγια η χρήση των aspects διασφαλίζει ότι τα POJOs θα παραμείνουν απλά (plain).

6.2 Αρχιτεκτονικό Μόρφημα MVC

Ο στόχος του μορφήματος αυτού είναι η αρχιτεκτονική σχεδίαση ενός συστήματος που θα κατανέμει τις ευθύνες της εφαρμογής σε ορισμένα σημαντικά συνθετήματά του, βάσει ορισμένων κανόνων. Ορισμένες από τις ευθύνες αυτές είναι η λήψη των αιτήσεων των χρηστών του συστήματος, ο προσδιορισμός της μονάδας που θα επεξεργαστεί την αίτηση του χρήστη, η ενημέρωση των δεδομένων του συστήματος, ο προσδιορισμός του τρόπου με τον οποίο το σύστημα θα ανταποκριθεί στον χρήστη και άλλες μικρότερης σημασίας. Το μόρφημα MVC όπως προκύπτει και από το όνομά του στηρίζεται σε τρία βασικά συνθετήματα: το Μοντέλο (MODEL), την Προβολή (VIEW) και τον Ελεγκτή (CONTROLLER) (Εικόνα 6-2).



Εικόνα 6-2. Αρχιτεκτονικό μόρφημα MVC.

Το Μοντέλο αποτελεί ένα συνθέτημα το οποίο αναπαριστά το μοντέλο της εφαρμογής. Ως μοντέλο της εφαρμογής εννοούνται τα δεδομένα του μοντέλου οι δομές δεδομένων στις οποίες αυτά αποθηκεύονται και οι διαδικασίες πρόσβασης, ενημέρωσης και αποθήκευσης των δεδομένων. Θα μπορούσε κάποιος να φανταστεί το μοντέλο ως μία οντότητα, που παρέχει μία διεπαφή, μέσω της οποίας άλλες οντότητες μπορούν να χρησιμοποιήσουν τα δεδομένα που περιέχει το μοντέλο.

Η Προβολή όπως πολύ εύκολα μπορεί να φανταστεί κάποιος δεν είναι τίποτε διαφορετικό από την εικόνα που το σύστημα εμφανίζει στον εξωτερικό χρήστη. Το συνθέτημα αυτό αποτελεί μία γέφυρα μεταξύ του συστήματος και των χρηστών του. Οποιαδήποτε αίτηση του χρήστη εκφράζεται μέσω της προβολής σε αίτηση προς το σύστημα και οποιαδήποτε αντίδραση του συστήματος προσφέρεται στον χρήστη πάλι μέσω της προβολής. Αν και ο χρήστης δεν το αντιλαμβάνεται αυτό, η προβολή δεν γνωρίζει καμία λεπτομέρεια σχετικά με το επιχειρησιακό μοντέλο του συστήματος και ούτε με τον τρόπο υλοποίησης των υπηρεσιών που του προσφέρονται. Στην ιδανική περίπτωση το συνθέτημα της προβολής αρκείται στο να λαμβάνει πληροφορίες από το μοντέλο και να τις παρουσιάζει στον χρήστη.

Το τρίτο συνθέτημα του μορφήματος MVC είναι ο ελεγκτής. Ο ελεγκτής αποτελεί την γέφυρα μεταξύ της προβολής και του μοντέλου. Πρόκειται για τον «αλγόριθμο» του συστήματος, σύμφωνα με τον οποίο το σύστημα λειτουργεί. Καθορίζει μετά από αίτηση του χρήστη ποιες αλλαγές και με ποια σειρά πρέπει να γίνουν στο μοντέλο του συστήματος.

Έπειτα προωθεί στον χρήστη το αποτέλεσμα, δίνοντας την σωστή εντολή στη διεπαφή της προβολή.

Όπως είναι προφανές ένα σύστημα σχεδιασμένο σύμφωνα με το μόρφημα MVC έχει συνθετήματα με ξεκάθαρες αρμοδιότητες. Η συντήρηση και η περαιτέρω ανάπτυξη της εφαρμογής είναι ευκολότερη. Δεδομένων μάλιστα της επεκτασιμότητας και ευελιξίας που είναι αναγκαίο να χαρακτηρίζουν την εφαρμογή, η επιλογή του μοντέλου MVC υπήρξε μονόδρομος. Όταν ένα σύστημα καλείται να αλλάζει τον τρόπο λειτουργίας του, ή την δομή δεδομένων του για να προσαρμοστεί σε άλλες εφαρμογές είναι εύλογη η ανάγκη σαφούς κατανομής των αρμοδιοτήτων των συνθετημάτων του. Ο προγραμματιστής μπορεί ανά πάσα στιγμή να γνωρίζει τι ακριβώς πρέπει να αλλάξει και σε ποιο σημείο θα βρει αυτό που αναζητά.

Η προβολή δεν πρέπει να περιέχει πληροφορίες σχετικά με το επιχειρησιακό μοντέλο του συστήματος. Καταρχήν τα διάφορα τμήματα της προβολής θα πρέπει να μπορούν να επαναχρησιμοποιούνται με ευκολία. Επίσης οι αλλαγές στα δύο παραπάνω μοντέλα δεν πρέπει να οδηγούν αναγκαστικά σε αλλαγές της προβολής. Τέλος θα πρέπει να τονιστεί ότι οι υπεύθυνοι για την προβολή του συστήματος, δεν σημαίνει ότι είναι άξιοι εμπιστοσύνης, ώστε να γνωρίζουν τον κώδικα που υλοποιεί την επιχειρησιακή λογική του συστήματος.

Αντίστοιχοι είναι και οι λόγοι για τους οποίους το μοντέλο δεδομένων πρέπει να είναι ανεξάρτητο του επιχειρησιακού μοντέλου που διέπει το σύστημα και ανεξάρτητο της προβολής, αλλά και το επιχειρησιακό μοντέλο να είναι ανεξάρτητο των δύο άλλων συνθετημάτων.

6.3 Η αρχιτεκτονική μιας εφαρμογής Spring MVC

Προτού αρχίσει η ανάλυση των χαρακτηριστικών της αρχιτεκτονικής Spring MVC, είναι σημαντικό να συζητηθεί πώς αναπτύσσεται μια τυπική εφαρμογή βασισμένη σε αυτό το αρχιτεκτονικό πρότυπο. Σε αυτό το κεφάλαιο, θα απαντηθούν ερωτήσεις του τύπου, «Που εμπεριέχεται η επιχειρησιακή λογική;» και «Ποια είναι τα σωστά επίπεδα αφαιρέσεων;» Θα παρουσιαστεί η ολοκληρωμένη εικόνα μιας Spring MVC εφαρμογής ιστού για να εξηγηθούν καλύτερα οι μεμονωμένοι ρόλοι που παίζει η αρχιτεκτονική Spring MVC καθώς και σε ποια σημεία εφαρμόζεται στα πλαίσια της αρχιτεκτονικής μιας εφαρμογής.

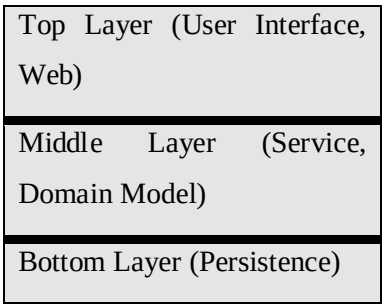
Η χρήση ενός νέου πλαισίου απαιτεί περισσότερα από τη μελέτη των προγραμματιστικών του διεπαφών (API: Application Programming Interface). Εξετάζοντας την αρχιτεκτονική μιας ολοκληρωμένης Spring MVC εφαρμογής ιστού κατανοούνται τα κίνητρα σχεδιασμού του ίδιου του πλαισίου, και συνεπώς επιτρέπει στον χρήστη να κάνει τις σωστές επιλογές κατά την κατασκευή της εφαρμογής.

6.3.1 Επίπεδα Αφαιρέσεων

Οι εφαρμογές Spring MVC αναλύονται σε μια σειρά στρωμάτων. Θεωρούμε ότι ένα στρώμα είναι μια διακριτή, ορθογώνια περιοχή ενδιαφέροντος μέσα σε μια εφαρμογή. Για παράδειγμα, ο κώδικας διατήρησης (persistence code) θεωρείται χωριστό στρώμα από τον κώδικα επεξεργασίας και εμφάνισης της προβολής (view rendering code). Τα στρώματα είναι αφαιρέσεις μέσα σε μια εφαρμογή, και οι διεπαφές παρέχουν τον τρόπο σύμφωνα με τον οποίο τα στρώματα αλληλεπιδρούν. Μερικά στρώματα μπορεί να είναι καλά κρυμμένα και να χρησιμοποιούνται μόνο από το στρώμα που βρίσκεται ακριβώς από πάνω τους. Αντίθετα, το σημαντικότερο στρώμα (το μοντέλο τομέα) εκτείνεται σχεδόν σε όλα τα στρώματα στο σύστημα.

Τα στρώματα είναι εννοιολογικά όρια και δεν είναι απαραιτήτως φυσικά απομονωμένα. Τις περισσότερες φορές, όλα τα στρώματα μιας εφαρμογής ιστού θα βρεθούν μέσα στην ίδια εικονική μηχανή Java (Java Virtual Machine - JVM).

Η χρήση στρωμάτων αφαίρεσης βοηθά στην κατανόηση της ροής μιας εφαρμογής. Η απεικόνιση των στρωμάτων της εφαρμογής με κέικ (τα στρώματα του κέικ στοιβαγμένα το ένα πάνω στο άλλο) είναι ένας συνήθης και βολικός τρόπος αναπαράστασης της οργάνωσης της εφαρμογής. Η Εικόνα 6-3 επεξηγεί τα συνήθη, πολύ γενικευμένα στρώματα για τις εφαρμογές Ιστού.



Εικόνα 6-3. Γενικευμένα στρώματα μιας εφαρμογής ιστού.

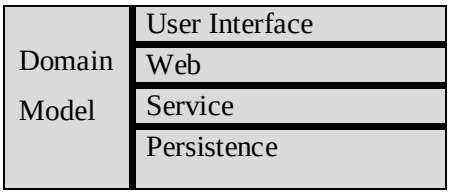
Όλες οι λειτουργίες διατήρησης είναι στο κατώτατο σημείο του κέικ, ενώ η διεπαφή του χρήστη είναι στην κορυφή. Το μεσαίο στρώμα και ο τρόπος οργάνωσής του αποτελούν το αντικείμενο αυτού του κεφαλαίου. Η μεταφορά αυτή θα χρησιμοποιηθεί κατά την ανάλυση της αρχιτεκτονικής.

Σε περαιτέρω ανάλυση, οι τυπικές εφαρμογές Spring MVC έχουν τουλάχιστον πέντε στρώματα αφαιρέσεων που θα υλοποιήσει ο χρήστης. Τα στρώματα αυτά είναι:

- στρώμα διεπαφής χρήστη (user interface layer)
- στρώμα ιστού (web layer)
- στρώμα υπηρεσιών (service layer)
- μοντέλο τομέα (domain object model)

- στρώμα διατήρησης (persistence)

Η Εικόνα 6-4 επεξηγεί πιο συγκεκριμένα τη σχετική τοποθέτηση των διαφορετικών στρωμάτων.



Εικόνα 6-4. Επίπεδα Αφαιρέσεων του αρχιτεκτονικού μοντέλου Spring MVC.

Παρατηρούμε ότι το μοντέλο τομέα εκτείνεται κάθετα σε όλα τα άλλα στρώματα. Αυτό συμβαίνει επειδή όλα τα άλλα στρώματα έχουν μια εξάρτηση από το μοντέλο τομέα. Είναι το μόνο στρώμα που διατρέχει όλα τα υπόλοιπα.

6.3.2 Απομόνωση στρωμάτων

Απομονώνοντας τους προβληματικούς τομείς, όπως η διατήρηση (persistence), η περιήγηση Ιστού, και η διεπαφή χρήστη, σε ξεχωριστά στρώματα δημιουργείται μια ευέλικτη και ελέγξιμη εφαρμογή. Η υλοποίηση κάθε στρώματος θα ποικίλει ανεξάρτητα, αυξάνοντας την ευελιξία της εφαρμογής. Μειώνοντας τη σύζευξη μεταξύ των περιοχών της εφαρμογής θα αυξηθεί η ελεγξιμότητα, καθιστώντας έτσι ευκολότερο τον έλεγχο κάθε μέρους της εφαρμογής ξεχωριστά.

Αυτή η απομόνωση ολοκληρώνεται με την ελαχιστοποίηση των εξαρτήσεων μεταξύ των στρωμάτων. Όσο λιγότερες εξαρτήσεις έχει ένα στρώμα, τόσο λιγότερο δαπανηρή είναι η αλλαγή αυτού του στρώματος. Είναι προτιμότερο να εξασφαλίζεται η εξάρτηση ενός στρώματος μόνο από ένα ή δύο άλλα στρώματα. Καλό είναι να αποφεύγεται η εξάρτηση ενός ενιαίου στρώματος από πολλά διαφορετικά μέρη της εφαρμογής.

6.4 Στρώματα αφαίρεσης σε μια εφαρμογή Spring MVC

Στην ενότητα αυτή αναλύεται κάθε ξεχωριστό στρώμα αφαίρεσης σε μια τυπική εφαρμογή που υιοθετεί το αρχιτεκτονικό μοντέλο Spring MVC καθώς και τυχόν αποκλίσεις που είναι πιθανόν να υπάρξουν από το μοντέλο αυτό.

6.4.1 Στρώμα διεπαφής χρήστη (User Interface)

Το στρώμα διεπαφής χρήστη είναι υπεύθυνο για την παρουσίαση της εφαρμογής στον τελικό χρήστη. Αυτό το στρώμα δίνει την απάντηση (response) που παράγεται από το στρώμα Ιστού στη μορφή που ζητείται από τον πελάτη. Για παράδειγμα, τα κινητά τηλέφωνα απαιτούν συνήθως σελίδες υλοποιημένες με χρήση κώδικα WML, XHTML ή μπορεί να χρησιμοποιηθούν PDFs για την υλοποίηση της διεπαφής με τον χρήστη. Το στρώμα διεπαφής

χρήστη διατηρείται χωριστά από το στρώμα Ιστού έτσι ώστε να μπορεί να επαναχρησιμοποιηθεί το στρώμα Ιστού όσο το δυνατόν περισσότερο.

Από την προοπτική του προγραμματιστή-σχεδιαστή, που είναι υπεύθυνος για την ανάπτυξη εφαρμογών ιστού, το στρώμα διεπαφής χρήστη είναι ένα πολύ σημαντικό επίπεδο αφαίρεσης. Είναι εύκολο να ελεγχθεί η διεπαφή χρήστη ως υπόστρωμα κάτω από το πλήρες στρώμα Ιστού. Για την ανάλυση εφαρμογών ιστού, έχουμε ανυψώσει τη διεπαφή χρήστη σε ένα ξεχωριστό στρώμα επειδή έχει το δικό του σύνολο λειτουργιών και λεπτομερειών εκτέλεσης.

Το στρώμα διεπαφής χρήστη είναι χαρακτηριστικά το υψηλότερο στρώμα. Εννοιολογικά, αυτό σημαίνει ότι είναι το τελευταίο στρώμα στην αλυσίδα επεξεργασίας προτού να σταλούν τα δεδομένα στον πελάτη. Από αυτό το σημείο, όλη η επιχειρησιακή λογική έχει εκτελεσθεί, όλες οι συναλλαγές δεσμεύονται, και όλοι οι πόροι έχουν απελευθερωθεί.

Το στρώμα διεπαφής χρήστη είναι αρμόδιο για την απόδοση των δεδομένων που στέλνονται στον πελάτη. Ο πελάτης, σε μια εφαρμογή Ιστού, είναι απομακρυσμένος και συνδεμένος μέσω ενός αναξιόπιστου δικτύου. Η μεταφορά των δεδομένων μπορεί να επιβραδυνθεί, να επαναληφθεί, ή ακόμα και να σταματήσει. Το στρώμα διεπαφής χρήστη διατηρείται χωριστό από τα άλλα στρώματα επειδή θέλουμε το σύστημα να συνεχίσει να επεξεργάζεται άλλα αιτήματα, με πολύτιμους πόρους όπως οι συνδέσεις βάσεων δεδομένων, χωρίς να χρειάζεται να περιμένουμε τις συνδέσεις δικτύου.

Ένας άλλος λόγος για την απομόνωση της διεπαφής χρήστη στο δικό του στρώμα είναι η ύπαρξη πολλών διαφορετικών εργαλείων για την απόδοση της διεπαφής χρήστη. Μερικά παραδείγματα αποτελούν τα JSP, Velocity, FreeMarker, και XSLT (που υποστηρίζονται από τη Spring). Η τοποθέτηση των διεπαφών χρήστη σε χωριστό στρώμα επιτρέπει την αλλαγή τεχνολογίας προβολής χωρίς να επηρεάζει τα άλλα στρώματα. Τα άλλα στρώματα του συστήματος θα πρέπει να είναι κρυμμένα από την επιλογή οποιουδήποτε εργαλείου προβολής. Υπάρχουν πάρα πολλές επιλογές, η καθεμιά με τα δικά της πλεονεκτήματα και μειονεκτήματα, για τη σύνδεση ενός συγκεκριμένου εργαλείου κατευθείαν μέσα στο σύστημα.

Ομάδες που αναθέτουν τον σχεδιασμό της διεπαφής χρήστη σε συγκεκριμένα μέλη τους επωφελούνται πολύ από τον διαχωρισμό αυτού του στρώματος. Οι σχεδιαστές της διεπαφής χρήστη συνήθως εργάζονται με διαφορετικά εργαλεία και σε ανεξάρτητο τομέα από τους υπεύθυνους για την ανάπτυξη της εφαρμογής. Δουλεύοντας πάνω σε ένα στρώμα που δημιουργήθηκε για τις ανάγκες τους, τους προστατεύει από τις εσωτερικές λεπτομέρειες ενός μεγάλου μέρους του συστήματος. Αυτό είναι ιδιαίτερα σημαντικό κατά τη διάρκεια των αρχικών σταδίων διαμόρφωσης και ανάπτυξης της διεπαφής.

6.4.1.1 Ενδιάμεσο Στρώμα Χρήστη της Spring MVC

Η αρχιτεκτονική Spring MVC βοηθά στην απομόνωση των αρμοδιοτήτων των σχεδιαστών της διεπαφής. Η διεπαφή org.springframework.web.servlet.View αντιπροσωπεύει μια προβολή, ή σελίδα, της εφαρμογής Ιστού. Είναι υπεύθυνη για τη μετατροπή του αποτελέσματος της ζητούμενης λειτουργίας από τον πελάτη (μέσω του μοντέλου) σε μια μορφή που μπορεί να δει και να καταλάβει ο πελάτης.

Το μοντέλο είναι μια συλλογή αντικειμένων. Οποιοδήποτε αντικείμενο που πρόκειται να αποδοθεί στην προβολή τοποθετείται μέσα στο μοντέλο. Το μοντέλο είναι εσκεμμένα γενικό έτσι ώστε να μπορεί να λειτουργήσει με οποιαδήποτε τεχνολογία για τη δημιουργία της προβολής. Τα εργαλεία δημιουργίας της προβολής είναι υπεύθυνα για την απόδοση κάθε αντικειμένου στο μοντέλο.

Η διεπαφή προβολής (view layer) είναι απολύτως γενική και δεν έχει καθορισμένες εξαρτήσεις από την προβολή. Κάθε τεχνολογία προβολής υλοποιεί αυτή τη διεπαφή. Η πλατφόρμα Spring MVC υποστηρίζει τα JSP, FreeMarker, Velocity, XSLT, JasperReports, Excel, και PDF.

Το πακέτο org.springframework.web.servlet.ViewResolver παρέχει ένα χρήσιμο στρώμα διασύνδεσης με την προβολή. Μέσω του ViewResolver παρέχεται η σύνδεση μεταξύ προβολών και των λογικών ονομάτων τους. Για παράδειγμα, μια σελίδα JSP με όνομα αρχείου /WEB-INF/jsp/onSuccess.jsp μπορεί να αναφερθεί μέσω του ονόματος «success». Αυτό αποσυνδέει την πραγματική προβολή από το σημείο του κώδικα που αναφέρεται. Είναι ακόμα δυνατό να συνδεθούν μεταξύ τους πολλαπλοί ViewResolvers για την περαιτέρω δημιουργία μιας ευέλικτης διαμόρφωσης.

6.4.1.2 Εξαρτήσεις

Το στρώμα της προβολής τυπικά έχει μια εξάρτηση από το μοντέλο. Δεν είναι πάντα αυτή η περίπτωση, αλλά συχνά είναι πολύ βολικό να εκτεθεί άμεσα και να αποδοθεί το μοντέλο μέσω της προβολής. Ένα μεγάλο μέρος της ευκολίας χρήσης της πλατφόρμας Spring MVC για την επεξεργασία μιας φόρμας/σελίδας (form) προέρχεται από το γεγονός ότι το αποτέλεσμα της προβολής λειτουργεί άμεσα με το μοντέλο.

Παραδείγματος χάριν, το μοντέλο τυπικά γεμίζει με στιγμιότυπα από το μοντέλο τομέα. Η τεχνολογία της δημιουργίας της προβολής θα αποδώσει έπειτα τις σελίδες με άμεση ανάκτηση των στιγμιότυπων του μοντέλου τομέα.

Ίσως κάποιος να διαφωνεί με αυτόν τον τρόπο ανάπτυξης διότι δημιουργείται χωρίς λόγο μια επιπλέον εξάρτηση με το σύστημα. Όμως μια αντίθετη προσέγγιση θα ήταν πολύ πιο άβολη και το κόστος θα ήταν δυσανάλογο σε σύγκριση με τα πλεονεκτήματα που θα μας παρείχε μια επιπλέον διαχώριση της διεπαφής. Για παράδειγμα το πλαίσιο Struts προωθεί μια

ιεραρχία κλάσεων για το μοντέλο που είναι ξεχωριστό από το μοντέλο τομέα, κάτι που επιφέρει διπλάσια προσπάθεια από μέρους του προγραμματιστή και προκαλεί επιπλέον εξαρτήσεις για το σύστημα.

Για λόγους απλούστευσης, η Spring MVC προωθεί την ενσωμάτωση των κλάσεων τομέα στο αποτέλεσμα της προβολής. Αυτό θεωρείται αποδεκτό, αλλά σε καμία περίπτωση δεν επιβάλλεται ούτε απαιτείται.

6.4.1.3 Σύνοψη

Το στρώμα διεπαφής χρήστη (επίσης γνωστό ως προβολή) είναι υπεύθυνο για την απόδοση του αποτελέσματος στον χρήστη. Τυπικά, αυτό σημαίνει ανάπτυξη κώδικα XHTML για μια εφαρμογή ιστού. Η Spring MVC υποστηρίζει πολλές διαφορετικές τεχνολογίες προβολής και για κείμενο και για δυαδική έξοδο. Οι βασικές διεπαφές είναι:

- org.springframework.web.servlet.View (που αντιπροσωπεύει μια μόνο σελίδα) και
- org.springframework.web.servlet.ViewResolver (που παρέχει μια χαρτογράφηση μεταξύ όψεων και λογικών ονομάτων).

6.4.2 Στρώμα Ιστού (Web Layer)

Η λογική περιήγησης είναι μια από τις δύο σημαντικές λειτουργίες που εκτελούνται από το στρώμα Ιστού. Είναι αρμόδιο για την οδήγηση του χρήστη μέσω των σωστών σελίδων στη σωστή σειρά. Αυτό μπορεί να είναι τόσο απλό όπως η απεικόνιση ενός μόνο URL σε μια μόνο σελίδα ή τόσο σύνθετο όσο ένας μηχανισμός πλήρους ροής εργασιών. Η διαχείριση του χρήστη μέσω μιας ιστοσελίδας είναι μια μοναδική ευθύνη του στρώματος Ιστού. Το στρώμα Ιστού τυπικά περιέχει βοήθεια στην καθοδήγηση του χρήστη μέσω του σωστού μονοπατιού.

Τυπικά δεν υπάρχει οποιαδήποτε λογική περιήγησης στο μοντέλο τομέα ή το στρώμα υπηρεσιών. Η λογική περιήγησης είναι αρμοδιότητα μόνο του στρώματος Ιστού. Αυτό δημιουργεί ένα πιο εύκαμπτο σχέδιο, επειδή οι μεμονωμένες λειτουργίες του μοντέλου τομέα μπορούν να συνδυαστούν με πολλούς διαφορετικούς τρόπους για τη καθοδήγηση του χρήστη μέσα στον ιστότοπο (site).

Η δεύτερη κύρια λειτουργία του στρώματος Ιστού είναι να παρέχει την σύνδεση μεταξύ του στρώματος υπηρεσιών και του κόσμου του HTTP. Αποτελεί ένα λεπτό στρώμα που εξουσιοδοτεί το στρώμα υπηρεσιών για όλο το συντονισμό της επιχειρησιακής λογικής. Το στρώμα Ιστού ασχολείται με παραμέτρους αιτήματος, διαχείριση συνεδρίας (session HTTP), κώδικες απάντησης HTTP, και γενικά την αλληλεπίδραση με το Servlet API.

Ο κόσμος HTTP είναι φορτωμένος με παραμέτρους αιτήματος, επικεφαλίδες HTTP, και cookies. Αυτά τα στοιχεία δεν είναι ορισμένα στην επιχειρησιακή λογική, και έτσι διατηρούνται απομονωμένα από το στρώμα υπηρεσιών. Το στρώμα Ιστού αντίστοιχα κρύβει τις λεπτομέρειες του Ιστού από την επιχειρησιακή λογική.

Μετακινώντας τις ευθύνες Ιστού από την επιχειρησιακή λογική καθίσταται πολύ εύκολος ο έλεγχός της, χωρίς να υπάρχει ανησυχία για τον καθορισμό των μεταβλητών αιτήματος, των session μεταβλητών, των κωδίκων απάντησης HTTP και άλλων. Αντίστοιχα, κατά τον έλεγχο του στρώματος Ιστού, αμελείται το επιχειρησιακό στρώμα και ο έλεγχος σε ζητήματα όπως οι παράμετροι αιτήματος.

Το στρώμα Ιστού μπορεί να εφαρμοστεί τόσο απλά όσο, για παράδειγμα, τα servlets. Τα servlets θα εκτελέσουν την εργασία της μετατροπής των παραμέτρων αιτήματος σε αντικείμενα που αντιλαμβάνεται το στρώμα υπηρεσιών, και έπειτα θα γίνει η κλήση της κατάλληλης μεθόδου στη διεπαφή υπηρεσιών. Το στρώμα Ιστού είναι επίσης αρμόδιο, μεταξύ άλλων, για τη μετατροπή οποιωνδήποτε επιχειρησιακών εξαιρέσεων σε κατάλληλα μηνύματα λάθους για τον τελικό χρήστη.

Τα πλαίσια υψηλού επιπέδου όπως το Spring MVC και Tapestry προσφέρουν εκλεπτυσμένους μηχανισμούς για τη μετάφραση των παραμέτρων μεταξύ του αιτήματος και του στρώματος της επιχειρησιακής λογικής. Για παράδειγμα στη Spring MVC οι παράμετροι του αιτήματος απεικονίζονται σε αντικείμενα java (POJOs Plain Old Java Objects) τα οποία μπορεί να διαχειριστεί άμεσα η επιχειρησιακή λογική.

6.4.2.1 Εξαρτήσεις

Το στρώμα Ιστού εξαρτάται από το στρώμα υπηρεσιών και το μοντέλο τομέα. Το στρώμα Ιστού αναθέτει την επεξεργασία του στο στρώμα υπηρεσιών και είναι υπεύθυνο για τη μετατροπή των πληροφοριών που αποστέλλονται από τον Ιστό σε αντικείμενα τομέα ώστε να είναι δυνατές περαιτέρω κλήσεις μεθόδων στο στρώμα υπηρεσιών.

6.4.2.2 Στρώμα Ιστού Spring MVC

Η πλατφόρμα Spring MVC παρέχει μια διεπαφή και μια πολύ πλούσια τάξη ιεραρχίας μέσω του πακέτου org.springframework.web.servlet.mvc.Controller που αντιστοιχεί στον ελεγκτή. Ο ελεγκτής είναι υπεύθυνος για την αποδοχή ενός αιτήματος ή μιας απάντησης μέσω του HttpServletRequest και του HttpServletResponse, εκτελώντας κάποια εργασία και διαβιβάζοντας τον έλεγχο στην Προβολή. Ο ελεγκτής δηλαδή μοιάζει πολύ με ένα τυπικό servlet.

Όταν ένας ελεγκτής θέλει να επιστρέψει πληροφορίες στον πελάτη, φορτώνει το Μοντέλο και την Προβολή (ModelAndView). Το ModelAndView εσωκλείει δύο τμήματα πληροφοριών: το μοντέλο ως απόκριση, το οποίο είναι τα δεδομένα που αποτελούν την απάντηση, και μια Προβολή αναφοράς (ή το όνομα αναφοράς Προβολής) για να ανατρέξει μέσω του ViewResolver.

6.4.2.3 Σύνοψη

Το στρώμα Ιστού διαχειρίζεται την περιήγηση του χρήστη μέσω του ιστοτόπου της εφαρμογής. Ενεργεί επίσης ως συνδετικός κρίκος μεταξύ του στρώματος υπηρεσιών και των λεπτομερειών του Servlet API.

Η πλατφόρμα Spring MVC παρέχει μια πλούσια βιβλιοθήκη εφαρμογών της διεπαφής Ελεγκτών. Για τις πολύ σύνθετες ροές εργασίας (work flows), η Spring Web Flow χτίζει ένα ισχυρό μηχανισμό για να διαχειριστεί την περιήγηση του χρήστη.

6.4.3 Στρώμα υπηρεσιών (Service Layer)

Το στρώμα υπηρεσιών παίζει πολύ σημαντικό ρόλο τόσο για τον πελάτη όσο και το σύστημα. Για τον πελάτη, εκθέτει και περιέχει την λειτουργικότητα του συστήματος (περιπτώσεις χρήσης-Use cases) για εύκολη χρήση από αυτόν. Μια μέθοδος που αφορά τη λειτουργικότητα είναι υψηλού επιπέδου, εμπεριέχοντας μια ευρεία ροή εργασίας και προστατεύοντας τον πελάτη από πολλές μικρές αλληλεπιδράσεις με το σύστημα. Το στρώμα υπηρεσιών πρέπει να είναι ο μόνος τρόπος που ένας πελάτης μπορεί να αλληλεπιδράσει με το σύστημα, διατηρώντας τη σύζευξη σε χαμηλό επίπεδο ώστε ο πελάτης να προστατεύεται από όλες τις αλληλεπιδράσεις με τα αντικείμενα (POJO) που υλοποιούν τις περιπτώσεις χρήσης.

Για το σύστημα, οι μέθοδοι του στρώματος υπηρεσιών αντιπροσωπεύουν τις βασικές λειτουργίες κάθε συναλλαγής. Αυτό σημαίνει ότι με μια κλήση μεθόδου, πολλά αντικείμενα POJOs και οι αλληλεπιδράσεις τους θα λειτουργήσουν κάτω από μια ενιαία συναλλαγή. Η εκτέλεση όλης της εργασίας μέσα στο στρώμα υπηρεσιών περιορίζει την επικοινωνία μεταξύ του πελάτη και του συστήματος στο ελάχιστο (στην πραγματικότητα, μέσα σε μία ενιαία κλήση). Μετακινώντας τις συναλλαγές προς έναν ενιαίο στρώμα καθίσταται εύκολη η παραμετροποίηση και η διαμόρφωση των συναλλαγών.

Κάθε μέθοδος στο στρώμα υπηρεσιών πρέπει να είναι stateless. Δηλαδή κάθε αλλεπάλληλη κλήση σε μια μέθοδο υπηρεσιών από τον ίδιο πελάτη εκλαμβάνεται ως νέα διαδικασία, μη συσχετιζόμενη με τυχόν προηγούμενες. Οτιδήποτε παραμένει σταθερό στις κλήσεις μεθόδου φυλάσσεται στο μοντέλο τομέα.

Σε μια τυπική εφαρμογή Spring MVC, ένα ενιαίο αντικείμενο στρώματος υπηρεσιών θα χειριστεί πολλά ταυτόχρονα νήματα (threads) εκτέλεσης. Παραμένοντας σε κατάσταση «stateless» είναι ο μόνος τρόπος να αποφευχθεί η καταστολή ενός νήματος από ένα άλλο. Αυτό το στρώμα προσπαθεί να ενσωματώσει όλες τις περιπτώσεις χρήσης του συστήματος και καθιστά εύκολη την αναφορά, σε ένα μοναδικό στρώμα, για όλη την υψηλού επιπέδου λειτουργικότητα του συστήματος

Η υλοποίηση των επιχειρησιακών λειτουργιών της εφαρμογής πίσω από ένα στρώμα υπηρεσιών δημιουργεί ένα ενιαίο σημείο εισόδου μέσα στο σύστημα για τον τελικό χρήστη

και τους πελάτες. Με αυτό τον τρόπο γίνεται πολύ εύκολη η σύνδεση πολλαπλών μηχανισμών επικοινωνίας πελατών με μια ενιαία διεπαφή υπηρεσιών. Για παράδειγμα, με τις ικανότητες διαχείρισης απομακρυσμένων κλήσεων (remoting) της Spring, είναι δυνατόν να εκτεθεί η ίδια υπηρεσία μέσω SOAP, RMI, Java serialization μέσω HTTP και, φυσικά, XHTML. Αυτό προωθεί την επαναχρησιμοποίηση κώδικα και την πολύ σημαντική αρχή μη επανάληψης κώδικα (DRY: Don't Repeat Yourself) με την αποσύνδεση της εργασίας συναλλαγών με τον τρόπο μεταφοράς των δεδομένων ή το στρώμα διεπαφής χρήστη.

6.4.3.1 Εξαρτήσεις

Το στρώμα υπηρεσιών εξαρτάται από το μοντέλο τομέα και το στρώμα διατήρησης, τα οποία παρουσιάζονται στις επόμενες ενότητες. Συνδυάζει και συντονίζει τις κλήσεις στα αντικείμενα πρόσβασης δεδομένων (DAO) και στα αντικείμενα του Μοντέλου. Το στρώμα υπηρεσιών δεν πρέπει ποτέ να έχει εξάρτηση από την όψη ή τα στρώματα Ιστού.

Είναι σημαντικό να σημειωθεί ότι είναι συνήθως περιττό για αυτό το στρώμα να έχει οποιεσδήποτε εξαρτήσεις από τον κώδικα που ορίζεται από το πλαίσιο (framework) ανάπτυξης της εφαρμογής, ή τον κώδικα υποδομής (infrastructure).

6.4.3.2 Υποστήριξη Spring για το στρώμα υπηρεσιών

Το πλαίσιο Spring υποστηρίζει οποιεσδήποτε διεπαφές (interfaces) ή κλάσεις (classes) για την εφαρμογή των επιχειρησιακών πτυχών του στρώματος υπηρεσιών (service layer). Αυτό δεν πρέπει να αποτελεί έκπληξη, αφού το στρώμα υπηρεσιών είναι συγκεκριμένο για την εφαρμογή.

Αντί του καθορισμού των επιχειρησιακών διεπαφών, η Spring βοηθά με ένα πρότυπο μοντέλο προγραμματισμού. Χαρακτηριστικά, το ApplicationContext του πλαισίου Spring εισάγει στιγμιότυπα της υπηρεσίας στους Ελεγκτές (Controllers) Ιστού. Η Spring ενισχύει επίσης το στρώμα υπηρεσιών με υπηρεσίες όπως η διαχείριση συναλλαγών (transaction management) και ελέγχου της εκτέλεσής τους (performance monitoring).

6.4.3.3 Σύνοψη

Το στρώμα υπηρεσιών παρέχει μια stateless διεπαφή στους πελάτες για την αλληλεπίδραση με το σύστημα. Κάθε μέθοδος στο στρώμα υπηρεσιών αντιπροσωπεύει χαρακτηριστικά μια περίπτωση χρήσης και αποτελεί μια ενιαία μονάδα συναλλαγής εργασίας.

Η χρησιμοποίηση ενός στρώματος υπηρεσιών διατηρεί σε χαμηλό επίπεδο τη σύνδεση μεταξύ του συστήματος και του πελάτη. Μειώνει τον αριθμό των κλήσεων που απαιτούνται για μια περίπτωση χρήσης και καθιστά το σύστημα απλούστερο στη χρήση. Σε ένα απομακρυσμένο περιβάλλον, αυτό βελτιώνει εντυπωσιακά την απόδοση.

6.4.4 Στρώμα Μοντέλου Τομέα (Domain Model Layer)

Το μοντέλο αντικειμένου τομέα είναι το σημαντικότερο στρώμα στο σύστημα. Αυτό το στρώμα περιέχει την επιχειρησιακή λογική του συστήματος, και συνεπώς, την εφαρμογή των περιπτώσεων χρήσης. Το μοντέλο τομέα είναι η συλλογή των ουσιαστικών στοιχείων στο σύστημα, υλοποιημένα ως αντικείμενα POJOs. Στοιχεία, όπως λογαριασμός χρήστη, διεύθυνση, και κάρτα αγοράς (ShoppingCart), περιέχουν και τη κατάσταση (όνομα χρήστη, επώνυμο χρήστη) και τη συμπεριφορά (shoppingCart.purchase()). Με τη συγκέντρωση της επιχειρησιακής λογικής μέσα σε POJOs μπορούν να εφαρμοστούν οι αντικειμενοστραφείς αρχές και πρακτικές, όπως ο πολυμορφισμός και η κληρονομικότητα.

Όταν λέμε επιχειρησιακή λογική εννοούμε οποιαδήποτε λογική εκτελεί το σύστημα για να ικανοποιήσει κάποιο κανόνα ή περιορισμό που υπαγορεύεται από τον πελάτη. Αυτό μπορεί να περιλαμβάνει οτιδήποτε, από τη σύνθετη επαλήθευση κατάστασης στους απλούς κανόνες επικύρωσης, ή ακόμα και μια φαινομενικά απλή εφαρμογή δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής (CRUD Create, Read, Update, and Delete) όπου θα έχει κάποιο επίπεδο επιχειρησιακής λογικής υπό μορφή περιορισμών σε μια βάση δεδομένων.

Νωρίτερα, υποστηρίξαμε ότι το μοντέλο τομέα πρέπει να εμπεριέχει την επιχειρησιακή λογική του συστήματος. Ακόμα έχουμε αναγνωρίσει ότι υπάρχουν μερικοί επιχειρησιακοί κανόνες που ζουν στη βάση δεδομένων, υπό μορφή περιορισμών όπως UNIQUE (μοναδικότητα) ή NOT NULL (υποχρεωτικά συμπληρωμένο). Ενώ πρέπει να προσπαθήσουμε να τοποθετήσουμε όλη την επιχειρησιακή λογική μέσα στο μοντέλο τομέα, υπάρχουν περιπτώσεις όπου μέρος της λογικής θα υπάρχει και σε άλλα μέρη της εφαρμογής. Θεωρούμε αυτή τη διάσπαση αποδεκτή, επειδή η βάση δεδομένων κάνει καλή δουλειά στην επιβολή των παραπάνω περιορισμών. Μπορούμε, εντούτοις, να συνεχίσουμε να εκφράζουμε τον επιχειρησιακό κανόνα που βρίσκεται στη βάση δεδομένων στο μοντέλο τομέα. Με αυτόν τον τρόπο, ο κανόνας δεν θα είναι κρυμμένος.

Παραδείγματος χάριν, ας υποθέσουμε ότι όλα τα ηλεκτρονικά μηνύματα ταχυδρομείου πρέπει να είναι μοναδικά στο σύστημα. Θα μπορούσαμε να υλοποιήσουμε αυτήν την λογική στο μοντέλο τομέα με το να φορτώσουμε πρώτα όλους τους χρήστες και να ερευνήσουμε τα mail του καθενός. Η απόδοση με αυτόν τον τρόπο ελέγχου, θα ήταν πολύ χαμηλή. Η βάση δεδομένων αντίθετα μπορεί εύκολα να χειριστεί αυτό το είδος της απαίτησης μοναδικότητας στοιχείων. Από το παράδειγμα αυτό συμπεραίνουμε ότι το μοντέλο τομέα πρέπει να περιέχει το μεγαλύτερο μέρος της επιχειρησιακής λογικής, αλλά τοποθετούμε μέρος της λογικής έξω από το μοντέλο όταν τίθεται θέμα καλύτερης απόδοσης.

Είναι σημαντικό να σημειωθεί ότι η επιχειρησιακή λογική δεν σημαίνει απλά ένα ισχυρό σύνολο σχέσεων μεταξύ των αντικειμένων. Αυτό συμβαίνει όταν υπάρχει ένα πλούσιο μοντέλο τομέα που δεν φαίνεται να εκτελεί οποιαδήποτε εργασία. Ίσως είναι δελεαστική η

τοποθέτηση όλης της επιχειρησιακής λογικής στο στρώμα υπηρεσιών ή ακόμα και το στρώμα Ιστού, αλλά με αυτόν τον τρόπο αφήνουμε ανεκμετάλλευτα οποιαδήποτε οφέλη μπορεί να μας αποφέρει ένα αντικειμενοστραφές σύστημα.

6.4.4.1 Εξαρτήσεις

Το μοντέλο τομέα, ή αλλιώς μοντέλο επιχειρησιακού αντικειμένου, είναι ένα καλό παράδειγμα ενός στρώματος που διαπερνά κατά ένα μεγάλο μέρος τα άλλα στρώματα. Είναι χρήσιμο να σκεφτόμαστε το μοντέλο τομέα ως κάθετο στρώμα. Με άλλα λόγια, πολλά άλλα στρώματα έχουν εξαρτήσεις από το μοντέλο τομέα. Είναι σημαντικό να σημειωθεί, εντούτοις, ότι το αντικείμενο δεν έχει καμία εξάρτηση από οποιοδήποτε άλλο στρώμα.

Το μοντέλο τομέα δεν πρέπει ποτέ να έχει οποιεσδήποτε εξαρτήσεις από το πλαίσιο ανάπτυξης της εφαρμογής, έτσι ώστε να μπορεί να αποσυνδεθεί από το περιβάλλον που θα φιλοξενηθεί. Πρώτα απ' όλα, αυτό σημαίνει ότι η επιχειρησιακή λογική μπορεί να εξεταστεί έξω από τον υποδοχέα (container) και ανεξάρτητα του πλαισίου. Αυτό επιταχύνει παρά πολύ την ανάπτυξη, διότι δεν απαιτείται καμία επέκταση για τη δοκιμή του κώδικα. Η ανάπτυξη δοκιμών ελέγχου (Unit tests) γίνεται πιο εύκολη, δεδομένου ότι αυτές εξετάζουν τον απλό κώδικα Java, χωρίς οποιαδήποτε εξάρτηση από τις συνδέσεις βάσεων δεδομένων, τα πλαίσια Ιστού, ή τα άλλα στρώματα στο σύστημα.

Όλα τα άλλα στρώματα έχουν μια εξάρτηση από το μοντέλο τομέα. Για παράδειγμα, το στρώμα υπηρεσιών τυπικά συνδυάζει τις πολλαπλές μεθόδους από το μοντέλο τομέα ώστε να τρέχει κάτω από μια συναλλαγή. Το στρώμα διεπαφής χρήστη μπορεί να σειριοποιεί το μοντέλο τομέα για έναν πελάτη μέσω ενός XML ή XHTML. Το στρώμα πρόσβασης στοιχείων είναι υπεύθυνο για την διατήρηση και την ανάκτηση των στιγμιότυπων των αντικειμένων από το μοντέλο.

Διαπιστώνουμε ότι κάθε στρώμα είναι υπεύθυνο για τις αρμοδιότητες του τομέα που ανήκει, αλλά όλα μαζί υπάρχουν για τη συντήρηση του μοντέλου τομέα. Η ανάπτυξη των εφαρμογών Ιστού με την Spring MVC είναι εύκολη, επειδή υποστηρίζει σε μεγάλο βαθμό την αντικειμενοστραφή ανάπτυξη.

6.4.4.2 Υποστήριξη Spring για το στρώμα Μοντέλου

Ακριβώς όπως το στρώμα υπηρεσιών, η Spring δεν παρέχει κάποια βασική διεπαφή για το μοντέλο αντικειμένου. Αυτό θα ήταν εντελώς ενάντια στις ιδεολογίες ενός ελαφριού πλαισίου όπως η Spring. Εντούτοις, η Spring παρέχει ορισμένες διεπαφές ευκολίας που κάποιος μπορεί να επιλέξει για να χρησιμοποιήσει όταν είναι απαραίτητο. Η ανάγκη για αυτό είναι σπάνια, και καλό είναι να αποφεύγεται η εισαγωγή οποιωνδήποτε διεπαφών πλαισίου στο βασικό μοντέλο αντικειμένων. Για τη Spring, το στρώμα υπηρεσιών και το μοντέλο τομέα είναι απλά ένα σύνολο από java αντικείμενα (POJOs).

6.4.4.3 Στρώμα Πρόσβασης Δεδομένων (Data Access Layer)

Το στρώμα πρόσβασης δεδομένων λειτουργεί ως διεπαφή με το μηχανισμό διατήρησης ώστε να αποθηκεύει και να ανακτά τα στιγμιότυπου μοντέλου αντικειμένου. Οι χαρακτηριστικές μέθοδοι δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής δεδομένων εφαρμόζονται από αυτό το στρώμα.

Η λειτουργία πρόσβασης στοιχείων έχει το δικό της στρώμα για δύο λόγους. Πρώτον για να προστατεύει το σύστημα από την οποιαδήποτε μελλοντική αλλαγή που μπορεί να πραγματοποιηθεί και δεύτερον να επιτρέπει τη γρήγορη εκτέλεση των δοκιμών ελέγχου.

Ένας από τους αρχικούς λόγους για την χρήση αφαίρεσης στα αντικειμενοστραφή συστήματα είναι να απομονωθεί κάθε στρώμα της εφαρμογής από την ενδεχόμενη αλλαγή που θα συμβεί στο ίδιο ή κάποιο γειτονικό στρώμα. Η λειτουργία πρόσβασης στοιχείων δεν είναι διαφορετική και έχει ως σκοπό να απομονώσει το σύστημα από αλλαγές στους μηχανισμούς ανάκτησης ή αποθήκευσης δεδομένων.

Για παράδειγμα, μια αλλαγή επιχειρησιακής απαίτησης μπορεί να αναγκάσει όλους τους λογαριασμούς χρηστών να αποθηκεύονται μέσα σε έναν LDAP server αντί μιας σχεσιακής βάσης δεδομένων. Με την αφαίρεση των διαδικασιών διατήρησης πίσω από μια ενιαία διεπαφή, κάθε αλλαγή έχει χαμηλό αντίκτυπο για το σύστημα.

Ένα πιθανότερο σενάριο είναι μια αλλαγή στην εφαρμογή και τις βιβλιοθήκες του στρώματος πρόσβασης στοιχείων. Πολλοί διαφορετικοί τύποι εφαρμογών είναι διαθέσιμοι, από την Java Database Connectivity (JDBC) έως τα ολοκληρωμένα πλαίσια απεικόνισης αντικειμένων όπως το Hibernate. Κάθε ένα προσφέρει τα δικά του πλεονεκτήματα, αλλά λειτουργεί με σημαντικά διαφορετικούς τρόπους. Όταν η πρόσβαση για όλα τα στοιχεία μεταβιβάζεται σε ένα μοναδικό στρώμα, η αλλαγή από έναν μηχανισμό διατήρησης σε άλλον είναι δυνατή χωρίς ιδιαίτερο κόστος.

Η οικοδόμηση του συστήματος για την αντιμετώπιση μιας μελλοντικής αλλαγής είναι σημαντική. Καταρχάς είναι σημαντική η αναγνώριση ενός ιδιαίτερου τομέα που μπορεί να δημιουργήσει πρόβλημα στο σύστημα σε περίπτωση κάποιας μεταβολής. Η απομόνωση εκείνων των περιοχών σε δικές τους διεπαφές και στρώματα βοηθά στην εύκολη προσαρμογή του συστήματος.

Η διατήρηση χαμηλού κόστους απαίτησης χρόνου ώστε να τρέξουν οι δοκιμές συστημάτων είναι ο δεύτερος βασικός λόγος που το στρώμα πρόσβασης στοιχείων είναι απομονωμένο. Κάθε εκτέλεση ελέγχου (unit test) εφαρμόζεται συνήθως σε μικρές δομικές μονάδες κώδικα, όπως για παράδειγμα σε μια συγκεκριμένη μέθοδο. Εάν οι έλεγχοι έπρεπε να στηριχθούν σε μια εξωτερική βάση δεδομένων, ο κώδικας υπό δοκιμή, κατόπιν, δεν θα απομονωνόταν. Εάν

ένα πρόβλημα προέκυπτε, θα έπρεπε να ελεγχθούν για προβλήματα και η εξωτερική βάση δεδομένων και ο πραγματικός κώδικας.

Οι συνδέσεις βάσεων δεδομένων είναι ακριβοί πόροι για να δημιουργηθούν και να διατηρηθούν. Οι έλεγχοι μονάδων πρέπει να εκτελούνται γρήγορα, και θα επιβραδυνθούν παρά πολύ εάν απαιτούν συνδέσεις στο σύστημα πρόσβασης δεδομένων (RDBMS). Η απομόνωση όλων των διαδικασιών διατήρησης σε ένα στρώμα το καθιστά εύκολο να διατηρήσει τις δοκιμαστικές λειτουργίες γρήγορες και αποδοτικές.

6.4.4.4 Εξαρτήσεις

Είναι σημαντικό να σημειωθεί ότι, τυπικά, μόνο το στρώμα υπηρεσιών έχει μια εξάρτηση από το στρώμα πρόσβασης δεδομένων. Αυτό σημαίνει ότι υπάρχει μόνο ένα στρώμα που ξέρει οτιδήποτε για το στρώμα πρόσβασης στοιχείων. Υπάρχουν δύο λόγοι για αυτό.

Το στρώμα υπηρεσιών θέτει τα όρια των συναλλαγών. Το στρώμα πρόσβασης στοιχείων ενδιαφέρεται μόνο για την αλληλεπίδραση με το μηχανισμό διατήρησης, και ο μηχανισμός διατήρησης τυπικά αφορά τις συναλλαγές. Εκτελώντας τις διαδικασίες πρόσβασης δεδομένων μέσα στα πλαίσια της υπηρεσίας σημαίνει ότι η τρέχουσα συναλλαγή διαδίδεται εύκολα στο στρώμα πρόσβασης δεδομένων.

Επίσης, το στρώμα υπηρεσιών ενσωματώνει πολλές μικρές διαδικασίες από τα POJOs, προστατεύοντας συνεπώς τον πελάτη από τις εσωτερικές εργασίες του συστήματος. Από μια πρακτική σκοπιά, το στρώμα υπηρεσιών συντονίζει το στρώμα πρόσβασης δεδομένων και το στρώμα του μοντέλου, έτσι ώστε να φορτώνονται τα κατάλληλα POJOs για την περίπτωση χρήσης.

Είναι σημαντικό να σημειωθεί ότι δεν είναι απαραίτητο να υπάρχει πρόσβαση στο στρώμα διατήρησης πίσω από το στρώμα υπηρεσιών. Ο κώδικας διατήρησης είναι ακριβώς ένα άλλο σύνολο JavaBeans. Για τις πολύ απλές εφαρμογές, η άμεση πρόσβαση στα Data Access Objects (DAOs) δεν είναι απαραιτήτως ανεπιθύμητη.

6.4.4.5 Διεπαφή πρόσβασης δεδομένων του Spring Framework

Το πλαίσιο Spring δεν έχει μια ενιαία κοινή διεπαφή για όλες τις διαδικασίες πρόσβασης δεδομένων, επειδή κάθε κουτί εργαλείων (Hibernate, JDBC, iBATIS, και ούτω καθεξής) είναι πολύ διαφορετικό. Οι επιχειρησιακές ανάγκες θα υπαγορεύσουν συνήθως το σχέδιο των διεπαφών. Εντούτοις, η Spring παρέχει τα κοινά πρότυπα για την αλληλεπίδραση με το στρώμα πρόσβασης δεδομένων. Παραδείγματος χάριν, χρησιμοποιείται συχνά ένα κοινό template από τις διαδικασίες πρόσβασης δεδομένων για να προστατεύσει τον προγραμματιστή από τον κοινό κώδικα έναρξης και καθαρισμού (cleanup code). Οι εφαρμογές προτύπων για Hibernate (HibernateTemplate), JDBC (JdbcTemplate), iBATIS

(SqlMapTemplate), και άλλα βρίσκονται μέσα στο πακέτο εργαλείων της Spring (org.springframework.jdbcand org.springframework.ormpackages).

Ένα από τα κύρια οφέλη της Spring είναι η πολύ πλούσια και βαθιά ιεραρχία εξαιρέσεων (exceptions) κατά τη πρόσβαση δεδομένων. Το πλαίσιο μπορεί να μετατρέψει τις συγκεκριμένες εξαιρέσεις της βάσης δεδομένων του server σε μια σημασιολογικά πλούσια εξαίρεση, κοινή σε όλες τις εφαρμογές βάσεων δεδομένων. Για παράδειγμα, το κρυπτογραφικό μήνυμα της βάσης δεδομένων του server "Error 223—Foreign Key Not Found" θα μετατραπεί σε ένα ευανάγνωστο μήνυμα «DataIntegrityViolationException». Όλες οι εξαιρέσεις στην ιεραρχία (DataAccessExceptionhierarchy) είναι τύπου RuntimeException και βοηθούν στη διατήρηση ενός καθαρού κώδικα. Αυτές οι εξαιρέσεις απεικονίζονται συνήθως και στις βάσεις δεδομένων των servers καθώς επίσης και στους μηχανισμούς διατήρησης. Δηλαδή τα HibernateTemplate, JdbcTemplate, και άλλα θα δώσουν το ίδιο σύνολο εξαιρέσεων.

Οι διαδικασίες σε αυτήν την διεπαφή δεν αναφέρουν οποιαδήποτε τεχνολογία διατήρησης. Αυτό βοηθάει στη διατήρηση της ένωσης χαμηλά, επειδή οι πελάτες του στρώματος πρόσβασης στοιχείων θα αλληλεπιδρούν μέσω αυτής της διεπαφής. Οι πελάτες δεν ενδιαφέρονται για το πώς το στοιχείο διατηρείται ή ανακτάται, και η διεπαφή τους προστατεύει από οποιεσδήποτε αλλαγές στις τεχνολογίες.

6.4.5 Σύνοψη

Μια χαρακτηριστική Spring MVC εφαρμογή έχει πολλά στρώματα. Απαιτείται η συγγραφή κώδικα για τη διαχείριση της διεπαφής χρήστη, την περιήγηση Ιστού, το στρώμα υπηρεσιών, το μοντέλο τομέα, και το στρώμα διατήρησης. Κάθε στρώμα είναι απομονωμένο με τέτοιο τρόπο ώστε να μειώνει τη σύζευξη και να αυξάνει την ελεγξιμότητα. Τα στρώματα χρησιμοποιούν τις διεπαφές σαν όρια, που προστατεύουν άλλα στρώματα από τις λεπτομέρειες της εκτέλεσης. Αυτό επιτρέπει στα στρώματα να αλλάξουν ανεξάρτητα από το υπόλοιπο σύστημα. Το σύστημα γίνεται πιο ελέγξιμο, αφού κάθε στρώμα μπορεί να εξεταστεί απομονωμένα. Τα άλλα στρώματα αγνοούνται από τους ελέγχους μονάδων (unit test), που κρατούν τις δοκιμαστικές λειτουργίες γρήγορες και στρέφονται στη δοκιμή μόνο συγκεκριμένου κώδικα.

Έχουμε δείξει ότι το σημαντικότερο στρώμα είναι το μοντέλο αντικειμένου. Το μοντέλο αντικειμένου περιέχει την επιχειρησιακή λογική του συστήματος. Όλα τα άλλα στρώματα διαδραματίζουν ενισχυτικούς ρόλους και χειρίζονται διαδικασίες συστήματος όπως διατήρηση ή συναλλαγές. Το στρώμα Ιστού διατηρείται απλό, χωρίς να εφαρμόζει κάποια επιχειρησιακή λογική και παρέχει μια γέφυρα μεταξύ του κόσμου του Ιστού και του μοντέλου αντικειμένου.

Με τη χρησιμοποίηση του πλαισίου Spring, είμαστε σε θέση να κρατήσουμε τον κώδικα που ορίζεται από το πλαίσιο εντελώς έξω από το μοντέλο αντικειμένου. Οι διεπαφές στο στρώμα πρόσβασης δεδομένων δεν εξαρτώνται από οποιοδήποτε πλαίσιο. Οι διεπαφές του στρώματος προβολής υπηρεσιών είναι επίσης απαλλαγμένες από την οποιαδήποτε επιλογή πλαισίου ανάπτυξης. Το πλαίσιο Spring δεσμεύει τα στρώματά μαζί διαφανώς, χωρίς να υπαγορεύει τον σχεδιασμό της εφαρμογής ή τον τρόπο υλοποίησής της.

7

Ανάπτυξη Συστήματος Ηλεκτρονικής Διαχείρισης Αδειών με την χρήση του Μοντέλου Αιτήσεων - Εγκρίσεων

Στο κεφάλαιο αυτό παρουσιάζεται ο τρόπος υλοποίησης της επιχειρησιακής λογικής του συστήματος, η οποία αναπτύχθηκε σε προηγούμενη ενότητα καθώς και η υλοποίηση επιμέρους λειτουργιών που υποστηρίζει η εφαρμογή. Η υλοποίηση της εφαρμογής βασίστηκε στις πλατφόρμες (frameworks) J2EE και Spring κάνοντας χρήση επιλεγμένων συστατικών λογισμικού για την υλοποίηση κάθε λειτουργίας. Για παράδειγμα η υλοποίηση του μέρους της εφαρμογής που αφορά τη διαχείριση των χρηστών (users) έγινε με χρήση των EJBs της πλατφόρμας J2EE, ενώ για τη διαχείριση των συναλλαγών της εφαρμογής με τη βάση δεδομένων χρησιμοποιήθηκε ο TransactionManager της Spring σε συνδυασμό με το JDBC template. Θα ξεκινήσουμε με τον τρόπο υλοποίησης του αρχιτεκτονικού μοντέλου MVC.

7.1 Υλοποίηση του στρώματος Μοντέλου (Model Layer)

Εδώ «κρύβεται» σχεδόν ολόκληρη η επιχειρησιακή λογική της εφαρμογής. Το μοντέλο είναι ένα σύνολο τριών άλλων συνθετημάτων:

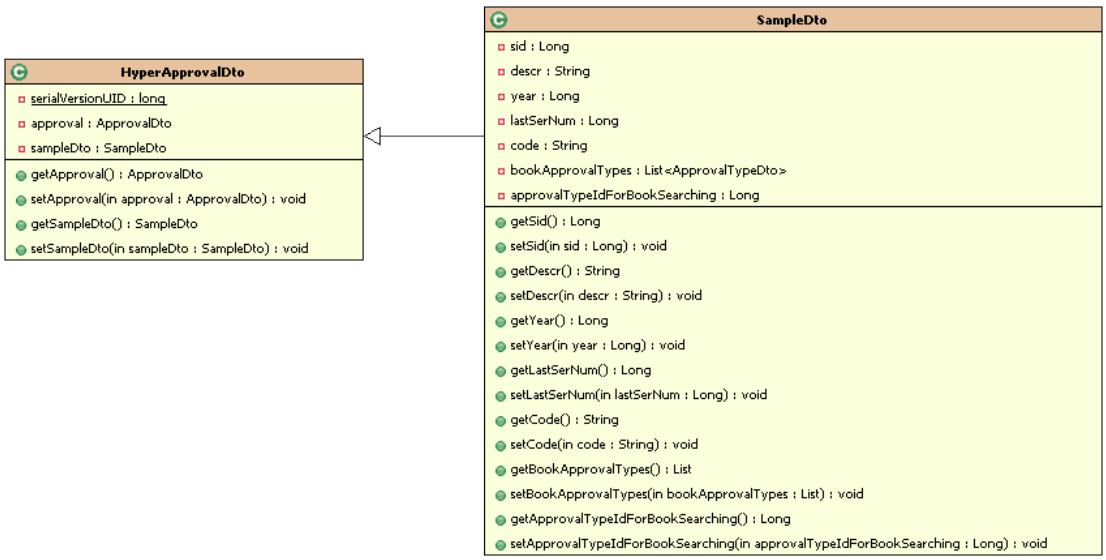
- Τα αντικείμενα μεταφοράς δεδομένων (DTO)

- Τα αντικείμενα πρόσβασης δεδομένων (DAO)
- Οι διαχειριστές των αντικειμένων (MANAGER)

7.1.1 Αντικείμενα Μεταφοράς Δεδομένων (DTO)

Τα αντικείμενα μεταφοράς δεδομένων (DTO-Data Transfer Object) περιέχουν τύπους δεδομένων και αντίστοιχες μεθόδους που αποθηκεύουν ή ανακτούν αυτά τα δεδομένα. Χρησιμοποιούνται για τη μεταφορά των δεδομένων ανάμεσα στα υποσυστήματα ή συνθετήματα της εφαρμογής. Για παράδειγμα αρχικοποιούνται από τους διαχειριστές των αντικειμένων και στη συνέχεια μεταφέρουν τις πληροφορίες, από τα αντικείμενα πρόσβασης δεδομένων DAO στην προβολή της εφαρμογής ή αντίστροφα δέχονται πληροφορίες από την προβολή της εφαρμογής και τις μεταφέρουν στα DAO ώστε να αποθηκευτούν στη Βάση Δεδομένων.

Για να είναι πιο εύκολα διαχωρίσιμα τα ιδιαίτερα δεδομένα κάθε άδειας χωρίζονται σε ξεχωριστά αντικείμενα μεταφοράς δεδομένων (DTOs) ενώ τα κοινά στοιχεία για όλες τις άδειες περιέχονται σε ένα κοινό αντικείμενο, το ApprovalDto. Τα DTOs των αδειών μαζί με το ApprovalDto περιέχονται στο HyperApprovalDto που αποτελεί το γενικό αντικείμενο μεταφοράς δεδομένων και χρησιμοποιείται από όλα τα διαφορετικά συνθετήματα για την ανταλλαγή των δεδομένων (Εικόνα 7-1). Κάθε αντικείμενο μεταφοράς δεδομένων περιέχει τις κατάλληλες δομές δεδομένων για κάθε άδεια (για παράδειγμα Long,int,String,ArrayList κ.α.), ολόκληρα στιγμιότυπα άλλων αντικειμένων DTO καθώς επίσης και κατάλληλες μεθόδους(Getters - Setters) για την ανάκτησή τους και την αποθήκευσή τους από και προς τη κλάση. Ένα παράδειγμα αντικειμένου μεταφοράς δεδομένων παρουσιάζεται στην Λίστα 7-1.



Εικόνα 7-1. Δομικό Διάγραμμα Αντικειμένου Μεταφοράς Δεδομένων.

Λίστα 7-1. Παράδειγμα DTO.

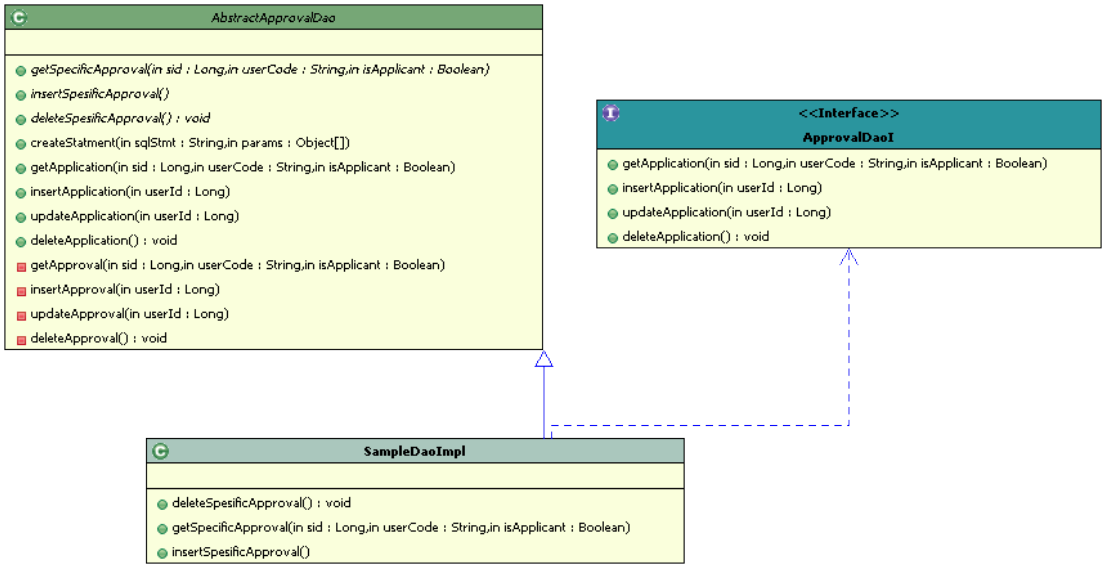
```
package gr.approvals.dto;
import java.util.List;
public class SampleDto {
        private Long sid;
        private String descr;
        ……
        public Long getSid() {return sid;}
        public void setSid(Long sid) {this.sid = sid;}
        public String getDescr() {return descr;}
        public void setDescr(String descr) {this.descr = descr;}
        ……
}
```

7.1.2 Αντικείμενα Πρόσβασης Δεδομένων (DAO)

Τα αντικείμενα Πρόσβασης Δεδομένων (DAO: Data Access Object) είναι κλάσεις που παρέχουν συγκεκριμένες μεθόδους πρόσβασης σε δεδομένα χωρίς να προδίδουν λεπτομέρειες για αυτά. Με τη χρήση τους γίνεται ο διαχωρισμός σχετικά με το τι είδους δεδομένα χρειάζεται η εφαρμογή και ποιος είναι ο τρόπος απόκτησής τους από τη Βάση Δεδομένων. Ο διαχωρισμός αυτός είναι το βασικό πλεονέκτημα χρήσης των DAO. Η αλλαγή της επιχειρησιακής λογικής της εφαρμογής δεν έχει καμία επίδραση στον τρόπο ανάκτησης ή αποθήκευσης δεδομένων και αντίστοιχα η πιθανή μελλοντική αλλαγή του σχήματος της βάσης ή της τεχνολογίας της δεν αφορά καθόλου την λογική του συστήματος.

Κάθε κλάση τύπου DAO υλοποιεί το interface ApprovalDaoI (Λίστα 7-2) και επεκτείνει την κλάση AbstractApprovalDao (Εικόνα 7-2, Λίστα 7-3). Υπάρχουν τέσσερις βασικές μέθοδοι με τις οποίες η εφαρμογή επικοινωνεί με τη βάση διαμέσου των DAOs. Αυτές είναι οι:

- getApplication
- insertApplication
- updateApplication
- deleteApplication



Εικόνα 7-2. Δομικό Διάγραμμα Αντικειμένου Πρόσβασης Δεδομένων.

Σύμφωνα με τη δομή που έχει επιλεχθεί για τα αντικείμενα μεταφοράς δεδομένων DTO οι μέθοδοι που αναφέραμε επιδρούν ξεχωριστά στο ApprovalDto (κοινό για όλα τα άλλα DTOs) και ύστερα στο κατάλληλο DTO ανάλογα με την άδεια που διαχειρίζεται το σύστημα. Δηλαδή κάθε μέθοδος αφού εκτελέσει τις αντίστοιχες διαδικασίες ανάκτησης, εισαγωγής, ενημέρωσης ή διαγραφής του ApprovalDto στη συνέχεια καλεί να εκτελεστεί μια πιο συγκεκριμένη αντίστοιχη μέθοδος (ανάκτησης, εισαγωγής, ενημέρωσης ή διαγραφής) για την αντίστοιχη άδεια. Οι μέθοδοι αυτές είναι οι:

- getSpecificApproval
- insertSpecificApproval
- updateSpecificApproval
- deleteSpecificApproval

Κάθε μια από αυτές τις μεθόδους υλοποιείται σε κάθε DAO για κάθε άδεια και εκτελεί τις αντίστοιχες εργασίες ανάκτησης, εισαγωγής, ενημέρωσης ή διαγραφής για το συγκεκριμένο DTO που αντιστοιχεί στην άδεια. Δηλαδή μέσα στη κλάση AbstractApprovalDao υπάρχει μια μικρή επιχειρησιακή λογική για τον τρόπο διαχείρισης των δεδομένων της βάσης ανάλογα με την άδεια που επεξεργάζεται ο συναλλασσόμενος. Η δομή που ακολουθείται φυσικά δεν είναι τυχαία. Ο προγραμματιστής πρέπει να αναλάβει να αναπτύξει μόνο την υλοποίηση των “Specific” μεθόδων αδιαφορώντας για τον τρόπο ή τη σειρά που θα πρέπει να εκτελεστούν αυτές. Αρκεί να γίνει η υλοποίηση σύμφωνα με τη τεχνολογία που χρησιμοποιεί η Βάση Δεδομένων.

Λίστα 7-2. Interface ApprovalDaoI**.**

```
package gr.approvals.dao.approvals;
import gr.approvals.dto.HyperApprovalDto;
public interface ApprovalDaoI {
        public HyperApprovalDto getApplication(Long sid, String userCode, Boolean
isApplicant);
        public HyperApprovalDto insertApplication(HyperApprovalDto hyperApproval, Long
userId);
        public HyperApprovalDto updateApplication(HyperApprovalDto hyperApproval, Long
userId);
        public void deleteApplication(HyperApprovalDto hyperApproval);
}
```

Λίστα 7-3. Παράδειγμα κλάσης AbstractApprovalDao.

```
import org.springframework.jdbc.core.PreparedStatementCreator;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
public abstract class AbstractApprovalDao extends JdbcDaoSupport {
public abstract HyperApprovalDto getSpecificApproval(Long sid,
        String userCode, Boolean isApplicant);
public abstract HyperApprovalDto insertSpesificApproval(HyperApprovalDto hyperApproval);
public abstract HyperApprovalDto updateSpesificApproval(HyperApprovalDto hyperApproval);
public abstract void deleteSpesificApproval(HyperApprovalDto
                                        hyperApproval);
```

```
public PreparedStatementCreator createStatment(String sqlStmt, Object[] params) {
                PreparedStatementCreator psc = new
        PreparedStatmentCreator(sqlStmt,params );
        return psc;}
public HyperApprovalDto getApplication(Long sid, String
        userCode, Boolean isApplicant) {
                HyperApprovalDto hyperApproval =
getSpecificApproval(sid,userCode,isApplicant);
                if(hyperApproval==null)
                        hyperApproval = new HyperApprovalDto();
if(hyperApproval.getApproval()==null||(!hyperApproval.getApproval().getHolderCode().equals(userC
ode)&&isApplicant))
hyperApproval.setApproval(getApproval(sid,userCode,isApplicant));
                return hyperApproval;}
        public HyperApprovalDto insertApplication(HyperApprovalDto
        hyperApproval, Long userId){
                ApprovalDto approval = hyperApproval.getApproval();
        approval = insertApproval(approval, userId);
                hyperApproval.setApproval(approval);
                hyperApproval = insertSpesificApproval(hyperApproval); return
hyperApproval;}
        public HyperApprovalDto updateApplication(HyperApprovalDto
                                        hyperApproval, Long userId) {
                ApprovalDto approval = hyperApproval.getApproval();
                approval = updateApproval(approval, userId);
                hyperApproval.setApproval(approval);
                hyperApproval = updateSpesificApproval(hyperApproval);
                return hyperApproval;}
        public void deleteApplication(HyperApprovalDto hyperApproval){
                deleteSpesificApproval(hyperApproval);
                deleteApproval(hyperApproval.getApproval());}
```

```
private ApprovalDto getApproval(Long sid,String userCode,
        Boolean isApplicant){
        String sqlquery = " SELECT approval.sid as APPROVAL_SID,
TYPE_DESCR, STATUS_DESCR ";
        sqlquery = sqlquery + " FROM APPROVAL approval,
RF_APPROVAL_TYPE, RF_APPROVAL_STATUS ";
        sqlquery = sqlquery + " WHERE approval.SID = ? ";
        sqlquery = sqlquery + " AND approval.RF_APPROVAL_STATUS_FK =
RF_APPROVAL_STATUS.SID ";
sqlquery = sqlquery + " AND approval.RF_APPROVAL_TYPE_FK =
RF_APPROVAL_TYPE.SID(+) ";
if(isApplicant)
        sqlquery = sqlquery + " AND approval.HOLDER_CODE = ? ";
        Object[] params = null;
if(isApplicant)
        params = new Object[] {sid, userCode};
else
        params = new Object[] {sid};
List<ApprovalDto> approvals = (List<ApprovalDto>) getJdbcTemplate().query(sqlquery,
params, new FullApprovalMapper());
        if(approvals==null || approvals.isEmpty())
                return null;
return approvals.get(0);}
private ApprovalDto insertApproval(ApprovalDto approval, Long userId){
        String sqlInsertApproval = " INSERT INTO APPROVAL";
        sqlInsertApproval = sqlInsertApproval + "(version_no, ";
        sqlInsertApproval = sqlInsertApproval + "is_approval, ";
        sqlInsertApproval = sqlInsertApproval +
"rf_submission_type_fk, ";
        sqlInsertApproval = sqlInsertApproval +
"APPLICATION_CODE, ";
        sqlInsertApproval = sqlInsertApproval + "SUBMISSION_DATE, ";
……………………………………………………………………………………………………………
………………………………
 "RF_APPROVAL_CATEGORY_FK ";
        sqlInsertApproval = sqlInsertApproval + ") ";
        sqlInsertApproval = sqlInsertApproval + " VALUES ( ?, ?, ?, ?,
"+IcisUtils.toDateSql()+",?,?,?,?,?,?,?,?,?,"+IcisUtils.toDateSql()+", "+IcisUtils.toDateSql()+",
"+IcisUtils.toDateSql()+",?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"+IcisUtils.toDateSql()+",
SYSDATE,?,?,?,?,?,?,?)";
Object[] params = new Object[]{   approval.getVersionNo(),
approval.getIsApproval(),
approval.getApprovalSubmisionTypeSID(),
……………………………………………………………………………………………………………
…};
        KeyHolder keyHolder = new GeneratedKeyHolder();
getJdbcTemplate().update(createStatment(sqlInsertApproval, params), keyHolder);
        System.out.println("************************ Just updated approval");

        Long generatedApprovalId=Long.valueOf(keyHolder.getKey().longValue());
        approval.setSid(generatedApprovalId);
        return approval;}
private ApprovalDto updateApproval(ApprovalDto approval, Long userId){
        String sqlUpdateApproval = " UPDATE APPROVAL SET ";
        sqlUpdateApproval = sqlUpdateApproval + "version_no=?,";
        sqlUpdateApproval = sqlUpdateApproval + "is_approval=?,";


……………………………………………………………………………………………………………
……
        sqlUpdateApproval = sqlUpdateApproval+"APPROVAL_DATE=?";
```

```
                sqlUpdateApproval = sqlUpdateApproval+"WHERE SID = ? ";
                Object[] params = new Object[]{     approval.getVersionNo(),
        approval.getIsApproval(),
        ……
……};
getJdbcTemplate().update(createStatment(sqlUpdateApproval, params))
                return approval;}
        private void deleteApproval(ApprovalDto approval){
                Object[] params = new Object[]{approval.getSid()};
        String deleteApprovalSQL="DELETE FROM APPROVAL WHERE SID = ? ";
                params                                          =                                       new
Object[]{approval.getSid()};getJdbcTemplate().update(createStatment(deleteApprovalSQL, params));}
}
```

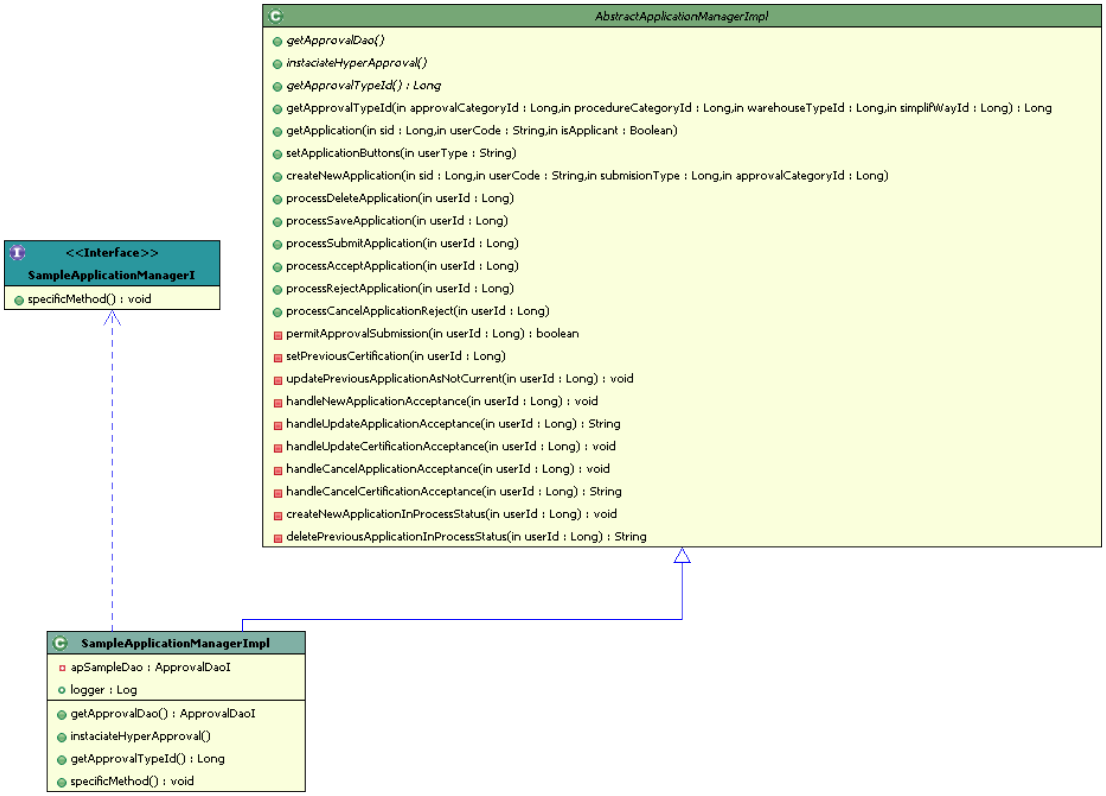
7.1.3 Διαχειριστής (Manager)

Οποιαδήποτε επικοινωνία ανάμεσα στον ελεγκτή (αναλύεται παρακάτω) και στο μοντέλο της εφαρμογής γίνεται μέσω του διαχειριστή (Manager). Ο διαχειριστής παρέχει στον ελεγκτή μέσω των μεθόδων που του προσφέρει πρόσβαση στα αντικείμενα διαχείρισης δεδομένων και από εκεί στη Βάση Δεδομένων του συστήματος. Βέβαια η αξιοποίησή του στην εφαρμογή δε περιορίζεται μόνο στη χρήση του ως γέφυρα ανάμεσα στα αναφερόμενα συνθετήματα της εφαρμογής. Βασικός του ρόλος είναι να μεταφέρει δεδομένα από και προς τις δυο κατευθύνσεις αλλά εφαρμόζοντας πρώτα την επιχειρησιακή λογική «Business Logic» που του δίνει ο προγραμματιστής. Δηλαδή ανάλογα με τα δεδομένα που δέχεται ως είσοδο και το σύνολο των παραμέτρων που καθορίζονται, ίσως από την περεταίρω επικοινωνία του με άλλα υποσυστήματα, παράγει νέα δεδομένα ως έξοδο μεταδίδοντάς τα στα κατάλληλα συνθετήματα-υποσυστήματα. Κατά τον σχεδιασμό της εφαρμογής ο προγραμματιστής προσπαθεί να εμφωλεύσει, όσο είναι δυνατόν, το μεγαλύτερο μέρος της λογικής του συστήματος στον Manager. Κατά την ανάπτυξη μιας επιχειρησιακής εφαρμογής σε κάθε προγραμματιστή ανατίθεται συγκεκριμένη εργασία που αφορά συνήθως την ανάπτυξη συγκεκριμένων συνθετημάτων. Με αυτόν τον τρόπο η λογική του συστήματος, που συνήθως αποτελεί εμπιστευτικό μέρος της εφαρμογής ,δεν εκτίθεται για παράδειγμα σε κάποιον που έχει αναλάβει την ανάπτυξη της προβολής της εφαρμογής.

Όλοι οι διαχειριστές των αδειών υλοποιούνται σε ξεχωριστές κλάσεις. Οι κλάσεις αυτές υλοποιούν μια διεπαφή (interface) στην οποία δηλώνονται οι συγκεκριμένες μέθοδοι που θέλουμε να καλέσουμε για την άδεια για την οποία είναι υπεύθυνος ο διαχειριστής και δεν αντιστοιχούν σε κάποια κοινή διεργασία που εκτελείται για όλες τις άδειες. Αντιθέτως όλες οι μέθοδοι που εκτελούνται για κάθε άδεια ανεξάρτητα της κατηγορίας που ανήκουν είναι δηλωμένες στην κλάση AbstractApplicationManager την οποία και επεκτείνουν (extend) όλοι οι διαχειριστές (Εικόνα 7-3). Η κλάση AbstractApplicationManager αναμένει από τον ξεχωριστό διαχειριστή να υλοποιηθούν τρεις συγκεκριμένες μέθοδοι :

- getApprovalDao
- instaciateHyperApproval
- getApprovalTypeId

Η πρώτη προσδιορίζει το αντικείμενο πρόσβασης δεδομένων DAO που είναι κατάλληλο για τον συγκεκριμένο τύπο άδειας. Η δεύτερη αφορά την αρχικοποίηση του αντικειμένο υ μεταφοράς δεδομένων DTO ανάλογα με την άδεια για την ο ποία είναι υπεύθυνος ο διαχειριστής και η τελευταία επιστρέφει τον τύπο της άδειας που έχει επιλεχθεί ή που προκύπτει ανάλογα με τις επιλογές του χρήστη μέσω της προβολής της άδειας. Ο προγραμματιστής που αναπτύσσει τον κώδικα για τον συγκεκριμένο διαχειριστή οφείλει να συμπληρώσει τις παραπάνω μεθόδους και αν χρειαστεί να δηλώσει επιπλέον μεθόδους που αφορούν τη συγκεκριμένη άδεια μέσα στο interface και φυσικά να τις αναπτύξει στην κλάση που το υλοποιεί.



Εικόνα 7-3. Δομικό Διάγραμμα Διαχειριστή (Manager).

Όλη η επιχειρησιακή λογική που αναπτύχθηκε νωρίτερα υλοποιείται μέσα στην κλάση AbstractApplicationManager που καλείται μέσω του Ελεγκτή ανάλογα με τις επιλογές του χρήστη πάνω στην προβολή της εφαρμογής. Αναλόγως δηλαδή την κατάσταση «status» που βρίσκεται κάθε άδεια, την ενέργεια «action» που επιλέγει να εκτελέσει ο χρήστης και την ιδιότητα «userType» που έχει ο συγκεκριμένος χρήστης, καλείται η κατάλληλη μέθοδος και το αποτέλεσμα εμφανίζεται τελικά στον χρήστη μέσω της προβολής.Οι ενέργειες του χρήστη επί το πλείστον περιορίζονται στη συμπλήρωση των κατάλληλων πεδίων μιας φόρμας που

του δίνεται ως διεπαφή με το σύστημα και την επιλογή κατάλληλων «πλήκτρων οθόνης» για την εκτέλεση συγκεκριμένων διαδικασιών. Όπως για παράδειγμα, μετά τη συμπλήρωση των πεδίων μιας αίτησης ο χρήστης μπορεί να αποθηκεύσει τις αλλαγές που έκανε, να καταχωρήσει την αίτησή του, να την ακυρώσει κ.ά. Οι ενέργειες στις οποίες μπορεί να προβεί ο χρήστης εξαρτώνται κάθε φορά από τη κατάσταση στην οποία βρίσκεται η αίτηση-άδεια. Ο διαχειριστής μέσω του αντικείμενου πρόσβασης δεδομένων ButtonsDao (Λίστα 7-4) μπορεί και εμφανίζει ή αφαιρεί από την προβολή της εφαρμογής τα κατάλληλα «πλήκτρα οθόνης» ώστε να καθοδηγήσει τον χρήστη στη σωστή σειρά ενεργειών ή αποτρέπει τον χρήστη από μη επιτρεπτές ενέργειες.

Λίστα 7-4. ButtonsDaoImpl.

```
import org.springframework.jdbc.core.support.JdbcDaoSupport;
public class ButtonsDaoImpl extends JdbcDaoSupport implements ButtonsDaoI{
        public ButtonsDto setApplicationButtons(HyperApprovalDto
                hyperApproval, String userType) {
                if(hyperApproval==null ||
        hyperApproval.getApproval()==null)
                        return null;
                ButtonsDto buttons = new ButtonsDto();
                String retrieveButtonsSql = " Select SID as code,
TO_RF_APPROVAL_STATUS_FK as name from RF_APPROVAL_ACTION ";
                retrieveButtonsSql = retrieveButtonsSql + " where  BY_APPLICANT = ? AND";

                retrieveButtonsSql = retrieveButtonsSql + " FROM_RF_APPROVAL_STATUS_FK
= ? ";
                Object[] params = new Object[] {userType,
hyperApproval.getApproval().getApprovalStatusSID()};
                List<NameCodeDto> actions = (List<NameCodeDto>)
getJdbcTemplate().query(retrieveButtonsSql, params, new NameCodeMapper());
                for(int i=0; i<actions.size();i++){
                switch(new Integer(actions.get(i).getCode()).intValue()){
case 1: buttons.setRenderSaveApplicationButton(true);
                break;
case 2: buttons.setRenderSubmitApplicationButton(true);
                break;
case 3: buttons.setRenderAcceptApplicationButton(true);
                break;
case 4: buttons.setRenderRejectApplicationButton(true);
                break;
case 5: buttons.setRenderSaveCertificateButton(true);
                break;
case 6: buttons.setRenderRejectCertificateButton(true);
                break;
case 7: buttons.setRenderIssueCertificateButton(true);
                break;
case 10:buttons.setRenderSuspendCertificateButton(true);
                break;
case 12: buttons.setRenderCancelCertificateButton(true);
                break;
case 13: buttons.setRenderCancelCertificationRecallButton(true);
                break;
case 14: buttons.setRenderCancelCertificationSuspendButton(true);
                break;
case 15: buttons.setRenderCancelCertificationCancelButton(true);                        break;
```

```
case 16: buttons.setRenderCancelApplicationRejectButton(true);                    break;
case 18: buttons.setRenderUpdateCertificateButton(true);
                    break;
case 19: buttons.setRenderRecallCertificateButton(true);
                    break;
case 20: buttons.setRenderDeleteApplicationButton(true);
                    break}
          }return buttons;}
}
```

Συγκεκριμένα, μέσω του καθορισμένου DAO ανακτάται από τη βάση η κατάσταση «status» της άδειας και αναλόγως τη τιμή που λαμβάνει, ενεργοποιεί ή απενεργοποιεί ξεχωριστά κάθε «πλήκτρο» της εφαρμογής. Ανάλογα με την επιλογή του χρήστη εκτελείται η κατάλληλη μέθοδος ή μια σειρά από μεθόδους, οι οποίες συνοψίζονται στις ακόλουθες:

- createNewApplication:

Εκτελείται κατά την επιλογή δημιουργίας μιας νέας αίτησης ή τροποποίησης μιας αίτησης εφόσον το επιτρέπει η κατάσταση της αίτησης.

- processDeleteApplication:

Εκτελείται κατά την επιλογή διαγραφής μιας αίτηση-άδειας, η οποία περνά σε κατάσταση «διαγραμμένη».

- processSaveApplication:

Εκτελείται κατά την επιλογή της Προσωρινής Αποθήκευσης. Γίνεται αποθήκευση της άδειας στη Βάση Δεδομένων καλώντας τις ανάλογες μεθόδους από το κατάλληλο DAO.

- processSubmitApplication:

Εκτελείται κατά την επιλογή Υποβολής της άδειας εφόσον αυτό επιτρέπεται με την ενημέρωση της Βάσης Δεδομένων και άλλων παραμέτρων του μοντέλου.

- processAcceptApplication:

Εκτέλεση κατάλληλων διαδικασιών κατά την αποδοχή μιας αίτησης ανάλογα με τη κατάσταση που βρίσκεται. Για παράδειγμα η εκτέλεση της μεθόδου handleNewApplicationAcceptance όταν πρόκειται για νέα αίτηση.

- processRejectApplication:

Εκτελείται κατά την επιλογή Μη αποδοχής της αίτησης και θέτει την άδεια στην αντίστοιχη κατάσταση.

- processCancelApplicationReject

Εκτελείται κατά την επιλογή ανάκλησης της διαδικασίας απόρριψης αλλάζοντας τη κατάσταση της άδειας.

- permitApprovalSubmission

Επιτρέπεται ή όχι η υποβολή μετά από την εκτέλεση συγκεκριμένων ελέγχων. Για παράδειγμα γίνεται απόρριψη της υποβολής νέας αίτησης αν ήδη υπάρχει εγκεκριμένη άδεια

του ίδιου τύπου με αυτόν της νέας αίτησης.

- setPreviousCertification

Εκτελείται για την διατήρηση του ιστορικού και την έκδοση νέου αριθμού πρωτοκόλλου όταν χρειάζεται.

- updatePreviousApplicationAsNotCurrent

Δήλωση του στιγμιότυπου της άδειας ως τρέχων, αλλάζοντας τις αντίστοιχες τιμές στους πίνακες της βάσης δεδομένων.

- handleNewApplicationAcceptance

Εκτέλεση της επιχειρησιακής λογικής που αναπτύχθηκε προηγουμένως σε περίπτωση νέας αίτησης. Για παράδειγμα, μια άδεια που βρίσκεται σε αναστολή περνά σε κατάσταση «κλεισμένη».

- handleUpdateApplicationAcceptance

Εκτελούνται με τη σειρά οι μέθοδοι deletePreviousApplicationInProcessStatus και createNewApplicationInProcessStatus.

- handleUpdateCertificationAcceptance

Εκτελείται η μέθοδος createNewApplicationInProcessStatus.

- handleCancelApplicationAcceptance

Εκτελείται η μέθοδος deletePreviousApplicationInProcessStatus.

- handleCancelCertificationAcceptance

Ανακτάται ο αριθμός της άδειας που είναι προς ακύρωση. Αναζητείται η άδεια που βρίσκεται σε κατάσταση «εκδοθείσα» ή «σε αναμονή» ή «σε αναστολή» και ενημερώνεται η σημαία που αφορά την τρέχουσα έκδοση της άδειας. Τέλος δημιουργείται μια νέα άδεια σε κατάσταση «ακυρωθείσα».

- createNewApplicationInProcessStatus

Δημιουργείται μια νέα εγγραφή σε κατάσταση «σε επεξεργασία», αλλάζει ο τύπος από «αίτηση» σε «άδεια» και αυξάνει ο αριθμός έκδοσης της αίτησης.

- deletePreviousApplicationInProcessStatus

Εκτελείται κατά την επιλογή Τροποποίησης μιας αίτησης-άδειας ώστε να ακυρωθούν προηγούμενες αιτήσεις που δεν είναι ακόμα σε κατάσταση επεξεργασίας.

7.2 Υλοποίηση του στρώματος της Προβολής (View Layer)

Η όψη-προβολή (View) παράγει μια παρουσίαση των δεδομένων στον χρήστη της εφαρμογής. Αναλόγως τη τεχνολογία που έχει επιλεγεί για την ανάπτυξη της διεπαφής χρήστη και εφαρμογής, η προβολή μπορεί να αποτελείται από αρχεία τύπου JSP, HTML, XHTML κ.ά. Στα αρχεία αυτά περιγράφεται ο τρόπος απεικόνισης των δεδομένων του

μοντέλου και επιπλέον δίνεται η δυνατότητα στον χρήστη να διαχειριστεί τα δεδομένα αυτά και να αλληλεπιδράσει με την εφαρμογή, μέσα στα στενά πάντα περιθώρια που η εφαρμογή καθορίζει, αναλόγως με τον τρόπο που έχει αναπτυχθεί από τον προγραμματιστή. Για κάθε τύπο άδειας αναπτύσσεται διαφορετική προβολή που αλληλεπιδρά με το αντίστοιχο μοντέλο της άδειας μέσω των αντικειμένων μεταφοράς δεδομένων (DTOs). Μεταξύ των διαφορετικών προβολών υπάρχουν αρκετά κοινά σημεία όσον αφορά τη παρουσίαση και διαχείριση κοινών δεδομένων για όλες τις άδειες. Για να αποφύγουμε την άσκοπη συγγραφή όμοιου κώδικα για κάθε ξεχωριστή προβολή, μπορούμε να δημιουργήσουμε ένα πρότυπο προβολής που θα ακολουθεί κάθε προβολή άδειας (Λίστα 7-5). Με αυτόν τον τρόπο κερδίζουμε, εκτός από χρόνο για συγγραφή κώδικα, ομοιομορφία στο τρόπο παρουσίασης των δεδομένων αλλά και καθοδήγηση για τον τρόπο ανάπτυξης κάθε προβολής.

Λίστα 7-5. Παράδειγμα προτύπου άδειας.

```
<?xml version="1.0" encoding="UTF-8"?>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ include file="/common/taglibs.jsp"%>
<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<title>Αίτηση για χορήγηση άδειας  </title>
</head>
<body>
<form:form modelAttribute="hyperApproval"  id="hyperApproval" name="hyperApproval"  >
<jsp:include page="/common/app_header.jsp" />
<table class="outer">
 <tr> <td>
        <table class="inner">
        <tr><td>
                        <p>Αίτηση:<form:input path="apGeneral.descr"  id="descr"
maxlength="20" /></p>
                </td></tr>
……
                CustomApprovalDetails
……
                </td></tr>
        </table>
 </td></tr>
</table>
        <jsp:include page="/common/app_basics.jsp" />
        <jsp:include page="/common/holder.jsp" />
        <jsp:include page="/common/comments.jsp" />
        <jsp:include page="/common/sign.jsp" />
        <jsp:include page="/common/app_footer.jsp" />
</form:form>
</body>
</html>
```

7.3 Υλοποίηση του στρώματος του Ελεγκτή (Controller Layer)

Όπως προαναφέρθηκε, ο ελεγκτής αποτελεί τον συνδετικό κρίκο των παραπάνω στρωμάτων της προβολής και του μοντέλου. Συνεπώς για κάθε κατηγορία άδειας αναπτύσσεται και ο αντίστοιχος ελεγκτής που πραγματοποιεί τη διασύνδεση με το μοντέλο και την αντίστοιχη προβολή της άδειας. Κάθε ελεγκτής υλοποιείται από μια κλάση, της οποίας το όνομα αντιπροσωπεύει την αίτηση ή την άδεια την οποία είναι αρμόδιος να διαχειριστεί. Αυτή η κλάση κληρονομεί και επεκτείνει μια ακόμα κλάση (Abstract class) στην οποία καθορίζονται όλες οι βασικές λειτουργίες που θα πρέπει να εκτελούνται από τον εκάστοτε ελεγκτή της άδειας (Εικόνα 7-4). Ανάλογα με το αν πρόκειται για αίτηση ή άδεια, κληρονομείται η κλάση AbstractApplicationController (Λίστα 7-6) ή AbstractCertificateController. Στην αρχή της κλάσης του ελεγκτή γίνεται η ζεύξη του με την αντίστοιχη προβολή, κάνοντας απλά χρήση μιας γνωστοποίησης (annotation) του Spring Framework.

Μέσα στην Abstract κλάση του Ελεγκτή είναι υλοποιημένες δύο βασικές μέθοδοι που εκτελούνται πάντα κατά τη διάρκεια επικοινωνίας του τελικού χρήστη με το σύστημα. Κατά την επικοινωνία του χρήστη μέσω του προγράμματος περιήγησης ιστοσελίδων (Browser), με την εφαρμογή εκτελούνται οι μέθοδοι Get και Post του περιηγητή. Η απάντηση σε αυτές τις κλήσεις δίνεται από τις βασικές μεθόδους και υλοποιούνται σε κάθε Ελεγκτή. Αυτές είναι οι ακόλουθες:

- setupForm
- processRequest

Η setupForm εκτελείται κατά την κλήση του περιηγητή μέσω της μεθόδου GET. Η εκτέλεση της μεθόδου συμβαίνει όταν ο τελικός χρήστης ζητά διαμέσου της προβολής να δημιουργήσει μια νέα αίτηση ή να αναζητήσει μια αίτηση που έχει ήδη καταχωρήσει. Αναλόγως λοιπόν από την επιλογή του χρήστη εμφωλεύονται μέσα στη συνεδρία «session» του περιηγητή κάποιοι παράμετροι που δίνουν πληροφορίες στον ελεγκτή για το ποιος χρήστης είναι και ποιο είναι το αίτημά του. Αφού ελεγχθούν οι παράμετροι, τότε καλείται και η αντίστοιχη μέθοδος του Διαχειριστή που αφορά τη συγκεκριμένη κατηγορία της αίτησης και αφού γεμίζει το μοντέλο με δεδομένα τα επιστρέφει μέσω της προβολής στον τελικό χρήστη.

Η processRequest εκτελείται κατά την κλήση του περιηγητή μέσω της μεθόδου Post. Η εκτέλεση της μεθόδου συμβαίνει όταν ο χρήστης έχει ήδη ανακτήσει μια άδεια και έχει προβεί σε μια ενέργεια όπως αυτή της «προσωρινής αποθήκευσης» μέσω των «πλήκτρων οθόνης». Σε κάθε περίπτωση, ανάλογα με την ενέργεια του χρήστη, που αναγνωρίζεται ξανά μέσω των παραμέτρων που περνιούνται από τη συνεδρία, εκτελείται η κατάλληλη μέθοδος από τον Διαχειριστή της άδειας.

Σύμφωνα με το σχήμα υλοποίησης ενός ελεγκτή μιας αίτησης ή μιας άδειας ο τρόπος υλοποίησης της αντίστοιχης κλάσης είναι και πάλι καθορισμένος για τον προγραμματιστή. Θα πρέπει δηλαδή να αναπτυχθούν συγκεκριμένες μέθοδοι που τελικά καθορίζουν την συμπεριφορά του ελεγκτή και τις συγκεκριμένες διαδικασίες που πρέπει να εκτελεστούν από αυτόν. Πιο συγκεκριμένα, η κλάση AbstractApplicationController απαιτεί την υλοποίηση των μεθόδων:

- getInitialBuldingForm

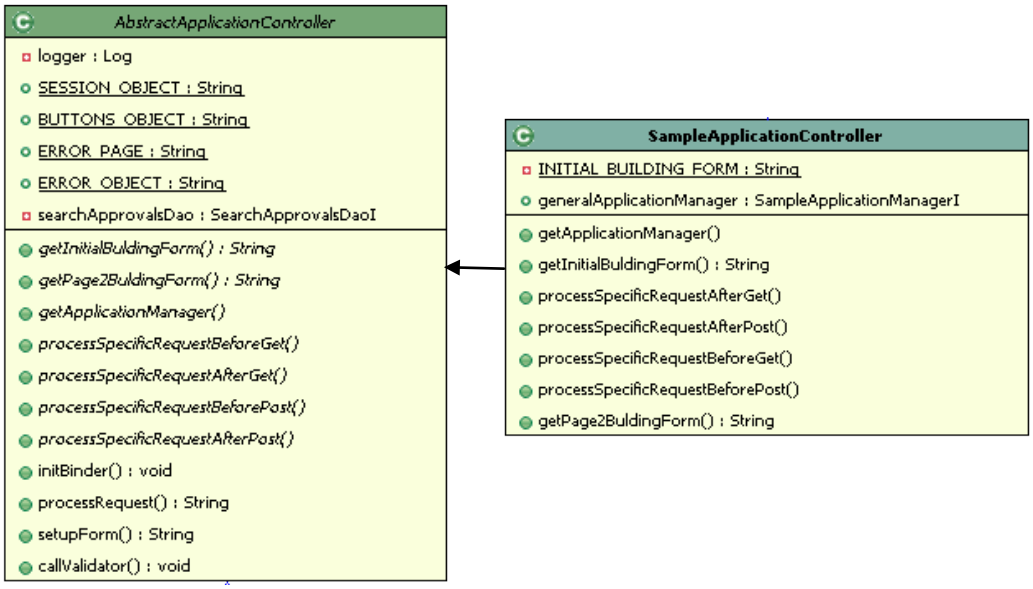
Σε αυτή τη μέθοδο επιστρέφεται το όνομα της προβολής που θα πρέπει να αναζητηθεί από την εφαρμογή και να παρουσιαστεί στον τελικό χρήστη.

- getApplicationManager

Σε αυτή τη μέθοδο δηλώνεται ο Διαχειριστής που είναι αρμόδιος για το συγκεκριμένο αίτημα του χρήστη.

- processSpecificRequestBeforeGet
- processSpecificRequestAfterGet
- processSpecificRequestBeforePost
- processSpecificRequestAfterPost

Οι υπόλοιπες τέσσερις μέθοδοι καλούνται να εκτελεστούν μέσα από τις setupForm και processRequest. Μέσα σε αυτές τις μεθόδους ο προγραμματιστής καθορίζει τις ιδιαίτερες λειτουργίες που πρέπει να εκτελεστούν από τον ελεγκτή για την εκάστοτε αίτηση-άδεια πριν να εμφανιστούν τα αποτελέσματα μέσω της προβολής.



Εικόνα 7-4. Δομικό Διάγραμμα Ελεγκτή (Controller).

Λίστα 7-6. AbstractApplicationController.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.propertyeditors.CustomDateEditor;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.ObjectError;
import org.springframework.validation.ValidationUtils;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.support.SessionStatus;
import org.springframework.web.util.WebUtils;
public abstract class AbstractApplicationController {
	private final Log logger = LogFactory.getLog(AbstractApplicationController.class);
	public static final String SESSION_OBJECT = "hyperApproval";
	public static final String BUTTONS_OBJECT = "buttons";
	public static final String ERROR_PAGE = "errorPage";
	public static final String ERROR_OBJECT = "errors";
	public abstract String getInitialBuldingForm();
	public abstract String getPage2BuldingForm(HyperApprovalDto hyperApproval);
	public abstract ApplicationManagerI getApplicationManager();
	public abstract HyperApprovalDto processSpecificRequestBeforeGet(HyperApprovalDto
hyperApproval, BindingResult result, SessionStatus status,HttpServletRequest request,Model model);
	public abstract HyperApprovalDto processSpecificRequestAfterGet(HyperApprovalDto
hyperApproval, BindingResult result, SessionStatus status,HttpServletRequest request,Model model);
	public abstract HyperApprovalDto processSpecificRequestBeforePost(HyperApprovalDto
hyperApproval, BindingResult result, SessionStatus status,HttpServletRequest request,Model model);
	public abstract HyperApprovalDto processSpecificRequestAfterPost(HyperApprovalDto
hyperApproval, BindingResult result, SessionStatus status,HttpServletRequest request,Model model);
	@RequestMapping(method = RequestMethod.POST)
	public String processRequest(@ModelAttribute(SESSION_OBJECT) HyperApprovalDto
hyperApproval, BindingResult result, SessionStatus status,HttpServletRequest
request,HttpServletResponse response, Model model) {
		HyperApprovalDto temp;
		temp = processSpecificRequestBeforePost(hyperApproval, result, status,request,
model);
		if(temp!=null)
			hyperApproval = temp;
		UserSession user = (UserSession) WebUtils.getSessionAttribute(request,
"userSession");
		Long userType = user.getAccount().getUserType();
		Long userId = user.getAccount().getUserId();
		String applicant = null;
		if(request.getParameter("_back")!=null) {
			try {String url = request.getContextPath() +
"/searchApplications.icis?searchApplications=1";
				response.sendRedirect(url);
			} catch(Exception e){}}
		if(userType.equals(Constants.USER_TYPE_APPLICANT))
			applicant = Constants.USER_TYPE_APPLICANT.toString();
		else applicant = Constants.USER_TYPE_OFFICER.toString();
		callValidator(hyperApproval, result, request);
	if (result.hasErrors()) {
	    logger.debug("There are errors.");
	    return getInitialBuldingForm();}
	if(request.getParameter("_deleteApplication")!=null)  {
```

```
                hyperApproval = getApplicationManager().processDeleteApplication(hyperApproval,userId, result);
                String url = request.getContextPath() + "/searchApplications.icis?searchApplications=1";
                try {response.sendRedirect(url);
                }catch(Exception e){;}}
            if(request.getParameter("_saveApplication")!=null)
                try {hyperApproval = getApplicationManager().processSaveApplication(hyperApproval,userId, result);
                } catch (Exception e) {e.printStackTrace();}
            else if(request.getParameter("_submitApplication")!=null)
                hyperApproval = getApplicationManager().processSubmitApplication(hyperApproval,userId, result);
            else if(request.getParameter("_acceptApplication")!=null)
                hyperApproval = getApplicationManager().processAcceptApplication(hyperApproval,userId, result);
            else if(request.getParameter("_rejectApplication")!=null)
                hyperApproval = getApplicationManager().processRejectApplication(hyperApproval,userId, result);
            else if(request.getParameter("_cancelApplicationReject")!=null)
                hyperApproval = getApplicationManager().processCancelApplicationReject(hyperApproval, userId, result);
            temp = processSpecificRequestAfterPost(hyperApproval, result, status,request, model);
            if(temp!=null)
                hyperApproval = temp;
            ButtonsDto buttons = getApplicationManager().setApplicationButtons(hyperApproval,applicant);
            model.addAttribute(SESSION_OBJECT, hyperApproval);
            model.addAttribute(BUTTONS_OBJECT, buttons);
            if(request.getParameter("_page2")!=null)
                return getPage2BuldingForm(hyperApproval);
            return getInitialBuldingForm();}
        @RequestMapping(method = RequestMethod.GET)
        public String setupForm(Model model, HttpServletRequest request) {
            HyperApprovalDto hyperApproval = null;
            Boolean isApplicant = null;
            String applicant = null;
            UserSession user = (UserSession) WebUtils.getSessionAttribute(request, "userSession");
            Long userId = user.getAccount().getUserId();
            String userCode = user.getAccount().getUserCode();
            Long userType = user.getAccount().getUserType();
            if(userType.equals(Constants.USER_TYPE_APPLICANT)){
                isApplicant = true;
                applicant = Constants.USER_TYPE_APPLICANT.toString();}
            else {isApplicant = false;
                applicant = Constants.USER_TYPE_OFFICER.toString();}
                hyperApproval = processSpecificRequestBeforeGet(hyperApproval, null, null,request, model);
            Long id = null;
            String stringId = (String)request.getParameter("id");
            if(stringId!=null)
                id = new Long(stringId);
            Long approvalCategoryId = null;
            String stringApprovalCategoryId = request.getParameter("approvalCategoryId");
            if(stringApprovalCategoryId!=null)
                approvalCategoryId = new Long(stringApprovalCategoryId);
            Long submisionTypeId = null;
            String stringSubmisionTypeId = request.getParameter("submissionType");
```

```
                if(stringSubmisionTypeId!=null)
                        submisionTypeId = new Long(stringSubmisionTypeId);
                if(submisionTypeId!=null)
                        hyperApproval = getApplicationManager().createNewApplication(id,
userCode, submisionTypeId, approvalCategoryId);
                else hyperApproval = getApplicationManager().getApplication(id, userCode,
isApplicant);
                        if(hyperApproval==null) {
                        ObjectError err = new ObjectError("errors.submissionDenied",
"submissionDenied");
                        model.addAttribute(ERROR_OBJECT, err);
                        return ERROR_PAGE;}
                if(hyperApproval.getApproval()==null) {
                        ObjectError err = new ObjectError("errors.ApplicationNotFound",
"ApplicationNotFound or AccessDenied");
                        model.addAttribute(ERROR_OBJECT, err);
                        return ERROR_PAGE;}
                else
if(hyperApproval.getApproval().getApprovalCategorySID().longValue()!=approvalCategoryId.longVal
ue()){
                        ObjectError err = new ObjectError("errors.ApplicationNotValidJsp",
"Application Not Valid Jsp");
                        model.addAttribute(ERROR_OBJECT, err);
                        return ERROR_PAGE;}
                else
if(hyperApproval.getApproval().getIsCurrent().equals(Constants.NOT_CURRENT_FLG)){
                        ObjectError err = new ObjectError("errors.ApplicationOldVersion",
"Application Old Version");
                        model.addAttribute(ERROR_OBJECT, err);
                        return ERROR_PAGE;}
                HyperApprovalDto temp = processSpecificRequestAfterGet(hyperApproval, null,
null,request, model);
                if(temp!=null)
                        hyperApproval = temp;
                ButtonsDto buttons =
getApplicationManager().setApplicationButtons(hyperApproval, applicant);
                model.addAttribute(SESSION_OBJECT, hyperApproval);
                model.addAttribute(BUTTONS_OBJECT, buttons);
                return getInitialBuldingForm();}
}
```

7.4 Έλεγχος Γνησιότητας Χρήστη (Authentication)

Όπως αναφέρθηκε στην ανάλυση του συστήματος αδειών - εγκρίσεων υπάρχουν δύο βασικές κατηγορίες χρηστών: οι πιστοποιημένοι/εξωτερικοί χρήστες και οι εξουσιοδοτημένοι χρήστες-υπάλληλοι/εσωτερικοί χρήστες.

7.4.1 Εφαρμογή διαχείρισης εσωτερικών χρηστών

Η εφαρμογή διαχείρισης εσωτερικών χρηστών είναι υπεύθυνη για την διαχείριση των λογαριασμών των εσωτερικών χρηστών, των ομάδων/οργανισμών στις οποίες ανήκουν, των ρόλων που έχει ο κάθε χρήστης ή ομάδα και των αντιστοίχων δικαιωμάτων τους.

Η εφαρμογή έχει σχεδιαστεί με την τεχνολογία των EJBs και παρέχει μια βιβλιοθήκη (jar) που περιλαμβάνει τις ανάλογες απομακρυσμένες διεπαφές (remote interfaces) στο σύστημα

των αδειών. Ενδεικτικά κάποιες από τις μεθόδους που μπορεί να καλέσει το σύστημα των αδειών είναι:

- loadInternalUser - Ανακτά πληροφορίες για ένα πιστοποιημένο χρήστη.
- retieveUserPermissions - Ανακτά τα δικαιώματα ενός πιστοποιημένου χρήστη.
- verifyAuthorization - Προσδιορίζει αν ο πιστοποιημένος χρήστης είναι εξουσιοδοτημένος να εκτελέσει μια συγκεκριμένη ενέργεια.

Οι μέθοδοι των απομακρυσμένων διεπαφών καλούν τα αντίστοιχα stateless EJB της εφαρμογής διαχείρισης των χρηστών που με τη σειρά τους εκτελούν την απαραίτητη επιχειρησιακή λογική για να επιστρέψουν το επιθυμητό αποτέλεσμα στο σύστημα των αδειών.

7.4.2 Εφαρμογή διαχείρισης εξωτερικών χρηστών

Η εφαρμογή διαχείρισης εξωτερικών χρηστών ή οποία είναι υπεύθυνη για την επικύρωση, ταυτοποίηση και την καταγραφή των ενεργειών των χρηστών που συνδέονται στο σύστημα των αδειών μέσω του Internet

Η εφαρμογή έχει σχεδιαστεί με τεχνολογία EJB με τον ίδιο τρόπο που αναφέραμε για τους εσωτερικούς χρήστες και έχει τις μεθόδους που χρειάζονται για τη διαχείριση των εξωτερικών χρηστών. Ενδεικτικά:

- retrieveExternalUserForPrincipal - Ανακτά πληροφορίες για τον εξωτερικό χρήστη
- isUserBlacklisted – Ελέγχει αν για τον συγκεκριμένο χρήστη υπάρχει κάποιου είδους απαγόρευση
- synchronousAuditLog – καταγράφει τις ενέργειες του χρήστη και τις αποθηκεύει σε μια βάση δεδομένων

Στο σύστημα των αδειών η εκτέλεση μεθόδων για την ταυτοποίηση των χρηστών γίνεται γίνεται σε μια κλάση που υλοποιεί τη διεπαφή javax.servlet.Filter και τρέχει πριν από κάθε αίτημα. Το ίδιο μπορούμε να επιτύχουμε υλοποιώντας μια κλάση τύπου org.springframework.web.servlet.handler.HandlerInterceptorAdapter του Spring.

Τα παραπάνω τρία συστήματα: των αδειών και των δύο εφαρμογών διαχείρισης χρηστών είναι ανεξάρτητα μεταξύ τους με αποτέλεσμα να επιτυγχάνεται η ασφάλεια και η επεκτασιμότητα του όλου έργου.

7.5 Βάση Δεδομένων

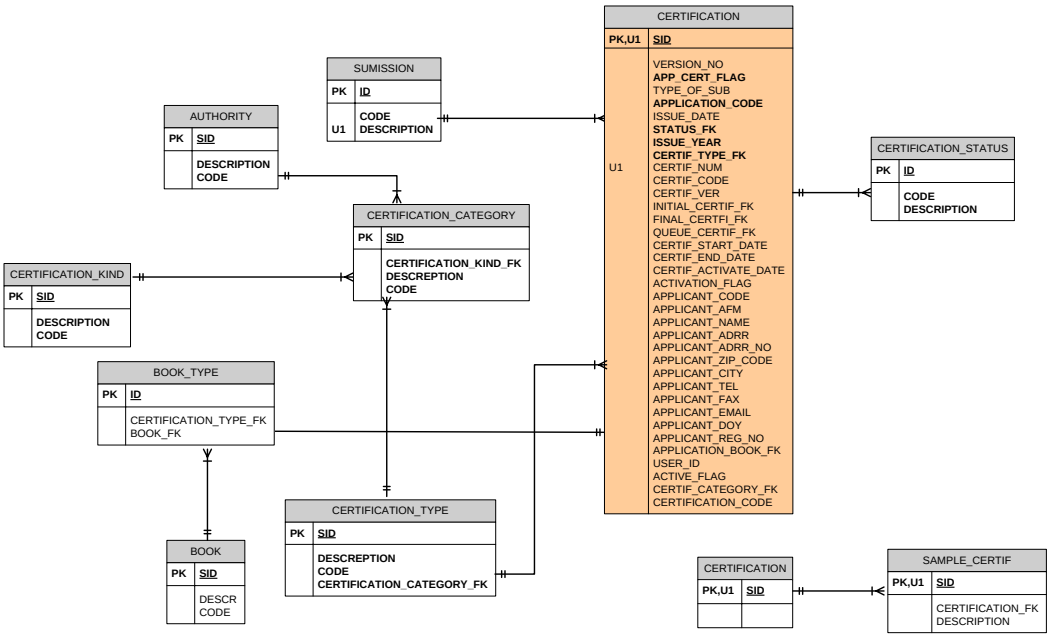
Μια σχεσιακή Βάση Δεδομένων που θα μπορούσε να υποστηρίζει το σύστημα αιτήσεων – εγκρίσεων θα πρέπει να έχει κάποιους βασικούς πίνακες που θα διατηρούν τα δεδομένα του συστήματος. Ο βασικός πίνακας είναι ο CERTIFICATION με κύριο κλειδί το SID. Όλα τα

βασικά στοιχεία μιας αίτησης ή άδειας που αντιστοιχούν στο αντικείμενο ApprovalDTO αποθηκεύονται σε αυτόν τον πίνακα. Παρατηρούμε ότι τα δεδομένα που διαφοροποιούν την αίτηση και την έγκριση είναι πολύ λίγα, ενώ τα βασικά στοιχεία είναι κοινά και για τις δυο περιπτώσεις, για αυτό το λόγο αποφεύγουμε τη χρήση διαφορετικού πίνακα για την αποθήκευση των εγκρίσεων. Συνεπώς αρκεί ένας κοινός πίνακας στη βάση δεδομένων με μια σημαία αναφοράς για το αν η εκάστοτε εγγραφή του πίνακα αναφέρεται σε αίτηση ή σε άδεια – πιστοποιητικό. Επιπλέον δημιουργούνται κάποιοι πίνακες αναφοράς ως προς τα στοιχειώδη δεδομένα της κάθε αίτησης - άδειας και κάθε εγγραφή του CERTIFICATION περιέχει ένα ξένο κλειδί προς αυτούς. Αυτοί οι πίνακες αναφέρονται παρακάτω:

- CERTIFICATION_STATUS ο οποίος περιέχει τις καταστάσεις (status) στις οποίες μπορεί να έχει μια αίτηση ή μια άδεια.
- CERTIFICATION_CATEGORY ο οποίος περιέχει τις κατηγορίες των αιτήσεων – αδειών που είναι διαθέσιμες για το σύστημα.
- CERTIFICATION_TYPE ο οποίος περιέχει τους τύπους αδειών που είναι διαθέσιμοι ανά κατηγορία.
- BOOK ο οποίος αναφέρεται σε κάποιο βιβλίο αρχειοθέτησης των αιτήσεων – αδειών που συντηρεί κάθε αρμόδια αρχή.

Για κάθε κατηγορία άδειας – έγκρισης δημιουργείται ένας επιπλέον πίνακας (για παράδειγμα ο SAMPLE_CERTIF) που θα αποθηκεύονται τα ιδιαίτερα στοιχεία της αίτησης - άδειας και θα έχει πάντα ένα ξένο κλειδί πάνω στον βασικό πίνακα CERTIFICATION.

Παρακάτω παρουσιάζεται το διάγραμμα σχεσιακού σχήματος μιας Βάσης Δεδομένων με τους βασικούς πίνακες Εικόνα 7-5.



Εικόνα 7-5. Διάγραμμα Σχεσιακού Σχήματος.

8

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Αυτό που ίσως κάνει τις δικτυακές υπηρεσίες μοναδικές είναι το γεγονός ότι κόσμος της πληροφορίας ελπίζει ότι μέσω αυτών θα καταφέρουν οι επιχειρήσεις και οι δημόσιες υπηρεσίες να οργανώσουν τις προσφερόμενες υπηρεσίες τους με τέτοιο τρόπο ώστε η αναζήτηση, η εύρεση και η χρήση αυτών από τους πελάτες τους να γίνεται εύκολα και γρήγορα. Παρόλο που τα τελευταία χρόνια γίνεται μια προσπάθεια, από μέρους του Δημόσιου φορέα, ανάπτυξης ηλεκτρονικών υπηρεσιών μέσω διαδικτύου, παρατηρείται έλλειψη εφαρμογής της υπάρχουσας τεχνολογίας σε δημόσιες υπηρεσίες που θα μπορούσαν να προσφέρουν πραγματικές λύσεις στα προβλήματα των πολιτών. Η αφύπνιση του κρατικού φορέα αποτελεί βασικό ζήτημα για την εκμετάλλευση και εφαρμογή των τεχνολογικών εξελίξεων στις κρατικές υπηρεσίες. Ωστόσο κάτι τέτοιο πρέπει να γίνει ύστερα από σωστό σχεδιασμό και μελέτη της κάθε περίπτωσης ξεχωριστά. Στη παρούσα εργασία έγινε μια προσπάθεια για τη μοντελοποίηση ενός συστήματος που βρίσκει εφαρμογή σε πολλούς διαφορετικούς δημόσιους φορείς. Θα πρέπει να γίνει μια συντονισμένη προσπάθεια για την τοποθέτηση των υπηρεσιών κάτω από συγκεκριμένα πλαίσια με σκοπό τη σωστή ανάπτυξη, την ομοιομορφία και την συνεργασία των δημόσιων υπηρεσιών.

9

Βιβλιογραφία

1. Expert Spring MVC and Web Flow, Seth Ladd with Darren Davison, Steven Devijver and Colin Yates, 2006.

2. Spring in Action, Craig Walls and Ryan Breidenbach, 2005.

3. The Spring Framework - Reference Documentation, Rod Johnson, 2008.

4. Expert One on One J2EE Design and Development, Rod Johnson, 2002.

5. Ηλεκτρονική Διακυβέρνηση: Υπηρεσίες και Εφαρμογές, Γ. Χαραλαμπίδης, Ιούνιος 2006.

6. Μέτρηση Αποτελεσματικότητας Υπηρεσιών Ηλεκτρονικής Διακυβέρνησης, Γιώργος-Μάριος Γιανναντωνάκης, Διπλωματικη Εργασία, 2008.

7. Ανάπτυξη εφαρμογής διαχείρισης μαθημάτων με χρήση τεχνολογιών Java 2 Enterprise Edition, Νικόλαος Κουρής, 2008.

8. Building a Local Administration Services Portal for Citizens and Businesses: Service Composition, Architecture and Back-Office Interoperability Issues, Sotirios Koussouris, Yannis Charalabidis, George Gionis, Tasos Tsitsanis and John Psarras, *e*GOV 2007 Conference, Regensburg, Germany.